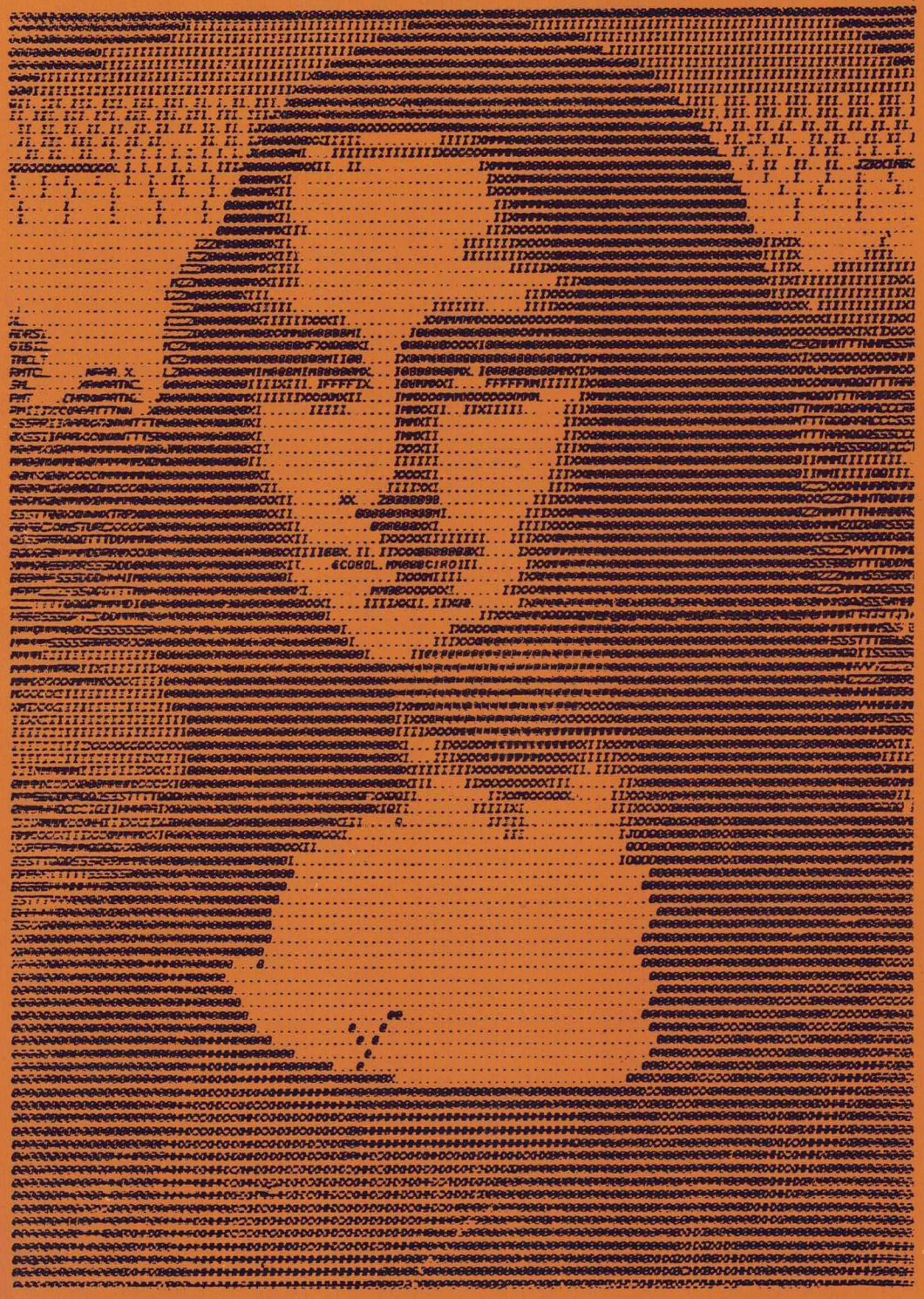


LENGUAJE COBOL

CIRO DE LA FUENTE

LENGUAJE

COBOL

CIRO DE LA FUENTE

Licenciado en Informática.
Profesor de análisis y programación en el I.N.A.P.

Obra declarada de utilidad para el Ejército (O.M. 1464/26/80)

© *Ciro de la Fuente,*
Editorial C & F, S.A,
Menorca, 38, Madrid -28009 ***Tfno. (91) 2.73.89.03***
Printed in Spain / Impreso en España
ISBN: 84 – 300 – 7214 – 4
Depósito legal: M – 15352 –1986

IMPRIME: Industria Gráfica MAE-c/ Hermanos Granda, 30 - Tfno. 747.50.00 - Madrid - 22

PROLOGO

Una de las grandes dificultades que encuentra el ser humano en los primeros meses de su vida es aprender a dar los primeros pasos, ya que aprender a correr consiste en practicar dichos pasos.

De forma análoga,este libro sólo pretende iniciar la andadura del lector en el estudio de un lenguaje de programación para ordenadores electrónicos, y será la posterior práctica la que le permita adquirir soltura y perfección.

El lenguaje de programación elegido es COBOL, por ser el que actualmente adapta mejor su aplicación a la solución de los numerosos problemas creados por la actividad comercial del hombre.

El lector encontrará,durante el estudio de este lenguaje de programación, numerosas dificultades, que se acrecentarán y llegarán a hacer imposible dicho estudio, de no tenerse en cuenta las siguientes recomendaciones:

- Antes de iniciar la lectura del texto es necesario tener conocimientos, aunque sólo sean generales, acerca del ordenador y su entorno (Unidad Central de Proceso, unidades de cinta, disco, etc.), diferentes tipos de soportes, organización de los datos en dichos soportes, así como lógica de la programación (ordinogramas).

- Es necesaria la explicación del profesor que le aclare las dudas que puedan surgirle, siendo conveniente que, en una primera lectura, no se pierda tiempo tratando de querer entender todo cuanto en el texto se especifica, pues es más importante,en principio, adquirir una visión de conjunto del lenguaje e ir profundizando en su estudio en posteriores lecturas.

Para facilitar al lector su trabajo y obtener el mayor aprovechamiento, he creído conveniente presentar, desde su comienzo, un caso práctico en el que se describen todas las fases a recorrer para llegar a la solución del problema, desde su enunciado hasta la obtención de resultados por el ordenador. Familiarizarse con este caso práctico es importante, ya que los siguientes casos propuestos están relacionados con él, así como gran parte de los ejemplos empleados como aclaración de la teoría.

Los ejercicios propuestos al final de cada capítulo, ayudan a comprobar el grado de aprovechamiento, por lo que será necesaria una nueva lectura si no se consigue,como mínimo,un setenta y cinco por ciento de aciertos.

He intentado que cuanto se describe en este libro, sea la parte común y fundamental empleada en la escritura de programas, que permita la comunicación del hombre con cualquiera de los grandes Sistemas de ordenadores existentes en la actualidad, por ejemplo: SIEMENS, UNIVAC, IBM, etc., sin embargo, al tener que emplear un ordenador para llevar a cabo la ejecución de los casos prácticos propuestos, se ha elegido un IBM; es el motivo por el cual la tendencia que puede advertirse al exponer la teoría se incline hacia este Sistema. Se omite la descripción de la parte no común a los citados Sistemas, no por ser accesoria, sino porque dificultaría más que facilitaría el estudio del lenguaje, siendo preferible que sea el usuario de un determinado Sistema el que profundice en el estudio de las particularidades del mismo,una vez que conozca lo fundamental.

Aprovecho la oportunidad para dedicar este libro a cuantos fueron mis alumnos, funcionarios pertenecientes a la Administración Civil y Militar, como forma de expresarles mi agradecimiento por haberme honrado con su presencia e interés a lo largo de los cursos de Análisis y Programación.

Ciro de la Fuente

RECONOCIMIENTO

Para guía e información del usuario, presentamos el siguiente extracto del Formulario Número 1965–0795689, editado por la U. S. Government Printing Office:

"Cualquier organización interesada en reproducir, parcial o totalmente, el informe y las especificaciones COBOL, utilizando las ideas extraídas de este informe como base para la creación de un manual de enseñanza, o para cualquier otra finalidad, es libre de hacerlo. Sin embargo, dichas organizaciones deberán reproducir esta sección en el prólogo de su documento. Aquellas que utilicen un breve pasaje, como en el caso de una reseña bibliográfica, no necesitarán reproducir totalmente esta sección, pero mencionarán "COBOL" como reconocimiento de la fuente de Información".

"El COBOL es un lenguaje industrial, y no es propiedad de ninguna empresa o grupo de empresas, ni de ninguna organización o grupo de organizaciones".

"Ni el Comité COBOL ni ninguna de las organizaciones constribuyentes garantizan, expresa o implícitamente, la exactitud y el funcionamiento del lenguaje y del sistema de programación. Asimismo, tanto las organizaciones contribuyentes, como el comité, no asumen responsabilidad alguna al respecto".

"Se han establecido normas para el mantenimiento del COBOL. Las consultas relativas a los procedimientos a seguir para proponer modificaciones deben dirigirse al Comité Ejecutivo de la Conferencia sobre Lenguajes de Sistemas de Proceso de Datos".

"Los autores y titulares de la propiedad intelectual del material empleado aquí, a saber,

> FLOW-MATIC (Marca registrada de la Sperry Rand Corporation), Programming for the Univac (R)I and II, Data Automation Systems, registrado en 1958, 1959, por Sperry Rand Corporation; IBM Commercial Translator, Form Núm. F28-8013, registrado en 1959, por IBM, y FACT, DSI 27 A5260-2760, registrado en 1960 por Minneapolis Honeywell,

han autorizado de manera específica el empleo total o parcial de este material, en las especificaciones COBOL. Tal autorización alcanza también a la reproducción y utilización de las especificaciones COBOL en los manuales de programación o publicaciones similares".

CONTENIDO

CAPITULO I.

LENGUAJE COBOL.

PAGINA

1.– Introducción 11
2.– Resolución de un caso práctico 13
2.1.– Enunciado 13
2.2.– Análisis 13
2.2.1.– Obtención de datos. 14
2.2.2.– Soportes de entrada/salida 14
2.2.3.– Diseño de entrada/salida 16
2.2.4.– Algoritmo del Proceso 20
2.2.5.– Ordinograma 21
2.3.– Programación 23
2.3.1.– Nombre del programa 28
2.3.2.– Equipo físico 28
2.3.3.– Descripción de archivos y registros 29
2.3.4.– Proceso con los datos 32
3.– Ejercicios 51

CAPITULO II.

ESTRUCTURA DEL LENGUAJE.

1.– Elementos del lenguaje COBOL 55
1.1.– Caracteres 55
1.2.– Palabras COBOL 57
1.2.1.– Nombres creados por el programador 58
1.2.2.– Palabras reservadas 60
1.3.– Constantes 60
1.3.1.– Literales numéricos 61
1.3.1.1.– Literales de coma fija 61
1.3.1.2.– Literales de coma flotante 62
1.3.3.– Literales no numéricos 63
1.3.4.– Constantes figurativas 63
1.4.– Hoja de codificación 66
1.5.– Reglas de puntuación 71
1.6.– Notación sobre formato de inscripciones 72
2.– Ejercicios 75

CAPITULO III.

ESTRUCTURA DE UN PROGRAMA.

PAGINA

1.– Generalidades 77
1.1.– Estructura de un Programa COBOL 77
2.– IDENTIFICATION DIVISION 79
2.1.– Generalidades 79
2.2.– Estructura 79
3.– ENVIRONMENT DIVISION 81
3.1.– Generalidades 81
3.2.– Estructura 81
3.3.– CONFIGURATION SECTION 82
3.3.1.– Párrafos SOURCE y OBJECT COMPUTER 82
3.3.2.– Párrafo SPECIAL–NAMES 83
3.4.– INPUT–OUTPUT SECTION 85
3.4.1.– Párrafo FILE–CONTROL 85
3.4.1.1.– Cláusula SELECT 86
3.4.1.2.– Cláusula ASSIGN 87
3.4.1.3.– Cláusula ACCESS 89
3.4.1.4.– Cláusula RESERVE 91
3.4.2.– I–O–CONTROL 92
3.4.2.1.– Cláusula RERUN 92
3.4.2.2.– Cláusula SAME 93
4.– Ejercicios 95

CAPITULO IV.

DATA DIVISION.

1.– Generalidades 97
2.– Estructura 98
3.– FILE–SECTION 99
3.1.– Descripción de Archivos 99
3.1.1.– Cláusula RECORDING MODE 100
3.1.2.– Cláusula BLOCK CONTAINS 101
3.1.3.– Cláusula RECORD CONTAINS 103
3.1.4.– Cláusula DATA RECORD 103
3.1.5.– Cláusula LABEL RECORD 105
3.2.– Descripción de Registros 106
3.2.1.– Números de Nivel 108
3.2.2.– Nombres de Datos 109
3.2.2.1.– Calificación de los Nombres de Datos 111

PAGINA

3.2.3.– Cláusula PICTURE 112
3.2.4.– Cláusula USAGE 125
3.2.5.– Cláusula JUSTIFIED 129
3.2.6.– Cláusula BLANK 130
3.2.7.– Cláusula OCCURS 131
3.2.8.– Cláusula REDEFINES 134
3.2.9.– Cláusula VALUE 137
3.2.10.–Relaciones entre VALUE, OCCURS Y REDEFINES 141
3.2.11.–Cláusula RENAMES 142
4.– WORKING–STORAGE SECTION 144
5.– Ejercicios 147

CAPITULO V.

PROCEDURE DIVISION.

1.– Generalidades 153
2.– Estructura 155
3.– Verbos aritméticos 156
3.1.– Declaraciones aritméticas 156
3.2.– Declaración ADD 156
3.3.– Declaración SUBTRACT 162
3.4.– Declaración MULTIPLY 163
3.5.– Declaración DIVIDE 164
3.6.– Declaración COMPUTE 165
3.7.– Expresión Aritmética 166
4.– Verbos de Movimiento 169
4.1.– Declaración MOVE 169
4.2.– Declaración EXAMINE 175
5.– Verbos de Comparación 179
5.1.– Declaraciones Condicionales 179
5.2.– Prueba de Condiciones 183
5.2.1.– Prueba de Relación 183
5.2.2.– Prueba de Clase 188
5.2.3.– Prueba de Signo 188
5.2.4.– Prueba de Nombre–de–Condición 189
5.3.– Condiciones Compuestas 190
5.4.– Jerarquías de IF 198
6.– Verbos de Detención 200
6.1.– Declaración STOP 200

PAGINA

7.– Verbos de Entrada/Salida 201
7.1.– Declaración OPEN 201
7.2.– Declaración READ 203
7.3.– Declaración WRITE 205
7.4.– Declaración CLOSE 209
7.5.– Declaración ACCEPT 210
7.6.– Declaración DISPLAY 212
8.– Verbos de Bifurcación 213
8.1.– Declaración GO 213
8.2.– Declaración ALTER 215
8.3.– Declaración PERFORM 217
8.4.– Declaración EXIT 233
9.– Verbos de Dirección del Compilador 234
9.1.– Declaración NOTE 234
10.– Ejercicios 235

CAPITULO VI.

TABLAS.

1.– Descripción 239
1.1.– Ordenación de tablas 241
1.2.– Tablas de longitud variable 243
1.3.– Creación de tablas en la WORKING–STORAGE SECTION 244
2.– Métodos para referenciar elementos de tablas 245
2.1.– Subindicación 245
2.1.1.– Ejemplo de manejo de tablas 248
2.2.– Indexación 254
2.2.1.– Tipos de Indexación 254
3.– Búsqueda en tablas 255
3.1.– Búsqueda Secuencial (con subíndices) 255
3.2.– Búsqueda Binaria (con subíndices) 257
3.3.– Búsqueda Secuencial (con índices) 267
3.3.1.– Declaración SET 270
3.3.2.– Cláusula USAGE INDEX 272
3.3.3.– Declaración SEARCH 273
3.4.– Búsqueda Binaria (con índices) 278
3.4.1.– Declaración SEARCH ALL 278
4.– Ejercicios 280

CAPITULO VII.

TECNICAS DE PROGRAMACION DE DISPOSITIVOS DE ACCESO DIRECTO.

PAGINA

1.– Introducción 285
2.– Dispositivos de acceso directo 286
3.– Métodos de organización 286
4.– Modos de acceso 286
5.– Organización Secuencial 288
5.1.– Creación del archivo 288
5.2.– Proceso del archivo 289
6.– Organización Secuencial Indexada 291
6.1.– Creación del archivo 292
6.2.– Proceso del archivo 296
6.2.1.– Lectura (recuperación) secuencial 296
6.2.2.– Actualización secuencial 299
6.2.3.– Lectura (recuperación) al azar 300
6.2.4.– Grabación (adiciones) al azar 302
6.2.5.– Actualización al azar 304
7.– Organización Directa 305
7.1.– Métodos de direccionamiento 306
7.1.1.– Direccionamiento directo 307
7.1.1.1.– Creación del archivo 307
7.1.1.2.– Actualización del archivo 311
7.1.2.– Direccionamiento indirecto 313
7.1.2.1.– Creación del archivo (Dirección relativa de pista) 314
7.1.2.2.– Actualización del archivo (Dirección relativa de pista) 318
7.1.2.3.– Creación del archivo (Dirección real) 319
7.1.2.4.– Actualización del archivo (Dirección real) . 324
8.– Ejercicios 324

CAPITULO VIII.

CARACTERISTICAS ESPECIALES.

PAGINA

1.– Característica de clasificación (SORT) 327
1.1.– Declaración SORT 328
1.1.1.– Opciones ASCENDING y DESCENDING 330
1.1.2.– Opciones USING y GIVING 330
1.1.3.– Opción INPUT PROCEDURE 331
1.1.3.1.– Declaración RELEASE 332
1.1.4.– Opción OUTPUT PROCEDURE 333
1.1.4.1.– Declaración RETURN 336

2.– Característica de Escritor de Informes 337
2.1.– Generalidades 337
2.2.– Formato de un informe impreso 338
2.3.– Escritura del informe 346
2.4.– Controles y rupturas de control 347
2.5.– Cláusula REPORT de la DATA DIVISION 347
2.6.– REPORT SECTION 348
2.6.1.– Entrada de descripción de informe 348
2.6.1.1.– Cláusula CONTROL 349
2.6.1.2.– Cláusula PAGE 350
2.6.2.– Entrada de descripción de grupos de informe 352
2.6.2.1.– Cláusula TYPE 354
2.6.2.2.– Cláusula COLUMN 354
2.6.2.3.– Cláusula SOURCE 355
2.6.2.4.– Cláusula SUM 355
2.6.2.5.– Contadores 356
2.6.2.6.– Cláusula LINE 357
2.6.2.7.– Cláusula GROUP INDICATE 358
2.6.2.8.– Cláusula RESET 360
2.7.– Declaraciones de la PROCEDURE DIVISION 361
2.7.1.– Declaración INITIATE 361
2.7.2.– Declaración GENERATE 362
2.7.3.– Declaración TERMINATE 362
2.8.– Programa ejemplo de la confección de la nómina 363
2.8.1.– Cláusula NEXT GROUP 373
2.8.2.– Cláusula LINE (Continuación) 374
2.8.3.– Cláusula SUM (Continuación) 374
2.8.4.– Sentencia USE 375

3.– Ejercicios 377

CAPITULO IX.

COMUNICACION ENTRE PROGRAMAS.

PAGINA

1.– Introducción 383
2.– Programa de LLAMADA y Programa LLAMADO 387
3.– Enlace entre programas 390
3.1.– Punto de llamada 390
3.1.1.– Declaración CALL 390
3.2.– Punto de entrada 393
3.2.1.– Declaración ENTRY 393
3.3.– Punto de regreso 393
3.4.– LINKAGE SECTION 394
3.5.– Consideraciones Generales 395
4.– Ejercicios (Repaso) 400

APENDICES.

A.– Soluciones a los Ejercicios 407
B.– Palabras reservadas COBOL 425
C.– Resumen de formatos 427

INDICE 437

CAPITULO I

LENGUAJE COBOL

1.- INTRODUCCION

Los ordenadores electrónicos, con su capacidad para ejecutar cálculos simples y complejos con gran rapidez y exactitud, constituyen un elemento tan valioso para el hombre que han permitido, desde liberarle de los trabajos repetitivos y rutinarios, como puede ser la confección de una nómina, hasta hacer realidad grandes proyectos, como son las exploraciones de planetas.

No obstante, aún son muchas las actividades del hombre en las que no es posible la intervención del ordenador, debido al número limitado de diferentes acciones que éste puede realizar.

Al ser el ordenador un elemento que sólo hace lo que se le ordena, es necesario conocer cuáles son las diferentes acciones que puede ejecutar, así como la forma de expresar cada una de ellas, a fin de poder comunicarse con él. Ambas cosas constituyen la materia de estudio de un lenguaje de programación.

Se puede comparar el estudio de un lenguaje de programación con el de un idioma, en el que si fundamental es el conocimiento de la gramática, necesario es el ejercicio de la práctica. Está comprobado que un programador aprende a programar cuando escribe sus programas.

Una de las cualidades que debe poseer el programador es: ser detallista, ya que si bien las personas, aún escribiendo con muchos errores de ortografía, llegan a entenderse, sin embargo, el ordenador no admite equivocación alguna, pues en la escritura de un programa, la simple omisión de un punto, guión, etc., es motivo para que el ordenador no ejecute dicho programa o, de ejecutarlo, se obtengan resultados distintos a los deseados por el programador.

Por tanto, la programación exige no omitir ninguna de las reglas que se citan en los manuales de cada lenguaje, ya que de lo contrario originaría un consumo innecesario de tiempo de ordenador y elevados gastos inútiles.

Historia del lenguaje COBOL.

El deseo de desarrollar un lenguaje de programación que fuera aceptado por cualquier marca de ordenador, así como ser apropiado para su aplicación a la realización de trabajos derivados de la actividad comercial, reunió en Estados Unidos, en Mayo de 1959, una comisión (denominada CODASYL: Conference On Data Systems Languages) integrada por fabricantes de ordenadores,

usuarios de empresas privadas y representantes del Gobierno. El resultado de sus trabajos, fue la creación de un lenguaje denominado COBOL, abreviatura de Common Business Oriented Language (Lenguaje común orientado a los negocios), siendo publicado en Abril de 1960, llamándose a esta primera versión: COBOL–60.

COBOL es un lenguaje en constante evolución, prueba de ello son las revisiones y actualizaciones llevadas a cabo por CODASYL, en 1961 (COBOL–61), en 1963 (COBOL–61 ampliado) y 1965 (COBOL, edición 1965), atendiendo a las sugerencias y recomendaciones de usuarios y expertos con la finalidad de mejorar las posibilidades primitivas.

El American National Standard Institute estandarizó el lenguaje, publicando la primera versión standard en 1968, siendo revisada en 1974. Ambas versiones se las conoce con el nombre de American National Standard COBOL (ANS COBOL). Actualmente son las utilizadas en la mayoría de los ordenadores.

El Lenguaje COBOL.

Cuando se lee un programa escrito en lenguaje COBOL, parece que se está leyendo un texto en inglés, ya que su gramática, vocabulario, etc., están tomados de dicho idioma.

En contraste con otros lenguajes de programación, COBOL no está concebido para ser aplicado a la realización de trabajos que exijan de complejos cálculos matemáticos o científicos (existen otros lenguajes como: Fortran, PL/1, etc., que son más adecuados para ello), sino que su empleo es apropiado para el proceso de datos en aplicaciones comerciales. Este tipo de aplicaciones requieren, normalmente, la ejecución de operaciones aritméticas sencillas, la utilización de gran cantidad de datos que entrarán al ordenador para ser procesados y la obtención de los resultados en forma de informes impresos.

Con COBOL se pretendía llegar a un lenguaje de programación universal, sin embargo, han sido numerosos los fabricantes que desde la creación del lenguaje han ido introduciendo para sus ordenadores especificaciones suplementarias. Este es un inconveniente que hace que en un programa escrito para ser ejecutado por un ordenador, haya que introducir pequeños retoques o modificaciones para poder ser ejecutado en otro ordenador de diferente marca.

Por otra parte, no todos los ordenadores, aún siendo de la misma marca, pueden realizar la totalidad de las especificaciones que comprende el lenguaje, ya que dicha posibilidad depende de la capacidad de la memoria de que esté dotado. Por este motivo, de una misma versión COBOL se han desarrollado varias opciones que incluyen mayor o menor número de especificaciones a fin de adaptarlo a la capacidad de la memoria de cada ordenador.

Por todo ello, en este texto se estudiarán aquellas partes del lenguaje utilizadas en la mayor parte de los ordenadores actualmente existentes y basadas en el ANS COBOL. Se deja para el futuro programador el estudio de aquellas especificaciones que son particulares de cada ordenador, y que se describen en los manuales respectivos.

2.-RESOLUCION DE UN CASO PRACTICO

El caso práctico que se presenta a continuación es un caso que,aunque ficticio, bien pudiera darse en la realidad; en él se siguen en forma muy simple –para facilitar su comprensión– cuantos pasos son necesarios para convertir un trabajo que había de hacerse de forma manual, a una forma mecanizable para que sea el ordenador el que lo lleve a cabo.

2.1.- ENUNCIADO

La empresa RICOSA está constituída por las factorías COIRSA, CIORSA, ROCISA, OCRISA, IRCOSA y OCIRSA , dedicadas a la fabricación de piezas de automóviles, así como al montaje de los mismos. Dispone en su Centro de Proceso de Datos de los medios siguientes:

- Un Ordenador IBM Sistema/370.
- Una lectora de fichas, modelo 2540.
- Una impresora, modelo 1403.
- Cuatro unidades de almacenamiento en discos, modelo 3340.
- Cuatro unidades de cinta magnética de la serie 2400.
- Cinco máquinas perforadoras de fichas.

La Dirección de la empresa encarga al equipo de análisis, realizar el estudio para la mecanización de la nómina de sus 10.000 operarios.

2.2.- ANALISIS

El equipo de análisis procede al estudio del problema con el que va a obtener una solución, la cual puede no ser la mejor, sino simplemente una de entre las múltiples soluciones que se pueden dar (1).

Un análisis elemental del problema, normalmente ha de basarse en los siguientes apartados:

a) Obtención de datos.

b) Soportes de entrada/salida.

c) Diseño de entrada/salida.

d) Algoritmo del proceso.

e) Ordinograma.

---

(1) Hay que advertir que la solución obtenida busca preferentemente fines didácticos y no se adapta a la legislación vigente en materia comercial, pero ello es intranscendente y no altera la finalidad propuesta.

2.2.1.- OBTENCION DE DATOS

La obtención de datos comprenderá el estudio de aquellos datos de cada operario necesarios para la confección de la nómina. Como consecuencia de dicho estudio, el equipo de análisis decide confeccionar el impreso de la figura 2, el cual posteriormente será enviado a la Sección de Personal de la empresa y,una vez cumplimentado, volverá al Centro de Proceso de Datos, donde asignarán un CODIGO a cada operario.

Observando dicho impreso, se solicita en él información como: fecha de nacimiento, provincia de nacimiento, etc. que, aunque no va a intervenir en la confección de la nómina, sin embargo, se incluye por haber sido solicitada por la Sección Estadística para la creación de su archivo a fin de ser utilizado en futuros trabajos.

2.2.2.- SOPORTES DE ENTRADA / SALIDA

Este apartado comprende el estudio que hará posible la elección del medio o soporte más adecuado (ficha, cinta magnética, etc.) donde registrar la información que ha de ser tratada por el ordenador, soporte de entrada , o la obtenida como resultado de dicho tratamiento, soporte de salida .

El soporte que ha de contener los datos de entrada ha de elegirse, dentro de las disponibilidades técnicas existentes en el Centro de Proceso, aquél con el que se invierta menos tiempo en introducir los datos en el ordenador.

Las disponibilidades del Centro sólo permiten utilizar la ficha perforada como soporte de entrada, aunque hubiera sido muy apropiado para este caso el empleo de la cinta magnética, pero se descarta su utilización por no disponer de unidades grabadoras de cinta.

Para la elección del soporte de salida que contendrá los resultados, hay que tener presente que éstos van a ser utilizados por la Sección de Contabilidad, lo cual exige que pueden ser leídos por el personal de dicha Sección. Como ni la cinta magnética ni el disco cumplen esta condición, deberá utilizarse como soporte de salida las hojas de papel en las que imprime la impresora.

También se utilizará como soporte de salida la cinta magnética, ya que la Sección de Estadística ha solicitado,para su empleo en trabajos propios de su Sección, se graben en dicha cinta, los datos de la ficha junto con el Sueldo Neto que se obtenga para cada operario.

Gráficamente,la disposición de dichos soportes será:

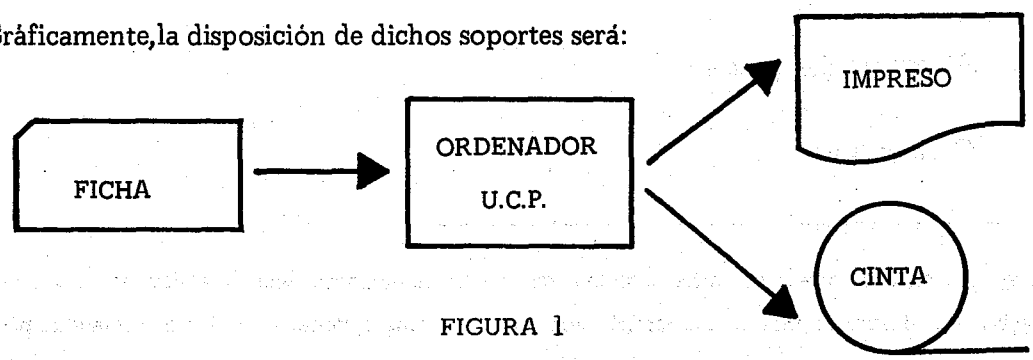


FIGURA 1

EMPRESA RICOSA **CENTRO DE PROCESO DE DATOS**

DECLARACION A EFECTOS DE MECANIZACION DE LA NOMINA (1)

APELLIDOS Y NOMBRE
BUENO VALERO ALBERTO

FECHA DE NACIMIENTO							
DIA		MES		AÑO			
2	5	0	8	1	9	4	0

FECHA ING. EN LA EMPRESA							
DIA		MES		AÑO			
1	4	0	5	1	9	6	0

SUELDO				
2	0	0	0	0

ESTADO CIVIL (2)						
S	O	L	T	E	R	O

SEXO (3)					
V	A	R	O	N	

PROVINCIA DE NACIMIENTO
ALAVA

TIPO DESCUENTO | 1 | 4 |

HORAS EXTRAS | 0 | 2 | 5 |

NUM. DE HIJOS | 0 | 0 |

NOMBRE FACTORIA					
C	O	I	R	S	A

NUM. DEPARTAMENTO | 0 | 5 |

NUM. SECCION | 6 |

CODIGO (4)				
0	0	0	0	1

(1) RELLENE ESTE IMPRESO CON LETRAS MAYUSCULAS TIPO IMPRENTA.

(2) SOLTERO, CASADO, VIUDO.

(3) VARON, HEMBRA.

(4) A RELLENAR POR EL CENTRO DE CALCULO.

FIGURA 2

2.2.3.- DISEÑO DE ENTRADA / SALIDA

El estudio del diseño de entrada/salida tiene por objeto buscar la disposición más adecuada de los datos dentro del respectivo soporte.

El impreso de la figura 2 contiene ciento seis cuadros con información que hay que incluir en cada ficha, pero teniendo en cuenta la correspondencia de cada cuadro del impreso con cada columna de la ficha,resulta en principio que son necesarias dos fichas para contener el total de los datos. No obstante, valiéndose de la codificación,se puede reducir a una el número de fichas necesarias para contener los datos de cada impreso.

Efectivamente, dentro del impreso existen datos que pueden codificarse,como es el nombre de la provincia, el estado y sexo; otros que pueden abreviarse, como son las fechas;y otros que no pueden codificarse, como es el nombre del operario.

Con la codificación de los nombres de cada provincia,tal como muestra la fig. 3,se obtendría un ahorro de 14 columnas, al tener que perforar en la ficha únicamente las dos cifras del código.

CODIFICACION DE LAS PROVINCIAS

01	ALAVA	26	LOGROÑO
02	ALBACETE	27	LUGO
03	ALICANTE	28	MADRID
04	ALMERIA	29	MALAGA
05	AVILA	30	MURCIA
06	BADAJOZ	31	NAVARRA
07	BALEARES	32	ORENSE
08	BARCELONA	33	OVIEDO
09	BURGOS	34	PALENCIA
10	CACERES	35	PALMAS, LAS
11	CADIZ	36	PONTEVEDRA
12	CASTELLON	37	SALAMANCA
13	CIUDAD REAL	38	SANTA CRUZ DE TENERIFE
14	CORDOBA	39	SANTANDER
15	CORUÑA, LA	40	SEGOVIA
16	CUENCA	41	SEVILLA
17	GERONA	42	SORIA
18	GRANADA	43	TARRAGONA
19	GUADALAJARA	44	TERUEL
20	GUIPUZCOA	45	TOLEDO
21	HUELVA	46	VALENCIA
22	HUESCA	47	VALLADOLID
23	JAEN	48	VIZCAYA
24	LEON	49	ZAMORA
25	LERIDA	50	ZARAGOZA

FIGURA 3

Codificando el dato ESTADO de la forma siguiente:

- SOLTERO [1]
- CASADO [2]
- VIUDO [3]

se ahorran seis columnas; y con la codificación del dato SEXO:

- VARON [V]
- HEMBRA [H]

se ahorran cinco columnas.

Suprimiendo las dos primeras cifras del año, y codificando las fechas como sigue: 290573 (29 de Mayo de 1973) se ahorran cuatro columnas: dos por la fecha de nacimiento y dos por la fecha de ingreso.

Con la reducción de veintinueve columnas hace posible el que todos los datos del impreso quepan en una ficha, y aún sobren tres columnas para introducir, si se desea, nuevos datos.

Queda,por último, establecer la distribución u orden en que se colocarán,dentro de la ficha, los diferentes conceptos contenidos en el cuadro de la fig. 2.

La figura 4 muestra la distribución que se ha creído oportuna (de las múltiples que se pueden hacer), así como el número de columnas de la ficha asignadas a cada concepto. Dicho número permanece invariable para todos los operarios.

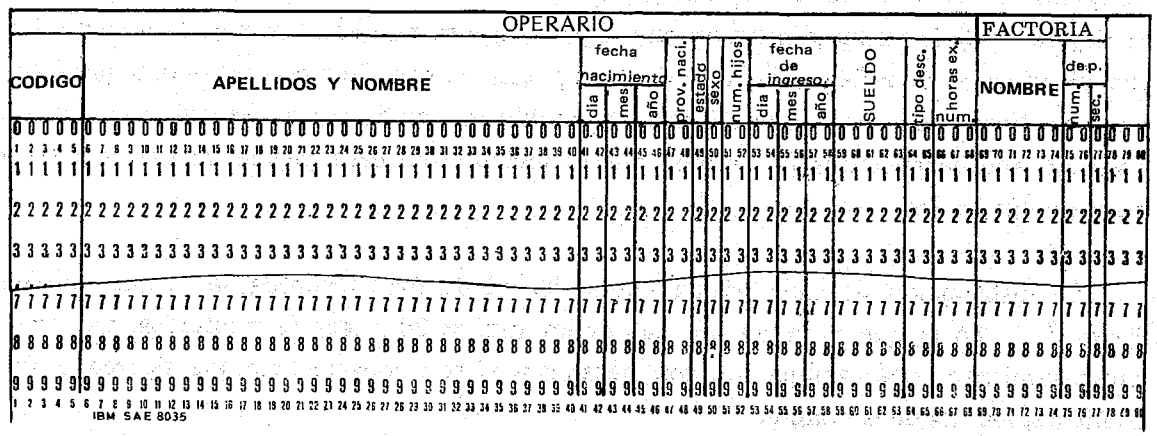


FIGURA 4

Cada uno de los conceptos: CODIGO, APELLIDOS y NOMBRE, etc., constituidos por uno o varios caracteres, reciben el nombre de CAMPOS de DATOS, y al conjunto de dichos campos se le denomina: REGISTRO.

Una vez recibidos de la Sección de Personal los impresos con los datos, se procede a perforarlos en las fichas. La figura 5 muestra una ficha perforada con los datos de un operario.

Al conjunto de las fichas conteniendo la información de todos los operarios recibe el nombre de ARCHIVO o FICHERO.

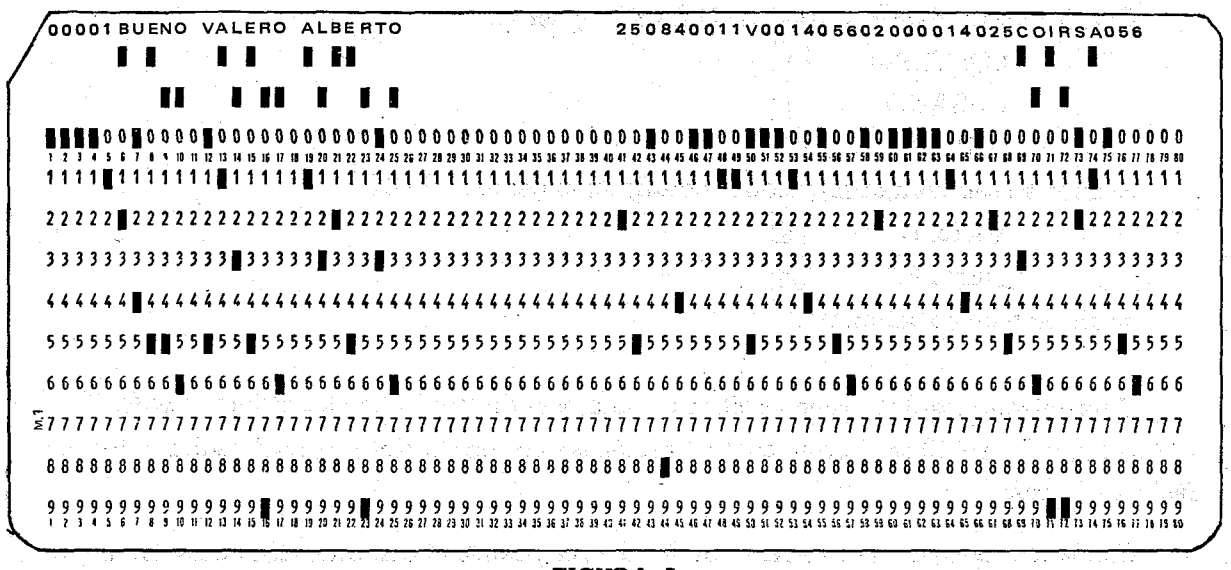


FIGURA 5

Para el estudio del diseño de salida, hay que tener en cuenta que la impresión de los resultados obtenidos durante el proceso, va a realizarlo la impresora sobre hojas de papel (dispuestas en forma de acordeón) de unas dimensiones predeterminadas, de acuerdo con el trabajo a realizar, en cada una de las cuales se puede escribir un número limitado de líneas y en cada línea hasta 132 caracteres.

Por tanto el trabajo de diseñar la salida consistirá en hacer la distribución del contenido que se ha de imprimir en cada página, utilizando una hoja-formato de salida de impresora (figura 7) de la misma dimensión que el papel que se va a emplear para la nómina. En dicha distribución se reflejará la línea de cada página en que debe comenzar la impresión; número de líneas que han de imprimirse; contenido que ha de figurar en cada línea, y su disposición; encabezamientos (cabeceras) que se incluirán en cada página; margenes y espaciado entre líneas, etc.

La distribución de cada página que se ha creído más adecuada, es la que muestra la figura 7, en la que se indica que han de imprimirse 24 líneas de resultados, así como los encabezamientos que, aunque no son necesarios, sí son convenientes que figuren, a fin de facilitar la lectura de la nómina.

El encabezamiento "NOMINA DE LA EMPRESA ....." figurará únicamente en la primera página de la nómina, y los dos siguientes: HOJA NUM.....,y CODIGO, APELLIDOS y NOMBRE .....en todas las páginas que componen la citada nómina.

Por último, la disposición de los datos en la cinta magnética será la misma que la realizada en la ficha, añadiendo únicamente el TOTAL NETO obtenido durante el proceso (figura 6).

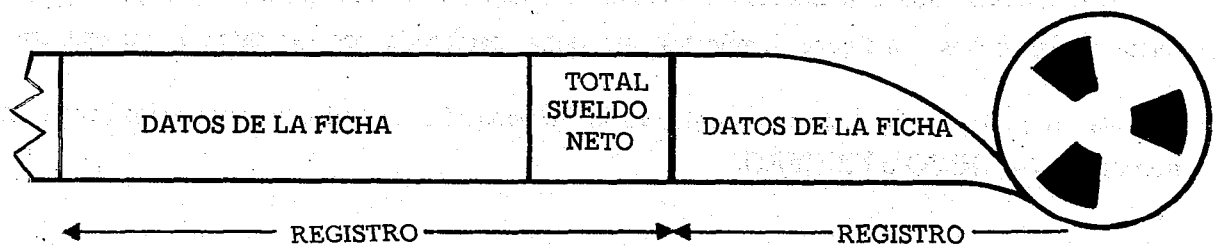
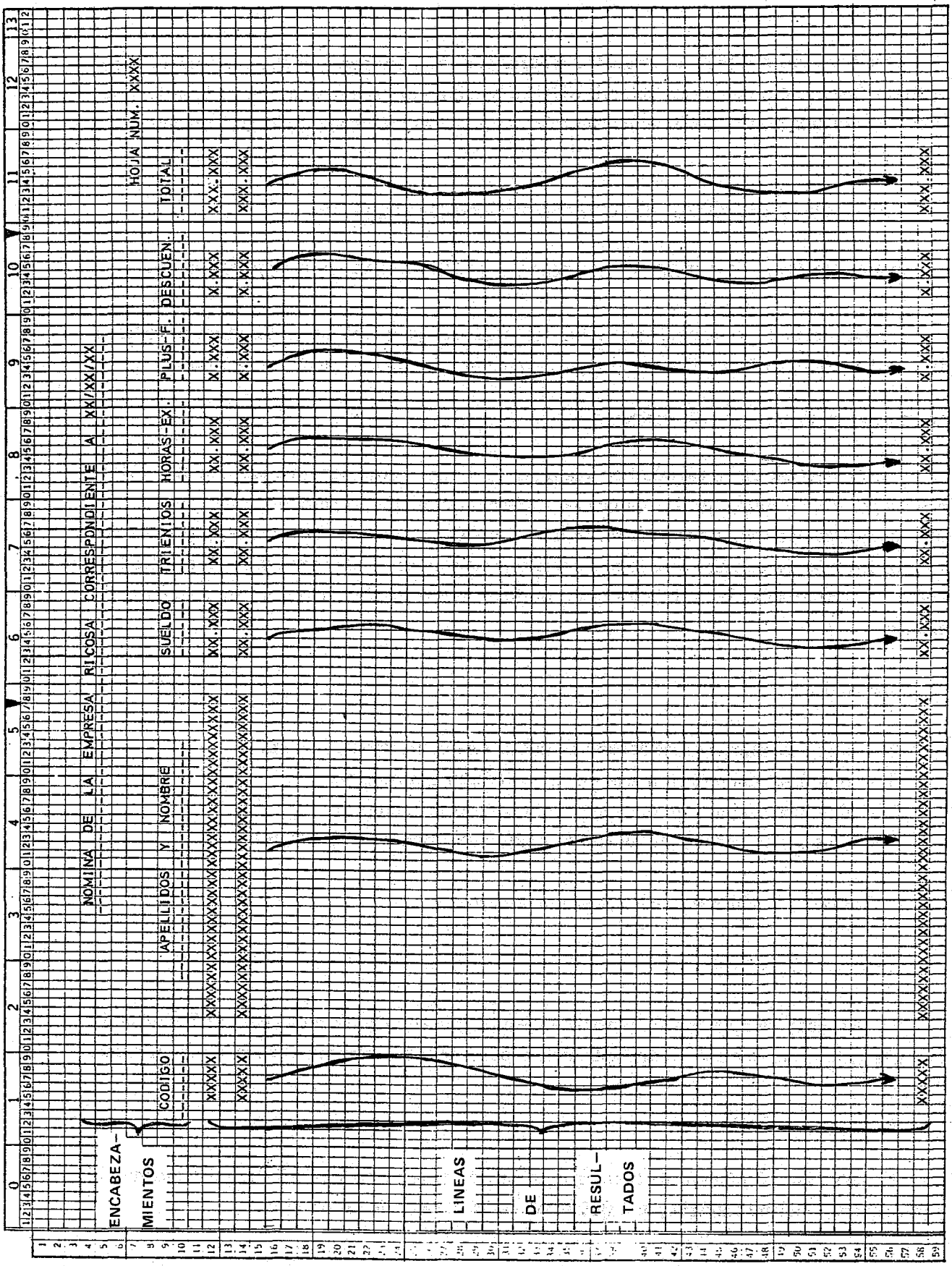


FIGURA 6

FORMATO DE SALIDA DE IMPRESORA


NOMINA DE LA EMPRESA RICOSA CORRESPONDIENTE A XX/XX/XX
HOJA NUM. XXXX
CODIGO
APELLIDOS Y NOMBRE
SUELDO
TRIENIOS
HORAS-EX.
PLUS-F.
DESCUEN.
TOTAL
XXXXX
XXXX

2.2.4.- ALGORITMO DEL PROCESO

Este apartado comprende: cálculos a realizar con los datos contenidos en la ficha de cada operario, hasta la obtención del TOTAL NETO a percibir por cada uno, así como la serie de acciones con dichos datos (movimientos, edición, etc.) necesarias para completar el trabajo.

Para el cálculo del TOTAL NETO se seguirán los siguientes pasos:

TOTAL BRUTO = SUELDO + TRIENIOS + PLUS FAMILIAR + HORAS EXTRAS

De esta fórmula, es conocido el valor del SUELDO, ya que figura en los datos de entrada de la ficha (columna 59 a 63), teniendo que calcular el valor de los restantes conceptos.

CALCULO DEL VALOR DE LOS TRIENIOS.

Para efectuar dicho cálculo hay que tener en cuenta que cada operario percibe 1000 pesetas por cada trienio, que la nómina se confeccionará el día primero de cada mes, y que, como los operarios no ingresaron en la empresa en la misma fecha, no se les puede aplicar a todos los mismos cálculos, por lo que habrá que especificar en el programa las diferentes fórmulas que resuelvan cada uno de los casos posibles que se pueden presentar, según su fecha de ingreso.

Llamando:

A – a las dos últimas cifras del año en que se va a confeccionar la nómina.
B – al número del mes en que se va a confeccionar la nómina.
C – al número del mes en que ingresó el operario en la empresa.
D – a los dos últimas cifras del año de ingreso del operario en la empresa.

Dos casos posibles se pueden presentar en este cálculo:

1.– $C < B$

En este caso, se calculará el número de trienios aplicando la fórmula (A–D) / 3. Tomando la parte entera del cociente y multiplicándola por 1.000 se obtendrá el valor de los trienios.

2.– $C \geqslant B$

En este caso, el número de trienios se calculará por la fórmula (A–D–1) / 3. Tomando la parte entera y multiplicándola por 1.000 se obtendrá el valor de los trienios.

CALCULO DEL PLUS FAMILIAR .

Para este cálculo hay que tener en cuenta que en la empresa no existen mujeres casadas y que a cada hombre casado se le abona un plus de 500 pesetas por la mujer y 250 pesetas por cada hijo.

Para el hombre casado:

PLUS FAMILIAR = 500 + (NUMERO HIJOS · 250)

Para el hombre o mujer viudo/a:

PLUS FAMILIAR = NUMERO HIJOS · 250.

CALCULO DEL VALOR DE LAS HORAS EXTRAS.

La empresa abona 150 pesetas por cada hora extra, por tanto:

VALOR HORAS EXTRAS = NUMERO HORAS EXTRAS · 150.

Conocido ya el valor de estos conceptos y sustituyéndolo en la fórmula de TOTAL BRUTO, se obtendrá el valor del mismo.

Las fórmulas siguientes permiten llegar al cálculo del TOTAL NETO:

DESCUENTO = TOTAL BRUTO · (TIPO DE DESCUENTO / 100)

TOTAL NETO = TOTAL BRUTO − DESCUENTO.

El resto de las acciones que completan el trabajo se describen en el ordinograma (fig. 8).

2.2.5.- ORDINOGRAMA

El análisis del problema se completa con la confección del Ordinograma, en el que se reflejan ordenadamente las acciones y operaciones que ha de realizar el ordenador con los datos de cada operario, tal como muestra la figura 8.

El ordinograma constituye una inestimable ayuda al programador, como medio de verificar la lógica del programa, así como llevar a cabo su codificación.

Es siempre aconsejable su confección, lo mismo si se trata de programas de cierta dificultad, con una lógica compleja, como si se trata de programas de lógica sencilla, ya que si bien en este último caso no sería necesaria su confección que sirviera de ayuda a la codificación del programa, sí lo sería a efectos de documentación.

ORDINOGRAMA

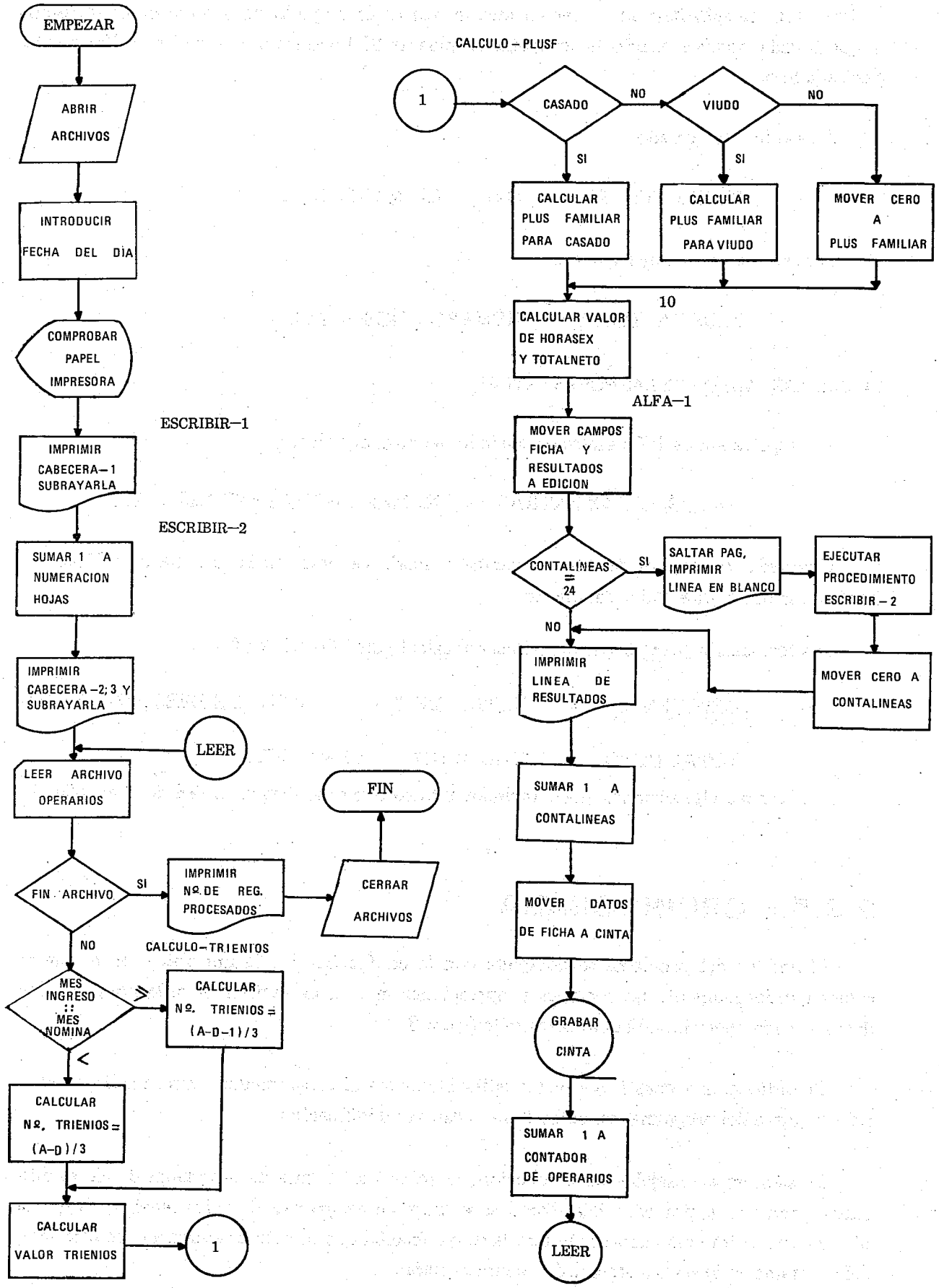


FIGURA 8

2.3.- PROGRAMACION

Una vez completo el análisis del problema, la documentación resultante se pasa al equipo de programación, el cual la estudia y, en consecuencia, procede a la escritura del PROGRAMA, última fase en la resolución del problema (figuras 9, 10, 11 y 12).

PROGRAMA

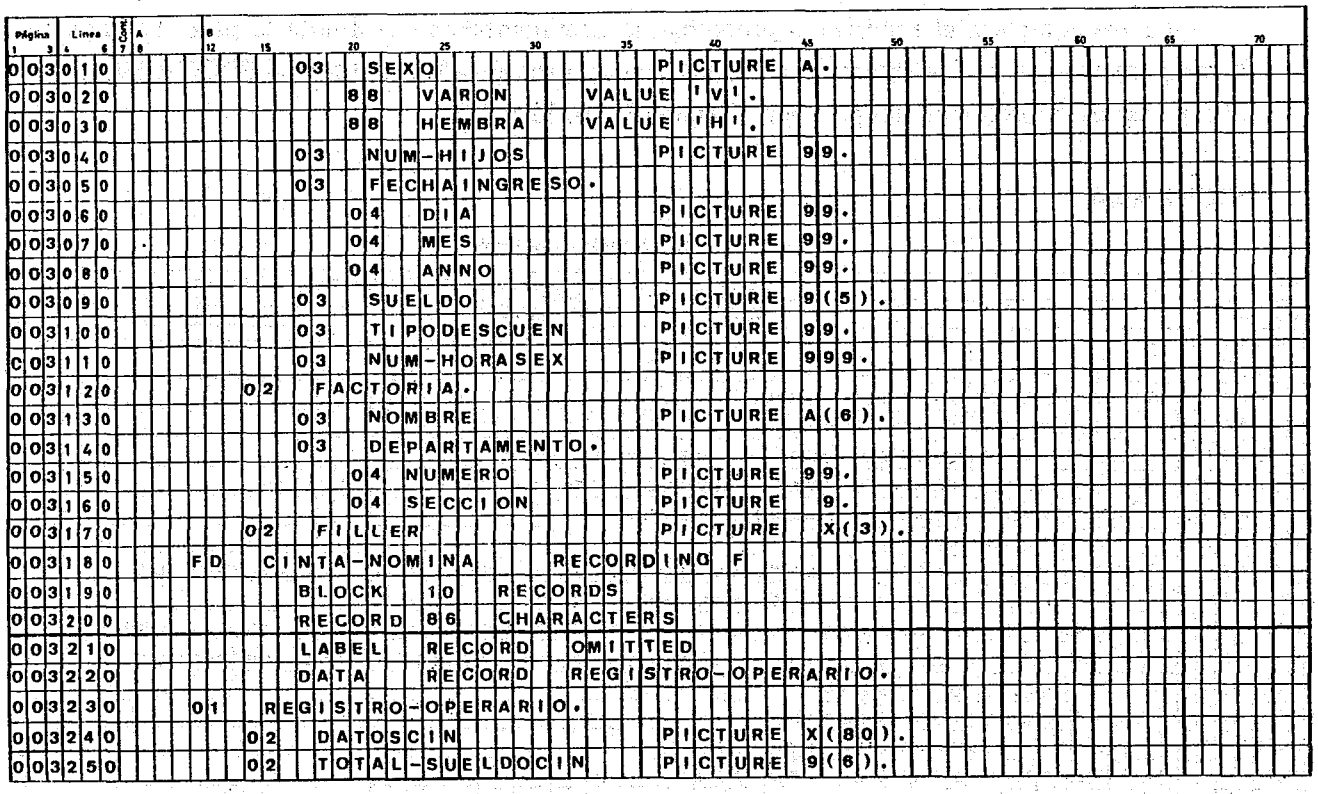
Hoja de Codificación en Cobol

PROGRAMA | SISTEMA | INSTRUCCIONES PARA PERFORACION | Identificación
PROGRAMADOR | FECHA | SIGNO | PERFORACIONES

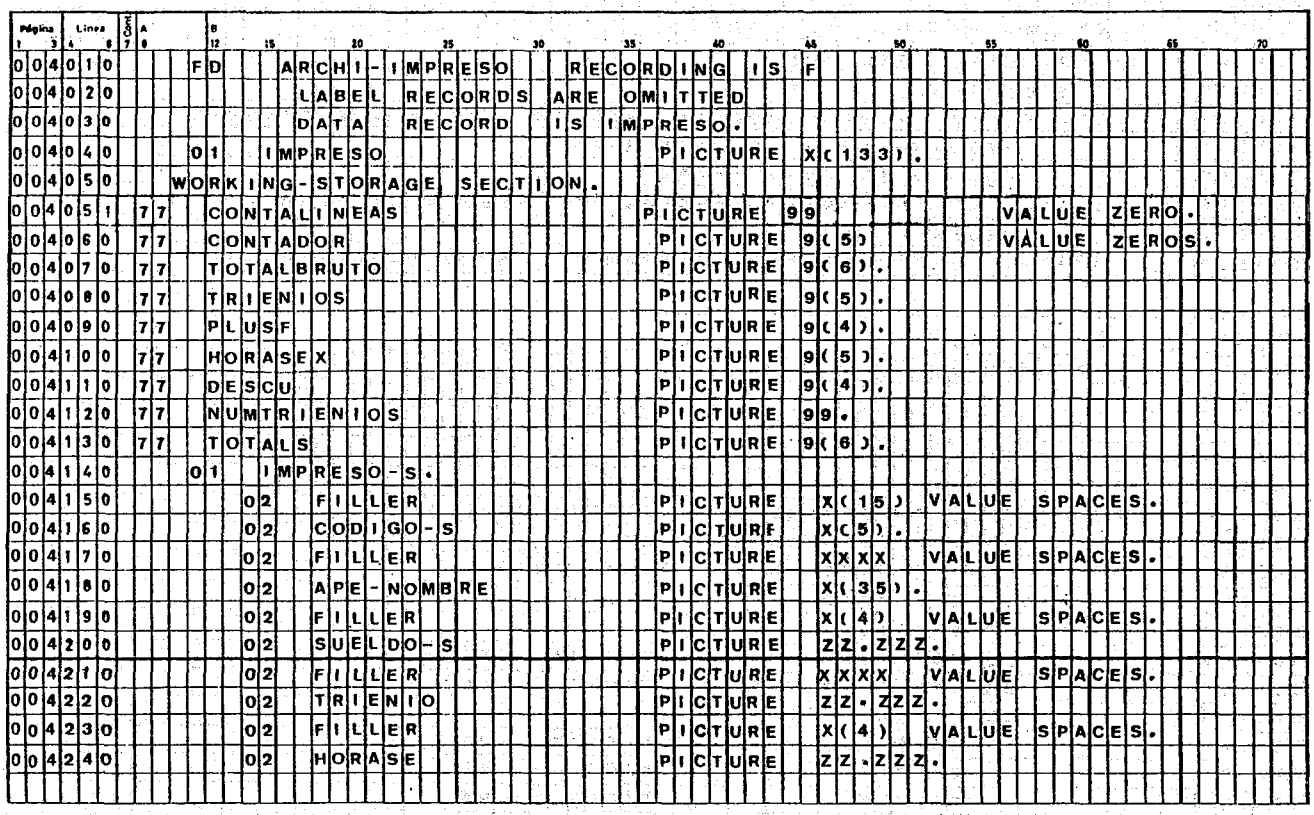
Página	Línea	Cont.	A	B
001	010			IDENTIFICATION DIVISION.
001	020			PROGRAM-ID. NOMINA.
001	030			AUTHOR. PROGRAMADOR C. DE LA FUENTE.
001	040			INSTALLATION. CENTRO PROCESO DE DATOS DE RICOSA.
001	050			DATE-WRITTEN. 15-MARZO-1977.
001	060			DATE-COMPILED.
001	070			SECURITY. ESTE PROGRAMA SOLO PUEDE SER UTILIZADO POR LA
001	080			SECCION DE CONTABILIDAD.
001	090			REMARKS. PROGRAMA PARA OBTENER LA NOMINA DE RICOSA Y CREAR
001	100	*		UN ARCHIVO EN CINTA PARA POSTERIORES TRABAJOS.
001	110			ENVIRONMENT DIVISION.
001	120			CONFIGURATION SECTION.
001	130			SOURCE-COMPUTER. IBM-370.
001	140			OBJECT-COMPUTER. IBM-370.
001	150			SPECIAL-NAMES.
001	160			C01 IS PAG-NUEVA
001	170			DECIMAL-POINT IS COMMA.
001	180			INPUT-OUTPUT SECTION.
001	190			FILE-CONTROL.
001	200			SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-UR-2540R-S
001	210			RESERVE NO ALTERNATE AREA
001	220			ACCESS IS SEQUENTIAL.
001	230			SELECT CINTA-NOMINA ASSIGN TO SYS008-UT-2400-S
001	240			RESERVE NO.
001	250			SELECT ARCHI-IMPRESO ASSIGN TO SYS009-UR-1403-S.

Página	Línea	Cont.	A	B
002	010			I-O-CONTROL.
002	020			
002	030			
002	040		DATA	DIVISION.
002	050			FILE SECTION.
002	060			FD ARCHIVO-OPERARIOS RECORDING MODE IS F
002	070			LABEL RECORD IS OMITTED
002	080			DATA RECORD IS FICHA-OPERARIO.
002	090			01 FICHA-OPERARIO.
002	100			02 OPERARIO.
002	110			03 CODIGO PICTURE X(5).
002	120			03 APELL-NOMBRE PICTURE X(35).
002	130			03 FECHANACI.
002	140			04 DIA PICTURE 99.
002	150			04 MES PICTURE 99.
002	160			04 ANNO PICTURE 99.
002	170			03 PROVINCIA-NACIM PICTURE 99.
002	180			03 ESTADO PICTURE 9.
002	190			88 SOLTERO VALUE 1.
002	200			88 CASADO VALUE 2.
002	210			88 VIUDO VALUE 3.

FIGURA 9

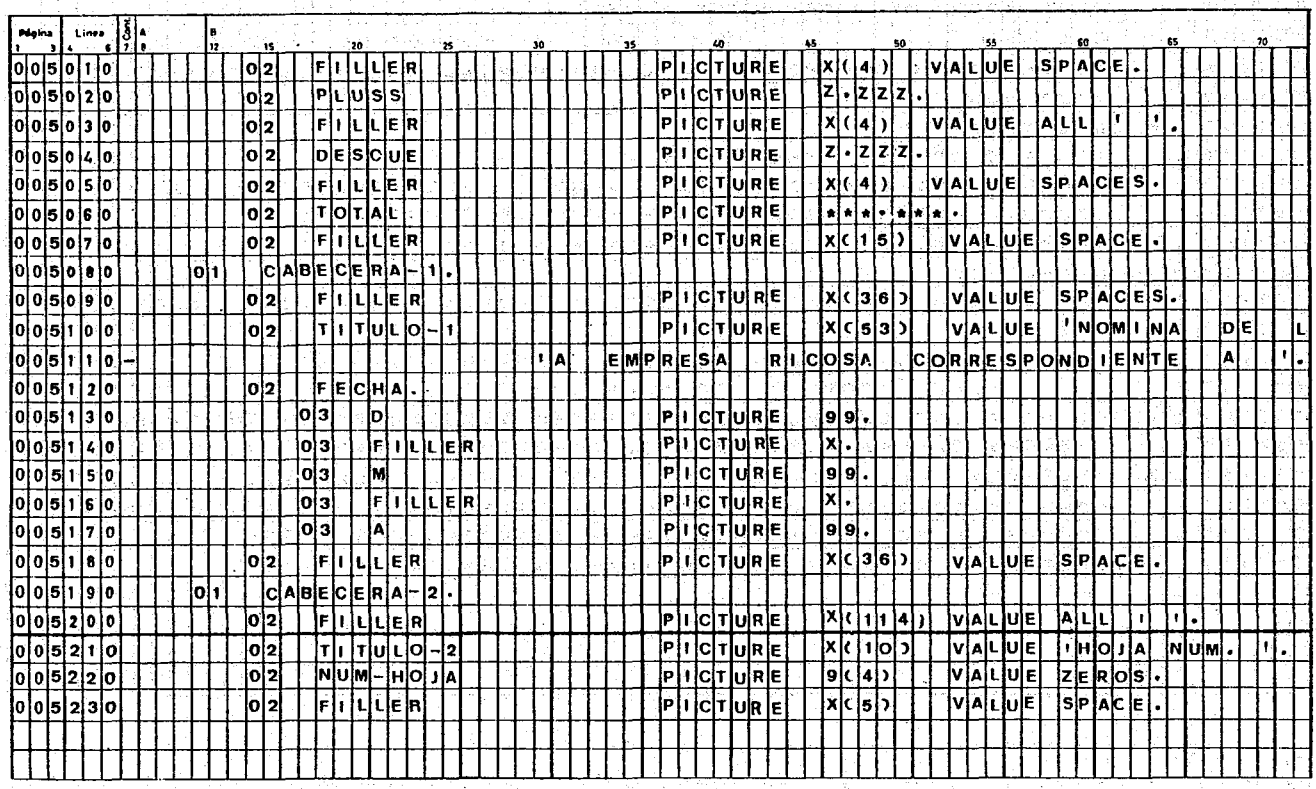


```
003010 03  SEXO                                  PICTURE A.
003020 88  VARON          VALUE 'V'.
003030 88  HEMBRA         VALUE 'H'.
003040 03  NUM-HIJOS                            PICTURE 99.
003050 03  FECHAINGRESO.
003060 04  DIA                                  PICTURE 99.
003070 04  MES                                  PICTURE 99.
003080 04  ANNO                                 PICTURE 99.
003090 03  SUELDO                               PICTURE 9(5).
003100 03  TIPODESCUEN                          PICTURE 99.
003110 03  NUM-HORASEX                          PICTURE 999.
003120 02  FACTORIA.
003130 03  NOMBRE                               PICTURE A(6).
003140 03  DEPARTAMENTO.
003150 04  NUMERO                               PICTURE 99.
003160 04  SECCION                              PICTURE 9.
003170 02  FILLER                               PICTURE X(3).
003180 FD  CINTA-NOMINA          RECORDING F
003190 BLOCK   10   RECORDS
003200 RECORD  86   CHARACTERS
003210 LABEL   RECORD   OMITTED
003220 DATA    RECORD   REGISTRO-OPERARIO.
003230 01  REGISTRO-OPERARIO.
003240 02  DATOSCIN                             PICTURE X(80).
003250 02  TOTAL-SUELDOCIN                      PICTURE 9(6).
```



```
004010 FD  ARCHI-IMPRESO    RECORDING IS F
004020 LABEL  RECORDS ARE OMITTED
004030 DATA   RECORD  IS IMPRESO.
004040 01  IMPRESO                             PICTURE X(133).
004050 WORKING-STORAGE SECTION.
004051 77  CONTALINEAS                         PICTURE 99          VALUE ZERO.
004060 77  CONTADOR                            PICTURE 9(5)        VALUE ZEROS.
004070 77  TOTALBRUTO                          PICTURE 9(6).
004080 77  TRIENIOS                            PICTURE 9(5).
004090 77  PLUSF                               PICTURE 9(4).
004100 77  HORASEX                             PICTURE 9(5).
004110 77  DESCU                               PICTURE 9(4).
004120 77  NUMTRIENIOS                         PICTURE 99.
004130 77  TOTALS                              PICTURE 9(6).
004140 01  IMPRESO-S.
004150 02  FILLER                              PICTURE X(15)  VALUE SPACES.
004160 02  CODIGO-S                            PICTURE X(5).
004170 02  FILLER                              PICTURE XXXX   VALUE SPACES.
004180 02  APE-NOMBRE                          PICTURE X(35).
004190 02  FILLER                              PICTURE X(4)   VALUE SPACES.
004200 02  SUELDO-S                            PICTURE ZZ.ZZZ.
004210 02  FILLER                              PICTURE XXXX   VALUE SPACES.
004220 02  TRIENIO                             PICTURE ZZ.ZZZ.
004230 02  FILLER                              PICTURE X(4)   VALUE SPACES.
004240 02  HORASE                              PICTURE ZZ.ZZZ.
```

FIGURA 10



```
005010  02  FILLER                  PICTURE  X(4)   VALUE SPACE.
005020  02  PLUSS                   PICTURE  Z.ZZZ.
005030  02  FILLER                  PICTURE  X(4)   VALUE ALL ' '.
005040  02  DESCUE                  PICTURE  Z.ZZZ.
005050  02  FILLER                  PICTURE  X(4)   VALUE SPACES.
005060  02  TOTAL                   PICTURE  *.***.**.
005070  02  FILLER                  PICTURE  X(15)  VALUE SPACE.
005080 01   CABECERA-1.
005090  02  FILLER                  PICTURE  X(36)  VALUE SPACES.
005100  02  TITULO-1                PICTURE  X(53)  VALUE 'NOMINA    DE    L
005110-                  'A  EMPRESA   RICOSA   CORRESPONDIENTE    A    '.
005120  02  FECHA.
005130    03  D                     PICTURE  99.
005140    03  FILLER                PICTURE  X.
005150    03  M                     PICTURE  99.
005160    03  FILLER                PICTURE  X.
005170    03  A                     PICTURE  99.
005180  02  FILLER                  PICTURE  X(36)  VALUE SPACE.
005190 01   CABECERA-2.
005200  02  FILLER                  PICTURE  X(114) VALUE ALL ' '.
005210  02  TITULO-2                PICTURE  X(10)  VALUE 'HOJA NUM.  '.
005220  02  NUM-HOJA                PICTURE  9(4)   VALUE ZEROS.
005230  02  FILLER                  PICTURE  X(5)   VALUE SPACE.
```

```
006010 PROCEDURE DIVISION.
006020 OPEN   INPUT  ARCHIVO-OPERARIOS,
006030        OUTPUT CINTA-NOMINA  ARCHI-IMPRESO,
006040 DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO  DD/MM/AA '
006050        UPON CONSOLE,
006060 ACCEPT  FECHA  FROM CONSOLE,
006070 STOP 'COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESP
006080-              'ECIAL PARA NOMINA'.
006090 ESCRIBIR-1.
006100 WRITE  IMPRESO   FROM  CABECERA-1  AFTER ADVANCING PAG-NUEVA
006110 DISPLAY '                                                  '
006120-              '                                           '.
006130 ESCRIBIR-2.
006140 ADD  1  TO  NUM-HOJA
006150 WRITE  IMPRESO FROM CABECERA-2  AFTER 2,
006160 DISPLAY '  '.  NOTE ESTE DISPLAY HACE QUE LA IMPRESORA DEJE UNA
006170                  LINEA EN BLANCO.
006180 DISPLAY '                  CODIGO          APELLIDOS  Y  NOMBRE
006190-              '                SUELDO   TRIENIOS  HORAS-EX.   PLUS-F.  DESC
006200-              'UEN.    TOTAL'.
006210 DISPLAY '                                                  '
006220-              '                                           '
006230-              '      '.
```

FIGURA 11

```
007010  LEER.
007020      READ ARCHIVO-OPERARIOS; AT END,
007030      DISPLAY ' REGISTROS PROCESADOS = ' CONTADOR,
007040      CLOSE ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050      STOP RUN.
007060  CALCULO-TRIENIOS.
007070      IF MES OF FECHAINGRESO < M,
007080      COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO)) / 3 ELSE
007090      COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100      COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
007110  CALCULO-PLUSF.
007120      IF CASADO COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007130          ELSE
007140              IF VIUDO COMPUTE PLUSF = NUM-HIJOS * 250
007150                  ELSE MOVE 0 TO PLUSF.
007160  AVISO. NOTE LAS CUATRO DECLARACIONES SIGUIENTES CALCULAN EL VALOR
007170          HORAS EXTRAS, TOTAL BRUTO, DESCUENTO (DESCU) Y TOTAL NETO.
007180  10. MULTIPLY NUM-HORASEX BY 150 GIVING HORASEX,
007190      COMPUTE TOTALBRUTO = SUELDO + TRIENIOS + PLUSF + HORASEX
007200      COMPUTE DESCU = (TOTALBRUTO * TIPODESCUEN) / 100
007210      SUBTRACT DESCU FROM TOTALBRUTO GIVING TOTALS.
```

```
008010  ALFA-1.
008020      MOVE CODIGO TO CODIGO-S
008030      MOVE APELL-NOMBRE TO APE-NOMBRE
008040      MOVE SUELDO TO SUELDO-S
008050      MOVE TRIENIOS TO TRIENIO
008060      MOVE PLUSF TO PLUSS
008070      MOVE HORASEX TO HORASE
008080      MOVE DESCU TO DESCUE
008090      MOVE TOTALS TO TOTAL, TOTAL-SUELDOCIN
008100      MOVE SPACES TO IMPRESO. NOTE ESTA DECLARACION LIMPIA EL
008110          AREA DE SALIDA (PONE BLANCOS).
008120      IF CONTALINEAS = 24 WRITE IMPRESO AFTER PAG-NUEVA
008130      PERFORM ESCRIBIR-2 MOVE ZERO TO CONTALINEAS.
008140      WRITE IMPRESO FROM IMPRESO-S AFTER 2 ADD 1 TO CONTALINEAS
008150      MOVE FICHA-OPERARIO TO DATOSCIN
008160      WRITE REGISTRO-OPERARIO
008170      ADD 1 TO CONTADOR
008180      GO TO LEER.
```

FIGURA 12

De forma similar que en el análisis, la solución dada es una de las múltiples que se pueden obtener y, aunque sigan distintos caminos, todas deben conducir a los mismos resultados.

Es difícil conocer por la simple inspección de un programa cuál es la mejor solución, pero, si se aplica un criterio económico, se puede afirmar que la mejor será aquélla en que el tiempo empleado por el ordenador en ejecutar el trabajo sea mínimo. Sin embargo, en este programa no se han buscado fines económicos, sino didácticos.

Es necesario insistir en la importancia de compenetrarse con esta solución, ya que gran parte de los ejemplos que se muestran a lo largo de este texto, estan tomados de dicho programa.

En esencia, en cada PROGRAMA, el programador debe proporcionar al ordenador información sobre los siguientes puntos:

- NOMBRE dado al programa para su identificación.

- EQUIPO FISICO (Unidad central, lectora, impresora, etc.) encargado de leer los datos de cada operario, introducirlos en el ordenador, procesarlos y escribir los resultados.

- DESCRIPCION DE LA ORGANIZACION de los archivos y registros que contienen los datos de los operarios y de aquéllos que contendrán los resultados, así como descripción de las áreas o zonas a reservar en la memoria donde almacenar datos y resultados parciales.

- PROCESO a ejecutar con los datos, a fin de obtener los resultados apetecidos.

La información de cada uno de estos apartados se corresponde con las cuatro partes o divisiones de que se compone todo programa, describiéndolas en el orden siguiente:

- IDENTIFICATION DIVISION (comprende de la línea 001010 a la 001100)

- ENVIRONMENT DIVISION (de la línea 001110 a la 002010)

- DATA DIVISION (de la línea 002040 a la 005230) y

- PROCEDURE DIVISION (de la línea 006010 a la 008180).

En los siguientes apartados se hace una somera descripción del contenido de cada DIVISION, con la finalidad de adquirir una visión de conjunto del contenido de un programa escrito en Lenguaje COBOL.

2.3.1.- NOMBRE DEL PROGRAMA

Dentro de la IDENTIFICATION DIVISION se describe,en la línea 001020, el nombre del programa: NOMINA, y, en las líneas siguientes, otra información fácilmente comprensible, que no obstante, se describirá cuando se estudien detenidamente cada una de las cuatro Divisiones.

```
001010 IDENTIFICATION DIVISION.
001020     PROGRAM-ID. NOMINA.
001030     AUTHOR. PROGRAMADOR C. DE LA FUENTE.
001040     INSTALLATION. CENTRO PROCESO DE DATOS DE RICOSA.
001050     DATE-WRITTEN. 15-MARZO-1977.
001060     DATE-COMPILED.
001070     SECURITY. ESTE PROGRAMA SOLO PUEDE SER UTILIZADO POR LA
001080                SECCION DE CONTABILIDAD.
001090     REMARKS. PROGRAMA PARA OBTENER LA NOMINA DE RICOSA Y
001100              CREAR UN ARCHIVO EN CINTA PARA POSTERIORES TRABAJOS.
```

2.3.2.- EQUIPO FISICO

En la ENVIRONMENT DIVISION se describe: en las líneas 001130 y 001140, cuál es el modelo de ordenador con el que se compila y ejecuta el programa, en este caso un IBM-370.

En la línea 001200 se especifica el nombre dado por el programador al archivo que contiene todos los datos de cada operario: ARCHIVO-OPERARIOS, así como que el equipo físico que va a leer las fichas de este archivo será la lectora de fichas modelo 2540.

```
001110 ENVIRONMENT DIVISION.
001120     CONFIGURATION SECTION.
001130         SOURCE-COMPUTER. IBM-370.
001140         OBJECT-COMPUTER. IBM-370.
001150         SPECIAL-NAMES.
001160             C01 IS PAG-NUEVA
001170             DECIMAL-POINT IS COMMA.
001180     INPUT-OUTPUT SECTION.
001190         FILE-CONTROL.
001200             SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-UR-2540R-S
001210                 RESERVE NO ALTERNATE AREA
001220                 ACCESS IS SEQUENTIAL.
001230             SELECT CINTA-NOMINA ASSIGN TO SYS008-UT-2400-S
001240                 RESERVE NO.
```

En la línea 001230 se especifica el nombre del archivo, CINTA-NOMINA, el cual es una cinta magnética donde se irán grabando, en cada registro, los datos de cada ficha del ARCHIVO-OPERARIOS junto con el Sueldo Neto de cada operario.

El equipo físico encargado de dicha grabación es una Unidad de Cinta de la serie 2400.

En la línea 001250 se especifica el nombre del archivo que contendrá los resultados de la nómina en forma impresa, ARCHI–IMPRESO, siendo el equipo físico encargado de escribirlos una impresora modelo 1403.

```
001250     SELECT ARCHI-IMPRESO ASSIGN TO SYS009-UR-1403-S.
```

Lo mismo que en la anterior DIVISION, existe otra información cuya descripción se hará más adelante.

2.3.3.- DESCRIPCION DE ARCHIVOS Y REGISTROS

Antes de hacer una somera descripción de la DATA DIVISION, y para mejor comprensión y conocimiento de cómo trabaja el ordenador, se tendrán en cuenta las siguientes consideraciones:

- Para que el ordenador pueda utilizar los datos contenidos en un archivo y poder realizar con ellos las acciones previstas, es necesario que éstos previamente se introduzcan en su memoria, por lo que habrá que reservar un área o zona de la misma que sea capaz de almacenar, como mínimo, la información contenida en un registro. A esta zona se la llama AREA DE ENTRADA.

- Para poder almacenar los resultados parciales, obtenidos durante el proceso al ejecutar operaciones con los datos previamente introducidos, se reservarán otras zonas de la memoria, llamadas AREAS o CAMPOS DE TRABAJO.

- Normalmente, para poder imprimir, grabar, etc. los resultados finales o cualquier otra clase de información es necesario que, previamente pase dicha información a ocupar una zona de la memoria que, como las anteriores, reserva el programador. A esta zona se la llama AREA DE SALIDA.

- Si existen varios archivos de entrada y/o salida, el programador debe reservar tantas áreas de entrada como archivos de entrada, y tantas áreas de salida como archivos de salida; sin embargo, puede reservar un número variable de campos de trabajo en función de las necesidades del proceso.

Para el proceso de cada una de las fichas del ARCHIVO–OPERARIOS se emplean las áreas citadas. En cualquier momento de dicho proceso, dentro del área de entrada, se hallará, únicamente, el contenido de los datos de una ficha, hasta que, finalizado su proceso, la siguiente ficha sea leída y pase la información contenida en ella a ocupar la citada área de entrada, sustituyendo, por tanto, a la de la ficha anterior.

Lo mismo se puede decir de las áreas de salida y de trabajo, en cuanto que si se introduce un nuevo dato, éste borra previamente al anterior.

La DATA DIVISION comienza con la descripción del archivo de entrada, al que el programador dió, en la ENVIRONMENT DIVISION, el nombre de ARCHIVO–OPERARIOS y del que el ordenador, por medio de su lectora, irá tomando los datos de cada operario. Esta descripción comprende, entre otras cosas, la indicación al ordenador de que todos los registros que componen este archivo tienen la misma longitud (línea 002060), ya que cada uno de ellos se compone de los datos contenidos en las ochenta columnas de la ficha y que el nombre dado por el programador a cada registro es FICHA–OPERARIO (línea 002080).

```
002040  DATA DIVISION.
002050      FILE SECTION.
002060      FD ARCHIVO-OPERARIOS    RECORDING MODE IS F
002070          LABEL RECORD IS OMITTED
002080          DATA  RECORD IS FICHA-OPERARIO.
```

Una vez descrito como está organizado el archivo de entrada, se describe como está organizado cada uno de los registros que constituyen dicho archivo. Desde la línea 002090 a 003170 se describe la organización del registro, constituída por los nombres que ha dado el programador a cada uno de los campos del registro y relacionados en el mismo orden en que figuran en la ficha, así como el número de posiciones de la memoria (octetos) a reservar para cada uno de ellos.

Así, por ejemplo, en la línea 002120 se especifica que el nombre simbólico dado al segundo campo de la ficha es APELL–NOMBRE, y que ha reservado en memoria treinta y cinco octetos, PICTURE X (35), y en la línea 002140 el nombre dado al tercer campo, DIA, y reserva dos octetos PICTURE 99 (un octeto por cada símbolo 9), etc.

```
002090  01 FICHA-OPERARIO.
002100      02 OPERARIO.                                          OCTETOS
002110          03 CODIGO              PICTURE X(5).              5
002120          03 APELL-NOMBRE        PICTURE X(35).             35
002130          03 FECHANACI.
002140              04 DIA            PICTURE 99.                2
002150              04 MES            PICTURE 99.                2
002160              04 ANNO           PICTURE 99.                2
002170          03 PROVINCIA-NACIM     PICTURE 99.                2
002180          03 ESTADO              PICTURE 9.                 1
002190              88 SOLTERO  VALUE 1.
002200              88 CASADO   VALUE 2.
002210              88 VIUDO    VALUE 3.
```

En el margen derecho de la descripción que antecede, se indica el número de octetos que se reservan para cada uno de los campos que componen el registro.

Sumando el conjunto de octetos, se obtiene una reserva de ochenta, equivalente a las ochenta columnas de la ficha. Este conjunto es lo que constituye el área de entrada, y será el lugar físico de la memoria donde se irán situando, sucesivamente y a medida que se procesen, los datos de cada ficha del ARCHIVO–OPERARIOS.

De la línea 003180 a la 003220 se hace la descripción de uno de los archivos de salida, al que se le ha dado el nombre simbólico de CINTA–NOMINA, y en el que, como ya se indicó, se irán grabando todos los datos de la ficha de cada operario, junto con el Total Neto. En la línea 003180 se describe que los registros de este archivo serán todos de la misma longitud, o, también denominados, de longitud fija (RECORDING F); en la línea 003190, que los registros se grabarán en la cinta agrupados formando bloques de 10 registros; y en la línea 003220, que el nombre dado a cada registro es REGISTRO–OPERARIO.

```
003180     FD  CINTA-NOMINA     RECORDING F
003190         BLOCK  10  RECORDS
003200         RECORD  86  CHARACTERS
003210         LABEL  RECORD  OMITTED.
003220         DATA  RECORD  REGISTRO-OPERARIO.
```

En las líneas 003240 y 003250 se especifica el nombre de los campos que constituyen el registro, junto con el número de octetos reservados para cada uno.

```
003230     01  REGISTRO-OPERARIO.                                  OCTETOS
003240         02  DATOSCIN                  PICTURE X(80).        80
003250         02  TOTAL-SUELDOCIN           PICTURE 9(6).          6
```

En este caso, se reserva un área de la memoria de 86 octetos para cada registro, pero por deseo del programador y para ahorrar tiempo en la grabación de los registros, así como espacio en la cinta, ha decidido reunirlos y que se vayan grabando formando bloques de diez registros (línea 003190), para lo cual el Sistema exige tener que reservar un área capaz de almacenar los diez registros; por consiguiente, la zona será de 860 octetos (86 x 10) para cada bloque, la cual constituye el área de salida, y desde ella se iran grabando en la cinta magnética los registros que se hayan introducido en dicho área.

Desde la línea 004010 a la 004030 se hace la descripción del otro archivo de salida denominado ARCHI–IMPRESO, sobre el que se imprimirán los resultados; y en la línea 004040 la descripción del registro, con su nombre y longitud del área reservada (133 octetos).

El conjunto de los 133 octetos constituye el otro área de salida, donde se irán colocando los resultados y desde allí pasarán a imprimirse.

```
004010     FD  ARCHI-IMPRESO  RECORDING IS F
004020         LABEL RECORDS ARE OMITTED
004030         DATA  RECORD  IS IMPRESO.
004040     01  IMPRESO                    PICTURE X(133).
004050   WORKING-STORAGE SECTION.
004051  77  CONTALINEAS              PICTURE 99      VALUE ZERO.
004060  77  CONTADOR                 PICTURE 9(5)    VALUE ZEROS.
004070  77  TOTALBRUTO               PICTURE 9(6).
004080  77  TRIENIOS                 PICTURE 9(5).
004090  77  PLUSF                    PICTURE 9(4).
004100  77  HORASEX                  PICTURE 9(5).
```

De la línea 004051 a la 005230 se encuentran las diferentes áreas o campos de trabajo que han de ser utilizadas para: CONTADORES (líneas 004051 y 004060); contener RESULTADOS PARCIALES (líneas 004070 a 004130); EDICION (hacer más legibles los resultados, líneas 004140 a 005070) y contener ENCABEZAMIENTOS (para ser escritos literalmente al comienzo de cada impreso, líneas 005080 a 005230).

La figura 13 muestra, de forma gráfica, la disposición física en la memoria del ordenador de las diferentes áreas y campos que se han reservado al hacer su descripción dentro de la DATA DIVISION. En dicha figura se reflejan para cada campo:

- Nombre dado por el programador.
- Número de posiciones de memoria reservadas.
- Valor con el que se inicializa cada uno de ellos, antes de la ejecución del programa.

En aquellos campos en los que figura el símbolo - I -, significa que no se han inicializado con un determinado valor, sino que su contenido es el que se introdujo durante la ejecución de programas anteriores.

2.3.4.- PROCESO CON LOS DATOS

El proceso se llevará a cabo por medio de las declaraciones o instrucciones contenidas en la PROCEDURE DIVISION (líneas 006010 a 008180).

Dichas declaraciones se irán ejecutando en el orden en que las especificó el programador, pudiendo ser alterado dicho orden cuando:

- Exista una declaración que obligue al ordenador a elegir entre dos caminos posibles
- Se le ordene así explícitamente.

Para facilitar el estudio de cuanto se especifica dentro de esta DIVISION, se seguirá

M E M O R I A

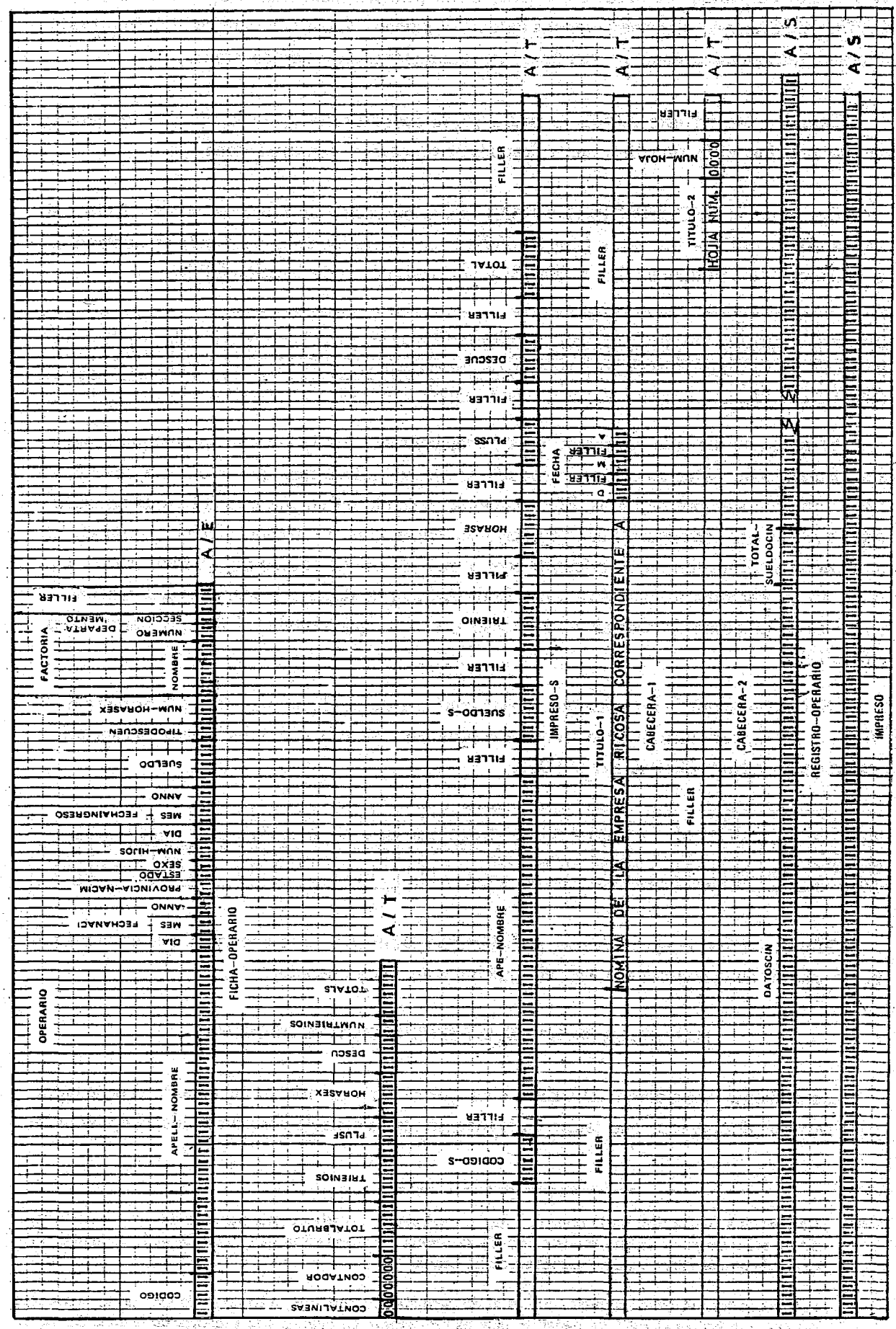


FIGURA 13

el proceso utilizando los datos contenidos en la primera ficha (figura 5) del ARCHIVO-OPERARIOS; siendo el proceso con los datos de las fichas restantes similar al de la primera, alterado únicamente cuando existan diferentes condiciones en los datos citados.

También facilita este estudio la figura 14, la cual muestra el contenido de cada área o campo de la memoria inmediatamente después de procesada la primera ficha.

Comienza la PROCEDURE DIVISION con la operación de preparar para ser leído el archivo que contiene los datos (línea 006020), así como los archivos donde se imprimirán o grabarán los resultados (línea 006030), por medio de la declaración OPEN.

```
006010  PROCEDURE DIVISION.
006020      OPEN   INPUT   ARCHIVO-OPERARIOS,
006030             OUTPUT  CINTA-NOMINA  ARCHI-IMPRESO,
```

El programa ha sido concebido para poder confeccionar la nómina en cualquier mes y año, tratando de evitar tener que introducir en él, cada nuevo mes que se ejecute, modificaciones que son siempre molestas. Efectivamente, teniendo en cuenta que el tipo de operaciones a realizar con los datos de cada operario es constante , y variable la fecha de confección de la nómina (día primero de diferente mes y aún año), se ha optado por introducir la citada fecha en el programa, utilizando un medio de entrada de datos a la memoria del ordenador como es la máquina de escribir de la consola, con lo que se evita el inconveniente antes citado (2).

Suponiendo que la fecha en que se está confeccionando la nómina es el día primero de Abril de 1978, para introducir la fecha en el ordenador se utilizan las declaraciones especificadas desde la línea 006040 a la 006060:

```
006040          DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA '
006050                  UPON CONSOLE,
006060          ACCEPT FECHA FROM CONSOLE.
```

Efectivamente, la ejecución de la declaración DISPLAY lleva consigo el que en la máquina de escribir de la consola aparezca el siguiente mensaje al operador:

MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA

La siguiente declaración, ACCEPT, hace que se detenga momentáneamente el ordenador, para permitir al operador que teclee en la máquina de la consola la fecha citada; después pulsa

---

(2) La máquina de escribir de la consola está, normalmente, dotada de una pantalla de rayos catódicos, o de un rollo de papel (según modelo) donde se refleja la información que se recibe de la Unidad de Proceso o, la que se teclea para ser enviada a dicha Unidad.

El modelo más utilizado actualmente es el de pantalla, sin embargo, a lo largo de este texto se nombra aquella que utiliza el rollo de papel, lo cual es indiferente en cuanto a la filosofía de la función que se realiza.

una tecla especial de dicha máquina que permite que:

- La fecha pase automáticamente a la memoria, colocándose en el campo o área de la misma denominado FECHA, citado a continuación de la palabra ACCEPT, y descrito de la línea 005120 a la 005170, y acto seguido,

- Que el ordenador continúe la ejecución del programa.

Dicha fecha quedaría colocada en el campo citado como sigue:

```
005120          02  FECHA.                                                  FECHA
005130            03  D                    PICTURE  99.                     01
005140            03  FILLER               PICTURE  X.                      /
005150            03  M                    PICTURE  99.                     04
005160            03  FILLER               PICTURE  X.                      /
005170            03  A                    PICTURE  99.                     78
005180          02  FILLER                 PICTURE  X(36)  VALUE SPACE.
```

A continuación, y por la declaración STOP (línea 006070), hace un nuevo alto el ordenador y envía otro mensaje al operador para que compruebe si el papel que en ese momento está colocado en la impresora es el que se va a emplear para confección de nóminas, ya que en ella podría estar colocado un papel que habría estado siendo utilizado en otra clase de trabajo y, por tanto, no ajustarse al empleado para la nómina. Una vez comprobado, el operador ordena que continúe el proceso pulsando la tecla especial de la máquina de escribir de la consola.

```
006070          STOP 'COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESP
006080-             'ECIAL PARA NOMINA'.
```

La impresora, obedeciendo la declaración WRITE (línea 006100), hace saltar el papel hasta que éste se coloque al principio de una nueva página, mueve al área de salida IMPRESO el contenido del campo CABECERA-1 (descrito de la línea 005080 a la 005180) y lo escribe sobre una línea de dicha página, es decir, imprimirá el encabezamiento:

NOMINA DE LA EMPRESA RICOSA CORRESPONDIENTE A 01/04/78

y por medio de la declaración DISPLAY (líneas 006110 y 006120) subraya el citado encabezamiento (figura 17):

```
006090          ESCRIBIR-1.
006100              WRITE IMPRESO FROM CABECERA-1 AFTER ADVANCING PAG-NUEVA
006110              DISPLAY '
006120-                     '.
```

Seguidamente, el ordenador suma el valor 1 al campo NUM–HOJA (descrito en la línea 005220), al cual, en el comienzo de la ejecución del programa, se le dió el valor 0000, (VALUE ZEROS), con lo que pasa a valer 0001; inmediatamente después, la impresora, ejecutando la declaración WRITE (línea 006150), traslada el contenido del campo CABECERA–2 (descrito de la línea 005190 a la 005230) al área de salida, salta dos líneas de la página del impreso desde la línea donde escribió el primer encabezamiento y escribe el contenido de dicha área, es decir, HOJA NUM. 0001.

```
006130 ESCRIBIR-2.
006140     ADD 1 TO NUM-HOJA
006150     WRITE IMPRESO FROM CABECERA-2 AFTER 2,
```

La declaración DISPLAY (línea 006160) hace que la impresora deje una línea en blanco en el impreso y, por la declaración DISPLAY (línea 006180), escriba sobre el impreso el título : CODIGO, APELLIDOS Y NOMBRE, SUELDO, etc., e inmediatamente después (línea 006210 a 006230) subraya dicho título.

```
006160     DISPLAY ' '. NOTE ESTE DISPLAY HACE QUE LA IMPRESORA DEJE UNA
006170                  LINEA EN BLANCO.
006180     DISPLAY '                CODIGO          APELLIDOS  Y  NOMBRE
006190-                '                SUELDO   TRIENIOS  HORAS-EX.  PLUS-F.  DESC
006200-                'UEN.    TOTAL'.
006210     DISPLAY '                -------          ----------  ----------  ----------
006220-                '                -------          ----------  ----------  ----------
006230-                '-----    ----------'.
```

Por tanto, sobre el impreso hay ya escritos tres títulos o encabezamientos, estando subrayados el primero y el tercero (figura 17).

La declaración READ de la línea 007020 lee la primera ficha e introduce los datos en el área de entrada en el mismo orden en que se encuentran en la ficha, por lo que el valor 00001 se colocará dentro del campo CODIGO (descrito en la línea 002110), BUENO VALERO ALBERTO se situará dentro del campo llamado APELL–NOMBRE (descrito en la línea 002120), y así sucesivamente los restantes datos (figura 14).

```
007010     LEER.
007020     READ ARCHIVO-OPERARIOS; AT END,
007030     DISPLAY ' REGISTROS PROCESADOS = ' CONTADOR,
007040     CLOSE ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050     STOP RUN.
```

Con los datos de la primera ficha dentro de la memoria es el momento en que el ordenador puede comenzar a calcular el valor de TRIENIOS, PLUS FAMILIAR, HORAS EXTRAS, etc., hasta llegar a la obtención del TOTAL NETO, de la forma siguiente:

Comienza el ordenador calculando el número de trienios del operario, para ello compara (línea 007070) si el contenido del campo MES (descrito en la línea 003070) donde se encuentra el valor del mes de la fecha de ingreso del operario, es decir, 05, es menor que el contenido del campo M (descrito en la línea 005150), en el cual introdujo el operador al comienzo del proceso, por la máquina de escribir de la consola, el valor 04.

Como en toda comparación, el ordenador se encuentra ante la elección entre dos caminos por los que dirigirse, y lo hará por uno u otro, según el resultado de la comparación. Como el resultado ha sido que 05 $\not<$ 04, elegirá el camino con el que calcular el número de trienios por la fórmula (A − D − 1) / 3, contenida en la declaración COMPUTE (línea 007090), omitiendo la ejecución de la declaración de la línea 007080; por tanto, como el año en que se ejecuta la nómina es 1978 y el año de la fecha de ingreso es 1960, el número de trienios es igual a (78 − 60 − 1) / 3 = 5, valor que introduce en el campo NUMTRIENIOS (descrito en línea 004120).

A continuación pasa a obtener el valor de los trienios, ejecutando la declaración COMPUTE de la línea 007100, para lo cual multiplica el valor contenido en NUMTRIENIOS, es decir, 05, por 1000, colocando el resultado 05000 en el campo de trabajo llamado TRIENIOS (descrito en la línea 004080).

```
007060 CALCULO-TRIENIOS.
007070     IF MES OF FECHAINGRESO < M.
007080     COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3 ELSE
007090     COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100     COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
```

Si, por el contrario, el valor del mes de la fecha de ingreso del operario hubiese sido menor que el valor del mes en que se confecciona la nómina, entonces el ordenador se hubiera dirigido por el otro camino, es decir, efectuaría el cálculo del número de trienios empleando la fórmula (A − D) / 3 al ejecutar la declaración COMPUTE (línea 007080) y, omitiendo la declaración de la línea 007090, continuaría, como en el caso anterior, a calcular el valor de los trienios.

Una vez calculado el valor de los trienios, comienza el cálculo para la obtención del plus familiar, volviendo el ordenador a encontrarse entre varios caminos a elegir, dependiendo su elección del valor que haya en el campo ESTADO (descrito en la línea 002180). Si en dicho campo encuentra la cifra 2, quiere decir, según el código establecido, que el hombre es casado y por tanto ejecuta el cálculo del valor del plus familiar para el hombre casado, aplicando la fórmula que figura en la declaración COMPUTE (línea 007120), continuando la ejecución del programa en la declaración de la línea 007180.

Si encuentra un 3, omite las declaraciones de las líneas 007120 y 007130 y ejecuta el cálculo del plus familiar para el hombre o mujer viudos, aplicando la fórmula especificada en COMPUTE de la línea 007140. Si encuentra un 1, como en el caso del operario que se está tratando, no realiza ninguna de las declaraciones de las líneas 007120 a la 007140 y pasa a ejecutar la declaración MOVE de la línea 007150, con lo cual coloca el valor 0000 en el campo de trabajo PLUSF (descrito en la línea 004090), como así debe suceder, puesto que el soltero no percibe plus familiar.

```
007110 CALCULO-PLUSF.
007120     IF CASADO COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007130     ELSE
007140         IF VIUDO COMPUTE PLUSF = NUM-HIJOS * 250
007150         ELSE MOVE 0 TO PLUSF.
007160 AVISO. NOTE LAS CUATRO DECLARACIONES SIGUIENTES CALCULA EL VALOR
007170     HORAS EXTRAS, TOTAL BRUTO, DESCUENTO (DESCU) Y TOTAL NETO.
```

A continuación, el ordenador no ejecuta acción alguna por cuanto se describe en las líneas 007160 y 007170, ya que son simples comentarios.

Por las declaraciones comprendidas de la línea 007180 a la 007210, calcula el valor de las Horas Extras, Total Bruto, Descuento y Total Neto de la forma siguiente: Toma el valor del campo NUM-HORASEX (descrito en la línea 003110) que, en ese momento, es 025 y lo multiplica por el valor de cada hora extra, 150. El resultado, 03750, lo coloca en el campo de trabajo HORASEX (línea 004100).

Para hallar el Total Bruto, suma (línea 007190) el contenido de los siguientes campos: SUELDO (descrito en la línea 003090), cuyo valor, según figura en los datos de entrada es 20000; TRIENIOS, que, según se calculó anteriormente, es 05000; PLUSF, cuyo valor es 0000, y HORASEX que es 03750; colocando el total 028750 en el campo de trabajo TOTAL BRUTO (descrito en la línea 004070).

Para hallar el Descuento, multiplica (línea 007200) el valor del campo TOTALBRUTO, que en ese momento es 028750, por el valor del campo TIPODESCUEN (descrito en la línea 003100) cuyo valor es 14 y, dividiendo el producto por 100, obtiene el resultado 4025, colocándolo en el campo de trabajo DESCU (descrito en la línea 004110).

```
007180 10. MULTIPLY NUM-HORASEX BY 150 GIVING HORASEX.
007190     COMPUTE TOTALBRUTO = SUELDO + TRIENIOS + PLUSF + HORASEX
007200     COMPUTE DESCU = (TOTALBRUTO * TIPODESCUEN) / 100
007210     SUBTRACT DESCU FROM TOTALBRUTO GIVING TOTALS.
```

Por último, para hallar el Total Neto resta (línea 007210) el contenido del campo DES CU, es decir, 4025, del contenido del campo TOTAL BRUTO, que es 028750, colocando el resultado, 024725, en el campo TOTALS (descrito en la línea 004130).

Hasta aquí, se han realizado todos los cálculos necesarios con los datos y se han guardado los resultados en campos de trabajo de la memoria (figura 14). Ahora es el momento de imprimir dichos resultados, junto con el contenido de otros campos de datos que no se modificaron durante el proceso, los cuales se encuentran en el área de entrada (CODIGO, APELL–NOMBRE, SUELDO), para lo cual el programador puede elegir entre dos opciones:

- Trasladar los citados datos y resultados desde los campos donde están colocados, al área de salida para, desde ella, dar salida a aquéllos, escribiéndolos sobre una línea de la hoja de impresora, tal como se encuentran en dichos campos, o

- Trasladarlos previamente a otra área de trabajo para proceder a su edición (forma de hacer legibles los resultados) y desde este área, moverlos al área de salida, desde la cual serán escritos.

En este programa se ha elegido la segunda opción, ya que de haber utilizado la primera, al imprimir el valor de un campo cualquiera, por ejemplo, el campo TOTALS que contiene el Total Neto, lo hubiera hecho: 024725, es decir, imprimiría el contenido del campo sin tener en cuenta las cifras enteras o decimales del mismo, lo cual, prácticamente, no tiene significado; pero, utilizando la segunda opción, la impresora escribirá: *24.725, donde se ha suprimido el cero de la izquierda, se ha insertado el asterisco, cosa muy útil para evitar falsificaciones a la vez que se han separado las centenas de las unidades de millar por un punto; cosa similar ocurre con los campos SUELDO–S, DESCUE, etc.

El área de trabajo encargada de la edición está descrita de la línea 004140 a la 005070, y en ella hay que ir introduciendo el contenido de cada uno de los campos que se quieren

```
004140     01  IMPRESO-S.
004150         02  FILLER                  PICTURE  X(15)  VALUE SPACES.
004160         02  CODIGO-S                PICTURE  X(5).
004170         02  FILLER                  PICTURE  XXXX  VALUE SPACES.
004180         02  APE-NOMBRE              PICTURE  X(35).
004190         02  FILLER                  PICTURE  X(4)  VALUE SPACES.
004200         02  SUELDO-S                PICTURE  ZZ.ZZZ.
004210         02  FILLER                  PICTURE  XXXX  VALUE SPACES.
004220         02  TRIENIO                 PICTURE  ZZ.ZZZ.
004230         02  FILLER                  PICTURE  X(4)  VALUE SPACES.
004240         02  HORASE                  PICTURE  ZZ.ZZZ.
005010         02  FILLER                  PICTURE  X(4)  VALUE SPACE.
005020         02  PLUSS                   PICTURE  Z.ZZZ.
005030         02  FILLER                  PICTURE  X(4)  VALUE ALL ' '.
005040         02  DESCUE                  PICTURE  Z.ZZZ.
005050         02  FILLER                  PICTURE  X(4)  VALUE SPACES.
005060         02  TOTAL                   PICTURE  ***.***.
005070         02  FILLER                  PICTURE  X(15)  VALUE SPACE.
```

editar, así como otros campos, como CODIGO y APELL–NOMBRE, que aunque se mueven a un área de edición, sin embargo no sufren modificación cuando se imprimen, cosa normal si se tiene en cuenta que, tanto en el área de entrada como en el área de edición, están descritos con: PICTURE X (5) y PICTURE X (35).

Las declaraciones comprendidas desde la línea 008020 a la 008090 muestran como se ejecuta uno a uno el movimiento o traslado del contenido de determinados campos al área de edición.

```
008010 ALFA-1.
008020     MOVE CODIGO      TO CODIGO-S
008030     MOVE APELL-NOMBRE TO APE-NOMBRE
008040     MOVE SUELDO      TO SUELDO-S
008050     MOVE TRIENIOS    TO TRIENIO
008060     MOVE PLUSF       TO PLUSS
008070     MOVE HORASEX     TO HORASE
008080     MOVE DESCU       TO DESCUE
008090     MOVE TOTALS      TO TOTAL, TOTAL-SUELDOCIN
```

Asi, la declaración MOVE de la línea 008020 transfiere el contenido 00001 del campo CODIGO de la ficha dentro del campo CODIGO–S del área de edición; la siguiente declaración MOVE transfiere el contenido del campo APELL–NOMBRE, es decir, BUENO VALERO ALBERTO dentro del campo APE–NOMBRE, y así sucesivamente, el resto de los campos.

En los campos denominados FILLER, además de reservarse un número determinado de posiciones de memoria, se ha introducido en ellos el carácter espacio o blanco (VALUE SPACES), a fin de intercalar espacios en blanco entre cada dos campos que contienen información de la que se va a imprimir en cada línea del impreso; es por lo que, antes de escribir el contenido del primer campo, CODIGO–S, deja catorce espacios en blanco; entre el campo CODIGO–S y APE–NOMBRE deja otros cuatro espacios en blanco, y así sucesivamente, con lo que se consigue facilitar la lectura de la nómina al usuario.

Es normal, antes de trasladar el contenido de los campos o áreas que se han de imprimir al área de salida (descrita en la línea 004040), limpiar (poner blancos) este área con objeto de evitar que si existe en ella alguna información procedente de la ejecución de programas anteriores pudiera imprimirse junto con los resultados. Hay que tener presente que el hecho de reservar un área de la memoria no lleva consigo que dicha área se limpie automáticamente de una información anterior que pueda contener, por lo que es preciso utilizar la declaración MOVE SPACES TO IMPRESO que lo realice, o bien, como se ha descrito anteriormente, llenar los campos FILLER del área de edición con espacios en blanco, por medio de la cláusula VALUE SPACES. De no haber realizado ninguna de las dos opciones descritas,

la línea de resultados podía haber estado impresa de la forma siguiente:

1/-, 0A,XFP---/00001-.$$BUENO VALERO ALBERTO -.AE20.000.-AX...

Sin embargo, en este programa, la declaración MOVE SPACES TO IMPRESO la ha utilizado el programador, no sólo para limpiar el área de salida, como se especifica en las líneas 008100 y 008110, sino también como artificio para evitar el imprimir el contenido de CABECERA-2, es decir, "NOMINA DE LA EMPRESA RICOSA...", al comienzo de cada página de la nómina.

Efectivamente, las declaraciones de las líneas 008120 a 008140 muestran lo que debe hacer la impresora cuando, al ir escribiendo líneas sobre el impreso, llega a una línea de la hoja a partir de la cual el programador no desea que siga escribiendo en ella.

```
008100     MOVE SPACES TO IMPRESO. NOTE  ESTA DECLARACION LIMPIA EL
008110           AREA DE SALIDA (PONE BLANCOS).
008120     IF CONTALINEAS = 24 WRITE IMPRESO AFTER PAG-NUEVA
008130     PERFORM ESCRIBIR-2   MOVE ZERO TO CONTALINEAS.
008140     WRITE IMPRESO FROM IMPRESO-S AFTER 2 ADD 1 TO CONTALINEAS
```

La impresora, mientras no reciba orden en contrario, va escribiendo línea tras línea sobre el papel contínuo, que está doblado en forma de acordeón; pero una nómina escrita así, resulta muy molesta de manejar y podría suceder que alguna de las líneas de resultados coincidiera con el doblez del papel contínuo. Es un inconveniente que hace necesario limitar el número de líneas a escribir en cada hoja, para lo cual es preciso indicar al ordenador lo que debe hacer la impresora cuando llegue a la línea de la hoja a partir de la cual no debe escribir más y, por tanto, salte a la página siguiente. Todo ello obliga a que el programador debe controlar a la impresora cada vez que haya de imprimir una línea sobre la hoja del impreso.

Para el control de la impresora se puede utilizar un contador denominado CONTALI NEAS (descrito en línea 004051) en el que se irán contabilizando el número de líneas de resultados, también llamadas de detalle, a medida que se van escribiendo sobre el impreso (en este programa el programador ha decidido escribir 24 líneas de resultados en cada hoja). El ordenador, antes de escribir cada línea de este tipo, consulta si el campo CONTALINEAS contiene el valor 24 (IF CONTALINEAS = 24) o, lo que es lo mismo, si ya ha escrito las 24 líneas.

Dos son los casos posibles:

- Que el valor de CONTALINEAS sea menor que 24, en este caso omite la ejecución de las declaraciones que siguen a IF CONTALINEAS = 24 hasta que encuentra el primer punto, es decir, omite la ejecución de las declaraciones WRITE IMPRESO AFTER PAG-NUEVA, PERFORM ESCRIBIR-2 y MOVE ZERO TO CONTALINEAS, pasando

a ejecutar la declaración WRITE IMPRESO FROM IMPRESO–S AFTER 2 con la que moverá los datos contenidos en el área de trabajo IMPRESO–S al área de salida IMPRESO, saltará dos líneas y escribirá una línea de resultados. Por la declaración ADD 1 TO CONTALINEAS incrementará en una unidad al campo CONTALINEAS, continuando la ejecución del programa en la declaración MOVE de la línea 008150.

— Que el valor de CONTALINEAS sea igual a 24, esto quiere decir que la impresora ya ha escrito 24 líneas de resultados en una página, por lo que pasará entonces a ejecutar la declaración WRITE IMPRESO AFTER PAG–NUEVA, la cual obligará a la impresora a situar el papel al principio de otra página y escribir una línea con el contenido existente en ese momento en el área de salida, que no es otra cosa que caracteres blancos, que se introdujeron en dicha área por la declaración MOVE SPACE TO IMPRESO; es decir, imprimirá una línea en blanco. Por tanto, con estas declaraciones se ha conseguido posicionar el papel al comienzo de una nueva página, y omitir la escritura del encabezamiento NOMINA DE LA EMPRESA..., como exigía el enunciado del problema (pág.18)

A continuación por la declaración PERFORM ESCRIBIR–2, ejecutará todas las declaraciones contenidas en el procedimiento ESCRIBIR–2 (descrito desde la línea 006140 a la 006230), que obligarán a imprimir el segundo y tercer encabezamiento, subrayando este último, y una vez ejecutado dicho procedimiento, regresa a ejecutar la declaración siguiente a PERFORM, es decir, MOVE ZERO TO CONTALINEAS, a fin de comenzar de nuevo la cuenta del número de líneas de resultados; continuando la ejecución del programa en la declaración de la línea 008140.

En el análisis del problema, se decía que la Sección de Estadística solicitaba una cinta magnética en la que se grabaran, en cada registro, todos los datos que contenía la ficha, junto con el Sueldo Neto, a fin de poder utilizarla en futuros trabajos de la Sección. Para la grabación del registro en la cinta es necesario terminar de llenar el área de salida, reservada para la grabación del archivo en cinta, con los datos de la ficha, por la declaración MOVE (línea 008150), pues el valor del Total Neto ya se introdujo en el área de salida por la declaración de la línea 008090; una vez completa este área, se da la orden de escritura (línea 008160) que hará que se grabe dicho contenido en la cinta.

La declaración ADD de la línea 008170 sirve para ir contabilizando el número de operarios que figuran en la nómina, ya que cada vez que se termine el proceso de la ficha de cada operario y se grabe sobre la cinta, se suma una unidad a un campo llamado CONTADOR (descrito en la línea 004060) cuyo valor inicial es 00000, con lo que al terminar de procesar el archivo de operarios, dentro del campo CONTADOR se tendrá un valor igual al número de registros que contiene dicho archivo.

```
008150     MOVE FICHA-OPERARIO TO DATOSCIN
008160     WRITE REGISTRO-OPERARIO
008170     ADD 1 TO CONTADOR
008180     GO TO LEER.
```

M E M O R I A

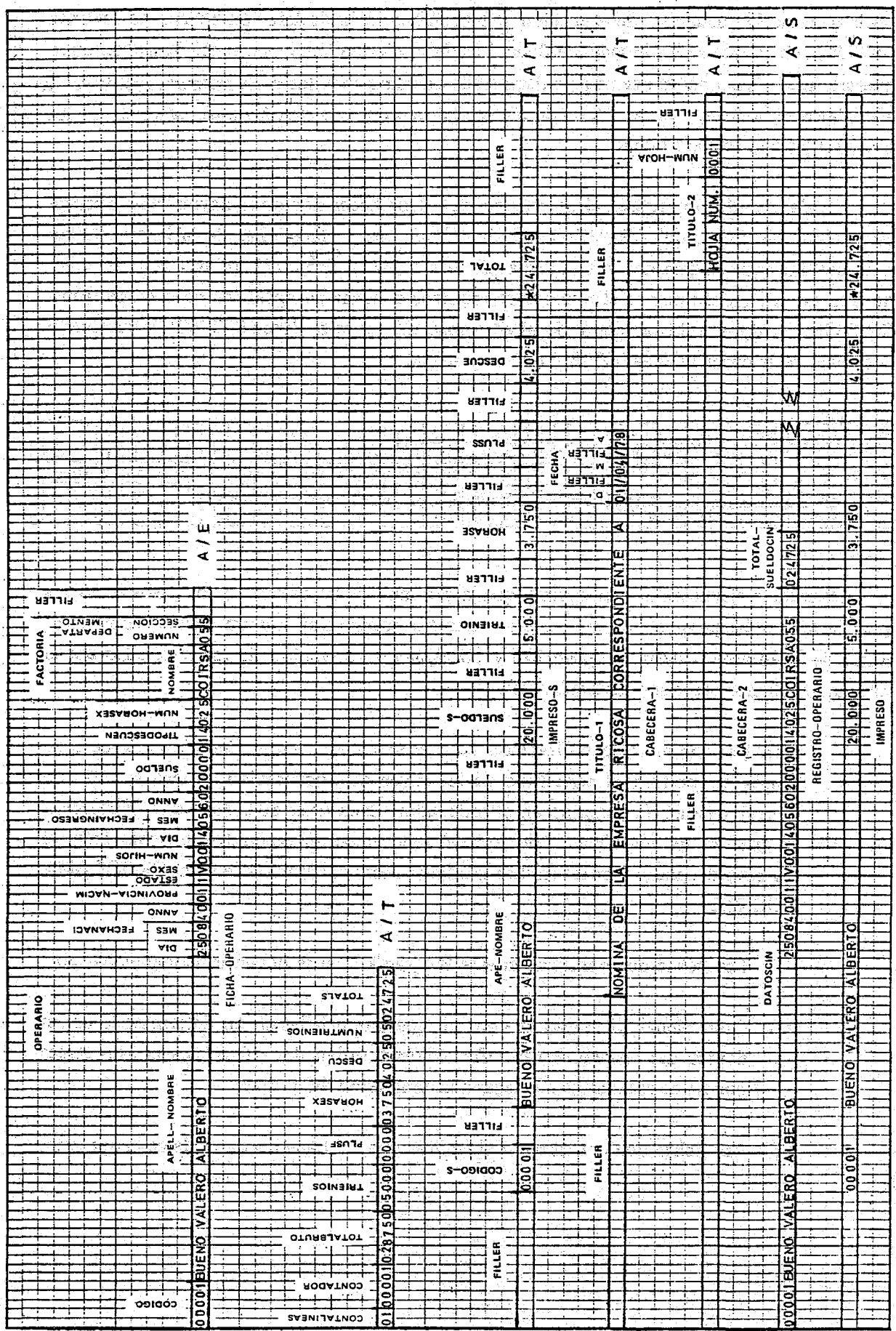


FIGURA 14

Finalizado el proceso de la primera ficha del archivo, la declaración de la línea 008180 hace regresar la secuencia de ejecución del programa a la declaración READ de la línea 007020, con lo que se lee la segunda ficha. Los datos de esta ficha, pertenecientes al segundo operario, pasan a ocupar la misma área de entrada que ocupaban los de la primera, sustituyendo a los datos existentes; comenzando de nuevo el proceso para seguir los mismos pasos que los descritos para la primera ficha.

Queda por último, indicar al ordenador qué debe hacer cuando se hayan leído y procesado todas las fichas del ARCHIVO–OPERARIOS. Las declaraciones de la línea 007030 a la 007050, indican que, una vez acabadas de leer todas las fichas del archivo, se le da orden de que escriba en el impreso el número de registros o fichas de operarios procesados (línea 007030), cierre los archivos que abrió al comienzo del proceso (línea 007050), y terminado el trabajo, se le ordena (línea 007050) que se detenga.

```
007010 LEER.
007020     READ ARCHIVO-OPERARIOS; AT END,
007030     DISPLAY '                REGISTROS PROCESADOS = ' CONTADOR,
007040     CLOSE   ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050     STOP RUN.
```

Descrita la ejecución del programa, queda por indicar (figura 15) los pasos sucesivos por los que debe pasar el mismo , desde que fue escrito por el programador sobre las Hojas de Codificación, hasta su ejecución y consiguiente obtención de los resultados por el ordenador; dichos pasos son los siguientes:

– Con las máquinas de perforación de que se dispone en el Centro de Proceso de Datos, se perfora el contenido de las Hojas de Codificación en fichas, a fin de introducir dicho contenido en el ordenador. Cada línea de la Hoja de Codificación se perfora en una ficha, con lo cual tendremos un conjunto de fichas igual al número de líneas escritas en dichas hojas. A este conjunto se le llama PROGRAMA FUENTE.

  Las fichas que constituyen este Programa Fuente deben ir acompañadas por:

  a) Otras fichas, llamadas de control, que, intercaladas en un determinado orden dentro del programa, van informando al ordenador sobre las sucesivas taréas que ha de realizar, hasta llegar a la conclusión del trabajo.

  b) Las fichas que constituyen el archivo que contiene los datos de cada operario.

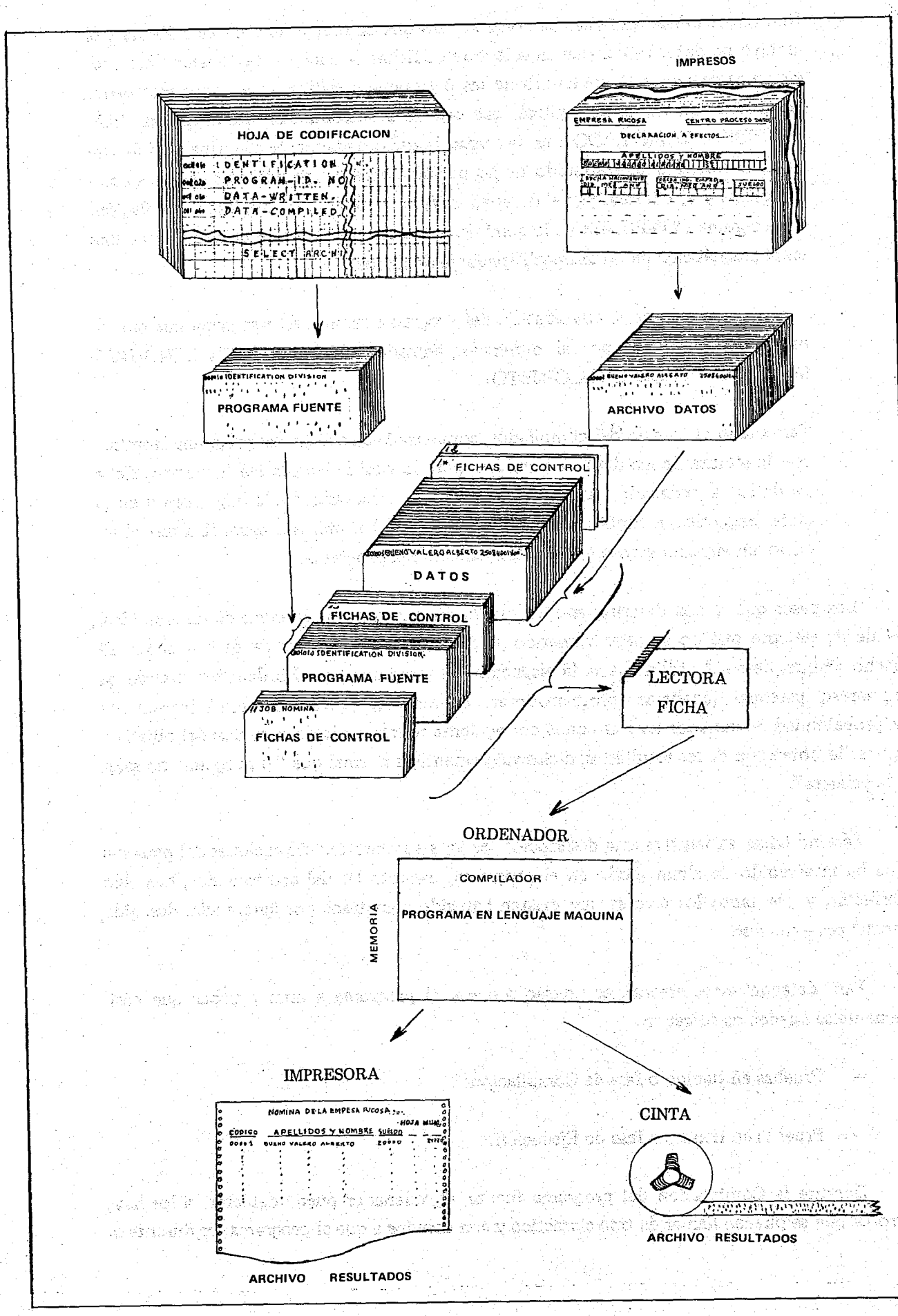

IMPRESOS
HOJA DE CODIFICACION
IDENTIFICATION
PROGRAM-I.D. NO.
DATA-WRITTEN.
DATA-COMPILED.
SELECT ARCH1
EMPRESA RICOSA
CENTRO PROCESO DATOS
DECLARACION A EFECTOS...
APELLIDOS Y NOMBRE
PROGRAMA FUENTE
ARCHIVO DATOS
FICHAS DE CONTROL
DATOS
FICHAS DE CONTROL
PROGRAMA FUENTE
FICHAS DE CONTROL
LECTORA FICHA
ORDENADOR
COMPILADOR
PROGRAMA EN LENGUAJE MAQUINA
MEMORIA
IMPRESORA
CINTA
NOMINA DE LA EMPRESA RICOSA
CODIGO
APELLIDOS Y NOMBRE
SUELDO
ARCHIVO RESULTADOS
ARCHIVO RESULTADOS

FIGURA 15

- Una vez reunidas las fichas de control, las que contienen el programa fuente y el archivo de datos, se colocan en la lectora de fichas, la cual, y antes de leer e introducir en la memoria el contenido de las fichas que constituyen el programa fuente, lee una de las fichas de control, que ordena la introducción del programa TRADUCTOR o COMPILADOR en la memoria del ordenador con la finalidad de traducir o convertir el contenido de las perforaciones de las fichas en órdenes comprensibles y ejecutables por el ordenador. A este trabajo se denomina Compilación. El programa COMPILADOR lo suministra la casa constructora grabado sobre una cinta magnética o en un discpack (paquete de discos).

  El resultado de la compilación del programa fuente es otro programa que ya puede ser ejecutado por el ordenador, llamado PROGRAMA EN LENGUAJE MAQUINA o PROGRAMA OBJETO.

- Terminado el trabajo del compilador, comienza la ejecución del programa Nómina con la lectura de los datos del archivo, para lo cual la lectora lee la primera ficha de datos, el ordenador la procesa y escribe los resultados por la impresora y en la cinta magnética; a continuación lee la segunda ficha, etc. Así hasta finalizar el archivo, en cuyo momento termina la ejecución del programa.

Los pasos que se han descrito son los que ha de recorrer un programa en un caso ideal, es decir, siempre que no se haya cometido ningún error en cualquiera de dichos pasos. El hecho de que, desde el análisis hasta la ejecución del programa por el ordenador, intervenga numeroso personal (analistas, programadores, operadores, perforistas, etc.) incrementa la probabilidad de cometer errores, con el consiguiente retraso en la consecución del objetivo, que es la obtención de los resultados; dicho vulgarmente, es normal que "el programa no salga a la primera".

Para no hacer exhaustiva esta descripción, se va a suponer que las acciones del personal que ha intervenido de algún modo en el programa, excepto las del programador, han sido perfectas, y, por tanto, los errores que existen han sido cometidos por ignorancia, descuido, etc, del programador.

Para detectar estos errores, es preciso someter al programa a unas pruebas que básicamente, se pueden clasificar en:

- Pruebas en tiempo o fase de Compilación.

- Pruebas en tiempo o fase de Ejecución.

Durante la Compilación del programa fuente, el ordenador puede detectar, si los hay, errores que se pueden llamar de tipo sintáctico y son debidos a que el programador, durante la

la escritura del programa, ha hecho caso omiso de alguna de las reglas de las que se especifican en el manual del lenguaje; por ejemplo: escribir MOBE en vez de MOVE, no escribir la palabra DATA DIVISION etc. Cuando esto sucede, el ordenador envía a través de la impresora un listado de errores, que informa sobre cuál es el motivo de cada error, en qué ficha del programa fuente se ha cometido y la mayor o menor importancia del error.

El programador debe corregirlos, ya que, en caso contrario, si los errores son de determinada importancia, el ordenador por sí mismo se detiene y, por tanto, no pasa a la fase de Ejecución o, de pasar, porque los errores son de poca importancia, puede llevar consigo el que los resultados obtenidos no sean fiables.

Como norma general, la compilación se repetirá, cuantas veces sean necesarias, hasta la total eliminación de todo tipo de error que pueda influir en los resultados finales.

A continuación, comienzan las pruebas en tiempo o fase de Ejecución, pudiendo aparecer otro tipo de errores, que, como sucedía en la fase de Compilación, pueden hacer que el ordenador interrumpa la ejecución del trabajo, así por ejemplo: si por olvido, el programador hubiera omitido la declaración de la línea 008180, GO TO LEER, el ordenador habría ejecutado la declaración de la línea 008170 y en ese momento se detendría por no saber qué camino seguir. A este tipo de errores se les llama lógicos. Al igual que en la fase de compilación hay que eliminar cuantos errores existan.

Una vez eliminados los errores sintácticos y lógicos, ya es posible llevar a cabo la ejecución del programa y, por tanto, llegar a la obtención de unos resultados; pero hay que tener presente que es posible que éstos pueden ser también falsos; efectivamente, si el campo CONTADOR (línea 004060), el programador lo hubiera inicializado con un valor distinto de cero y, por la declaración ADD 1 TO CONTADOR ir incrementándole en una unidad, cuando llegara al final del programa y mostrara el ordenador el número de registros procesados, éste no sería correcto.

Para evitar estos inconvenientes, es muy importante tener en cuenta que antes de llevar a cabo el trabajo con el archivo de datos, se debe probar el programa ejecutándolo con una serie de fichas de datos, que contemplen todos los casos posibles que se pueden dar en los operarios (que sea soltero o casado, con hijos o sin ellos, etc.) a fin de que el ordenador pueda recorrer todas las declaraciones del programa, y, una vez ejecutado, poder comprobar si los resultados que obtiene el ordenador son los correctos, para lo cual se recomienda calcular manualmente los resultados con aquella serie de fichas de datos y ver si coinciden con los obtenidos por el ordenador. En caso afirmativo, se puede confiar al ordenador la ejecución del trabajo y en caso negativo corregir el error o errores que impiden la obtención de resultados correctos, y volver a ejecutar el programa.

Las figuras 17 y 18 muestran los resultados obtenidos sobre el impreso al utilizar como datos de prueba los que se indican en la figura 16, en la que cada línea representa el contenido de una ficha.

DATOS DE PRUEBA

OPERARIO															FACTORIA		
CODIGO	APELLIDOS Y NOMBRE	fecha nacimiento			prov. naci.	estado	sexo	núm. hijos	fecha de ingreso			SUELDO	tipo desc.	núm. horas extras	NOMBRE	dep.	
		día	mes	año					día	mes	año					núm.	sec.
00001	BUENO VALERO ALBERTO	25	08	40	11	V	0	01	14	05	60	20000	14	025	COIRSA	055	
00002	GARCIA GOMEZ MARTA	12	07	45	04	3	H	02	12	06	63	25000	14	030	COIRSA	043	
00003	LOPEZ SANCHO ANSELMO	10	12	39	09	3	V	04	05	07	60	38000	02	100	ROCISA	102	
00005	FERNANDEZ GUTIERREZ SONIA	01	05	42	12	1	H	00	08	09	63	12000	07	125	IRCOSA	121	
00010	SANCHEZ GONZALEZ JOSE	04	09	43	48	3	V	04	09	11	66	14000	02	010	CIORSA	134	
00025	SERRANO SANTIESTEBAN ROBERTO	07	06	43	50	2	V	05	15	12	67	29000	02	070	CRISA	103	
00028	GOMEZ DE LA IGLESIA MARIA	10	10	43	29	3	H	06	25	12	58	30000	02	040	IRCOSA	112	
00080	LOPEZ VILAPLANA GUSTAVO	10	08	43	25	3	V	07	24	10	67	28000	02	043	ROCISA	121	
00125	VILAGRA RUIPEREZ MERCEDES	15	07	50	38	1	H	00	23	11	70	19000	02	027	COIRSA	136	
00182	FERNANDEZ PONCE DE LEON EUSEBIO	19	09	38	44	1	V	00	19	09	55	15000	02	025	OCIRSA	092	
00320	GONZALEZ BARRADA ANTONIA	10	12	41	43	3	H	03	18	08	63	18000	07	023	ROCISA	083	
00415	RIOBO MATEO CARLOS	27	11	35	44	3	V	01	17	07	57	25000	14	021	OCRISA	072	
00585	FREIRE PEREZ ENCARNACION	24	10	43	45	1	H	00	16	08	71	40000	14	024	IRCOSA	062	
00890	GALACHE VILLATORO JOSE	20	09	44	42	V	00	15	06	64	18	00007	06	0CI	ORSA	054	
00995	GALAN MENDICUTI ROSA MARIA	18	08	45	07	3	H	07	14	05	66	39000	02	065	CIORSA	084	
01000	IRAYZOZ BLASCO JOAQUIN	14	07	42	04	1	V	00	18	04	64	45000	14	066	IRCOSA	091	
01255	ESCOBAR GIMENO NIEVES	12	06	41	08	1	H	00	05	03	61	39000	14	044	OCRISA	105	
01347	ESPINOSA MARTINEZ ROBERTO	10	05	40	93	V	05	03	08	58	60	00002	09	2RO	CISA	125	
01458	ECHEVARRIA ALVAREZ EUSTAQUIO	09	04	43	14	1	V	00	08	10	65	40000	02	054	OCIRSA	135	
01983	RODRIGJEZ SANPEDRO MIGUEL	05	03	42	15	2	V	01	13	11	65	33000	14	027	ROCISA	043	
01995	DOMINGUEZ DE LA GUARDIA FERNANDA	03	02	45	18	3	H	00	09	12	66	17500	07	029	OCIRSA	082	
02100	CABRERA BOSQUE NIEVES	02	01	43	12	1	H	00	06	10	67	19800	07	030	IRCOSA	096	
02325	SOBRINO RUIZ ANDRES	09	11	47	29	2	V	05	08	07	68	42000	02	033	COIRSA	086	
02326	ORBANANOS OZORES ENRIQUE	14	12	43	03	V	03	15	04	68	25	00014	04	4RO	CISA	061	
02428	CASTILLA VALERO ANTONIO	13	10	50	30	1	V	00	25	05	68	37	00002	061	CIORSA	041	
02580	MATEO REAL RAFAEL	10	08	33	10	1	V	00	30	12	55	25	02514	037	OCIRSA	041	
03000	PEREZ BONIFILIS PEDRO	09	03	37	15	2	V	01	28	07	56	12	00007	044	OCIRSA	072	
03100	LIZ RODRIGUEZ ALFONSO	07	02	36	25	3	V	02	24	06	55	11	00007	068	ROCISA	094	
03250	PEMAN CONDE ROSARIO	11	01	42	26	1	H	00	15	05	63	10	00002	100	ROCISA	105	
03258	GARCIA SAEZ BLANCA	12	04	45	28	0	H	13	10	04	67	13500	02	094	IRCOSA	116	
03329	ROJO GIL LUIS	25	12	48	24	3	V	09	07	03	68	13250	02	032	CIORSA	137	
04212	HERNANDEZ SUAREZ ANGEL	28	11	50	25	3	H	14	08	02	70	14720	02	000	ROCISA	044	
04515	MARTINEZ RUEDA FELIX	24	10	52	15	2	V	01	04	01	70	20000	07	000	OCIRSA	063	
04516	GARCIA SAN MARTIN MARIA	29	09	29	10	3	H	07	19	02	49	12732	02	000	CIORSA	082	
05001	AGUIRRE BILBAO PAULA	30	08	28	07	3	H	08	20	08	50	13450	02	003	CIORSA	101	
05150	RUENES SANCHEZ MARIANO	22	07	25	12	1	V	00	21	06	50	12700	02	001	COIRSA	125	
05225	TRUCHARTE ALBAJARA IGNACIO	14	06	35	14	1	V	00	25	04	56	25000	02	002	OCIRSA	014	
05422	ALONSO LAGUNA ERNESTO	15	05	24	01	1	V	00	24	03	64	13000	07	003	ROCISA	031	
05750	FERNANDEZ MEJIAS JOSEFA	18	04	36	01	3	H	00	23	01	56	20000	14	007	COIRSA	056	
06100	BLASCO SANZ GUILLERMO	06	03	40	43	V	07	22	09	60	18	00002	010	CIORSA	072		
06220	REYES BORREGO MANUEL	02	02	43	06	2	V	01	14	10	63	19000	07	105	ROCISA	091	
06222	ALARCON RUIZ MERCEDES	08	01	43	09	1	H	00	18	11	62	25075	14	004	IRCOSA	114	
06666	TENA ALEGRE FRANCISCO	07	09	43	08	3	V	02	20	12	61	24000	14	008	IRCOSA	133	
06777	PORRAS DE CALATRAVA GONZALO	08	07	38	42	V	03	27	07	59	22	05514	006	OCRISA	042		

FIGURA 16

NOMINA DE LA EMPRESA RICOSA CORRESPONDIENTE A 01/04/78

HOJA NUM. 0001

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX.	PLUS-F.	DESCUEN.	TOTAL
00001	BUENO VALERO ALBERTO	20.000	5.000	3.750		4.025	*24.725
00002	GARCIA GOMEZ MARTA	25.000	4.000	4.500	500	4.760	*29.240
00003	LOPEZ SANCHO ANSELMO	38.000	5.000	15.000	1.000	1.180	*57.820
00005	FERNANDEZ GUTIERREZ SONIA	12.000	4.000	18.750		2.432	*32.318
00010	SANCHEZ GONZALEZ JOSE	14.000	3.000	1.500	1.000	390	*19.110
00025	SERRANO SANTIESTEBAN ROBERTO	29.000	3.000	10.500	1.750	885	*43.365
00028	GOMEZ DE LA IGLESIA MARIA	30.000	6.000	6.000	1.500	870	*42.630
00080	LOPEZ VILAPLANA GUSTAVO	28.000	3.000	6.450	1.750	784	*38.416
00125	VILAGRA RUIPEREZ MERCEDES	19.000	2.000	4.050		501	*24.549
00182	FERNANDEZ PONCE DE LEON EUSEBIO	15.000	7.000	3.750		515	*25.235
00320	GONZALEZ BARRADA ANTONIA	18.000	4.000	3.450	750	1.834	*24.366
00415	RIOBO MATEO CARLOS	25.000	6.000	3.150	250	4.816	*29.584
00585	FREIRE PEREZ ENCARNACION	40.000	2.000	3.600		6.384	*39.216
00890	GALACHE VILLATORO JOSE	18.000	4.000	9.000	500	2.205	*29.295
00995	GALAN MENDICUTI ROSA MARIA	39.000	3.000	9.750	1.750	1.070	*52.430
01000	IRAYZOZ BLASCO JOAQUIN	45.000	4.000	9.900		8.246	*50.654
01255	ESCOBAR GIMENO NIEVES	39.000	5.000	6.600		7.084	*43.516
01347	ESPINOSA MARTINEZ ROBERTO	60.000	6.000	13.800	1.250	1.621	*79.429
01458	ECHEVARRIA ALVAREZ EUSTAQUIO	40.000	4.000	8.100		1.042	*51.058
01983	RODRIGUEZ SANPEDRO MIGUEL	33.000	4.000	4.050	750	5.852	*35.948
01995	DOMINGUEZ DE LA GUARDIA FERNANDA	17.500	3.000	4.350		1.739	*23.111
02100	CABRERA BOSQUE NIEVES	19.800	3.000	4.500		1.911	*25.389
02325	SOBRINO RUIZ ANDRES	42.000	3.000	4.950	1.750	1.034	*50.666
02326	ORBANANOS OZORES ENRIQUE	25.000	3.000	6.600	750	4.949	*30.401

FIGURA 17

HOJA NUM. 0002

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX.	PLUS-F.	DESCUEN.	TOTAL
02428	CASTILLA VALERO ANTONIO	37.000	3.000	9.150		983	*48.167
02580	MATEO REAL RAFAEL	25.025	7.000	5.550		5.260	*32.315
03000	PEREZ BONIFILIS PEDRO	12.000	7.000	6.600	750	1.844	*24.506
03100	LIZ RODRIGUEZ ALFONSO	11.000	7.000	10.200	500	2.009	*26.691
03250	PEMAN CONDE ROSARIO	10.000	4.000	15.000		580	*28.420
03258	GARCIA SAEZ BLANCA	13.500	3.000	14.100		612	*29.988
03329	ROJO GIL LUIS	13.250	3.000	4.800	2.250	466	*22.834
04212	HERNANDEZ SUAREZ ANGEL	14.720	2.000		3.500	404	*19.816
04515	MARTINEZ RUEDA FELIX	20.000	2.000		750	1.592	*21.158
04515	GARCIA SAN MARTIN MARIA	12.732	9.000		1.750	469	*23.013
05001	AGUIRRE BILBAO PAULA	13.450	9.000	450	2.000	498	*24.402
05150	RUENES SANCHEZ MARIANO	12.700	9.000	150		437	*21.413
05225	TRUCHARTE ALBAJARA IGNACIO	25.000	7.000	300		646	*31.654
05422	ALONSO LAGUNA ERNESTO	13.000	4.000	450		1.221	*16.229
05750	FERNANDEZ MEJIAS JOSEFA	20.000	7.000	1.050		3.927	*24.123
06100	BLASCO SANZ GUILLERMO	18.000	5.000	1.500	1.750	525	*25.725
06220	REYES BORREGO MANUEL	19.000	4.000	15.750	750	2.765	*36.735
06222	ALARCON RUIZ MERCEDES	25.075	5.000	600		4.294	*26.381
06665	TENA ALEGRE FRANCISCO	24.000	5.000	1.200	500	4.298	*26.402
06777	PORRAS DE CALATRAVA GONZALO	22.055	6.000	900	1.250	4.228	*25.977

REGISTROS PROCESADOS = 00044

FIGURA 18

3.-EJERCICIOS

1.— ¿Cuál es el motivo principal por el que es preferible el empleo de la cinta magnética en vez de la ficha, para ser utilizada como soporte de entrada de datos al ordenador?

A.— Por ser la cinta más barata que las fichas.
B.— Por ocupar menos espacio el archivo en cinta que en fichas.
C.— Por cometerse menos errores en la lectura de datos contenidos en cinta que en ficha.
D.— Por ser más rápida la lectura de datos de una cinta que de una ficha.

2.— El nombre del programa, que sirve para que el compilador pueda identificar el programa, se describe dentro de la:

A.— IDENTIFICATION DIVISION
B.— ENVIRONMENT DIVISION
C.— DATA DIVISION
D.— PROCEDURE DIVISION

3.— Si en la ejecución de un trabajo interviniesen dos archivos de entrada y uno de salida, se necesitarían como mínimo:

A.— Dos áreas de salida y una de entrada
B.— Un área de salida y dos de entrada
C.— Dos áreas de entrada y dos de salida
D.— Un área de entrada y otra de salida.

4.— La declaración que prepara los archivos para ser leídos es la:

A.— READ
B.— DISPLAY
C.— OPEN
D.— EXHIBIT

5.— La declaración que detiene momentáneamente el ordenador, mientras el operador ejecuta una operación manual, es:

A.— STOP RUN
B.— STOP
C.— ACCEPT
D.— Las soluciones B.- y C.-

6.— A la vista de la figura5 ¿cuál es el contenido de las columnas 66 a 68, y a qué concepto se refieren ?

7.— El orden en que hay que especificar dentro de un programa las cuatro DIVISIONES que lo constituyen, es:

A.— IDENTIFICATION, DATA, ENVIRONMENT, PROCEDURE.
B.— PROCEDURE, IDENTIFICATION, DATA, ENVIRONMENT.
C.— IDENTIFICATION, ENVIRONMENT, DATA, PROCEDURE.
D.— Cualquiera de las tres soluciones, es correcta, ya que se pueden especificar en cualquier orden.

8.— En la ejecución del programa Nómina se han empleado:

A.— Dos archivos de entrada y uno de salida.
B.— Dos de salida y uno de entrada.
C.— Uno de entrada y uno de salida.
D.— Dos de entrada y dos de salida.

9.— ¿Cuál es el nombre con que el programador ha denominado, en el programa NOMINA, a los registros del archivo de entrada?

A.— ARCHIVO—OPERARIOS
B.— CINTA—NOMINA
C.— REGISTRO—OPERARIO
D.— FICHA—OPERARIO

10.— El programador, además de poder reservar posiciones de la memoria por medio de las descripciones que haga en la DATA DIVISION para posteriormente introducir los datos, puede también, antes de llevar a cabo la ejecución del programa:

A.— Inicializar algunos campos con ceros.
B.— Inicializar algunos campos con espacios en blanco.
C.— Inicializar con un determinado contenido.
D.— Cualquier de las tres soluciones anteriores es correcta.

11.— Si se desea que el ordenador escriba sobre la hoja de papel de la impresora, hay que especificar en el programa la declaración:

A.— ESCRIBIR—1
B.— WRITE
C.— DISPLAY
D.— Cualquiera de las soluciones A.- o B.-

12.— La máquina de escribir de la consola permite,entre otras cosas:

A.— Recibir mensajes del ordenador
B.— Enviar mensajes al ordenador.
C.— Introducir datos y recibir resultados al / desde el ordenador.
D.— Cualquiera de las soluciones A.- B.- y C.-

13.— En el programa NOMINA, si el mes de la fecha de ingreso del operario en la empresa fuera igual al mes en que se ejecuta la nómina¿qué fórmula emplearía el ordenador para calcular el número de trienios del operario?

14.— La declaración MOVE puede transferir el contenido de un campo de datos de la memoria:

A.— Solamente a otro campo.
B.— A varios campos a la vez
C.— Unicamente al área de salida
D.— Ninguna de las soluciones es correcta.

15.— En el programa NOMINA, antes de grabarse un registro en la cinta, se llena el área de salida correspondiente al archivo CINTA—NOMINA, con el contenido de la FICHA—OPERARIO más el SUELDO NETO ¿Qué declaraciones realizan dicha función?

16.— ¿Qué declaración sirve para indicar al ordenador que el trabajo ha concluído?

A.— FIN
B.— STOP
C.— STOP RUN
D.— END

17.— Si en el programa NOMINA, el campo CONTALINEAS (línea 004051) se le hubiera inicializado con el valor 01, ¿cuántas líneas de resultados habría escrito la impresora en la primera página? ¿ cuántas en las páginas siguientes?

18.— Durante la ejecución del programa NOMINA, al comienzo de todas las páginas de la nómina aparecerá el encabezamiento "NOMINA DE LA EMPRESA RICOSA ... "

CIERTO FALSO

19.— ¿Qué declaración hace que se introduzca un registro de datos en el área de entrada?

A.— MOVE
B.— OPEN
C.— READ
D.— Ninguna de las anteriores.

20.— La DIVISION donde se describe el proceso a realizar con los datos es la:

A.— IDENTIFICATION
B.— ENVIRONMENT
C.— DATA
D.— PROCEDURE

21.— ¿Cuántos octetos ha reservado el programador, en el programa NOMINA, para el área de trabajo CABECERA—2?

A.— 80
B.— 114
C.— 133
D.— 860

22.— A la vista de la figura 2 de la página 15, además del nombre del operario, decir tres nombres de conceptos en los que no puede emplearse la codificación.

23.— ¿ Qué información contiene el campo TRIENIOS, antes de comenzar la ejecución del programa con los datos del ARCHIVO—OPERARIOS ?

A.— Ceros
B.— Blancos
C.— No contiene ninguna información
D.— La que se halle escrita procedente de la ejecución de programas anteriores.

CAPITULO II

ESTRUCTURA DEL LENGUAJE

1.- ELEMENTOS DEL LENGUAJE COBOL

El lenguaje COBOL está basado en el idioma inglés, utiliza un juego básico de palabras inglesas y símbolos aritméticos convencionales, e incluye reglas sintácticas de dicho idioma.

La partícula más elemental del lenguaje es el CARACTER. La combinación de caracteres forma las PALABRAS, NUMEROS, etc., siendo la síntaxis específica del lenguaje la que al enlazar estos elementos compone las frases e inscripciones que constituyen las órdenes para el gobierno del ordenador.

A continuación se describe cada elemento que forma parte del lenguaje: CARACTERES, PALABRAS, etc., así como las reglas que gobiernan su formación, dejando para capítulos posteriores la enunciación de reglas sintácticas que rigen la formación de las frases.

1.1.- CARACTERES

El Lenguaje Cobol dispone del siguiente Juego de Caracteres:

CARACTERES NUMERICOS.

Números del 0 al 9

CARACTERES ALFABETICOS.

Letras de la A a la Z

CARACTERES ESPECIALES.

Carácter	Significado
(	Paréntesis izquierdo
)	Paréntesis derecho
>	Mayor que
<	Menor que
=	Igual
,	Coma
;	Punto y coma
.	Punto o coma decimal
'	Apóstrofo
$	Signo de dólar
	Blanco o espacio
+	Signo más
–	Signo menos
*	Asterisco
/	Barra

Para distinguir el carácter alfabético O del carácter numérico cero, se conviene en cruzar con una barra uno cualquiera de los dos, debiendo indicarse en la hoja de codificación a cuál de los dos caracteres corresponde: normalmente es el cero.

En total son 51 caracteres, que se descomponen en: 10 números, 26 letras y 15 caracteres especiales.

Los caracteres especiales se utilizan en los siguientes casos:

CARACTERES EMPLEADOS EN PUNTUACION.

Carácter	Significado
(	Paréntesis Izquierdo
)	Paréntesis Derecho
,	Coma
'	Apóstrofo
.	Punto
;	Punto y coma

CARACTERES EMPLEADOS EN EXPRESIONES ARITMETICAS.

Carácter	Significado
+	Adición
–	Substracción
*	Multiplicación
/	División
**	Potenciación
=	Igual
( )	

CARACTERES EMPLEADOS EN DECLARACIONES CONDICIONALES.

Carácter	Significado
>	Mayor que
<	Menor que
=	Igual a

1.2.- PALABRAS COBOL

Las palabras COBOL están constituidas por una combinación de caracteres comprendidos entre los treinta y siete siguientes:

Letras de la A a la Z
Dígitos del 0 al 9
Carácter – (guión)

Todas las palabras se agrupan en:

– NOMBRES CREADOS POR EL PROGRAMADOR

– PALABRAS RESERVADAS

```
003160                  04 SECCION                  PICTURE 9.
003170              02  FILLER                      PICTURE X(3).
003180          FD  CINTA-NOMINA        RECORDING F
003190                  BLOCK  10  RECORDS
```

SECCION, FILLER, PICTURE, FD, CINTA–NOMINA, RECORDING, etc., son palabras COBOL.

1.2.1.- NOMBRES CREADOS

Los inventa el programador, y debe definirlos dentro de determinada parte del programa para poder ser reconocidos por el programa compilador, ya que no tienen un sentido predeterminado.

La definición o descripción de nombres creados consiste normalmente en dar nombres a los campos de la memoria y especificar el número de octetos a reservar para cada campo donde se colocarán los datos.

EJEMPLO:

```
003040          03  NUM-HIJOS                  PICTURE 99.
003050          03  FECHAINGRESO.

004070    77    TOTALBRUTO                    PICTURE 9(6).
004080    77    TRIENIOS                      PICTURE 9(5).
004090    77    PLUSF                         PICTURE 9(4).
```

TRIENIOS es un nombre creado, para el cual el compilador reservará un campo de la memoria de cinco octetos.

Todo nombre creado al que se haga mención dentro de la PROCEDURE DIVISION para ser utilizado durante el proceso, debe estar previamente definido en la ENVIRONMENT o DATA DIVISION, ya que de no hacerlo así, impide que pueda llevarse a cabo la ejecución del programa por no poder reconocer el programa compilador a qué campo de datos es al que se hace referencia:

```
007110  CALCULO-PLUSF.
007120      IF  CASADO    COMPUTE  PLUSF  = (500 + NUM-HIJOS * 250)
007130      ELSE
007140          IF  VIUDO COMPUTE  PLUSF  = NUM-HIJOS * 250
007150              ELSE MOVE 0 TO PLUSF.
```

así, PLUSF (línea 004090) es el nombre con que ha definido el programador al campo de la memoria que va a ser utilizado durante el proceso para contener el valor del plus familiar de cada operario, y para el cual se han reservado cuatro octetos. Lo mismo ocurre en el campo NUM–HIJOS (línea 003040) al que se le han reservado dos octetos y contendrá el número de hijos de cada operario.

REGLAS DE FORMACION:

- Pueden tener un máximo de 30 caracteres.
- No deben comenzar o terminar con un guión, pero puede haber varios guiones en cualquier otra parte del nombre, aunque sean consecutivos.

- Ningún nombre creado puede escribirse igual que una palabra reservada.

- No puede haber espacios en blanco dentro de un nombre creado, ni tampoco el punto, la coma o punto y coma, ya que indican el final de dicho nombre.

Existen diversos tipos de nombres creados, denominándose de diferente forma según en la parte del programa donde se describan, y siguen para su formación, además de las reglas citadas, la de contener como mínimo un carácter alfabético:

- NOMBRES DE DATOS. Se refiere a los nombres de los campos que forman parte de un registro o a campos de trabajo.

  Se integran en este grupo los: Nombres de archivo
  Nombres de registro
  Nombres de informe.

- NOMBRES DE CONDICION. Son aquéllos a los que se ha asociado valores particulares dentro de los posibles que se pueden dar a un campo de datos.

```
002180     03 ESTADO                          PICTURE 9.
002190         88  SOLTERO    VALUE 1.
002200         88  CASADO     VALUE 2.
002210         88  VIUDO      VALUE 3.
```

SOLTERO, CASADO y VIUDO son nombres particulares del campo ESTADO a los que se les asocia el valor 1, 2 y 3 respectivamente, de entre los 10 posibles.

- NOMBRES ESPECIALES, también llamados nombres NEMOTECNICOS, se especifican en la ENVIRONMENT DIVISION dentro del párrafo SPECIAL-NAMES, y quedan definidos por el hecho de asociarlos a un nombre de función (ver pág. 83).

```
001150     SPECIAL-NAMES.
001160         C01  IS  PAG-NUEVA
001170         DECIMAL-POINT  IS  COMMA.
```

PAG-NUEVA es un nombre especial o nemotécnico y C01 es un nombre de función.

Se exceptúan de la regla anterior los NOMBRES de PROCEDIMIENTO, ya que pueden estar compuestos únicamente por caracteres numéricos. Se especifican dentro de la PROCEDURE DIVISION.

```
007180 10.  MULTIPLY  NUM-HORASEX  BY  150  GIVING  HORASEX,
007190         COMPUTE TOTALBRUTO = SUELDO + TRIENIOS + PLUSF + HORASEX

008010 ALFA-1.
008020         MOVE CODIGO TO CODIGO-S
```

10 y ALFA-1 son nombres de procedimiento.

1.2.2.- PALABRAS RESERVADAS

Son aquéllas que tienen un significado predeterminado, es decir, puede reconocerlas e interpretarlas el programa traductor o compilador sin que sea preciso definirlas en el programa, como sucedía con los nombres creados.

El programador deberá tener en cuenta que:

- No puede añadir palabras reservadas a la lista ya establecida (ver APENDICE B).
- Debe escribirlas exactamente igual que figuran en la lista.

EJEMPLO:

```
Página  Línea  A  B
1  3    4  6   7  8  12  15  20  25  30  35  40  45  50  55  60
0040 10          FD   ARCHI-IMPRESO   RECORDING IS F
0040 20               LABEL RECORDS ARE OMITTED
0040 30               DATA  RECORD  IS IMPRESO.
0040 40          01   IMPRESO                  PICTURE X(1331).
0040 50     WORKING-STORAGE SECTION.
```

RECORDING, IS, ARE, LABEL, RECORD, OMITTED, DATA, PICTURE, WORKING-STORAGE, SECTION son Palabras Reservadas.

1.3.- CONSTANTES

Los nombres creados necesitan la previa definición en alguna parte del programa (ENVIRONMENT, DATA) para ser reconocidos por el compilador; sin embargo, hay veces que se utilizan datos que no es necesario definirlos, sino que por la escritura del dato mismo en el programa (PROCEDURE) queda automáticamente definido; a este tipo de datos se denomina constantes.

Se pueden clasificar las constantes en dos tipos:

- LITERALES
- CONSTANTES FIGURATIVAS.

A su vez los literales pueden ser de dos tipos:

- NUMERICOS
- NO NUMERICOS

1.3.1.- LITERALES NUMERICOS

El ordenador para realizar las operaciones aritméticas emplea números representados en cualquiera de los dos formatos: coma fija o coma flotante. El primer formato lo utiliza en aplicaciones comerciales y el segundo en aplicaciones científicas.

En el formato de coma fija cada número se representa mediante un solo conjunto de dígitos (es la representación utilizada, en la vida diaria, para expresar cualquier número), por ejemplo:12,750.

En el formato de coma flotante cada número se representa por un par de numerales (mantisa y exponente), y su valor equivale a multiplicar uno de esos numerales (mantisa) por 10 elevado al valor del otro numeral (exponente). El número decimal 12,745 podía representarse de las formas siguientes (2):

Mantisa Exponente

1.2745 E+ 01 .12745 E 02 1274.5 E – 02

La denominación de números representados en coma fija y coma flotante, encuentra su justificación por permitir, en esta última, expresar un mismo número de varias formas según la posición variable en que se situe la coma.

A lo largo de este texto a los números representados en cualquiera de los dos formatos recibirán la denominación común de: LITERALES NUMERICOS.

Clasificándose, según el formato, en:

- LITERALES DE COMA FIJA y
- LITERALES DE COMA FLOTANTE.

1.3.1.1.- Literales de COMA FIJA

Los literales de coma fija están formados por una serie de caracteres elegidos entre los dígitos 0 al 9, el signo más, signo menos y el punto decimal o coma.

REGLAS DE FORMACION:

- Pueden contener hasta 18 dígitos y un punto decimal.

---

(2) Hay que advertir que en la notación anglosajona la coma (,) que separa la parte entera de la parte decimal de un número se sustituye por un punto (.). En el lenguaje COBOL puede elegirse entre el punto y la coma para separar la parte entera de la parte decimal. Por tanto, cuantas veces se haga mención en este texto al punto decimal, el lector debe interpretarlo como una coma y, a la inversa.

- Pueden llevar un solo signo más, o un signo menos, pero siempre en el extremo izquierdo. Si carece de signo se supone que el literal es positivo.

- El punto decimal puede estar en cualquier lugar del literal, excepto en el extremo derecho. Si no lleva punto decimal, el literal es un entero.

EJEMPLO:

```
007090          COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100          COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
007110 CALCULO-PLUSF.
007120     IF CASADO   COMPUTE PLUSF. = (500 + NUM-HIJOS * 250)
```

En estas inscripciones 1, 3, 1000, 500 y 250 son literales numéricos de coma fija.

1.3.1.2.- Literales de COMA FLOTANTE

Permiten representar números muy grandes o muy pequeños que no podrían representarse con los literales de coma fija, debido a la limitación que impone el Sistema de que la longitud de dicho literal sea, como máximo, de 18 dígitos (entre parte entera y decimal).

Existen marcas de ordenadores que pueden operar con literales cuyos valores están comprendidos entre: $.72 \times 10^{76}$ y 5.4×10^{-79}, mientras otros pueden variar entre: $.83 \times 10^{307}$ y $-.16 \times 10^{-308}$.

Con estas cifras tan elevadas y tan pequeñas se puede obtener gran precisión en los cálculos. Sin embargo, en las aplicaciones comerciales, por no realizarse cálculos que requieren gran precisión, practicamente no se emplean.

FORMATO:

$$\left[\begin{matrix}+\\-\end{matrix}\right] \text{ mantisa } \underline{E} \left[\begin{matrix}+\\-\end{matrix}\right] \text{ exponente}$$

- La mantisa constituye la parte decimal del literal.

- El exponente indica la pontencia a la que hay que elevar 10.

- La letra E indica que la cifra o cifras que se especifiquen a continuación constituyen el exponente.

- El valor del literal es el producto de la mantisa por diez elevado a la potencia indicada por el exponente.

```
120     02  CAMPOA   COMPUTATIONAL-1  VALUE -.004E+60.
.....
210         ADD 12.5E+25  TO  CAMPOA
```

En estas incripciones, 12 . 5E+ 25 y −.004E60 son literales de coma flotante, cuyos valores equivalen a: $12 . 5 \times 10^{25}$ y $-.004 \times 10^{60}$. Estos literales pueden también especificarse: .125E +27 y −.4E+ 58, ya que al desplazar el punto decimal, se modifica, únicamente, el valor del exponente pero sigue siendo el mismo el valor del literal.

1.3.2.- LITERALES NO NUMERICOS

Están constituidos por una serie de palabras Cobol y cualquiera de los caracteres del juego E.B.C.D.I.C., excepto el apóstrofo.

REGLAS DE FORMACION:

- No pueden exceder de 120 caracteres, debiendo ir encerrados entre apóstrofos (2).

- Pueden contener cualquier carácter del Juego de Caracteres E.B.C.D.I.C. excepto el apóstrofo, ya que al tener que encerrarse todo el literal entre apóstrofos, al colocar un apóstrofo intermedio, el compilador consideraría como literal desde el primer al segundo apóstrofo, ignorando el resto del literal.

- Dentro del literal pueden incluirse palabras reservadas, ya que en este caso pierden el significado de palabras reservadas.

```
006010  PROCEDURE DIVISION.
006020      OPEN   INPUT   ARCHIVO-OPERARIOS,
006030             OUTPUT  CINTA-NOMINA  ARCHI-IMPRESO,
006040      DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO  DD/MM/AA '
006050              UPON CONSOLE
006060      ACCEPT  FECHA  FROM CONSOLE
006070      STOP 'COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESP
006080-             'ECIAL PARA NOMINA'.
```

En estas inscripciones, los literales no numericos son los encerrados entre apóstrofos:

- 'MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA ' y

- ' COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESPECIAL PARA NOMINA '

Observando el ejemplo anterior, se puede comprobar que este tipo de literales los utiliza el programador para componer mensajes que, normalmente, serán recibidos por el operador en la consola del Sistema. Su finalidad es indicar la acción manual que dicho operador u otra persona debe realizar, por ser necesaria para llevar a cabo la ejecución del programa (líneas 006040 y 006070).

Otro uso frecuente, es imprimir dichos literales junto con los resultados obtenidos por el ordenador, a fin de indicar de que contenido se trata:

```
007030          DISPLAY ' REGISTROS PROCESADOS = ' CONTADOR,
```

(2) Aunque la longitud de 120 caracteres es la establecida para la mayor parte de los ordenadores, sin embargo,en alguno de ellos dicha longitud puede alcanzar hasta los 132 caracteres.

1.3.3.- CONSTANTES FIGURATIVAS

Son constantes a las que se les ha asignado un nombre fijo (se consideran palabras reservadas) y un valor predeterminado.

Las constantes figurativas son las siguientes:

ZERO Representa el valor numérico cero o ceros. Puede utilizarse, dentro de un programa, como literal numérico o literal no numérico; es la única constante figurativa que goza de esta propiedad, las demás constantes figurativas se consideran no numéricas.

No puede emplearse con expresiones aritméticas.

ZEROS
ZEROES Es la forma plural y equivalente de la constante ZERO. Representa el valor numérico cero o ceros.

Puede usarse indistintamente la forma singular o plural, tanto para esta constante como para las siguientes.

EJEMPLO:

```
005190  01  CABECERA-2.
005200      02  FILLER          PICTURE  X(114)  VALUE ALL '-'.
005210      02  TITULO-2        PICTURE  X(10)   VALUE 'HOJA NUM.'.
005220      02  NUM-HOJA        PICTURE  9(4)    VALUE ZEROS.
005230      02  FILLER          PICTURE  X(5)    VALUE SPACE.
```

En la descripción del campo NUM–HOJA (línea 005220), se ha reservado una zona de la memoria de cuatro octetos, PICTURE 9(4), cuyo contenido se desconoce, resultante de la ejecución de programas anteriores, pero, al emplear la constante figurativa ZEROS, lo que se consigue es sustituir aquel contenido por el valor cero, es decir, la citada zona pasará a contener el valor 0000.

Esta constante figurativa puede ser sustituída por un literal y a la inversa; así en la línea 005220 en vez de especificar la constante ZEROS podía haberse especificado VALUE 0000.

SPACE
SPACES Representan indistintamente uno o más espacios (blancos), según la descripción que se haya hecho de los campos.

En el ejemplo anterior, línea 005230, el empleo de la constante SPACE, hace que dentro de la zona de cinco octetos que se han reservado en la memoria para el campo FILLER, se inicialice con cinco caracteres blancos en su lugar.

QUOTE Representan indistintamente uno o más apostrofos.
QUOTES

EJEMPLO:

```
005080     01  CABECERA-1.
005090       02  FILLER                    PICTURE  X(36)  VALUE SPACES.
005100       02  TITULO-1                  PICTURE  X(53)  VALUE 'NOMINA DE L
005110-    'A EMPRESA RICOSA CORRESPONDIENTE A '.
```

La descripción del campo TITULO–1 (línea 005100) sirve para escribir sobre el impreso el encabezamiento de la nómina: NOMINA DE LA EMPRESA RICOSA CORRESPONDIENTE A; pero, si se quiere escribir comillas delante de este encabezamiento, es decir "NOMINA DE LA EMPRESA ....., la inscripción se modificaría de la siguiente forma:

```
005080     01  CABECERA-1.
005090       02  FILLER                    PICTURE  X(32)  VALUE SPACES.
005100       02  ALFA                      PICTURE  XX      VALUE QUOTES.
005110       02  TITULO-1                  PICTURE  X(53)  VALUE 'NOMINA DE L
005120-    'A EMPRESA RICOSA CORRESPONDIENTE A '.
```

ALL `carácter´ –Carácter– representa cualquier aparición del juego E.B.C.D.I.C. excepto el apóstrofo y genera una secuencia de apariciones del único carácter situado dentro de los apóstrofos.

En la línea 005200, la constante ALL`´ hace que dentro de los apóstrofos se situe el carácter blanco, por lo que el campo FILLER, al que se le reservaron 114 octetos en la memoria, se llenará con blancos.

Si dentro de los apóstrofos, en vez de un blanco, estuviera otro cáracter, se llenaría todo el campo con dicho carácter.

HIGH – VALUE
HIGH – VALUES Representa uno o más sucesos del valor más elevado en la secuencia del código E.B.C.D.I.C., es decir, el FF hexadecimal.

LOW – VALUE
LOW – VALUES Representa uno o más sucesos del valor más bajo en la secuencia del código E.B.C.D.I.C., es decir, el 00 hexadecimal.

1.4.- HOJA DE CODIFICACION

La Hoja de Codificación está diseñada de tal forma que proporcione un medio para facilitar la escritura de programas en COBOL. Está compuesta de 80 columnas que se corresponden con las 80 columnas de la ficha.

El contenido de cada línea de la Hoja se perfora en una ficha diferente, excepto la información que se escribe en la parte superior de la Hoja dentro de los epígrafes PROGRAMA, PROGRAMADOR, etc. Opcionalmente, y a juicio del programador, puede perforarse la información contenida dentro de PAGINA, LINEA e IDENTIFICACION.

La figura 19 muestra un modelo de Hoja de Codificación en Cobol.

El programador, al utilizar esta Hoja, debe tener en cuenta las reglas siguientes:

COLUMNAS 1 a 3.

Sirven para numerar cada página del programa. Por tanto, las tres primeras columnas de cada hoja llevarán las mismas cifras.

COLUMNAS 4 a 6.

Sirven para numerar las líneas de página. Normalmente van numeradas de diez en diez en vez de uno en uno, ya que ofrece la ventaja de que cuando por olvido, el programador deje de escribir alguna línea o tenga que rectificar el programa, pueda efectuar las intercalaciones que fueran necesarias, para lo cual utilizará las líneas sin numerar que hay al final de la página, asignándoles el número de línea que le corresponda.

La combinación de los números de página y línea se llaman: NUMEROS DE SECUENCIA.

Los números de secuencia sólo constan de dígitos y es opcional el escribirlos o no; si el programador incluye el número de secuencia, el compilador comprueba la secuencia en orden ascendente, e indica los errores que pudiera haber en dicha secuencia; caso de omitirse, es decir, si quedan las seis primeras columnas en blanco, el compilador no indica ningún error. No obstante, esta clase de errores carecen de importancia, ya que no influyen en la ejecución del programa y por tanto sobre los resultados.

COLUMNA 7.

Sólo puede figurar dentro de ella un guión o un asterisco:

Hoja de Codificación en Cobol

PROGRAMADOR:

PROGRAMA:

HOJA DE

SISTEMA

FECHA

INSTRUCCIONES PARA PERFORACION

SIGNO

PERFORACIONES

Identificación 73 80

Página 1 3 | Linea 4 6 | Cont. 7 | A 8 | B 12 15 20 25 30 35 40 45 50 55 60 65 70

| | | |

Si se emplea un guión, servirá como indicador de:

a) Continuación de un literal no numérico, cuando la longitud de éste es tal que ocupa más de una línea:

```
006180     DISPLAY '                              CODIGO                    APELLIDOS  Y  NOMBRE
006190-    '                          SUELDO   TRIENIOS  HORAS-EX.   PLUS-F.  DESC
006200-    'UEN.   TOTAL'.
```

El literal no numérico del ejemplo, comprende desde el apóstrofo que sigue a DISPLAY (línea 006180) hasta el último apóstrofo que precede al punto final (línea 006200); por tanto, ocupa 116 posiciones o columnas, siendo los guiones situados en la columna 7 de las líneas 006190 y 006200 indicativos de su continuación.

Los apóstrofos pueden situarse, a partir de la columna 12, en cualquiera de ellas donde el programador desee continuar el literal. En este ejemplo, utilizó la columna 20 para los tres apóstrofos, pero podía haber utilizado otra columna cualquiera y diferente para cada uno de ellos:

```
006180     DISPLAY '                              CODIGO                    APELLIDOS  Y  NOMBRE
006190-    '                          SUELDO   TRIENIOS  HORAS-EX.   PLUS-F.  DESCUE
006200-    'N.   TOTAL'.
```

b) Continuación de un literal o palabra COBOL, de tal manera que el primer carácter escrito en la columna 12 de esa línea, es continuación del último carácter escrito sobre la columna 72 de la línea anterior:

```
007140     IF VIUDO    COMPUTE PLUSF = NUM-HIJOS * 250 ELSE                    MO
007150-    VE  0  TO PLUSF.
```

Si se especifica un asterisco, es para intercalar un comentario explicativo que puede contener cualquier combinación del juego de caracteres E.B.C.D.I.C. Dicho comentario puede hacerse en cualquier línea del programa, así como comenzar a partir de la columna 8:

```
004120  77  NUMTRIENIOS                          PICTURE 99.
004130  77  TOTALS                               PICTURE 9(6).
004140* EL SIGUIENTE REGISTRO DESCRIBE EL AREA DE EDICION.
004150      01  IMPRESO-S.
004160          02  FILLER                      PICTURE X(15) VALUE SPACES.
```

COLUMNAS 8 a 72

Se emplean para escribir el Programa Fuente. Estas 65 columnas están agrupadas en dos márgenes, el margen A, que engloba las columnas de la 8 a la 11, y el margen B, de la 12 a la 72.

INSCRIPCIONES QUE COMIENZAN DENTRO DEL MARGEN A

Debe iniciarse la escritura dentro del margen A:

- Los nombres de División (IDENTIFICATION, ENVIRONMENT, DATA Y PROCEDURE)
- Los nombres de Sección (CONFIGURATION, INPUT–OUTPUT, FILE, WORKING–STORAGE).
- Los nombres de Párrafo (PROGRAM–ID, AUTHOR, INSTALLATION, DATE–WRITTEN, DATE–COMPILED, SECURITY, REMARKS, SOURCE–COMPUTER, OBJECT–COMPUTER, FILE–CONTROL, I-O-CONTROL).
- Los nombres de Procedimiento (En el ejemplo, ESCRIBIR–1, ESCRIBIR–2, LEER, CALCULO–TRIENIOS, CALCULO–PLUSF, AVISO, 10, ALFA–1).
- El indicador FD de descripción de archivo en la DATA DIVISION.
- Los números de nivel 01 y 77 de descripciones de datos.

Pueden escribirse dentro del margen A los números de nivel del 02 al 49, 66 y 88.

INSCRIPCIONES QUE COMIENZAN DENTRO DEL MARGEN B

Debe iniciarse la escritura dentro del margen B:

- El resto de las inscripciones que constituyen el programa.

EJEMPLO:

```
Página Línea  A
1   3  4   6  7  8  12  15  20  25  30  35  40  45  50  55  60  65  70
004010     FD   ARCHI-IMPRESO   RECORDING IS F
004020         LABEL RECORDS ARE OMITTED
004030         DATA  RECORD  IS IMPRESO.
004040     01  IMPRESO                    PICTURE X(133).
004050     WORKING-STORAGE SECTION.
004051 77  CONTALINEAS                    PICTURE 99          VALUE ZERO.
004060 77  CONTADOR                       PICTURE 9(5)        VALUE ZEROS.

007090         COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100         COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
007110 CALCULO-PLUSF.
007120         IF  CASADO   COMPUTE  PLUSF  = (500 + NUM-HIJOS * 250)
007130             ELSE
007140             IF  VIUDO COMPUTE  PLUSF  = NUM-HIJOS * 250  ELSE
007150             MOVE 0  TO PLUSF.
```

COLUMNAS 73 a 80

– Su escritura es opcional.

Caso de utilizarse, se puede escribir en ellas cualquier carácter COBOL, que servirá al programador para identificar el programa. Se perfora su contenido en todas las fichas que constituyen el Programa Fuente, y al hacerlo así se tiene la ventaja de que cualquier ficha que pudiera extraviarse, al ser localizada se sabría a qué programa pertenece.

CONSIDERACIONES GENERALES.

– Los nombres de División seguidos de un blanco, de la palabra DIVISION y punto, deben escribirse sólos en una línea.

– Los nombres de Sección seguidos de un blanco, la palabra SECTION y un punto, deben figurar al igual que los nombres de DIVISION en línea única.

– Los nombres de Párrafo seguidos de un punto, no es obligatorio escribirlos sólos en una línea, pues a continuación de ellos puede ir cualquier inscripción.

```
001110  ENVIRONMENT DIVISION.
001120      CONFIGURATION SECTION.
001130      SOURCE-COMPUTER. IBM-370.
001140      OBJECT-COMPUTER. IBM-370.
001150      SPECIAL-NAMES.
001160          C01 IS PAG-NUEVA
```

En este ejemplo, el nombre de párrafo SOURCE–COMPUTER no va escrito sólo en una línea, pues a continuación está escrito IBM–370, mientras el párrafo SPECIAL–NAMES ocupa una línea. El escribirlo de una forma u otra la decide el criterio del programador. Normalmente, el dejar un nombre de párrafo solo en una línea, obedece a que resalte más su lectura para cualquier usuario del programa.

– No es obligatorio llenar totalmente de inscripciones una línea.
El lenguaje COBOL da gran libertad en su escritura, de tal manera que la inscripción que ocupa toda la línea 001200 podía escribirse:

```
001170          SELECT
001180              ARCHIVO-OPERARIOS
001190                  ASSIGN    SYS007-UR-2540R-S
001200                      RESERVE NO
```

– Si una inscripción es tan larga que no cabe en una línea, se continúa en la línea o líneas siguientes, pero a partir del margen B:

```
006040      DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA'
006050              UPON CONSOLE,
006060      ACCEPT FECHA FROM CONSOLE,
```

La declaración DISPLAY ' MARQUE..... ' UPON CONSOLE, al no caber en una línea, se continúa a partir de la columna doce en cualquiera de las siguientes, es decir, dentro del margen B.

– Se pueden dejar líneas en blanco dentro del programa, siempre que no se haya colocado un guión en la columna 7 para indicar la continuación de una palabra o literal.

1.5.- REGLAS DE PUNTUACION

En la escritura de un programa en el lenguaje COBOL hay que tener en cuenta las reglas de puntuación siguientes:

– No deberán ir precedidos de un espacio, pero sí seguidos de un espacio por lo menos, los caracteres punto, punto y coma y la coma (línea 007010 a 007050).

EJEMPLO:

```
Página  Línea  Cont A                B
  1  3  4  6   7  8                 12
007010          LEER.
007020          READ ARCHIVO-OPERARIOS; AT END,
007030          DISPLAY '                REGISTROS PROCESADOS = ' CONTADOR,
007040          CLOSE   ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050          STOP RUN.
007060      CALCULO-TRIENIOS.
007070          IF   MES OF FECHAINGRESO < M,
007080          COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3   ELSE
007090          COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100          COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
```

– La coma o el punto y coma pueden utilizarse para separar declaraciones u operandos de esas declaraciones (línea 007020).

– El paréntesis izquierdo no debe ir seguido de un espacio (línea 007080).

– El paréntesis derecho no debe ir precedido de un espacio (línea 007080).

– El signo igual y los operadores aritméticos + – * / deben ir precedidos y seguidos de un espacio (línea 007090 y 007100).

– Entre dos palabras sucesivas y/o literales y/o expresiones aritméticas encerradas entre paréntesis debe existir, por lo menos, un espacio; si existe más de uno, el ordenador lo interpreta como un solo espacio. Así, desde donde termina la inscripción de la línea 007020 hasta que comienza la siguiente inscripción DISPLAY existen espacios en blanco, pero el ordenador lo considera como uno sólo.

– Si un elemento (carácter, literal numérico, palabra reservada) termina en la columna 72, el compilador inserta al final de él un espacio.

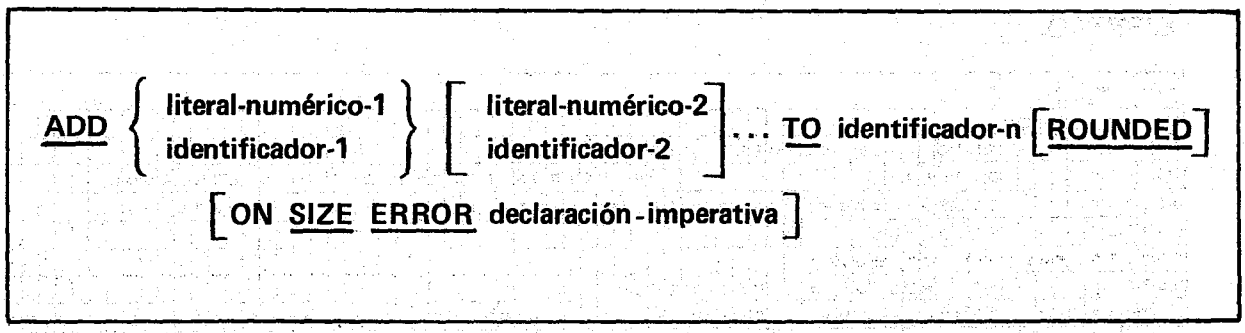
1.6.- NOTACION SOBRE FORMATOS

En la creación del lenguaje, a cada una de las partes componentes del mismo (DIVISIONES, declaraciones, etc.) se le asignó un determinado FORMATO (ver resumen de formatos: APENDICE C) al cual debe el programador ajustarse estrictamente al confeccionar el programa.

Cualquier cambio que se pretenda introducir en dicho FORMATO, es motivo para que el ordenador no sólo no lo tome en cuenta, sino que impida el llevar a cabo la ejecución del programa

Cada FORMATO contiene un sistema de símbolos convencionales (llaves, corchetes, subrayado, etc.) que constituyen una NOTACION cuyo significado es común para todos los formatos, y es necesario conocer, a fin de que cada inscripción que se haga en el programa sea correcta.

Tomando como ejemplo el formato de la declaración ADD:



Tal como se presenta este formato no sería comprensible para un profano. Su significado, a grandes rasgos, sería:

Sumar (ADD) el valor de uno o varios campos de datos de la memoria y/o literales numéricos ({ literal-numérico-1 / identificador-1 } [literal-numérico-2 / identificador-2] ...) al (TO) contenido de otro campo de datos (identificador-n) guardando el resultado en este campo y redondeando su valor ([ROUNDED]); caso de que el resultado no quepa dentro del espacio reservado en la memoria para identificador-n ([ON SIZE ERROR]), ejecutar una declaración imperativa.

El ejemplo siguiente ayudará a seguir la explicación del significado, reglas que gobiernan el empleo de cada símbolo de la Notación, y su aplicación al ejemplo propuesto.

EJEMPLO. Utilizando el formato de la declaración ADD se quiere realizar la siguiente operación:

(CAMPOB+ CAMPOC+32.5)+CAMPOA (redondeado)

Es decir, sumar el valor que contengan los campos de la memoria: CAMPOB y CAMPOC al literal numérico 32.5, y el resultado sumarlo al contenido del campo denominado CAMPOA, guardando la suma, redondeada, en este campo.

Examinando el formato de la declaración ADD se puede ver que sí es posible realizar dicha operación, y la forma de codificarla en el programa sería:

```
ADD CAMPOB, CAMPOC, 32.5 TO CAMPOA ROUNDED
```

La interpretación de los símbolos que forman parte de la NOTACION, así como los pasos sucesivos en la construcción de la declaración del ejemplo es como sigue:

MAYUSCULAS.

- Las palabras escritas con mayúscula son palabras reservadas. En el formato ejemplo: **ADD, TO ON, SIZE, ERROR, ROUNDED.**
- Las palabras reservadas que estén subrayadas, y no se encuentren dentro de llaves o corchetes, es obligatorio especificarlas; es decir, deben ir copiadas literalmente en el programa: **ADDyTO**
- Las palabras reservadas no subrayadas son siempre opcionales **(ON)**, es decir, pueden escribirse u omitirse según el deseo del programador, ya que, normalmente, sirven para completar una frase a fin de ser más legible el programa por aquella persona que conozca el idioma inglés
  Aplicando la regla del párrafo segundo a la declaración ejemplo, permite escribir:

ADD ..... TO .....

MINUSCULAS

- Las palabras escritas en letras minúsculas indican la clase de información que debe crear o incluir el programador. Esta información es obligatorio escribirla en el programa, excepto cuando está dentro de la parte opcional del formato. Es decir, según el formato citado después de la palabra TO debe escribirse el nombre de un campo de datos **(identificador-n)**, y después de la palabra ADD deberá especificar el nombre de un campo de datos o literal numérico. Aplicando dicha regla a la declaración del ejemplo, permite escribir:

ADD ..... TO CAMPOA .....

LLAVES.

Las llaves indican que es obligatoria la elección entre dos o más posibilidades escritas en columna. Por tanto, en este formato es obligatorio elegir entre: **literal-numérico-1 e identificador-1.** Aplicándolo a la declaración del ejemplo, permite escribir:

ADD CAMPOB ..... TO CAMPOA .....

CORCHETES.

- Indican que el texto incluido entre ellos es opcional, es decir, se podrá incluir u omitir en la inscripción según las exigencias o resultados a obtener con el programa. En el formato del ejemplo: **ROUNDED y SIZE ERROR,** son partes opcionales.
  Aplicando la regla a la declaración del ejemplo, permite escribir:

ADD CAMPOB ..... TO CAMPOA ROUNDED

Si en el formato hay dos o más partes opcionales que estén encerradas dentro del mismo corchete podrá elegirse una o ninguna de ellas.

En el formato ejemplo podría elegirse entre literal-numérico-2 e identificador-2, y aplicando la regla a la declaración del ejemplo, permite escribir:

ADD CAMPOA, CAMPOC ..... TO CAMPOA ROUNDED

PUNTOS (Elipsis).

— Los tres puntos (...) indican que una palabra, o grupo de ellas, en minúscula y una o más palabras reservadas encerradas entre conchetes o llaves, puede repetirse una o cualquier número de veces sucesivas (a discreción del programador). Si los puntos van detrás de una palabra se refieren a esa palabra y si van detrás de corchetes y llaves, indican que puede repetirse todo lo que hay entre ellos.

En el formato se podía haber especificado en vez de los ..., la expresión equivalente:

[literal-numérico-3 / identificador-3] [literal-numérico-4 / identificador-4] [

Por tanto, aplicando esta regla a la declaración del ejemplo, permite escribir:

ADD CAMPOB, CAMPOC, 32.5 TO CAMPOA ROUNDED.

Con lo que queda construída la declaración.

GUION.

— Su objeto es facilitar la referencia a las palabras minúsculas, colocando a continuación de él un dígito o una letra. En el formato del ejemplo, el guión hace diferenciar entre identificador-1 e identificador-2. En el formato de algunas declaraciones figura algún carácter de puntuación, así como los signos + — */> < = los cuales, aunque no están subrayados, es necesario escribirlos siempre que figuren en el mismo.

Las inscripciones del ejemplo siguiente muestran diferentes formas de escribir una declaración ADD, aplicando las reglas citadas:

```
010    ADD 3.52 TO CAMPOA ROUNDED.
020    ADD CAMPOB, .34, +5.25E02 TO TOTAL SIZE ERROR GO TO B.
030    ADD CAMPOA, CAMPOB, CAMPOC TO CAMPOE
040    ADD 27.450 TO CAMPOA ROUNDED ON SIZE ERROR MOVE 1 TO LIST
```

Cada una de estas inscripciones obliga al ordenador a realizar la operación aritmética, sumar, pero según la elección que haga el programador de las partes opcionales del formato, en dicha suma intervendrán más o menos operandos, se redondeará o no el resultado, etc.

La elección de las partes opcionales del formato no es caprichosa, sino que la hace el programador de acuerdo con los resultados que desee obtener (ver pág. 156).

2.- EJERCICIOS

1.— ¿Cuál de los siguientes caracteres no es un carácter COBOL?

A.— :
B.— ;
C.— .
D.— ,

2.— ¿Es correcto escribir en la PROCEDURE DIVISION ?

MOVE PLUSF TO PLUS

3.— Las siguientes palabras no son nombres creados, indicar el motivo.

A.— NUM HIJOS
B.— DIVISION
C.— ALFA/1
D.— ESCRIBIR-1-

4.— Los siguientes literales no son correctos, indicar el motivo.

A.— 45.
B.— 147.5—
C.— + 483
D.— 7000$

5.— Alguna de las siguientes inscripciones no son literales no numéricos, explicar el motivo.

A.— '1248 '
B.— MARQUE LA FECHA
C.— 'DATA DIVISION'
D.— ' '

6.— Alguno de los siguientes nombres-de-datos no es correcto, indicar el motivo.

A.— ABCDE
B.— X— — — B
C.— 12
D.— A5678

7.— ¿Cuál de las siguientes constantes no es figurativa? Indicar el motivo.

A.— ZEROES
B.— SPACOES
C.— ALL B
D.— HIGH VALUE

8.— En la declaración del programa NOMINA:

COMPUTE TRIENIOS = NUMTRIENIOS * 1000

¿ Cuáles son los literales numéricos y los nombres de datos ?

9.— En las columnas 1 a 6 y 73 a 80 de la Hoja de Codificación, el programador puede hacer cualquier inscripción, ya que su contenido no influye durante la ejecución del programa.

CIERTO FALSO

10.— Al escribir las siguientes líneas — del programa NOMINA — el programador ha cometido errores en alguna de ellas. Decir cuáles son dichos errores.

```
001180 INPUT-OUTPUT SECTION.
001190 FILE-CONTROL. SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-UR-2540R-S
001200                      RESERVE NO ALTERNATEAREA
```

```
004010     FD  ARCHI-IMPRESO  RECORDING IS F
004020     LABEL RECORD OMITTED
004030     DATA RECORD IS IMPRESO.
004040 01  IMPRESO          PICTURE X(133).
004050 WORKING-STORAGE SECTION
004060 77  CONTALINEAS      PICTURE 99   VALUE ZEROS.
```

```
006180     DISPLAY '                    CODIGO          APELLIDOS  Y  NOMBRE
006190             '                SUELDO    TRIENIOS  HORAS-EX.  PLUS-F.  DESC
006200             'UEN.    TOTAL '.
```

```
007010 LEER.
007020     READ ARCHIVO-OPERARIOS; AT END,
007030     DISPLAY '                REGISTROS PROCESADOS= ' CONTADOR,
```

```
007060 CALCULO-TRIENIOS. IF MES OF FECHAINGRESO < M
007070     COMPUTE NUMTRIENIOS = ( A - ANNO OF FECHAINGRESO) /3 ELSE
007080     COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO-1) / 3.
```

CAPITULO III

ESTRUCTURA DE UN PROGRAMA

1.-GENERALIDADES

Según se ha expuesto en el Capítulo I, la escritura de un programa está estructurada en cuatro DIVISIONES cuyo nombre es fijo y deben especificarse en el orden que se preestableció cuando se creó el lenguaje COBOL.

Cada DIVISION se subdivide en una serie de inscripciones: SECCIONES,PARRAFOS, etc., donde se debe incluir la información que sea requerida y que no tiene por qué ser la misma en todos los programas. Al igual que las DIVISIONES, tienen asignado un nombre fijo y han de especificarse según el orden preestablecido.

Por tanto, el programador,cuando lleve a cabo la confección de un programa,debe ir escribiendo los títulos o nombres de las DIVISIONES, SECCIONES, PARRAFOS, etc, junto con la información solicitada en cada una de ellas, en el orden que se establece en la estructura siguiente.

2.- ESTRUCTURA DE UN PROGRAMA COBOL

{ IDENTIFICATION DIVISION. / ID DIVISION. }

PROGRAM-ID. nombre-de-programa.
[AUTHOR. entrada-de-comentarios .....]
[INSTALLATION. entrada-de-comentarios .....]
[DATE-WRITTEN. entrada-de-comentarios .....]
[DATE-COMPILED. entrada-de-comentarios .....]
[SECURITY. entrada-de-comentarios .....]
[REMARKS. entrada-de-comentarios .....]

```
ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

SOURCE-COMPUTER.  entrada modelo-de-ordenador.
OBJECT-COMPUTER.  entrada modelo-de-ordenador.
SPECIAL-NAMES.    entrada nombres especiales.]

[INPUT-OUTPUT SECTION.

{FILE-CONTROL.  entrada de control de archivos.}
[I-O-CONTROL.   entrada de control de entrada/salida.]]

DATA DIVISION.

[FILE SECTION.
{entrada de descripción de archivos ...
{entrada de descripción de registros} ...]
[WORKING-STORAGE SECTION.
[entrada de descripción de campos de datos] ...
[entrada de descripción de registros] ...]
[LINKAGE SECTION.
[entrada de descripción de campos de datos] ...
[entrada de descripción de registros] ...]
[REPORT SECTION.
{entrada de descripción de informe
{entrada de descripción de grupos de informe } ...]

PROCEDURE DIVISION.

[DECLARATIVES.
{SECTION nombre-de-sección. Sentencia USE
{nombre-de-párrafo.{Sentencia }...}...
 END DECLARATIVES.]
{SECTION nombre-de-sección.
{nombre-de-párrafo. {Sentencia }...} ...
```

De acuerdo con este formato, existen partes obligatorias que, como es obvio, deberán especificarse y partes opcionales de las que de alguna o de todas ellas el programador podrá prescindir; así por ejemplo, si en la ejecución de un programa no se van a utilizar campos de trabajo para situar resultados parciales, o no se han de escribir encabezamientos en informes impresos, puede prescindirse de la WORKING–STORAGE SECTION, etc.

2.-IDENTIFICATION DIVISION

2.1.- GENERALIDADES

La función principal de la IDENTIFICATION DIVISION es proporcionar un nombre mediante el cual el compilador pueda identificar al programa, cuando se lleve a cabo su ejecución. Puede además contener información acerca del autor, nombre de la instalación, fecha en que fue escrito, fecha en que fue compilado , nivel de seguridad y observaciones.

2.2.- ESTRUCTURA

```
{ IDENTIFICATION DIVISION. }
{ ID DIVISION.             }
  PROGRAM-ID. nombre-de-programa.
[ AUTHOR.        entrada-de-comentarios ... ]
[ INSTALLATION.  entrada-de-comentarios ... ]
[ DATE-WRITTEN.  entrada-de-comentarios ... ]
[ DATE-COMPILED. entrada-de-comentarios ... ]
[ SECURITY.      entrada-de-comentarios ... ]
[ REMARKS.       entrada-de-comentarios ... ]  (2)
```

- El párrafo PROGRAM-ID es obligatorio; los párrafos restantes son opcionales y pueden especificarse en cualquier orden; sin embargo, algunos compiladores exigen que los párrafos sean especificados en el orden en que se describen en el formato.

- Nombre-de-programa es un nombre que, aunque creado por el programador, debe procurar que el número de caracteres no exceda de ocho, por ser inútil especificar más de dicho número, ya que el compilador sólo considera los ocho primeros, debiendo ser el primero de éllos alfabético y el resto pueden ser letras y/o números.

---

(2) La palabra REMARKS puede sustituirse por un asterisco (*) en la columna 7, seguida del comentario a partir de la columna 8 ó siguientes.

- Cualquier inscripción que se haga dentro del párrafo DATE–COMPILED lo ignora el ordenador, sustituyéndola por la fecha en que se lleva a cabo la compilación del programa.

- IDENTIFICATION, PROGRAM–ID y restantes párrafos deben comenzar dentro del margen A.

- Cada inscripción puede comenzar a continuación del nombre del párrafo, o bien en línea propia a partir del margen B.

- Cualquier carácter del juego de caracteres COBOL, incluso palabras reservadas, pueden emplearse dentro de las inscripciones de cada párrafo.

Haciendo uso de las reglas especificadas, el contenido de la IDENTIFICATION DIVISION del programa Nómina se escribirá de la forma siguiente:

```
Página  Línea  COD  A       B
1   3   4   6   7  8   12   15   20   25   30   35   40   45   50   55   60   65
001010  IDENTIFICATION DIVISION.
001020      PROGRAM-ID. NOMINA.
001030      AUTHOR. PROGRAMADOR C. DE LA FUENTE.
001040      INSTALLATION. CENTRO PROCESO DE DATOS DE RICOSA.
001050      DATE-WRITTEN. 15-MARZO-1977.
001060      DATE-COMPILED.
001070      SECURITY. ESTE PROGRAMA SOLO PUEDE SER UTILIZADO POR LA
001080                SECCION DE CONTABILIDAD.
001090      REMARKS. PROGRAMA PARA OBTENER LA NOMINA DE RICOSA Y
001100               CREAR UN ARCHIVO EN CINTA PARA POSTERIORES TRABAJOS.
```

El significado de cada párrafo se encuentra en la lista de palabras reservadas.

- Al ser obligatoria la descripción de IDENTIFICATION DIVISION y PROGRAM–ID nombre-del-programa, y opcional la del resto de los párrafos, la misma función haría para el compilador si se prescindiera de los párrafos que siguen a PROGRAM–ID, tal como muestra la siguiente descripción:

```
Página  Línea  COD  A       B
1   3   4   6   7  8   12   15   20   25   30   35   40   45   50   55   60
001010  ID DIVISION.
001020      PROGRAM-ID. NOMINA.
001030      ENVIRONMENT DIVISION.
```

3.-ENVIRONMENT DIVISION

3.1.- GENERALIDADES

La ENVIRONMENT DIVISION tiene las siguientes funciones :

- Especificar las características físicas del ordenador que se ha de utilizar para compilar y ejecutar el programa.

- Proporcionar al compilador información acerca del modelo de los dispositivos físicos (lectoras, unidades de cinta etc.) que se van a encargar de leer los archivos donde se encuentran los datos del problema y de escribir los resultados del proceso, así como forma en que se ha de realizar esta lectura y/o escritura (secuencial o aleatoriamente).

- Relacionar los nombres lógicos que se dan en el programa a los archivos, con los nombres de los modelos de dispositivos en que han de ser leídos y/o escritos.

- Especificar técnicas especiales a emplear durante el proceso, como: control del carro de la impresora, establecer puntos de control,etc.

3.2.- ESTRUCTURA

ENVIRONMENT DIVISION.

[**CONFIGURATION SECTION.**

Párrafo SOURCE-COMPUTER. entrada modelo-de-ordenador.

Párrafo OBJECT-COMPUTER. entrada modelo-de-ordenador.

Párrafo SPECIAL-NAMES. entrada nombres especiales.]

[**INPUT-OUTPUT SECTION.**

Párrafo FILE-CONTROL. entrada de control de archivos.

[**Párrafo I-O-CONTROL. entrada de control de entrada/salida.**]]

Todas las secciones y párrafos de esta DIVISION son opcionales.

3.3.-CONFIGURATION SECTION

FUNCION: Servir como documentación en un programa COBOL, así como asociar nombres utilizados por el compilador con los nombres nemotécnicos especificados por el programador.

FORMATO:

[CONFIGURATION SECTION.
SOURCE-COMPUTER. modelo ordenador-fuente.
OBJECT-COMPUTER. modelo ordenador-objeto.
SPECIAL-NAMES. nombres especiales.]

Tanto CONFIGURATION SECTION como los párrafos que la componen son opcionales y su escritura debe comenzar dentro del margen A.

3.3.1.- Párrafos SOURCE y OBJECT

Especifican el modelo del ordenador (SOURCE-COMPUTER) con el en que se hará la compilación o traducción del programa fuente y el modelo del ordenador (OBJECT-COMPUTER) en el cual se ejecutará el programa objeto.

```
001110  ENVIRONMENT DIVISION.
001120      CONFIGURATION SECTION.
001130      SOURCE-COMPUTER. IBM-370.
001140      OBJECT-COMPUTER. IBM-370.
```

En la descripción hecha en el ejemplo Nómina se utiliza el mismo modelo de ordenador para compilar y ejecutar el programa; el nombre del modelo lo fija la casa constructora.

Si el programador hubiera omitido escribir estos dos párrafos, el programa se ejecutaría de la misma forma, ya que el compilador COBOL trata la información contenida en ellos como comentarios.

3.3.2.- Párrafo SPECIAL-NAMES

FUNCION: Relacionar nombres de función, establecidos por el fabricante para cada ordenador, con los nombres nemotécnicos creados por el programador. Intercambiar las funciones de la coma por el punto, tanto en los literales numéricos como en la serie de caracteres de una PICTURE; así como permitir que pueda imprimirse delante de cantidades numéricas, un carácter representativo del símbolo monetario.

FORMATO:

SPECIAL-NAMES.

[nombre-de-función IS nombre-nemotécnico] ...
[DECIMAL-POINT IS COMMA]
[CURRENCY SIGN IS literal] .

Nombre-nemotécnico es un nombre creado por el programador y para su formación siguen las mismas reglas que para los nombres creados.

Los nombres-de-función difieren considerablemente de un ordenador a otro. En este texto se describen los nombres utilizados en el ordenador con el que se ha ejecutado el programa NOMINA. Son nombres-de-función (3):

CONSOLE	:Nombre con el que reconoce el compilador a la máquina de escribir de la consola.
SYSIN **SYSIPT**	:Nombres con los que reconoce al dispositivo físico que se utilizará como unidad de entrada de los programas (normalmente la lectora de fichas).
SYSOUT **SYSLST**	:Nombres con los que reconoce al dispositivo físico que se utilizará como unidad de salida de los resultados.
C01 a C12 y CSP	:Nombres con los que el compilador reconoce las perforaciones hechas en la cinta de control de carro de la impresora para controlar el espaciado en la escritura sobre el impreso, y supresión del espaciado en la salida impresa respectivamente.

En el programa Nómina se ha asociado un nombre nemotècnico, PAG-NUEVA, al nombre de función, C01, pudiendo especificarse aquél dentro de una declaración WRITE (línea 008120), la cual obliga a la impresora a avanzar el impreso hasta el comienzo de una nueva página, una vez que terminó de escribir en la página anterior.

```
001150     SPECIAL-NAMES.
001160         C01   IS   PAG-NUEVA
001170         DECIMAL-POINT   IS COMMA.
```

(3) Hay marcas de ordenadores que utilizan para reconocer los diferentes tipos de dispositivos físicos, nombres-de-función como: PRINTER (impresora) UNISERVO (unidad de cinta) etc., además de alguno de los ya citados: CONSOLE, SYSIN, etc.
Esta diversidad obliga a tener que consultar siempre el manual específico de cada ordenador.

Igualmente, si se asocia un nombre de función, por ejemplo CONSOLE, a otro nombre nemotécnico, CON, éste se puede emplear con una declaración ACCEPT o DISPLAY.

```
030    SPECIAL-NAMES.
040        CONSOLE IS CON.
```

Sería válido, por tanto, escribir en la PROCEDURE DIVISION:

ACCEPT FECHA FROM CON en vez de: ACCEPT FECHA FROM CONSOLE (línea 006060).

Así como en el caso de ACCEPT y DISPLAY no es obligatoria la asociación, sin embargo, en el caso de WRITE sí es preceptivo. No es correcto escribir: **WRITE IMPRESO AFTER C01.**

Opción DECIMAL-POINT.

La inscripción DECIMAL-POINT IS COMMA es opcional, y por tanto si se omite en el programa, los resultados numéricos que se escriban en la impresora llevarán el sistema de puntuación sajón, es decir, un punto entre las cifras enteras y decimales, y una coma separando los millares.

Si se especifica esta opción, el programador debe cambiar en las PICTURES y literales numéricos los puntos por las comas y viceversa, con lo que se obtendrán los resultados impresos en sistema de representación decimal (ver página 49).

Opción CURRENCY SIGN.

Esta opción es, generalmente, utilizada en aquellos paises en los que la moneda oficial no es el dolar ($), ya que permite imprimir delante de las cifras que indiquen valores monetarios el carácter representativo de la moneda del país.

El literal que figura a continuación de CURRENCY SIGN está limitado a un sólo carácter, debiendo, también, especificarse delante de la serie de caracteres que siguen a la palabra PICTURE del campo que ha de contener las cantidades monetarias.

Este carácter no puede ser uno de los siguientes:

- Dígitos del 0 al 9.
- Caracteres alfabéticos A, B, C, D, P, R, S, V, X, Z.
- Caracteres especiales * + - , . ; ( ) ' y el blanco.

Por ejemplo, si deseamos que en el programa NOMINA figure en vez de un asterisco delante del SUELDO NETO, otro símbolo que se convenga sea significativo de pesetas, por ejemplo: T (por no disponer del símbolo Pts.), habría que especificar en el programa las siguientes inscripciones:

```
001150  SPECIAL-NAMES.
001155          CURRENCY SIGN IS 'T'.
.....
005060  02 TOTAL          PICTURE T999.999.
```

Verdaderamente, la letra (T) que se ha elegido como indicativa de pesetas es poco significativa, sin embargo, se puede incorporar físicamente a la cadena de caracteres de la impresora un caracter: Pts , sustituyéndolo por otro carácter de la misma que sea poco utilizado.

3.4.-INPUT-OUTPUT SECTION

FUNCION: Asignar nombres a cada uno de los archivos de entrada/salida, así como asegurar una eficiente transmisión de los datos entre los dispositivos físicos que contienen dichos archivos y la unidad de proceso.

ESTRUCTURA:

```
[INPUT-OUTPUT SECTION.
 FILE-CONTROL. {entrada de control de archivos } ...
[I-O-CONTROL.  entrada de control de entrada/salida ].]
```

- Esta Sección es opcional, pudiéndose omitir cuando no se utilizan archivos para la entrada y/o salida de los datos al ordenador, empleando para ello cualquier otro medio, como la máquina de escribir de la consola, o por describir los datos en la WORKING-STORAGE SECTION, etc.
- No puede omitirse cuando se utilicen archivos de entrada y/o salida.

3.4.1.-Párrafo FILE-CONTROL

FUNCION:
- Dar un nombre a cada archivo.
- Asignar cada archivo a un dispositivo físico de entrada/salida.
- Especificar los dispositivos físicos que se van a encargar de la lectura y/o escritura de los archivos.
- Especificar el número de áreas de entrada y/o salida a reservar para los datos de cada archivo.

FORMATO:

```
FILE-CONTROL.
Cláusula SELECT serie de cláusulas.
```

3.4.1.1.- Cláusula SELECT

La cláusula SELECT se compone de una serie de cláusulas, de las cuales el programador especificará dentro del programa sólo aquéllas que contribuyan a hacer más eficiente la ejecución del mismo.

Normalmente, el número de cláusulas a especificar dependerá de:

- Como están organizados los archivos de entrada.
- Como han de organizarse los archivos de salida.
- Como han de leerse o escribirse dichos archivos (secuencial o aleatoriamente).
- El dispositivo físico que haya de leerlos y/o escribirlos.

FORMATO:

```
{ SELECT [OPTIONAL]   nombre-de-archivo
  Cláusula ASSIGN
 [Cláusula RESERVE]
 [Cláusula ACCESS MODE]
 [Cláusula NOMINAL KEY]
 [Cláusula RECORD KEY]
 [Cláusula ACTUAL KEY] } ...
```

- Para cada archivo de entrada y/o salida es obligatorio escribir una sentencia SELECT.

- Tanto la cláusula SELECT como las demás cláusulas, comienzan dentro del margen B y pueden escribirse en cualquier orden.

- Nombre-de-archivo debe ser único dentro de un programa, es decir, no puede haber dos archivos que se llamen de la misma manera, ni se puede emplear dicho nombre para hacer cualquier otra descripción dentro del programa. Su formación sigue las reglas de los nombres creados.

```
001200     SELECT ARCHIVO-OPERARIOS  ASSIGN  SYS007-UR-2540R-S
001210         RESERVE NO ALTERNATE AREA
001220         ACCESS IS SEQUENTIAL.
001230     SELECT CINTA-NOMINA  ASSIGN TO  SYS008-UT-2400-S
001240         RESERVE NO.
001250     SELECT ARCHI-IMPRESO  ASSIGN TO  SYS009-UR-1403-S.
```

En este ejemplo se especifican tres cláusulas SELECT, una para el archivo de los datos de entrada, que el programador ha llamado ARCHIVO-OPERARIOS, y dos para los archivos de salida, denominados CINTA-NOMINA y ARCHI-IMPRESO.

En un programa, aunque figure la descripción de varios archivos de entrada junto con el proceso a realizar con los mismos, no quiere decir que deban estar todos ellos presentes cada vez que se lleve a cabo la ejecución del citado programa, ya que el programador ha podido codificarlos con la intención de utilizar, en cada ejecución, unas veces unos archivos y otras veces otros o todos.

Ante tal posibilidad se debe escribir después de la palabra SELECT, de cada archivo de entrada de los que no vayan a intervenir siempre en el proceso, la palabra OPTIONAL. En este caso cuando se vaya a ejecutar la primera declaración READ del archivo que no va a estar presente en el proceso, no será posible su lectura, por lo que ejecutará la declaración imperativa que sigue a las palabras AT END; es decir, prácticamente es como si ignorara a dicho archivo; continuando normalmente la ejecución del programa.

— La palabra OPTIONAL sólo debe especificarse en archivos de entrada de acceso secuencial.

3.4.1.2.- Cláusula ASSIGN

FUNCION: Asignar un archivo a un determinado dispositivo físico en el cual será leído y/o escrito.

FORMATO:

```
ASSIGN  TO [entero] nombre-de-equipo-1 [nombre-de-equipo-2] ...
        [ FOR MULTIPLE { REEL } ]
                       { UNIT }
```

— Entero se refiere al número de dispositivos físicos que se han asociado con el archivo; por ejemplo, a un archivo multivolumen contenido en tres carretes de cinta, pueden asignarse tres lectoras de cinta, colocando un carrete en cada una, por tanto el valor de Entero sería 3. Normalmente, el valor de Entero lo considera el compilador como comentario, excepto en el caso citado anteriormente de archivos en cinta compuestos de varios volúmenes, en que es necesario especificar dicho valor, ya que el compilador no sabe distinguir entre un final de volumen y un final de archivo, cada vez que termina de procesar un volumen. Se puede omitir Entero cuando un archivo está grabado en un solo carrete.

— Nombre-de-equipo está compuesto por un conjunto de inscripciones que presentan diferencias notables de una a otra marca de ordenador. Se limita su estudio al formato utilizado en el ordenador con el que se llevó a cabo el programa NOMINA.

El Formato de nombre–de–equipo–1 es el siguiente:

SYSnnn–clase–dispositivo–organización–nombre

– SYSnnn, representa el nombre simbólico al que se asocia el archivo; nnn, representan dígitos cuyos valores normalmente no exceden de 200.

Con la asociación del nombre de un archivo a un nombre simbólico, se consigue hacer independiente el archivo, de la unidad física que había de encargarse de leerlo y/o escribirlo, es decir, permite poder colocar el archivo para ser leído y/o grabado en cualquiera de las unidades que en ese momento estén libres. De no emplear la asociación, se haría rígida la asignación de un archivo a una unidad fija, la cual si por cualquier causa estuviera fuera de uso, sería necesario esperar a ponerla en orden o bien modificar el programa escribiendo una nueva cláusula ASSIGN.

– Clase, es una inscripción compuesta de dos caracteres: UR, UT o DA.

- UR (Unit Record) se especifica cuando se utilizan dispositivos físicos que leen, escriben o perforan registros únicos (3) contenidos en un archivo y organizados en forma secuencial (lectoras de fichas, impresoras, etc.).

- UT (Utility) se especifica cuando se utilizan dispositivos físicos que han de leer o grabar registros de un archivo organizado secuencialmente (unidades de cinta o disco).

- DA (Direct Access) se especifica cuando se utilizan dispositivos que, además de leer y grabar registros con organización secuencial, lo pueden hacer con organización al azar (disco, tambor, etc.).

– Dispositivo, está representado por un número de cuatro dígitos e indican el número del modelo de dispositivo dado por el fabricante.

– Organización, es una inscripción constituída por un caracter, el cual especifica el tipo de organización del archivo. Los caracteres que se pueden especificar son:
S - Archivo organizado secuencialmente.
D - Archivo con organización directa.
I - Archivo con organización indexada.

– -Nombre, es una inscripción compuesta de uno a siete caracteres, por la que el compilador reconoce al archivo. Es opcional su escritura.

---

(3) Registros únicos o unitarios son aquellos en que la información que se encuentra perforada sobre una ficha o escrita sobre una línea del impreso, constituyen un solo registro, es decir, en una ficha o en una línea de impreso no puede existir más de un registro.

EJEMPLO:

```
001180     INPUT-OUTPUT SECTION.
001190     FILE-CONTROL.
001200         SELECT ARCHIVO-OPERARIOS   ASSIGN   SYS007-UR-2540R-S
001210             RESERVE NO ALTERNATE AREA
001220             ACCESS IS SEQUENTIAL.
001230         SELECT CINTA-NOMINA   ASSIGN TO   SYS008-UT-2400-S
001240             RESERVE NO.
001250         SELECT ARCHI-IMPRESO   ASSIGN TO   SYS009-UR-1403-S.
```

Para los archivos: ARCHIVO–OPERARIOS y ARCHI–IMPRESO se especifica la clase UR, ya que será una lectora (modelo 2540R) la que leerá el primero de los archivos y una impresora (modelo 1403) la que imprimirá el segundo. Sin embargo, por ser una unidad de cinta (modelo 2400) la que grabará el archivo CINTA–NOMINA se especifica la clase UT.

Para los tres archivos se ha especificado la organización –S– por ser la única con la que se pueden organizar los archivos dentro de los soportes: ficha, cinta e impresora, ya que en ellos los registros están situados o se sitúan en secuencia física (4).

– FOR MULTIPLE REEL/UNIT normalmente se emplea como comentarios y, si se especifica, es para indicar al ordenador que un archivo que se va a leer y/o escribir ocupa más de un volumen, por ejemplo, dos o más carretes de cinta magnética.

Hay que recalcar que las inscripciones que forman parte de la cláusula ASSIGN y el párrafo SPECIAL–NAMES, es donde existen más diferencias de un programa escrito para ser ejecutado en un ordenador, a otro de diferente marca. Por ello, el programador deberá remitirse a las especificaciones del manual del ordenador que haya de utilizar.

3.4.1.3.-Cláusula ACCESS

FUNCION: Indicar la forma en que se va a acceder a la lectura y/o escritura de los registros constituidos en un archivo.

FORMATO:

$$\underline{\text{ACCESS}}\ \text{MODE}\ \underline{\text{IS}}\ \left\{ \begin{array}{l} \underline{\text{SEQUENTIAL}} \\ \underline{\text{RANDOM}} \end{array} \right\} \quad (5)$$

ACCESO es el procedimiento mediante el cual se sitúan o se recuperan subsiguientemente los registros en un dispositivo o soporte de almacenamiento.

---

(4) En otras marcas de ordenadores, en vez de las inscripciones que siguen a la palabra TO, se especifica: CARD–READER, TAPE, PRINTER, etc., según el tipo de dispositivo empleado (lectora de fichas, cinta, impresora, etc.). Por tanto, la cláusula ASSIGN quedaría completa escribiendo: ASSIGN TO CARD–READER o ASSIGN TO TAPE.

(5) El American National Standard Cobol admite otro tipo de acceso denominado DYNAMIC empleado en diversos modelos de ordenadores.

De acuerdo con el formato, dos son los modos de acceso: SECUENCIAL, y DIRECTO o al AZAR, cada uno de los cuales puede ser empleado tanto en el proceso de creación de un archivo como en posteriores procesos con dicho archivo.

Si para la creación del archivo se emplea el acceso secuencial, los registros de SITUAN en el soporte de almacenamiento en secuencia física, es decir, unos a continuación de otros sin dejar espacios intermedios que permitan intercalaciones. Se aplica este modo de acceso a la creación de archivos que van a estar contenidos en soportes como: discos, tambores, cinta, ficha e impresora.

Para que el Sistema (mediante la declaración WRITE) cree el archivo con este modo de acceso, el programador puede especificar ACCESS IS SEQUENTIAL, o también omitir esta cláusula, ya que con ello se consigue el mismo efecto que si se hubiera especificado:

```
001200     SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-UR-2540R-S
001210         RESERVE NO ALTERNATE AREA
001220         ACCESS IS SEQUENTIAL.
001230     SELECT CINTA-NOMINA ASSIGN TO SYS008-UT-2400-S
001240         RESERVE NO.
001250     SELECT ARCHI-IMPRESO ASSIGN TO SYS009-UR-1403-S.
```

es decir, en el programa NOMINA, hubiera sido también correcto haber descrito el archivo que se va a crear, CINTA–NOMINA, de la forma siguiente:

```
001230     SELECT CINTA-NOMINA ASSIGN TO SYS008-UT-2400-S
001240         RESERVE NO ACCESS MODE IS SEQUENTIAL.
```

Si para la creación del archivo se emplea el acceso directo, los registros se sitúan físicamente en el soporte en orden no secuencial. En este caso se debe especificar: ACCESS IS RANDOM. Este modo de acceso se emplea en la creación de archivos que van a ser grabados en disco o tambores magnéticos (ver capítulo VII).

Una vez creado un archivo habrá que realizar con sus registros procesos que requerirán acciones como leer, actualizar, etc., para lo cual será necesario OBTENER dichos registros desde el soporte de almacenamiento donde se encuentran e introducirles en la memoria, a fin de realizar con ellos el adecuado tratamiento. Al igual que en la creación del archivo, se emplearán para su posterior proceso los dos modos de acceso citados.

Con acceso secuencial cada uno de los sucesivos accesos al almacenamiento, extrae el registro inmediato siguiente. Así, en el programa NOMINA, al especificar en ARCHIVO–OPERARIOS la cláusula ACCESS IS SEQUENTIAL (línea 001220) el Sistema irá obteniendo (mediante la declaración READ) sucesivamente las fichas que componen dicho archivo en el orden en que fueron colocadas.

Con acceso directo o al azar los registros del archivo se obtienen en forma no secuencial, y sólo es aplicable a archivos en disco o tambor magnético (ver capítulo VII).

3.4.1.4.-Cláusula RESERVE

FUNCION: Indicar al compilador el número de áreas de E/S adicionales que el programador ha decidido se reserven para cada archivo, además del área de entrada o salida normal que se requiere para cada uno de ellos. Estas áreas adicionales reciben el nombre de ALTERNATIVAS.

FORMATO:

```
RESERVE { entero } ALTERNATE [ AREA  ]
        { NO    }            [ AREAS ]
```

- Entero, cuando se especifique, ha de ser el valor 1 (5).
- Si se especifica RESERVE NO, no se reserva ningún área alternativa de memoria.
- Si se omite esta cláusula o se especifica RESERVE 1, el compilador reserva un área alternativa además de la normal.
- La opción RESERVE 1, se emplea para archivos secuenciales contenidos en cinta, ficha impreso o discos.

Area alternativa de memoria es aquélla en la que se introduce, por lectura, un bloque de registros, o desde la cual se escribe un bloque. Es, simplemente, otro área de entrada y/o salida.

Reservar áreas alternativas agiliza la ejecución del programa, ya que permite el proceso solapado; es decir, que simultáneamente a la escritura de un registro que ha sido procesado, se esté leyendo el registro siguiente para su proceso.

La conveniencia de reservar áreas alternativas es criterio del programador, teniendo en cuenta, entre otras cosas, la capacidad de la memoria. Si ésta es de gran capacidad, normalmente se pueden utilizar dos áreas de entrada para cada archivo de entrada y dos áreas de salida para cada archivo de salida; pero en el caso de que si utilizando dos áreas de entrada/salida, no se puede ejecutar el programa por falta de capacidad de la memoria del ordenador, se resuelve el problema especificando la opción RESERVE NO.

En el ejemplo Nómina, se reserva en la memoria del ordenador un área de entrada (normal) para ARCHIVO-OPERARIOS y CINTA-NOMINA y dos áreas de salida (normal y alternativa) para ARCHI-IMPRESO. Al especificar la cláusula RESERVE, sólo se han perseguido fines didácticos. Se hubiera disminuído el tiempo de ejecución del programa reservando un área alternativa para cada archivo, lo que se hubiera conseguido omitiendo la escritura de la cláusula RESERVE en todas ellas o especificando RESERVE 1.

Las cláusulas ACTUAL KEY, NOMINAL (SYMBOLIC) KEY y RECORD KEY se estudiarán en capítulos posteriores.

---

(5) Existen modelos de ordenadores que permiten reservar más de un área alternativa de entrada/salida.

3.4.2.-Párrafo I-O-CONTROL

FUNCION: Definir técnicas especiales de entrada / salida (E/S).

Especificar que dos o más archivos van a compartir un área de la memoria.

Registrar puntos de control a partir de los cuales poder reanudar el programa (caso de producirse una interrupción en el mismo por avería , datos erróneos, etc.).

FORMATO:

```
[ I-O-CONTROL.
      [Cláusula  RERUN] ...
      [Cláusula  SAME]  ...
      [Cláusula  APPLY] ... ] .
```

La cláusula APPLY se describe en las páginas 294 y 324.

3.4.2.1.-Cláusula RERUN

FUNCION: Especificar al compilador que se deben grabar registros de puntos de control (6).

Los registros de puntos de control en algunos tipos de trabajos son necesarios y recomendables; por ejemplo, en la ejecución de un programa en el que se hayan de ir acumulando resultados parciales para obtener al final del trabajo un total general. En la realización de este trabajo podía suceder que cuando se llevaran procesados un número elevado de registros, el ordenador, por cualquier causa, como pudiera ser por un error en los datos de entrada, o defeciente funcionamiento de alguno de los elementos físicos, interrumpiera la ejecución del programa, en este caso sería necesario volver a iniciar el trabajo, ya que los campos que contienen dichos resultados parciales, y debido a la interrupción no conservarían su valor, por lo que resultaría inútil todo el trabajo que anteriormente se hubiera realizado. Para evitarlo se emplea esta cláusula, la cual hace que cada determinado número de registros (a criterio del programador) leídos o escritos vaya grabando sobre una cinta o disco registros de puntos de control, con lo que al sufrir una interrupción, se puede restaurar la memoria con el contenido del último punto de control y así continuar normalmente el trabajo.

Puede también reiniciarse el trabajo a partir de otro punto de control distinto del último.

Cada vez que se graba un registro de punto de control, aparece un mensaje en la consola del operador informando que se ha grabado dicho registro, por lo que el se podrá conocer cuantos se han grabado durante la ejecución del trabajo o hasta la interrupción del mismo.

---

(6) Un registro de punto de control es aquél que guarda en un soporte externo (cinta, disco, etc.) el contenido de cada una de las posiciones de la memoria del ordenador en un momento dado durante la ejecución del programa.

FORMATO:

RERUN ON nombre - de - equipo

EVERY entero RECORDS OF nombre - de -archivo

- Nombre-de-equipo tiene la misma estructura que se describió en la cláusula ASSIGN.
- Entero indica el número de registros que se han de procesar antes de escribir un registro de punto de control.

```
002010     I-O-CONTROL.
002020         RERUN ON SYS010-UT-2400-S EVERY 125 RECORDS
002030             OF ARCHIVO-OPERARIOS.
```

El empleo de esta cláusula dentro del programa Nómina permite registrar puntos de control en una cinta cada vez que se procesen 125 registros del ARCHIVO–OPERARIOS.

Es importante observar que el nombre simbólico SYSnnn debe ser distinto al de cualquiera de los archivos especificados dentro de FILE–CONTROL, no debiéndose emplear en este párrafo una cláusula SELECT para especificar el nombre del archivo donde se grabarán los puntos de control, ni se debe,por tanto, hacer su descripción dentro de la DATA DIVISION.

3.4.2.2.- Cláusula SAME

FUNCION: Indicar que dos o más archivos compartirán el mismo área de la memoria del ordenador durante el proceso.

FORMATO:

SAME [**SORT** / **RECORD**] **AREA FOR nombre-de-archivo-1** { **nombre-de-archivo-2** } ...

La cláusula SAME fué concebida para optimizar el empleo de la memoria del ordenador. Su uso permite el ahorro de memoria, ya que evita que el compilador reserve por cada archivo de entrada / salida un área de entrada / salida (ver pág. 29), y en su lugar reserve únicamente un área de la memoria que será compartida por todos los archivos que intervengan en el proceso.

En ordenadores de pequeña capacidad de memoria es aconsejable y a veces necesario el empleo de dicha cláusula, a fin de hacer posible la ejecución de programas que por su extensión excederían de los límites de la memoria y por tanto hacer imposible su ejecución.

En ordenadores de gran capacidad de memoria es prácticamente innecesario su empleo.

En el programa que no se especifique la cláusula SAME, las áreas de memoria (E/S) requeridas para el proceso de los archivos se obtienen, durante la ejecución, cuando se abre el archivo y se liberan cuando se cierra. Sin embargo:

– Si se especifica SAME RECORD AREA todos los archivos que aparecen nombrados en la cláusula pueden abrirse al mismo tiempo, sin que se obtenga o reserve un área de E/S para cada archivo, sino que el compilador reservará solamente un área independiente que compartirán todos los archivos. En dicho área puede residir únicamente un registro lógico que pertenecerá, caso de existir varios archivos de entrada, al último que haya leído, el cual podrá ser utilizado por los archivos de salida. Prácticamente, esta cláusula realiza una redefinición de los registros lógicos de cada archivo que interviene en el proceso, lo cual permite escribir, el mismo registro que se acaba de leer, en uno o varios archivos sin necesidad de un movimiento (MOVE) previo a las áreas de salida.

EJEMPLO : Codificar un programa que sirva para crear dos archivos en cinta magnética. Cada archivo debe contener todos los registros de ARCHIVO–OPERARIOS.

```
001010  IDENTIFICATION DIVISION.
001020    PROGRAM-ID. SAMERECO.
001030  *ESTE PROGRAMA ES UN EJEMPLO DE EMPLEO DE LA CLAUSULA SAME
001040  *GRABA EN DOS CINTAS EL CONTENIDO DEL ARCHIVO-OPERARIOS.
001050  ENVIRONMENT DIVISION.
001060    INPUT-OUTPUT SECTION.
001070      FILE-CONTROL.
001080          SELECT ARCHIVO-OPERARIOS   ASSIGN  SYS007-UR-2540R-S.
001090          SELECT ARCHIVO-CINTA1      ASSIGN  SYS008-UT-2400-S.
001100          SELECT ARCHIVO-CINTA2      ASSIGN  SYS009-UT-2400-S.
001110      I-O-CONTROL.
001120          SAME RECORD AREA FOR  ARCHIVO-OPERARIOS  ARCHIVO-CINTA1
001130                                ARCHIVO-CINTA2.
001149  DATA DIVISION.
001150    FILE SECTION.
001160      FD ARCHIVO-OPERARIOS   RECORDING F  LABEL RECORD OMITTED
001170                             DATA RECORD FICHA-OPERARIO.
001180      01 FICHA-OPERARIO      PIC X(80).
001190      FD ARCHIVO-CINTA1      RECORDING F  LABEL RECORD OMITTED
001200                             BLOCK 10 RECORDS  DATA RECORD REGCINTA1.
001210      01 REGCINTA1           PIC X(86).
001220      FD ARCHIVO-CINTA2      RECORDING F  LABEL RECORD STANDARD
001230                             BLOCK 15 RECORDS  DATA RECORD REGCINTA2.
001240      01 REGCINTA2           PIC X(77).
002010  PROCEDURE DIVISION.
002020    ABRIR.
002030          OPEN  INPUT  ARCHIVO-OPERARIOS
002040                OUTPUT ARCHIVO-CINTA1   ARCHIVO-CINTA2.
002050    LEER.
002060          READ  ARCHIVO-OPERARIOS
002070                AT END  CLOSE  ARCHIVO-OPERARIOS
002080                               ARCHIVO-CINTA1   ARCHIVO-CINTA2
002090                        STOP RUN.
002100          WRITE  REGCINTA1
002110          WRITE  REGCINTA2
002120          GO TO  LEER.
```

– Si se especifica SAME AREA los archivos no pueden estar todos abiertos a la vez, ya que el área que se compartiría serían todas las áreas requeridas por los archivos que intervendrán en el proceso y, por tanto, no se ahorraría memoria. El empleo de esta opción tiene el inconveniente de que por no poder abrirse todos los archivos a la vez, limita las posibilidades del proceso.

– La opción SAME SORT optimiza la asignación de memoria para las declaraciones SORT. Su empleo es prácticamente nulo, ya que el sistema gestiona automáticamente la asignación de memoria en un trabajo de clasificación de archivos (ver capítulo VIII). El compilador la toma como simples comentarios.

4.- EJERCICIOS

1.— ¿Cuál de las siguientes inscripciones no comienza en el Margen A?

A.- FILE-CONTROL
B.- INPUT-OUTPUT SECTION
C.- SPECIAL-NAMES
D.- SELECT

2.— En un programa que utiliza dos archivos de entrada y tres de salida ¿cuántas cláusulas SELECT son necesarias?

A.- Una para todos los archivos de entrada y otra para todos los de salida.
B.- Una para cada archivo de entrada y dos para cada archivo de salida.
C.- Una para cada archivo de entrada y otra para cada archivo de salida.
D.- Una que incluya todos los archivos.

3.— Todos los nombres de párrafo que componen la Identification Division, a excepción de PROGRAM-ID deben comenzar en el Margen B.

CIERTO FALSO

4.— Al escribir el programador las Divisiones: IDENTIFICATION y ENVIRONMENT correspondientes al programa NOMINA, ha cometido errores en algunas de las líneas. Decir en qué consisten dichos errores.

```
001010  IDENTIFICACION DIVISION.
001020      PROGRAM-ID. NOMINA.
001030      AUTOR. PROGRAMADOR C. DE LA FUENTE.
001040      INSTALLATION. CENTRO PROCESO DE DATOS DE RICOSA.
001050      DATE-WRITEN. 15-MARZO-1977.
001060      DATE-COMPILED.
001070      SEGURITY. ESTE PROGRAMA SOLO PUEDE SER UTILIZADO POR LA
001080                SECCION DE CONTABILIDAD.
001090      REMARKS. PROGRAMA PARA OBTENER LA NOMINA DE RICOSA Y CREAR
001100                UN ARCHIVO EN CINTA PARA POSTERIORES TRABAJOS.
001110  ENVIRONMENT DIVISION
001120      CONFIGURATION SECCION.
001130      SOURCE-COMPUTER. IBM-370.
001140      OBJECT-COMPUTER. IBM-370.
001150      SPECIAL-NAMES.
001160          C01 IS PAG-NUEVA
001170          DECIMAL-POINT IS COMA.
001180      IMPUT-OUTPUT SECTION.
001190      FILE CONTROL.
001200          SELECT ARCHIVO-OPERARIOS ASSING SYS007-UR-2540R-S
001210              RESERVE NO ALTERNATE AREA
001220              ACCES IS SECUENTIAL.
001230          SELECT CINTA-NOMINA ASSIGN TO SYS008-UT-2400-S.
001240              RESERVE NO.
001250          SELECT ARCHI IMPRESO ASSIGN TO SYS009-UR-1403-S.
```

5.— De las inscripciones que forman la ENVIRONMENT DIVISION del programa NOMINA, decir qué líneas podían suprimirse,por ser opcional su escritura, sin que por ello sufriera alteración el proceso o los resultados durante la ejecución de dicho programa.

6.— ¿Son correctas las siguientes descripciones tomadas del programa NOMINA?

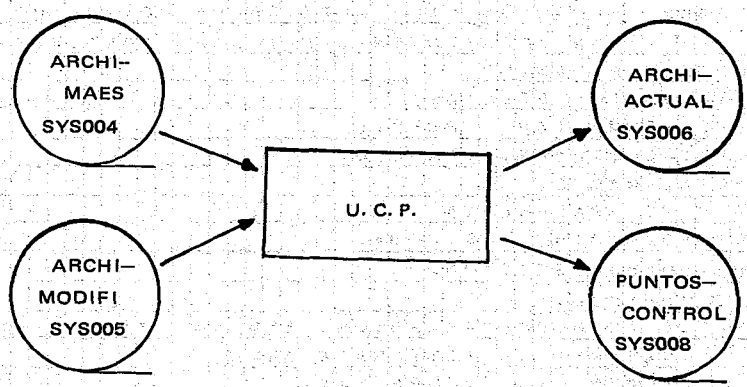
```
001180  INPUT-OUTPUT SECTION.
001190  FILE-CONTROL. SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-UR-2540R-S
001200      RESERVE NO ALTERNATE AREA ACCESS IS SEQUENTIAL.
```

7.— Se precisa escribir las divisiones: IDENTIFICATION y ENVIRONMENT, de un programa del que se conocen los datos siguientes:

En el Centro de Proceso de Datos de RICOSA se actualiza diariamente un archivo maestro denominado ARCHI—MAES, grabado en cinta magnética, cuyo contenido son las existencias en almacén de las piezas de repuesto para automóviles fabricados en la factoría COIRSA. Todas las altas, bajas y variaciones a introducir en ARCHI—MAES, se encuentran en otro archivo denominado ARCHI—MODIFI, grabado como el anterior en cinta magnética.

El archivo resultante de la actualización se denominará ARCHI—ACTUAL y se grabará en una nueva cinta.

- El nombre identificativo del programa será: ACTUALIZ.
- Cada 100 registros que se graben en ARCHI—ACTUAL, se tomarán registros de puntos de control grabándolos sobre otra cinta.
- El modelo de ordenador para compilar y ejecutar,así como los modelos de Unidades de Lectura/Escritura de cinta, son los que dispone el Centro de Proceso RICOSA.



8.— ¿Es necesario en el programa anterior especificar el párrafo SPECIAL—NAMES? Razonar la contestación.

CAPITULO IV

DATA DIVISION

1.- GENERALIDADES

Normalmente, la primera operación a realizar durante un proceso es introducir los datos en la memoria del ordenador, para lo cual es necesario proveer al compilador de información en la que se especifique la organización de los archivos y registros, con el fin de que reserve zonas de la memoria donde ir sucesivamente introduciendo dichos datos, así como asignar un nombre a cada una de estas zonas, a fin de que sea capaz de reconocerlas e identificarlas cada vez que se las mencione dentro de la PROCEDURE DIVISION. Esta información se le proporciona al compilador por medio de las descripciones que se hacen en la DATA DIVISION.

Cada descripción de la organización de un archivo comprende, entre otras cosas, información al compilador acerca de:

- Si los registros del archivo están o no agrupados en bloques.
- Si los registros son de longitud fija, variable o indefinida.
- Si en cada archivo hay grabadas etiquetas identificadoras del mismo.
- Nombre del archivo y de los registros que lo componen.etc.

Cada descripción de la organización de un registro informará al compilador sobre:

- Orden en que están dispuestos los campos de datos dentro del registro.
- Nombre dado a cada campo con el que poder referirse a él en la PROCEDURE DIVISION.
- Número y tipo de caracteres que puede almacenar cada campo, etc.

Por tanto, en base a estas descripciones, el compilador reserva el continente, es decir, las posiciones de memoria donde se irá colocando el contenido, que son los datos y/o resultados.

2.- ESTRUCTURA

```
DATA DIVISION

[FILE SECTION.
    {Entrada de descripción del archivo. . . .
    {Entrada de descripción del registro } . . . ]

[WORKING-STORAGE SECTION.
    [Entrada de descripción de campos individuales] . . .
    [Entrada de descripción de registros] . . . ]

[LINKAGE SECTION.
    [Entrada de descripción de campos] . . .
    [Entrada de descripción de registro.] . . . ]

[REPORT SECTION.
    {Entrada de descripción de informe.
    {Entrada de descripción de grupo de informe } . . . ]
```

Todas las secciones son opcionales y, por tanto, pueden omitirse aquéllas que no sean necesarias. Caso de utilizarse, deben especificarse en el orden descrito.

En la FILE SECTION se describe la organización de cada archivo de entrada/salida, así como la de los registros que lo componen.

En la WORKING-STORAGE SECTION se describe la organización de los registros que no forman parte de archivos de entrada/salida y de los campos individuales (que no forman parte de un registro).

En la LINKAGE SECTION se describen registros para proporcionar nombres ficticios que sirvan para enlazar los datos del programa principal con el subprograma.

En la REPORT SECTION se especifica el formato de un informe que se ha de imprimir, minimizando el número de declaraciones en la PROCEDURE DIVISION.

Con las descripciones que se hagan en la FILE SECTION el compilador reserva áreas de la memoria donde se colocarán los datos que estén perforados o grabados en un soporte externo (cinta, disco, etc.).

Con las descripciones de la WORKING-STORAGE SECTION, el compilador reserva áreas de la memoria para situar datos o resultados que se creen internamente como consecuencia del proceso, o bien para almacenar datos constantes durante la ejecución del trabajo, por inicialización de estas áreas por el programador.

3.- FILE SECTION

La FILE SECTION describe la organización de cada archivo y de los registros que lo componen.

3.1.- Descripción de ARCHIVOS

Cada descripción de archivo proporciona, dentro de las cláusulas que componen el siguiente formato, información al compilador acerca de la organización del archivo.

FORMATO:

```
FD  nombre-de-archivo
    [Cláusula  BLOCK    CONTAINS ]
    [Cláusula  RECORD   CONTAINS ]
    [Cláusula  RECORDING MODE ]
     Cláusula  LABEL RECORDS
    [Cláusula  DATA RECORDS ]
    [Cláusula  REPORT ] .
```

- Excepto FD nombre-de-archivo y la cláusula LABEL, el resto de las cláusulas son opcionales (7).
- Nombre-de-archivo debe ser único, es decir, no puede haber dos archivos que, en un programa, se denominen de la misma forma.
- FD (FILE DESCRIPCION) se emplea para indicar el comienzo de la descripción de la organización de un archivo. Debe comenzar en el margen A e ir seguido por el nombre del archivo en el margen B. El resto de las cláusulas comienzan en el margen B y pueden especificarse en cualquier orden.

A continuación se describen las cláusulas que constituyen la descripción del archivo, excepto la cláusula REPORT que se describe en el Capítulo VIII.

---

(7) Aunque algunos tipos de ordenadores emplean compiladores para los que es necesario especificar en el programa todas las cláusulas; sin embargo, la mayoría de los compiladores actuales prescinden de las cláusulas: RECORD CONTAINS, RECORDING MODE y DATA RECORDS.

3.1.1.- Cláusula RECORDING MODE

FUNCION: Especificar el formato de los registros lógicos dentro del soporte físico.

FORMATO:

RECORDING MODE IS modo

- El modo puede ser F, V o U.
- El modo F es el más empleado y se especifica cuando los registros lógicos del archivo son de la misma longitud. Dichos registros puede agruparse formando bloques (registros físicos) conteniendo cada uno el mismo número de registros.
- El modo V se especifica cuando los registros lógicos del archivo son de igual o distinta longitud (longitud variable). Pueden agruparse formando bloques de igual o distinto número de registros.
- El modo U se especifica en archivos en los que los registros lógicos que contienen pueden ser de igual o distinta longitud. No pueden agruparse en bloques.

Cuando se utilice la cláusula RECORDING en la descripción de un archivo que esté contenido en fichas perforadas o, que haya de imprimirse, hay que especificar el modo F, ya que cada ficha o cada línea a imprimir se considera como un registro lógico de longitud fija de 80 y 132 caracteres respectivamente (aunque no estén llenos de información).

```
002060     FD ARCHIVO-OPERARIOS   RECORDING MODE IS F
002070         LABEL RECORD IS OMITTED
```

Sin embargo, en cinta magnética y discos etc., al poder agrabarse registros de la misma o diferente longitud, el modo puede ser F, V o U según decida el programador:

```
003180     FD CINTA-NOMINA    RECORDING F
003190         BLOCK 10 RECORDS
```

La diferencia entre el modo V y los F y U, es que el primero, durante la creación del archivo, el Sistema incorpora delante de cada registro un campo de cuatro octetos que le sirve para control, a fin de conocer la longitud de cada registro, así como otro campo delante de cada bloque para indicar su longitud. Sin embargo en los modos F y U el Sistema no incorpora ningún campo delante de cada registro.

Una vez creado un archivo con modo V, el campo de cuatro octetos que precede a cada registro no se debe especificar al hacer la descripción del mismo, ya que dicho campo sólo sirve de información al compilador, pero no como información para ser utilizada por el programador.

Es importante tener presente que el modo que se especifique durante la creación de un archivo, no debe variarse cuando se utilice este archivo en otro programa que realice otro trabajo.

3.1.2.- Cláusula BLOCK CONTAINS

FUNCION: Especificar el número de registros lógicos contenidos en un registro físico (bloque), o el número de caracteres contenidos en un registro físico.

FORMATO:

BLOCK CONTAINS entero $\left\{\begin{matrix}\text{CHARACTERS}\\ \text{RECORDS}\end{matrix}\right\}$

- Cuando los registros no estén grabados en bloques, es decir, cuando cada registro físico contiene un solo registro lógico, puede omitirse esta cláusula; en los demás casos es necesaria. Por tanto, se puede omitir en archivos en fichas o impreso por no poderse agrupar en bloques, pero es necesaria para archivos con registros grabados, o que se hayan de grabar, en bloques en cinta magnética, disco etc.

- Se debe omitir en archivos de modo U por no poder agruparse en los registros en bloque.

- Cuando se emplee la opción RECORDS, la palabra entero se referirá al número de registros contenidos en cada bloque.

- Cuando se emplee la opción CHARACTERS, la palabra entero se referirá al número de octetos que contiene el bloque más largo. Es el caso de registros grabados en la modalidad V.

- Si no se especifica RECORDS ni CHARACTERS, el compilador lo asocia a CHARACTERS.

EJEMPLO:

```
003180     FD  CINTA-NOMINA     RECORDING F
003190         BLOCK   10   RECORDS
003200         RECORD  86   CHARACTERS
```

De acuerdo con las reglas citadas, se ha omitido esta cláusula en la descripción de ARCHIVO–OPERARIO, por ser un archivo perforado en fichas, así como en la descripción de ARCHI–IMPRESO, por ser un archivo que va a ser escrito sobre papel de la impresora.

Sin embargo, es necesario especificar dicha cláusula en el archivo CINTA–NOMINA, ya que los registros se grabarán en la cinta magnética agrupados en bloques de diez. El número de registros por bloque lo fija el programador, teniendo en cuenta, entre otros conceptos: capacidad de la memoria, longitud de cada registro, etc.

La opción RECORDS, es la de uso más general y sencillo, ya que si el programador utiliza la opción CHARACTERS, deberá tener en cuenta, para fijar el valor de Entero, el Modo de los registros. En el ejemplo anterior como todos los registros de CINTA–NOMINA son de Modo F, están agrupados en bloques de 10 y cada registro tiene una longitud fija de 86 octetos, el valor de Entero sería: 86 x 10 = 860, por lo que se describirá la cláusula de cualquiera de las formas siguientes:

```
190    BLOCK   860
200    BLOCK   CONTAINS   860
210    BLOCK   CONTAINS   860   CHARACTERS
```

Si en un archivo los registros están grabados o se van a grabar con Modo V, y con factor de agrupamiento uno (8), el valor de Entero será el valor del registro mayor, más los cuatro octetos que le preceden y que son indicativos de la longitud de registro. Es decir, si hubiera en el archivo tres tipos de registros de longitudes 80, 100 y 140 respectivamente, el programador deberá especificar: BLOCK CONTAINS 144 CHARACTERS. Si el factor de agrupamiento fuera mayor de uno, por ejemplo, cinco, y los registros fueran de la longitud citada anteriormente, el programador haría el cálculo del modo siguiente: tomaría el valor del registro de mayor longitud, le sumaría los cuatro octetos que preceden a dicho registro, lo multiplicaría por el factor de agrupamiento y le sumaría los cuatro octetos que preceden a cada bloque, es decir,(140 + 4) . 5 + 4 = 724, con lo que la cláusula BLOCK CONTAINS la describiría de la forma siguiente:

```
140    BLOCK   CONTAINS   724 CHARACTERS
```

De todo este cálculo se deduce que es complicado el empleo de la opción CHARACTERS, prestándose a errores y por tanto,preferible el empleo de la opción RECORDS.

Si en la creación de un archivo se especificó un determinado factor de agrupamiento, este mismo factor debe especificarse cuando se confeccione un nuevo programa en el que intervenga este archivo para la realización de otro trabajo.

---

(8) Es otra forma de denominar al número de registros lógicos que contiene cada bloque.

3.1.3.- Cláusula RECORD CONTAINS

FUNCION: Especificar el tamaño de los registros lógicos.

FORMATO:

RECORD CONTAINS entero CHARACTERS

Normalmente esta cláusula se omite, y en ese caso el compilador se encarga de realizar su cometido, explorando la longitud de cada registro a partir de su descripción.

Por tanto,de acuerdo con la descripción hecha en FICHA–OPERARIO, el compilador sumaría los octetos asignados a cada campo y reservaría un área de memoria de 80 octetos. No obstante,podía haberse especificado en el programa:

```
002060     FD  ARCHIVO-OPERARIOS  RECORDING MODE  IS F
002070         RECORD  CONTAINS  80 CHARACTERS
002080         LABEL  RECORD  IS  OMITTED
```

La ventaja de especificar esta cláusula es la de reducir el tiempo de compilación.

El valor máximo de entero depende del tipo de compilador empleado, actualmente no suele ser superior a 4092.

3.1.4.- Cláusula DATA RECORD

FUNCION: Dar un nombre a cada formato distinto de los registros lógicos contenidos en el archivo.

FORMATO:

DATA { **RECORD IS** / **RECORDS ARE** } **nombre–de–datos–1** [**nombre–de–datos–2**]...

– Es opcional en cada descripción de archivo y sólo sirve como documentación.

– Normalmente,todos los registros de un archivo son del mismo formato, por lo que en la cláusula aparecerá un solo nombre de registro; pero si el archivo tiene registros con distinto formato, cada uno de ellos tendrá un nombre de registro diferente, aunque implícitamente

compartirán la misma zona de la memoria.

Nombre–de–datos–1, nombre–de–datos–2,... se refieren al nombre de los registros y deberán ir precedidos del número de nivel 01.

```
004010     FD   ARCHI-IMPRESO    RECORDING IS F
004020          LABEL RECORDS ARE OMITTED
004030          DATA  RECORD  IS IMPRESO.
004040     01   IMPRESO                PICTURE X(133).
```

En la cláusula DATA sólo se ha especificado un nombre de registro (IMPRESO).

En el siguiente ejemplo, se especifican tres nombres de registro, cada uno con el mismo número de caracteres (133), y aunque el formato es diferente, comparten la misma área de memoria por pertenecer al mismo archivo.

EJEMPLO:

Hoja de Codificación en Cobol

PROGRAMA:
PROGRAMADOR:
INSTRUCCIONES PARA PERFORACION — SIGNO / PERFORACIONES — HOJA DE / SISTEMA / FECHA

Página | Línea | Cont | A | B

```
010     DATA DIVISION.
030     FD   ARCHI-IMPRESO    RECORDING  F
040          LABEL  RECORD OMITTED
050          DATA RECORD   IS   IMPRESO, LINEAIM,  REGISIM.
060
070     01   IMPRESO                PICTURE  X(133).
080
090     01   LINEAIM.
100          02   FILLER            PICTURE  X(43).
110          02   APENOM            PICTURE  X(35).
120          02   FILLER            PICTURE  X(30).
130          02   SUELDO-TOT        PICTURE  9(5).
140          02   FILLER            PICTURE  X(20).
150
160     01   REGISIM.
170          02   FILLER            PICTURE  X(10).
180          02   CODIGO            PICTURE  X(5).
190          02   FILLER            PICTURE  X(20).
200          02   HORAS             PICTURE  9(3).
210          02   FILLER            PICTURE  X(95).
```

El programador, al escribir la PROCEDURE DIVISION, puede utilizar uno u otro nombre de registro, según los resultados que desee imprimir.

3.1.5.- Cláusula LABEL RECORD

FUNCION: Especificar la existencia o no de etiquetas asociadas a un archivo. Cada etiqueta constituye un registro de 80 caracteres, pueden ir al comienzo y final de cada archivo y sirven como identificación del mismo.

FORMATO:

```
LABEL { RECORD IS  } { OMITTED          }
      { RECORDS ARE} { STANDARD         }
                     { nombre-de-datos-1 ... }
```

- Esta cláusula es obligatoria para toda clase de archivos (fichas, cintas, discos, etc.).

- La opción OMITTED debe especificarse para archivos contenidos en fichas o impreso. El efecto de esta opción, es que no se graban etiquetas en el archivo que se va a crear, o en un archivo ya creado, este carece de etiquetas.

- La opción STANDARD debe especificarse para archivos contenidos en dispositivos de acceso directo (disco, tambor, etc.), indicando al compilador que en el archivo que se va a crear, deben grabarse etiquetas Standard, o que el archivo, ya creado, tiene grabadas dicho tipo de etiquetas (9).

  Las opciones OMITTED o STANDARD pueden especificarse para archivos grabados en cinta magnética.

- La opción nombre-de-datos sólo puede especificarse para archivos secuenciales en cinta y disco. La función de esta opción es procesar otra clase de etiquetas que son creadas por cada programador, por lo que su formato no es fijo como en las Standard.
  La descripción de este tipo de etiquetas debe hacerse dentro de la LINKAGE SECTION.

  EJEMPLO:

```
003180     FD  CINTA-NOMINA     RECORDING F
003190         BLOCK  10  RECORDS
003200         RECORD  86  CHARACTERS
003210         LABEL  RECORD  OMITTED
```

En este ejemplo se ha elegido la opción LABEL RECORD OMITTED, por lo que no se grabarán etiquetas. Si en un trabajo posterior se utilizara este archivo como archivo de entrada, se deberá especificar nuevamente LABEL RECORD OMITTED, ya que de lo contrario el ordenador, al comienzo del proceso, buscaría la etiqueta para comprobarla y al no encontrarla detendría la ejecución del programa.

---

(9) Los ochenta posiciones o caracteres de que consta una etiqueta Standard, cuyo formato es invariable, están repartidos en una serie de campos en los que el programador debe incluir la información que para ello se indica en cada manual de programación del modelo de ordenador a utilizar.

3.2.- Descripción de REGISTROS

La descripción de un registro sirve para especificar, como la información contenida en él, está organizada en campos. Para ser procesada esta información, ya esté perforada o grabada en un determinado soporte, es necesario introducirla en la memoria del ordenador, para lo cual es preciso informar al compilador acerca de las características de cada campo de datos, por ejemplo: número de caracteres de que consta, posición que ocupan dentro del registro etc., a fin de que reserve áreas de la memoria capaces de contener la información de los citados campos.

La descripción de un registro está constituída por una o varias inscripciones jerarquizadas y compuestas cada una de ellas por un número de nivel seguido de un nombre de datos y, opcionalmente, de una serie de cláusulas.

El formato de cada inscripción es el siguiente:

FORMATO:

número–de–nivel { **nombre–de–datos** / **FILLER** }

[**Cláusula REDEFINES**]
[**Cláusula PICTURE**]
[**Cláusula VALUE**]
[**Cláusula JUSTIFIED RIGHT**]
[**Cláusula BLANK WHEN ZERO**]
[**Cláusula OCCURS**]
[**Cláusula USAGE**]
[**Cláusula RENAMES**] .

- A excepción de la cláusula REDEFINES, que, cuando se emplee, debe ir inmediatamente después de nombre–de–datos, las demás cláusulas pueden escribirse en cualquier orden.

- Toda inscripción debe terminar en un punto.

Observando la figura 4 y la descripción que se hace en el programa Nómina del registro FICHA–OPERARIO, se advierte una estructura en niveles, en la que el nombre del registro va acompañado del número de nivel más alto: 1 ó 01 (es indiferente). Dentro del registro aparecen una serie de grupos de campos, también denominados campos compuestos, descritos con un número de nivel inferior, 02, como: OPERARIO y FACTORIA, los cuales a su vez engloban grupos de campos con otro nivel inferior como: FECHANACI, FECHAINGRESO y DEPARTAMENTO, y otros campos que ya no forman grupos y a los que se denomina campos elementales como: CODIGO, APELL–NOMBRE, ESTADO, etc.

OPERARIO																FACTORIA		
CODIGO	APELLIDOS Y NOMBRE	fecha nacimiento			prov. naci.	estado	sexo	num. hijos	fecha de ingreso			SUELDO	tipo desc.	horas ex. num.		NOMBRE	dep.	
		día	mes	año					día	mes	año						num.	sec.

```
00209001 FICHA-OPERARIO.
00210002   OPERARIO.
00211003     CODIGO          PICTURE X(5).
00212003     APELL-NOMBRE    PICTURE X(35).
00213003     FECHANACI.
00214004       DIA           PICTURE 99.
00215004       MES           PICTURE 99.
00216004       ANNO          PICTURE 99.
00217003     PROVINCIA-NACIM PICTURE 99.
00218003     ESTADO          PICTURE 9.
```

La estructura en niveles permite trabajar con registros completos, con campos compuestos y con campos elementales. Por ejemplo, en la PROCEDURE DIVISION, el programador, al especificar la declaración de la línea 008150: MOVE FICHA–OPERARIO TO DATOSCIN, opera con un registro completo, y transfiere, con una sóla declaración, todos los datos contenidos en la ficha de cada operario al campo elemental DATOSCIN; mientras que si especifica (línea 008020): MOVE CODIGO TO CODIGO–S, lo que está haciendo es transferir el contenido de un campo elemental CODIGO del registro FICHA–OPERARIO, al campo de edición CODIGO–S.

Los números de nivel que se han especificado en la descripción de los diferentes campos que componen FICHA–OPERARIO, son sucesivos, más ello no es necesario, sino que lo preceptivo es asignar números de niveles superiores (10) a los grupos que menos se subdividen, sin que se sobrepase el nivel 49 para los campos elementales. Este número de nivel, 49, permite hacer en un registro cuarenta y nueve subdivisiones de campos; cantidad que se considera suficiente para cualquier tipo de registro en cualquier trabajo.

La estructuración jerárquica del registro en campos compuestos y elementales es responsablilidad del programador, y con ella debe buscar en todo momento la eficiencia en la ejecución del programa, bien por la disminución del número de declaraciones dentro de la PROCEDURE DIVISION, o bien por eliminación de inscripciones innecesarias de campos de datos en la DATA DIVISION.

---

(10) Hay que hacer notar que los números de nivel se consideran descendentes desde el nivel 01 al nivel 49; es decir, el número de nivel 17 es superior al 25.

3.2.1.- Números de NIVEL

FUNCION: Proporcionar información acerca de la relación entre los campos de datos del registro.

— Los números de nivel que pueden utilizarse son: del 01 al 49, 66, 77 y 88 . Se considera a 01 como numero de nivel superior, seguido de 02, 03 etc.

- El número de nivel 01 sólo puede preceder al nombre de registro. El resto de los números de nivel deben preceder al nombre de cada campo de datos.

- El número de nivel 66 se emplea únicamente con la cláusula RENAMES.

- El número de nivel 77 se emplea cuando se describen campos independientes (que no forman parte de un registro ni se subdividen) en la WORKING-STORAGE SECTION.

- El número de nivel 88 se emplea para describir nombres de condición, cuyos valores particulares se asocian a la variable condicional de la cual dependen (línea 002190).

— Los números de nivel superiores a 10 pueden escribirse precedidos de cero, como 01, 02, etc., o precedidos de un espacio en blanco: b1, b2, b3.

A la vista de la fig. 9 y aplicando la regla precedente, los números de nivel podían especificarse en la DATA DIVISION de la forma siguiente:

```
002080          DATA RECORD IS FICHA-OPERARIO.
002090  01  FICHA-OPERARIO.
002100  02  OPERARIO.
002110  03  CODIGO          PICTURE X(5).
002120   3  APELL-NOMBRE    PICTURE X(35).
002130   3  FECHANACI.
002140   4  DIA             PICTURE 99.
002150  04  MES             PICTURE 99.
002160  04  ANNO            PICTURE 99.
002170   3  PROVINCIA-NACIM  PICTURE 99.
002180  03  ESTADO          PICTURE 9.
002190  88  SOLTERO         VALUE 1.
```

Sin embargo, debe evitarse describir un registro de esta forma, ya que en ella no se pueden apreciar las diferentes subdivisiones de cada campo, y, por tanto, no favorece la legibilidad del programa.

3.2.2.- Nombres de Datos

FUNCION: Proporcionar nombres simbólicos a campos o áreas de la memoria del ordenador destinadas a contener los datos, a fin de poder hacer referencia a ellas dentro del programa.

FORMATO:

{ **Nombre–de–datos** }
{ **FILLER** }

- Los nombres–de–datos siguen en su formación las mismas reglas que las citadas para la formación de los nombres creados por el programador.

- Nombre–de–datos es la primera palabra que debe seguir al número de nivel.

- Si ninguna de las declaraciones que contiene la PROCEDURE DIVISION va a hacer referencia a algún campo de los descritos en la DATA, no es necesario darle un nombre–de–datos, sino escribir despúes del número de nivel la palabra FILLER, ya que ésta es el nombre standard con el que se puede denominar a cualquier campo, lo cual ahorra el tener que crear un nombre de datos.

Observando el contenido de la PROCEDURE (figs. 11 y 12), no se hace referencia en ella a los nombres de los campos FACTORIA, NOMBRE, DEPARTAMENTO, NUMERO, SECCION, PROVINCIA–NACIM y SEXO. En este caso se puede sustituir el nombre PROVINCIA–NACIM por el nombre FILLER (línea 002170), el campo SEXO por otro FILLER (línea 003010), y al conjunto de los campos descritos desde la línea 003120 a la 003170, por un FILLER que agrupe a todos ellos (línea 003120).

```
002090     01  FICHA-OPERARIO.
002100       02  OPERARIO.
002110         03  CODIGO                    PICTURE  X(5).
002120         03  APELL-NOMBRE              PICTURE  X(35).
002130         03  FECHANACI.
002140           04  DIA                     PICTURE  99.
002150           04  MES                     PICTURE  99.
002160           04  ANNO                    PICTURE  99.
002170         03  FILLER                    PICTURE  99.
002180         03  ESTADO                    PICTURE  9.
002190           88  SOLTERO   VALUE  1.
002200           88  CASADO    VALUE  2.
002210           88  VIUDO     VALUE  3.
003010         03  FILLER                    PICTURE  A.
003040         03  NUM-HIJOS                 PICTURE  99.
003050         03  FECHAINGRESO.
003060           04  DIA                     PICTURE  99.
003070           04  MES                     PICTURE  99.
003080           04  ANNO                    PICTURE  99.
003090         03  SUELDO                    PICTURE  9(5).
003100         03  TIPODESCUEN               PICTURE  99.
003110         03  NUM-HORASEX               PICTURE  999.
003120     02  FILLER                        PICTURE  X(12).
```

En el caso de que haya que realizar un proceso en el que de la FICHA–OPERARIO se utilicen solamente los campos APELLIDOS Y NOMBRE, SUELDO y NUMERO HORAS EXTRAS, se describiría el registro tal como muestra la descripción siguiente, ya que el compilador tendría suficiente información para localizar, dentro del campo OPERARIO, cada uno de los campos citados anteriormente:

```
002090     01  FICHA-OPERARIO.
002100         02  OPERARIO.
002110             03  FILLER          PICTURE X(5).
002120             03  APELL-NOMBRE    PICTURE X(35).
002130             03  FILLER          PICTURE X(18).
002140             03  SUELDO          PICTURE 9(5).
002150             03  FILLER          PICTURE X(2).
002160             03  NUM-HORASEX     PICTURE 9(3).
002170         02  FILLER              PICTURE X(12).
```

– A lo largo de este texto, y al describir los formatos de las declaraciones que constituyen la PROCEDURE DIVISION, en la mayor parte de ellos aparece la palabra: Identificador, la cual no significa otra cosa que ser un nombre–de–datos, que debe reunir cualquiera de las siguientes características:

- Ser exclusivo en sí mismo, es decir, que no haya otro nombre–de–datos que se denomine de la misma forma dentro del programa, o

- Se pueda hacer exclusivo por la combinación sintáctica correcta de los calificadores (pág. 111), subíndices y/o índices (ver capítulo referente a Tablas).

3.2.2.1.- Calificación de los Nombres de Datos

El programador debe procurar que todos los nombres–de–datos y nombres–de–condición descritos en la DATA DIVISION no se escriban igual, y caso de que existan algunos nombres iguales, hay que buscar un medio para poderlos diferenciar, con el fin de que cuando se haga mención a alguno de ellos en la PROCEDURE DIVISION, el compilador pueda, sin lugar a dudas, saber a qué campo se refiere. A esta operación se la llama: Calificación.

Para la calificación de los nombres–de–datos se emplean las palabras OF o IN (indistintamente) seguidos de otros nombres–de–datos, llamados calificadores, cuyo número de nivel es superior al nombre que califican:

```
007050  CALCULO-TRIENIOS.
007070      IF  MES OF FECHAINGRESO < M,
007080      COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3.
```

Con la descripción: ANNO OF FECHAINGRESO el compilador diferencia el año de la fecha de ingreso del operario, del año de la fecha de nacimiento (FECHANACI, línea 002130).

Cada nombre que haya que calificar debe tener, por lo menos, un calificador que sea distinto de cualquier otro posible calificador de los demás, debiendo estar comprendido dentro de la misma jerarquía de niveles que el nombre que califica.

Observando la descripción del registro FICHA–OPERARIO (págs. 23 y 24), la calificación de los dos campos denominados ANNO se puede hacer de la forma siguiente:

ANNO OF FECHAINGRESO OF OPERARIO IN FICHA–OPERARIO.
ANNO OF FECHANACI OF OPERARIO IN FICHA–OPERARIO.

FECHAINGRESO y FECHANACI son los calificadores que diferencian al campo ANNO, ya que los dos últimos nombres por ser comunes a ambos no son calificadores, aunque en cualquiera de los dos casos la descripción sea correcta. Como norma, se debe utilizar, únicamente, calificación suficiente para convertir un nombre en único.

Los calificadores deben especificarse en orden ascendente de nivel. No sería correcto: ANNO OF OPERARIO OF FECHANACI.

El nivel superior para calificar a un nombre–de–datos es el del nombre–de–archivo que sigue a FD; niveles sucesivos son: 01, 02, ...

Los nombres de archivo deben ser únicos, ya que no pueden calificarse por contener el nivel superior (FD).

Los nombres–de–datos que siguen a los números de nivel 01 ó 77 en la WORKING–STORAGE SECTION deben ser únicos, ya que ninguno de ellos puede calificarse.

3.2.3.- Cláusula PICTURE

FUNCION: Especificar el tipo de caracteres que pueden almacenarse en cada campo de la memoria.

FORMATO:

$$\left\{ \begin{array}{l} \underline{\text{PICTURE}} \\ \underline{\text{PIC}} \end{array} \right\} \quad \text{IS serie de caracteres}$$

- La cláusula PICTURE sólo puede especificarse en campos elementales.

- Serie de caracteres puede ser uno o varios símbolos constituídos por una combinación válida de cualquiera de los siguientes (11):

  A X 9 V P S . Z * CR DB 0 B $ + - ,

- Según la combinación empleada así será la clase de datos (alfabéticos, numéricos, etc.) que pueda admitir el ordenador en la memoria.

- La serie de caracteres que acompañan a la PICTURE indican el tamaño de los datos que pueden almacenarse, es decir, si se especifica: 01 ALFA PICTURE A(12), quiere decir que en dicho campo podrán almacenarse hasta doce letras y/o espacios en blanco, uno por cada símbolo A. Sin embargo, ésto no quiere decir que el compilador reserve en memoria un octeto por cada símbolo que figure a continuación de PICTURE, ya que hay casos como en los símbolos P, V y S por los que no reserva ningún octeto, o como CR o DB por los que reserva dos octetos, uno por cada letra; o en el caso del símbolo 9 por el que puede reservar uno o medio octeto de la memoria (según el tipo de USAGE que se emplee).

- Un número entero colocado entre paréntesis y precedido de A X 9 * Z P 0 - + B $ significa el número consecutivo de veces que se repite dicho símbolo en la PICTURE.

Las descripciones de la línea 004170 y 004190 del ejemplo siguiente son equivalentes:

```
004160     02  CODIGO-S          PICTURE  X(5).
004170     02  FILLER            PICTURE  XXXX  VALUE SPACES.
004180     02  APE-NOMBRE        PICTURE  X(35).
004190     02  FILLER            PICTURE  X(4)  VALUE SPACES.
004200     02  SUELDO-S          PICTURE  ZZ.ZZZ.
```

(11) Aunque a continuación de PICTURE figuren una serie de caracteres COBOL, sin embargo, para el compilador dichos caracteres son símbolos que sirven cada uno para realizar una función determinada, como sucede con los verbos COBOL, constantes figurativas, etc, los cuales también tienen un sentido predeterminado. Por ello a lo largo de la descripción de esta cláusula se les denomina símbolos en vez de caracteres.

– En una PICTURE sólo pueden aparecer una vez los símbolos S, CR, DB, V.

EJEMPLO:

```
090    02  ALFA   PICTURE  S999V999.
100    02  BETA   PICTURE  99.9CR.
110    02  ZETA   PICTURE  ZZZZDB.
```

Los símbolos S, CR y DB indicadores del signo del campo de datos se excluyen entre sí.

No sería correcto especificar:

```
140    02  ALFA   PICTURE  S999V99CR.
```

ya que tanto S como CR y DB indican el signo del campo, por tanto, incluir dos símbolos sería una redundancia.

– El tamaño de la PICTURE no puede exceder de 30 símbolos.

EJEMPLO:

```
170    02  ALFA  PICTURE  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ999999999.99.
```

No sería correcto, ya que se han especificado a continuación de PICTURE una serie de 32 símbolos; sin embargo, sería correcto escribir la inscripción equivalente:

```
200    02  ALFA   PICTURE  Z(22)9(7).99.
```

en la que sólo se especifican 12 símbolos dentro de la PICTURE.

– Una PICTURE debe estar formada, por lo menos, de un símbolo A X 9 Z * o dos símbolos seguidos $ + ó –

EJEMPLO:

```
170    03  ALFA   PICTURE  A.
180    03  BETA   PICTURE  XXX.
190    03  ZETA   PICTURE  ***.**.
200    03  CAMP   PICTURE  --.
210    03  CAM    PICTURE  $$$$.
220    03  CAMPO  PICTURE  +++.99.
230    03  CA     PICTURE  Z.
```

En la descripción de las líneas 200 y 220, el primer símbolo – ó + representa el signo del campo y el siguiente o siguientes indican que en ellos se pueden almacenar caracteres (pág. 121).

CLASIFICACION de los CAMPOS.

Según el tipo de símbolos que se describan en la serie de caracteres que siguen a la cláusula PICTURE, los campos se clasifican en diversos tipos:

- **ALFABETICOS.**
- **ALFANUMERICO.**
- **NUMERICOS.**
- **EDICION NUMERICA.**
- **EDICION ALFANUMERICA.**

CAMPOS ALFABETICOS

Para describir en un programa un campo de tipo alfabético se emplea el símbolo A, el cual indica al compilador que reserve un campo de la memoria cuya longitud (número de octetos) sea igual al número de A,s especificadas, y que los datos que en él se han de almacenar serán únicamente las letras del alfabeto y/o blancos.

CAMPOS ALFANUMERICOS

Para describir un campo de tipo alfanumérico se emplea normalmente el símbolo X, con el que el campo que se reserva en la memoria admite cualquier carácter del juego de caracteres COBOL.

También pueden emplearse los símbolos A y 9 cuya función es la misma que para X, ya que pueden almacenar cualquier letra, número o carácter especial; sin embargo, para que una serie de caracteres PICTURE que contenga los símbolos A ó 9 se considere alfanumérica, éstos deben combinarse con la X, es decir, en cada PICTURE debe existir un símbolo X como mínimo.

Por ello, no es correcto considerar los campos ALFA y BETA como campos alfanuméricos en la descripción siguiente:

```
020   02  ALFA   PIC  AA99.
030   02  BETA   PIC  AAA.
040   02  DELTA  PIC  AXX9.
```

- Por cada símbolo A, X y 9 el compilador reserva un octeto en la memoria.

CAMPOS NUMERICOS.

Según se indicó en la página 61, el ordenador puede operar con números representados en coma fija y en coma flotante, por tanto es necesario poder describir PICTURE,s que sean capaces de reservar campos en la memoria del ordenador, donde poder almacenar números representados en ambos formatos.

Los campos numéricos se clasifican en:

- Campos de COMA FIJA y
- Campos de COMA FLOTANTE.

CAMPOS NUMERICOS de COMA FIJA.

Para hacer la descripción de la PICTURE de un campo numérico de coma fija, se utiliza una combinación, según determinadas reglas, de los símbolos: 9 V P y S.

- El símbolo 9 indica que en la posición correspondiente se almacenará un dígito.

EJEMPLO. La descripción de un campo ALFA capaz de almacenar el número 7584,sería:

```
01 02 ALFA PIC 9999.
```

por cada símbolo 9 reservaría el compilador un octeto en la memoria, donde se colocarán cada una de las cifras del número que ha de almacenar.

- El símbolo V indica al compilador la posición de un punto decimal implícito.

EJEMPLO. La descripción de un campo ALFA capaz de almacenar el número 125.43, sería :

```
01 02 ALFA PIC 999V99.
```

El compilador no reserva un octeto de la memoria para el símbolo V, sino que lo utiliza internamente para alinear las partes decimales de los campos, cuando se quiere realizar alguna operación aritmética o alguna transferencia de datos. Sin embargo, si se desea imprimir el contenido de dicho campo, el resultado sería 12543, es decir, no tendría en cuenta el punto decimal, ya que tampoco se reservó para V un octeto. Por tanto, si se desea imprimir el punto decimal será necesario utilizar caracteres de edición.

- El símbolo S indica que el campo tiene un signo operacional. Es obligatorio cuando se describe el campo con la cláusula USAGE COMPUTATIONAL. En los restantes casos es opcional, pero siempre debe ser el primer carácter de la serie de caracteres de la PICTURE.

  Si se omite el símbolo S, considera positivos a todos los dígitos almacenados, (aunque fueran negativos), por lo que, si se presume que en los resultados puede haber números negativos, es necesario colocar S, a fin de evitar que aquéllos resultados sean incorrectos.

EJEMPLO. Se quiere restar al contenido del campo ALFA, 2.53, el del campo BETA, 753.40, y almacenar la diferencia en el campo DELTA. Suponiendo que la descripción de los campos es:

```
070   02  ALFA  PIC  9V99.
080   02  BETA  PIC  999V9.
090   02  DELTA PIC  999V99.
```

La primera operación a realizar por el ordenador sería la alineación de la coma decimal y a continuación restaría las dos cantidades:

```
      V
    2 5 3
  7 5 3 4 0
- 7 5 0 8 7
```

para después poner la diferencia 75087 dentro del campo DELTA, con lo que se perdería el signo; mientras si se hubiera especificado:

```
120   02  DELTA PIC  S999V99.
```

el resultado sería −75087.

– El símbolo P se usa para indicar la situación del punto decimal a la izquierda o derecha del campo de datos. Cada P representa una posición de dígito que no ocupa memoria y sólo contendrá ceros, no pudiendo haber más de 17 P,s dentro de una PICTURE (11).

EJEMPLO. En los campos ALFA Y BETA descritos de la forma siguiente:

```
010   02  ALFA  PIC  PP999.
020   02  BETA  PIC  999PPP.
```

se quiere almacenar el número 346.

El resultado será que en el campo de la memoria definido por ALFA se almacenará 00346 y en BETA: 346000.

– No pueden intercalarse en una misma PICTURE P,s y 9,s, es decir, no es correcto:

```
040   02  ALFA  PIC  9PPP9.
```

– El número máximo de dígitos que puede almacenar un campo numérico de coma fija es de 18.

---

(11) Cuando la memoria del ordenador es de pequeña capacidad, es importante utilizar cuantos medios facilita el lenguaje para ahorrar posiciones (octetos) de memoria; sin embargo la tendencia actual de incorporar al ordenador memorias de gran capacidad (más de 4 millones de octetos), hace cada vez más innecesario gastar tiempo en la búsqueda de dicho ahorro.

CAMPOS NUMERICOS de COMA FLOTANTE

Estos campos sirven para definir valores cuya gama no puede ser cubierta por los campos de coma fija (para el ordenador que se utiliza en la realización de los ejercicios de este texto, corresponde a números mayores que 5.4×10^{-79} y menores de $.72 \times 10^{76}$).

El formato de un campo numérico de coma flotante es el siguiente:

$\left\{\begin{matrix}+\\-\end{matrix}\right\}$ **mantisa** E $\left\{\begin{matrix}+\\-\end{matrix}\right\}$ **exponente**

- La serie de caracteres de la PICTURE de la mantisa, está constituída por los símbolos 9 V y..
  - El símbolo 9 permite reservar un octeto en la memoria y, por tanto permite almacenar un dígito. Pueden especificarse de 1 a 16 nueves.
  - El símbolo V, al igual que en los campos de coma fija, indica la posición del punto decimal implícito. No ocupa un octeto en la memoria.
  - El símbolo punto (.) indica, también, la posición del punto decimal. Ocupa un octeto en la memoria.

- La serie de caracteres de la PICTURE del exponente está constituída por dos nueves consecutivos.

EJEMPLO. Se quiere describir la serie de caracteres de una PICTURE capaz de almacenar el número -254.78×10^{12}.

Soluciones válidas serían cualquiera de las siguientes:

```
010  02  ALFA      PIC -9(3).99E+99.
020  02  BETA      PIC -9(4).9E+99.
030  02  ZETA      PIC -.9(5)E+99.
```

ya que con cualquier descripción de ellas, podría almacenarse dicho número en la memoria, haciendolo, en cada una, de la forma siguiente:

ALFA: −254.78 E +12 BETA:−2547.8 E+11 ZETA:−.25478 E+15

Es decir, la cantidad -254.78×10^{12} se introduce en cualquiera de los tres campos sin afectarle la posición de la coma, ya que ésta, automáticamente, se traslada a la posición definida en el campo. Operación que no ocurre con los campos de coma fija, dando lugar a truncamientos (ver pág. 170 declaración MOVE).

CAMPOS DE EDICION NUMERICA.

Los símbolos hasta ahora descritos proporcionan información al compilador acerca del tamaño de memoria a reservar para cada campo de datos, tipo de caracteres que se pueden almacenar en cada uno de ellos, situación del punto decimal, etc.,pero esta información resulta insuficiente si de lo que se trata es de obtener la escritura del contenido de los campos NUMERICOS en una forma que sea legible, ya que la impresión de dicho contenido sólo tiene en cuenta las cifras del número, dejando de imprimir el punto decimal, la coma, los signos + ó – etc., por lo que, de lo que se imprima, sólo se puede conocer las cifras que componen dichos números, pero se desconoce si éstos son enteros,o tienen parte entera y decimal, positivos o negativos, etc. Por tanto, es necesario un paso previo a la impresión, EDICION, es decir,la preparación de los datos numéricos para la impresión, lo cual se consigue empleando determinados símbolos en la PICTURE, llamados de EDICION o INFORME.

Los símbolos que definen un campo de edición son:

9 V P Z * . , CR DB 0 B $ + –

(9) (V) (P) tienen las mismas funciones que las descritas para los campos numéricos.

(Z) Cada Z incluída en una PICTURE permite almacenar un dígito, pero si alguno de estos dígitos son ceros no significativos no se imprimen, sustituyéndolos por espacios en blanco:

(12)

VALOR CAMPO ORIGEN	PICTURE CAMPO RECEPTOR	SALIDA IMPRESA
008012	ZZZZZZ	8012

La no impresión de ceros no significativos termina al llegar al punto decimal o a la V, excepto cuando el valor del campo es cero, en cuyo caso imprime sólo blancos.

0000.01	ZZZZ.ZZ	.01
0000.00	ZZZZ.ZZ	

(*) Cada * incluído en la PICTURE permite almacenar un dígito, pero si alguno de estos dígitos son ceros no significativos, se suprimen y en su lugar se imprimen asteriscos.

0012.00	****.**	**12.00

(12) Los ejemplos que se citan a lo largo de la descripción de los símbolos de edición están,la mayoría,tomados de la pág. 124 omitiéndose el encabezamiento VALOR CAMPO ORIGEN... etc, para no ser repetitivos.

Tanto los asteriscos como las Z,s no pueden ir precedidos,pero sí seguidos,de nueves.

0012.00	**** . 99	* * 1 2 . 0 0

Cuando el valor del campo es cero,se imprimen en su lugar asteriscos.

0000.00	* , *** . **	* * * * * . * *

Este símbolo es muy empleado en informes y cuentas bancarias, para evitar falsificaciones.

(.) El punto hace que se inserte un punto decimal en la posición donde termina la parte entera y comienza la parte decimal del campo.

Si el número no tiene parte decimal,no se debe incluir ningún punto al final de la serie de caracteres de la PICTURE. Es decir,no es correcto especificar:

02 ALFA PICTURE ZZZ ..

(,) (0) (B) Permiten que se inserte una coma, un cero o un blanco en la posición que se especifica dentro de la PICTURE, pero no permiten almacenar un dígito en su lugar.

12345	ZZ ,BBZ , Z0Z . 99	1 2 , 3 , 4 0 5 . 0 0
12345	000ZZ . 99	0 0 0 4 5 . 0 0

Cuando la serie de caracteres de una PICTURE está formada de Z 0 ó B y el campo contiene algún cero no significativo, si alguno de estos tres tipos de símbolos no son alcanzados, se suprimen y en su lugar se imprimen blancos.

1402.5	ZZ , 909 , 999 . 99	1 , 4 0 2 . 5 0

(CR) (DB) Son símbolos empleados en medios bancarios, indicativos de crédito y débito. Sólo pueden colocarse a la derecha de la serie de caracteres de la PICTURE.

Si el valor del campo es negativo se imprimirán dichos símbolos, pero si es positivo dejará de imprimirse y en su lugar aparecerán espacios.

+ 0012.00	ZZZ9 . 99CR	1 2 . 0 0
– 0012.00	ZZZ9 . 99CR	1 2 . 0 0 C R
– 1234.56	* , ***. 99DB	1 , 2 3 4 . 5 6 D B
+ 1234.56	* , ***. **DB	1 , 2 3 4 . 5 6

(\$) (+) (−) Tienen doble función:

a.− Cuando se emplean como símbolos aislados dentro de una PICTURE y

b.− Cuando forman parte de una fila flotante, es decir, de una sucesión continua de dichos símbolos interrupidos solamente por algún 0 B V ó ,

En el caso a) el símbolo \$ debe aparecer en el extremo izquierdo de la PICTURE, no se reserva en memoria un octeto y, por tanto, no se puede almacenar un dígito, imprimiéndose delante del contenido del campo:

00123.45	\$ZZ , ZZZ . 99 CR	\$ 1 2 3 . 4 5
000.23	\$ZZZ . 99	\$. 2 3
000.23	\$**9 . 99	\$ * * 0 . 2 3
0000.00	\$ **, ***. **	* * * * * * * . * *
0000.00	\$ZZ , ZZZ . ZZ	
0000.00	\$99 , 999 . 99	\$ 0 0 , 0 0 0 . 0 0

El símbolo + se puede colocar en el extremo derecho o izquierdo de la PICTURE. Si el valor del campo es positivo o sin signo, se imprimirá un signo más y si es negativo se imprimirá un signo menos:

0012.00	+9999 . 99	+ 0 0 1 2 . 0 0
0012.00	ZZZZ. 99+	1 2 . 0 0 +
0012.00	+**** . 99	+ * * 1 2 . 0 0
+0012.00	+9999 . 99	+ 0 0 1 2 . 0 0
+0012.00	+ZZZZ. 99	+ . 1 2 . 0 0
+0012.00	+**** . 99	+ * * 1 2 . 0 0
+0012.00	9999 . 99+	0 0 1 2 . 0 0 +
−0012.00	+9999 . 99	− 0 0 1 2 . 0 0
−0012.00	+ZZZZ. 99	− . 1 2 . 0 0
−0012.00	+**** . 99	− * * 1 2 . 0 0

El símbolo − se puede colocar en la misma posición que el símbolo + . Si el valor del campo es positivo o sin signo, se imprimirá un blanco en el lugar del signo, y si el valor del campo es negativo se imprimirá un signo menos:

−0012.00	−9999 . 99	− 0 0 1 2 . 0 0
−0012.00	−ZZZ9. 99	− . 1 2 . 0 0
−0012.00	ZZZZ. 99−	. 1 2 . 0 0 −
1234.56	\$*,** 9 . 99−	\$ 1 , 2 3 4 . 5 6

En el caso b) para que exista en una PICTURE una fila flotante, es necesario que en el extremo izquierdo de la PICTURE aparezcan, por lo menos, dos símbolos seguidos de $ + ó −

EJEMPLO:

```
020  03  ALFA   PICTURE  $$.
030  03  BETA   PICTURE  ------.--.
040  03  NUM    PICTURE  ++++.
```

La serie de caracteres de una PICTURE de fila flotante no tiene por qué estar constituida por los mismos símbolos en su totalidad:

```
070  03  ALFA   PICTURE  $$$999.99.
080  03  BETA   PICTURE  --99.9.
```

ya que por el hecho de ser los primeros símbolos $ ó − y existir más de uno ya es fila flotante.

La aparición en una PICTURE de n caracteres consecutivos de $ + ó − hace que se reserven en memoria un número de octetos capaz de almacenar n−1 dígitos.

El símbolo extremo izquierdo de la PICTURE es el que va flotando (avanzando) a través de todos los ceros no significativos hasta encontrar el primer dígito significativo, imprimiéndose inmediatamente delante del primer dígito del campo.

+1234.56	$$,$$9.99−	$1,234.56
+1234.56	$$,$$9.99+	$1,234.56+
+1234.56	$$,$$9.99DB	$1,234.56
0012.00	++++++.++	+12.00
−0012.00	------.--	−12.00
0012.00	$$$$$.$$	$12.00
−0012.00	------.99	−12.00

Existen dos excepciones a la impresión de los símbolos flotantes + y – :

1.– Si en un campo con PICTURE de fila flotante conteniendo el símbolo –, se almacena un número positivo o sin signo, el símbolo – flota a través de todos los ceros no significativos, pero no se imprime.

+0012.00	––––– . 99	12.00
0012.00	––––– . 99	12.00

2.– Si en un campo con PICTURE de fila flotante del símbolo + , se almacena un número negativo, el símbolo + flota pero no se imprime, haciendolo en su lugar el símbolo –.

–0012.00	+ + + + + . + +	–12.00

Otras reglas para la serie de caracteres de edición de la PICTURE son:

– Las Z y * no pueden formar parte de una misma PICTURE (se excluyen entre sí):

```
110   02  B   PICTURE  ZZZ****.
```

– Las Z y * no pueden ir a continuación de una fila flotante:

```
140   02  A   PICTURE  $$$$ZZZZ.ZZ.
```

– Los símbolos + – CR y DB se excluyen entre sí dentro de una PICTURE.

– Si a la derecha del punto decimal hay alguno de los símbolos Z * + – ó $,el resto de los símbolos de la PICTURE debe ser del mismo tipo:

```
160   02  ALFA    PICTURE  ZZZ.99.
170   02  BETA    PICTURE  ZZZZ.ZZ.
180   02  ALF     PICTURE  ***.***.
```

No sería correcto escribir:

```
160   02  ALFA    PICTURE  ZZZ.Z9.
170   02  BETA    PICTURE  ZZZZ.9Z.
180   02  ALF     PICTURE  ***.9*9.
```

CAMPOS DE EDICION ALFANUMERICA

Son aquellos campos en los que la serie de caracteres de su PICTURE está formada por una combinación válida de los símbolos: A, X, 9, B y 0 (cero), cuya función es la descrita para los campos de edición numérica.

— Las combinaciones válidas de la serie de caracteres deben contener como mínimo:

- Un 0 y una X
- Un 0 y una A
- Una X y una B

VALOR CAMPO ORIGEN	PICTURE CAMPO RECEPTOR	SALIDA IMPRESA
BUENAVENTURA´	X(5) B B X(7)	BUENA VENTURA
´125´	X(3) 0 B 000	125 000
´C´	0 0 A	00C
´FIN´	0BAAXB0	0 FIN 0

	Cantidad movida	PICTURE Campo Receptor	SALIDA Impresa
01	0012.00	9999.99	0012.00
02	0012.00	ZZZZ.ZZ	12.00
03	0012.00	****.**	**12.00
04	0012.00	+++++.++	+12.00
05	0012.00	-----.--	12.00
06	0012.00	$$$$$.$$	$12.00
07	0012.00	+9999.99	+0012.00
08	0012.00	+ZZZZ.99	+ 12.00
09	0012.00	+****.99	+**12.00
10	+0012.00	+9999.99	+0012.00
11	+0012.00	+ZZZZ.99	+ 12.00
12	+0012.00	+****.99	+**12.00
13	+0012.00	$$$$.99+	$12.00+
14	+0012.00	-****.99	**12.00
15	+0012.00	-----.99	12.00
16	+0012.00	-9999.99	0012.00
17	+0012.00	-ZZZ9.99	12.00
18	+0012.00	ZZZ9.99CR	12.00
19	+0012.00	ZZZZ.99-	12.00
20	-0012.00	+9999.99	-0012.00
21	-0012.00	+ZZZZ.99	- 12.00
22	-0012.00	+****.99	-**12.00
23	-0012.00	$$$$.99+	$12.00-
24	-0012.00	-****.99	-**12.00
25	-0012.00	-----.99	-12.00
26	-0012.00	-9999.99	-0012.00
27	-0012.00	-ZZZ9.99	- 12.00
28	-0012.00	ZZZ9.99CR	12.00CR
29	-0012.00	ZZZZ.99-	12.00-
30	-1234.56	$$,$$9.99-	$1,234.56-
31	-1234.56	$$,$$9.99+	$1,234.56-
32	-1234.56	$$,$$9.99DB	$1,234.56DB
33	+1234.56	$$,$$9.99-	$1,234.56
34	+1234.56	$$,$$9.99+	$1,234.56+
35	+1234.56	$$,$$9.99DB	$1,234.56
36	00123.45	$ZZ,ZZZ.99CR	$ 123.45
37	00123.45	$**,***.99+	$***123.45+
38	00123.45	$99,999.99-	$00,123.45
39	000.23	$ZZZ.99	$ .23
40	000.23	$**9.99	$**0.23
41	000.23	$**9.99+	$**0.23+
42	12345	ZZ,BBZ,Z0Z.99	12, 3,405.00
43	12345	000ZZ.99	00045.00
44	0000.00	$**,***.**	*******.**
45	0000.00	$ZZ,ZZZ.ZZ	
46	0000.00	$99,999.99	$00,000.00

3.2.4.- Cláusula USAGE

FUNCION: Indicar la forma que se representarán los datos dentro de la memoria.

FORMATO:

```
[ USAGE IS ] { DISPLAY
             { COMPUTATIONAL }
             { COMP          }
             { COMPUTATIONAL-1 }
             { COMP-1          }
             { COMPUTATIONAL-2 }
             { COMP-2          }
             { COMPUTATIONAL-3 }
             { COMP-3          }
             { INDEX           }
```

La opción DISPLAY puede emplearse en la descripción de campos de cualquier tipo: alfabético, alfanumérico, etc.; el resto de las opciones se deben emplear para describir campos numéricos.

Opción DISPLAY.

Un campo de datos descrito con la opción DISPLAY, hace que se reserve en la memoria un octeto para almacenar cada carácter (ya sea numérico o no).

EJEMPLO. Si a los campos ALFA y BETA descritos de la forma siguiente:

```
150  02  ALFA  PIC A(6)  USAGE DISPLAY.
160  02  BETA  PIC 9,999.99.
```

se transfieren los valores 'RICOSA' y 1.248,73 respectivamente, su representación en memoria sería:

R	I	C	O	S	A
11011001	11001001	11000011	11010110	11100010	11000001

1	,	2	4	8	.	7	3
11110001	01101011	11110010	11110100	11111000	01001011	11110111	11110011

Si en un campo compuesto o elemental no se especifica la cláusula USAGE, hace la misma función que si se hubiera especificado USAGE DISPLAY; en el ejemplo anterior, BETA es un campo que lleva implícito el USAGE DISPLAY.

Es obligado el empleo de la opción DISPLAY en la descripción de los campos que serán receptores de datos contenidos en fichas, o que hayan de imprimirse.

A los campos numéricos descritos con la opción DISPLAY se les denomina: decimales EXTERNOS.

Opción COMPUTATIONAL (COMP).

Un campo descrito con PICTURE numérica y con USAGE COMPUTATIONAL, hace que se almacenen en formato binario puro los números positivos, y en forma de complemento a dos los números negativos. A los campos así descritos se les denomina BINARIOS.

En binario puro el número de octetos que reserva el compilador en la memoria para almacenar cualquier número expresado en el sistema decimal es:

2 octetos para los números que contienen de 1 a 4 dígitos.
4 octetos para los números que contienen de 5 a 9 dígitos.
8 octetos para los números que contienen de 10 a 18 dígitos.

La pág. 128 muestra la forma como se almacenará en el campo ZETA, en formato binario el número +715, es decir, en el bit extremo izquierdo del campo se almacena el signo (1 para indicar números negativos y 0 para los positivos) y en el resto de los bits las cifras del número.

Con esta opción hay ahorro de memoria, al poder colocar un número de tres cifras en dos octetos, mientras que con USAGE DISPLAY se necesitarían tres octetos. Si el número hubiera sido de dieciocho cifras, podría almacenarse en ocho octetos.

El empleo más frecuente de esta opción, es en campos que van a servir de subíndices para la búsqueda en TABLAS.

Opción COMPUTATIONAL–3 (COMP–3).

Un campo descrito con PICTURE numérica y con la opción COMPUTATIONAL–3, almacena dos dígitos en cada octeto (formato empaquetado), utilizando los cuatro bits del extremo derecho del campo para representar el signo del número: (1100 para los positivos, 1101 para los negativos y 1111 para las números sin signo). A estos campos se les denomina decimales INTERNOS.

En la pág. 128 se muestra el almacenamiento interno del campo BETA, al que por medio de la declaración MOVE, se le ha transferido la cantidad + 715.

Opciones COMPUTATIONAL–1 y COMPUTATIONAL–2 (COMP–1, COMP–2).

Las opciones COMPUTATIONAL–1 y COMPUTATIONAL–2 se emplean cuando en los campos de datos se quieren almacenar números muy grandes o muy pequeños, es decir, números mayores de $5 . 4 \times 10^{-79}$ y menores de $. 72 \times 10^{76}$, o bien para realizar la potenciación de números con exponente fraccionario.

A los campos descritos con USAGE COMPUTATIONAL–1 ó COMPUTATIONAL–2 se les denominan campos de coma FLOTANTE INTERNA.

Con la opción COMPUTATIONAL–1, cada número se almacena en memoria en cuatro octetos, empleando el primer bit para el signo, los 7 siguientes bits para el exponente (característica) y hasta el bit 31 para la fracción. Su empleo es en operaciones con números que requieren poca precisión.

Si al campo LANDA se le transfiere la cantidad $+.715 \times 10^3$, ésta se almacenaría en la memoria de la forma que se indica en la pág. 128.

Con COMPUTATIONAL–2 los ocho primeros bits tienen el mismo uso que en COMPUTATIONAL–1, pero para la fracción se emplean hasta el bit 63. Se utiliza en operaciones que requieren gran precisión.

Si al campo COM se le transfiere la cantidad $+.715 \times 10^3$, ésta se almacenaría en la memoria de la forma indicada en la pág. 128.

En un campo numérico que se describa con cualquiera de los dos tipos de USAGE citados, no se debe especificar la cláusula PICTURE.

Teniendo en cuenta que para las operaciones comerciales no se requiere este tipo de cantidades, ni gran precisión en los valores decimales, el empleo de dichas opciones es casi nulo.

Opción INDEX.

Se describe en el capítulo referente a Tablas.

CONSIDERACIONES GENERALES.

Todos los campos de datos que se han descrito enel programa NOMINA, se han hecho con USAGE DISPLAY, ya que,como se dijo anteriormente, cuando en un campo no se especifica la cláusula USAGE, el compilador toma para el campo la opción DISPLAY. Por tanto,al ser válida esta opción para todos los tipos de campos, el compilador nunca indicará la existencia de error. No ocurre lo mismo con los otros tipos de USAGE , ya que sólo pueden especificarse en determinados tipos de campos.

Por tanto, para un programador inexperto se aconseja que utilice la opción DISPLAY, ya que no le dará ningún tipo de problemas; pero a medida que tome soltura en la utilización del lenguaje, debe estudiar cuál es el tipo de USAGE que mejor conviene a cada tipo de campo, con el fin de buscar una mayor eficiencia en la ejecución del programa.

Es difícil asignar el tipo de USAGE que mejor conviene a cada campo de datos. A título orientativo, y como complemento de cuanto se ha especificado, es aconsejable que:

- Si de lo que se trata es de ahorrar memoria del ordenador, se especificará en los campos numéricos USAGE COMP; igualmente, cuando se describen campos que han de utilizarse como subíndices en el manejo de Tablas.
- En aquellos campos numéricos con los que se haya de realizar operaciones aritméticas, lo más apropiado es describirlos con USAGE COMP–3.
- En los campos de cualquier tipo, cuyo contenido se vaya a imprimir, lo más apropiado es describirlos con USAGE DISPLAY.

La utilización de uno u otro USAGE es consecuencia de la forma como trabaja el ordenador, ya que éste, para llevar a cabo cada una de las distintas acciones que realiza (operaciones aritméticas, búsqueda en Tablas, impresión, etc.) utiliza el USAGE apropiado, es decir, si un campo numérico se ha descrito en el programa con USAGE DISPLAY y hay que realizar con él una operación aritmética, el compilador realiza la conversión del contenido de dicho campo al USAGE COMP–3 y posteriormente ejecuta la operación; o en el caso de que se quiera transferir el contenido de un campo numérico descrito con USAGE COMP a otro descrito con USAGE DISPLAY, el compilador realiza la operación de cambio de USAGE del campo emisor al USAGE del campo receptor, y posteriormente realiza la transferencia.

El problema surge cuando, por ejemplo, un campo numérico se va a utilizar en el proceso: primero para realizar una operación aritmética y posteriormente para imprimir el contenido de dicho campo. En este caso, para la primera operación sería conveniente especificar USAGE COMP–3 y para la segunda operación la mejor sería USAGE DISPLAY. Ante esta disyuntiva, ¿con cuál de los dos tipos de USAGE se describiría dicho campo para que el programa fuera lo más eficiente? La solución está en el buen sentido y la mucha experiencia del programador, ya que en este caso lo que se ahorra en tiempo por describirlo con un USAGE para efectuar la primera operación, se pierde al efectuar la segunda por no haberlo descrito con el otro USAGE.

FORMAS DE ALMACENAMIENTO DE LOS CAMPOS EN LA MEMORIA.

```
020   77  ALFA          PIC 999   USAGE DISPLAY.
030   77  BETA          PIC S999  USAGE COMP-3.
040   77  ZETA          PIC S999  USAGE COMP.
050   77  LANDA                   USAGE COMP-1.
060   77  COM                     USAGE COMPUTATIONAL-2.

190   PROCEDURE DIVISION.
200   ..................
210       MOVE  715    TO ALFA BETA ZETA
220       MOVE  .715E03  TO LANDA  COM
230   ..................
```

```
                    7        1        5
ALFA                11110111 11110001 11110101

                    +              715
ZETA                00000010 11001011

                    7   1    5   +
BETA                01110001 01011100

                    +   3                715
LANDA               0 0000011 00000000 00000010 11001011

                    +   3                                715
COM                 0 0000011 00000000 00000000 00000000 00000000 00000000 00000010 11001011
```

3.2.5.- Cláusula JUSTIFIED

FUNCION: Posicionar los caracteres no numéricos, situándolos de derecha a izquierda en el campo receptor.

FORMATO:

$$\left\{ \begin{array}{l} \underline{\text{JUSTIFIED}} \\ \underline{\text{JUST}} \end{array} \right\} \text{ RIGHT}$$

- Esta cláusula puede utilizarse en la descripción de campos alfabéticos y/o alfanuméricos.

- En la transferencia de datos a aquellos campos en que se ha especificado dicha cláusula, el carácter extremo derecho del campo emisor se sitúa en la posición extrema derecha del campo receptor y, a partir de ahí, se van situando de derecha a izquierda el resto de los caracteres, es decir, al contrario como lo haría el ordenador de no haber especificado dicha cláusula.

- Si la longitud del campo emisor es mayor que la del receptor, la transferencia de caracteres termina una vez se haya llenado el campo receptor, produciéndose un truncamiento o pérdida del resto de los datos. Por el contrario, si la longitud del campo emisor es menor que la del receptor las posiciones extremo izquierda se llenarán con blancos.

- Para los campos numéricos no se debe especificar esta cláusula,ya que el Sistema hace la transferencia,automáticamente,apoyándose a la derecha.

EJEMPLO:

```
120    DATA DIVISION.
       .....................
149        02 ALFA    PICTURE X(9).
150        02 BETA    PICTURE X(9)   JUSTIFIED RIGHT.
160        02 GAMMA   PICTURE A(4).
170        02 ZETA    PICTURE A(4)   JUSTIFIED RIGHT.
180    .....................
190    PROCEDURE DIVISION.
       .....................
210        MOVE 'FINAL' TO ALFA BETA GAMMA ZETA.
```

Por la declaración de movimiento contenida en la PROCEDURE DIVISION, transferiría a cada campo el literal 'FINAL', siendo el contenido de los campos ALFA, BETA, GAMMA y ZETA después de dicha transferencia el siguiente:

ALFA	F I N A L	BETA	F I N A L
GAMMA	F I N A	ZETA	I N A L

3.2.6.- Cláusula BLANK

FUNCION: Llenar un campo con espacios en blanco, siempre que su valor sea cero.

FORMATO:

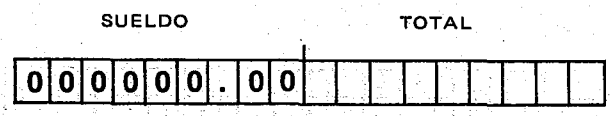
BLANK WHEN ZERO

— Se especifica esta cláusula únicamente en campos elementales de informe.

EJEMPLO. Si los campos de informe, SUELDO y TOTAL están descritos como sigue:

```
010   DATA DIVISION.
020   ..............
030     01  REGIS-OPE.
040         02 SUELDO    PICTURE  999999.99.
050         02 TOTAL     PICTURE  999999.99 BLANK WHEN ZERO.
      ..........................................
070   PROCEDURE DIVISION.
      ..........................................
090       MOVE 0000000.00  TO  TOTAL  SUELDO
100       WRITE  REGIS-OPE  AFTER 2 ..........
110
```

Una vez ejecutada la declaración de movimiento de la línea 090, el contenido de dichos campos sería 000000.00, pero al ejecutar la declaración WRITE se escribiría sobre la hoja del impreso:



Es decir, en el espacio reservado en el impreso para el campo SUELDO escribiría: 000000.00 y en el espacio reservado para TOTAL escribiría espacios en blanco.

El empleo de esta cláusula no es frecuente y, en algunos casos, puede sustituirse por PICTURE ZZZZZZ.ZZ, pudiendo optar el programador por cualquiera de las dos formas. Sin embargo, el resultado es el mismo, únicamente, cuando el valor del campo sea igual a cero.

3.2.7.- Cláusula OCCURS

FUNCION: Permitir simplificar las inscripciones de campos de datos que se repiten con la misma descripción.

FORMATO 1:

OCCURS entero **TIMES**

Entero indica el número de veces que se repite un campo con la misma descripción; su valor tiene que ser mayor que cero.

EJEMPLO. Se dispone de una cinta magnética grabada con un sólo registro de 3254 caracteres cuyo contenido es el siguiente:

- Un campo de seis caracteres alfabéticos en los que figura el nombre de la empresa.
- Un campo de ocho caracteres alfanuméricos en el que figura el modelo de automóvil fabricado.
- 1080 campos de tres caracteres numéricos cada uno,cuyo contenido es el número diario de automóviles fabricados durante los años 1971, 72 y 73 (los meses se consideran de 30 días).

Se quiere,partiendo del formato de dicho registro, que el programador haga la descripción del mismo:

```
010  01 REGISTRO-AUTOS.
020      02 NOMBRE          PICTURE A(6).
030      02 MODELO          PICTURE X(8).
040      02 DIARIA1         PICTURE 999.
050      02 DIARIA2         PICTURE 999.
060      02 DIARIA3         PICTURE 999.
070         //    //            //    //
080         //    //            //    //
090         //    //            //    //
100         //    //            //    //
110      02 DIARIA1078       PICTURE 999.
120      02 DIARIA1079       PICTURE 999.
130      02 DIARIA1080       PICTURE 999.
```

Como se ve, esta descripción sería muy laboriosa y necesitaría mucho tiempo el llevarla a cabo, ya que serían necesarias varias hojas de codificación.

Pero teniendo en cuenta que al ser los 1080 campos de la misma longitud,y del mismo tipo los caracteres a almacenar, se puede simplificar la descripción empleando la cláusula OCCURS, sustituyendo la anterior descripción por la equivalente:

```
010    01 REGISTRO-AUTOS.
020       02 NOMBRE                    PICTURE A(6).
030       02 MODELO                    PICTURE X(8).
040       02 DIARIA OCCURS 1080 PICTURE 999.
```

– A la descripción del campo en que entra a formar parte la cláusula OCCURS (línea 040) se le da el nombre de TABLA.

– Al nombre del campo de datos, DIARIA, el cual es común para los 1080 campos se le denomina: Nombre Genérico o Sujeto, y a cada uno de los campos, elementos de la TABLA.

– La cláusula OCCURS no puede especificarse en una descripción que lleve nivel 01 ó 77.

FORMATO 2:

OCCURS entero–1 TO entero–2 TIMES DEPENDING ON nombre–de–datos–1

Este formato se emplea cuando el sujeto o nombre genérico tiene un número variable de apariciones.

Si en el ejemplo anterior, en vez de haber grabado la cinta con un único registro conteniendo la producción de 36 meses, cada uno de treinta días, se hubieran grabado 36 registros de longitud variable como consecuencia del número diferente de días de cada mes; en este caso la descripción sería:

```
020    FD ARCHIVO-AUTOS  RECORDING V  LABEL RECORD OMITTED.
030    01 REGISTRO-AUTOS.
040       02  NOMBRE              PIC A(6).
050       02  MODELO              PIC X(8).
060       02  NUMERO-DIAS         PIC 99.
070       02  DIARIA    OCCURS 28  TO  31  TIMES
080           DEPENDING ON  NUMERO-DIAS PIC 9(3).
```

A la vista de dicha descripción, entero–1 representa el número mínimo de apariciones y entero–2 el número máximo. El valor mínimo que puede tomar entero–1 es cero y el valor máximo de entero–2 depende del tipo de compilador utilizado, pudiendo llegar a ser superior a 300.000.

Existen compiladores para los que entero–1 es opcional. Siendo correcto escribir:

```
050       02  MODELO              PIC X(8).
060       02  NUMERO-DIAS         PIC 99.
070       02  DIARIA    OCCURS  31  TIMES  DEPENDING ON  NUMERO-DIAS PIC 9(3).
```

DEPENDING debe aparecer en la última inscripción de los campos de datos que constituyen el registro y, como consecuencia, nombre–de–datos–1 debe especificarse antes de la descripción del nombre genérico de la cláusula OCCURS, cuando forma parte del registro, como se muestra en el ejemplo precedente, ya que no sería correcto escribir:

```
100  FD ARCHIVO-AUTOS  RECORDING V  LABEL RECORD OMITTED.
110  01 REGISTRO-AUTOS.
120     02 NOMBRE                  PIC A(6).
130     02 MODELO                  PIC X(8).
140     02 DIARIA      OCCURS 28 TO 31 TIMES
150                   DEPENDING ON NUMERO-DIAS PIC 9(3).
160     02 NUMERO-DIAS             PIC 99.
```

por no ser DIARIA el último campo del registro.

La serie de caracteres de la cláusula PICTURE perteneciente a nombre–de–datos–1 debe ser numérica.

FORMATO 3:

OCCURS entero–1 **TO** entero–2 **TIMES** [**DEPENDING ON** nombre–de–datos–1]

[{ **ASCENDING** / **DESCENDING** } **KEY IS** nombre–de–datos–2 [nombre–de–datos–3] . . .] . . .

[**INDEXED BY** nombre–de–índice–1 [nombre–de–índice–2] . . .]

En este formato incluye a los dos anteriores, añadiendo la cláusula ASCENDING o DESCENDING, para indicar si los elementos de la tabla están ordenados en forma ascendente o descendente por uno o varios campos (nombre–de–datos–2, nombre–de–datos–3...), que formen parte de dichos elementos.

La utilización de este formato es obligatoria, por ejemplo, cuando para localizar a un determinado elemento de la tabla, se emplee el método de búsqueda binaria o dicotómica por medio de la declaración SEARCH ALL (ver búsqueda en TABLAS), la cual utiliza como Indice para realizar dicha búsqueda, al nombre–de–índice que se describe a continuación de INDEXED BY.

Los valores que pueden asignarse a entero–1 y entero–2 son los mismos que los especificados en el Formato–2.

3.2.8.- Cláusula REDEFINES

FUNCION: Proporcionar una nueva descripción a un campo de la memoria, es decir, dar a dicho campo más de un nombre y más de un formato.

FORMATO:

nombre–de–datos–1 REDEFINES nombre–de–datos–2

Cada campo de la memoria puede almacenar un determinado tipo de caracteres, de acuerdo con la serie de caracteres de la PICTURE que para dicho campo se haya especificado, es decir, si un campo se describe con PICTURE de tipo alfabético, almacenará caracteres alfabéticos, pero si se intentara almacenar en él caracteres numéricos o alfanuméricos, el ordenador interrumpiría la ejecución del programa, debido a que el campo citado no los aceptaría por haber sido preparado para admitir únicamente caracteres alfabéticos.

EJEMPLO. Si en el programa NOMINA, el programador hubiera hecho la descripción del registro FICHA–OPERARIO de la siguiente forma:

```
002090     01  FICHA-OPERARIO.
002100       02  OPERARIO.
002110         03  CODIGO                  PICTURE  X(5).
002120         03  APELL-NOMBRE            PICTURE  A(35).
002130         03  FECHANACI.
002140           04  DIA                   PICTURE  99.
002150           04  MES                   PICTURE  99.
```

según esta descripción el campo APELL–NOMBRE sólo podría almacenar treinta y cinco letras y/o espacios en blanco, por ser los caracteres de su PICTURE de tipo alfabético, sin embargo, en este caso no se ha previsto la posibilidad de almacenar en dicho campo apellidos compuestos, por ejemplo, GARCIA–OBREGON SANCHEZ ANTONIO, ya que el campo APELL–NOMBRE no los aceptaría, debido a que el guión existente en dicho apellido es un carácter alfanumérico. La solución más sencilla a este problema sería sustituir la PICTURE A(35) de la anterior descripción por PICTURE X(35).

No obstante, si no se desea sustituir la anterior descripción, puede hacerse que dicho campo sea capaz de aceptar indistintamente caracteres alfabéticos y alfanuméricos, añadiendo la inscripción de la línea 002121:

```
002090     01  FICHA-OPERARIO.
002100       02  OPERARIO.
002110         03  CODIGO                  PICTURE  X(5).
002120         03  APELL-NOMBRE            PICTURE  A(35).
002121         03  APE-NOM  REDEFINES  APELL-NOMBRE  PICTURE  X(35).
002130         03  FECHANACI.
002140           04  DIA                   PICTURE  99.
```

Con esta descripción, se ha dado a un campo de la memoria dos nombres APELL–NOMBRE y APE–NOM, así como dos formatos A(35) y X(35), con lo que cuando en la PROCEDURE DIVISION se nombre el segundo, podrá almacenarse en él cualquier carácter del código E.B.C.D.I.C. y cuando se nombre el primero, sólo se podrán almacenar caracteres alfabéticos.

– Relacionando el formato de la cláusula REDEFINES con el ejemplo anterior, APE–NOM se refiere a nombre–de–datos–1 y APELL–NOMBRE a nombre–de–datos–2.

– REDEFINES debe ser la primera cláusula que siga al nombre–de–datos.

No sería correcto escribir:

```
002120          03 APELL-NOMBRE          PICTURE A(35).
002121          03 APE-NOM  PICTURE X(35)  REDEFINES APELL-NOMBRE.
```

– Los números de nivel empleados en la descripción de APE–NOM y APELL–NOMBRE deben ser iguales.

– Por ser APE–NOM y APELL–NOMBRE campos elementales, la descripción de APE–NOM debe ir inmediatamente después de la descripción de APELL–NOMBRE.

No sería correcto escribir:

```
002090      01 FICHA-OPERARIO.
002100          02 OPERARIO.
002110             03 CODIGO                PICTURE X(5).
002120             03 APELL-NOMBRE          PICTURE A(35).
002130             03 FECHANACI.
002140                04 DIA                PICTURE 99.
002150             ----------------------------------------
002160             ----------------------------------------
002170             03 APE-NOM  REDEFINES APELL-NOMBRE  PICTURE X(35).
```

– La redefinición de un campo compuesto se hace inmediatamente después del último de los campos elementales que forman parte de aquél,y con el mismo número de nivel del campo compuesto:

```
003040          03 NUMHIJOS              PIC 99.
003050          03 FECHAINGRESO.
003060             04 DIA                PIC 99.
003070             04 MES                PIC 99.
003080             04 ANNO               PIC 99.
003090          03 FECHAIN  REDEFINES FECHAINGRESO  PIC X(6).
```

El siguiente ejemplo muestra otro empleo de la cláusula REDEFINES.

EJEMPLO. Introducir dentro del registro FICHA–OPERARIO del programa NOMINA las inscripciones necesarias que permitan calcular el número de operarios de la empresa en la que su CODIGO contenga el dígito 7 como cifra de las unidades.

Tal como se hizo la descripción de CODIGO (línea 002110) con PICTURE X(5) y debido al número limitado de tipos de declaraciones que puede ejecutar el ordenador, no hay declaración COBOL con la que poder preguntar si la última cifra de CODIGO es 7; por tanto, una posible solución se obtendría empleando la cláusula REDEFINES:

```
002090     01  FICHA-OPERARIO.
002100         02  OPERARIO.
002110             03  CODIGO                    PICTURE X(5).
002120             03  CODI    REDEFINES CODIGO.
002130                 04  DECENAS              PICTURE X(4).
002140                 04  UNIDADES             PICTURE 9.
002150             03  APELL-NOMBRE              PICTURE X(35).
002160     .....
```

Con la redefinición del campo CODIGO se ha conseguido:

– Que dicho campo se pueda denominar también como CODI.

– Darle un nuevo formato, ya que se le ha subdividido en dos partes, asignando a la primera cuatro caracteres, dándola el nombre de DECENAS; mientras que a la segunda se le asigna un carácter y se la da el nombre de UNIDADES; es decir, la redefinición incluye a todos los campos subordinados a CODI, siendo la suma de sus longitudes igual a la del campo CODIGO.

A partir de esta descripción, ya es posible interrogar si el contenido del campo UNIDADES es igual o no a 7. Es decir, en la PROCEDURE DIVISION se podría escribir:

```
180         IF UNIDADES = 7 ADD 1 TO CONTA ELSE .....
190
```

Existen compiladores con los que es posible la redefinición múltiple de un campo de datos:

```
.....
160     02  ALFA              PIC X(4).
170     02  BETA    REDEFINES ALFA    PIC 999V9.
180     02  ZETA    REDEFINES ALFA    PIC 9V999.
.....
```

Un empleo muy frecuente de la cláusula REDEFINES es en la creación de tablas (ver páginas 244 y 249).

3.2.9.- Cláusula VALUE

FUNCION: Inicializar un campo con un determinado contenido en la WORKING-STORAGE SECTION, así como especificar valores asociados a campos elementales.

FORMATO 1:

VALUE IS literal

- Literal puede ser del tipo numérico, no numérico, o constante figurativa.

El programador por medio de las descripciones hechas en la DATA DIVISION, puede reservar áreas o campos de la memoria donde irán, sucesivamente, situándose los datos del problema contenidos en cada registro del archivo. El contenido de estos campos antes del comienzo de la ejecución del programa es desconocido, ya que contendrá información o datos de trabajos anteriores. Sin embargo, el programador, con el empleo de la cláusula VALUE, puede en el mismo programa, además de reservar campos de la memoria, asignarles un contenido, el cual será su valor inicial al comienzo de la ejecución del programa.

EJEMPLO:

```
004051  77  CONTALINEAS                PICTURE 99          VALUE ZERO.
004060  77  CONTADOR                   PICTURE 9(5)        VALUE ZEROS.
004070  77  TOTALBRUTO                 PICTURE 9(6).
004080  77  TRIENIOS                   PICTURE 9(5).
```

Para el campo TOTALBRUTO se reservan seis posiciones en la memoria, siendo imprevisible su contenido hasta que no se procesen los datos contenidos en la ficha del primer operario. Sin embargo, al campo CONTADOR se le reservan cinco posiciones y se le fija el contenido inicial: CERO. En este caso la inicialización es necesaria, ya que dicho campo va a ser utilizado para contabilizar el número de operarios de la empresa y de no haberlo inicializado con dicho valor, los resultados que se obtendrían serían imprevisibles y, en algunos casos, se detendría la ejecución del programa. Por el contrario, inicializado con el valor cero, ya sí se le puede ir sumando unidad a unidad cada vez que se procesan los datos de cada operario.

En el ejemplo anterior, el campo CONTADOR se inicializó con el valor cero y durante la ejecución del programa fue variando su valor; sin embargo, también puede inicializarse un determinado campo, y permanecer CONSTANTE su contenido durante la ejecución del programa, como sucede con los campos descritos en las líneas 005090, 005100 y 005180:

```
005080      01  CABECERA-1.
005090        02  FILLER                    PICTURE  X(34)  VALUE SPACES.
005100        02  TITULO-1                  PICTURE  X(53)  VALUE 'NOMINA DE L
005110                                  'A EMPRESA RICOSA CORRESPONDIENTE A '.
005120        02  FECHA.
005130          03  D                       PICTURE  99.
005140          03  FILLER                  PICTURE  X.
005150          03  M                       PICTURE  99.
005160          03  FILLER                  PICTURE  X.
005170          03  A                       PICTURE  99.
005180        02  FILLER                    PICTURE  X(36)  VALUE SPACE.
```

La cláusula VALUE con la función de inicialización, sólo se puede utilizar en campos elementales que se describan en la WORKING–STORAGE SECTION. Es importante recalcar, pues es un defecto muy común en el programador, la prohibición de inicializar campos de datos en la FILE SECTION.

– El literal o constante figurativa que se especifique con la cláusula VALUE ha de estar de acuerdo con el tipo de PICTURE que se especifique para el campo, es decir, si la PICTURE es de tipo numérico, el literal ha de ser numérico, o si la PICTURE es de tipo no numérico, el literal ha de ser no numérico:

```
005190      01  CABECERA-2.
005200        02  FILLER                    PICTURE  X(114)  VALUE ALL ' '.
005210        02  TITULO-2                  PICTURE  X(10)   VALUE 'HOJA NUM. '.
005220        02  NUM-HOJA                  PICTURE  9(4)    VALUE ZEROS.
005230        02  FILLER                    PICTURE  X(5)    VALUE SPACE.
```

No sería correcto, sin embargo, si se especificaran la PICTURE de los campos descritos en las líneas 005210, 220 y 230 de la forma siguiente:

```
005190      01  CABECERA-2.
005200        02  FILLER                    PICTURE  X(114)  VALUE ALL ' '.
005210        02  TITULO-2                  PICTURE  A(10)   VALUE 'HOJA NUM. '.
005220        02  NUM-HOJA                  PICTURE  A(4)    VALUE ZEROS.
005230        02  FILLER                    PICTURE  9(5)    VALUE SPACE.
```

ya que el campo NUM–HOJA sólo puede admitir letras o espacios en blanco, y no ceros como se ha especificado.

– La constante figurativa ZERO se considera literal numérico y el resto de las constantes como literales no numéricos.

Otro empleo de la cláusula VALUE, es en combinación con los nombres–de–condición, pudiendo, a diferencia del anterior formato, especificarse dentro de la FILE, WORKING y LINKAGE SECTION según el formato siguiente:

FORMATO 2:

88 nombre–de–condición VALUE IS literal

Los nombres–de–condición son nombres creados por el programador a los que se les asigna un posible valor de los que puede tomar el campo elemental a que pertenecen. Para su formación siguen las mismas reglas que los nombres creados.

Al confeccionar el programa, el programador en vez de describir el campo elemental ESTADO junto con los nombres–de–condición asociados a él, podía haberlos omitido por no ser obligatoria su descripción:

```
002170     03  PROVINCIA-NACIM   PICTURE  99.
002180     03  ESTADO            PICTURE  9.
003010     03  SEXO              PICTURE  A.
003020         88  VARON     VALUE 'V'.
003030         88  HEMBRA    VALUE 'H'.
003040     03  NUM-HIJOS         PICTURE 99.
```

lo cual obligaría a especificar la prueba de las diferentes condiciones para el cálculo del plus familiar, en la PROCEDURE DIVISION, de la forma siguiente:

```
007110 CALCULO-PLUSF.
007120     IF  ESTADO = 2  COMPUTE  PLUSF = (500 + NUM-HIJOS * 250)
007130     ELSE
007140         IF ESTADO = 3 COMPUTE  PLUSF = NUM-HIJOS * 250
007150         ELSE  MOVE  0  TO  PLUSF.
```

Sin embargo, el empleo del nuevo formato de la cláusula VALUE permite especificar dentro del programa el nombre de los posibles estados de cada operario, asociándolos a un valor determinado, es decir:

```
002180     03  ESTADO            PICTURE  9.
002190         88  SOLTERO   VALUE  1.
002200         88  CASADO    VALUE  2.
002210         88  VIUDO     VALUE  3.
```

Con esta descripción el programa se hace más legible, pues permite, dentro de la PROCEDURE DIVISION, hacer las interrogaciones de la forma siguiente:

```
007110 CALCULO-PLUSF.
007120     IF  CASADO  COMPUTE  PLUSF  = (500 + NUM-HIJOS * 250)
007130     ELSE
007140         IF  VIUDO COMPUTE  PLUSF  = NUM-HIJOS * 250
```

Permite, también, no tener que consultar el dossier de información, que acompaña a cada programa, a fin de conocer a qué estado civil pertenece.

Los nombres–de–condición deben especificarse inmediatamente después de cada campo elemental con el cual se asocian (líneas 002190 a 002210).

Al igual que en el formato anterior, el literal debe estar de acuerdo con la PICTURE del campo elemental de quien depende:

```
003010          03  SEXO                        PICTURE A.
003020              88  VARON                   VALUE 'V'.
003030              88  HEMBRA                  VALUE 'H'.
```

No sería correcto si se especificara:

```
003010          03  SEXO                        PICTURE A.
003020              88  VARON                   VALUE  1.
003030              88  HEMBRA                  VALUE  2.
```

ya que el tipo de PICTURE es alfabética y los valores de VALUE son literales numéricos.

Es importante insistir en que la cláusula VALUE asociada a nombre–de–condición se puede especificar en cualquier Sección de la DATA DIVISION excepto en la REPORT.

FORMATO 3:

```
88 nombre-de-condicion  { VALUE IS  }  literal–1
                        { VALUE ARE }

[ THRU literal–2 ] [ literal–3 [ THRU literal–4 ] ] ...
```

La diferencia que presenta este formato respecto del anterior es que a cada nombre–de–condición se le puede asociar, no un solo valor, sino un valor, valores o gama de valores (THRU).

```
    010         02  ALFA                PIC 99.
    020             88  A               VALUE 1.
    030             88  B               VALUE 5  THRU  8.
    040             88  C               VALUE 12  14  25  26.
```

En este caso se podía especificar en la PROCEDURE DIVISION:

IF B GO TO CALCULO, **o su equivalente:**
IF ALFA NOT < 5 AND NOT > 8 GO TO CALCULO

según estas dos descripciones, se puede apreciar que los nombres de condición simplifican y facilitan, en algunos casos, la confección del programa.

3.2.10.- Relaciones entre VALUE, OCCURS y REDEFINES

El empleo conjunto de las cláusulas REDEFINES, VALUE y OCCURS está sometido a las siguientes limitaciones:

— La cláusula VALUE sólo puede utilizarse con cláusulas OCCURS y REDEFINES cuando van acompañadas del número de nivel 88, es decir, sería correcto especificar:

```
190    03  ESTADO      PIC 9     OCCURS 7.
200        88 CASADO    VALUE 1.
210        88 SOLTERO   VALUE 2.
220    03  SEXO        PIC A.
230    03  PROVINCIA-NACIM  PIC 99.
```

Sin embargo, no es correcto:

```
180    02  ESTADO  OCCURS 7  PIC 9  VALUE 1.
```

— No se puede redefinir un campo de datos que contenga la cláusula OCCURS o esté subordinado a dicha cláusula.

EJEMPLO. No es correcto redefinir el campo de datos ALFA:

```
010    02  ALFA   OCCURS 25  PICTURE A(3).
020    02  BETA   REDEFINES  ALFA  PIC X(75).
```

por contener dicho campo la cláusula OCCURS.

Tampoco es correcto especificar:

```
050    02  ALFA   OCCURS 25.
060        03 ZETA   PIC X(4).
070        03 DELTA  REDEFINES  ZETA  PIC A(4).
080        03 BETA   OCCURS 4  PIC X.
```

por ser el campo ZETA subordinado a ALFA, el cual contiene la cláusula OCCURS.

Es obvio afirmar que los ejemplos anteriores son incorrectos cuando se emplea la cláusula OCCURS con la opción DEPENDING ON.

3.2.11.- Cláusula RENAMES

FUNCION: Proporcionar un nuevo nombre a un campo elemental, un campo compuesto o un grupo de campos elementales.

FORMATO:

66 de nombre–de–datos–1 RENAMES nombre–de–datos–2 [THRU nombre–de–datos–3]

El formato de la página 106 es un formato generalizado en el que se hace referencia a las posibles cláusulas que el programador puede utilizar, para llevar a cabo la descripción de un campo elemental o un grupo de campos (registro). Sin embargo, no se menciona en dicho formato la cláusula RENAMES, por no poder entrar a formar parte de la descripción de los citados campos mezclándose con dichas cláusulas, y ser obligatorio especificarla inmediatamente después de la inscripción que se haga del último campo de un registro. Por ello, esta cláusula puede considerarse independiente del registro y no se incluye junto con las demás.

Aunque esta cláusula presenta cierta similitud con la cláusula REDEFINES en cuanto a que ambas proporcionan nuevos nombres a diferentes áreas de la memoria, sin embargo, una diferencia esencial es que con la cláusula RENAMES no puede hacerse una nueva descripción de dichas áreas, modificando la serie de caracteres de la PICTURE de los campos que las componen.

Por otra parte, esta cláusula presenta una serie de peculiaridades que se van a exponer a continuación, apoyándose en las siguientes descripciones, tomadas de un programa cualquiera:

```
010
020     01  FICHA-OPERARIO.
030         02  OPERARIO.
040             03  CODIGO                 PIC X(5).
050             03  APENOM                 PIC X(35).
060             03  SUELDO                 PIC 9(5).
070             03  FECHANACI              PIC 9(6).
080         02  IDENTIFI.
090             03  DNI                    PIC 9(8).
100             03  FECHACADU              PIC 9(6).
110         02  FACTORIA.
120             03  NOMBRE                 PIC A(6).
130             03  DEPARTAMEN             PIC 9(3).
140     66  SUELDOBRUTO    RENAMES SUELDO.
150     66  DATOSOPE       RENAMES OPERARIO.
160     66  RESTODATOS     RENAMES IDENTIFI THRU FACTORIA.
170     66  INFORM         RENAMES FECHANACI THRU NOMBRE.
180     01  BETA.
190
200
```

Observando el ejemplo, se deduce que:

- Pueden describirse una o varias cláusulas RENAMES, pero todas ellas deben ir al final de cada registro lógico (01).

- La primera cláusula RENAMES (línea 140) da un nuevo nombre, SUELDOBRUTO, al campo elemental SUELDO.

- La segunda cláusula RENAMES da un nuevo nombre, DATOSOPE, al campo compuesto OPERARIO.

- La tercera cláusula RENAMES da un nuevo nombre, RESTODATOS, a la zona de la memoria que incluye todos los campos elementales pertenecientes a IDENTIFI y FACTORIA. Tiene la ventaja, por ejemplo, de que si se quiere transferir el contenido de los campos compuestos citados, a otro campo denominado ALFA, con una sola declaración MOVE sería suficiente,es decir:

  MOVE RESTODATOS TO ALFA.

- La cuarta cláusula RENAMES redenomina con el nombre INFORM, a los campos comprendidos entre FECHANACI y NOMBRE, ambos incluídos. En este caso, INFORM, aunque esté comprendido entre campos elementales,pasa a tener las mismas propiedades y se comporta como un campo compuesto a efecto , por ejemplo, de seguir las reglas que se describen para este tipo de campo en la declaracion MOVE acerca de transferencias permitidas,o en la IF sobre comparaciones permitidas.

La redenominación descrita en las líneas 160 y 170, son las de mayor aplicación pues dan un nombre a un grupo de campos compuestos o a un grupo de campos elementales, que como se ha visto, puede suponer, entre otras cosas, ahorro de declaraciones en la PROCEDURE. No ocurre lo mismo con las redenominaciones de las líneas 140 y 150, ya que tiene menos aplicación el que dentro de un programa un campo elemental o compuesto puedan llamarse de dos formas diferentes, tanto más teniendo en cuenta que dichos campos conservan sus descripciones en cuanto a PICTURE , VALUE , etc.

Otras reglas:

- Nombre–de–datos–2 no puede estar descrito con nivel 01,77,88.
- Una cláusula RENAMES no puede redenominar a otra cláusula RENAMES.
- Tanto nombre–de–datos–2 como nombre–de–datos–3 no pueden tener una cláusula OCCURS, ni puede ninguno de ellos estar subordinado a un campo que contenga una cláusula OCCURS en una descripción.

4.- WORKING-STORAGE SECTION

Normalmente, el empleo que se hace de las áreas de entrada de la memoria, que se describieron en la FILE SECTION, es para ir almacenando los datos del problema procedentes de un medio externo como es el archivo de entrada, sin embargo,hay veces que son necesarias otras áreas de la memoria donde almacenar los resultados parciales que se vayan produciendo durante el proceso. Teniendo en cuenta, que si en un área o en parte de ella (campo de datos) hay almacenado un dato y se pretende introducir otro, éste último sustituye al anterior, es lógico, que si para ello se utilizarán las áreas de entrada y existiera en éllas algún dato cuyo contenido se desea conservar mientras dure toda o parte de la ejecución del programa, no se podría utilizar dicho campo para almacenar resultados parciales, por lo que será necesario el empleo de otros campos de la memoria que realicen dicha función.

Como en la mayor parte de los programas, los datos del problema una vez introducidos en la memoria del ordenador, necesitan una previa elaboración (cálculos aritméticos, transferencias, etc) antes de llegar a obtener los resultados, es normal tener que utilizar otras áreas o campos de la memoria que no sean las áreas de entrada/salida, donde se efectúe dicha elaboración.

FUNCION:

La WORKING-STORAGE SECTION proporciona el medio de poder describir registros que no forman parte de los archivos de datos de entrada/salida, así como campos independientes (13) donde, entre otras funciones:

- Se puedan almacenar, bien los resultados intermedios producidos durante el proceso de cada registro o bien un determinado contenido, el cual se conservará constante hasta el final del programa.
- Se editen los resultados antes de ser impresos,a fin de facilitar su lectura.
- Servir de acumuladores de totales, parciales, etc.

En el ejemplo de la página siguiente:

TOTALBRUTO, TRIENIOS, etc, son campos que se emplearán para almacenar resultados intermedios,cuyos valores cambiarán con el proceso de cada registro que contiene los datos de cada operario.

IMPRESO-S es un registro donde se editarán los resultados.

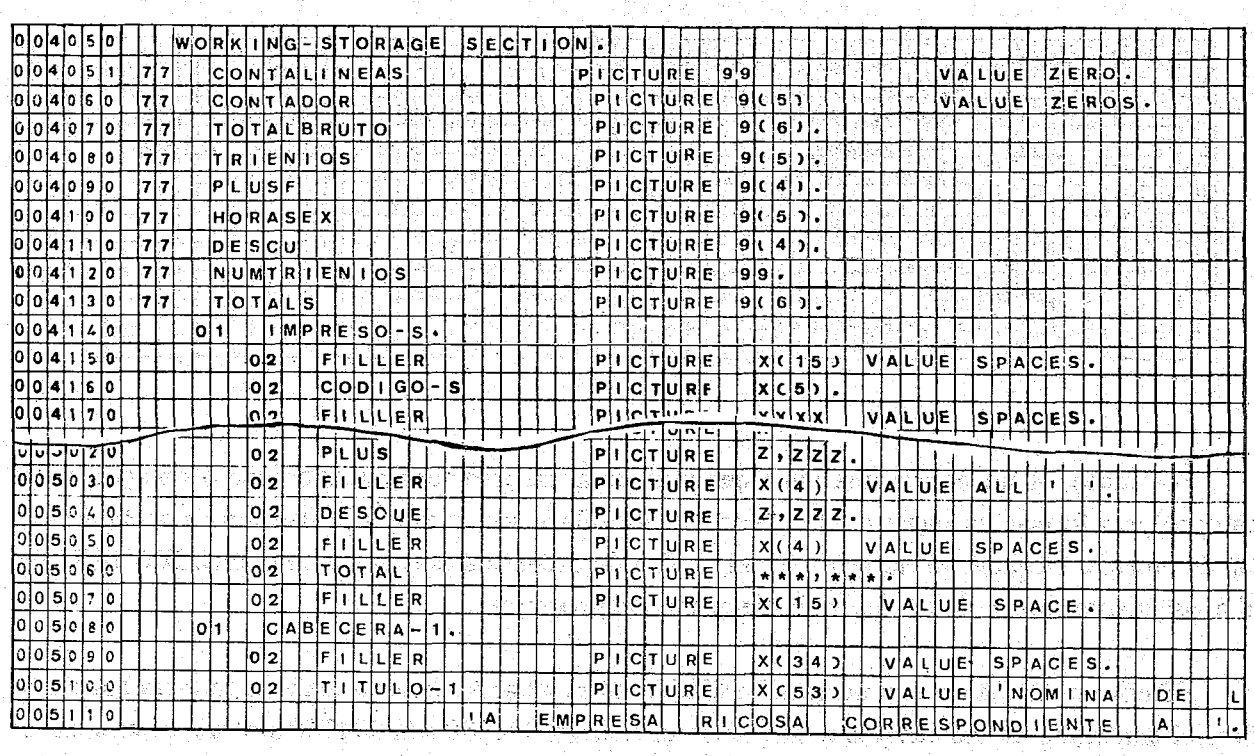
CABECERA-1 es un registro que sirve para escribir una línea con un encabezamiento.

TITULO-1 es un campo cuyo contenido es "NOMINA DE LA EMPRESA ..." el cual permanecerá constante durante la ejecución del programa.

CONTADOR es un acumulador donde se irán contabilizando el número de operarios que componen la Nómina.

---

(13) La denominación de campos independientes es con la que se acostumbra a nombrar a los campos elementales de la WORKING-STORAGE SECTION que no forman parte de un registro.



FORMATO:

[77 Cláusulas de descripción de un campo independiente] ...

[01 Cláusulas de descripción de un registro] ...

Los campos que se especifican en la WORKING pueden agruparse siguiendo el formato empleado en la descripción de un registro, de la misma forma que se hizo en la FILE SECTION (14) o bien ser campos independientes, en cuyo caso van precedidos del número de nivel 77.

Así, el registro IMPRESO–S agrupa una serie de campos entre los cuales existe una relación, puesto que el contenido de los mismos es la información de cada individuo que se imprimirá en cada línea del impreso.

Sin embargo, los campos independientes no están agrupados, y normalmente contendrán resultados parciales que sólo interesará conservar temporalmente, como en el caso de NUMTRIENIOS, o bien como en el de CONTADOR, que tampoco necesita estar formando parte de un registro, para mostrar, por medio de la declaración de la línea 007030, su contenido al final del proceso.

No obstante, no hay una regla que impida que los campos independientes puedan agruparse formando un registro, como sigue:

---

(14) Hay que hacer notar que no sólo se entiende por registro aquél que se describe con nivel 01 en la FILE SECTION a continuación de la descripción de un archivo, sino también aquél que se describe con nivel 01 en cualquier Seccion de la DATA DIVISION, aunque no formen parte de un archivo.

```
004050   WORKING-STORAGE SECTION.
004060 01   CAMPOS.
004070      02   CONTALINEAS          PICTURE 99      VALUE ZERO.
004080      02   CONTADOR             PICTURE 9(5)    VALUE ZERO.
004090      02   TOTALBRUTO           PICTURE 9(6).
004100      02   TRIENIOS             PICTURE 9(5).
004110      02   PLUSF                PICTURE 9(4).
004120      02   HORASEX              PICTURE 9(5).
004130      02   DESCU                PICTURE 9(4).
004140      02   NUMTRIENIOS          PICTURE 99.
004150      02   TOTALS               PICTURE 9(6).
```

Por tanto, al no existir una regla que indique al programador cuando debe o no agrupar los campos para formar un registro, el criterio que este debe seguir es que la elección de cualquiera de las dos formas elegidas, se traduzca en un ahorro de órdenes a especificar en la PROCEDURE, con el consiguiente ahorro de tiempo en la ejecución del programa.

Efectivamente, si en el programa NOMINA, entre otras acciones a realizar, hubiera que transferir el contenido de los campos descritos desde la línea 004080 a la 004130 a otra u otras áreas de la memoria, lo efectivo sería hacer la descripción siguiente:

```
004050   WORKING-STORAGE SECTION.
004060   77  CONTALINEAS          PIC 99      VALUE ZERO.
004070   77  CONTADOR             PIC 9(5)    VALUE ZEROS.
004080   77  TOTALBRUTO           PIC 9(6).
004090   01  REGIS.
004100       02  TRIENIOS         PIC 9(5).
004110       02  PLUSF            PIC 9(4).
004120       02  HORASEX          PIC 9(5).
004130       02  DESCU            PIC 9(4).
004140       02  NUMTRIENIOS      PIC 99.
004150       02  TOTALS           PIC 9(6).
```

ya que permite con una sola declaración: MOVE REGIS TO ... hacer la transferencia del contenido de todos los campos, mientras que describiéndolos con nivel 77 habría que escribir una declaración MOVE para transferir cada campo.

La cláusula VALUE puede utilizarse en esta Sección, bien como se hizo en la FILE SECTION junto con los nombres-de-condición, o bien para inicializar campos de datos con un determinado valor (líneas 004150, 005100, 005210 etc. del programa NOMINA).

El resto de las cláusulas:REDEFINES, OCCURS, etc. que se especifiquen dentro de la descripción de un registro, tienen las mismas limitaciones en esta Sección, que las descritas en la FILE SECTION.

5.- EJERCICIOS

1.– ¿Cuál de las siguientes inscripciones se debe codificar dentro del margen B?

A.- FD
B.- 03
C.- 01
D.- BLOCK CONTAINS

2.– Por cada archivo de entrada y/o salida que se utilice en un programa, debe especificarse una FD (FILE DESCRIPTION).

CIERTO FALSO

3.– Por cada FD (descripción de archivo)deben codificarse tantas descripciones de registro con nivel 01 como registros de diferentes formato contenga cada archivo.

CIERTO FALSO.

4.– Simplificar las inscripciones hechas en el programa NOMINA (pág. 23 y 24) en la descripción de los archivos: ARCHIVO–OPERARIOS, CINTA–NOMINA y ARCHI–IMPRESO, escribiendo únicamente las cláusulas obligatorias y omitiendo aquéllas que son opcionales.

5.– Teniendo en cuenta cuanto se ha especificado en el programa ACTUALIZ, cuya descripción se ha iniciado en el ejercicio 7 del CapítuloIII, el programador debe hacer en la DATA DIVISION las descripciones de cada uno de los archivos y registros,para lo cual tendrá presente que:

– El archivo denominado ARCHI–MAES está constituído por registros de longitud fija, agrupados en bloques de quince registros y lleva grabadas etiquetas standard.

La composición y contenido de cada registro es el siguiente:

POSICIONES	CONTENIDO	TIPO DEL CAMPO
De la 1 a la 8	Código de la pieza	NUMERICO
De la 9 a la 18	Existencias en almacén	NUMERICO

– El segundo archivo, ARCHI–MODIFI, está compuesto por registros de longitud fija, no agrupados en bloque,ni existen grabadas etiquetas. Cada registro contiene las variaciones (altas, bajas o modificaciones) habidas durante el día, siendo su composición y contenido el siguiente:

POSICIONES	CONTENIDO	TIPO DEL CAMPO
De la 1 a la 8	Código de la Pieza	NUMERICO
De la 9 a la 18	Entradas o salidas en el día	NUMERICO
En la 19	Clave	NUMERICO

– El tercer archivo, que será resultado de la actualización, ARCHI–ACTUAL, tiene la misma constitución que ARCHI–MAES.

En cada registro debe figurar la siguiente información:

POSICIONES	CONTENIDO	TIPO DEL CAMPO
De la 1 a la 8	Código de la Pieza	NUMERICO
De la 9 a la 18	Existencias actualizadas	NUMERICO

– Describir en la WORKING–STORAGE SECTION, tres campos de trabajo de tipo numérico,denominándose CAMBIO1, CAMBIO2 y CAMBIO3, que sean capaces de reservar una posición de memoria.

6.– En las siguientes descripciones, que nada tienen que ver con el programa NOMINA, se ha cometido un error. Explicar el motivo.

```
003010  FD  ARCHI-CINTA      LABEL RECORD STANDARD.
003020  01  REGIS.
003030      02 CODIGO         PIC 9(5).
003040      02 NOMBRE         PIC A(40).
003050      02 NUMHIJOS       PIC 99.
003060      02 FILLER         PIC X(10).
003070  FD  ARCHI-CINTA      LABEL RECORD OMITTED.
003080  01  PRODUCCION.
003090      02 CODIGO         PIC X(3).
003100      02 DATOS          PIC X(50).
```

7.– ¿Es correcto hacer en un programa las descripciones siguientes ?

```
004160  FD  ARCHIOPE     LABEL RECORD OMITTED  RECORDING F.
004170      01 FICHA          PIC X(80).
004180  FD  ARCHIMPRE    LABEL RECORD STANDARD.
004190      01 FICHA.
004200         02 CAMPOA      PIC X(5).
004210         02 CAMPOB      PIC 9(18).
```

8.– ¿Cuál de las siguientes PICTURES no es de tipo numérico?

A.- S999V99
B.- PPPP99
C.- 999V99
D.- 999 . 99

9.– Describir un campo elemental de tipo numérico, capaz de contener al número –1512,750

10.– La serie de caracteres que deben seguir a una PICTURE, que sean capaces de reservar siete octetos donde almacenar el literal no numérico 'AMX 3052' sería:

A.- PIC 9(7)
B.- PIC X(7)
C.- PIC X(3)9(4)
D.– PIC A(3)9(4)

11.– Describir la serie de caracteres de la PICTURE de un campo de edición que reuna las siguientes propiedades:

– Almacene tres cifras enteras y una decimal.

– Suprima ceros no significativos de la parte entera sustituyéndolos por blancos, pero no la parte decimal aunque su valor sea 0.

– Si el número a almacenar es negativo,aparezca impresa la palabra DB.

12.– Describir la serie de caracteres de la PICTURE de un campo de edición que reuna las siguientes propiedades:

– Que almacene cinco cifras enteras y dos decimales.

– Que suprima todos los ceros no significativos de la parte entera, excepto el más próximo al punto decimal, por ejemplo, si el número a editar es 00000 . 28, pueda imprimirse 0 . 28.

– Que entre las unidades de millar y centenas intercale una coma, y entre la parte entera y decimal un punto.

– Que inmediatamente delante de la primera cifra significativa, o de la cifra cero más próxima al punto decimal, imprima el simbolo $, por ejemplo, $ 74 . 22 ó $ 0 . 15.

– Que al final del número coloque un signo + si el número es positivo o un signo – si el número es negativo.

13.– ¿Son válidas las cláusulas USAGE, BLANK y VALUE, especificadas en algunos de los siguientes campos de datos tomados del programa NOMINA?

```
001070
001080     01 REGISTRO-OPERARIO.
001090        02 DATOSCIN          PIC X(80) USAGE COMP.
001100        02 TOTAL-SUELDOCIN   PIC 9(6)  USAGE COMP-3.
001110
001120     WORKING-STORAGE SECTION.
001130     77 CONTALINEAS          PIC 99    VALUE ZERO USAGE COMP.
001140     77 CONTADOR             PIC 9(5)  VALUE ZERO USAGE COMP-2.
001150     77 TOTALBRUTO           PIC 9(6)  USAGE DISPLAY.
001160     77 TRIENIOS             PIC 9(5)  JUSTIFIED RIGHT.
001170
001180        02 SUELDO-S          PIC ZZ.ZZZ  BLANK WHEN ZERO.
001190        02 FILLER            PIC X(4)    VALUE ALL ' '.
```

14.– La siguiente descripción podría haberse especificado dentro del programa NOMINA, pero antes deberá el programador corregir un error que ha cometido ¿Cuál es éste?

```
200   02 PROVINCIA-NACIM    PIC XX.
210      88 ALAVA           VALUE 01.
220      88 ALBACETE        VALUE 02.
230      88 ALICANTE        VALUE 03.
...      ...                ...
410      88 ZARAGOZA        VALUE 50.
```

15.– La siguiente descripción en la que se especifican los nombres de condición SOLTERO, CASADO, etc., parece correcta, pero no lo es ¿Cuál es el motivo?

```
110
120   03 ESTADO-CIVIL      PIC 9.
130      04 SOLTERO        VALUE 3.
140      04 CASADO         VALUE 4.
150      04 VIUDO          VALUE 5.
160      04 DIVORCIADO     VALUE 6 THRU 9.
170   03 ...
180
```

16.– ¿Es correcta la Redefinición del campo ALFA?

```
010
020   02 ALFA             PIC X(7).
030   02 BETA   REDEFINES ALFA.
040      03 BE1           PIC A(3).
050      03 BE2           PIC 999.
060   02 ...
```

17.– El programador ha querido describir una tabla dentro de la WORKING-STORAGE SECTION ¿Puede hacerlo dentro de esta Sección? En caso afirmativo ¿es correcta la descripción de la tabla?

```
070   WORKING-STORAGE SECTION.
080      77 SUBINDICE       PIC S99 USAGE COMP   VALUE 01.
090      01 ELEMEN-TABLA     PIC AAAA    OCCURS 50 TIMES.
100
```

18.– La siguiente descripción es incorrecta. Explicar el motivo.

```
120   WORKING-STORAGE SECTION.
130   FD  ARCHIMPRE      LABEL RECORD OMITTED.
140      01 IMPRESO       PIC X(133).
150
```

19.– En las siguiente descripción existe un error. Explicar el motivo.

```
006010 WORKING-STORAGE SECTION.
006020  01  CAMPOS.
006030      02  DATOS          PIC X(120).
006040      02  FILLE          PIC 9(3).
006050  01  CAMPOS.
006060      02  NUMRIEZA       PIC 9(5).
006070      02  CALIBRE        PIC 9(4)V99.
```

20.– El programador, al escribir algunas de las siguientes líneas del programa NOMINA, ha cometido errores que serán detectados por el compilador. Decir cuáles son estos errores.

```
002040 DATA DIVISION.
002050     FILE SECTION.
002060     FD ARCHIVO-OPERARIOS    RECORDING MODE IS F
002070             LABEL RECORD IS OMITTED   BLOCK   0  RECORDS
002080             DATA  RECORD IS FICHA-OPERARIO.
002090      02 FICHA-OPERARIO.
002100         02  OPERARIO.
002110           03  CODIGO                PICTURE  X( 5 ).
002120           03  APELL-NOMBRE          PICTURE  X(35).
002130           03  FECHANACI             PICTURE  9(6).
002140               05 DIA                PICTURE  99.
002150               05 MES                PICTURE  99.
002160               05 ANNO               PICTURE  99.
002170           03  PROVINCIA-NACIM       PICTURE  99     VALUE 01.
002180           03  ESTADO.
002190               88  SOLTERO  VALUE  1.
002200               88  CASADO   VALUE  2.
002210               88  VIUDO    VALUE  3.

003180     FD  CINTA-NOMINA     RECORDING F
003190         BLOCK  10  RECORDS
003200         RECORD 86  CHARACTERS
003220         DATA  RECORD  REGISTRO-OPERARIO.
003230  01  REGISTRO-OPERARIO.
003240      06  DATOSCIN              PICTURE 9(80).
003250      06  TOTAL-SUELDOCIN       PICTURE 9(6).

004010     FD   ARCHI-IMPRESO   RECORDING IS F.
004020          LABEL RECORDS ARE OMITTED.
004040  01  IMPRESO                   PICTURE X(133)    VALUE SPACES.
004050 WORKING-STORAGE SECTION.
004051 01  CONTALINEAS                PICTURE 9(2)      VALUE ZERO.
004060 77  CONTADOR                   PICTURE 9(5)      VALUE ZEROS.
004070 77  TOTALBRUTO                 PICTURE 9(6)      VALUE 000000.
004080 88  TRIENIOS                   PICTURE 9(5)

005080     01  CABECERA-1.
005090         02  FILL               PICTURE  X(36)    VALUE SPACES.
005100         02  TITULO-1           PICTURE  A(53)    VALUE 'NOMINA  DE  L
005110-                                'A  EMPRESA  RICOSA  CORRESPONDIENTE  A  '.

005190     01  CABECERA-2.
005200         02  FILLER             PICTURE  X(114)   VALUE ALL  .
005210         02  TITULO-2           PICTURE  X(10)    VALUE 'HOJA NUM. '.
005220         02  NUM-HOJA           PICTURE  9(4)     VALUE ZEROS.
005230         02  FILLER             PICTURE  X(5)     VALUE BLANCOS.
```

CAPITULO V

PROCEDURE DIVISION

1.- GENERALIDADES

Por la descripción de archivos y registros realizada en la DATA DIVISION, el compilador a la vez que reserva áreas de la memoria, da un nombre a cada una de ellas para poder ser identificada. Queda, sin embargo, por enumerar dentro del programa la serie de acciones necesarias para llevar a cabo lo que generalmente se denomina PROCESO. Dicha enumeración se hace en la PROCEDURE DIVISION.

Un proceso normal, en líneas generales, comprende:

- Preparación de los archivos para poder ser leídos.
- Introducción de los datos en las áreas de la memoria que se han reservado para cada campo de los registros.
- Realización de acciones como: operaciones aritméticas, transferencias, etc, con el contenido de dichos campos.
- Grabación o impresión de resultados.
- Cierre de archivos y finalización del proceso.

En el lenguaje COBOL dichas acciones están expresadas por los siguientes verbos:

Aritméticos: ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE.

Bifurcación: GO , PERFORM, ALTER, EXIT.

Comparación: IF (16).

Detención: STOP.

(16) Aunque IF no es un verbo, sin embargo, se puede utilizar como tal, ya que durante la compilación del programa genera acciones a ejecutar con los datos.

Movimiento: MOVE, EXAMINE, SET.

Clasificación: SORT, RELEASE, RETURN.

Entrada/Salida: READ, WRITE, REWRITE, ACCEPT, DISPLAY, OPEN, CLOSE.

Manejo de tablas: SEARCH.

Directrices para el Compilador: NOTE.

Creador de listados: INITIATE, GENERATE, TERMINATE.

Cada uno de los verbos citados, seguidos de uno o varios operandos y/o palabras COBOL recibe el nombre de DECLARACION o INSTRUCCION. Si el operando u operandos son el nombre de: un campo de datos, un registro, archivo, sección, etc., de los descritos en alguna parte de la ENVIRONMENT o DATA DIVISION, la declaración se llama IMPERATIVA.

Si el operando es una condición que ha de probarse, para determinar por cual de los dos caminos posibles ha de continuar la ejecución del programa, se denomina declaración CONDICIONAL.

```
007010 LEER.
007020     READ ARCHIVO-OPERARIOS; AT END,
007030     DISPLAY '                REGISTROS PROCESADOS = ' CONTADOR,
007040     CLOSE   ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050     STOP RUN.
007060 CALCULO-TRIENIOS.
007070     IF  MES OF FECHAINGRESO < M,
007080     COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3 ELSE
007090     COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100     COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
007110 CALCULO-PLUSF.
```

Cualquiera de las declaraciones de las líneas 007030, 007040, 007050 son declaraciones imperativas, mientras que la declaración de la línea 007020, es decir, el verbo READ con la condición AT END, es una declaración condicional.

Una o varias declaraciones sucesivas que terminen en un punto se denominan SENTENCIAS.

Las declaraciones de las líneas 007070, 007080 y 007090 hasta el punto, constituyen una sentencia. La declaración de la línea 007010, además de ser una declaración es también una sentencia por terminar en un punto.

Una o varias sentencias sucesivas pueden también reunirse formando un PROCEDIMIENTO (17) o PARRAFO. Todo procedimiento comienza con un nombre de procedimiento seguido de

(17) Por prestarse a posible confusión los párrafos descritos en la IDENTIFICATION, ENVIRONMENT y DATA DIVISION, de los que se especifican en las PROCEDURE, se prefiere llamar a estos párrafos últimos: procedimientos.

un punto, y termina en el siguiente nombre de procedimiento. Los nombres: CALCULO–TRIENIOS, CALCULO–PLUSF, etc., son nombres de procedimiento, y las declaraciones comprendidas de la línea 007070 a la 007100, constituyen en procedimiento.

Uno o varios procedimientos sucesivos forman una SECCION. Para designarla se emplea un nombre seguido de la palabra SECTION y un punto, y comprende desde dicho nombre hasta el siguiente nombre de sección.

Tanto los nombres de procedimiento como los de sección que se especifiquen en esta DIVISION son variables, ya que los crea el programador, a diferencia de los nombres de párrafo y sección en las otras tres Divisiones, que son fijos.

Dos nombres de procedimiento dentro de una misma Sección no pueden ser iguales, ya que de ser así, cualquier referencia a este nombre hecha en una declaración dentro de esta División, haría imposible al compilador poder conocer a cual de los dos procedimientos se refiere.

Es necesario especificar nombres de procedimiento, siempre que haya que alterar el orden secuencial en que normalmente se ejecutarían las declaraciones como consecuencia del resultado de la prueba de una declaración condicional, ya que estos nombres han de servir como punto de referencia hacia donde dirigir al ordenador para continuar la ejecución del programa. Es decir, si en el programa NOMINA, los datos de todos los operarios reunieran las mismas condiciones, el ordenador ejecutaría todas las declaraciones contenidas en la PROCEDURE, una tras otra en el orden especificado hasta el final; pero al ser las condiciones diferentes para cada operario, ya que los hay solteros, casados, viudos, etc., el ordenador ejecutará unas declaraciones y omitirá otras, y por tanto efectuará saltos o bifurcaciones a los nombres de procedimiento o de sección.

2.- Estructura de la PROCEDURE

PROCEDURE DIVISION

[DECLARATIVES.

{nombre-de-sección SECTION. Sentencia USE.

{nombre-de-procedimiento. {sentencia} ... } ... } ...

END DECLARATIVES.]

[nombre-de-sección SECTION [prioridad .]

{nombre-de-procedimiento. {sentencia } ...} ...} ...

3.-VERBOS ARITMETICOS

3.1.-DECLARACIONES ARITMETICAS

El lenguaje COBOL fue concebido para resolver problemas de tipo comercial, por lo que no se incluyó dentro de sus especificaciones la posibilidad de resolver cálculos complejos,como pudieran ser: ecuaciones lineales, trigonométricas, integrales, etc, sino simplemente poder realizar las operaciones aritméticas elementales,como la suma, resta, multiplicación y división o una combinación de dichas operaciones.

Las operaciones aritméticas las ejecuta el ordenador por medio de las declaraciones aritméticas, las cuales están formadas por los verbos aritméticos: ADD,SUBTRACT, etc., seguidos de uno o varios operandos elegidos entre una combinación de:

- Campos elementales con PICTURE de tipo numérico y/o de edición (18).
- Literales numéricos.

En cualquiera de los dos casos,su tamaño no debe ser superior a 18 dígitos.

3.2.- Declaración ADD

FUNCION: Sumar dos o más campos y/o literales, colocando el resultado en el último campo (identificador) especificado en la declaración.

FORMATO 1:

```
ADD { identificador-1 } [ identificador-2 ] ... TO identificador-m [ ROUNDED ]
    { literal-1       } [ literal-2       ]

[ identificador-n [ ROUNDED ] ] ... [ ON SIZE ERROR declaración-imperativa ]
```

FORMATO 2:

```
ADD { identificador-1 } { identificador-2 } [ identificador-3 ] ... GIVING identificador-m [ ROUNDED ]
    { literal-1       } { literal-2       } [ literal-3       ]

[ ON SIZE ERROR declaración-imperativa ]
```

(18) Los campos de edición pueden especificarse a continuación de la palabra GIVING, pero sólo intervendrán en la operación aritmética como receptores del resultado calculado.

El siguiente programa contiene las inscripciones que se utilizarán como ejemplos en la descripción de las diferentes declaraciones aritméticas. En él se incluyen declaraciones que utilizan como operandos campos inicializados con un determinado valor, y campos que no se han incializado por no ser necesario, ya que van a emplearse,únicamente, como receptores del resultado.

Hay, también, que tener presente que a medida que se van ejecutando las declaraciones incluidas en el programa, algunos de los campos, que fueron inicializados con un determinado valor, van modificando su contenido, por lo que al ejecutar una determinada declaración, hay que tomar el valor que en ese momento contengan los campos y no aquél con el que se inicializó; así,por ejemplo, CAMPOB se inicializa con el valor 463.20, pero después de ejecutar la declaración de la línea 001190 pasa a contener el valor 475.70, y cuando ejecuta la declaración de la línea 001200 pasará a contener el valor 748.9 etc.

```
PROGRAMADOR                    FECHA          PERFORACIONES
Página  Línea  A
1   3   4   6  7  8  12   16   20   25   30   35   40   45   50   55   60   65
001 010    IDENTIFICATION DIVISION.
001 020      PROGRAM-ID.  CALCULO.
001 030    ENVIRONMENT DIVISION.
001 040    DATA DIVISION.
001 050      WORKING-STORAGE SECTION.
001 060        77   CAMPOA          PICTURE    99V9   VALUE 12.5.
001 070        77   CAMPOB          PICTURE   999V99  VALUE 463.20.
001 080        77   CAMPOC          PICTURE    99      VALUE 35.
001 090        77   CAMPOI          PICTURE  9999      VALUE 57.
001 100        77   CAMPOK          PICTURE     9      VALUE  4.
001 110        77   CAMPOL          PICTURE   999V9    VALUE 126.4.
001 120        77   CAMPOX          PICTURE    99V9    VALUE 89.5.
001 130        77   CAMPOP          PICTURE   999.
001 140        77   CAMPOH          PICTURE   999V9.
001 150        77   CAMPOE          PICTURE ZZZ9.99.
001 160        77   CAMPOD          PICTURE 99999V99.
001 170        77   CAMPOT          PICTURE   999V9.
001 180    PROCEDURE DIVISION.
001 190      A.  ADD  CAMPOA  TO  CAMPOB
001 200          ADD  100, CAMPOA, 125.7, CAMPOC  TO  CAMPOB
001 205          ADD CAMPOA, CAMPOK, -9  TO CAMPOB, CAMPOX
001 210          ADD CAMPOA, 250, CAMPOC  GIVING  CAMPOD
002 010          ADD  CAMPOA, 250, CAMPOC  GIVING  CAMPOE
002 020          ADD  CAMPOA, 1400.28  GIVING  CAMPOH
002 030          ADD  CAMPOA, 1400.28  GIVING  CAMPOH  ON SIZE ERROR
002 040                                                GO TO C.
002 050      B.  ADD  CAMPOA, 200.74, 5.125  GIVING  CAMPOT  ROUNDED
002 060          SUBTRACT  CAMPOK  FROM  CAMPOC
002 070          SUBTRACT  5, CAMPOK, 2.1  FROM  CAMPOA
002 080          MULTIPLY  20  BY  CAMPOI
002 090          DIVIDE  CAMPOK  INTO  CAMPOL
002 100          COMPUTE  CAMPOP = 125
002 110          COMPUTE  CAMPOH = CAMPOT.
002 120      C.  STOP RUN.
```

Opción TO.

Si en la declaración se emplea la opción TO, el valor de identificador–m se suma a los valores de todos los identificadores y/o literales que se especifiquen en la declaración, y el valor resultante sustituye al valor existente en identificador–m.

```
001180 PROCEDURE DIVISION.
001190     A. ADD CAMPOA TO CAMPOB
001200        ADD 100, CAMPOA, 125.7, CAMPOC TO CAMPOB
001205        ADD CAMPOA, CAMPOK, -9 TO CAMPOB CAMPOX
```

Según las descripciones de las líneas 001060 y 001070 el contenido de CAMPOA y CAMPOB en el momento de ejecutarse la declaración de la línea 001190 es: 12.5 y 463.20 respectivamente. El ordenador ejecuta dicha declaración en el orden siguiente:

– Alinea los operandos por el punto decimal:

CAMPOA			1	2	5	
CAMPOB	4	6	3	2	0	

– Efectúa la suma 12.5 + 463.20 = 475.70

– El resultado lo transfiere a CAMPOB.

Por tanto, después de ejecutada la declaración, CAMPOA continúa con su valor inicial, y CAMPOB sustituirá su valor inicial por el resultado de la suma:

4	7	5	7	0

La declaración de la línea 001200 está compuesta por operandos que son mezcla de identificadores (campos de datos) y literales numéricos. Los valores contenidos en cada campo son: CAMPOA = 12.5, CAMPOC = 35 y CAMPOB = 475.70, valor éste último adquirido durante la ejecución de la declaración anterior; por tanto: 100 + 12.5 + 125.7 + 35 + 475.70 = 748.90, valor que se introduciría en CAMPOB, permaneciendo invariables CAMPOA y CAMPOC.

Si a continuación de la palabra TO se especifica más de un identificador se suman los valores de todos los campos y literales que preceden a TO, y el resultado se añade al valor que contenga cada uno de los identificadores que siguen a TO.

Por tanto, la ejecución de la declaración de la línea 001205 se haría de la forma siguiente:

– Efectúa la operación CAMPOA + CAMPOK + (–9) = 12.5 + 4 + (–9) = 7.5 y

– Suma este resultado a CAMPOB y a CAMPOX. Por tanto, si antes de ejecutar dicha declaración, los valores de estos campos son: CAMPOB = 748.90 y CAMPOX = 89.5, el valor resultante será: CAMPOB = 748.90 + 7.5 = 758.40 , y
  CAMPOX = 89.5 + 7.5 = 97.0

Opción GIVING.

Si se emplea la opción GIVING identificador–m, dicho identificador puede ser un campo de tipo numérico o de informe , e interviene en la declaración no como sumando, sino como receptor del resultado.

```
001210     ADD  CAMPOA, 250. CAMPOC  GIVING  CAMPOD
.....      .....................................
002010     ADD  CAMPOA, 250, CAMPOC  GIVING  CAMPOE
```

La declaración de la línea 001210 se ejecuta de la forma siguiente:

– Suma el valor de CAMPOA, el literal 250 y el valor de CAMPOC, es decir:

$$12.5 + 250 + 35 = 297.5.$$

– Transfiere el valor 297.5 al CAMPOD, con lo que el contenido de dicho campo pasará a ser:

0	0	2	9	7	5	0

Hay que hacer notar que si el número de dígitos significativos que se transfieren a CAMPOD es inferior al número de posiciones reservadas en memoria para dicho campo, se completan dichas posiciones con ceros de orden superior e inferior, por ser CAMPOD un campo numérico.

Sin embargo,en la declaración de la línea 002010,al ser CAMPOE un campo de informe el contenido de dicho campo después de efectuar la suma sería:

		2	9	7	.	5	0

TRUNCAMIENTO.

Si el número de dígitos del resultado es superior al número de posiciones reservadas para el campo receptor del mismo, habrá truncamiento en uno o en ambos extremos del campo:

```
.....................................
002020     ADD  CAMPOA, 1400.28  GIVING  CAMPOH
```

El resultado de ejecutar esta declaración sería: 12.5 + 1400.28 = 1412.78; sin embargo, como CAMPOH sólo es capaz de albergar tres cifras enteras y una decimal, el contenido de dicho campo pasaría a ser:

4	1	2	7

El truncamiento en uno o en ambos extremos no lleva consigo que el ordenador detenga la ejecución del programa, sino que el resultado final de una operación aritmética, como se ha visto en el ejemplo anterior, puede ser imprevisible.

El truncamiento, si se desea, se evita reservando para el campo receptor del resultado un número de posiciones de la memoria igual o mayor al número que se prevea va a ser las cifras del resultado.

Opción ON SIZE ERROR.

Existe la posibilidad de poder obligar al ordenador a realizar una determinada acción, incluso detener la ejecución del programa, en el caso de que el número de cifras enteras del resultado de la operación aritmética sea mayor que el número de posiciones enteras reservadas para el campo receptor del mismo. La opción ON SIZE ERROR es la que permite dicha posibilidad.

```
002030     ADD CAMPOA, 1400.28 GIVING CAMPOH ON SIZE ERROR
002040                                        GO TO C.
002050  B. ADD CAMPOA, 200.74, 5.125 GIVING CAMPOT ROUNDED
```

Con la declaración de la línea 002030 se pretende, en el caso de que el número de cifras enteras del resultado de la operación exceda de las que puede admitir el campo receptor CAMPOH, ejecutar una bifurcación a la declaración de la línea 002120.

Las operaciones que tienen lugar, como consecuencia de la ejecución de la declaración de la línea 002030, son las siguientes:

- Suma las cantidades 12.5 + 1400.28 = 1412.78

- Comprueba si las cifras enteras del resultado caben dentro del campo receptor, CAMPOH, en este caso. Como no caben, no se mueve al campo receptor del resultado el valor: 1412.78, sino que la opción ON SIZE ERROR obliga a ejecutar la declaración que sigue a dicha opción, es decir, bifurca al procedimiento C.

- Por el contrario, si el número de cifras enteras del resultado, cabe dentro del campo receptor, coloca dicho resultado en el citado campo, omitiendo la declaración o declaraciones que siguen a ON SIZE ERROR, y continuando la ejecución del programa en la siguiente declaración.

En las operaciones con campos numéricos que contienen cifras decimales, pueden darse resultados en los que el número de dichas cifras sea excesivo y el programador desee eliminarlas total o parcialmente; en cualquier caso, el programador resuelve el problema al hacer la descripción del campo receptor del resultado, especificando en la PICTURE el número de cifras decimales que desee contenga dicho campo.

Opción ROUNDED.

Existe también la posibilidad –por medio de la opción ROUNDED– de, además de truncamiento de parte o todas las cifras decimales del resultado, poder incrementar en una unidad el dígito decimal de orden inferior que ha de figurar en el resultado de la suma, siempre que el dígito mas significativo de la parte que se ha de despreciar sea igual o mayor que cinco.

EJEMPLO:

```
002050  B. ADD CAMPOA, 200.74, 5.125 GIVING CAMPOT ROUNDED.
```

Efectuada la declaración de la línea 002050, el resultado de la suma sería:

$$12.5 + 200.74 + 5.125 = 218.365$$

sin embargo, en el campo receptor se almacenaría: | 2 | 1 | 8 | 4 | , por ser mayor de cinco la cifra de las centésimas , a partir de la cual hay que despreciar.

FORMATO 3:

ADD { **CORRESPONDING** / **CORR** } **identificador–1** **TO** **identificador–2** [**ROUNDED**]

[**ON SIZE ERROR** **declaración–imperativa**]

Identificador–1 e identificador–2 deben ser campos compuestos.

La opción CORRESPONDING hace que se sumen los campos de datos elementales de identificador–1 con los campos de datos elementales del mismo nombre de identificador–2.

Sean los campos ALFA y BETA cuya descripción es la siguiente:

```
010 .....................................
020  01  ALFA.
030      02  UNO      PIC 99      VALUE 25.
040      02  DOS      PIC 999V99  VALUE 175.50.
050      02  TRES     PIC 999     VALUE 450.
060  01  BETA.
070      02  SIGMA    PIC 999V9   VALUE 111.5.
080      02  TRES     PIC 9999V9  VALUE 1800.
090      02  GAMMA    PIC 9       VALUE 1.
100      02  DOS      PIC 9999V99 VALUE 1285.5
110
```

La declaración ADD CORRESPONDING ALFA TO BETA, haría que los campos elementales TRES y DOS del campo compuesto BETA pasaran a contener 2250 = (450 + 1800) y 1461.0 = (1285.5 + 175.50) respectivamente, no sufriendo modificación los campos TRES y DOS del campo compuesto ALFA, ni los campos GAMMA, SIGMA y UNO, por no ser comunes en los dos campos compuestos ALFA y BETA.

3.3.- Declaración SUBTRACT

FUNCION: Restar dos campos de datos, o sumar varios campos de datos y el resultado restarlo de otro campo, sustituyendo el valor de la diferencia al valor del último campo especificado en la declaración.

FORMATO 1:

```
SUBTRACT { identificador-1 } [ identificador-2 ] ... FROM identificador-m [ ROUNDED ]
         { literal-1       } [ literal-2       ]

[ identificador-n [ ROUNDED ] ] ... [ ON SIZE ERROR declaración-imperativa ]
```

FORMATO 2:

```
SUBTRACT { identificador-1 } [ identificador-2 ] ... FROM { identificador-m }
         { literal-1       } [ literal-2       ]        { literal-m       }

GIVING identificador-n [ ROUNDED ] [ ON SIZE ERROR declaración-imperativa ]
```

El minuendo, identificador-m, se debe especificar a continuación de la partícula FROM y el sustraendo, identificador-1, a continuación de SUBTRACT:

```
002060     SUBTRACT  CAMPOK  FROM  CAMPOC
002070     SUBTRACT  5, CAMPOK, 2.1  FROM  CAMPOA
```

La declaración de la línea 002060 equivale a la expresión:

CAMPOC-CAMPOK

El valor de la diferencia sustituye al valor del minuendo, permaneciendo con su valor inicial el sustraendo. Por tanto, el valor del CAMPOC después de ejecutada dicha declaración será 31, mientras que CAMPOK permanece invariable.

Cuando son varios los campos de datos y/o literales numéricos que se especifiquen a continuación de SUBTRACT, se suman todos ellos, y el resultado se resta del minuendo, es decir, la declaración de la línea 002070 equivale a:

CAMPOA - (5 + CAMPOK + 2.1)

Si se especifica la opción GIVING, el valor de la diferencia sustituye al valor del campo de datos que sigue a GIVING, permaneciendo con sus valores iniciales minuendo y sustraendo.

Utilizando esta opción, tanto el minuendo como el sustraendo pueden ser literales numéricos.

FORMATO 3 :

```
SUBTRACT { CORRESPONDING } identificador-1 FROM identificador-2 [ROUNDED]
         { CORR          }

[ON SIZE ERROR declaración-imperativa]
```

Identificador–1 e identificador–2 deben ser campos compuestos.

Los campos de datos elementales que dependen de identificador–1 se restan de los campos de datos elementales de igual nombre que dependan de identificador–2, almacenando la diferencia en estos últimos.

Utilizando la descripción de los campos ALFA y BETA hecha en la página 161, si se ejecutara la declaración: SUBTRACT CORRESPONDING ALFA FROM BETA, el contenido de los campos TRES y DOS del campo BETA, pasaría a valer: 1350.0 y 1110.00 respectivamente.

3.4.- Declaración MULTIPLY

FUNCION: Multiplicar dos campos de datos y sustituir el valor del último campo especificado en la declaración, por el valor del producto.

FORMATO 1:

```
MULTIPLY { identificador-1 } BY identificador-2 [ROUNDED]
         { literal-1       }

[ON SIZE ERROR declaración-imperativa]
```

FORMATO 2:

```
MULTIPLY { identificador-1 } BY { identificador-2 } GIVING identificador-3 [ROUNDED]
         { literal-1       }    { literal-2       }

[ON SIZE ERROR declaración-imperativa]
```

El multiplicando, identificador–1, se debe especificar inmediatamente después de MULTIPLY y el multiplicador, identificador–2, después de BY.

Si no se especifica en la declaración GIVING (formato 1), el valor del producto se sitúa dentro del multiplicador permaneciendo invariable el valor del multiplicando. Este formato permite que únicamente el multiplicando puede ser un literal numérico:

```
002080     MULTIPLY 20 BY CAMPOI
```

En la declaración de la línea 002080 al ser el valor inicial de CAMPOI = 57, ejecuta la

operación 20 x 57 = 1140, colocando el resultado en CAMPOI. En este caso, por no especificarse la opción GIVING, el multiplicador no puede ser un literal numérico, ya que éste no reserva posiciones de memoria y por lo tanto el resultado de la operación no puede sustituir al valor constante que contiene este segundo operando.

Si se especifica la opción GIVING (Formato 2), el valor del producto de los dos operandos se transfiere al campo de datos que sigue a GIVING, permaneciendo invariables el contenido del multiplicando y el multiplicador.

El tamaño máximo del producto resultante no puede ser superior a 18 dígitos.

Con este formato, tanto el multiplicando como multiplicador pueden ser literales numéricos.

3.5.- Declaración DIVIDE

FUNCION: Dividir un campo de datos entre otro, sustituyendo el valor del cociente al valor del último campo de datos especificado en la declaración.

FORMATO 1:

```
DIVIDE { identificador-1 } INTO identificador-2 [ ROUNDED ]
       { literal-1       }
[ ON SIZE ERROR declaración-imperativa ]
```

Caso de que el valor contenido en indentificador-1 sea cero, se cumple la condición ON SIZE ERROR, lo cual lleva consigo que no se efectúe la división, continuando la ejecución del programa en la declaración imperativa que sigue a SIZE ERROR.

El dividendo debe especificarse a continuación de INTO y el divisor a continuación de DIVIDE:

```
002090     DIVIDE CAMPOK INTO CAMPOL
```

La declaración de la línea 002090 equivale a la expresión: CAMPOL: CAMPOK.

El cociente sustituye al valor del dividendo permaneciendo invariable el valor del divisor.

El divisor puede ser un campo de datos o un literal numérico.

El resultado de la operación anterior sería:

126.4 : 4 = 31.6

valor que se introduciría en CAMPOL, sustituyendo a 126.4 que era su valor inicial.

El siguiente formato utiliza la opción GIVING, la cual permite que tanto el dividendo como el divisor pueden ser literales numéricos, o campos de datos, o bien una mezcla de un literal el dividendo y un campo de datos el divisor y a la inversa.

FORMATO 2:

```
DIVIDE { identificador-1 } { INTO } { identificador-2 } GIVING identificador-3 [ROUNDED]
       { literal-1       } { BY   } { literal-2       }
  [REMAINDER identificador-4] [ON SIZE ERROR declaración-imperativa]
```

Utilizando la opción INTO, divide el valor de identificador–2 ó el literal–2 por el valor de identificador–1 ó el literal–1 almacenando el cociente en identificador–3 y el resto en identificador–4.

Si se emplea la opción BY, divide identificador–1 ó literal–1 por el valor de identificador–2 ó literal–2. En este caso, el dividendo se debe especificar a continuación de la palabra DIVIDE y el divisor a continuación de BY.

El tamaño máximo del cociente y del resto es de 18 dígitos.

La división por cero produce una condición ON SIZE ERROR.

3.6.- Declaración COMPUTE

FUNCION: Transferir a un campo de datos el valor de otros campos de datos numéricos, de un literal, o el resultado de una expresión aritmética.

FORMATO:

```
COMPUTE identificador-1 [ROUNDED] = { identificador-2          }
                                    { literal-numérico         }
                                    { expresión-aritmética     }
  [ON SIZE ERROR declaración-imperativa]
```

El tipo de PICTURE para identificador–1 puede ser de edición o numérico; sin embargo, identificador–2, caso de utilizarse, debe ser un campo de tipo numérico.

```
002100        COMPUTE  CAMPOP = 125
002110        COMPUTE  CAMPOE = CAMPOT
```

La declaración de la línea 002100 equivale a una declaración MOVE, ya que la acción que ejecuta es transferir el valor 125 a CAMPOP.

En la declaración de la línea 002110, CAMPOE es un campo de edición, al que se transfiere el contenido de CAMPOT, siendo éste, en el momento de ejecutar dicha declaración, 218.4, valor que se obtuvo en la ejecución de la declaración de la línea 002050.

Es fácil comprender que estos dos ejemplos no constituyen el empleo más adecuado de la declaración COMPUTE, ya que es una declaración que requiere más tiempo para su ejecución que aquélla por la que puede ser sustituída como es la MOVE. Por tanto, su empleo adecuado y para el que está concebida es para el cálculo de expresiones aritméticas en las que intervengan operaciones aritméticas elementales.

3.7.- Expresión ARITMETICA

Una expresión aritmética está compuesta por una serie de operadores aritméticos, que indican el conjunto de operaciones a efectuar con los campos de datos y/o literales numéricos que forman parte de dicha expresión.

Para codificar una expresión aritmética o fórmula matemática se emplearán los operadores aritméticos : + – * / y ** , siendo opcional el empleo de paréntesis.

EJEMPLO. Se quiere codificar la expresión:

$$\text{GAMMA} = 4 + \text{ALFA}^2 - \frac{15}{\text{BETA}}$$

Para su codificación hay que tener en cuenta que:

– Cada operador aritmético debe ir precedido y seguido de un espacio en blanco.

– Según el tipo de operación a realizar habrá que utilizar el operador adecuado.

Por tanto dicha expresión quedará codificada de la forma siguiente:

```
080        COMPUTE GAMMA = 4 + ALFA ** 2 - 15 / BETA.
090
```

Es muy importante la utilización o supresión de paréntesis en una expresión aritmética, ya que éstos al poder alterar el orden de ejecución de las operaciones, hacen que los resultados puedan ser erróneos .

— En una expresión aritmética que no se especifican paréntesis, el ordenador ejecuta las operaciones comenzando de izquierda a derecha en el siguiente orden:

- Signo + y — que afecte a toda la expresión aritmética.
- Potenciación * *
- Multiplicación y División * y /
- Suma y resta + y —

La expresión aritmética de la página anterior se ha codificado prescindiendo de los paréntesis, en este caso, suponiendo que ALFA y BETA en esta expresión contengan los valores iniciales 12 y 3 respectivamente, las operaciones se ejecutarán como sigue:

- Eleva al cuadrado el campo ALFA: 12 * 12 = 144.
- Divide 15 por el valor de BETA: 15 / 3 = 5 .
- Suma el literal numérico 4 al valor de ALFA elevado al cuadrado: 4 + 144 = 148 .
- Resta de este último valor el obtenido en la segunda operación: 148 — 5 = 143 .
- Guarda en GAMMA el resultado calculado : 143 .

— Si se especifican paréntesis en una expresión aritmética, dentro de cada uno de ellos se ejecutan en el orden citado anteriormente todas las operaciones que se hayan especificado.

Si la expresión aritmética de la página anterior la hubiera codificado el programador de la forma siguiente:

```
110    COMPUTE GAMMA = (4 + ALFA ** 2 - 15) / BETA.
```

En este caso se ejecutan las operaciones en el siguiente orden:

- Eleva ALFA al cuadrado, le suma 4 unidades y le resta 15: 144 + 4 — 15 = 133 .
- El resultado anterior se divide por el valor de BETA: 133 / 3 = 44.3

Es decir, se ha obtenido un resultado diferente al del primer caso, que era el correcto, siendo incorrecto este último por haber especificado el programador mal los paréntesis.

Lo correcto hubiera sido especificar los paréntesis de la forma siguiente:

```
110    COMPUTE GAMMA = (4 + ALFA ** 2) - (15 / BETA).
```

Existen expresiones aritméticas cuya complejidad exige que se especifiquen paréntesis que encierren a otros paréntesis, en este caso, las operaciones se ejecutan comenzando por las incluidas en los paréntesis más internos, hasta llegar a los más externos.

Así, en la expresión aritmética:

$$\text{GAMMA} = \frac{(25 - \text{ALFA}^2)^3 + 7}{5} - \text{BETA}$$

el programador ha hecho su codificación de la forma siguiente:

```
050     COMPUTE GAMMA = ((((25 - ALFA ** 2) ** 3 + 7) / 5) - BETA.
```

es decir, ha especificado correctamente una serie de paréntesis que, aunque todos no son necesarios, facilitan la labor del programador, ya que aseguran el orden correcto en que han de realizarse las operaciones.

Así, el orden de ejecución de las operaciones en la expresión precedente sería:

- ALFA^2
- $(25 - \text{ALFA}^2)$
- $(25 - \text{ALFA}^2)^3$
- $(25 - \text{ALFA}^2)^3 + 7$
- $((25 - \text{ALFA}^2)^3 + 7) / 5$
- $(((25 - \text{ALFA}^2)^3 + 7) / 5) - \text{BETA}$

Aunque no existe en COBOL un símbolo indicador de la raíz, sin embargo, el ordenador puede calcular la raíz de cualquier índice (cuadrada, cúbica, etc.) de un número, transformándola en una potencia de exponente fraccionario.

EJEMPLO. La expresión:

$$\text{GAMMA} = \sqrt[4]{\text{ALFA}} - (\sqrt[7]{\text{BETA}})^3 = \text{ALFA}^{1/4} - \text{BETA}^{3/7}$$

se puede codificar de la forma siguiente:

```
    050     COMPUTE GAMMA = ALFA ** (1 / 4) - BETA ** (3 / 7).
```

o también:

```
    050     COMPUTE GAMMA = AL. A .** .25 - BETA ** (3 / 7).
```

4.-VERBOS de MOVIMIENTO

4.1.- Declaración MOVE

FUNCION: Transferir datos de un área a otra u otras de la memoria.

FORMATO 1:

$$\underline{\text{MOVE}} \left\{ \begin{array}{l} \text{Identificador} - 1 \\ \text{Literal} \end{array} \right\} \underline{\text{TO}} \quad \text{Identificador} - 2 \quad [\text{Identificador} - 3] \cdots$$

– Los datos contenidos en Identificador–1 ó en el literal se mueven al área de la memoria reservada por Identificador–2.

Una vez efectuado el movimiento, Identificador–1 conserva su contenido, mientras el contenido de Identificador–2 es sustituído por el de Identificador–1, es decir, las áreas de la memoria denominadas por Identificador–1 e Identificador–2, después del movimiento contendrán los mismos datos.

EJEMPLO:

```
008010 ALFA-1.
008020     MOVE CODIGO TO CODIGO-S
008030     MOVE APELL-NOMBRE TO APE-NOMBRE
008040     MOVE SUELDO TO SUELDO-S
```

Si el contenido del campo CODIGO en el momento de ejecutar la declaración MOVE es: 00001 y el de CODIGO–S : XA15B, una vez ejecutada dicha declaración, CODIGO y CODIGO–S contendrán el valor 00001.

Si a continuación de la partícula TO se describe más de un Identificador, se mueve a todos ellos el contenido de Identificador–1 (línea 008090):

```
008080     MOVE DESCU TO DESCUE
008090     MOVE TOTALS TO TOTAL, TOTAL-SUELDOCIN
008100     MOVE SPACES TO IMPRESO. NOTE ESTA DECLARACION LIMPIA
```

El tipo de literal que se ha de mover a Identificador–2... puede ser numérico, no numérico o constante figurativa.

EJEMPLO:

```
007140     IF VIUDO     COMPUTE PLUSF = NUM-HIJOS * 250
007150     ELSE MOVE 0 TO PLUSF.
```

```
008100     MOVE SPACES TO IMPRESO. NOTE ESTA DECLARACION LIMPIA
008110     EL AREA DE SALIDA (PONE BLANCOS).
```

```
008120     IF CONTALINEAS = 24 WRITE IMPRESO AFTER 0
008130     PERFORM ESCRIBIR-2 MOVE ZERO TO CONTALINEAS.
```

Para facilitar posteriores explicaciones se llamará campo origen a Identificador–1 ó al literal, y campo receptor a Identificador–2, Identificador–3, etc.

TIPOS DE MOVIMIENTOS.

La transferencia de datos de un campo a otro u otros campos de la memoria puede realizarse utilizando dos tipos de movimiento:

- **NUMERICO.**
- **NO NUMERICO.**

Movimiento de Tipo NUMERICO:

Está permitido cuando el campo origen es un campo con PICTURE de tipo numérico,un literal numérico o la constante figurativa ZERO, y el campo receptor tiene su PICTURE de tipo numérico o de informe (ver cuadro de movimientos permitidos pág. 174).

Este tipo de movimiento se realiza de la forma siguiente:

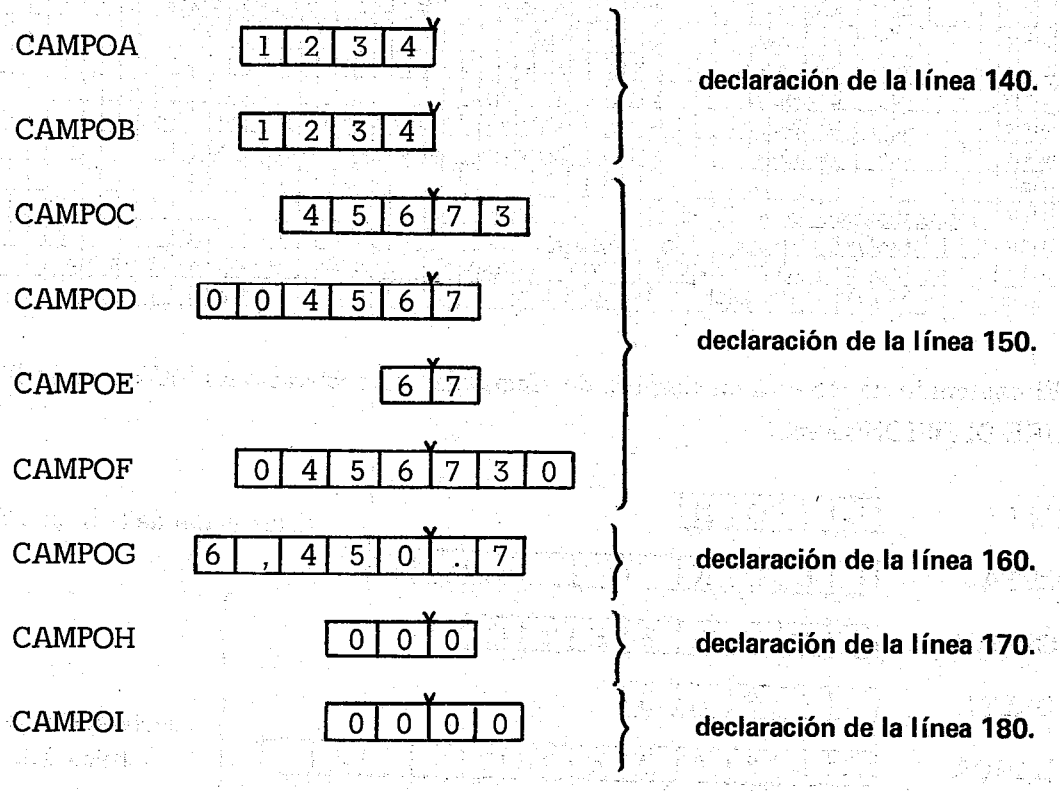
- Alinea los campos origen y receptor por el punto decimal, y a derecha e izquierda del mismo van colocándose en el campo receptor los dígitos contenidos en el campo origen, pudiendo haber tuncamiento en uno o en ambos extremos si el número de dígitos del campo origen es mayor que la capacidad del receptor; o relleno con ceros en uno o ambos extremos, cuando el número de dígitos del campo origen es menor que la capacidad del receptor.

- Si el campo receptor tiene una PICTURE de tipo de edición, además del movimiento de los datos, se transforman éstos de acuerdo con las reglas de EDICION según el tipo de caracteres de edición que contenga dicha PICTURE.

El siguiente ejemplo, es parte de un programa en el que a algunos de los campos se les ha asignado un valor incial y se quiere conocer cuál será el valor de los mismos una vez ejecutadas las declaraciones MOVE:

```
010 WORKING-STORAGE SECTION.
020  77  CAMPOA      PIC  9999          VALUE 1234.
030  77  CAMPOB      PIC  9(4)          VALUE 3987.
040  77  CAMPOC      PIC  999V99        VALUE 456.73.
050  77  CAMPOD      PIC  9(5)V9        VALUE 83945.1.
060  01  CAMPOS.
070      02  CAMPOE  PIC  9V9           VALUE 3.5.
080      02  CAMPOF  PIC  9999V999      VALUE 1345.782.
090      02  CAMPOG  PIC  9,999.9.
100      02  CAMPOH  PIC  99V9.
110      02  CAMPOI  PIC  99V99         VALUE 20.45.
120
130 PROCEDURE DIVISION.
140     MOVE  CAMPOA   TO  CAMPOB
150     MOVE  CAMPOC   TO  CAMPOD  CAMPOE  CAMPOF
160     MOVE  6450.7   TO  CAMPOG
170     MOVE  0        TO  CAMPOH
180     MOVE  ZERO     TO  CAMPOI
190
```

El contenido de cada uno de los campos, después de ejecutadas las declaraciones de la PROCEDURE serán:



Movimiento de tipo NO NUMERICO.

Tiene lugar cuando la PICTURE de los campos origen y receptor no es ninguna de las combinaciones citadas en los movimientos numéricos (ver cuadro de movimientos permitidos, pág. 174).

Este tipo de movimientos se realiza de la forma siguiente:

- Durante la transferencia, los caracteres contenidos en el campo origen se van colocando en el campo receptor de izquierda a derecha, excepto cuando el campo receptor lleva la cláusula JUSTIFIED RIGHT, en cuyo caso los caracteres se colocan de derecha a izquierda.

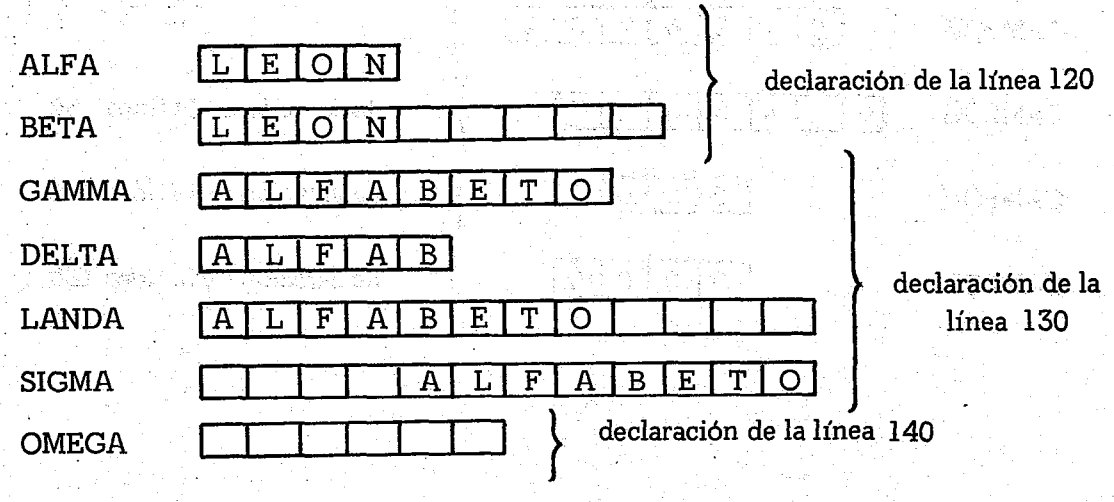
- Si el número de caracteres del campo origen es mayor que el número de caracteres que puede admitir el campo receptor, el movimiento se termina cuando se ha llenado el campo receptor.

- Si el campo origen es menor que el receptor, las posiciones que le falte por llenar se rellenan con blancos (espacios).

EJEMPLO:

```
010  WORKING-STORAGE SECTION.
020    01  REGIS.
030        02   ALFA        PIC   A(4)      VALUE 'LEON'.
040        02   BETA        PIC   A(9)      VALUE 'SALAMANCA'.
050        02   GAMMA       PIC   X(8).
060        02   DELTA       PIC   X(5).
070        02   LANDA       PIC   X(12).
080        02   SIGMA       PIC   X(12)     JUSTIFIED RIGHT.
090        02   OMEGA       PIC   A(6)      VALUE 'REGION'.
100
110  PROCEDURE DIVISION.
120        MOVE  ALFA  TO  BETA
130        MOVE 'ALFABETO' TO GAMMA LANDA SIGMA DELTA
140        MOVE  SPACES  TO OMEGA
```

El contenido de los campos después de efectuadas las declaraciones MOVE de la PROCEDURE DIVISION serán:



FORMATO 2:

MOVE { **CORRESPONDING** / **CORR** } **Identificador–1** **TO** **Identificador–2**

Utilizando este formato, una declaración MOVE puede transferir el contenido de un número variable de campos elementales, componentes del campo compuesto Identificador–1, a los campos elementales que se denominen de la misma forma (correspondientes) incluidos en el campo compuesto Identificador–2, es decir, una sola declaración MOVE CORR hace el mismo efecto que si se hubiera especificado para cada par de nombres–de–datos (campos) correspondientes, una declaración MOVE simple.

EJEMPLO. Con el siguiente programa se obtiene un listado, en el que en cada línea del mismo figuran: apellidos y nombre, sueldo bruto y número de hijos de cada operario, especificados en este orden. Se utiliza como archivo de entrada de datos el ARCHIVO–OPERARIO, descrito en el programa NOMINA (pág. 23).

```
001010  ID DIVISION.
001020   PROGRAM-ID. MOVECORR.
001030  ENVIRONMENT DIVISION.
001040    INPUT-OUTPUT SECTION.
001050     FILE-CONTROL.
001060          SELECT ARCHIVO-OPERARIOS ASSIGN SYS005-UR-2540R-S.
001070          SELECT ARCHI-IMPRESO     ASSIGN SYS006-UR-1403-S.
001080  DATA DIVISION.
001090   FILE SECTION.
001100     FD ARCHIVO-OPERARIOS LABEL RECORD OMITTED.
001110     01 FICHA-OPERARIO.
001120          02 CODIGO      PIC X(5).
001130          02 APENOM      PIC X(35).
001140          02 FILLER      PIC X(10).
001150          02 NUM-HIJOS   PIC 99.
001160          02 FILLER      PIC 9(6).
001170          02 SUELDO      PIC 9(5).
001180          02 FILLER      PIC X(17).
001190     FD ARCHI-IMPRESO  LABEL RECORD  OMITTED.
001200     01 IMPRESO.
001210          03 FILLER      PIC X(7).
001220          03 APENOM      PIC X(35).
001230          03 FILLER      PIC X(7).
001240          03 SUELDO      PIC 9(5).
001250          03 FILLER      PIC X(7).
002010          03 NUM-HIJOS   PIC 99.
002020  PROCEDURE DIVISION.
002030    ABRIR. OPEN INPUT ARCHIVO-OPERARIOS OUTPUT ARCHI-IMPRESO
002040           DISPLAY '                    RELACION DE OPERARIOS
002050-                  'SUELDO Y NUM-HIJOS'.
002060           MOVE SPACES TO IMPRESO.
002070    LEER. READ ARCHIVO-OPERARIOS AT END GO TO FIN.
002080           MOVE CORRESPONDING FICHA-OPERARIO TO IMPRESO
002090           WRITE IMPRESO  AFTER 2 LINES  GO TO LEER.
002100    FIN.  CLOSE ARCHIVO-OPERARIOS ARCHI-IMPRESO  STOP RUN.
```

La declaración MOVE CORRESPONDING (línea 002080) transfiere el contenido del campo APENOM (línea 001130) al campo APENOM (línea 001220); el del campo SUELDO (línea 001170) al campo SUELDO (línea 001240) y, por último, el contenido de NUM–HIJOS (línea 001150) al campo NUM–HIJOS (línea 002010). En los campos FILLER componentes de IMPRESO no se lleva a cabo transferencia alguna, aunque se denominen de la misma forma que los de FICHA–OPERARIO, por ser aquél un nombre standard.

A cada pareja de campos correspondientes son aplicables las reglas que se describieron para la declaración MOVE simple.

Identificador–1 e Identificador–2 ... no pueden llevar un nivel 66, 77 u 88.

Los campos elementales que contengan una cláusula OCCURS, REDEFINES, USAGE INDEX o RENAMES se ignoran.

CUADRO DE MOVIMIENTOS PERMITIDOS (20)

CAMPO RECEPTOR / CAMPO ORIGEN	COMPUESTO	ALFABETICO	ALFA-NUMERICO	NUMERICO	COMA FLOTANTE	EDICION O INFORME
COMPUESTO	NN	NN	NN			
ALFABETICO	NN	NN	NN			
ALFANUMERICO	NN	NN	NN			
NUMERICO	NN		NN (2)	NU	NU	NU
COMA FLOTANTE	NN			NU	NU	NU
EDICION O INFORME	NN		NN			
ZEROS	NN		NN	NU	NU	NU
SPACES	NN	NN	NN			
RESTO DE CONSTANTES FIGURATIVAS	NN		NN			

(2) Sólo permitido para números enteros.
NN Movimiento de tipo no numérico.
NU Movimiento de tipo numérico.
Las casillas en blanco indican que el movimiento no está permitido.

– Los literales numéricos se consideran como campos numéricos.

– Los literales no numéricos se consideran como campos alfanuméricos.

El empleo de la tabla consiste en la doble entrada, tomando el campo origen y siguiendo la línea hasta encontrar la columna correspondiente al campo receptor; la intersección de la línea y columna indica si está o no permitida la transferencia.

(20) **El cuadro expuesto es el más generalizado para cualquier tipo de ordenador, sin embargo, es necesario el estudio del manual correspondiente al ordenador que se utilice, ya que existen otros movimientos que están permitidos y no se citan en el cuadro.**

4.2.- Declaración EXAMINE

FUNCION: Contar el número de veces que un carácter COBOL aparece en un campo de datos, a la vez que poder sustituirlo por otro carácter.

FORMATO 1:

EXAMINE identificador TALLYING { ALL / LEADING / UNTIL FIRST } carácter-1 [REPLACING BY carácter-2]

FORMATO 2:

EXAMINE identificador REPLACING { ALL / LEADING / UNTIL FIRST / FIRST } carácter-1 BY carácter-2

- Identificador en los dos formatos debe ser un campo de datos descrito con USAGE DISPLAY.

- Carácter-1 y carácter-2 pueden ser literales numéricos o no numéricos, según que la PICTURE del identificador sea de tipo numérico o no numérico.

- Las constantes figurativas,excepto ALL,pueden emplearse en vez de carácter-1 y/o carácter-2.

- Si se emplea la opción TALLYING (CONTAR), el ordenador lleva una cuenta del número de veces que el carácter-1 aparece en el campo denominado por identificador. El resultado de esta cuenta queda colocado dentro de un registro especial del Sistema capaz de almacenar números de cinco cifras denominado: TALLY. No es necesario definir TALLY en la DATA DIVISION, ya que este contador lo establece el compilador, e incluso se puede utilizar como nombre-de-datos en una declaración:

```
ADD 1 TO TALLY; MOVE ZEROS TO TALLY
```

- El examen de un campo de datos para llevar a cabo la cuenta, siempre comienza de izquierda a derecha.

- Cuando se emplea la opción ALL, se cuentan las veces que se repite carácter-1 dentro del campo de datos.
  Si en la declaración:

```
EXAMINE ALFA TALLYING ALL 'A'
```

se supone que el contenido de ALFA y TALLY antes de ejecutar la declaración EXAMINE son:

ALFA	I	N	F	O	R	M	A	T	I	C	A

TALLY	0	0	0	3	8

Una vez ejecutada la declaración, el contenido de TALLY será: | 0 | 0 | 0 | 0 | 2 |, ya que existen dos A,s en la palabra INFORMATICA.

— Cuando se emplea la opciónLEADING, el registro TALLY pasará a contener el número de veces que aparece el carácter—1 antes de encontrar algún carácter distinto de él:

EXAMINE BETA TALLYING LEADING 7

BETA | 7 | 7 | 7 | 7 | 3 | 2 | 5 | 7 | TALLY | 0 | 0 | 0 | 0 | 2 | (antes) TALLY | 0 | 0 | 0 | 0 | 4 | (después)

ya que es cuatro el número de veces que la cifra 7 aparece antes de la cifra distinta de ella.

— Cuando se emplea la opción UNTIL FIRST, el registro TALLY contendrá el número de caracteres diferentes del carácter—1 hasta la aparición del primer carácter igual a carácter—1:

EXAMINE ALFA TALLYING UNTIL FIRST 'T'

ALFA | I | N | F | O | R | M | A | T | I | C | A | TALLY | 0 | 0 | 0 | 0 | 4 | (antes) TALLY | 0 | 0 | 0 | 0 | 7 | (después)

ya que existen antes de la T siete caracteres diferentes a ella.

— Dentro de este FORMATO 1 se puede utilizar la opción REPLACING (reemplazar, substituir), la cual al realizar las mismas acciones que si se utilizara dentro del FORMATO 2,todo cuanto se indica a continuación es válido para los dos formatos:

— Cuando se emplea la opción ALL, cada carácter—1 queda sustituído por carácter—2 cada vez que aparezca en el campo de datos:

EXAMINE ZETA TALLYING ALL '—' REPLACING BY 'O'

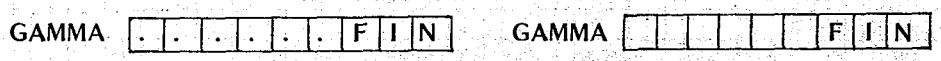
ZETA | C | — | B | — | L | (antes) TALLY | 0 | 0 | 0 | 0 | 7 | (antes) ZETA | C | O | B | O | L | (después) TALLY | 0 | 0 | 0 | 0 | 2 | (después)

EXAMINE ZETA REPLACING ALL '—' BY 'O'

ZETA | C | — | B | — | L | (antes) ZETA | C | O | B | O | L | (después) TALLY no se modifica

- Cuando se emplea la opción LEADING, la sustitución por carácter−2 termina cuando aparece un carácter diferente de carácter−1, o bien cuando se alcanza el extremo derecho del campo.

EXAMINE GAMMA REPLACING LEADING '.' BY SPACE



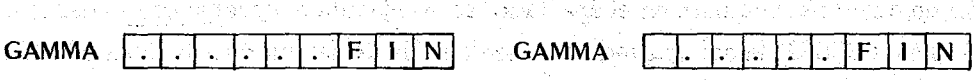
- Cuando se emplea UNTIL FIRST, la sustitución por carácter−2 termina cuando aparece un carácter−1, o cuando se alcanza el extremo derecho del campo:

EXAMINE GAMMA REPLACING UNTIL FIRST 'F' BY '.'



- Cuando se emplea la opción FIRST, se sustituye por el carácter−2 el primer carácter−1 que aparezca:

EXAMINE GAMMA REPLACING FIRST '.' BY SPACE



Para completar el estudio de la declaración EXAMINE, se muestra una aplicación práctica referida al programa NOMINA.

EJEMPLO.

El Jefe de Contabilidad de la Empresa a la vista de los resultados de la Nómina, desea introducir las modificaciones oportunas dentro del programa, a fin de que:

a) Los ceros no significativos que se imprimen para designar el código de cada operario, sean sustituidos por espacios (blancos).

b) El asterisco que aparece delante de cada sueldo neto sea reemplazado por el carácter $.

c) El encabezamiento NOMINA DE LA EMPRESA RICOSA . . . lleve doble subrayado en vez de sencillo.

d) Utilice el registro TALLY en vez del campo CONTADOR, para contabilizar el número de operarios que integran la NOMINA.

El equipo de programación,utilizando la declaración EXAMINE y el registro TALLY, introduce las siguientes modificaciones en el programa NOMINA :

```
000610  PROCEDURE DIVISION.
000690      ESCRIBIR-1.
000700          WRITE IMPRESO FROM CABECERA-1 AFTER 0
000710          EXAMINE TITULO-1 REPLACING UNTIL FIRST '/' BY '='
000720          WRITE IMPRESO FROM CABECERA-1 AFTER 1.
000730      ESCRIBIR-2.

000710      LEER.
000720          READ ARCHIVO-OPERARIOS AT END
000730          DISPLAY ' REGISTROS PROCESADOS = ' TALLY.

000801      ALFA-1.
000802          MOVE CODIGO TO CODIGO-S
000802          EXAMINE CODIGO-S REPLACING LEADING ZEROS BY SPACES

000891          EXAMINE TOTAL REPLACING FIRST '*' BY '$'

000817          ADD 1 TO TALLY
000818          GO TO LEER.
```

La ejecución de cuanto se requiere en el apartado a) lo ejecuta el ordenador mediante la declaración de la línea 008021; la del apartado b) mediante la declaración de la línea 008091; la del apartado c) mediante la declaración de la línea 006110; y la del apartado d) mediante la declaración de la línea 008170.

Como se puede ver,la solución dada no es la idónea, pero cumple el objetivo de practicar con la declaración EXAMINE.

La declaración de movimiento: SET, se describe en el Capítulo VI.

5.-VERBOS de COMPARACION

5.1.-DECLARACIONES CONDICIONALES

Entre las declaraciones que se especifican dentro de un programa, existe normalmente alguna que obliga al ordenador a tomar una DECISION,consistente en elegir un camino entre los dos posibles que se le presentan para continuar la ejecución del programa.

La elección de uno u otro camino depende del resultado –CIERTO o FALSO– de la prueba de una CONDICION. En la ejecución del programa Nómina, por ejemplo, el ordenador no ejecuta el mismo proceso con los datos de cada operario, ya que según se describía en el análisis del problema: Si el operario era CASADO recibía un plus familiar de 500 pts. por la mujer, mientras que no recibía plus alguno si era SOLTERO; de aquí que, dependiendo del resultado de la CONDICION –CASADO, SOLTERO, VIUDO– tendrá el ordenador que ejecutar unas declaraciones y omitir otras.

Un tipo de declaración que puede alterar el orden secuencial en que normalmente se van ejecutanto todas las declaraciones del programa, según el resultado de la prueba de una condición, está constituido por la palabra IF seguida de la CONDICION que hay que probar, de la declaración o declaraciones a ejecutar caso de que la condición sea cierta, y de la declaración o declaraciones a ejecutar caso de que la condición sea falsa.

FORMATO:

IF condición THEN { {declaración – 1}… / NEXT SENTENCE } { ELSE {declaración – 2} … / ELSE NEXT SENTENCE }

La forma como se ejecuta una declaración IF es la siguiente:

– El ordenador prueba la condición, y si resulta ser CIERTA, ejecuta la declaración o declaraciones que siguen a la condición hasta la palabra ELSE, omite la ejecución de las declaraciones que siguen a ELSE hasta el primer punto, y continúa la ejecución del programa a partir de la declaración que sigue a dicho punto:

```
007060  CALCULO-TRIENIOS.
007070      IF MES OF FECHAINGRESO < M
007080          COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3
007085      ELSE
007090          COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
007100      COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
```

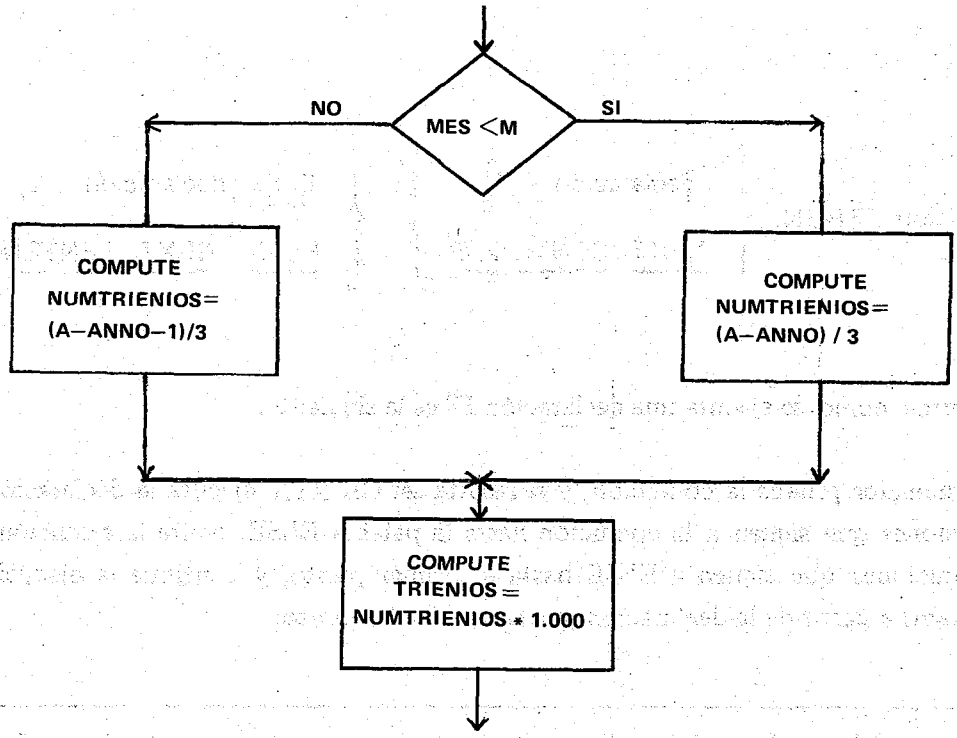
La condición que se prueba es: "Si el mes de la fecha de ingreso del operario en la empresa, denominado MES, es menor que el mes en que se confecciona la nómina, denominado M". Suponiendo que al ejecutar la declaración IF el valor de MES es 7 y el de M es 12, en este caso la prueba MES OF FECHAINGRESO < M es CIERTA, por lo que el ordenador ejecutará la declaración de la línea 007080, omitirá la declaración de la línea 007090, continuando la ejecución del programa a partir de la declaración de la línea 007100.

Si al probar la condición, resulta ser FALSA, omite la ejecución de la declaración o declaraciones que siguen a la condición hasta la palabra ELSE, continuando la ejecución del programa en la declaración o declaraciones que siguen a ELSE.

Es decir, suponiendo que el valor de MES es 7 y el de M es 5, al probar la condición, ésta resultará falsa por no ser MES < M, por lo que se omite la ejecución de la declaración de la línea 007080, pasando la secuencia de ejecución del programa a ejecutar las declaraciones que siguen a ELSE, es decir, líneas 007090, 007100 etc.

En forma de ordinograma se pueden representar las declaraciones que componen el procedimiento CALCULO–TRIENIOS como sigue:

CALCULO – TRIENIOS



Opciones ELSE NEXT SENTENCE y NEXT SENTENCE.

En el siguiente ejemplo se ha utilizado la opción ELSE NEXT SENTENCE:

```
008120     IF  CONTALINEAS = 24
008121         WRITE IMPRESO  AFTER  PAG-NUEVA
008130         PERFORM  ESCRIBIR-2
008131         MOVE ZERO  TO CONTALINEAS
008132     ELSE
008135         NEXT SENTENCE.
008140     WRITE IMPRESO FROM IMPRESO-S  AFTER ADVANCING 2 LINES
008145     ADD 1 TO CONTALINEAS
```

En la descripción precedente,si se supone que en la hoja de papel de la impresora se llevan ya escritos los nombres de veinticuatro operarios con sus correspondientes totales, antes de escribir el operario veinticinco, prueba si la condición es cierta, es decir, si CONTALINEAS es igual a 24, como en ese momento sí lo es, la secuencia de ejecución del programa ejecutaría: WRITE IMPRESO AFTER PAG-NUEVA, PERFORM ESCRIBIR-2 y MOVE ZERO TO CONTALINEAS, continuando la ejecución del programa en la declaración de la línea 008140. Por el contrario,si la condición hubiera sido falsa (CONTALINEAS no fuera igual a 24), omitiría la ejecución de las declaraciones WRITE IMPRESO AFTER PAG-NUEVA, PERFORM ESCRIBIR-2 y MOVE ZERO TO CONTALINEAS para continuar la ejecución del programa en la declaracion de la línea 008140.

En la codificación del siguiente ejemplo se ha empleado la opción NEXT SENTENCE:

```
008120     IF  CONTALINEAS  NOT = 24
008121         NEXT SENTENCE
008125     ELSE
008132         WRITE IMPRESO  AFTER  PAG-NUEVA
008134         PERFORM ESCRIBIR-2
008135         MOVE ZERO TO CONTALINEAS.
008140     WRITE IMPRESO FROM IMPRESO-S AFTER ADVANCING 2 LINES
008145     ADD 1 TO CONTALINEAS
```

En este caso, si el resultado de la prueba de la condición: CONTALINEAS NOT = 24 es cierta, la secuencia de ejecución del programa pasa a ejecutar la primera declaración que sigue al punto en el que termina la declaración IF, es decir, a declaración de la línea 008140. Por el contrario, si el resultado de la prueba citada fuera falso pasaría a ejecutar la declaración de la línea 008132 y siguientes.

Sustitución de ELSE NEXT SENTENCE por un PUNTO.

La opción ELSE NEXT SENTENCE puede omitirse y ser sustituída por un PUNTO, con lo que, aunque aparentemente queda incompleta la declaración, sin embargo, el ordenador considera a dicho punto como el de comienzo de las declaraciones a ejecutar cuando el resultado de la prueba de la condición sea falso.

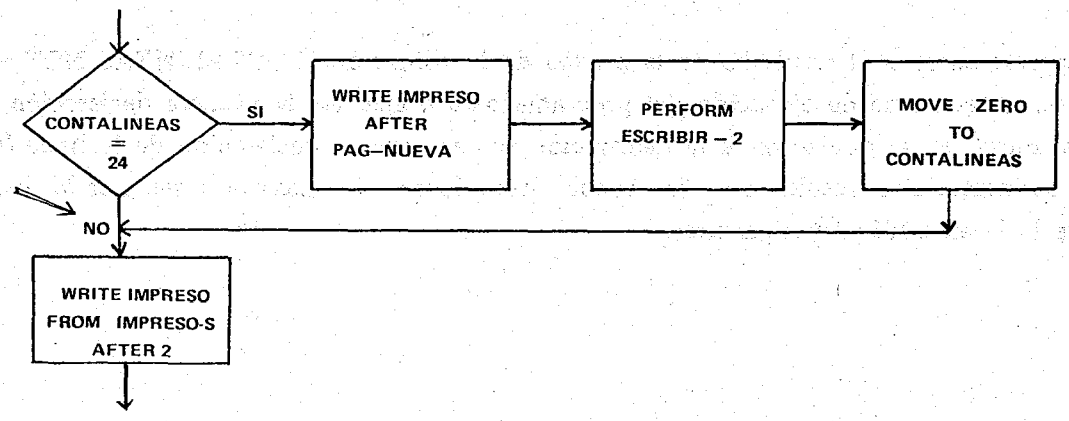
Es decir, prueba la condición, si resulta ser cierta, ejecuta las declaraciones que siguen a dicha condición; pero si el resultado de la prueba es falso, omite la ejecución de la declaración o declaraciones que siguen a dicha condición hasta el citado punto, continuando la ejecución del programa a partir de este punto.

En el siguiente ejemplo se ha omitido la opción ELSE NEXT SENTENCE de la declaración IF del ejemplo anterior, sustituyéndola por el punto que sigue a CONTALINEAS:

```
008120      IF  CONTALINEAS = 24
008121          WRITE IMPRESO AFTER PAG-NUEVA
008130          PERFORM ESCRIBIR-2
008131          MOVE ZERO TO CONTALINEAS. <
008140      WRITE IMPRESO FROM IMPRESO-S AFTER ADVANCING 2 LINES
008145      ADD 1 TO CONTALINEAS
```

Su efecto es el siguiente: si el resultado de la prueba de la condición CONTALINEAS = 24, es falso, es decir, que aún no se han impreso veinticuatro líneas, omite la ejecución de las declaraciones de las líneas 008121, 008130 y 008131, pasando a ejecutar la declaración de la línea 008140 que es la primera declaración a partir del punto indicativo del final de la declaración IF. Por el contrario, si el resultado de la prueba fuera cierto, ejecutaría las declaraciones de la línea 008121, 008130 y 008131 y continuaría la ejecución del programa en la declaración de la línea 008140.

Para poder sustituir ELSE NEXT SENTENCE por un punto, es necesario que, el ordinograma sea de la forma siguiente:



Como resumen, se puede decir que es aconsejable, sobre todo para el programador poco experimentado, NO prescindir del ELSE, es decir, que cada declaración IF vaya acompañada del correspondiente ELSE o ELSE NEXT SENTENCE ya que hace más legible y comprensible el programa.

5.2.- PRUEBA de CONDICIONES

El ordenador prueba cada tipo de condición para determinar cuál es el resultado –CIERTO o FALSO– de dicha prueba. Cada prueba la hace de diferente forma, dependiendo del tipo de condición; por tanto,habrá tantos tipos de pruebas como tipos de condiciones.

A continuación se describen los siguientes tipos de pruebas:

- **RELACION**
- **CLASE**
- **SIGNO**
- **NOMBRE–DE–CONDICION**

5.2.1.- Prueba de RELACION

FUNCION: Comparar dos operandos, obteniendo de dicha comparación si un operando es mayor, igual o menor que el otro.

FORMATO:

```
IF { identificador–1          }  IS [NOT] { >                 }  { identificador–2          }
   { expresión–aritmética–1   }             { <                 }  { expresión–aritmética–2   }
   { constante–figurativa–1   }             { =                 }  { constante–figurativa–2   }
   { literal–1                }             { GREATER THAN      }  { literal–2                }
                                            { LESS THAN         }
                                            { EQUAL TO          }
```

- Identificador–1, identificador–2, literal–1, etc. son los operandos.

- Los signos >, < e = equivalen a GREATER THAN, LESS THAN y EQUAL TO, por lo que su empleo es indistinto. Así, la declaración condicional:

```
030     IF  CAMPOA > CAMPOB
040         MOVE  ZEROS TO CAMPOA
050     ELSE
060         ADD 125 TO CAMPOB.
```

es equivalente a:

```
030     IF  CAMPOA IS GREATER THAN CAMPOB
040         MOVE ZEROS TO CAMPOB
050     ELSE
060         ADD 125 TO CAMPOB.
```

Con la prueba de una condición de relación lo que se pretende al comparar dos operandos, es conocer si uno de ellos es mayor, menor o igual que el otro, para lo cual el ordenador debe estar provisto de una tabla de intercalación o clasificación –cuyo contenido será cada uno de los posibles caracteres que han de formar parte de los operandos con los pesos o valores que se les asigne– que le ha de servir de punto de referencia.

El peso o valor dado a cada carácter,en orden descendente de valores,es el que figura en la tabla siguiente:

HIGH–VALUE	9	8	7	6	5	4	3	2	1	0	Z	Y	X	W	V
	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
	F	E	D	C	B	A	=	'	>	,	/	–	;	)	*
	$	+	(	<	.	SPACE		LOW–VALUE							

TABLA DE CLASIFICACION

– La constante figurativa HIGH–VALUE es el mayor valor de la tabla.

– El valor de cualquier número es mayor que cualquier letra o carácter especial.

– La constante figurativa LOW–VALUE es el menor valor de la tabla, etc.

El orden en que están especificados dentro de la tabla los diferentes caracteres que la componen lo establece el constructor.

Las pruebas de comparación entre dos operandos se pueden clasificar en:NUMERICAS y NO NUMERICAS, según el tipo de PICTURE de cada uno de los operandos a comparar.

Comparaciones NUMERICAS

Se llevan a cabo cuando la PICTURE de los dos operandos son de tipo numérico, o bien, uno de los operandos es un campo con PICTURE numérica y el otro un literal numérico,etc. (ver cuadro de comparaciones permitidas).

Una comparación numérica utiliza, de la tabla de clasificación, las diez cifras decimales.

EJEMPLO.

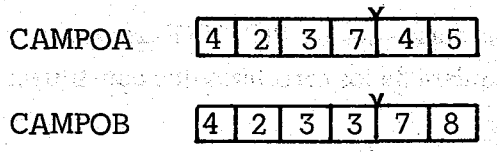
Con la declaración siguiente se va a efectuar la comparación numérica:

```
080     IF  CAMPOA  <  CAMPOB
090         ADD  1  TO  CAMPOA
100     ELSE
110         SUBTRACT 1 FROM CAMPOB.
```

Se supone que el contenido de CAMPOA en el momento de ejecutar la declaración es 4237.45, y el del CAMPOB : 4233 . 78.

Los pasos que el ordenador lleva a cabo para realizar una comparación numérica son los siguientes:

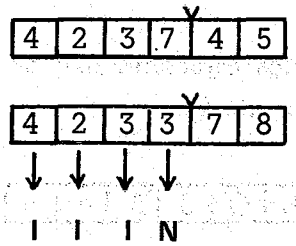
– Alinéa los operandos por el punto decimal.



– Compara el signo de los dos operandos basándose en que cualquier número positivo o sin signo es mayor que cualquier número negativo, y que, si los dos números son negativos, el menor de ellos en valor absoluto es el mayor.

En este ejemplo, tanto CAMPOA como CAMPOB carecen de signo, por lo que el ordenador los considera positivos; y por tanto, de este primera comparación no puede deducir cuál de los dos números es el mayor. En el caso de que el signo de CAMPOA fuera negativo y el de CAMPOB positivo, obtendría que CAMPOA es menor que CAMPOB, y en ese momento finalizaría la comparación.

– Compara cada cifra de un operando con su correspondiente en el otro comenzando por la izquierda, basándose en la secuencia de la tabla anteriormente descrita:

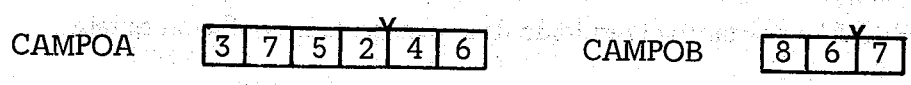


I = IGUAL
N = NO IGUAL

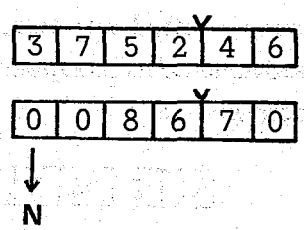
hasta que la comparación de dos cifras resulte ser desigual, en cuyo caso, el operando que contenga la cifra mayor, es el mayor si el signo de los operandos es positivo, o por el contrario es el menor si el signo de los dos operandos es negativo. Por tanto, en este ejemplo deduciría que CAMPOA es el mayor.

– Si los operandos son de distinta longitud, el de menor longitud, automáticamente se iguala en número de cifras al operando de mayor longitud, añadiéndole ceros a izquierda y derecha de la coma, realizándose la comparación como en el caso anterior.

EJEMPLO. Comparar CAMPOA con CAMPOB, siendo sus valores respectivos:



Efectuaría la comparación como sigue:



De la primera comparación deduce que CAMPOA es mayor que CAMPOB puesto que 3 es mayor que 0, según la tabla de clasificación.

Comparaciones NO NUMERICAS

Estas comparaciones se llevan a cabo con operandos de PICTURE de tipo no numérico cuyo contenido sea una combinación de cualquiera de los caracteres que constituyen la tabla de clasificación.

Sea la declaración:

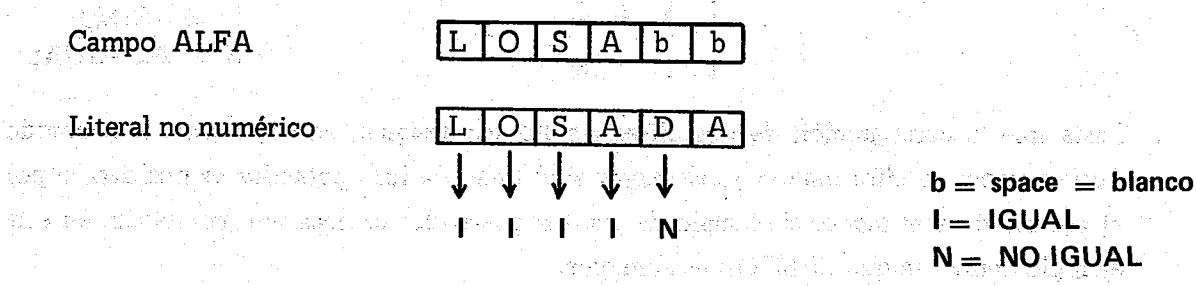
```
120     IF  ALFA = 'LOSADA'
130         MOVE  SPACES TO ALFA
140     ELSE
150         PERFORM  COMPARAR.
```

en la que se supone que el contenido del campo ALFA es [L|O|S|A]

El ordenador realiza la comparación siguiendo los pasos siguientes:

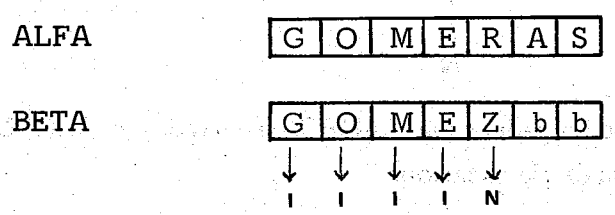
- Alinéa por la izquierda los dos operandos, completando con blancos el operando de menor longitud.

- Comienza comparando cada dos caracteres que ocupen la misma posición en cada operando,a partir del extremo izquierdo:



- La comparación termina cuando se encuentra con un par de caracteres diferentes, o bien, si cada par de caracteres hasta llegar al extremo derecho de cada operando resultan ser iguales, quiere decir que los dos operandos son iguales. En el ejemplo, la comparación termina al comparar el carácter D con b; ya que por ser la letra D mayor que el carácter blanco, el contenido del campo ALFA es menor que el literal no numérico 'LOSADA' y por tanto el resultado de la prueba de la condición es falso.

- Si un operando es de mayor longitud que otro, no quiere decir que en la comparación resulte ser el mayor. Efectivamente, si se supone que el contenido de los campos de datos ALFA y BETA son respectivamente:



GOMEZ es mayor que GOMERAS, ya que el primer par de caracteres desiguales que encuentra, indican que Z es mayor que R.

Las comparaciones entre dos literales (numéricos o no numéricos) no está permitida, ya que no es lógico especificar IF 125 < 183 ..... o IF 'RODRIGUEZ' > 'PEREZ' porque explícitamente ya se conoce el resultado.

CUADRO DE COMPARACIONES PERMITIDAS

	SEGUNDO OPERANDO					
PRIMER OPERANDO	COMPUESTO	ALFA-BETICO	ALFANU-MERICO	NUMERICO	INFORME	CONSTANTE FIGURA-TIVA
CAMPO COMPUESTO	NN	NN	NN	NN	NN	NN
CAMPO ALFABETICO	NN	NN	NN			NN (3)
CAMPO ALFANUMERICO	NN	NN	NN	NN (2)	NN	NN
CAMPO NUMERICO	NN	NN (2)	NN (2)	NU		NN (1)
EDICION O INFORME	NN		NN		NN	NN (4)
CONSTANTE FIGURATIVA	NN	NN (3)	NN	NN (1)	NN (4)	

NN Permitida la comparación no numérica.
NU Permitida la comparación numérica.
(1) Permitida si la constante figurativa es ZERO o ALL 'carácter numérico'
(2) Permitida solamente si el campo es decimal externo y entero.
(3) Permitida si la constante figurativa es SPACE o ALL 'carácter alfabético'
(4) No está permitida si la constante figurativa es QUOTE.

Los literales no numéricos se consideran como campos alfanuméricos.

Los literales numéricos se consideran como campos numéricos.

5.2.2.- Prueba de CLASE

FUNCION: Probar si el contenido de un campo de datos es alfabético o numérico.

FORMATO:

IF identificador **IS** [**NOT**] { **NUMERIC** / **ALPHABETIC** }

La condición NUMERIC es cierta cuando al probar Identificador, éste contiene solo dígitos con o sin signo. Esta prueba sólo puede realizarse con campos con PICTURE de tipo numérico y alfanumérico.

La condición ALPHABETIC es cierta cuando Identificador contiene letras y/o espacios. Este tipo de prueba sólo puede realizarse con campos de PICTURE de tipo alfabético y alfanumérico.

El nombre del identificador que se prueba debe llevar USAGE DISPLAY.

EJEMPLO.

En el programa Nómina, el código de cada operario contiene 5 posiciones y su contenido siempre son números (NUMERICO); pero, si por error en la perforación, se ha perforado en alguna de esas posiciones una letra o cualquier otro carácter en vez de un número, es necesario que el ordenador detecte dicho error y ejecute las acciones oportunas, por lo que el programador deberá especificar dentro del programa:

```
150     IF CODIGO IS NOT NUMERIC
160         PERFORM RUTINA-ERROR
170     ELSE
180         PERFORM CALCULAR.
```

5.2.3.- Prueba de SIGNO

FUNCION: Probar si un campo numérico o expresión aritmética es POSITIVO, NEGATIVO o CERO.

FORMATO:

IF { identificador / expresión – aritmética } **IS** [**NOT**] { **POSITIVE** / **NEGATIVE** / **ZERO** }

EJEMPLO. En las declaraciones:

```
190     IF  CAMPOA  IS  NEGATIVE
200         MULTIPLY  -1  BY  CAMPOA
210     ELSE
220         ADD  1000  TO  CAMPOA.
```

```
231     IF  (ALFA * 7) / 4  IS  POSITIVE
232         ADD  -25  TO  ALFA
233     ELSE
234         MOVE  BETA TO ALFA.
```

La prueba con la condición POSITIVE es cierta cuando el contenido del campo de datos CAMPOA o la expresión aritmética (ALFA * 7) / 4 es mayor que cero.

La prueba con la condición NEGATIVE es cierta cuando el contenido del campo de datos o la expresión aritmética es menor que cero.

La prueba con la condición ZERO es cierta cuando el contenido del campo de datos o la expresión aritmética es igual a cero.

Si se emplea la opción NOT, la prueba con una cualquiera de las tres condiciones, POSITIVE, NEGATIVE o ZERO, es cierta cuando se cumpla cualquiera de las dos restantes.

En la práctica, la prueba de signo se puede sustituir por una prueba de relación equivalente. Por ejemplo:

IF CAMPOA IS POSITIVE equivale a IF CAMPOA > ZERO

IF CAMPOA IS NEGATIVE equivale a IF CAMPOA < ZERO

IF CAMPOA IS ZERO equivale a IF CAMPOA = ZERO

5.2.4.- Prueba de NOMBRE-DE-CONDICION

FUNCION: Probar si un nombre–de–condición al que se ha asociado un determinado valor, coincide con el contenido del campo de datos de quien depende.

FORMATO:

IF [NOT] nombre – de – condición

La condición se cumple, o es cierta, cuando el valor asignado al nombre–de–condición coincide con el valor que, en el momento de la prueba, contenga el campo elemental de quien depende.

EJEMPLO:

```
007110  CALCULO-PLUSF.
007120      IF CASADO COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007130      ELSE
007140          IF VIUDO COMPUTE PLUSF = NUM-HIJOS * 250
007150          ELSE MOVE 0 TO PLUSF.
```

si en el momento de ejecutar la declaración IF, (línea 007120), el contenido del campo ESTADO es 2, el resultado de la prueba de la condición será cierto, ya que en el programa NOMINA el valor asignado al nombre–de–condición CASADO es 2, ejecutando por tanto:

```
        COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
```

si por el contrario, el contenido de ESTADO es distinto de 2, la condición no se cumplirá, por lo que omitirá dicha declaración, continuando la ejecución del programa en la declaración de la línea 007140.

La anterior descripción se puede sustituir por una prueba de relación equivalente:

```
007110  CALCULO-PLUSF.
007120      IF ESTADO = 2 COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007130      ELSE
007140          IF ESTADO = 3 COMPUTE PLUSF = NUM-HIJOS * 250
007150          ELSE MOVE 0 TO PLUSF.
```

Comparando las dos descripciones, se obtiene como conclusión que con el primer sistema se abrevia la escritura del programa, y se hace más comprensible la PROCEDURE DIVISION para cualquier programador que lea el programa por primera vez, mientras que utilizando la segunda descripción es necesario consultar el dossier de documentación del programa para conocer la codificación que el programador asignó a los diferentes ESTADOS civiles del operario.

5.3.– CONDICIONES COMPUESTAS

Dos o más condiciones simples, enlazadas por medio de los operadores lógicos AND y/o OR según determinadas restricciones, dan origen a una condición compuesta.

FORMATO:

$$\underline{\text{IF}} \quad \text{condición – simple – 1} \left\{ \begin{matrix} \underline{\text{AND}} \\ \underline{\text{OR}} \end{matrix} \right\} \quad \text{condición – simple – 2} \left\{ \begin{matrix} \underline{\text{AND}} \\ \underline{\text{OR}} \end{matrix} \right\} \cdots$$

- Condición–simple–1, condición–simple–2, etc., pueden ser una cualquiera de las cuatro clases de condiciones simples.

- Una condición compuesta, constituida únicamente por dos condiciones simples y enlazadas por el operador AND es CIERTA, si son ciertas las dos condiciones simples. Sin embargo, si las dos condiciones simples están enlazadas por el operador OR, basta que sea CIERTA una cualquiera de ellas, para que la condición compuesta sea CIERTA.

EJEMPLO. Las dos declaraciones que siguen, sólo se diferencian en el operador lógico:

```
080     IF  ALFA = FIN AND BETA = 5
090         PERFORM UNO
100     ELSE
110         PERFORM DOS.
```

```
130     IF  ALFA = FIN OR BETA = 5
140         PERFORM UNO
150     ELSE
160         PERFORM DOS.
```

En la siguiente tabla se muestra en las dos primeras columnas las posibles combinaciones –CIERTA o FALSA– de cada condición simple, así como en las dos columnas siguientes el resultado de la condición compuesta, conforme al resultado de cada una de las condiciones simples que la forman.

ALFA = FIN	BETA = 5	ALFA = FIN AND BETA = 5	ALFA = FIN OR BETA = 5
CIERTA	CIERTA	CIERTA	CIERTA
FALSA	CIERTA	FALSA	CIERTA
FALSA	FALSA	FALSA	FALSA
CIERTA	FALSA	FALSA	CIERTA

Lo mismo que en las expresiones aritméticas se utilizan los paréntesis para indicar el orden en que se han de ejecutar las operaciones, también se emplean éstos para indicar el orden en que se han de probar las distintas condiciones simples que forman la condición compuesta. Por tanto:

El programador puede modificar el orden en que se probarán las condiciones, según que en una condición compuesta:

- No se especifiquen paréntesis, o

- Se especifiquen.

– Si en una condición compuesta no se especifican paréntesis, el ordenador, siguiendo el orden de izquierda a derecha, va realizando la prueba de cada pareja de condiciones simples que esten enlazadas por el operador AND, y a continuación prueba todas las parejas de condiciones enlazadas por el operador OR.

EJEMPLO:

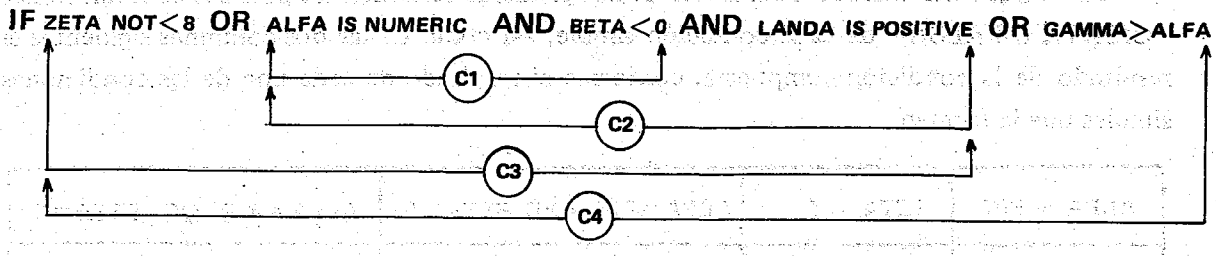
Se quiere hallar el resultado, CIERTO o FALSO, de la prueba de la siguiente condición compuesta a fin de deducir hacia qué procedimiento ha de bifurcar la secuencia de ejecución.

```
160     IF  ZETA  NOT < 8  OR  ALFA IS NUMERIC  AND  BETA < 0
170         AND  LANDA  IS POSITIVE  OR  GAMMA > ALFA
180         PERFORM  LEER
190     ELSE
200         PERFORM 12.
```

Para simplificar se designa:

C1 a la condición compuesta: ALFA IS NUMERIC AND BETA < 0.
C2 a la condición compuesta: C1 AND LANDA IS POSITIVE.
C3 a la condición compuesta: ZETA NOT < 8 OR C2.
C4 a la condición compuesta: C3 OR GAMMA > ALFA.

Gráficamente,se muestra el orden que se irán efectuando las diferentes pruebas:



Para conocer si el resultado de la prueba de la condición compuesta es FALSO o CIERTO, se supondrá que el contenido de cada uno de los campos, en el momento de ejecutar la declaración IF, es:

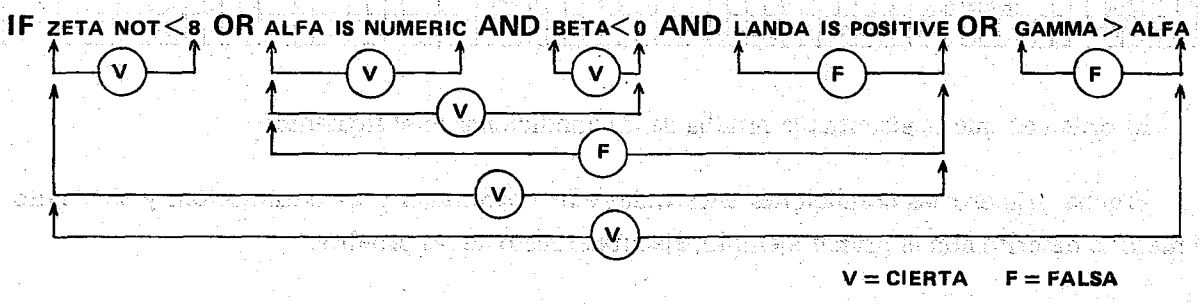
ZETA = 9 ALFA = 5 BETA = −3 LANDA = −1 GAMMA = −2

La prueba de la condición la ejecutará el ordenador siguiendo las reglas citadas anteriormente y en el siguiente orden:

- Prueba C1, dando un resultado CIERTO, ya que la condición: ALFA IS NUMERIC es cierta, y BETA < 0 es cierta.
- Prueba C2, dando un resultado FALSO, ya que si bien C1 es cierta, sin embargo, la condición: LANDA IS POSITIVE es falsa.
- Prueba C3, dando un resultado CIERTO, ya que aunque la condición C2 es falsa, sin embargo, la condición: ZETA NOT < 8 es cierta.
- Prueba C4, dando un resultado CIERTO, ya que aunque la condición: GAMMA > ALFA es falsa, sin embargo, C3 es cierta.

Por tanto, el resultado de la prueba de la condición compuesta es CIERTO, por lo que la secuencia de ejecución bifurcará hacia el procedimiento LEER.

El gráfico siguiente ayuda a comprender la solución anterior:



- Si en una condición compuesta se han especificado paréntesis, el ordenador, dentro de cada paréntesis y comenzando por el más interno hasta llegar al más externo, realiza la prueba de cada pareja condiciones simples, en el orden que se ha indicado cuando en una condición compuesta no se especificaban paréntesis, es decir, prueba, de izquierda a derecha, las parejas de condiciones simples unidas por el operador AND, y posteriormente las unidas por el operador OR; finalizando la evaluación de la condición compuesta con la prueba de aquellas parejas de condiciones que queden sin encerrar entre paréntesis.

EJEMPLO:

```
160     IF  (ZETA NOT < 8 OR ((ALFA IS NUMERIC AND BETA < 0)
170         AND LANDA IS POSITIVE)) OR GAMMA > ALFA
180     PERFORM LEER
190 ELSE
200     PERFORM 12.
```

Tal como se han especificado los paréntesis en este ejemplo, las pruebas a efectuar por el ordenador serían las mismas y ejecutadas en el mismo orden que se realizaron con el ejemplo anterior donde no se especificaron los paréntesis, con lo que se llegaría al mismo resultado.

Para el programador puede ser éste el método más sencillo de especificar al ordenador el orden en que debe realizar las pruebas, ya que pueden existir condiciones compuestas muy ambiguas que con el uso de los paréntesis dejan de serlo.

No obstante, el uso de paréntesis en las condiciones compuestas debe hacerse con precaución, ya que éstos al poder alterar el orden en que el ordenador hará cada prueba, da lugar a que según su colocación el resultado de la prueba de una condición compuesta pueda ser también diferente.

La condición compuesta del ejemplo siguiente está constituida por las mismas condiciones simples que la del ejemplo anterior, pero se han colocado los paréntesis en diferente orden, y se va a probar dicha condición suponiendo que el contenido de los campos sea el mismo que el

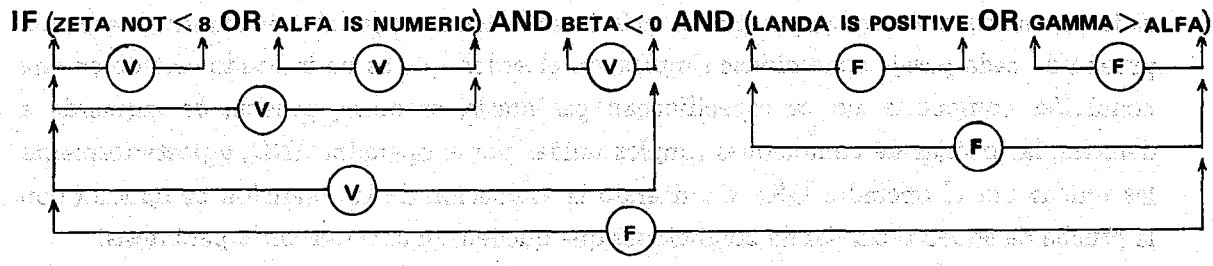
fijado en la página 192.

```
160      IF (ZETA NOT < 8 OR ALFA IS NUMERIC) AND BETA < 0
170          AND (LANDA IS POSITIVE OR GAMMA > ALFA)
180          PERFORM LEER
190      ELSE
200          PERFORM 12.
```

El orden en que se ejecutan la prueba de las condiciones es el siguiente:

– Prueba primero las condiciones encerradas entre paréntesis y a continuación, y siguiendo el método descrito con el primer ejemplo, ejecuta el resto de las pruebas.

En forma gráfica, se especifica el orden en que van ejecutándose:



En este caso, el resultado obtenido es contrario (FALSO) al que se obtuvo en el ejemplo anterior, en el que no se utilizaron los paréntesis.

El ejemplo que se expone a continuación servirá como práctica de la declaración IF con condiciones compuestas.

EJEMPLO:

La Sección de Estadística de la empresa RICOSA encarga al Centro de Proceso de Datos, la confección de un programa con el que, ejecutándolo el último día de cada mes, se obtenga una relación de operarios que reunan las siguientes características:

– Varones, casados o viudos.

– De 30 años de edad.

– Con familia numerosa, y

– Sueldo neto comprendido entre 20.000 y 40.000 pts.

Una solución del problema consiste en introducir en la memoria del ordenador, el mes y año en que se va a ejecutar el programa, para poderlo comparar con el de la fecha de nacimiento de cada operario, y ver si cumple la condición de que la edad es de 30 años; y a continuación,

comparar los datos de cada operario para ver si cumplen las restantes condiciones exigidas, relacionando en el impreso a aquéllos que las reunan.

Por ser uno de los datos a tener en cuenta en cada operario el valor de su sueldo neto, hay que utilizar como archivo de entrada la cinta magnética grabada durante la ejecución del programa NOMINA, ya que en ella figura, además de los datos de cada operario, dicho sueldo.

La codificación del programa que resuelve el problema es la que figura en la página siguiente, de la que a continuación se hace una breve descripción de su contenido:

De la línea 001010 a la línea 002110 se describen: el nombre del programa (línea 001020), el archivo de entrada (líneas 001120 a la 001150) y los registros de dicho archivo (líneas 001160 a la 002110).

De la línea 003010 a la 003160 se describe el proceso a llevar a cabo con los datos, el cual, el programador lo ha realizado de la siguiente forma:

La declaración DISPLAY (línea 003020) imprime en la máquina de escribir de la consola el mensaje al operador: 'ESCRIBIR MES Y AÑO.....' e inmediatamente después la declaración ACCEPT (línea 003040) detiene el ordenador para dar tiempo al operador a que teclee la fecha que se le solicita; lo cual una vez realizado, se introduce ésta en la memoria del ordenador, colocándose dentro del campo FECHA, descrito de la línea 002150 a la 002180.

A continuación escribe por la impresora el encabezamiento que ha de llevar el informe: 'RELACION DE PERSONAS ..... ' (líneas 003050 y 003060), abre el archivo en cinta que contiene los datos de los operarios y lee el primer registro (línea 003080).

Resta el año de nacimiento de cada operario del año que tecleó el operador (línea 003090), introduciendo la diferencia en el campo de trabajo ANNOS, descrito en la línea 002140.

Con la declaración IF se prueba la condición de si el operario tiene la edad de 30 años (línea 003100), junto con el resto de las condiciones que deben cumplir (líneas 003110 a la 003130); aquéllos que las cumplan se relacionan en el informe (línea 003140), y en caso contrario se ignoran y se ordena ir a leer un nuevo registro (línea 003150).

Finalizada la lectura del archivo de entrada, termina el proceso (línea 003160).

Hay que advertir, que cuando la condición compuesta que se haya de especificar en el programa resulte ser muy compleja, como la que se muestra en este ejemplo, no es aconsejable su empleo; debiendo ser sustituida por varias declaraciones IF acompañadas de la correspondiente condición simple, que en conjunto realicen el mismo trabajo.

```
000100  IDENTIFICATION DIVISION.
000200      PROGRAM-ID.  ESTADIS.
000300*                 ESTE PROGRAMA OBTIENE UN LISTADO DE HOMBRES CASADOS
000400*                 O VIUDOS DE EDAD IGUAL A 30 AÑOS, CON MAS DE
000500*                 3 HIJOS Y SUELDO COMPRENDIDO ENTRE 20000 Y 40000 PTS.
000600  ENVIRONMENT DIVISION.
000700      INPUT-OUTPUT SECTION.
000800      FILE-CONTROL.
000900          SELECT CINTA-NOMINA ASSIGN  SYS005-UT-2400-S      RESERVE NO.
001000  DATA DIVISION.
001100      FILE SECTION.
001200      FD  CINTA-NOMINA      RECORDING  F
001300          BLOCK 10  RECORDS
001400          LABEL  RECORD OMITTED
001500          DATA  RECORD  REGISTRO-OPERARIO.
001600  01      REGISTRO-OPERARIO.
001700      02    FILLER                          PICTURE X(40).
001800      02    FECHANACI.
001900          03    DIA                         PICTURE 99.
002000          03    MES                         PICTURE 99.
002100          03    ANNO                        PICTURE 99.
002200      02    FILLER                          PICTURE 99.
002300      02  ESTADO                           PICTURE   9.
002400          88    SOLTERO       VALUE  1.
002500          88    CASADO        VALUE  2.
002600          88    VIUDO         VALUE  3.
002700      02  SEXO                           PICTURE   X.
002800          88    VARON         VALUE `V'.
002900          88    HEMBRA        VALUE `H'.
003000      02    NUM-HIJOS                       PICTURE  99.
003100      02  FILLER                           PICTURE  X(28).
003200      02  SUELDO                           PICTURE  9(6).
003300  WORKING-STORAGE SECTION.
003400  77  ANNOS                                PICTURE  99.
003500  01      FECHA.
003600      02    MM                              PICTURE  9(2).
003700      02    FILLER                          PICTURE  X.
003800      02    AA                              PICTURE  9(2).
003900  PROCEDURE DIVISION.
004000      DISPLAY `ESCRIBIR MES Y ANO ACTUAL SEGUN FORMATO MM/AA'
004100                                  UPON CONSOLE.
004200      ACCEPT FECHA FROM CONSOLE
004300      DISPLAY `          RELACION DE PERSONAS QUE REUNEN LAS CONDICION
004400-                         `ES FIJADAS'.
004500      OPEN INPUT CINTA-NOMINA.
004600  LEER.  READ CINTA-NOMINA AT END  GO TO FIN.
004700      SUBTRACT ANNO FROM  AA  GIVING ANNOS
004800      IF  ((ANNOS  =  30  AND  MM  NOT  <  MES)  OR  (ANNOS  =  31
004900          AND  MM  <  MES))  AND  VARON  AND  (CASADO  OR  VIUDO)
005000          AND  NUM-HIJOS NOT LESS THAN 4
005100          AND  SUELDO  NOT GREATER 40000  AND SUELDO  NOT  <  20000
005200          DISPLAY REGISTRO-OPERARIO
005300      ELSE      GO TO LEER.
005400  FIN.  CLOSE  CINTA-NOMINA    STOP  RUN.
```

En una condición compuesta todos sus elementos integrantes: operadores lógicos AND y OR, NOT, los paréntesis y condiciones simples, no se pueden especificar caprichosamente en cualquier orden, sino que deben combinarse según determinadas reglas, como se muestra en la Tabla siguiente:

	SEGUNDO SIMBOLO					
PRIMER SIMBOLO	C	AND	OR	NOT	(	)
C		A	A			A
AND	A			A	A	
OR	A			A	A	
NOT	A				A	
(	A			A	A	
)		A	A			A

C — Condición simple.
A — Indica que la combinación del par de símbolos es admisible.
Las casillas en blanco indican que no es admisible la combinación.

INTERPRETACION DE LA TABLA

Un ejemplo muestra la forma de interpretar la Tabla: Se trata de escribir una declaración en la que se especifique que, cuando NUMTRIENIOS = 6 y NUM–HIJOS = 4 ó SUELDO < 20.000 pts., ejecute el procedimiento GRATIFICAR.

Si se escribiera la declaración:

```
IF NUMTRIENIOS = 6 ( AND NUM - HIJOS = 4 OR SUELDO < 20.000 )
        GO TO GRATIFICAR.
```

no sería correcta, ya que teniendo en cuenta que si el primer símbolo es una condición simple: NUMTRIENIOS = 6, el segundo símbolo no puede ser un paréntesis izquierdo tal como se especifica en la declaración, pues, según la Tabla, una condición simple sólo puede ir seguida de AND, OR o un paréntesis derecho.

Por tanto,lo correcto hubiera sido especificar:

```
IF NUMTRIENIOS = 6 AND ( NUM - HIJOS = 4 OR SUELDO < 20.000 )
        GO TO GRATIFICAR.
```

la cual no quebranta ninguna regla de la Tabla.

5.4.- JERARQUIAS de IF

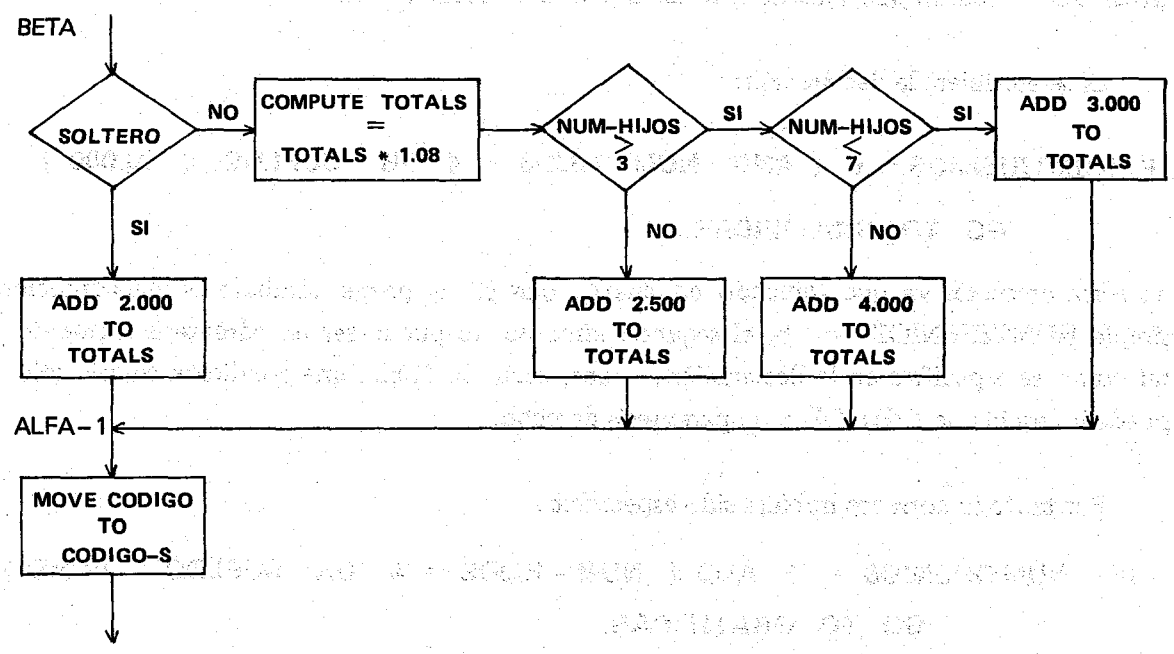
De acuerdo con el formato de la declaración IF: declaración–1, declaración–2, ... pueden ser declaraciones imperativas y/o condicionales. Cuando las declaraciones son condicionales, se está en el caso de declaraciones IF que engloban o comprenden otras declaraciones IF. En este caso, hay que formar parejas de combinaciones ELSE e IF de izquierda a derecha de tal manera que cada ELSE se empareje con la IF que inmediatamente le preceda, o con la IF más próxima que no haya sido emparejada con otra ELSE.

EJEMPLO:

La dirección de la Empresa RICOSA encarga al Centro de Proceso de Datos que haga las necesarias modificaciones en el programa NOMINA, a fin de que, cuando se confeccione la próxima nómina, se gratifique a sus operarios incrementando el sueldo neto en las siguientes cantidades:

- Con 2.000 pts. si es SOLTERO/A.
- Con el 8 % del sueldo neto a todos los CASADOS o VIUDOS más:
  - 2.500 pts. si tiene menos de cuatro hijos.
  - 3.000 pts. si el número de hijos está comprendido entre 4 y 6.
  - 4.000 pts. si tiene más de 6 hijos.

El ordinograma muestra la forma de calcular las gratificaciones:



En el programa NOMINA, al ejecutar la declaración de la línea 007210 es cuando obtiene el valor del sueldo neto, el cual lo introduce en el campo TOTALS, siendo este el momento en que puede utilizarse dicho campo para ir acumulando al sueldo neto las gratificaciones de cada operario, según muestra el ordinograma y cuya codificación podría hacerse de la siguiente forma:

```
BETA.
    ┌→IF NOT SOLTERO COMPUTE TOTALS = TOTALS * 1.08
    │  ┌→IF NUM-HIJOS > 3
    │  │  ┌→IF  NUM-HIJOS < 7  ADD 3000 TO TOTALS
    │  │  └→ELSE  ADD 4000 TO TOTALS
    │  └→ELSE  ADD 2500  TO  TOTALS
    └→ELSE  ADD 2000  TO  TOTALS.

ALFA-1.
        MOVE CODIGO TO CODIGO-S
```

En esta codificación, las flechas indican como están emparejadas cada ELSE con su correspondiente IF. Su interpretación es como sigue:

Si el operario no es SOLTERO, es decir, cumple la primera CONDICION, ejecuta COMPUTE TOTALS = TOTALS * 1.08, continuando la ejecución del programa en el siguiente IF. Si el operario es SOLTERO no se cumple la primera CONDICION y, por tanto, no ejecuta ninguna declaración hasta su ELSE correspondiente, es decir, ejecutaría ADD 2000 TO TOTALS, continuando en MOVE CODIGO TO CODIGO-S. Si no es soltero y con no más de tres hijos, ejecuta la declaración que sigue al ELSE que corresponde a la segunda IF, es decir, ADD 2500 TO TOTALS y continuará en MOVE CODIGO TO CODIGO-S, etc.

Por tanto, esta codificación se puede incorporar al programa NOMINA sin alterar en él ninguna declaración de las ya escritas, sino únicamente integrando las declaraciones descritas anteriormente entre las declaraciones de las líneas 007210 y 008020:

```
007210     SUBTRACT DESCU FROM TOTALBRUTO GIVING TOTALS.
007220   BETA.
007230       IF NOT SOLTERO COMPUTE TOTALS = TOTALS * 1.08
007240           IF NUM-HIJOS > 3
007250               IF NUM-HIJOS < 7  ADD 3000 TO TOTALS
007260               ELSE  ADD 4000 TO TOTALS
007270           ELSE  ADD 2500  TO  TOTALS
007280       ELSE  ADD 2000  TO  TOTALS.
008010   ALFA-1.
008020       MOVE CODIGO TO CODIGO-S
```

6.-VERBO de DETENCION

6.1.-Declaración STOP

FUNCION: Detener temporalmente o finalizar la ejecución del programa.

FORMATO:

STOP { **RUN** / **literal** }

Si se especifica un literal en la declaración STOP, se llevan a cabo las funciones siguientes:

- Se imprime el texto del literal en la máquina de escribir de la consola.
- Se detiene temporalmente la ejecución del programa. Esta detención finaliza cuando el operador pulsa una determinada tecla de dicha máquina, en cuyo momento se reanuda el programa en la declaración siguiente a la STOP.

```
006060          ACCEPT  FECHA  FROM CONSOLE,
006070          STOP  'COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESP
006080-                'ECIAL PARA NOMINA'.
006090 ESCRIBIR-1.
```

El literal puede ser de tipo numérico o no numérico.

Si se especifica STOP RUN, cuando se ejecute dicha declaración hace que termine la ejecución del programa sin posibilidad de poder reanudarse la ejecución del mismo. Por tanto debe ser la última declaración de una sentencia:

```
007040          CLOSE  ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050          STOP RUN.
007060 CALCULO-TRIENIOS.
007070          IF  MES OF FECHAINGRESO < M.
```

ya que sería inútil prescindir del punto que sigue a STOP RUN y especificar:

```
007040          CLOSE ARCHIVO-OPERARIOS  CINTA-NOMINA  ARHI-IMPRESO
007050          STOP  RUN    IF MES OF FECHAINGRESO < M
007060                       COMPUTE NUMTRIENIOS = (A - ANNO OF .....
```

por no ejecutar ninguna declaración que siga a STOP RUN.

7.-VERBOS de ENTRADA/SALIDA

7.1.- Declaración OPEN

FUNCION: Permitir la lectura de los archivos de entrada, comprobando el contenido de las etiquetas de principio de archivo, si estas se grabaron en su creación, así como la escritura de archivos de salida y crear etiquetas al comienzo de los mismos si el programador así lo ha especificado.

En archivos de entrada/salida realiza las funciones anteriormente citadas.

FORMATO:

```
OPEN  [ INPUT { nombre-de-archivo      [ WITH NO REWIND ] } ... ]
      [ OUTPUT { nombre-de-archivo     [ WITH NO REWIND ] } ... ]
      [ I-O { nombre-de-archivo } ... ]
```

Se emplea:

- INPUT para abrir los archivos de entrada.
- OUTPUT para abrir los archivos de salida.
- I-O para abrir los archivos de entrada/salida.

La declaración OPEN permite la lectura de un registro, pero no su proceso, ya que previamente debe ser leído por medio de una declaración READ.

Una declaración OPEN posiciona el carrete de una cinta magnética o de un disco al comienzo del primer registro del archivo.

Con una sola declaración OPEN se puede abrir uno o varios archivos :

```
000610  PROCEDURE DIVISION.
000620      OPEN   INPUT   ARCHIVO-OPERARIOS,
000630             OUTPUT  CINTA-NOMINA   ARCHI-IMPRESO,
000640      DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA'
```

Tambien hubiera sido correcto escribir :

```
006010  PROCEDURE DIVISION.
006020      OPEN  INPUT   ARCHIVO-OPERARIOS
006030      OPEN  OUTPUT  CINTA-NOMINA
006040      OPEN  OUTPUT  ARCHI-IMPRESO
006050      .....
```

aunque no sería rentable, ya que cada palabra OPEN que se escriba, obliga a generar el compilador más instrucciones de máquina, cuya ejecución, en este caso, sería innecesaria.

En el programa NOMINA se abrieron los tres archivos a la vez sin que ello fuese obligatorio, ya que pudieron abrirse individualmente en el momento preciso, es decir: ARCHI-IMPRESO sería correcto abrirlo en el lugar donde se abrió (línea 006020) o antes de la declaración WRITE (línea 006090); ARCHIVO-OPERARIOS y CINTA-NOMINA antes de ejecutarse la declaración READ de la línea 007020:

```
006010  PROCEDURE DIVISION.
006020      OPEN  OUTPUT  ARCHI-IMPRESO
006030      DISPLAY 'MARQUE LA FECHA DEL DIA SEGUN .....
.....

007008  ABRE. OPEN  INPUT ARCHIVO-OPERARIOS
007009                OUTPUT  CINTA-NOMINA.
007020  LEER. READ  ARCHIVO-OPERARIOS  AT END .....
.....

008150      MOVE FICHA-OPERARIO  TO DATOS-CIN
008160      WRITE  REGISTRO-OPERARIO
008170      .....
```

La opción NO REWIND se aplica a archivos grabados únicamente en cinta, y su efecto es suprimir el robobinado del carrete, normalmente asociado con la apertura de un archivo.

Se debe utilizar esta opción cuando existen varios archivos grabados en un mismo carrete y se quiere que una vez finalizado el proceso de un archivo, se comience inmediatamente después el proceso del siguiente archivo.

La opción I-O se describe en el Capítulo VII.

7.2.- Declaración READ

FUNCION: Dejar disponible un registro lógico para ser procesado.

FORMATO 1:

READ nombre–de–archivo RECORD [INTO identificador]

AT END declaración-imperativa

- Cada declaración READ deja disponible un registro lógico en el área de entrada para ser procesado, permaneciendo en dicha área hasta que se ejecute la siguente declaración READ, en cuyo momento se sustituye su contenido por el del siguiente registro.
- La palabra AT END es una condición que se prueba con cada declaración READ, resultando ser cierta cuando el ordenador detecta la marca de final de archivo, en cuyo caso se ejecutan las declaraciones que siguen a dicha condición hasta el punto, y en el resto de los casos, al ser la condición falsa, omite las declaraciones que siguen a la condición hasta el primer punto.

Por la forma de realizar sus acciones la declaración READ, se puede comparar con la declaración IF, ya que la función de la CONDICION en la IF equivale a la de la palabra AT END en la READ, y la función de la palabra ELSE en la IF la realiza el punto que sigue inmediatamente a la declaración READ:

```
007010  LEER.
007020      READ ARCHIVO-OPERARIOS, AT END
007030      DISPLAY ' REGISTROS PROCESADOS = ' CONTADOR
007040      CLOSE ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO.
007050      STOP RUN.
007060  CALCULO-TRIENIOS.
007070      IF MES OF FECHAINGRESO < M
```

Es decir, mientras queden registros por leer en el archivo, ejecutará la declaración READ e inmediatamente después la secuencia de ejecución del programa pasará a la declaración IF (línea 007070). Cuando se acaben los registros, se cumple la condición AT END, y la secuencia de ejecución pasará a ejecutar las declaraciones de las líneas 007030 a la 007050, en cuyo momento finaliza la ejecución del programa.

La declaración READ con la opción AT END es necesaria especificarla cuando se acceda secuencialmente a los registros de un archivo, es decir, en archivos contenidos en ficha o cinta, en los que para leer, por ejemplo, el registro que ocupa el lugar vigesimo en el archivo, hay antes que haber leido los diecinueve que le preceden. Sin embargo, es opcional y por lo tanto puede omitirse la palabra AT END cuando se accede aleatoriamente a los registros del archivo, como es el caso de archivos contenidos en disco (ver formato 2 de la declaración READ en la página siguiente).

La opción INTO identificador hace que la declaración READ, además de dejar disponible un registro lógico en el área de entrada, transfiera sus datos al área reservada por Identificador, es decir, equivale a una declaración READ seguida de una declaración MOVE. Identificador ha de ser el nombre de un registro descrito en la WORKING–STORAGE SECTION o el nombre de un registro de un área de salida.

El sistema operativo controla las operaciones de entrada que comporta la ejecución de declaración READ. Dicho control, a título informátivo, lo realiza de la forma siguiente:

- Si los registros no están bloqueados, cada declaración READ introduce en el área de entrada un registro lógico, dejándolo preparado para su proceso.

- Si los registros están bloqueados, la primera declaración READ introduce en el área de entrada el bloque de registros, dejando disponible el primer registro lógico para su proceso; la siguiente READ deja disponible para su proceso el siguiente registro y así sucesivamente hasta que se procese el último registro lógico del bloque, momento en el cual la siguiente declaración READ leerá el siguiente bloque con el que realizará el ciclo citado.

FORMATO 2:

READ nombre–de–archivo RECORD [**INTO** identificador]

INVALID KEY declaración–imperativa

Este formato se emplea cuando para la lectura de los registros de un archivo se accede aleatoriamente; es decir, cuando se ha especificado en el programa: ACCESS IS RANDOM lo cual exige que los archivos hayan sido grabados en disco , tambor, etc., y organizados en forma secuencial indexada o directa.

La opción INVALID obliga al Sistema, antes de llevar a cabo la lectura de un registro para ser introducido en la memoria del ordenador, a comprobar si la clave o código del registro es válida o inválida.

La clave se considera válida cuando en el archivo en que se va a leer, existe grabado un registro que contiene dicha clave; en cuyo caso el Sistema lee el registro y continúa la ejecución del programa a partir del punto que sigue a la declaración imperativa especificada después de INVALID.

La clave se considera inválida cuando en el archivo no existe un registro que contenga dicha clave o porque en la dirección (número de la pista) donde se le ordenó al Sistema que hiciera la lectura, no se encuentra dicho registro. En ambos casos, no lo lee y ejecuta la declaración imperativa que sigue a INVALID.

En el ejemplo siguiente (tomado del programa de la página 301):

```
002090          READ ARCHIDISC  INVALID KEY DISPLAY CODIGO '= CLAVE INVALI
002100-             'DA    GO TO LEE.
002120          MOVE REGIS  TO CINT
```

si la clave es válida, lee un registro del archivo ARCHIDISC y pasa a ejecutar la declaración MOVE REGIS TO CINT. Por el contrario, si la clave es inválida, no lee el registro y ejecuta DISPLAY CODIGO ..... GO TO LEE. (Más información en el capítulo VII) .

7.3.- Declaración WRITE

FUNCION: Proporcionar la escritura de registros en un archivo en el que se ha especificado la declaración OPEN OUTPUT, controlando el espaciado vertical de cada registro en cada página.

FORMATO 1:

```
WRITE nombre-de-registro [ FROM identificador-1 ]

[ { BEFORE } ADVANCING { identificador-2 LINES }                    ]
[ { AFTER  }           { entero LINES        }                    ]
[                      { nombre-nemotécnico  }                    ]

[ AT { END-OF-PAGE } declaración-imperativa ]
[    { EOP         }                        ]
```

- Este formato se emplea en archivos cuyos registros se han de imprimir (listados), no siendo válido para archivos cuyos registros se han de grabar en cintas o discos.

- Proporciona al programador un medio para controlar el espaciado en la impresión de los registros dentro de cada página.

- Una declaración WRITE con la opción FROM identificador-1, transfiere el contenido del campo denominado por identificador-1 al área de salida denominada por nombre-de-registro y lo escribe. Es decir, equivale a una declaración MOVE seguida de una WRITE:

```
006090 ESCRIBIR-1.
006100     WRITE IMPRESO FROM CABECERA-1 AFTER ADVANCING PAG-NUEVA
006110     DISPLAY '
006120-    '-------------------------------------------------------'.
```

La declaración WRITE de la línea 006100 equivale a:

```
006090 ESCRIBIR-1.
006100     MOVE CABECERA-1 TO IMPRESO
006101     WRITE IMPRESO AFTER PAG-NUEVA
006110     .....................................................
```

Si se emplea la opción AFTER ADVANCING entero, se obliga a avanzar el papel de la impresora el número de líneas que se haya especificado en el valor de Entero; escribiendo posteriormente un registro. Los valores que puede tomar Entero dependen del compilador empleado, normalmente están comprendidos entre 0 y 99:

```
   030     WRITE IMPRESO AFTER 5 LINES
```

Si a Entero se le asigna el valor 0 (cero) se produce una sobreimpresión.

Cuando el programador desee que la impresora, una vez haya escrito un número determinado de líneas en cada página, salte a la página siguiente para comenzar a escribir en la misma posición que lo hizo en la página anterior, debe emplear la opción:

AFTER ADVANCING nombre - nemotécnico

Nombre–nemotécnico debe definirse en el párrafo SPECIAL–NAMES de la ENVIRONMENT DIVISION, a fin de asociarlo al canal 1 de la cinta de control de carro (22) de la impresora cuyo nombre–de–función es: C01, lo cual obliga a saltar al papel a la página siguiente, deteniéndose cuando la célula que controla a dicho canal detecte la perforación realizada en él por el programador. Esta opción permite además del salto a la página siguiente, el salto dentro de la misma página hasta llegar a la posición donde encuentre las perforaciones, que se hayan realizado, en los canales 2 a 11 de la cinta de control de carro.

EJEMPLO:

```
001150     SPECIAL-NAMES.
001160         C01   IS   PAG-NUEVA
001170         DECIMAL-POINT  IS  COMMA.

006090 ESCRIBIR-1.
006100     WRITE IMPRESO   FROM  CABECERA-1  AFTER ADVANCING PAG-NUEVA
006110     DISPLAY
```

En el ejemplo, el nombre–nemotécnico es PAG–NUEVA, asociado al nombre–de–función C01.

Los nombres–de–función que pueden utilizarse y su acción es la siguiente:

Nombre–de–Función	Acción
CPS	Supresión del espaciado
C01 a C12	Salto a canal 1 a 12 respectivamente.

Si se emplea la opción AFTER ADVANCING identificador–2, éste debe estar descrito dentro de la DATA DIVISION como campo elemental numérico y entero pudiendo tomar valores comprendidos entre 0 y 99, en cuyo caso la página de la impresora avanzará tantas líneas como indique el valor que contenga Identificador–2 en el momento de ejecutarse la declaración WRITE.

(22) Actualmente, aunque los nuevos modelos de impresoras prescinden de la cinta de control de carro, sin embargo, el control del salto y espaciado sigue haciéndose de la forma especificada, aunque el funcionamiento técnico de los mismos sea diferente.

EJEMPLO :

```
020  FD  ARCHI-IMPRESO  LABEL  RECORD  OMITTED.
030  01  IMPRESO              PIC X(133).
.....
050  WORKING-STORAGE SECTION.
060  77  SALTO                PIC 99.
```

Con el empleo del identificador –SALTO–, antes de cada declaración WRITE se debe introducir en él, mediante una declaración de movimiento, un valor que represente el número de líneas que se desean saltar. Así, si se quiere que el papel de la impresora avance 10 líneas y escriba el registro IMPRESO, las declaraciones necesarias serían:

```
090  PROCEDURE DIVISION.
100  .....
110      MOVE  10  TO  SALTO
120      WRITE  IMPRESO  AFTER  SALTO .....
```

Así como en la opción AFTER ADVANCING la función de la impresora es avanzar el impreso el número de líneas deseado y a continuación escribir el registro, sin embargo con la opción BEFORE ADVANCING, primero escribe el registro y después avanza el impreso el número de líneas especificado en Entero, identificador–2 ó en nombre–nemotécnico.

Opción END–OF–PAGE.

END–OF–PAGE es una condición que se cumple cuando al ir la impresora escribiendo línea tras línea, ésta detecte la perforación del canal 12 de la cinta de control de carro (dicha perforación es indicativa de que no deben escribirse más líneas en dicha página), en cuyo caso ni imprime ni espacía hasta despues de ejecutar la declaración imperativa que sigue a EOP.

Para poder detectar con certeza la perforación en dicho canal, debe ordenarse avanzar la impresora una línea antes o después de imprimir (AFTER ó BEFORE ADVANCING 1 LINES) o bien el programador hará el cálculo de líneas que debe avanzar para que pueda detectar la perforación, ya que, de no localizarla, los resultados serían imprevisibles.

FORMATO 2:

WRITE nombre–de–registro [**FROM** identificador]

Este formato se emplea para grabar registros en cinta magnética, disco y tambor siempre que dicha grabación se haga con una organización secuencial.

Si se desea crear un archivo con registros no agrupados en bloque, cada declaración WRITE grabará en el archivo el registro que en ese momento se encuentre en el área de salida, pero si se desea que los registros se graben agrupados en bloques, la primera declaración WRITE coloca un registro en el área de salida pero no se graba en el archivo hasta que no se haya completado el llenado de dicha área por la ejecución de las siguientes declaraciones WRITE, en cuyo momento,

la declaración WRITE que completó el llenado de dicha área graba el bloque de registros. Si al llegar al final de la ejecución del programa no se ha completado el llenado del área de salida, en este caso, se escriben los registros que en dicha área se encuentren:

```
008160          WRITE REGISTRO-OPERARIO
008170          ADD 1 TO CONTADOR
```

En este ejemplo cada declaración WRITE introduce un registro en el área de salida denominada REGISTRO–OPERARIO y cada diez declaraciones WRITE completarán el llenado de dicha área, por haberlo así especificado el programador en la cláusula BLOCK 10 RECORDS, siendo éste el momento en que se grabarán en la cinta dichos registros.

La función de FROM en este formato es la misma que en la del formato 1.

Una vez ejecutada una declaración WRITE no queda disponible para ser procesada la información contenida en nombre–de–registro.

FORMATO 3:

WRITE nombre–de–registro [**FROM** identificador]

INVALID KEY declaración–imperativa

Este formato se emplea para grabar archivos en disco, tambor, etc. ya sea en organización secuencial normal, secuencial indexada o directa.

La opción INVALID obliga al Sistema a comprobar, antes de grabar un registro, la validez de la clave o código del registro. Si la clave es válida el Sistema graba el registro y continúa la secuencia de ejecución del programa a partir del punto que sigue a la declaración imperativa. Por el contrario, si la clave es inválida, no graba el registro y ejecuta la declaración imperativa que sigue a INVALID.

En el siguiente ejemplo (tomado del programa de la página 288):

```
001180          WRITE REGISDISC FROM FICHAOPE INVALID KEY DISPLAY 'NO CABEN
001190-         'TODOS LOS REGISTROS EN EL ARCHIVO' GO TO FIN. GO TO LEE.
```

si la clave es válida ejecuta WRITE REGISDISC FROM FICHAOPE y continúa en la declaración GO TO LEE; si la clave es inválida omite WRITE REGISDIS FROM FICHAOPE y pasaría a ejecutar la declaración DISPLAY 'NO CABEN .....' GO TO FIN.

Es, prácticamente, una forma de operar similar a una declaración IF.

En el Capítulo VII se indican los motivos de invalidez de las claves, según el tipo de organización empleado para crear el archivo.

7.4.-Declaración CLOSE

FUNCION: Finalizar el proceso de uno o varios archivos.

FORMATO:

```
CLOSE nombre-de-archivo [REEL | UNIT] [WITH { NO REWIND | LOCK }] ...
```

Cada archivo de entrada o salida que se haya abierto mediante una declaración OPEN debe cerrarse con una declaración CLOSE, pudiendo una sola declaración CLOSE cerrar varios archivos a la vez.

Un archivo puede, dentro de un mismo programa, cerrarse y abrirse más de una vez.

EJEMPLO:

```
007020     READ ARCHIVO-OPERARIOS; AT END,
007030     DISPLAY '                REGISTROS PROCESADOS =' CONTADOR,
007040 --> CLOSE  ARCHIVO-OPERARIOS CINTA-NOMINA ARCHI-IMPRESO,
007050     STOP RUN.
```

En este ejemplo, se han cerrado a la vez los tres archivos que previamente se abrieron con una declaración OPEN, ya que los tres han finalizado su proceso en el mismo momento, pero hay casos en que su final no es simultáneo, por lo que puede emplearse una declaración CLOSE para cada uno, para ir cerrándolos a medida que acabe el proceso cada uno de ellos.

La opción UNIT se emplea en archivos de acceso directo (disco, tambor, etc.). La opción REEL se especifica en aquellos archivos contenidos en cinta magnética que ocupen más de un carrete.

La opción NO REWIND está permitida en archivos asignados a unidades de cinta magnética. Si no se especifica dicha opción se rebobina el carrete hasta que las cabezas de lectura / escritura se posicionen al principio del archivo. Si se especifica dicha opción, no se realiza la reposición (rebobinado) al principio del mismo. En la práctica es poco usada esta opción, sin embargo se emplea cuando, una vez finalizada la grabación de un archivo, se quiere grabar a continuación otro archivo en dicha cinta.

Si se especifica la opción LOCK, al cerrarse el archivo, el carrete de cinta se rebobina y queda protegido de forma tal que no puede utilizarse de nuevo si antes el operador no ejecuta una acción especial en la Unidad de cinta, que consiste en pulsar una tecla que obliga a descender las cabezas de lectura/escritura de la Unidad, para permitir de nuevo su lectura.

7.5.- Declaración ACCEPT

FUNCION: Introducir pequeños volumenes de datos en la memoria del ordenador desde la máquina de escribir de la consola, o cualquier otro dispositivo de entrada.

FORMATO:

$$\underline{\text{ACCEPT}}\ \text{identificador}\left[\underline{\text{FROM}}\ \left\{\begin{array}{l}\underline{\text{SYSIPT}}\\ \underline{\text{CONSOLE}}\\ \text{nombre-nemotécnico}\end{array}\right\}\right] \qquad (21)$$

La entrada de datos en la memoria a través de la consola u otro dispositivo de lectura lenta debe restringirse a los más imprescindible, debido a su lentitud, y por consiguiente, mal aprovechamiento del ordenador; sin embargo, en el ejemplo NOMINA se ha utilizado esta declaración para introducir la fecha del día en que se ejecutará la NOMINA sólo con fines didácticos, ya que podría haberse resuelto de otro modo más rápido la entrada de dicha fecha, no utilizando la opción FROM.

Teniendo en cuenta las descripciones:

```
005120     02  FECHA.
005130         03  D                    PICTURE  99.
005140         03  FILLER               PICTURE  X.
005150         03  M                    PICTURE  99.
005160         03  FILLER               PICTURE  X.
005170         03  A                    PICTURE  99.
```

```
006060     ACCEPT  FECHA  FROM  CONSOLE.
```

El campo FECHA es el Identificador y, por haberse especificado la opción FROM CONSOLE, el ordenador al ejecutar la declaración ACCEPT, se detiene para dar tiempo al operador para que teclee los datos a introducir en la memoria. Una vez que el operador ha tecleado los datos solicitados, pulsa una tecla de dicha máquina y los datos pasarán al área de la memoria denominada en Indentificador (FECHA), es decir, tecleará 01/04/78, (fecha en que se va a obtener la nómina) situándose 01 dentro del campo D, el primer / dentro del primer campo FILLER, 04 dentro del campo M y así sucesivamente; continuando la ejecución del programa en la siguiente instrucción.

- Si no se especifica la parte opcional FROM, los datos se introducirán en la memoria por lectura de los mismos desde el dispositivo de entrada lógica del Sistema, (SYSIPT), que normalmente es la lectora de fichas. Es decir, si en el programa NOMINA se hubiera especificado:

```
006060     ACCEPT  FECHA
```

(21) Existen tipos de ordenadores que emplean las opciones TERMINAL, DATE-TIME, etc. cuya función se describe en sus respectivos manuales.

el programador debería colocar, a continuación de las fichas que constituyen el programa fuente y delante de las que contienen los datos de cada operario, una ficha cuyo contenido fuera 01/04/78 y, al ejecutarse la declaración ACCEPT, la lectora leería dicha ficha pasando sus datos al campo de la memoria denominado en Identificador, es decir, al campo FECHA.

Esta opción es más recomendable para la entrada de los datos que la anterior por ser más rápida, al evitar tener que emplear tiempo en teclear en la máquina de la consola.

El número de caracteres que pueden transferirse a un campo de la memoria, aunque es limitado, su capacidad excede de cuatro mil.

Si el tamaño del campo en el que han de introducirse los datos, es mayor que el número de caracteres que el ordenador puede introducir en una operación física (lectura de una ficha, escritura de una línea en la consola etc.), una sola declaración ACCEPT efectuará las operaciones necesarias hasta completar su llenado.

El siguiente ejemplo muestra un programa en el que una sola declaración ACCEPT, ejecuta cuatro operaciones de lectura, con lo cual lee las cuatro primeras fichas de datos que se supone se encuentran colocadas a continuación de las fichas que constituyen el programa e introduce su contenido en la memoria, a fin de completar el llenado del campo denominado DATOS. Posteriormente, este programa, por medio de una declaración DISPLAY, escribe por la impresora el contenido de los 320 caracteres que introdujo en el citado campo:

```
010    ID DIVISION.
020    PROGRAM-ID. ACCEDISP.
030    REMARKS. EJEMPLO DE DECLARACIONES ACCEPT Y DISPLAY.
040  ENVIRONMENT DIVISION.
050  DATA DIVISION.
060    WORKING-STORAGE SECTION.
070    77  DATOS  PIC X(320).
080  PROCEDURE DIVISION.
090  1. ACCEPT DATOS
100       DISPLAY DATOS
110       STOP RUN.
```

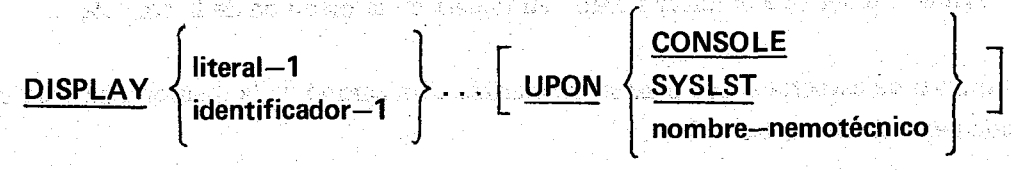
– Identificador, puede ser un campo compuesto o un campo elemental con PICTURE de tipo alfabético, alfanumérico, decimal externo o de coma flotante.

– Nombre–nemotécnico puede asociarse a la opción CONSOLE o a SYSIPT, haciéndose dicha asociación en el párrafo SPECIAL–NAMES; así, en el ejemplo descrito en la página 84, CON era el nombre–nemotécnico que debía utilizarse en la declaración ACCEPT en vez de CONSOLE, cuando se introducían los datos por la consola. Sin embargo, utilizar el párrafo SPECIAL–NAMES con el único objeto de sustituir la palabra CONSOLE por CON no es la utilización más idónea, ya que, según el formato, puede especificarse la palabra CONSOLE dentro de la declaración ACCEPT sin tener que realizar dicha asociación en el párrafo SPECIAL–NAMES.

7.6.-Declaración DISPLAY

FUNCION: Escribir información (datos y/o resultados) o mensajes por la máquina de escribir de la consola, impresora u otro dispositivo de salida.

FORMATO:



- El número de identificadores y/o literales que se pueden especificar a continuación de DISPLAY es variable dependiendo del compilador empleado.
- Literal–1 puede ser de tipo no numérico o cualquier constante figurativa, excepto ALL.
- Nombre–nemotécnico debe especificarse dentro del párrafo SPECIAL–NAMES, pudiéndose asociar a las palabras CONSOLE y SYSLST (impresora).
- Si se especifica en una declaración DISPLAY la opción UPON CONSOLE, ésta hace que se escriba en la máquina de escribir de la consola,y en el mismo orden en que se han especificado, el contenido que en el momento de ejecutar dicha declaración exista en los identificadores y/o literales. Si la suma de las posiciones que ocupa dicho contenido excede al de caracteres que caben en una línea del papel de la máquina de la consola, se continúa en la línea siguiente. El número de caracteres que puede el ordenador escribir en una línea del impreso de la máquina de la consola es variable según el tipo de consola empleado (ver nota de la pág. 34).

El efecto de la declaración DISPLAY del ejemplo siguiente:

```
140     DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA'
150         '    ' QUOTE RICOSA QUOTE SPACE SPACE
160         CODIGO '  ' SUELDO UPON CONSOLE.
```

haría que se escribiese por la consola el mensaje siguiente :

```
MARQUE LA FECHA DE HOY SEGUN FORMATO DD/MM/AA  'RICOSA'  00001  20000
```

suponiendo que el contenido del campo CODIGO es 00001 y el del campo SUELDO es 20.000 .

Cuando se utilice la opción UPON SYSLST, el contenido de los identificadores y/o literales que figuran en la declaración se escribirán por la impresora, por lo que en cada línea se escribirá un número de caracteres dependiente del tipo de impresora, normalmente un valor comprendido entre 120 y 150 caracteres; si el contenido de dichos identificadores no cabe en una línea se continúa en la siguiente.

8.- VERBOS de BIFURCACION

8.1.- Declaración GO

FUNCION: Proporcionar un medio de ruptura (bifurcación) a la forma secuencial en que se ejecutan las declaraciones del programa.

FORMATO 1:

GO TO [nombre–de–procedimiento]

Las declaraciones que se describen en la PROCEDURE DIVISION se van ejecutando en la secuencia en que se han especificado, siempre que no exista entre ellas alguna declaración GO que obligue a no seguir dicho orden secuencial y bifurcar a un determinado punto (procedimiento) del programa, continuando la ejecución del mismo a partir de la primera declaración del procedimiento al cual se bifurcó.

EJEMPLO:

```
007010  LEER.
007020      READ ARCHIVO-OPERARIOS; AT END,
007030      DISPLAY ' REGISTROS PROCESADOS = ' CONTADOR,

008170      ADD 1 TO CONTADOR
008180      GO TO LEER.
```

La declaración GO de la línea 008180 obliga a bifurcar al procedimiento LEER (línea 007010), continuando la ejecución del programa en la declaración READ.

Si el programador no especifica ningún nombre–de–procedimiento a continuación de GO TO, el programa se detiene, finalizando anormalmente su ejecución, ya que no se le indica en que punto del mismo debe continuar. No obstante, se puede evitar este final anormal, utilizando GO TO en combinación con una declaración ALTER en la cual se indique hacia qué procedimiento debe bifurcar (ver ejemplo de la pág. 216).

FORMATO 2:

GO TO nombre–de–procedimiento–1 [nombre–de–procedimiento–2] . . .

DEPENDING ON identificador

Cuando se utiliza este formato,la secuencia de ejecución del programa bifurca al nombre–de –procedimiento–1, siempre que el valor que exista en Identificador en el momento de ejecutar la declaración sea igual a 1; bifurcará al nombre–de–procedimiento–2 cuando el valor de Identificador sea igual a 2 y así sucesivamente, es decir, la bifurcación a uno u otro nombre-de-procedimiento de los descritos a continuación de GO TO depende (DEPENDING ON) del valor que en ese momento contenga el campo Identificador.

Identificador debe ser un campo elemental con PICTURE de tipo numérico y entero. Los valores que puede contener dicho campo dependen del compilador empleado, no excediendo su longitud de cuatro cifras.

En el programa NOMINA pueden sustituirse,por realizar el mismo trabajo,las declaraciones de las líneas 007110 a la 007150,haciendo las siguientes modificaciones:

```
007110  CALCULO-PLUSF.
007120          GO TO ALFA  BETA  GAMM  DEPENDING ON ESTADO
007125          DISPLAY 'ERROR EN ESTADO CIVIL' FICHA-OPERARIO
007126          STOP RUN.
007130  BETA.
007135          COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007140          GO TO 10.
007145  GAMM.
007146          COMPUTE PLUSF = NUM-HIJOS * 250
007147          GO TO 10.
007148  ALFA.
007150          MOVE 0 TO PLUSF.
```

Si en el campo ESTADO se hallara la cifra 1, bifurcaría la secuencia de ejecución al procedimiento ALFA; si encontrara la cifra 2 lo haría al procedimiento BETA, y a GAMM si encontrara la cifra tres. Si por error, el contenido del campo ESTADO fuera cifra distinta de 1, 2 ó 3, por ejemplo: cero, cuatro, etc., la secuencia de ejecución del programa omitiría la declaración GO, continuando en la declaración siguiente en secuencia, es decir, DISPLAY 'ERROR EN ESTADO...' terminando su ejecución, ya que a continuación se ha especificado la declaración STOP RUN.

Actualmente, el uso de la declaración GO es muy controvertido, existiendo partidarios del estilo tradicional de programación en el que no se pone limitación a su empleo, y partidarios del estilo de programación, denominado estructurado, en el que se pretende minimizar el empleo de dicha declaración (con cualquiera de sus dos formatos).

Cada estilo de programación tiene sus ventajas e inconvenientes, no entran en competencia, y sí sólo intentan que los programas sean más fáciles de leer, corregir y mantener.

8.2.-Declaración ALTER

FUNCION: Modificar y/o indicar el punto de bifurcación de una declaración GO.

FORMATO:

ALTER nombre–de–procedimiento–1 TO [PROCEED TO] nombre–de–procedimiento–2

[nombre–de–procedimiento–3 TO [PROCEED TO] nombre–de–procedimiento–4]...

Una declaración GO, no siempre obliga a bifurcar incondicionalmente hacia el procedimiento que se especifica en dicha declaración, ya que la declaración ALTER puede impedir la bifurcación hacia el citado procedimiento, obligando a bifurcar hacia al procedimiento o procedimientos especificados en esta declaración; es decir, a nombre–de–procedimiento–2, nombre–de–procedimiento–4, etc.

```
010   A.
020   B. ALTER D TO PROCEED TO G.
030   C.
040   D. GO TO F.
050   E.
060   F. CLOSE ARCHIDISC STOP RUN.
070   G. ADD 1 TO CONTA.
080   H. MOVE ALFA TO BETA GO TO E.
090   P.
100
```

En el ejemplo precedente, la declaración ALTER obliga a alterar el punto de bifurcación especificado en la declaración GO del procedimiento D, de tal forma que, cuando la secuencia de ejecución del programa llegue a dicho procedimiento, en lugar de bifurcar hacia el procedimiento F como indica la declaración GO, bifurcará hacia el procedimiento G. Es decir, el orden en que se ejecutarán los procedimientos será el siguiente: A, B, C, D, G, H, E y F, terminando la ejecución del programa al ejecutar la declaración STOP RUN.

– Una declaración GO en la que haya sido modificado su punto de bifurcación por una declaración ALTER, siempre bifurcará al mismo punto mientras no exista otra declaración ALTER que vuelva a modificarlo:

```
010   A.
020   B. ALTER D TO PROCEED TO G.
030   C.
040   D. GO TO F.
050   E.
060   F. CLOSE ARCHIDISC STOP RUN.
070   G. ADD 1 TO CONTA.
080   H. MOVE ALFA TO BETA GO TO C.
090   P.
```

En este ejemplo, el orden en que se ejecutarían los procedimientos sería:

A, B, C, D, G, H, C, D y al llegar a este punto la declaración GO como no ha vuelto a ser modificada por otra ALTER, la secuencia regresaría para volver a ejecutar G, H, C, D, entrando en este bucle, el cual lo realizaría un número indeterminado de veces.

— Una misma declaración ALTER puede modificar a varias declaraciones GO en las que se especifique el nombre del procedimiento al que deben bifurcar incondicionalmente, o también indicar el procedimiento al que deben bifurcar a aquellas declaraciones GO en que no se especifique el nombre del procedimiento:

```
100
110  A. .....
120  B. ALTER D TO G I TO E.
130  C. .....
140  D. GO TO F.
150  E. .....
160  F. CLOSE ARCHIDISC STOP RUN.
170  G. ADD 1 TO CONTA.
180  H. MOVE ALFA TO BETA.
190  I. GO TO.
200  P. .....
```

En este caso, el orden en que ejecutarían los procedimientos sería: A, B, C, D, G, H, I, E y F. Al ejecutar la declaración STOP RUN de este procedimiento el ordenador finalizaría la ejecución del programa.

Si no se hubiese especificado en la declaración ALTER del ejemplo anterior:

I TO E o lo que es lo mismo: I TO PROCEED TO E

los procedimientos se ejecutarían en el siguiente orden: A, B, C, D, G, H, I, en este punto terminaría anormalmente la ejecución del programa, por especificarse una declaración GO en la que no se indica el nombre del procedimiento al que se ha de bifurcar, ni haber sido activada por una declaración ALTER.

Observando los dos ejemplos precedentes, toda declaración GO que vaya a ser activada o modificada por una declaración ALTER, debe ser la única declaración que se especifique dentro del procedimiento. Una vez activada una declaración GO debe utilizarse otra declaración ALTER para desactivarla, a fin de que dicha declaración GO obligue a bifurcar al procedimiento que en ella se indica.

Tal como realiza su función la declaración ALTER, se puede deducir que, en vez de ayudar al programador a seguir la secuencia de ejecución de las declaraciones que forman el programa, complica dicho seguimiento; por lo que dicha declaración está en desuso.

No obstante, se ha descrito la declaración ALTER para conocimiento de aquellos programadores que hayan de mantener programas en los que esté incluída dicha declaración.

8.3.- Declaración PERFORM

FUNCION: Alterar el orden secuencial de ejecución del programa para ejecutar un procedimiento o grupo de procedimientos un número específico de veces (TIMES) o hasta que se cumpla una determinada condición (UNTIL), regresando la secuencia, una vez ejecutados dichos procedimientos, a la declaración que sigue a PERFORM.

FORMATO 1 :

PERFORM nombre–de–procedimiento–1 [THRU nombre–de–procedimiento–2]

– Si no se especifica THRU nombre–de–procedimiento–2, la declaración PERFORM hace que se ejecuten, una vez, las declaraciones contenidas en el procedimiento denominado por nombre–de–procedimiento–1 y cuando haya ejecutado la última declaración de dicho procedimiento, la secuencia de ejecución del programa continúa en la declaración escrita después de PERFORM. Así, en el ejemplo siguiente, la declaración PERFORM ESCRIBIR–2 hace que la secuencia de ejecución bifurque a realizar las declaraciones especificadas en el procedimiento ESCRIBIR–2 (línea 006130), y una vez finalizada su ejecución continuaría en la declaración MOVE ZERO TO CONTALINEAS:

```
008120          IF  CONTALINEAS = 24
008125              WRITE IMPRESO  AFTER ADVANCING  PAG-NUEVA
008130              PERFORM  ESCRIBIR-2
008135              MOVE ZERO  TO  CONTALINEAS.
008140          WRITE IMPRESO  FROM  IMPRESO-S  AFTER ADVANCING 2 LINES
```

La declaración PERFORM obliga a ejecutar las declaraciones contenidas en el procedimiento ESCRIBIR–2, por lo que ahorra tener que repetir dentro del programa las mismas declaraciones que se especificaron en dicho procedimiento. De no haber empleado la declaración PERFORM, se debería haber escrito en el programa NOMINA :

```
008120          IF  CONTALINEAS = 24
008125              WRITE IMPRESO  AFTER ADVANCING  PAG-NUEVA
008126              ADD 1  TO  NUM-HOJA
008127              WRITE IMPRESO  FROM  CABECERA-2  AFTER ADVANCING 2
008128* LA SIGUIENTE DECLARACION HACE QUE SE IMPRIMA UNA LINEA EN BLANCO
008129              DISPLAY ' '
008130              DISPLAY '                    CODIGO                APELLIDOS  Y  NOMB
008131-             'RE                  SUELDO   TRIENIOS  HORAS-EX.  PLUS
008132-             '-F. DESCUEN.   TOTAL'
008133              DISPLAY '                    -------               ----------
008134-             '------              -------   --------  ---------  ----
008135-             '---  ----------     -------'
008137              MOVE ZERO  TO  CONTALINEAS.
008140          WRITE IMPRESO  FROM  IMPRESO-S  AFTER ADVANCING 2 LINES
```

Hay casos en que la secuencia no regresa a la siguiente declaración a PERFORM; esto sucede cuando entre las declaraciones que forman parte de nombre–de–procedimiento–1, exista alguna declaración GO que obligue a bifurcar hacia otro procedimiento. En este caso, la secuencia de ejecución del programa no regresará a la siguiente declaración a PERFORM, sino que continuará en el procedimiento ordenado por la declaración GO.

EJEMPLO:

```
120 ALFA.
130         PERFORM  DELTA
140         COMPUTE  TRIEN = NUMTRIEN * 1000
...         ..........................................
200 DELTA.
210         MOVE FICHA TO DATOSC
220         IF TOTAL = 12
240             GO TO BETA
250         ELSE ADD 1 TO TOTAL
260              WRITE IMPRESO FROM CABECERA AFTER 1.
270 BETA.
280         MOVE DECIM TO CAMPOA
```

Este ejemplo comienza con la ejecución de la declaración PERFORM, que obliga a bifurcar hacia el procedimiento DELTA. En dicho procedimiento se encuentra con la declaración IF TOTAL = 12, la cual al probarse puede suceder que el resultado de la prueba sea cierto, en cuyo caso, ejecutaría la declaración GO TO BETA que obliga a bifurcar a dicho procedimiento, lo cual supone que la secuencia de ejecución del programa no regrese a la declaración de la línea 140, por haber salido, la secuencia de ejecución, del procedimiento DELTA.

Caso de que la prueba IF TOTAL = 12 fuera falsa, ejecutaría las declaraciones ADD 1 TO TOTAL y WRITE IMPRESO FROM CABECERA AFTER 1, regresando a ejecutar la declaración de la línea 140.

Puede también suceder que la última declaración contenida en el procedimiento donde obliga a bifurcar una PERFORM sea una declaración GO; en este caso, aunque se ejecute dicha declaración y sea esta la última del procedimiento, sin embargo la secuencia tampoco regresará a la siguiente declaración a la PERFORM, sino que bifurcará al procedimiento donde le ordene la declaración GO.

— Si se especifica THRU nombre–de–procedimiento–2, ejecuta las declaraciones contenidas desde el comienzo del procedimiento denominado por nombre–de–procedimiento–1 hasta el final de nombre–de–procedimiento–2, regresando a la declaración siguiente a PERFORM en cuanto ejecute la última declaración contenida en nombre–de–procedimiento–2, siempre que ésta no sea una declaración GO.

Es importante tener en cuenta que no es obligatorio el que tengan que ejecutarse todas las declaraciones comprendidas entre nombre –de–procedimiento–1 y nombre–de–proce-

dimiento–2, pues puede haber declaraciones como GO que permiten omitir la ejecución de procedimientos completos comprendidos entre nombre–de–procedimiento–1 y nombre–de–procedimiento–2, pero sí siempre ha de ejecutar la última declaración de nombre–de–procedimiento–2 para poder regresar a la declaración especificada después de PERFORM.

EJEMPLO:

```
020   F.  PERFORM  B  THRU  D   WRITE IMPRESO AFTER 2
030       MOVE  ZERO  TO  TOTAL.
040   A.  MOVE  CAMPOA  TO  CAMPOB   ADD  3  TO  ALFA
050       ------------------------------------------.
060   B.  IF  ALFA > 5  GO  TO  D
070       ELSE MOVE SUELDO  TO SUELDO-S.
080   C.  WRITE  CINTASAL  FROM  CINTAENT  ----------
090   D.  DISPLAY  CAMPOB
100       MOVE  'FIN'  TO  CAMPOC.
110   E.  ------------------------------------------
```

En este ejemplo, al ejecutar la declaración PERFORM obliga a bifurcar al procedimiento B, en el cual al probar la condición ALFA > 5 será o no cierta; si ALFA es mayor que 5 se cumple la condición, y por tanto la secuencia del programa omitiría las declaraciones del procedimiento C, continuando en la declaraciones contenidas en el procedimiento D y, una vez ejecutada la última declaración de dicho procedimiento MOVE 'FIN' TO CAMPOC, pasaría a ejecutar WRITE IMPRESO AFTER 2. Si ALFA no es mayor que 5, no se cumple la condición y, por lo tanto, la secuencia del programa ejecutaría las declaraciones que siguen a ELSE, las contenidas en los procedimientos C y D, regresando a la declaración WRITE IMPRESO AFTER 2.

Las declaraciones comprendidas entre el comienzo del procedimiento denominado por nombre–de–procedimiento–1 y el final de nombre–de–procedimiento–2, forman el ámbito o dominio de la PERFORM.

FORMATO 2 :

PERFORM nombre–de–procedimiento–1 [**THRU** nombre–de–procedimiento–2]

{ entero / Identificador } **TIMES**

Esta opción hace que se ejecuten las declaraciones contenidas en un procedimiento (cuando no se especifica THRU nombre de procedimiento–2) o en un grupo de procedimientos (si se especifica THRU), el número de veces especificado por Identificador o por el entero, continuando la secuencia de ejecución de programa en la declaración que sigue a PERFORM.

EJEMPLO.

Utilizando la IDENTIFICATION, ENVIRONMENT y DATA DIVISION del programa NOMINA, la PROCEDURE siguiente, sirve para listar por la impresora el código y el nombre de los diez primeros operarios de la empresa RICOSA.

```
020 PROCEDURE DIVISION.
030     A. OPEN INPUT ARCHIVO-OPERARIOS  OUTPUT ARCHI-IMPRESO
040        MOVE SPACES  TO  IMPRESO-S.
050     B. READ ARCHIVO-OPERARIOS  AT END  GO TO E.
060        MOVE CODIGO  TO  CODIGO-S
070        MOVE APELL-NOMBRE  TO  APE-NOMBRE.
080     C. WRITE IMPRESO FROM  IMPRESO-S  AFTER 2.
090     D. PERFORM  B  THRU  C  9  TIMES.
100     E. CLOSE ARCHIVO-OPERARIOS  ARCHI-IMPRESO
110        STOP RUN.
```

Antes de que llegue la secuencia de ejecución del programa a la declaración PERFORM (línea 090), ya se ha listado el código y nombre del primer operario, siendo la declaración PERFORM la que obliga a listar los nueve siguientes.

En la anterior declaración PERFORM se ha empleado la opción entero, pero al mismo resultado se llegaría empleando la opción Identificador:

```
120 PROCEDURE DIVISION.
130     A. OPEN INPUT ARCHIVO-OPERARIOS  OUTPUT ARCHI-IMPRESO
140        MOVE SPACES  TO  IMPRESO-S
150        MOVE  9  TO  CONTADOR.
160     B. READ  ARCHIVO-OPERARIOS  AT END  GO  TO  E.
170        MOVE  CODIGO  TO  CODIGO-S
180        MOVE  APELL-NOMBRE  TO  APE-NOMBRE.
190     C. WRITE  IMPRESO  FROM  IMPRESO-S  AFTER 2.
200     D. PERFORM  B  THRU  C  CONTADOR  TIMES.
210     E. CLOSE ARCHIVO-OPERARIOS  ARCHI-IMPRESO
220        STOP RUN.
```

La única diferencia con el ejemplo anterior, es que previa a la ejecución de la declaración PERFORM es necesario colocar dentro del identificador CONTADOR el valor 9, a fin de que cuando la secuencia llegue a PERFORM, obligue a ejecutar los procedimientos B y C un número de veces igual al valor que exista en ese momento en CONTADOR, es decir, se listarían nueve operarios, los cuales, junto con el listado del primer operario que se hizo antes de la ejecución de PERFORM, forman el listado de los diez primeros operarios.

Tanto si se emplea la opción Entero como Identificador, su valor debe ser un número entero y positivo y su valor máximo depende del tipo de compilador empleado.

Caso de que Entero o Identificador fueran igual a cero o negativos, no se ejecutaría la declaración PERFORM y la secuencia de ejecución del programa continuaría en la declaración siguiente a PERFORM.

FORMATO 3 :

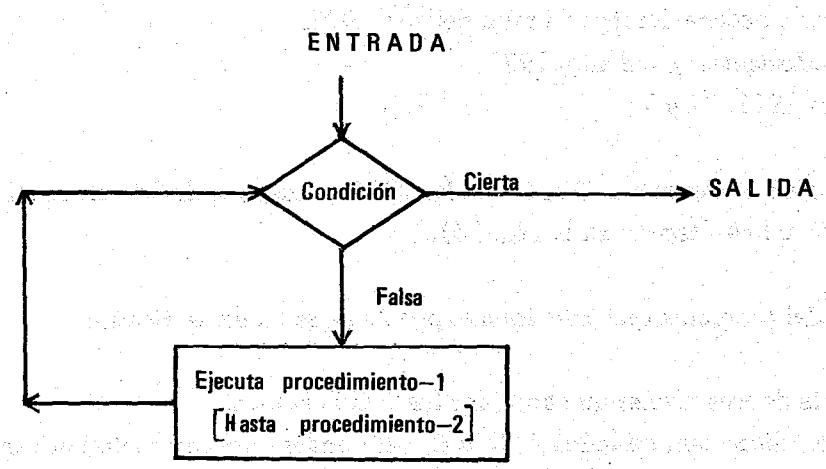
PERFORM nombre–de–procedimiento–1 [**THRU** nombre–de–procedimiento–2]

UNTIL condición

Esta opción opera de la forma siguiente: previamente a la ejecución de los procedimientos comprendidos desde nombre–de–procedimiento–1 a nombre–de–procedimiento–2 se prueba si la condición es cierta, caso negativo, se ejecutan los procedimientos citados; terminada dicha ejecución, se vuelve a probar si la condición es cierta, si el resultado sigue siendo negativo, se vuelven a ejecutar de nuevo dichos procedimientos, y así continúa hasta que al hacer la prueba de dicha condición, ésta es cierta, en cuyo momento dejan de ejecutarse dichos procedimientos y la secuencia de ejecución del programa pasa a la declaración siguiente a PERFORM.

Si al realizar la primera prueba de la condición, resulta ser cierta, en ese caso, no se ejecuta ningún procedimiento y la secuencia de ejecución del programa pasa a la siguiente declaración a PERFORM.

El ordinograma ayuda a comprender cuanto se ha especificado:



El ejemplo que sigue, ejecuta el mismo trabajo que con la PERFORM del Formato 2, para listar los diez primeros operarios del ARCHIVO–OPERARIOS:

```
010  PROCEDURE DIVISION.
020  A.  OPEN INPUT ARCHIVO-OPERARIOS OUTPUT ARCHI-IMPRESO
030      MOVE SPACES TO IMPRESO-S
040      MOVE 0 TO CONTADOR.
050  B.  READ ARCHIVO-OPERARIOS AT END GO TO E.
060      MOVE CODIGO TO CODIGO-S
070      MOVE APELL-NOMBRE TO APE-NOMBRE.
080  C.  WRITE IMPRESO FROM IMPRESO-S AFTER 2
090      ADD 1 TO CONTADOR.
100  D.  PERFORM B THRU C UNTIL CONTADOR = 10.
110  E.  CLOSE ARCHIVO-OPERARIOS ARCHI-IMPRESO
120      STOP RUN.
```

Las declaraciones especificadas de la línea 020 a la línea 090 imprimen el código, nombre y apellidos del primer operario; siendo la declaración PERFORM de la línea 100 la que se encarga de listar los nueve operarios restantes. Efectivamente, el campo CONTADOR, inmediatamente antes de ejecutar la declaración PERFORM, tiene un valor igual a uno, y al ir incrementándose de unidad en unidad, obligado por la declaración de la línea 090, cuando el valor de dicho campo llegue a ser igual a diez, hace que sea cierta la condición y la secuencia de ejecución continuará en la declaración de la línea 110.

Para facilitar la comprensión de los siguientes Formatos de la declaración PERFORM, se expone un ejemplo, y se recomienda el estudio previo de las páginas 239 a la 248.

EJEMPLO.

La sección de Estadística para emitir un informe, necesita conocer el número de automóviles fabricados durante:

- La segunda decena del mes de julio del año 1973.
- El segundo semestre del año 1971.
- Los años 1971, 72 y 73.

Para ello, en el Centro de Cálculo de RICOSA se dispone de la cinta magnética cuyas características se han descrito en la pág. 131.

Dentro del programa que soluciona el problema se ha especificado:

- Una tabla de tres niveles que contiene los 1080 campos.
- Tres subíndices denominados A, B y C, para poder designar cualquier campo de la tabla.
- Un campo ALFA como totalizador del número de automóviles.

```
001010   IDENTIFICATION DIVISION.
001020     PROGRAM-ID.    EJPERFOR.
001030*                  PROGRAMA EJEMPLO DE EMPLEO  DE LA CLAUSULA PERFORM.
001040   ENVIRONMENT DIVISION.
001050     INPUT-OUTPUT SECTION.
001060       FILE-CONTROL.
001070         SELECT CINTA ASSIGN   SYS004-UT-2400-S.
001080   DATA DIVISION.
001090     FILE SECTION.
001100       FD   CINTA    RECORDING F
001110                     LABEL RECORDS OMITTED
001120                     DATA RECORD FABRICACION.
001130       01   FABRICACION.
001140            02 NOMBRE      PIC A(6).
001150            02 MODELO      PIC X(8).
001160            02 ANUAL              OCCURS 3.
001170               03  MENSUAL        OCCURS 12.
001180                   04  DIARIA     OCCURS 30    PIC 999.
001190       WORKING-STORAGE SECTION.
001200         01   ALFA          PIC 9(6)   VALUE ZERO.
001210         01   A             PIC 9      USAGE COMP.
001220         01   B             PIC 99     USAGE COMP.
001230         01   C             PIC 99     USAGE COMP.
001235
002010   PROCEDURE DIVISION.
002020     ABRIR.
002025           OPEN   INPUT CINTA.
002030     LEER.
002035           READ   CINTA  AT END  CLOSE CINTA    STOP RUN.
002038     CALCULO1.
002040           MOVE 3  TO A          MOVE 7  TO  B
002050           PERFORM UNO    VARYING C  FROM 20 BY -1  UNTIL C = 10
002060           DISPLAY `NUMERO DE AUTOMOVILES FABRICADOS DURANTE LA SEGUN
002070-                  `DA DECENA DE JULIO DE 1.973 = ' ALFA.
002075     CALCULO2.
002080           PERFORM DOS           MOVE 1  TO  A
002090           PERFORM UNO    VARYING B  FROM 7 BY 1  UNTIL B = 13
002100                                 AFTER C  FROM 1 BY 1  UNTIL C = 31
002110           DISPLAY `NUMERO DE AUTOMOVILES FABRICADOS DURANTE EL SEGUN
002120-                  `DO SEMESTRE DE 1.971 = ' ALFA.
002125     CALCULO3.
002130           PERFORM DOS
002140           PERFORM UNO    VARYING A  FROM 1 BY 1  UNTIL A = 4
002150                                 AFTER B  FROM 1 BY 1  UNTIL B = 13
002160                                 AFTER C  FROM 1 BY 1  UNTIL C = 31
002170           DISPLAY `NUMERO DE AUTOMOVILES FABRICADOS POR RICOSA  EN
002180-                  `1.971, 72, 73 = ' ALFA.
002185           PERFORM FIN.
002190     DOS.
002195           MOVE 0  TO  ALFA.
002196     UNO.
002200           ADD  DIARIA (A, B, C)  TO  ALFA.
002210     FIN.
002220           CLOSE  CINTA
002230           STOP RUN.
```

FORMATO 4 :

```
PERFORM nombre-de-procedimiento-1   [THRU  nombre-de-procedimiento-2]
        VARYING { nombre-de-indice-1 }      FROM { nombre-de-indice-2 }
                { Identificador-1    }           { literal-numérico-2 }
                                                 { Identificador-2    }
        BY      { literal-numérico-3 }      UNTIL condición-1
                { Identificador-3    }
```

Esta opción se emplea cuando se quiere aumentar o disminuir el valor de un campo, denominado en identificador–1, entre cada ejecución de un procedimiento o de una serie de ellos. Así mismo, es aplicable cuanto se describe a continuación para los nombres–de–indice.

La forma como se ejecuta una PERFORM con el Formato 4 es como sigue:

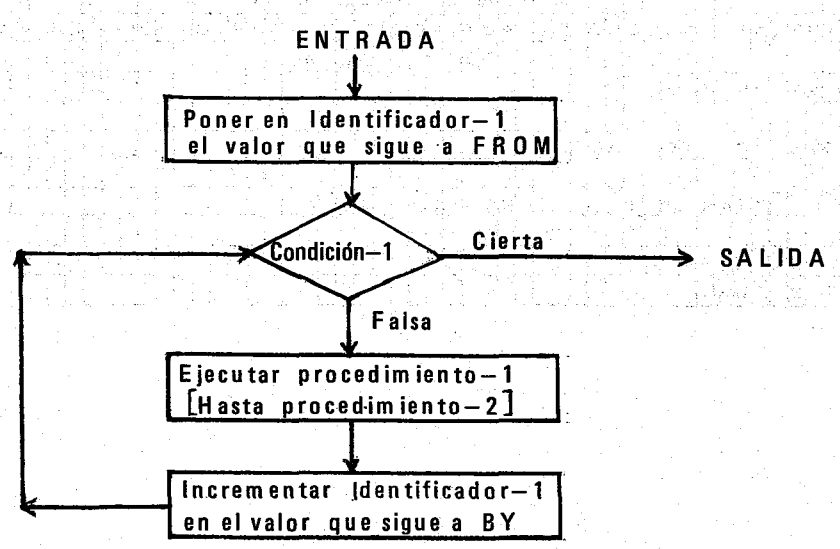
- Iguala identificador–1 al valor del contenido del campo identificador–2 ó del literal–2 que sigue a FROM.

- Evalúa (comprueba) la condición–1 para ver si es cierta o no. Si es cierta, no se ejecutarán los procedimientos comprendidos desde nombre–de–procedimiento–1 a nombre–de–procedimiento–2 y la secuencia de ejecución del programa pasará a la declaración siguiente a PERFORM. Si no es cierta la condición, se ejecuta una vez desde nombre–de–procedimiento–1 a nombre–de–procedimiento–2, y a continuación:

  Suma o resta del valor que tenga en ese momento identificador–1, el valor de identificador–3 ó literal–3 que siguen a BY (se suma si el valor del campo denominado por identificador–3 ó por literal–3 es positivo y se resta si es negativo).

  Vuelve a evaluar la condición–1, repitiéndose el ciclo a partir del segundo apartado.

El siguiente ordinograma facilita su comprensión:



EJEMPLO.

Las declaraciones que calculan el número de automóviles fabricados en la segunda decena del mes de julio de 1973 son:

```
002038  CALCULO1.
002040          MOVE 3 TO A          MOVE 7 TO B
002050          PERFORM UNO VARYING C FROM 20 BY -1 UNTIL C = 10
002060          DISPLAY `NUMERO DE AUTOMOVILES FABRICADOS DURANTE LA
002070-                 `SEGUNDA DECENA DE JULIO DE 1.973 = ' ALFA.
.....
002190  DOS.
002195          MOVE 0 TO ALFA.
002196  UNO.
002200          ADD DIARIA (A, B, C) TO ALFA.
```

En efecto, las declaraciones de la línea 002040 mueven el valor 3 al subíndice A para poder referirse a la fabricación del tercer año (1973) y el valor 7 al subíndice B para referirse al séptimo mes del año (Julio); dichos valores permanecerán constantes, mientras C irá variando, obligado por la declaración PERFORM, de −1 en −1 desde el valor inicial 20 hasta el 11, obteniendo con cada variación los automóviles fabricados los días 20, 19, 18 ... y 11 de Julio respectivamente.

– Los identificadores y literales pueden ser enteros o fraccionarios, positivos, negativos y cero.

También sería correcto especificar:

```
002150          PERFORM UNO VARYING C FROM 11 BY 1 UNTIL C = 21
```

En este caso, se iría obteniendo la fabricación comenzando en el día 11 hasta el 20 de Julio, es decir, en orden inverso a como se obtuvo en el ejemplo anterior, ya que C comienza con el valor inicial 11 y va incrementándose en una unidad.

También se habría obtenido el número de automóviles fabricados en la segunda decena de Julio, sin emplear la declaración PERFORM, como sigue:

```
    010         MOVE 3 TO A
    020         MOVE 7 TO B
    030         MOVE 11 TO C.
    040 UNO.
    050         IF C = 21
    060                 GO TO IMPRIME
    070         ELSE    ADD DIARIA (A, B, C) TO ALFA
    080                 ADD 1 TO C
    090                 GO TO UNO.
    ...  .....................................
    120 IMPRIME.  DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS DURANTE
```

Con esta solución se deduce que una PERFORM es una declaración IF que se ejecuta hasta que una determinada condición es cierta.

FORMATO 5 :

```
PERFORM nombre-de-procedimiento-1 [THRU nombre-de-procedimiento-2]
    VARYING { nombre-de-índice-1 } FROM { nombre-de-índice-2 }
            { Identificador-1    }      { literal-numérico-2 }
                                        { Identificador-2    }
    BY      { Literal-numérico-3 } UNTIL condición-1
            { Identificador-3    }
    AFTER   { nombre-de-índice-4 } FROM { nombre-de-índice-5 }
            { Identificador-4    }      { literal-numérico-5 }
                                        { Identificador-5    }
    BY      { literal-numérico-6 } UNTIL condición-2
            { Identificador-6    }
```

Esta opción se utiliza para aumentar o disminuir el valor de identificador-1, identificador-4, nombre-de-índice-1 y nombre-de-índice-4, sin embargo, se omite la referencia a los nombres-de-índice por ser aplicables a los mismos cuanto se especifica a continuación.

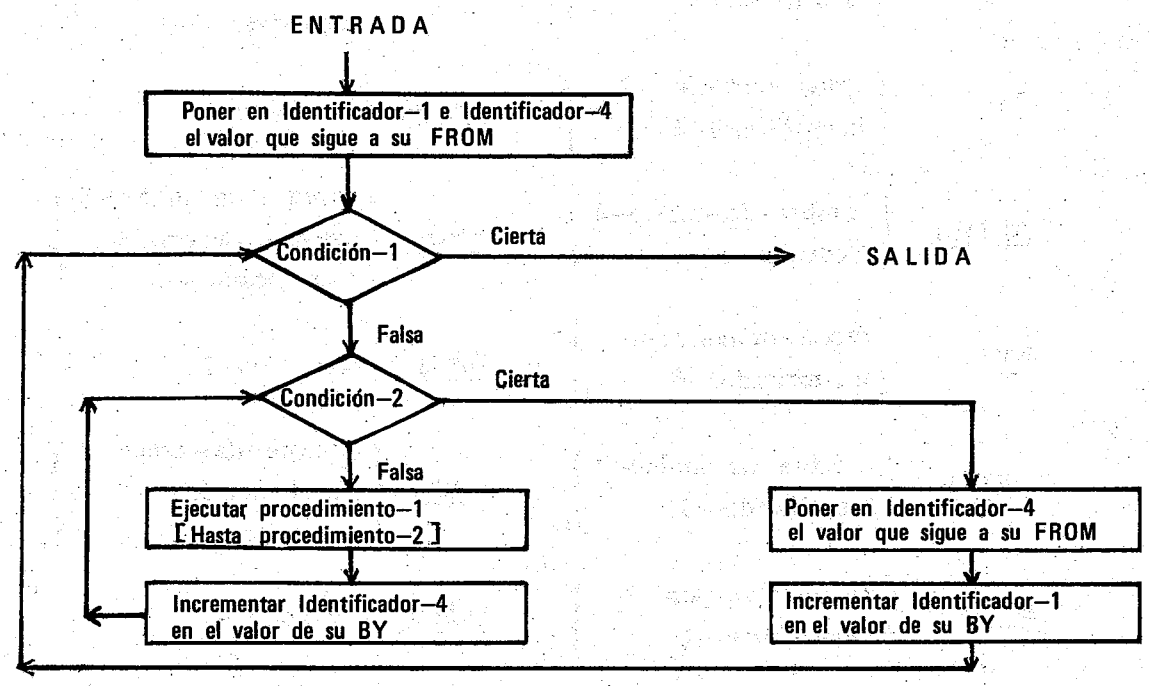
La ejecución de la declaración PERFORM se lleva a cabo de la forma siguiente:

Se iguala el contenido de identificador-1 e identificador-4 a los valores respectivos de los identificadores o literales que siguen a cada FROM.

Se evalúa la condición-1, si es cierta, no se ejecuta ningún procedimiento y la secuencia de ejecución continúa en la siguiente declaración a PERFORM. Si la condición-1 es falsa, se evalúa la condición-2, si ésta es cierta, se vuelve a igualar el valor identificador-4 al valor de identificador-5 ó del literal-5, se incrementa identificador-1 en el valor de identificador-3 y se repite el mismo proceso que se especifica al comienzo de este apartado; si el resultado de esta evaluación es que la condición-2 no es cierta, se ejecuta desde nombre-de-procedimiento-1 a nombre-de-procedimiento-2, y a continuación se incrementa identificador-4 en el valor de identificador-6 ó del literal-6; se evalúa de nuevo la condición-2, si es falsa, se vuelve a ejecutar desde nombre-de-procedimiento-1 a nombre-de-procedimiento-2, repitiéndose esta operación hasta que la condición-2 sea cierta, en cuyo momento se vuelve a igualar identificador-4 al valor de identificador-5 ó literal-5 que sigue a FROM, y se incrementa identificador-1 en el valor de identificador-3 ó literal-3. Repitiéndose todo el proceso descrito a partir del comienzo de este apartado.

Como resumen, se puede decir que el valor de identificador–4 se va variando con el valor de identificador–6, hasta pasar por un ciclo completo, cada vez que se incremente identificador–1 en el valor de identificador–3.

En forma de ordinograma:



EJEMPLO:

Con las siguientes declaraciones se obtendrá el número de automóviles fabricados en el segundo semestre de 1971:

```
002075   CALCULO2.
002080            PERFORM DOS          MOVE 1 TO A
002090            PERFORM UNO    VARYING B FROM 7 BY 1 UNTIL B = 13
002100                                AFTER C FROM 1 BY 1 UNTIL C = 31
002110            DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS DURANTE EL
002120-                   'SEGUNDO SEMESTRE DE 1.971 = ' ALFA.
.......  .....................................................
002190   DOS.
002195            MOVE 0 TO ALFA.
002196   UNO.
002200            ADD DIARIA (A, B, C) TO ALFA.
```

La primera PERFORM (línea 002080) inicializa el contador ALFA con el valor cero. La declaración MOVE pone el valor 1 en el campo A para poder referirse a la fabricación del primer año, permaneciendo dicho valor constante mientras se ejecuta la declaración PERFORM (línea 002090), obligando,sin embargo,a variar los subíndices B (identificador–1) y C (identificador–4) para obtener la fabricación de los meses 7, 8, 9 ... 12, y dentro de cada mes los días 1, 2, 3 ... 30.

FORMATO 6 :

PERFORM nombre–de–procedimiento–1 [**THRU** nombre–de–procedimiento–2]

VARYING { nombre–de–índice–1 / Identificador–1 } **FROM** { nombre–de–índice–2 / literal–numérico–2 / Identificador–2 }

BY { literal–numérico–3 / Identificador–3 } **UNTIL** condición–1

AFTER { nombre–de–índice–4 / Identificador–4 } **FROM** { nombre–de–índice–5 / literal–numérico–5 / Identificador–5 }

BY { literal–numérico–6 / Identificador–6 } **UNTIL** condición–2

AFTER { nombre–de–índice–7 / Identificador–7 } **FROM** { nombre–de–índice–8 / literal–numérico–8 / Identificador–8 }

BY { literal–numérico–9 / Identificador–9 } **UNTIL** condición–3

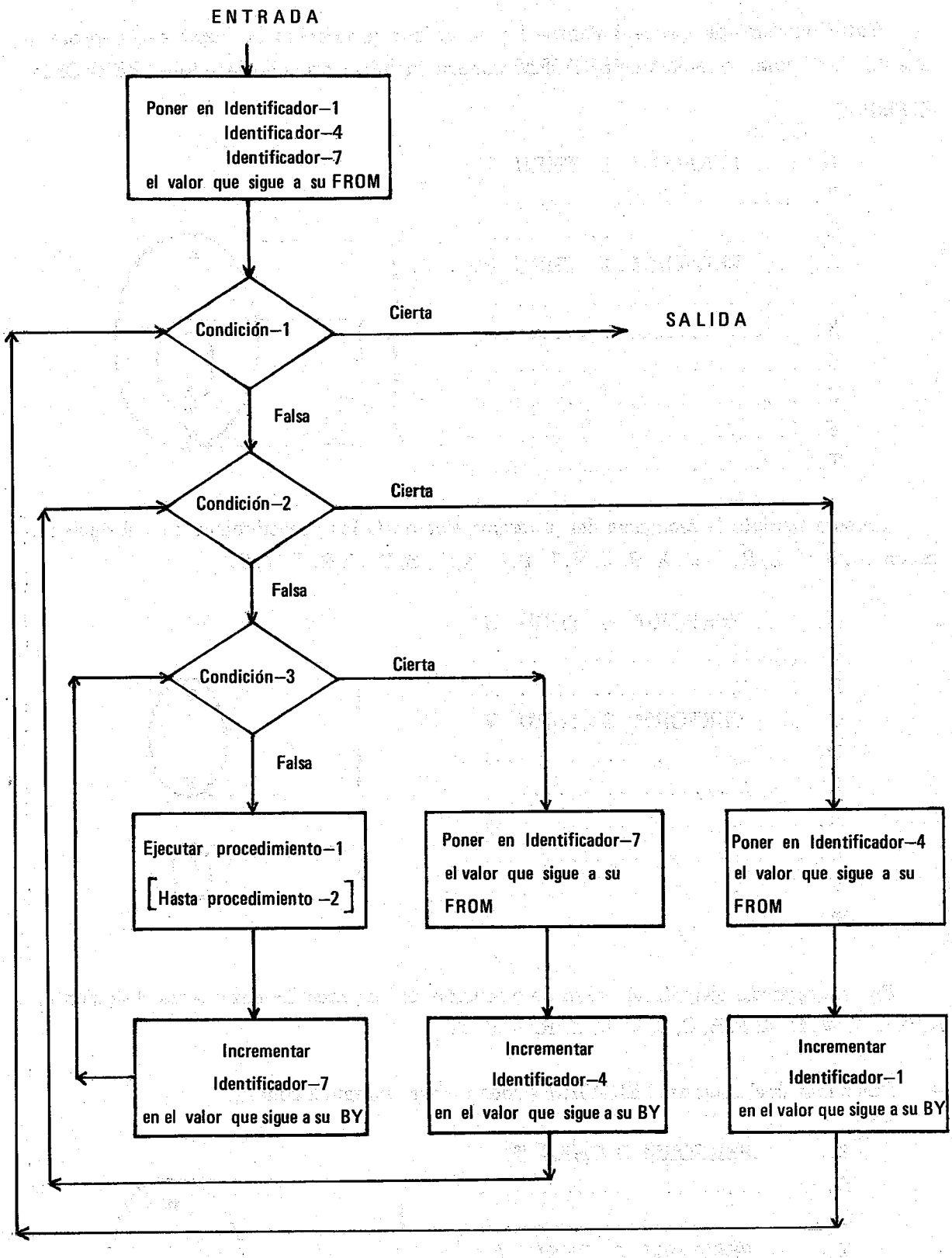
Como en las anteriores opciones, identificador–7 va variando de valor y pasa por un ciclo completo (FROM, BY, UNTIL) cada vez que se aumenta identificador–4, en el valor de identificador–6, y a su vez identificador–4, pasa por un ciclo completo cada vez que se incrementa identificador–1 en el valor de identificador–3.

Para mejor comprensión, ver el ordinograma de la página siguiente.

Las siguientes declaraciones calculan el número de automóviles fabricados durante los años 1971, 72 y 73:

```
002125   CALCULO3.
002130            PERFORM DOS
002140            PERFORM UNO    VARYING A   FROM 1 BY 1   UNTIL A = 4
002150                                 AFTER B   FROM 1 BY 1   UNTIL B = 13
002160                                 AFTER C   FROM 1 BY 1   UNTIL C = 31
002170            DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS POR RICOSA
002180-                   'EN 1.971, 72, 73 = '   ALFA.
.....
002190   DOS.
002195            MOVE 0 TO ALFA.
002196   UNO.
002200            ADD   DIARIA (A, B, C)   TO   ALFA.
```

Ordinograma :

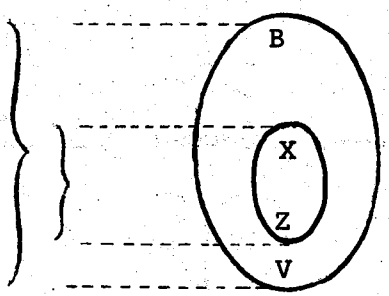


OTRAS REGLAS DE LA DECLARACION PERFORM

– Entre nombre–de–procedimiento–1 y la última declaración de nombre–de–procedimiento–2 de una declaración PERFORM pueden incluirse otras declaraciones PERFORM.

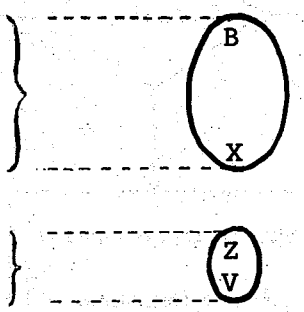
EJEMPLO:

```
A. ....PERFORM B THRU V.
P. ....................
B. ....................
C. ....PERFORM X THRU Z.
D. ....................
X. ....................
R. ....................
Z. ....................
V. ....................
T. ....................
```



En este ejemplo,la secuencia del programa ejecutaría los procedimientos en el siguiente orden: A, B, C, X, R, Z, D, X, R, Z, V, P, B, C, X, R, Z, D, X, R, Z, V, T.

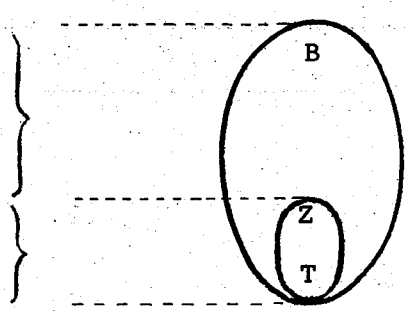
```
A. ....PERFORM B THRU X.
P. ....................
B. ....................
C. ....PERFORM Z THRU V.
D. ....................
X. ....................
R. ....................
Z. ....................
V. ....................
T. ....................
```



En este segundo ejemplo,el orden de ejecución de los procedimientos sería el siguiente: A, B, C, Z, V, D, X, P, B, C, Z, V, D, X, R, Z, V, T.

– Dos o más declaraciones PERFORM pueden tener una salida común:

```
A. ....PERFORM B THRU T.
B. ....................
C. ....................
X. ....PERFORM Z THRU T.
R. ....................
Z. ....................
V. ....................
T. ....................
M. ....................
```



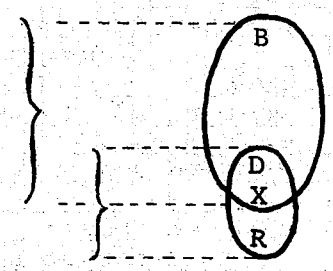
En este caso, el orden de ejecución de los procedimientos sería A, B, C, X, Z, V, T, R, Z, V, T, B, C. X, Z, V, T, R, Z, V, T, M.

En los ejemplos anteriores, la ejecución de cada PERFORM se realiza en el orden descrito anteriormente, es decir, la secuencia de ejecución del programa regresa siempre, una vez ejecutados los procedimientos que constituyen su ámbito, a la declaración que sigue a cada PERFORM.

No ocurre así cuando el ámbito de la segunda PERFORM engloba parte del ámbito de los procedimientos de la primera PERFORM.

EJEMPLO:

```
A.....PERFORM B THRU X.
P.....
B.....
C.....PERFORM D THRU R.
M.....
D.....
X.....
R.....
Z.....
V.....
T.....
```



El procedimiento X está comprendido entre los procedimientos D y R que constituyen el ámbito de la segunda PERFORM, por lo que la secuencia de ejecución sería: A, B, C, D, X, en este punto, no continuaría en R como en los casos anteriores, sino que regresaría a P, continuando en B, C, D, X, R y regresar a M, D, X, R, entrando en un bucle formado por estos cuatro procedimientos, los cuales ejecutaría un número indefinido de veces.

Existen, no obstante, algunos tipos de ordenadores en los que la secuencia de ejecución de este ejemplo sería la siguiente: A, B, C, D, X, P, B, C, D, X, R, M, D, X, R, Z, V, T.

Los gráficos que acompañan a los ejemplos precedentes, facilitan la comprensión acerca de cuando una declaración PERFORM que engloba a otra PERFORM, lleva a cabo su ejecución de forma normal o anormal.

En la programación estructurada se minimiza el empleo de la declaración GO a la vez que se potencia el de la declaración PERFORM, por lo que de llevar a cabo la confección de programas con este estilo de programación, se aconseja un estudio detenido de cuanto se ha especificado para dicha declaración (ver el ejemplo de la pág. 250).

La codificación de la PROCEDURE DIVISION del ejemplo de la página 223 en programación estructurada se haría de la forma siguiente:

```
000210  PROCEDURE DIVISION.
000220    PROCESO.
000225      OPEN  INPUT CINTA
000228      READ  CINTA  AT END  DISPLAY 'NO HAY REGISTROS'
000231      PERFORM  CALCULO1
000232      PERFORM  CALCULO2
000233      PERFORM  CALCULO3
000234      CLOSE  CINTA
000235      STOP RUN.
000238    CALCULO1.
000239      MOVE ZERO TO ALFA
000240      MOVE 3 TO A
000241      MOVE 7 TO B
000250      PERFORM UNO  VARYING  C  FROM 11 BY 1  UNTIL C = 21
000260        DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS DURANTE LA SE
000270-       'GUNDA DECENA DE JULIO DE 1.973 = '  ALFA.
000275    CALCULO2.
000280      MOVE ZERO  TO  ALFA
000285      MOVE 1 TO A
000290      PERFORM UNO  VARYING  B  FROM 7 BY 1  UNTIL B = 13
002100                           AFTER  C  FROM 1 BY 1  UNTIL C = 31
002110      DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS DURANTE EL SE
002120-       'GUNDO SEMESTRE DE 1.971 = '  ALFA.
002125    CALCULO3.
002130      MOVE ZERO  TO ALFA
002140      PERFORM  UNO  VARYING  A  FROM 1 BY 1  UNTIL A = 4
002150                           AFTER  B  FROM 1 BY 1  UNTIL B = 13
002160                           AFTER  C  FROM 1 BY 1  UNTIL C = 31
002170      DISPLAY 'NUMERO DE AUTOMOVILES FABRICADOS DURANTE LOS
002180-       'AÑOS 1.971, 1.972 Y 1.973 = '  ALFA.
002196    UNO.
002200      ADD DIARIA (A, B, C) TO ALFA.
```

8.4.- Declaración EXIT

FUNCION: Proporcionar un punto final a un procedimiento a fin de permitir que la secuencia de ejecución del programa continue en la siguiente declaración a PERFORM.

FORMATO:

nombre – de – procedimiento . **EXIT** [**PROGRAM**]

– EXIT debe ser la única declaración que siga a nombre–de–procedimiento.

Cada declaración PERFORM obliga a bifurcar hacia un procedimiento, a fin de ejecutar las declaraciones en él incluídas y, una vez ejecutada la última declaración de dicho procedimiento, regresar a la declaración que sigue a PERFORM. En el caso de que en dicho procedimiento exista alguna declaración IF, puede suceder que, como resultado de la prueba de alguna de las condiciones –cierta o falsa–, obligue o no a seguir ejecutando más declaraciones de este procedimiento y, por tanto, la secuencia de ejecución del programa no regresaría a la declaración que sigue a PERFORM. Para evitar este inconveniente se emplea la declaración EXIT, en combinación con la opción THRU de la declaración PERFORM, en la que el nombre–de–procedimiento que sigue a THRU es el nombre–de–procedimiento que precede a la declaración EXIT.

EJEMPLO:

```
140          MOVE  6 TO TOTAL.
150  ALFA.   MOVE  FICHA TO DATOSCIN
160          WRITE REGISTRO-OPE FROM REGIS
170          IF TOTAL = 10  GO TO DELTA
175             ELSE ADD 2 TO TOTAL
180                  WRITE IMPRESO FROM CABECERA AFTER 1.
190  DELTA.  EXIT.
200  BETA.   MOVE  DESCUENTO TO CAMPOA
210          ADD  2  TO TOTAL
220          ADD  1  TO CONTADOR.
230  GAMMA.  PERFORM ALFA THRU DELTA
240          COMPUTE TRIEN = NUMTRIEN * 1500.
250          .....
```

En este ejemplo, la secuencia del programa irá ejecutando las declaraciones de las líneas 140, 150, 160 y al ejecutar la declaración de la línea 170 probaría la condición: si TOTAL es igual a diez, pero al ser falso el resultado de dicha prueba, pasaría a ejecutar las declaraciones de las líneas 175 y 180, continuando en el procedimiento DELTA. En este momento la declaración EXIT no tiene efecto alguno, ya que no se encuentra activada por una declaración PERFORM, por lo que la secuencia del programa pasa a través de ella para continuar con la ejecución de las declaraciones contenidas en las líneas 200, 210, 220 y al ejecutar la declaración PERFORM la secuencia bifurcaría de nuevo a ALFA, ejecutaría las declaraciones de las líneas 150, 160 y como el valor de TOTAL en ese momento es igual a 10, bifurcaría al procedimiento DELTA, en donde la declaración EXIT, por estar activada por la declaración PERFORM, proporcionará una salida o bifurcación hacia la declaración siguiente a PERFORM, es decir ejecutaría:

```
COMPUTE TRIEN = NUMTRIEN * 1500
```

– En un programa se pueden especificar varios nombres–de–procedimiento en los que se especifique la declaración EXIT, ya que por sí misma dicha declaración no realiza acción alguna durante la ejecución del programa si no es activada previamente por una declaración PERFORM; por tanto, la secuencia de ejecución pasaría a través de ellas hasta encontrar la primera declaración del procedimiento en que no figure EXIT.

– La opción EXIT PROGRAM sirve para indicar el final lógico de un programa o parte de él. Es necesaria en la comunicación entre programas, actuando bajo el control de una declaración CALL, a fin de devolver el control al programa que realizó la llamada (ver capítulo IX).

9.- VERBO DIRECTRIZ PARA EL COMPILADOR

9.1.- Declaración NOTE

FUNCION: Permite escribir comentarios en la PROCEDURE DIVISION.

FORMATO:

NOTE serie de comentarios

La serie de comentarios que siguen a NOTE sólo se transcriben textualmente en el listado del programa fuente, pudiendo constar de cualquier combinación de los caracteres permitidos en el lenguaje COBOL; estos comentarios no producen efecto alguno durante la compilación ni en la ejecución del programa, es decir, en la compilación no se detectará ningún error por cuanto se especifique en el comentario, ni durante la ejecución podrá afectar a ninguna declaración.

Si NOTE es la primera palabra que sigue a un nombre de procedimiento, los comentarios y declaraciones que se incluyan hasta el punto final del procedimiento se considerarán como comentarios:

```
007150                MOVE 0 TO PLUSF.
007160 AVISO. NOTE LAS CUATRO DECLARACIONES SIGUIENTES CALCULAN EL VALOR
007170          HORAS EXTRAS, TOTAL BRUTO, DESCUENTO (DESCU) Y TOTAL NETO.
```

es decir, es un comentario: LAS CUATRO ..... TOTAL NETO; sin embargo, si antes del punto final del procedimiento AVISO se añadiera una declaración, esta no se ejecutaría, ya que el compilador la consideraría como continuación del comentario.

Si NOTE no es la primera palabra que sigue a un nombre de procedimiento, el comentario termina en el primer punto:

```
008100          MOVE SPACES TO IMPRESO. NOTE ESTA DECLARACION LIMPIA EL
008110                AREA DE SALIDA (PONE BLANCOS).
008120          IF CONTALINEAS = 24 WRITE IMPRESO AFTER PAG-NUEVA
```

Los compiladores COBOL basados en la versión del año 1974 prescinden de esta declaración, por pasar a realizar sus funciones un asterisco (*) en la columna siete (ver pág. 68). Sin embargo, se ha descrito dicha declaración por si algún usuario utiliza aún compiladores basados en la versión 1968, o necesita mantener programas escritos con dicha versión.

Por tanto, las inscripciones de las líneas 007160 y 007170 podían sustituirse por un asterisco seguido del comentario:

```
007150                MOVE 0 TO PLUSF.
007160*LAS CUATRO DECLARACIONES SIGUIENTES CALCULAN EL VALOR DE LAS
007170*HORAS EXTRAS, TOTAL BRUTO, DESCUENTO (DESCU) Y TOTAL NETO.
```

9.- EJERCICIOS

1.— Decir cuál de las palabras siguientes no es un nombre de procedimiento:

A) 1257
B) ALFA—
C) CALCULOTRIENIOS
D) A12

2.— Decir si existe algún error en las inscripciones siguientes y explicar el motivo.

```
140    ADD ALFA, 128 TO BETA GIVING GAMMA
150    COMPUTE ALFA = 0
150    COMPUTE NETO = (SUELDOBRUTO - DESCUEN) ROUNDED
170    MULTIPLY ALFA FOR BETA SIZE ERROR GO TO RUTINA-ERROR.
180    DIVIDE DIVISOR BY DIVIDENDO GIVING COCIENTE ROUNDED
```

3.— ¿Cuál sería el valor de CAMPOA después de ejecutada la declaración ADD?

```
100    .....
110    02 CAMPOA PIC 999 VALUE 740.
120    .....
130    ADD '260' TO CAMPOA
```

4.— En las inscripciones siguientes:

```
010        02 BETA  PIC 9(3)V9    VALUE 225.5.
020        02 ALFA  PIC 9(6)V99.
           .....
040 MULTI. MULTIPLY 5000 BY BETA GIVING ALFA ON SIZE ERROR MO
050-       VE ZEROS TO ALFA.
```

Después de efectuada la declaración MULTIPLY ¿Cuál sería el valor que se transferiría al campo ALFA?

5.— En las siguientes inscripciones :

```
           .....
170        02 CAMPOA PIC 9999 VALUE 9150.
180        02 CAMPOB PIC 999.
190        02 CAMPOC PIC 999.
           .....
210 CALCULO. DIVIDE 125 INTO CAMPOA GIVING CAMPOB REMAINDER CAMPOC
```

¿Cuál será el valor que se introducirá en CAMPOB y CAMPOC después de ejecutada la declaración DIVIDE?

6.— En el programa NOMINA, para contabilizar el número de operarios de la Nómina se utilizó el campo CONTADOR con la declaración ADD 1 TO CONTADOR. Escribir una declaración COMPUTE que haga la misma función que la ADD.

7.— *¿Son equivalentes las declaraciones siguientes?*

```
010        COMPUTE ALFA = BETA ** 2 / GAMMA * ZETA.
020        COMPUTE ALFA = BETA ** 2 / (GAMMA * ZETA).
```

¿Cuál sería el valor de ALFA si se supone que BETA = 8, GAMMA = 4 y ZETA = 2, en cada una de dichas declaraciones?

8.— Suponiendo correcta la descripción de los campos ALFA, BETA y GAMMA, explicar cuál es el motivo por el que alguna de las siguientes declaraciones es incorrecta.

```
080     02 GAMMA      PIC A(3).
090     02 ALFA       PIC 9(4).
100     02 BETA       PIC X(4).
110     ..............................

130     MOVE '1234' TO ALFA BETA
140     MOVE SPACES TO GAMMA
```

9.— Conforme a las descripciones siguientes:

```
160     02 ALFA  PIC X(5)  VALUE 'SIETE'.
170     02 BETA  PIC 9(5)  VALUE 00003.
180     ..........................................
190     IF ALFA > BETA GO TO GRATIFICAR ELSE GO TO TOTALIZAR
200
```

Al ejecutar la declaración IF ¿hacia qué procedimiento bifurcaría la secuencia de ejecución del programa ?

10.— El procedimiento CALCULO—TRIENIOS del programa NOMINA lo ha codificado el programador de dos formas:

```
010 CALCULO-TRIENIOS.
020     IF MES OF FECHAINGRESO NOT < M
030         COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3
040         NEXT SENTENCE ELSE COMPUTE NUMTRIENIOS = (A - ANNO OF
050         FECHAINGRESO) / 3.
060     COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
```

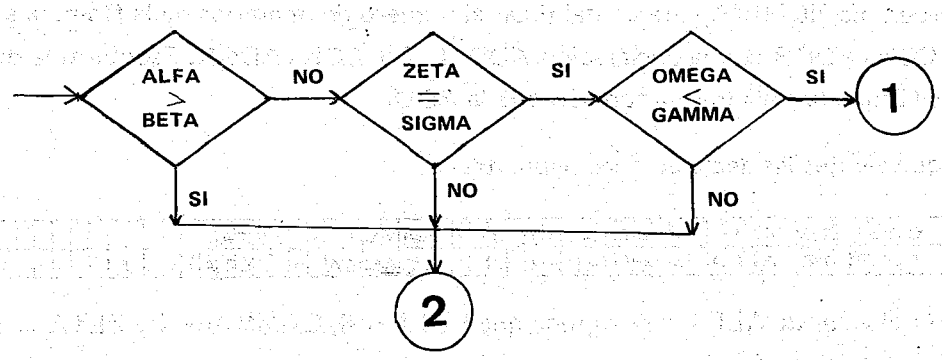
```
080 CALCULO-TRIENIOS.
090     IF MES OF FECHAINGRESO < M
100         COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3 NEXT
110         SENTENCE ELSE COMPUTE NUMTRIENIOS = (A - ANNO OF
120         FECHAINGRESO - 1) / 3 NEXT SENTENCE.
130     COMPUTE TRIENIOS = NUMTRIENIOS * 1000.
```

¿Cuál de las dos codificaciones es correcta?

11.— Codificar el siguiente ordinograma empleando:

1) Jerarquías de IF

2) Condiciones compuestas



12.— Se quiere ejecutar dentro de un programa el procedimiento CALCULO—PRIMA cada vez que el el contenido del campo EDAD sea un valor comprendido entre 25 y 50,ambos inclusive. Decir:

1.- Cuál o cuáles de las cuatro soluciones obligaría la ejecución de dicho procedimiento,y

2.- Caso de que alguna solución no lo permitiera, explicar el motivo e introducir la modificación oportuna para que lo ejecute.

```
040
050      IF EDAD NOT LESS 25 AND EDAD NOT > 50  PERFORM CALCULO-PRIMA.
060      IF EDAD > 25 OR EDAD = 25 AND EDAD EQUAL TO 50 OR EDAD < 50
070                                             PERFORM CALCULO-PRIMA.
080      IF EDAD = 25 OR EDAD > 25 AND EDAD < 50 OR EDAD = 50  PERFORM
090                                                   CALCULO-PRIMA.
100      IF EDAD NOT < 25 AND (EDAD = 50 OR EDAD < 50) PERFORM  CALCU
110-                                                  LO-PRIMA.
```

13.— ¿Qué sucedería si en la línea 003100 del programa de la pág. 196 se suprimiera el primer paréntesis que sigue inmediatamente a IF y el segundo que sigue al campo MES en la línea 003110 ?

14.— ¿Qué acciones ejecuta la declaración WRITE LINEA BEFORE 3 ?

15.— En las descripciones:

```
080 .....
090      02 ALFA PIC 99 VALUE 66.
100      02 BETA PIC 9.
... .....
120      PERFORM SUMAR VARYING BETA FROM 1 BY 3 UNTIL BETA = 5.
... .....
140  SUMAR. ADD 3 TO ALFA.
```

a) ¿Cuántas veces obligaría la declaración PERFORM a ejecutar el procedimiento SUMAR?

b) Inmediatamente después de ejecutada dicha PERFORM. ¿Cuál sería el valor almacenado en el campo ALFA?

16.— Teniendo en cuenta las siguientes inscripciones:

```
Página  Línea  A  B
                12   15   20   25   30   35   40   45   50   55   60   65
010      77 ALFA PIC 99 VALUE 8.
020      77 BETA PIC 999 VALUE 0.
... .....
040  UNO. ADD 25 TO BETA.
050  DOS. EXIT.
```

¿Qué número de veces obligarían a ejecutar los procedimientos UNO y DOS cada una de las siguientes declaraciones PERFORM?:

```
070      PERFORM UNO THRU DOS ALFA TIMES
080      PERFORM UNO THRU DOS UNTIL BETA > 200
090      PERFORM UNO THRU DOS VARYING ALFA FROM 16 BY -2
100                   UNTIL ALFA = 0
```

17.— Conforme a las inscripciones siguientes:

```
120
130   77 BETA  PIC 99.
140   .....................
150 CALCULO. SUBTRACT 15 FROM BETA DISPLAY BETA.
```

¿Qué número de veces obligaría a ejecutar el procedimiento CALCULO la declaración PERFORM?:

```
160
190          PERFORM CALCULO 3 TIMES.
200
```

18.— A la vista del enunciado del programa ACTUALIZ descrito en los ejercicios del capítulo III y IV, y teniendo en cuenta las soluciones que a dicho enunciado haya dado el programador, se quiere completar la codificación de dicho programa con la escritura de la PROCEDURE DIVISION. Para lo cual tendrá presente que:

- El campo que contiene la clave en el ARCHI—MODIFI puede contener uno de los siguientes valores:

  [1] Significa sumar las entradas habidas en el día a las existencias en almacén.

  [2] Significa restar las salidas habidas en el día de las existencias en almacén.

  [3] Significa dar de baja el registro.

- Si el código de un registro en el ARCHI—MODIFI no existe en ARCHI—MAES significa que es un ALTA y, por tanto, hay que incorporar dicho registro al ARCHI—ACTUAL, eliminando, previamente a su grabación, la última posición que contiene la CLAVE.

- Si el código de un registro del ARCHI—MAES no existe en ARCHI—MODIFI, se graba el registro del ARCHI—MAES tal cual, en el ARCHI—ACTUAL.

- Si existe igualdad en el número de código de un registro de ARCHI—MAES y ARCHI—MODIFI, habrá que sumar o restar a las existencias en almacén las entradas o salidas habidas en el día, siempre que la CLAVE contenga un 1 ó un 2 respectivamente, o dar de BAJA si la clave contiene un 3; en este caso se eliminará dicho registro (no grabándolo en ARCHI—ACTUAL).

- Se supone que no existen errores en Códigos, Cantidades y Claves y están perfectamente ordenados los registros por el número de código, así como que la cantidad pedida es siempre menor que las existencias en el almacén.

- El valor más alto del código, en cualquiera de los dos archivos, es hasta el momento: 1.500.

CAPITULO VI

TABLAS

1.- DESCRIPCION

Una tabla es un conjunto de elementos capaces de almacenar el mismo tipo de datos.

Elemento de tabla es un campo de datos que en su descripción incluye una cláusula OCCURS. Es equivalente a un nivel de la tabla.

El número de veces que un elemento se repite en la tabla se especifica en la cláusula OCCURS. El nombre del elemento junto con las demás cláusulas (PICTURE, USAGE, etc.) que entran a formar parte de su descripción se aplican a cada repetición del citado elemento.

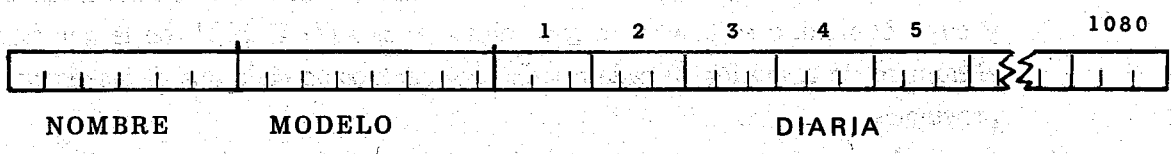
En el ejemplo de la página 132 se hace la descripción de REGISTRO-AUTOS, el cual contiene una tabla de 1080 elementos denominados DIARIA, capaces de contener caracteres numéricos.

```
070    01 REGISTRO-AUTOS.
080       02 NOMBRE                  PICTURE A(6).
090       02 MODELO                  PICTURE X(8).
100       02 DIARIA  OCCURS 1080 PICTURE 999.
```

Con esta descripción el compilador reserva una zona de la memoria donde poder almacenar el NOMBRE de la Empresa, MODELO de automóvil y el número diario de automóviles fabricados durante los tres años. Estos datos se encuentran grabados en una cinta magnética y por medio de una declaración READ (ver pág. 223, línea 002030) pasarán desde la cinta a la zona de la memoria que se ha reservado, a fin de poder realizar su tratamiento.

La reserva de memoria para cada uno de los elementos de la tabla se dispone físicamente en forma LINEAL, es decir, cada elemento se sitúa inmediatamente a continuación del anterior.

Por tanto, la disposición física de REGISTRO-AUTOS en la memoria del ordenador sería:



El lenguaje COBOL permite especificar hasta tres cláusulas OCCURS en la descripción de una tabla.

A la tabla que en su descripción se emplea una cláusula OCCURS se denomina de un NIVEL para distinguirla de las que en su descripción se emplean dos o tres cláusulas OCCURS, en cuyo caso se denominan tablas de dos o tres NIVELES respectivamente.

Esta posibilidad permite poder realizar la descripción de REGISTRO–AUTOS de la forma siguiente:

```
160  01 REGISTRO-AUTOS.
170     02 NOMBRE                    PICTURE  A(6).
180     02 MODELO                    PICTURE  X(8).
190     02 MENSUAL  OCCURS 36 TIMES.
200        03 DIARIA  OCCURS 30  PICTURE 999.
```

En la que por haberse empleado dos cláusulas OCCURS constituye una tabla de dos niveles. Con ello se ha conseguido repartir los 1080 elementos de la tabla en treinta y seis grupos de treinta elementos, denominando a cada grupo: MENSUAL.

El campo de datos MENSUAL por estar descrito con una cláusula OCCURS se le denomina también, elemento de tabla. Es realmente un elemento que contiene 30 elementos simples.

Los treinta y seis grupos MENSUAL, pueden, a su vez,agruparse en tres bloques, conteniendo cada bloque doce grupos, es decir, 3 (años) x 12 (meses) con lo que la descripción del REGISTRO–AUTOS sería la siguiente:

```
130  01 REGISTRO-AUTOS.
140     02 NOMBRE                    PICTURE  A(6).
150     02 MODELO                    PICTURE  X(8).
160     02 ANUAL  OCCURS  3.
170        03 MENSUAL  OCCURS  12.
180           04 DIARIA  OCCURS  30  PICTURE 999.
```

Describir una tabla con uno o varios niveles depende del tipo de tratamiento que haya de llevarse a cabo con ella, ya que cada descripción tiene sus ventajas e inconvenientes como se verá en el apartado SUBINDICACION.

TABLAS DE ESTRUCTURA MAS COMPLEJA

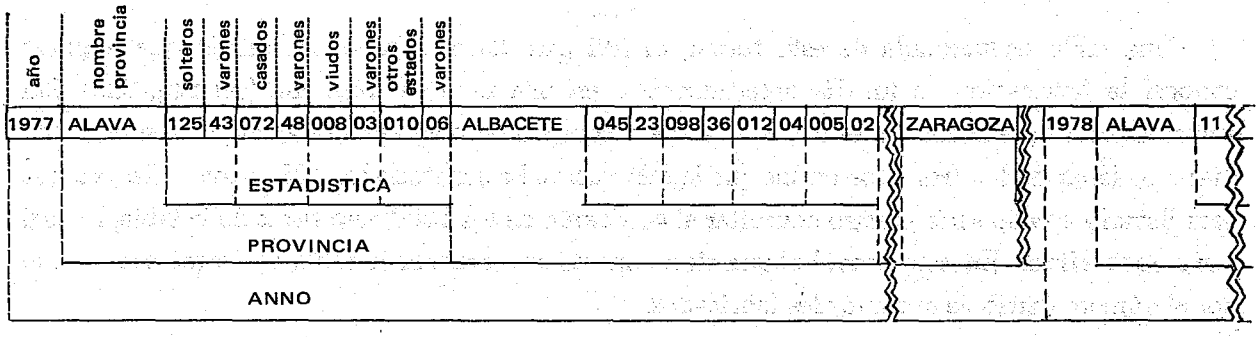
Las tablas anteriores se han presentado en su estructura más simple. Sin embargo, hay trabajos que requieren tablas descritas con una estructura más compleja.

EJEMPLO. A fin de emitir un informe acerca del estado civil del personal de la empresa, es necesario utilizar un registro que se encuentra grabado en una cinta magnética y cuyo contenido es una estadística de los años 1977 al 1981, en la que figura el número de operarios en cada estado civil nacidos en cada una de las cincuenta provincias.

Junto al número de operarios en cada estado civil figura el número de los que son varones.

La disposición de la información dentro del registro es la siguiente:

TABLA – ESTADISTICA



La descripción de la tabla capaz de contener los datos de dicho registro sería la siguiente:

```
010  01  TABLA-ESTADISTICA.
020    02  ANNO      OCCURS 5 TIMES.
030      03  NUMERO-ANNO      PIC 9(4).
040      03  PROVINCIA    OCCURS 50 TIMES.
050        04  NOMBRE-POVINCIA    PIC X(16).
060        04  ESTADISTICA    OCCURS 4 TIMES.
070          05  DATOS-ESTADO-CIVIL    PIC 999.
080          05  NUMERO-VARONES        PIC 99.
```

En esta descripción el campo NOMBRE–PROVINCIA aparece cincuenta veces por cada repetición de ANNO ; por lo tanto, NOMBRE–PROVINCIA se repite a lo largo de la tabla 5 x 50 =250 veces. Dicho valor es el producto de las cifras que siguen a las dos cláusulas OCCURS a las que está subordinado.

El campo DATOS–ESTADO–CIVIL aparece cuatro veces en cada repetición de PROVINCIA , el cual, a su vez, aparece cicuenta veces en cada uno de las cinco repeticiones del campo ANNO, es decir, el campo DATOS–ESTADO–CIVIL se repite: 5 x 50 x 4 = 1.000 veces dentro de la tabla.

1.1.- ORDENACION DE TABLAS

Dentro de la tabla los datos se sitúan en forma apropiada a fin de facilitar su consulta. Normalmente, la forma apropiada consiste en disponer los datos siguiendo un orden ascendente o descendente por un determinado concepto, clave, código, etc. que sea parte integrante de dichos datos.

Hay trabajos en los que la ordenación de los datos es sólo CONVENIENTE y otros en los que es NECESARIA.

Efectivamente, los datos que se encuentran en la cinta magnética, y que se introducirán en la tabla de cualquier nivel especificada dentro de REGISTRO–AUTOS, no han sufrido una clasificación previa, estando colocados en el orden de los días de fabricación durante los tres años, es decir, el elemento primero de la tabla contendrá el número de automóviles fabricados el

día primero de Enero, el elemento segundo contendrá los automóviles fabricados el día segundo de dicho mes y, así sucesivamente.

Una tabla estructurada de esta forma, es útil para llevar a cabo un trabajo que requiera conocer la fabricación en un día determinado o en una serie de días, etc. (ver pág. 223). Sin embargo, esta ordenación no es útil si el trabajo consiste en conocer, por ejemplo, el número de días a lo largo de los tres años en los que la fabricación ha superado los 125 automóviles, ya que para llevarlo a cabo sería preciso consultar el contenido de los 1080 elementos de la tabla, lo cual sería poco eficaz. En este caso, hubiera sido conveniente ordenar la tabla en forma ascendente por el número diario de automóviles fabricados.

Por tanto, una tabla que fue ordenada, según un determinado criterio, para realizar un trabajo, puede ser necesario modificar dicha ordenación para realizar otro tipo de trabajo, (ver ejemplo pág. 256).

La ordenación de la tabla es necesaria cuando se trata de localizar con la mayor rapidez a un determinado elemento de la misma empleando el método de búsqueda binaria (pág. 257), o la declaración SEARCH ALL (pág. 278).

En este caso, hay que especificar las opciones ASCENDING y/o DESCENDING de la cláusula OCCURS en la descripción de aquellos elementos cuyo contenido ha servido como criterio para realizar la ordenación, a fin de informar al compilador la forma en que se encuentra ordenada la tabla.

EJEMPLO. Observando los datos del registro que contiene la estadística de los estados civiles (pág. 241), estos están ordenados en forma ascendente por el número del año y por el nombre de la provincia; en este caso la TABLA–ESTADISTICA que va a contener dichos datos podía describirse:

```
010  01  TABLA-ESTADISTICA.
020      02  ANNO      OCCURS 5 TIMES
030                     ASCENDING NUMERO-ANNO.
040          03  NUMERO-ANNO     PIC 9(4).
050          03  PROVINCIA       OCCURS 50 TIMES
060                               ASCENDING NOMBRE-PROVINCIA.
070              04  NOMBRE-PROVINCIA    PIC X(16).
080              04  ESTADISTICA         OCCURS 4 TIMES.
090                  05  DATOS-ESTADO-CIVIL   PIC 999.
100                  05  NUMERO-VARONES       PIC 99.
```

con esta descripción se indica al compilador que el contenido de la tabla sigue el orden ascendente por el número del año (NUMERO–ANNO) y el orden alfabético por el nombre de las provincias (NOMBRE–PROVINCIA).

Ver ejemplo de la página 267.

1.2.- TABLAS DE LONGITUD VARIABLE

La opción DEPENDING de la cláusula OCCURS define una tabla en el que el número de sus elementos va variando mientras se lleva a cabo la ejecución del programa. Se denomina tabla de longitud variable.

EJEMPLO: Se dispone de una cinta magnética grabada con treinta y seis registros, uno por cada mes de los años 1971,72 y 73.
Cada registro contiene la fabricación diaria de automóviles de cada uno de los modelos A y B. Así mismo, figura el número de año, el nombre del mes y el número de días laborables en dicho mes.
El número máximo de días laborables en un mes fueron veintisiete.

La disposición de los datos en cada registro es la siguiente:

	AÑO	MES	Nº DIAS LABOR.	Nº AUTOS MODELO-A	Nº AUTOS MODELO-B										
REGISTRO 1º	1971	ENERO	19	125	058	182	096	145	087	107	104	098	086	100	104
REGISTRO 2º	1971	FEBRERO	17	112	089	122	079	098	055	045	067				
----------	.....														
REGISTRO 36º	1973	DICIEMBRE	18	090	085	111	099	065	088	092	110	046	082		

La longitud de cada registro es variable, ya que el número de campos en los que figura la fabricación diaria de ambos modelos de automóviles depende del número variable de días laborables en cada mes.

La descripción del registro en el que sucesivamente se irán almacenando cada uno de los treinta y seis registros de la cinta es la siguiente:

```
010  01  REGISTRO-FABRICACION.
020      02  NUM-ANNO              PIC 9(4).
030      02  NOMBRE-MES            PIC X(10).
040      02  NUM-DIAS-LABOR        PIC 99.
050      02  DIARIA    OCCURS      0  TO  27
060                                DEPENDING NUM-DIAS-LABOR.
070          03  MODELO-A          PIC 999.
080          03  MODELO-B          PIC 999.
```

Con esta descripción el compilador reserva en la memoria una zona capaz de almacenar el registro de mayor longitud (el que contiene veintisiete elementos DIARIA).

El Sistema conoce en cada momento el número de elementos de la tabla por estar su valor incluido en el campo NUM-DIAS-LABOR.

El número variable de elementos de la tabla puede figurar (como en el ejemplo precedente) inmediatamente antes de la tabla, es lo más cómodo, o en cualquier campo de datos de la DATA DIVISION.

1.3.- CREACION DE TABLAS EN LA WORKING

La descripción de una tabla puede relizarse en la FILE y en la WORKING–STORAGE SECTION. En cualquier caso y a la vista de la descripción de la tabla, el compilador reservará una zona de la memoria donde se almacenarán los datos.

Para llenar con datos una tabla habrá que tener en cuenta que:

Si se describe la tabla en la FILE SECTION, el llenado de la misma se hará, normalmente, desde un medio externo: cinta, ficha, disco, etc. donde estarán contenidos los datos, por medio de una declaración READ.

Si se describe la tabla en la WORKING–STORAGE SECTION, se llenará, bien con el contenido existente en otros campos de la memoria y/o los resultados producidos durante el proceso, o bien si los datos están contenidos en un medio externo: ficha, cinta,etc., mediante una declaración ACCEPT.

Sin embargo, el lenguaje COBOL permite al programador, a la vez que describe la tabla en la WORKING–STORAGE SECTION poder llenarla con datos sin necesidad de recurrir a un medio o soporte externo que los contenga.

Efectivamente, el siguiente ejemplo muestra la forma de describir una tabla y llenarla con el nombre de las provincias andaluzas.

```
010   WORKING-STORAGE SECTION.
020   01  TABLAPROV.
030       02  FILLER      PIC X(7)  VALUE  `HUELVA ´.
040       02  FILLER      PIC X(7)  VALUE  `SEVILLA´.
050       02  FILLER      PIC X(7)  VALUE  `CORDOBA´.
060       02  FILLER      PIC X(7)  VALUE  `JAEN   ´.
070       02  FILLER      PIC X(7)  VALUE  `ALMERIA´.
080       02  FILLER      PIC X(7)  VALUE  `MALAGA ´.
090       02  FILLER      PIC X(7)  VALUE  `GRANADA´.
100       02  FILLER      PIC X(7)  VALUE  `CADIZ  ´.
110   01  TABLAP    REDEFINES  TABLAPROV.
120       02  PROVINCIA   OCCURS 8   PIC X(7).
```

Con las inscripciones de las líneas 020 a 100 el compilador reserva una zona de la memoria de 56 posiciones (octetos) denominada TABLAPROV, en la que almacena los nombres de las ocho provincias.

Sin embargo, TABLAPROV sólo es el nombre de un conjunto de campos, pero no es una tabla, ya que no contiene en su descripción la cláusula OCCURS, por lo que no podrá emplearse como tal. Por tanto, si se quiere que los campos integrantes de TABLAPROV (con su contenido) constituyan una tabla, es necesario describir el artificio de las líneas 110 y 120.

Efectivamente, con la inscripción de la línea 110, la cláusula REDEFINES hace que el conjunto de las 56 posiciones reservadas para TABLAPROV puedan denominarse también TABLAP.

A continuación (línea 120) se reparte TABLAP en ocho campos de siete posiciones cada uno, con lo que queda constituida la tabla.

El anterior artificio no puede sustituirse por:

```
120  01  TABLAP  REDEFINES  TABLAPROV  OCCURS 8  PIC X(7).
```

por existir la limitación en COBOL de no poder especificarse la cláusula OCCURS junto a una cláusula REDEFINES, ni en inscripciones que lleven el número de nivel 01.

Otra razón por la que la creación de este tipo de tablas se debe hacer dentro de la WORKING–STORAGE SECTION, es por no estar permitido el empleo de la cláusula VALUE dentro de la FILE SECTION para inicializar los campos de la memoria con un determinado valor (en el ejemplo, los nombres de las provincias andaluzas).

2.- METODOS PARA REFERENCIAR ELEMENTOS

A cada elemento de una tabla debe poder hacerse referencia (dentro de la PROCEDURE) de forma única y exclusiva. No debe, por tanto, prestarse a equívocos al compilador.

Dos métodos son los empleados para poder referirse a cualquier elemento:

- **SUBINDICACION**
- **INDEXACION**

A lo largo de este capítulo se describen ambos métodos, haciendo especial énfasis en las ventajas e inconvenientes de cada uno.

El problema que se muestra en las páginas 257 y 258, se resuelve empleando independientemente cada método con el fin de mostrar las diferencias entre ambos, a la vez que posibilite al programador la elección del método más adecuado para el tratamiento de cada tabla.

2.1.- SUBINDICACION

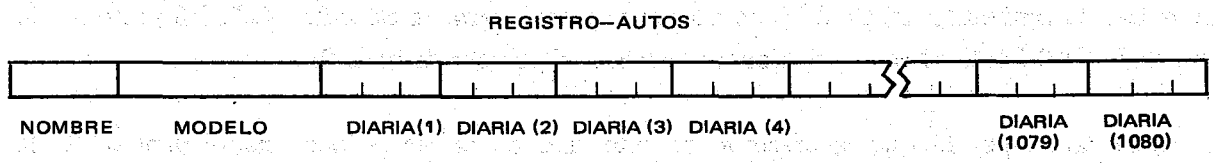
La descripción del REGISTRO–AUTOS que incluye la tabla de un nivel, los 1080 elementos que la integran se denominan de la misma forma –DIARIA–, y si se desea hacer referencia dentro del programa, a un elemento en particular, habrá que hacerlo de forma inequívoca. Para ello, es preciso el empleo del nombre general con el que se denominan todos los elementos –DIARIA– seguido del número de orden, encerrado entre paréntesis, que el elemento ocupa en la tabla.

Así, por ejemplo, si se quiere exhibir por la máquina de escribir de la consola los automóviles fabricados el día 950, se escribiría en el programa:

```
010  DISPLAY  DIARIA (950)  UPON CONSOLE
```

A este número de orden encerrado entre paréntesis, se le denomina SUBINDICE. Puede estar representado por un literal o un identificador (nombre–de–datos), siendo su contenido, en cualquiera de los casos, un valor entero.

Por tanto, los valores que tomará un subíndice para ir refiriendose a los 1080 elementos de la tabla serán:



Una tabla descrita con un nivel tiene el inconveniente de que si se desea hacer referencia al elemento que contenga los automóviles fabricados el día 20 del mes 32, tendrá el programador que hacer necesariamente el cálculo previo del número de orden que dicho elemento ocupa en la tabla, es decir: (31 x 30)+20 = 950 (el valor 31 indica el número de meses anteriores al mes 32) y por tanto, especificar: DIARIA (950).

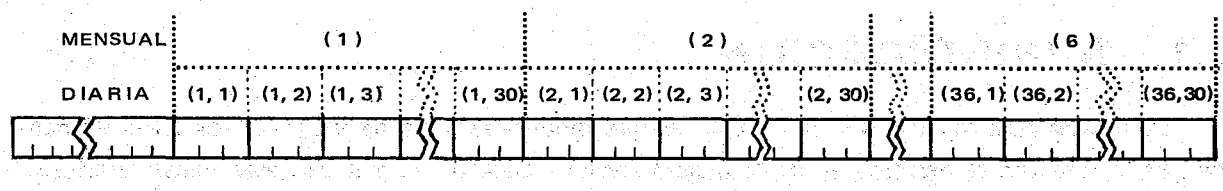
Se puede evitar este cálculo utilizando en el programa la descripción de REGISTRO–AUTOS en el que se incluye la tabla de dos niveles. En dicha tabla, equivalente a la anterior, para poder referirse a un determinado día de fabricación se deben emplear dos SUBINDICES; es decir, si se quiere referir a la fabricación del día 20 del mes 32 se especificará:

```
010 DISPLAY DIARIA (32, 20) UPON CONSOLE
```

El orden en que se especifican los subíndices, es el mismo en que están descritas las cláusulas OCCURS, del nivel mayor al menor .

Una tabla descrita con dos niveles permite también poder eferirse a un grupo de elementos DIARIA; por ejemplo, los treinta que componen el mes catorce utilizando un sólo subíndice: MENSUAL (14).

El siguiente cuadro muestra el valor que deben tomar los subíndices para poder refirse a los diferentes elementos DIARIA y MENSUAL de la tabla de dos niveles:

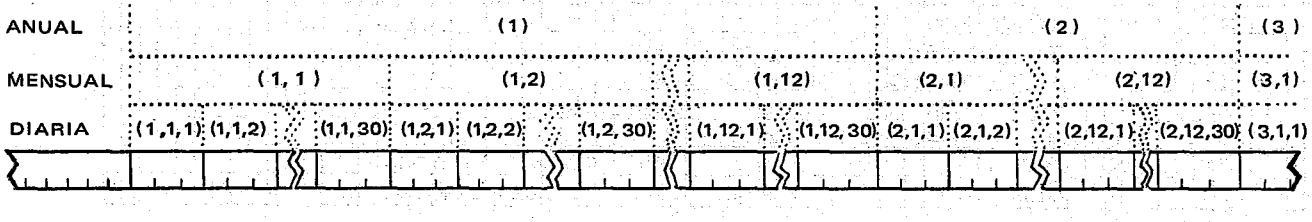


Si en la tabla de dos niveles del REGISTRO–AUTOS se quisiera hacer referencia a un determinado elemento DIARIA que contenga, por ejemplo, la fabricación del día 8 del mes 5 del año segundo, habría que hacer el cálculo para conocer a que mes y día corresponde, es decir, número del mes = (1 x 12)+ 5 =17, por lo tanto se especificará: DIARIA (17, 8).

Sin embargo, se puede evitar este cálculo si se describe en el programa la tabla con tres niveles. En cuyo caso, serían necesarios tres subíndices para refenciar el elemento indicado, es decir: DIARIA (2, 5, 8).

Una tabla de tres niveles permite también poder referirse a un bloque de elementos DIARIA utilizando el nombre del elemento de un nivel superior y menos subíndices, es decir, si lo que se prentende es referirse al conjunto de los primeros 360 elementos DIARIA contenidos en un año, se escribiría: ANUAL (1).

El siguiente cuadro muestra el valor que deben tomar los subíndices para poder referirse a los diferentes elementos ANUAL, MENSUAL y DIARIA.



En una tabla más compleja como es TABLA–ESTADISTICA de la pág 242, si se quiere hacer referencia al número de varones de la empresa nacidos en MADRID y cuyo estado civil en el año 1979 era el de casado, habrá que especificar: NUMERO–VARONES (3, 24, 2), ya que:

3 – es el lugar que ocupa en la tabla el año 1979 (comienza la estadística en 1977).
24 – es el orden que ocupa en la tabla la provincia de Madrid (según código pág. 16).
2 – es el segundo campo de ESTADISTICA donde figuran los varones casados.

Si en dicha tabla, se quisiera hacer referencia a la estadística del año 1981 correspondiente a la provincia de ALAVA, se especificará: PROVINCIAS (5, 1), (por ser quinto año y primera provincia).

Los subíndices que se han utilizado en los ejemplos anteriores son literales numéricos sin embargo, pueden utilizarse también como subíndices los nombres–de–datos, por ejemplo MENSUAL (ALFA, BETA), o bien una mezcla de los dos, es decir, DIARIA (ALFA, 7, BETA). El empleo de un tipo u otro de subíndice es criterio del programador, de acuerdo con el trabajo a realizar.

El siguiente ejemplo muestra un programa en el que se puede apreciar la utilidad de emplear como subíndices los nombres–de–datos en vez de los literales.

EJEMPLO. Se quiere escribir un programa con el que se obtenga el número de automóviles fabricados por la empresa durante el primer año de fabricación (1971), utilizando como archivo de entrada, la cinta magnética del ejemplo de la página 131.

La IDENTIFICATION, ENVIRONMENT y DATA DIVISION del programa son prácticamente iguales, tanto si se emplean como subíndices los literales numéricos como si se emplean nombres–de–datos, con la única adición en la WORKING–STORAGE SECTION, de la descripción de los nombres–de–datos que se han de emplear como subíndices (en este ejemplo sería el campo ALFA).

```
010  IDENTIFICATION DIVISION.
020    PROGRAM-ID. SUBINDIC.
030  ENVIRONMENT DIVISION.
040    INPUT-OUTPUT SECTION.
050      FILE-CONTROL.
060          SELECT ARCHIVO-AUTOS   ASSIGN SYS006-UT-2400-S.
070  DATA DIVISION.
080    FILE SECTION.
090      FD   ARCHIVO-AUTOS   RECORDING F
100                           LABEL RECORD OMITTED
110                           DATA RECORD REGISTRO-AUTOS.
120      01   REGISTRO-AUTOS.
130             02   NOMBRE          PIC A(6).
140             02   MODELO          PIC X(8).
150             02   DIARIA    OCCURS 1080 TIMES PIC 999.
160    WORKING-STORAGE SECTION.
170      77   TOTAL                  PIC 9(6)   VALUE ZEROS.
180      77   ALFA                   PIC 999    VALUE ZEROS.
```

Caso de utilizar los literales numéricos como subíndices, la PROCEDURE se escribiría:

```
010  PROCEDURE DIVISION.
020  1.   OPEN INPUT   ARCHIVO-AUTOS
030            READ ARCHIVO-AUTOS  AT END  PERFORM  FIN.
040            ADD DIARIA (1)   TO   TOTAL
050            ADD DIARIA (2)   TO   TOTAL
060             "      "          "      "
070             "      "          "      "
080             "      "          "      "
090            ADD DIARIA (359) TO TOTAL
100            ADD DIARIA (360) TO TOTAL.
110  FIN.  DISPLAY  TOTAL  UPON CONSOLE
120            CLOSE  ARCHIVO-AUTOS   STOP RUN.
```

Como se ve, se haría exhaustiva su descripción, ya que serían necesarias diecinueve hojas de codificación, mientras que si se emplean nombres-de-datos como subíndices, la PROCEDURE se escribiría de la siguiente forma:

```
010  PROCEDURE DIVISION.
020  1.   OPEN   INPUT   ARCHIVO-AUTOS.
030            READ   ARCHIVO-AUTOS   AT END   PERFORM   FIN.
040            PERFORM   CALCULO   UNTIL   ALFA = 360.
050  FIN.
060            DISPLAY   TOTAL   UPON   CONSOLE.
070            CLOSE  ARCHIVO-AUTOS     STOP RUN.
080  CALCULO.
090            ADD   1   TO   ALFA
100            ADD   DIARIA (ALFA)   TO   TOTAL.
```

2.1.1.- EJEMPLO DE MANEJO DE TABLAS

La Sección de Contabilidad de la Empresa RICOSA, con el fin de facilitar la labor del cajero, encarga la confección de un programa que obtenga el número de billetes de 5.000, 1.000, 500 y 100 pts., así como de monedas de 50, 25, 5 y 1 pts., necesarios para pagar a todos los operarios.

El trabajo va a consistir en tomar los sueldos netos que se encuentran reflejados en el conjunto de impresos que constituye la nómina de la Empresa, la cual se obtuvo en la ejecu-

ción del programa NOMINA, y perforarlos en fichas, de tal forma, que utilizando para cada sueldo cinco columnas de la ficha, pueden perforarse en cada una el contenido de 16 sueldos. Si la ultima ficha no quedara completamente llena de sueldos (caso de que hubiera causado baja alguno de los 10.000 operarios), se perforarán ceros en las columnas que sobren.

Cada sueldo neto se desglosará en los diferentes billetes y monedas de que se componen, acumulandoles en los correspondientes totalizadores.

Los resultados se imprimirán de acuerdo con el formato de salida de impresora (pág. 253).

```
001010 IDENTIFICATION DIVISION.
001020  PROGRAM-ID. DESGLOSE.
001030  AUTHOR. C. DE LA FUENTE.
001040 ENVIRONMENT DIVISION.
001050  CONFIGURATION SECTION.
001060   SPECIAL-NAMES.  C01  IS  PAG-NUEVA.
001070  INPUT-OUTPUT SECTION.
001080   FILE-CONTROL.
001090       SELECT  FICHAS  ASSIGN SYS005-UR-2540R-S.
001100       SELECT  IMPRESORA  ASSIGN SYS006-UR-1403-S.
001110 DATA DIVISION.
001120  FILE SECTION.
001130   FD  FICHAS  LABEL  RECORD OMITTED  RECORDING F.
001140   01  PAGAS.
001150       02  PAGA  OCCURS  16  PICTURE 9(5).
001160   FD  IMPRESORA  LABEL  RECORD OMITTED  RECORDING F.
001170   01  IMPRESO          PIC X(133).
001180  WORKING-STORAGE SECTION.
001190   77  N                 PIC 99    VALUE 1.
001200   77  A                 PIC 9     VALUE 1.
001210   77  COCIENTE          PIC 99.
001220   77  PRODUCTO          PIC 9(5).
001230   77  NUM-REGIS         PIC 9     VALUE 1.
002010   01  CONTADORES.
002020       02  CONTADOR  OCCURS 8  PICTURE 9(6).
002030  01  BILLETES.
002040   02 FILLER PICTURE X(32)
002050                 VALUE '50001000050001000050002500050001'.
002060  01  BILLET  REDEFINES  BILLETES.
002070   02 BILLETE  OCCURS 8  PICTURE 9(4).
002080  01  TITULARES.
002090   02 FILLER PICTURE X(27) VALUE 'BILLETES DE CINCO MIL PTS.  '.
002100   02 FILLER PICTURE X(27) VALUE 'BILLETES DE MIL PTS.        '.
002110   02 FILLER PICTURE X(27) VALUE 'BILLETES DE QUINIENTAS PTS. '.
002120   02 FILLER PICTURE X(27) VALUE 'BILLETES DE CIEN PTS.       '.
002130   02 FILLER PICTURE X(27) VALUE 'MONEDAS DE CINCUENTA PTS.   '.
002140   02 FILLER PICTURE X(27) VALUE 'MONEDAS DE VEINTICINCO PTS. '.
002150   02 FILLER PICTURE X(27) VALUE 'MONEDAS DE CINCO PTS.       '.
002160   02 FILLER PICTURE X(27) VALUE 'MONEDAS DE UNA PTS.         '.
002170  01  TITULOS  REDEFINES  TITULARES.
002180       02  TITULO  OCCURS 8  PICTURE X(27).
002190  01 CABECERA-A.
002200   02 FILLER  PICTURE X(30)  VALUE SPACES.
002210   02 FILLER  PICTURE X(103) VALUE 'NUMERO DE BILLETES Y MONEDAS
002220-                 'NECESARIAS PARA PAGAR A LOS OPERARIOS DE RICOSA'.
003010  01  CABECERA-B.
003020   02 FILLER  PICTURE X(50)  VALUE SPACES.
003030   02 TITULIN  PICTURE X(30).
003040   02 TOTAL    PICTURE ZZZ,ZZ9.
003050   02 FILLER  PICTURE X(46)  VALUE SPACES.
```

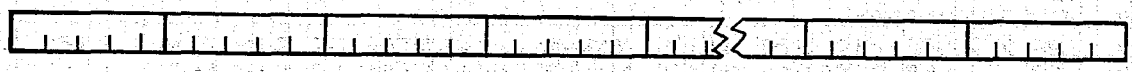
```
003070  PROCEDURE DIVISION.
003080          OPEN INPUT FICHAS  OUTPUT IMPRESORA
003090          MOVE ZEROS TO CONTADORES
003100          READ FICHAS AT END MOVE 0 TO NUM-REGIS.
003110          PERFORM DESGLOSAR UNTIL NUM-REGIS = 0
003120          PERFORM IMPRIMIR-CABECERA
003130          PERFORM IMPRIMIR-TITULOS  VARYING A FROM 1 BY 1 UNTIL A = 9
003140          CLOSE FICHAS IMPRESORA
003150          STOP RUN.
003160  DESGLOSAR.
003170          PERFORM DESGLOSE-PAGAS
003180                                    VARYING  N  FROM  1  BY  1  UNTIL  N = 17
003190                                    AFTER  A  FROM  1  BY  1  UNTIL  A = 9.
003200          READ FICHAS AT END MOVE 0 TO NUM-REGIS.
003210  DESGLOSE-PAGAS.
003220          DIVIDE BILLETE (A) INTO PAGA (N) GIVING COCIENTE
003230          ADD COCIENTE TO CONTADOR (A)
003240          MULTIPLY COCIENTE BY BILLETE (A) GIVING PRODUCTO
003250          SUBTRACT PRODUCTO FROM PAGA (N).
004010  IMPRIMIR-CABECERA.
004020          WRITE IMPRESO FROM CABECERA-A  AFTER  PAG-NUEVA.
004030  IMPRIMIR-TITULOS.
004040          MOVE TITULO (A)  TO  TITULIN
004050          MOVE CONTADOR (A)  TO  TOTAL
004060          WRITE IMPRESO  FROM  CABECERA-B  AFTER 2.
```

Dentro del programa se hacen las siguientes inscripciones:

- De la línea 001010 a la 001100 el nombre del programa, y el nombre de los archivos de entrada y de salida.

- De la línea 001130 a la 001150 descripción del archivo de entrada, así como la descripción del registro de entrada –PAGAS–, que está constituído por una tabla en la cual se irá colocando , sucesivamente, el contenido de cada ficha del archivo de entrada para su proceso.

Gráficamente se puede representar este registro de entrada, junto con la denominación de los diferentes campos que lo componen, como sigue:



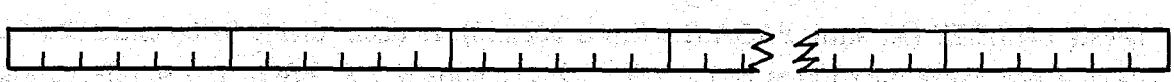
PAGA (1) PAGA (2) PAGA (3) PAGA (4) PAG (14) PAGA (15) PAGA (16)

- En las líneas 001160 y 001170 descripción del archivo y registro de salida.

- De la línea 001190 a la 001220 se describen los campos N y A que se utilizarán como subíndices para poder referise a los elementos de las cuatro tablas que se especifican en el programa, así como los campos COCIENTE y PRODUCTO que contendrán resultados parciales.

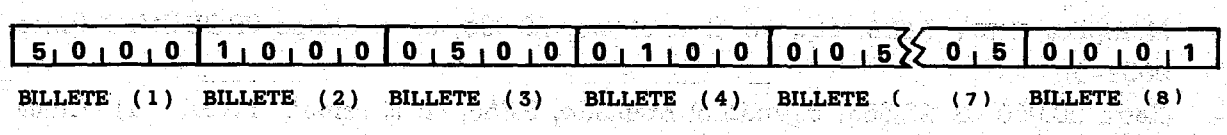
- En las líneas 002010 y 002020 se describe una tabla compuesta de ocho campos que van a servir de contadores, y en cada uno de los cuales se irá acumulando la respectiva clase de billetes y monedas de cada paga. Por tanto, al final de la ejecución del programa su contenido será el total general de billetes y monedas.

Gráficamente la tabla estaría representada como sigue:

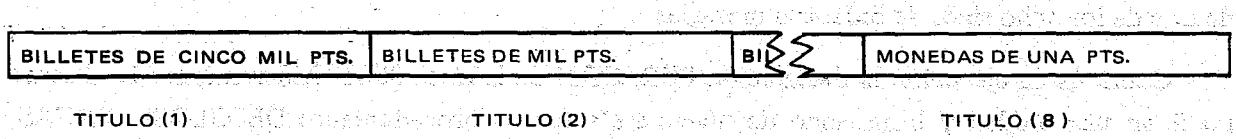


– De la línea 002030 a la 002070 se describe una tabla inicializada con el valor de cada uno de los diferentes tipos de billetes y monedas.

Gráficamente dicha tabla estaría representada como sigue:



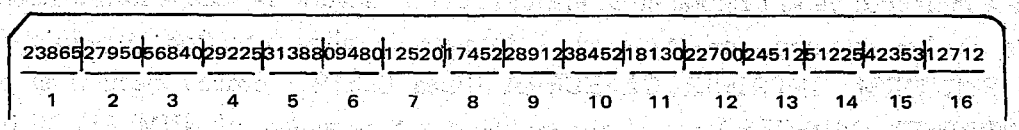
– De la línea 002080 a la 002180 se describe una tabla cuyo contenido son los ocho titulares que se han de imprimir en el informe, y que gráficamente se representaría:



– De la 002190 a la 002220 se describe el encabezamiento que ha de llevar el informe.

– De la línea 003010 a 003050 se describe un registro en el que, sucesivamente, se irá almacenando, en el campo TITULIN, cada uno de los ocho titulares de la tabla precedente, así como en el campo TOTAL se almacenará la cantidad total de billetes o monedas que se encuentre en cada uno de los respectivos contadores. Este registro, una vez completo, se enviará al área de salida para imprimirse, constituyendo una línea de detalle del impreso,y así hasta completar las ocho líneas.

Para facilitar la explicación de la PROCEDURE, se seguirá el proceso, tomando el primer sueldo de la primera ficha del archivo cuyo contenido es el siguiente:



De acuerdo con el proceso descrito por el programador, las operaciones se realizarían de la forma siguiente: Después de abrir los archivos de entrada y salida (línea 003080), e inicializar con ceros los contadores que totalizarán los diferentes tipos de billetes y monedas (línea 003090) lee la primera ficha del archivo (línea 003100) pasando su contenido a ocupar el área de la memoria, que se reservó con la descripción de la tabla: PAGAS (líneas 001140 y 001150).

La declaración PERFORM de la línea 003110 obliga a ejecutar el procedimiento DESGLOSAR (línea 003160), en el cual la declaración PERFORM DESGLOSE–PAGAS (línea 003170) hace que se inicializen los campos A y N con el valor 1 (FROM 1), y a continuación bifurque al procedimiento DESGLOSE– PAGA (línea 003210).

La ejecución de este procedimiento se realiza de la forma siguiente:

- Divide BILLETE (A) entre PAGA (N) (línea 003220), es decir, divide 5.000 entre 23.865, ya que por haber inicializado con el valor 1 los campos N y A, PAGA (N) se referirá a la primera paga de las 16 contenidas en la tabla PAGAS, y BILLETE (N) se referirá el valor del primer elemento de la tabla denominada BILLET cuyo contenido es el valor de cada billete o moneda. El resultado lo coloca en el campo COCIENTE.
- Suma el valor del campo COCIENTE al valor del CONTADOR (A) (línea 003230), es decir, al primer contador, el cual por haber sido inicializado con el valor cero (línea 003090) pasará a contener el valor cuatro, que será el número de billetes de cinco mil pesetas.
- Multiplica 4 por 5.000, dejando el resultado, 20.000, en el campo PRODUCTO (línea 003240).
- Resta 20.000 de 23.865, dejando el resultado, 3.865, en el campo PAGA (1) (línea 003250).

Por tanto, cada vez que ejecuta el procedimiento DESGLOSE-PAGA se obtiene el desglose de uno de los ocho tipos de billetes o monedas.

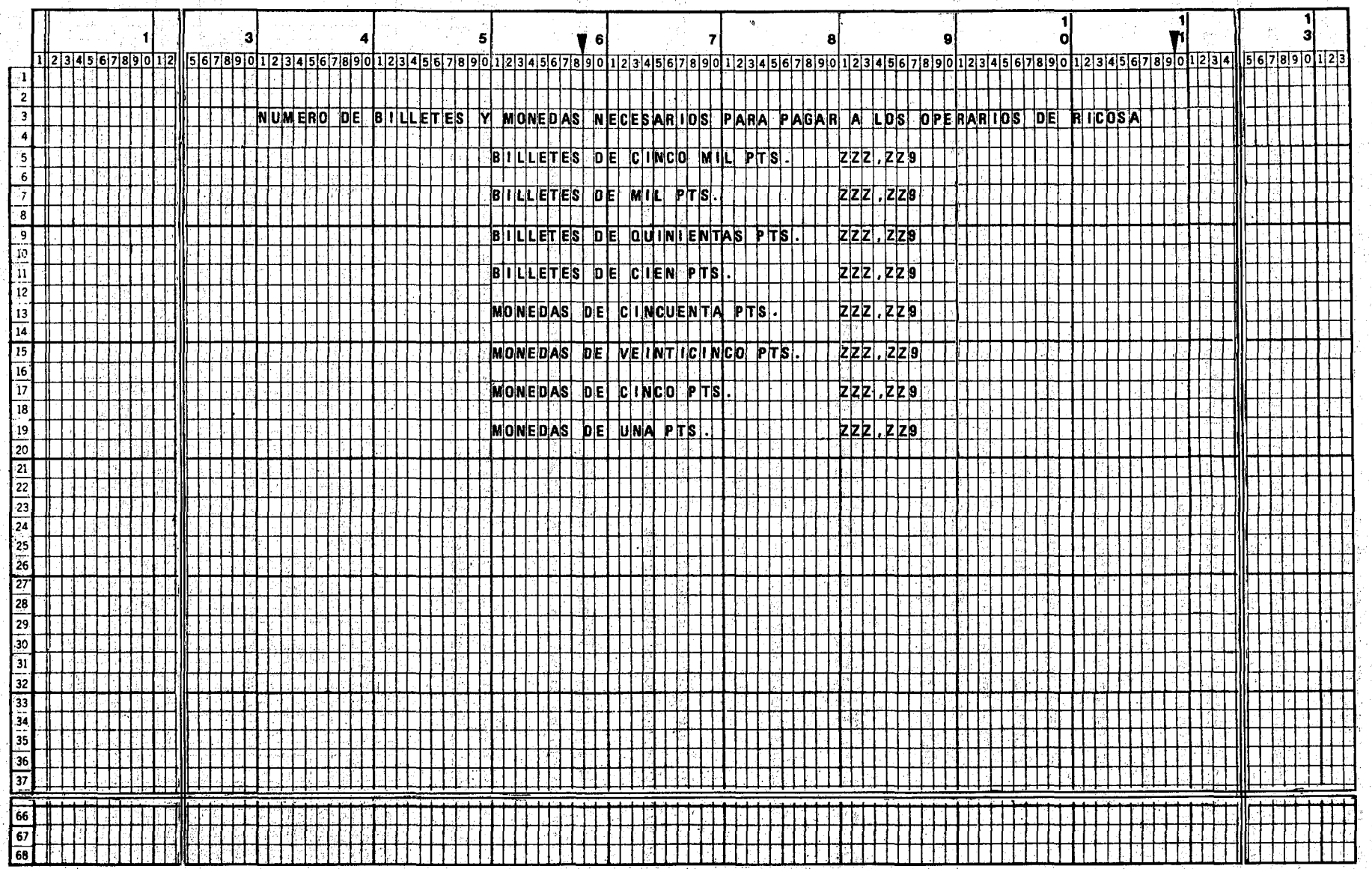
Continúa su ejecución la declaración PERFORM de la línea 003170, incrementando el campo A en una unidad y bifurcando de nuevo a efectuar el procedimiento DESGLOSE-PAGAS, a fin de obtener el número de billetes de 500 pts. Este ciclo lo repite ocho veces, con lo que habrá obtenido el desglose de la primera paga. A continuación, inicializa de nuevo el campo A al valor 1 e incrementa en una unidad el valor del campo N a fin de realizar el desglose de la segunda paga. Cuando el valor de N sea igual a 17 se habran procesado las 16 pagas de la primera ficha, en cuyo momento, pasa la secuencia de ejecución a la declaración READ (línea 003200), con la que lee la siguiente ficha.

Todo el proceso anterior se realiza con la segunda y restantes fichas del archivo, siendo ordenado, igualmente que para la primera ficha, por la declaración PERFORM de la línea 003110, la cual se está ejecutando, hasta que el valor de NUM-REGIS sea igual a 0. Esto sucederá cuando lea la ficha que contiene la marca de fin de archivo, ya que en ese caso al cumplirse la condición AT END transferirá el valor 0 a NUM-REGIS (línea 003200).

En ese momento ya se dispone en la memoria del ordenador de toda la información deseada, por lo que hay que imprimirla conforme al formato de salida (pag. 253) para lo cual la declaración PERFORM IMPRIMIR-CABECERA (línea 003120) ordena ejecutar el procedimiento IMPRIMIR-CABECERA con el que escribe el encabezamiento " NUMERO DE BILLETES Y ..... ", y a continuación, la declaración PERFORM IMPRIMIR-TITULOS (línea 003130) pone el valor 1 en el campo A y pasa a ejecutar el procedimiento IMPRIMIR-TITULOS, con lo que mueve el contenido del primer elemento, TITULO (1), es decir, BILLETES DE MIL PTS. al campo TITULIN (línea 004040), y el valor de CONTADOR (1), es decir, el número de billetes de mil pts. al campo TOTAL (línea 004050). Una vez lleno el registro denominado CABECERA-B, escribe la primera línea de detalle del impreso (línea 004060). Vuelve a sumar 1 al campo A, por lo que este pasa a valer 2, a fin de poder referirse a TITULO (2), CONTADOR (2), etc, repitiendose el ciclo anterior. Y así sucesivamente se irán imprimiendo las restantes líneas del informe.

Una vez finalizada la impresión, cierra los archivos (línea 003140) y finaliza el proceso .

FORMATO DE SALIDA DE IMPRESORA



2.2.- INDEXACION

Al describir el FORMATO 3 de la cláusula OCCURS (pág. 133)se incluyó la opción INDEXED BY nombre–de–índice; dicha opción servirá para que el compilador genere internamente un campo de datos binario de cuatro bytes de longitud y cuya misión es la equivalente al subíndice en el método de subindicación, es decir, permitir poder hacer referencia a un elemento de la tabla y proceder a su búsqueda en la misma.

Con el empleo de este método se consigue:

- No tener que hacer la descripción del nombre–de–índice.
- Ahorro de memoria del ordenador.
- Mayor rapidez en la búsqueda de cualquier elemento de la tabla.
- Que el programador no tiene necesidad de ocuparse de controlar la validez de los valores de los índices,ya que el Sistema se encarga de ello.

2.2.1.- Tipos de INDEXACION

La indexación puede ser:
- Directa.
- Relativa.

En la Indexación Directa,el nombre–de–índice se especifica de la misma forma que si fuera un subíndice, es decir,el nombre–de–índice encerrado entre paréntesis.

En la Indexación Relativa se especifica dentro del paréntesis, el nombre–de–índice seguido de un espacio, del signo + ó -, de otro espacio y de un literal numérico entero.

En el siguiente ejemplo (tomado del programa de la página 268) se emplean los dos tipos de indexación para ordenar alfabéticamente los nombres de los operarios contenidos en TABLA2.

```
002040  .....
002050      01  TABLA2.
002060          02  ELEMEN  OCCURS  45    INDEXED BY ALFA.
002070              03  NOMBRE     PIC X(35).
002080              03  VENTAS     PIC 9(4).
.....  .....
003100      INICIO.  PERFORM ORDENA  THRU  SALIDA  VARYING CONTADOR
003110                                    FROM 1 BY 1  UNTIL CONTADOR = 45
003120                                    AFTER ALFA FROM 1 BY 1 UNTIL ALFA = 45
003130                   GO TO IMPRE.
003140      ORDENA.  IF NOMBRE (ALFA + 1) > NOMBRE (ALFA) GO TO SALIDA
003150               ELSE    MOVE ELEMEN (ALFA + 1) TO CAMPOAUX
003160                       MOVE ELEMEN (ALFA) TO ELEMEN (ALFA + 1)
003170                       MOVE CAMPOAUX  TO ELEMEN (ALFA).
003180      SALIDA.  EXIT.
```

Al especificar MOVE ELEMEN (ALFA + 1) TO CAMPOAUX (línea 003150) se está haciendo uso de la indexación relativa; mientras que al especificar MOVE CAMPOAUX TO ELEMEN (ALFA) (línea 003170) se está haciendo uso de la indexación directa.

3.- BUSQUEDA en TABLAS

En los ejemplos anteriores, referentes a manejo de tablas, el programador se ha limitado a proporcionar al Sistema, mediante subíndices, la dirección de un elemento de la tabla, con la cual ha localizado dicho elemento para posteriormente utilizar su contenido. Existe , sin embargo, otro tipo de aplicaciones en los que el trabajo con tablas requiere dos tipos de acciones:

- Proporcionar al Sistema una dirección a fin de localizar un determinado elemento.
- Comprobar si el contenido de dicho elemento satisface una determinada condición.

En esta última acción reside la diferencia entre lo que es la simple localización de un elemento de una tabla y una búsqueda.

Entre los diversos métodos de búsqueda que existen, se han elegido, por su sencillez y rapidez, los siguientes:

- **BUSQUEDA SECUENCIAL**
- **BÚSQUEDA BINARIA o DICOTOMICA**

A continuación se describen ambos métodos de búsqueda, en los que se utilizan subíndices e índices para poder referirse a los elementos de la tabla.

3.1.- Búsqueda SECUENCIAL (Con Subíndices)

El proceso de búsqueda secuencial tiene por objeto recorrer sucesivamente todos los elementos de la tabla hasta encontrar un elemento de la misma que satisfaga una condición determinada.

La búsqueda secuencial comienza inicializando el o los subíndices con un determinado valor, lo cual servirá para localizar en la tabla al elemento que ocupa un lugar igual al valor del subíndice. Se prueba si el contenido de dicho elemento satisface una condición (simple o compuesta), caso afirmativo, finaliza la búsqueda y, caso contrario, se procede a ir dando al subíndice nuevos valores en serie (ascendentes o descendentes) hasta la localización del elemento cuyo contenido satisfaga dicha condición, en cuyo caso finaliza la búsqueda. Cuando se ha recorrido toda la tabla y no existe ningún elemento que cumpla la condición, finaliza la búsqueda.

El método de búsqueda secuencial es obligado:

- Cuando los elementos de la tabla no están ordenados en secuencia ascendente o descendente por el contenido de un determinado campo, clave, etc., de dichos elementos.

Es el método que conviene utilizar:

- Cuando la búsqueda de un elemento se ha de realizar en un segmento reducido de la tabla, por ser el método más rápido.

El siguiente ejemplo muestra la forma de realizar la búsqueda secuencial, en el segmento de una tabla no ordenada, de aquel elemento que cumpla una determinada condición.

EJEMPLO. Con el programa siguiente se quiere obtener cuál es el lugar que ocupa en la tabla descrita en el ejemplo de la pág. 131, el primer elemento a partir del tercer año de producción cuyo contenido sea de 80 automóviles.

```
001010  ID DIVISION.
001020  PROGRAM-ID. BUSECUEN.
001030  REMARKS. PROGRAMA PARA BUSCAR EN FORMA SECUENCIAL AL
001040                   ELEMENTO DE LA TABLA CUYO CONTENIDO SEA 080.
001050  ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070  FILE-CONTROL.
001080  SELECT ARCHIVO-AUTOS ASSIGN SYS006-UT-2400-S.
001090  DATA DIVISION.
001100  FILE SECTION.
001110  FD ARCHIVO-AUTOS LABEL RECORD OMITTED.
001120  01 REGISTRO-AUTOS.
001130  02 NOMBRE PIC A(6).
001140  02 MODELO PIC X(8).
001150  02 DIARIA PIC 999 OCCURS 1080.
001160  WORKING-STORAGE SECTION.
001170  77 ALFA PIC 9(4).
001180  PROCEDURE DIVISION.
001190  ABRIR. OPEN INPUT ARCHIVO-AUTOS.
001200  LEER. READ ARCHIVO-AUTOS AT END GO TO FIN.
001210  INICIO. PERFORM BUSCAR THRU SALIDA VARYING ALFA
001220                   FROM 721 BY 1 UNTIL ALFA = 1081.
001230                   GO TO NOEXIS.
002010  BUSCAR. IF DIARIA (ALFA) = 080
002020                   DISPLAY 'EL ELEMENTO QUE CONTIENE EL VALOR 080
002030                   'OCUPA EN LA TABLA EL LUGAR = ' ALFA
002040                   GO TO FIN.
002050  SALIDA. EXIT.
002060  NOEXIS. DISPLAY 'NO EXISTE UN DIA A PARTIR DEL TERCER AÑO
002070                   'CUYA PRODUCCION ES DE 80 AUTOMOVILES'.
002080  FIN. CLOSE ARCHIVO-AUTOS
002090                   STOP RUN.
```

Comienza la búsqueda en la tabla a partir del elemento 721, por ser el que contiene el primer día de producción del tercer año, y a partir de él se recorre la tabla secuencialmente, utilizando el campo ALFA como subíndice. Efectivamente, por medio de la declaración PERFORM (línea 001210) se va incrementando dicho subíndice de unidad en unidad, para así referirse a cada elemento de la tabla y poder comparar el contenido (línea 002010) de cada uno de ellos con el valor 80. Esta operación se repite hasta localizar o no al elemento que satisfaga la condición de ser su contenido igual al citado valor.

El primer elemento que cumpla dicha condición hace que se imprima el mensaje 'EL ELEMENTO QUE CONTIENE ... ' junto con el lugar que ocupa en la tabla (línea 002020). Caso de no existir algún elemento que cumpla la condición se imprimirá el mensaje 'NO EXISTE UN DIA A .....' (línea 002060), finalizando el proceso.

En este programa no se describe el archivo de salida por no ser necesario, ya que utiliza la declaración DISPLAY para dar salida a los resultados.

3.2.- Búsqueda BINARIA (Con Subíndices)

El método de búsqueda binaria tiene por objeto localizar en una tabla, en la que sus elementos están dispuestos en orden ascendente o descendente por una determinada clave, código, campo, etc., a aquel elemento que satisfaga la condición de contener una clave, código, valor, etc., IGUAL al que se busca.

Por tanto, este método de búsqueda debe utilizarse siempre que la tabla esté ordenada, ya que esta ordenación permite que cualquier elemento sea localizado con un número muy reducido de búsquedas.

La búsqueda binaria se realiza de la forma siguiente: Se comienza entrando en la tabla en el elemento que ocupa el lugar central (con lo cual se ha dividido la tabla en dos partes), a continuación se compara el contenido de este elemento con el contenido a localizar. Si el resultado de la comparación es de igualdad, quiere decir que se ha localizado al elemento que se busca. Pero si el resultado de la comparación no es de igualdad debido, por ejemplo, a que el valor del elemento central de la tabla es mayor que el valor buscado, se prescinde de la mitad de la tabla donde se encuentran los valores superiores al buscado, y se toma para nueva búsqueda la otra mitad de la tabla, donde se encuentran los valores menores o igual al citado.

Esta operación se repetirá un número limitado de veces (ver pág. 265) hasta la localización del elemento buscado, lo cual sucederá siempre que exista en la tabla algún elemento con el contenido objeto de la búsqueda.

EJEMPLO.

La empresa RICOSA dispone de una plantilla de 45 agentes de ventas de automóviles distribuidos por toda la geografía nacional.

El Centro de Proceso de Datos emite mensualmente un listado, ordenado alfabéticamente, en el que figura el nombre de cada agente de ventas junto con el total de automóviles vendidos durante dicho mes.

Para poder efectuar el citado trabajo, al final de cada mes cada vendedor remite una copia de las facturas, en cada una de las cuales figura el número de automóviles vendidos cada día. Esta información es grabada en una cinta por medio de una unidad grabadora, constituyendo un archivo formado por cuarenta y cinco registros cada uno de los cuales con la siguiente estructura:

POSICION	CONTENIDO
1 a 35	Apellidos y nombre del agente
36 a 37	Número de días en que realizaron ventas
38 a (1)	Número de ventas de cada día.

(1) – Representa un número variable de posiciones, que depende del número de días en que hayan

efectuado ventas, reservando dos posiciones por día, es decir, si durante el mes el agente ha tenido diez días de ventas, el número de posiciones a reservar será 2 x 10 = 20.

Si, durante el mes, el agente no ha efectuado ninguna venta, no se reserva posición alguna a partir de la posición 38, y se coloca la cifra 00 en las posiciones 36 y 37.

Los registros se han ido grabando a medida que se han ido recibiendo por correo las copias de las facturas de ventas de cada agente, por lo que el archivo resultante no está ordenado por clave alguna.

El programa a confeccionar debe realizar los pasos siguientes:

1) Cálculo del número de automóviles vendidos por cada agente.

2) Crear una tabla dentro de la memoria del ordenador en la que cada elemento contenga los apellidos y nombre de cada agente, así como la cantidad de automoviles vendidos.

3) Clasificar dicha tabla alfabéticamente por apellidos y nombre de cada uno de los agentes.

4) Imprimir el contenido de dicha tabla conforme el formato de salida de impresora (ver pág. 259).

5) Cálculo del número máximo de búsquedas que sería necesario realizar para localizar un elemento cualquiera de la citada tabla, haciendo dicha búsqueda en forma binaria.

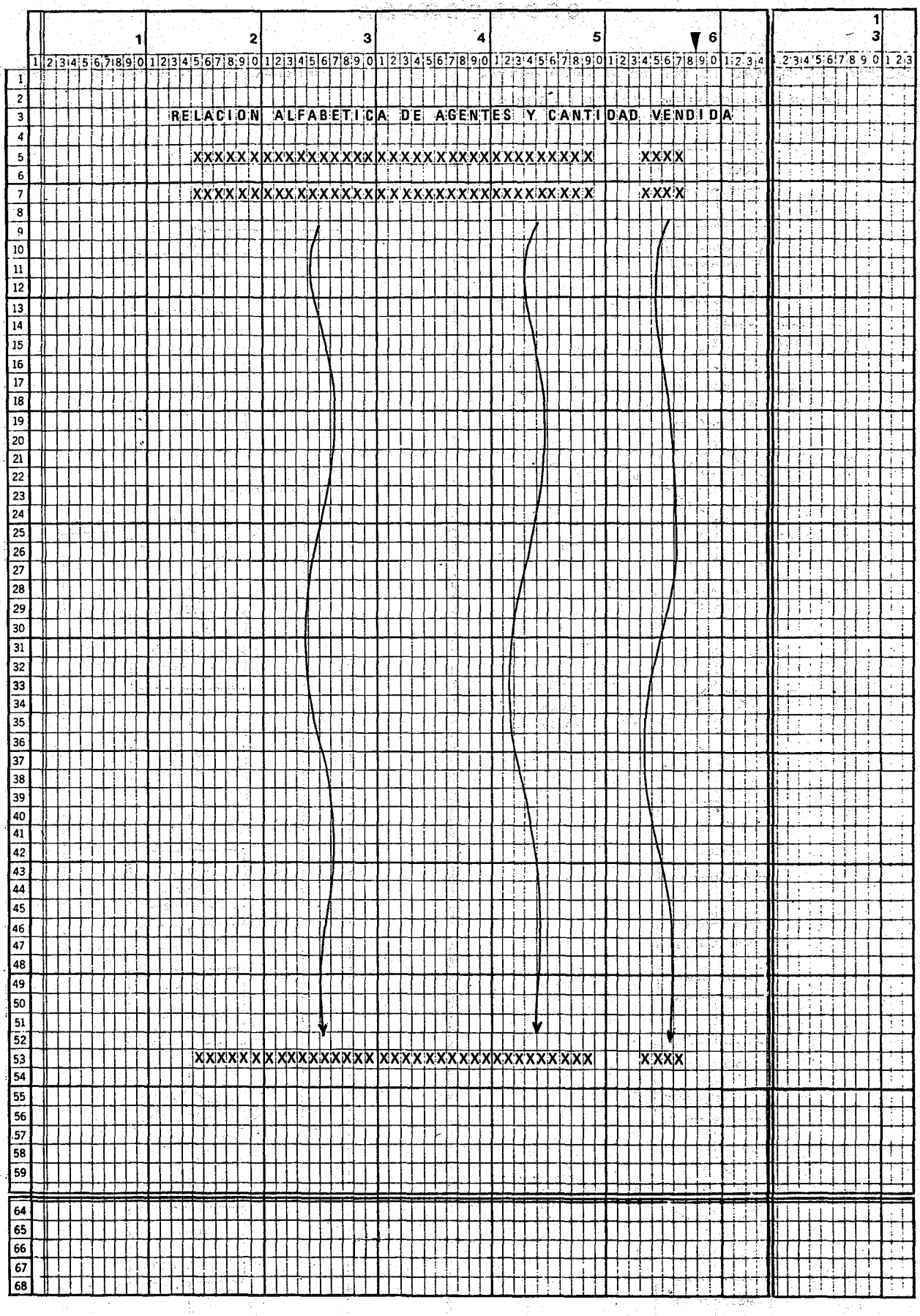
6) Búsqueda, en la forma citada, del elemento de la tabla que contenga al agente GOMEZ MANRIQUE JUAN, imprimiendo el nombre del citado agente junto con el número de automóviles vendidos en ese mes; caso de no ser localizado, escribir un mensaje por la impresora en el que figure que dicho agente no existe.

CONDICIONES:

- No existen dos agentes con los mismos apellidos y nombre.
- El número máximo de automóviles que pueden vender en un día no es superior a 99.
- Si existe el agente GOMEZ MANRIQUE JUAN, imprimir, al comienzo de una página, su nombre y número de automóviles vendidos.

SOLUCION:

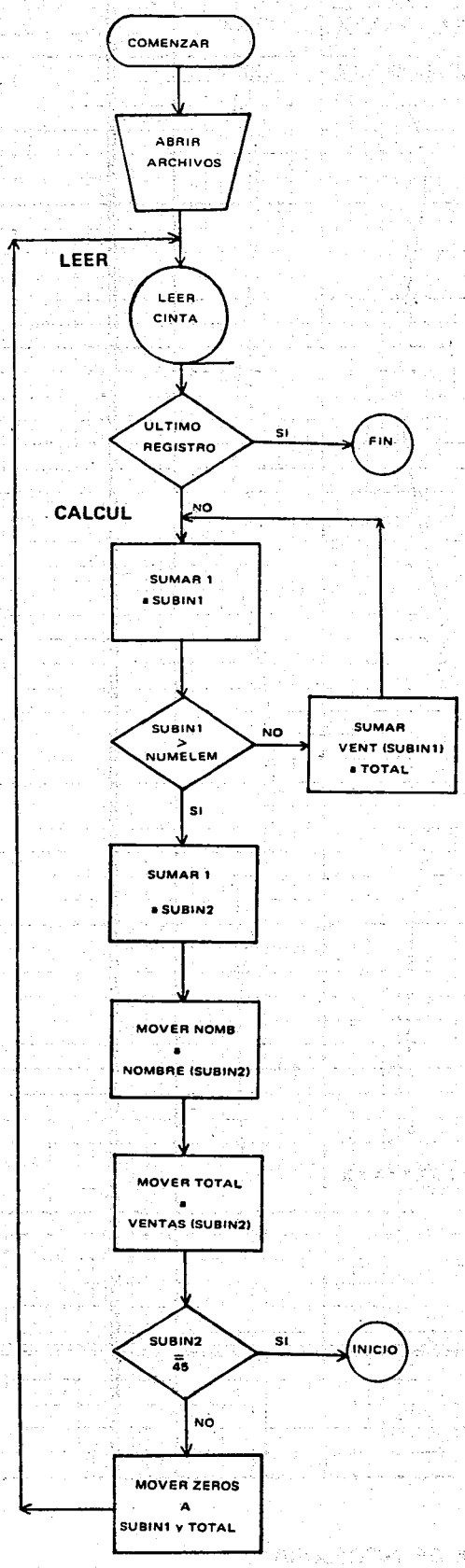
A fin de facilitar la comprensión de la solución del problema, se muestran los ordinogramas correspondientes:



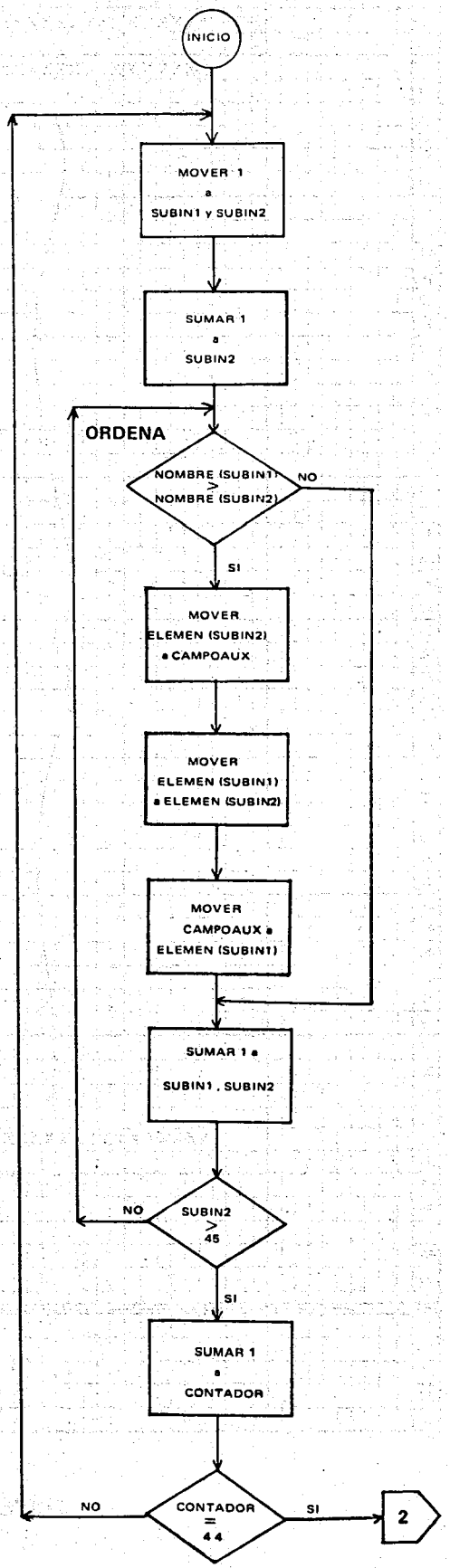
FORMATO DE SALIDA DE IMPRESORA

ORDINOGRAMA

CALCULO DEL NUMERO DE VENTAS DE CADA AGENTE Y LLENADO DE LOS ELEMENTOS DE LA TABLA2

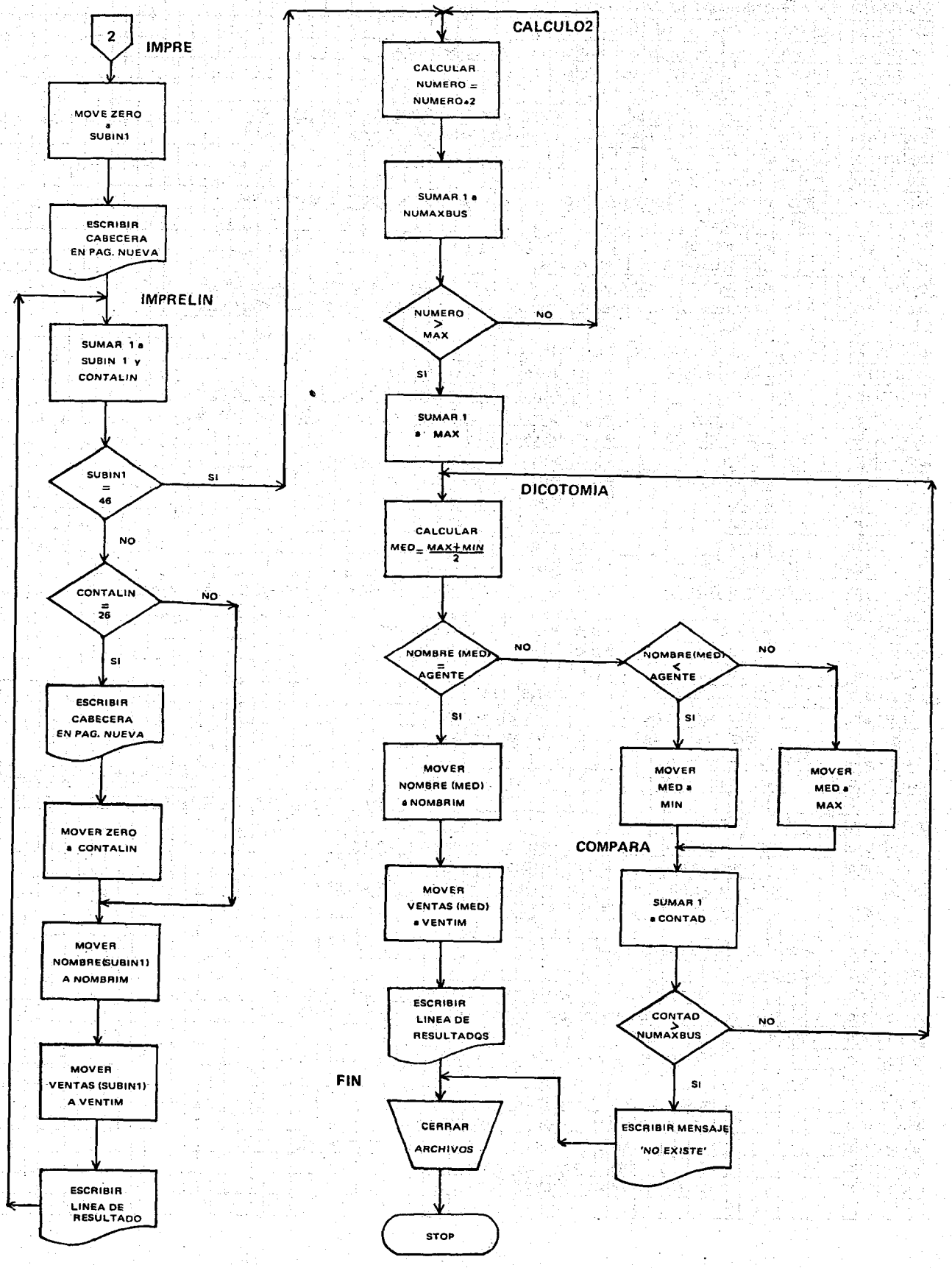


ORDENACION ALFABETICA DE LA TABLA2



IMPRESION DE AGENTES Y CANTIDAD VENDIDA

CALCULO DEL NUMERO MAXIMO DE BUSQUEDAS Y BUSQUEDA DE UN ELEMENTO DETERMINADO DE LA TABLA2



```
001010 ID DIVISION.
001020  PROGRAM-ID. BUSBINAR.
001030  AUTHOR. C. DE LA FUENTE.
001040 ENVIRONMENT DIVISION.
001050  CONFIGURATION SECTION.
001060   SPECIAL-NAMES.  C01 IS PAG-NUEVA.
001070  INPUT-OUTPUT SECTION.
001080   FILE-CONTROL.
001090       SELECT CINTA     ASSING SYS006-UT-2400-S.
001100       SELECT IMPRESORA ASSING SYS007-UR-1403-S.
001110 DATA DIVISION.
001120  FILE SECTION.
001130   FD CINTA   RECORDING U   LABEL RECORD OMITTED.
001140   01   TABLA1.
001150        02 NOMB        PIC X(35).
001160        02 NUMELEM     PIC 99.
001170        02 VENT   OCCURS 0 TO 31   DEPENDING NUMELEM  PIC 99.
001180   FD  IMPRESORA   RECORDING  F   LABEL RECORD OMITED.
001190   01  IMPRESO          PIC X(133).
001200  WORKING-STORAGE SECTION.
001210   77 CONTADOR         PIC 999     VALUE ZEROS.
001220   77 SUBIN1           PIC 99      VALUE ZEROS USAGE COMP.
001230   77 SUBIN2           PIC 99      VALUE ZEROS USAGE COMP.
001240   77 CAMPOAUX         PIC X(39).
002010   77 TOTAL            PIC 9(4)    VALUE ZEROS.
002020   77 CONTALIN         PIC 99      VALUE ZERO.
002030   77 MAX              PIC 99      VALUE 45.
002040   77 MIN              PIC 99      VALUE 1.
002050   77 MED              PIC 99.
002060   77 NUMAXBUS         PIC 99      VALUE 00.
002070   77 AGENTE           PIC X(35) VALUE 'GOMEZ MANRIQUE JUAN
002080-                                          '.
002090   77 NUMERO           PIC 99      VALUE 01.
002100   77 CONTAD           PIC 99      VALUE ZERO.
002110   01   TABLA2.
002120        02   ELEMEN     OCCURS 45.
002130             03 NOMBRE        PIC X(35).
002140             03 VENTAS        PIC 9(4).
002150   01 CABECERA.
002160      02 FILLER      PIC X(61) VALUE '                    RELACION ALFABETI
002170-                                'CA DE AGENTES Y CANTIDAD VENDIDA'.
002180   01 LINDETAL.
002190      02 FILLER      PIC X(14) VALUE SPACES.
002200      02 NOMBRIM     PIC X(35).
002210      02 FILLER      PIC X(4)  VALUE SPACES.
002220      02 VENTIM      PIC 9(4).

003010 PROCEDURE DIVISION.
003020  ABRIR.
003025                  OPEN INPUT CINTA OUTPUT IMPRESORA.
003030* CALCULO DEL NUMERO DE VENTAS DE CADA AGENTE.
003040  LEER.
003045                  READ CINTA AT END GO TO FIN.
003050  CALCUL.
003055                  ADD 1 TO SUBIN1
003060                  IF SUBIN1 NOT > NUMELEM
003065                                ADD VENT (SUBIN1) TO TOTAL
003070                                GO TO CALCUL
003075                  ELSE NEXT SENTENCE.
```

```
003080*  LLENADO DE LOS ELEMENTOS DE LA TABLA2.
003090                ADD 1 TO SUBIN2
003100                MOVE NOMB  TO NOMBRE (SUBIN2)
003110                MOVE TOTAL TO VENTAS (SUBIN2)
003120                IF SUBIN2 = 45  GO TO INICIO
003130                ELSE MOVE ZEROS TO SUBIN1 TOTAL
003140                     GO TO LEER.
003150*  ORDENACION ALFABETICA DE LA TABLA2.
003160    INICIO.
003165                MOVE 1 TO SUBIN1 SUBIN2
003166                ADD 1 TO SUBIN2.
003170    ORDENA.
003175                IF NOMBRE (SUBIN1) > NOMBRE (SUBIN2)
003180                     MOVE ELEMEN (SUBIN2) TO CAMPOAUX
003190                     MOVE ELEMEN (SUBIN1) TO ELEMEN (SUBIN2)
003200                     MOVE CAMPOAUX TO ELEMEN (SUBIN1)
003205                ELSE  NEXT SENTENCE.
003210                ADD  1 TO SUBIN1  SUBIN2
003220                IF SUBIN2 > 45   ADD 1  TO  CONTADOR
003230                     IF CONTADOR = 44   GO TO IMPRE
003240                     ELSE GO TO INICIO
003250                ELSE   GO TO   ORDENA.
004010*  IMPRESION DE AGENTES Y CANTIDAD VENDIDA.
004020    IMPRE.
004025                MOVE ZERO TO SUBIN1
004030                WRITE IMPRESO FROM CABECERA  AFTER PAG-NUEVA.
004040    IMPRELIN.
004045                ADD 1  TO  SUBIN1  CONTALIN
004050                IF  SUBIN1 = 46   GO TO  CALCULO2.
004060                    IF  CONTALIN = 26
004070                        WRITE IMPRESO FROM CABECERA  AFTER  PAG-NUEVA
004075                        MOVE  ZERO  TO  CONTALIN
004076                    ELSE  NEXT SENTENCE.
004080                MOVE NOMBRE (SUBIN1)  TO  NONBRIM
004090                MOVE VENTAS (SUBIN1)  TO  VENTIM
004100                WRITE IMPRESO  FROM  LINDETAL  AFTER 2
004110                GO TO IMPRELIN.
004120*  CALCULO DEL NUMERO MAXIMO DE BUSQUEDA EN LA TABLA2.
004130    CALCULO2.
004135                COMPUTE NUMERO = NUMERO * 2
004140                ADD 1 TO  NUMAXBUS
004150                IF NUMERO > MAX   ADD 1  TO  MAX
004160                ELSE  GO  TO  CALCULO2.
004170*  BUSQUEDA BINARIA DEL ELEMENTO DE LA TABLA2 QUE CONTIENE
004180*  AL AGENTE DE VENTAS  GOMEZ MANRIQUE JUAN.
004190    DICOTOMIA.
004200                    COMPUTE MED = (MAX + MIN) / 2
004210                IF NOMBRE (MED) = AGENTE
004220                     MOVE  NOMBRE (MED)  TO  NOMBRIM
004230                          MOVE VENTAS (MED)  TO  VENTIM
004240                          WRITE IMPRESO FROM LINDETAL AFTER PAG-NUEVA
005010                          GO TO FIN
005020                ELSE
005025                     IF NOMBRE (MED) < AGENTE
005030                          MOVE MED  TO  MIN
005040                     ELSE  MOVE MED  TO  MAX.
005050    COMPARA.
005055                ADD 1 TO CONTAD
005056                IF CONTAD > NUMAXBUS
005060                     DISPLAY  '  'NO EXISTE EN LA TABLA'  AGENTE
005070                ELSE  GO  TO  DICOTOMIA.
005080    FIN.
005085                CLOSE CINTA  IMPRESORA
005090                STOP RUN.
```

DESCRIPCION DEL PROGRAMA

– CALCULO DEL NUMERO DE VENTAS DE CADA AGENTE

Comienza la ejecución del programa con la apertura del archivo en cinta que contiene los 45 registros, cada uno de los cuales se denomina TABLA1, siendo su contenido el nombre de cada agente junto con el número de ventas de cada día. A continuación, lee el primer registro y comienza el cálculo del número de ventas del primer agente, para lo cual es necesario el empleo de un campo que haga de subíndice, SUBIN1, el cual se irá incrementando de unidad en unidad para poder ir recorriendo el número variable de elementos, en los que se encuentra la venta de cada día, e ir sumando su contenido a un campo de trabajo denominado TOTAL. Este cálculo se repite tantas veces como días de ventas haya tenido el agente, con lo que al final se habrá acumulado, en el campo TOTAL, el número de automóviles del primer agente.

– LLENADO DE LA TABLA2

Calculadas las ventas de dicho agente, se transfieren los apellidos y nombre, junto con las ventas efectuadas, acumuladas en TOTAL, al primer elemento de otra tabla denominada TABLA2, para lo cual se emplea un nuevo subíndice: SUBIN2, inicializado previamente a cero e incrementándole en una unidad para indicar el primer elemento de dicha tabla.

Finalizada la transferencia, se vuelve a leer un nuevo registro de la cinta y se repiten todas las operaciones citadas anteriormente, hasta que el valor de SUBIN2 sea igual a 45, en cuyo caso, quiere decir que se han procesado los 45 registros que contiene la cinta y se encuentra llena la TABLA2 con el nombre de cada agente junto con la cantidad de automóviles vendidos.

– ORDENACION ALFABETICA DE LA TABLA2

Para la ordenación alfabética de TABLA2 se emplean los campos SUBIN1 y SUBIN2, inicializándolos previamente con el valor uno, se suma una unidad a SUBIN2, con el fin de poder efectuar la comparación entre el valor del contenido del elemento que ocupa el primer lugar de la tabla con el que ocupa el segundo; si de dicha comparación resulta que el valor del primer elemento es mayor que el del segundo, se efectuan las siguientes operaciones:

- Se trasfiere el contenido del segundo elemento de la tabla a un campo de trabajo, CAMPOAUX.
- Se trasfiere el contenido del primer elemento de la tabla al lugar ocupado por el segundo elemento.
- Se trasfiere el contenido del CAMPOAUX al primer elemento de la tabla.

Si el resultado de la comparación es que el contenido del segundo elemento es mayor que el del primero, el contenido de este elemento permanece en su lugar, se incrementan SUBIN1 y SUBIN2 en una unidad para poder comparar el contenido del segundo elemento con el tercero; repitiéndose la operación 45 veces, con lo que se consigue que el contenido del elemento cuyo valor sea el mayor de la tabla pase a ocupar el último lugar.

Realizando una vez más las operaciones anteriores, se logrará situar el valor inmediatamente inferior al valor mayor contenido en la tabla, en el penúltimo lugar. Para que la tabla quede ordenada en forma ascendente, hay que repetir las operaciones citadas cuarenta y cuatro veces, es decir, el número de elementos de la tabla menos uno, controlando dicho número por medio del campo CONTADOR (ver otra forma más completa de ordenar TABLA2 en el ejercicio de la pág. 284).

– IMPRESION DE AGENTES Y VENTAS

El tercer paso del programa consiste en la impresión de una cabecera y una serie de líneas de detalle, conteniendo cada una el nombre del agente de ventas y la cantidad de automóviles vendida, lo cual se encuentra almacenado en TABLA2.

Para realizar este trabajo, se vuelve a emplear el subíndice SUBIN1, para ir extrayendo sucesivamente el contenido de cada elemento de la TABLA2 y transferirlo al área de salida y de ahí imprimirlo. Se utiliza también el campo CONTALIN, para controlar que el número de líneas de detalle a imprimir en cada hoja de la impresora sea de veinticinco y que, cada vez que salte a una nueva página vuelva a escribir la cabecera según formato de la pág. 259.

– CALCULO DEL NUMERO MAXIMO DE BUSQUEDAS NECESARIAS PARA LOCALIZAR UN ELEMENTO EN LA TABLA2

El número máximo de búsquedas necesarias para localizar a un elemento de la tabla que contenga un valor determinado, se obtiene por la expresión:

$$2^x > \text{número de elementos de la tabla} \geqslant 2^{x-1}$$

en la que x indica el número máximo de búsquedas, es decir, si la tabla contiene 235 elementos el valor de x será igual a ocho, ya que:

$$2^8 > 235 > 2^7$$

Para efectuar dicho cálculo, se emplea el campo de trabajo NUMERO, el cual al irle sucesivamente multiplicando por 2, llega un momento en que su valor es mayor que el número de elementos de la tabla (MAX), en cuyo caso el número máximo de búsquedas será igual al número de veces que se ha multiplicado NUMERO por la cifra 2, las cuales se han ido contabilizando por medio del campo NUMAXBUS.

El algoritmo empleado en el programa para realizar el citado cálculo (líneas 004130 a 004160) es independiente del número de elementos que contenga la tabla, es decir, puede aplicarse en tablas con cualquier número de elementos. Sin embargo, si el programador, como en este caso, conoce el número de elementos de que consta la tabla, puede calcular de acuerdo con la expresión citada, el número de busquedas e introducirlo en el campo NUMAXBUS, con lo que evitaría tener que incluir dicho algoritmo dentro del programa.

- BUSQUEDA BINARIA DE UN ELEMENTO EN LA TABLA2

La operación de búsqueda comienza calculando el valor del campo MED, para conocer el número de los elementos en que se ha de investigar sucesivamente según la fórmula:

$$MED = (MAX + MIN) / 2$$

El valor inicial del campo MAX, con el que comienza la búsqueda en la tabla, es igual al número de elementos de la tabla incrementado en una unidad, mientras que el valor inicial del campo MIN es uno, ya que de no hacerlo así, no sería posible la localización de cualquier elemento, pues se llevaría a cabo una búsqueda menos de las necesarias para investigar en todos los elementos de la tabla, y podía suceder que el elemento de la tabla que quedara sin investigar fuera el que cumpliera la condición (21).

Con la primera aplicación de dicha fórmula se obtendrá el número del elemento que ocupa el lugar medio de la tabla. Compara el contenido de dicho elemento con el contenido que se desea buscar; si el resultado de la comparación es de igualdad, quiere decir que existe en la tabla un elemento que contiene el citado valor y por tanto terminará la búsqueda; por el contrario, si el contenido del elemento medio de la tabla es menor que el contenido a buscar, se transfiere el contenido del campo MED al campo MIN y se vuelve a calcular el número del elemento de la tabla en que de nuevo se ha de investigar por la fórmula anterior, siendo el valor de dicho número el que ocupa el lugar medio de una de las dos mitades en que quedó dividida la tabla. Si, al realizar la comparación, el resultado de la misma es que el contenido del elemento es mayor que el contenido a buscar, se transfiere el valor del campo MED a MAX y se vuelve a calcular el valor de MED.

El ciclo descrito se repite un número de veces que no puede ser superior al contenido del campo NUMAXBUS, hasta la localización del elemento, en cuyo caso termina la búsqueda. Puede suceder que la búsqueda finalice porque no exista en dicha tabla un elemento que cumpla la condición de que su contenido sea igual a un determinado valor, en cuyo caso, la secuencia de ejecución del programa continúa en la declaración DISPLAY y termina la ejecución del mismo.

---

(21) Aunque el valor inicial del campo MAX que figura en el programa es 45, sin embargo, se incrementa en una unidad (línea 004150) antes de iniciar la búsqueda en la tabla (línea 004200), para así poder aplicar la fórmula de la línea 004200.

3.3.- Búsqueda SECUENCIAL (Con índices)

La búsqueda en forma secuencial de una tabla en que se ha especificado la opción INDEXED BY nombre–de–índice, puede hacerse: utilizando subíndices, o bien, en forma más eficaz y rápida, utilizando los nombres–de–índice.

Para que el Sistema pueda realizar la búsqueda secuencial en una tabla indexada, es necesario especificar dentro del programa:

- Una declaración SET con la que dar un valor inicial al índice. Dicho valor será el número del elemento a partir del cual comenzará la búsqueda.

- Una declaración SEARCH por la que el Sistema irá incrementando el citado índice de unidad en unidad, a la vez que realiza la búsqueda en la tabla.

El programa que se muestra a continuación resuelve el problema cuyo enunciado se especifica en la página 257, y va a servir como ejemplo para hacer la descripción de los dos métodos de búsqueda en tablas empleando índices. Aunque en él se utilice la búsqueda binaria, sin embargo, introduciendo leves modificaciones puede adaptarse para realizar la búsqueda en forma secuencial.

Comparando esta solución con la de las páginas 262 y 263, hay que hacer notar las diferencias siguientes:

- Los campos de datos SUBIN1 y SUBIN2 que fueron empleados como subíndices, no se describen dentro de este programa por no ser necesario, ya que su función la realizan los índices: BETA el cual indexa a la tabla denominada con el nombre genérico VEN (línea 001170), y ALFA que indexa a la tabla denominada con el nombre genérico ELEMEN.

- No se describe el campo NUMAXBUS, por no ser necesario tener que realizar el cálculo del número máximo de búsquedas necesarias para localizar a un determinado elemento de una tabla, ya que cuando se emplea la declaración SEARCH ALL, obliga al Sistema a realizarlo. Por tanto, si no es necesario realizar dicho cálculo, tampoco será necesario describir los campos que intervenían en él : MAX, MED, MIN y CONTAD.

- Se emplean las declaraciones SET y SEARCH ALL, así como la cláusula USAGE INDEX, las cuales permiten que el número de inscripciones dentro del programa sea apreciablemente inferior a las hechas en el programa anterior, además de agilizar su ejecución.

La descripción del programa se realiza junto con el estudio de las especificaciones de cada una de las declaraciones citadas. Esto servirá para aclarar aquellos conceptos que no son fácilmente comprensibles para el lector que examina por primera vez el programa.

```
000010 ID DIVISION.
000020  PROGRAM-ID. ORBUSBIN.
000030  AUTHOR. C. DE LA FUENTE.
000040 ENVIRONMENT DIVISION.
000050  CONFIGURATION SECTION.
000060   SPECIAL-NAMES.  C01  IS PAG-NUEVA.
000070   INPUT-OUTPUT SECTION.
000080   FILE-CONTROL.
000090     SELECT CINTA  ASSIGN SYS006-UT-2400-S.
000100     SELECT IMPRESORA ASSIGN SYS007-UR-1403-S.
000110 DATA DIVISION.
000120  FILE SECTION.
000130   FD  CINTA RECORDING U
000135             LABEL RECORD OMITTED
000136             DATA RECORD TABLA1.
000140   01 TABLA1.
000150      02 NOMB          PIC X(35).
000160      02 NUMELEM       PIC 99.
000170      02 VEN  OCCURS 0  TO 31
000175             DEPENDING ON NUMELEM INDEXED BY BETA
000180                       PIC 99.
000190   FD  IMPRESORA RECORDING F
000195             LABEL RECORD  OMITTED
000196             DATA RECORD IMPRESO.
000200   01 IMPRESO          PIC X(133).
000210  WORKING-STORAGE SECTION.
000220   77 INDICE          USAGE  INDEX.
000230   77 CONTADOR        PIC 999   VALUE ZEROS.
000240   77 CAMPOAUX        PIC X(39).
000250   77 TOTAL           PIC 9(4)  VALUE ZEROS.
000260   77 CONTALIN        PIC 9(4)  VALUE ZEROS.
000270   77 AGENTE          PIC X(35) VALUE 'GOMEZ MANRIQUE JUAN
000280-                                     '.
000290   01  TABLA2.
000300       02  ELEMEN  OCCURS  45
000305                   ASCENDING KEY IS NOMBRE  INDEXED BY  ALFA.
000310           03  NOMBRE    PIC X(35).
000320           03  VENTAS    PIC 9(4).
000330   01  CABECERA.
000340       02  FILLER  PIC X(61) VALUE '                RELACION ALFABETI
000350-                                 'CA DE AGENTES Y CANTIDAD VENDIDA'.
000360   01  LINDETAL.
000370       02  FILLER  PIC X(14) VALUE SPACES.
000380       02  NOMBRIM PIC X(35).
000390       02  FILLER  PIC X(4)  VALUE SPACES.
000400       02  VENTIM  PIC 9(4).
000405*
```

```
000216 0      PROCEDURE DIVISION.
000217 0        ABRIR.
000217 5                    OPEN INPUT CINTA  OUTPUT IMPRESORA.
000218 0*       CALCULO DEL NUMERO DE VENTAS DE CADA AGENTE.
000219 0                    SET ALFA BETA  TO  ZERO.
000220 0        LEER.
000220 5                    READ CINTA AT END  GO TO FIN.
000221 0        CALCUL.
000221 5                    SET BETA  UP  BY 1
000222 0                    IF  BETA NOT > NUMELEM
000222 5                        ADD VEN (BETA)  TO  TOTAL
000223 0                        GO TO CALCUL
000224 0                    ELSE  NEXT SENTENCE.
000301 0*       LLENADO DE LOS ELEMENTOS DE LA TABLA2.
000302 0                    SET ALFA  UP  BY 1
000303 0                    MOVE NOMB  TO  NOMBRE (ALFA)
000304 0                    MOVE TOTAL TO  VENTAS (ALFA)
000305 0                    IF ALFA = 45
000305 5                        GO TO INICIO
000306 0                     ELSE  SET BETA TO ZERO
000307 0                           MOVE ZERO  TO TOTAL
000308 0                           GO TO LEER.
000309 0*       ORDENACION ALFABETICA DE LA TABLA2.
000310 0        INICIO.
000310 5                    PERFORM  ORDENA  THRU  SALIDA
000310 6                             VARYING CONTADOR FROM 1 BY 1 UNTIL CONTADOR = 45
000312 0                                 AFTER ALFA  FROM 1 BY 1 UNTIL ALFA = 45
000313 0                    GO TO IMPRE.
000314 0        ORDENA.
000314 5                    IF NOMBRE (ALFA + 1) > NOMBRE (ALFA) GO TO SALIDA
000315 0                    ELSE MOVE ELEMEN (ALFA + 1)  TO CAMPOAUX
000316 0                         MOVE ELEMEN (ALFA)  TO ELEMEN (ALFA + 1)
000317 0                         MOVE CAMPOAUX  TO  ELEMEN (ALFA).
000318 0        SALIDA.  EXIT.
000319 0*       IMPRESION DE AGENTES Y CANTIDAD VENDIDA.
000320 0        IMPRE.
000320 5                    SET  ALFA  TO 46
000321 0                    SET  INDICE  TO  ALFA
000322 0                    WRITE IMPRESO  FROM CABECERA AFTER PAG-NUEVA
000323 0                    PERFORM IMPRELIN
000323 5                            VARYING ALFA FROM 1 BY 1 UNTIL  ALFA = INDICE
000325 0                    GO TO BUSQUEDA.
000401 0        IMPRELIN.
000401 5                    ADD 1  TO CONTALIN.
000402 0                    IF CONTALIN = 26
000403 0                        WRITE IMPRESO FROM CABECERA AFTER PAG-NUEVA
000404 0                        MOVE ZERO  TO CONTALIN
000404 5                    ELSE  NEXT SENTENCE.
000405 0                    MOVE NOMBRE (ALFA) TO NOMBRIM
000406 0                    MOVE VENTAS (ALFA) TO VENTIM
000407 0                  WRITE IMPRESO FROM LINDETAL AFTER 2.
000408 0*       BUSQUEDA EMPLEANDO SEARCH ALL DEL ELEMENTO DE LA TABLA2
000409 0*          QUE CONTIENE AL AGENTE  GOMEZ MANRIQUE JUAN.
000410 0        BUSQUEDA.
000411 0                    SEARCH ALL ELEMEN  AT END GO TO NOEXIS
000412 0                        WHEN NOMBRE (ALFA) = AGENTE
000413 0                             MOVE NOMBRE (ALFA) TO NOMBRIM
000414 0                             MOVE VENTAS (ALFA) TO VENTIM
000415 0                             WRITE IMPRESO FROM LINDETAL AFTER PAG-NUEVA
000416 0                             GO TO  FIN.
000417 0        NOEXIS.
000417 5                    DISPLAY  AGENTE  'NO EXISTE EN LA TABLA'.
000418 0        FIN.
000418 5                    CLOSE CINTA  IMPRESORA
000419 0                    STOP RUN.
```

3.3.1.- Declaración SET

FUNCION: Inicializar con un determinado valor los nombres–de–índice que se especificaron dentro de una cláusula OCCURS.

FORMATO 1:

```
SET { nombre–de–índice–1   [ nombre–de–índice–2 ]... }  TO  { nombre-de-índice-3 }
    { identificador–1      [ identificador–2    ]... }      { identificador–3   }
                                                            { literal           }
```

El efecto de esta declaración es trasferir al operando u operandos que siguen a SET el contenido del operando que sigue a la palabra TO, es decir, es una transferencia realizada en sentido contrario a como la realizaba la declaración MOVE.

El siguiente cuadro muestra las transferencias permitidas dentro de una declaración SET entre nombres–de–índice, campos–de–índice (USAGE INDEX), identificadores y literales que figuran en el FORMATO 1 de dicha declaración.

CUADRO DE TRANSFERENCIAS PERMITIDAS

PRIMER OPERANDO \ SEGUNDO OPERANDO		Identificador–3		Nombre–de–índice–3	Literal
		campo-de-índice	nombre-de-datos		
Identificador–1 Identificador–2...	Campo–de–índice (USAGE INDEX)	X		X	
	nombre–de–datos			X	
Nombre–de–índice–1 Nombre–de–índice–2 ...		X	X	X	X

Los cuadros en blanco indican que no es posible la trasferencia del contenido del operando que figura a continuación de TO al o los operandos que siguen a SET.

A un nombre–de–índice puede transferirse cualquier valor perteneciente a un campo–de–índice,literal, nombre–de–índice, etc.:

```
002190     SET ALFA BETA  TO ZERO
.....      ..........................................................
003200     SET ALFA  TO  46
003210     SET INDICE TO ALFA
```

La declaración SET de la línea 002190 transfiere el valor cero a los nombres-de-índice ALFA y BETA pertenecientes a TABLA1 y TABLA2. La de la línea 003200, el valor 46 lo transfiere al nombre-de-índice ALFA,y la de la línea 003210 transfiere el valor del nombre-de-índice ALFA, es decir, 46, al campo-de-índice denominado INDICE.

FORMATO 2:

```
SET nombre-de-índice-1 [nombre-de-índice-2] ... { UP BY   } { identificador }
                                                { DOWN BY } { literal     }
```

Este formato representa una de las formas de ir incrementando o decrementando el valor inicial dado a un nombre-de-índice,ya que no se puede modificar dicho valor empleando declaraciones aritméticas.

Cada vez que se ejecuta la declaración SET, el contenido de los nombres-de-índice que siguen a dicha declaración se incrementan (UP BY) o decrementan (DOWN BY) en el valor contenido en el Identificador o en el literal que siga a estas dos opciones.

El Identificador tiene la limitación de que no puede ser un campo-de-índice (USAGE INDEX)

EJEMPLO:

Las declaraciones siguientes calculan el número de automóviles vendidos por cada agente durante un mes:

```
002160  PROCEDURE DIVISION.
002170    ABRIR.
002175                    OPEN INPUT CINTA  OUTPUT IMPRESORA.
002180*   CALCULO DEL NUMERO DE VENTAS DE CADA AGENTE.
002190                    SET ALFA BETA  TO  ZERO.
002200    LEER.
002205                    READ CINTA AT END  GO TO FIN.
002210    CALCUL.
002215                    SET BETA  UP BY 1
002220                    IF  BETA NOT > NUMELEM
002225                        ADD VEN (BETA)  TO  TOTAL
002230                        GO TO CALCUL
002240                    ELSE  NEXT SENTENCE.
```

una vez inicializado el nombre-de-índice, BETA,con el valor cero (línea 002190), a continuación, por medio de la declaración SET de la línea 002215,se va incrementando dicho nombre-de-índice de unidad en unidad para recorrer todos los elementos de TABLA1, a fin de ir acumulando su contenido en el campo TOTAL.

3.3.2.- Cláusula USAGE INDEX

FUNCION: Describir un campo de datos que generará el Sistema.

FORMATO:

[USAGE IS] INDEX

Un campo-de-índice, al ser generado internamente por el Sistema, no necesita ser descrito con la cláusula PICTURE.

EJEMPLO.

Las declaraciones siguientes sirven para imprimir el contenido de la TABLA2, una vez que fue ordenada alfabéticamente.

```
001210   WORKING-STORAGE SECTION.
001220     77   INDICE        USAGE   IS   INDEX.
.....  ..........................................
003190*  IMPRESION DE AGENTES Y CANTIDAD VENDIDA.
003200   IMPRE.
003205             SET  ALFA  TO 46
003210             SET  INDICE  TO  ALFA
003220             WRITE  IMPRESO  FROM CABECERA AFTER PAG-NUEVA
003230             PERFORM IMPRELIN
003235                     VARYING ALFA FROM 1 BY 1 UNTIL  ALFA = INDICE
003250           GO TO BUSQUEDA.
004010   IMPRELIN.
004015             ADD 1  TO CONTALIN.
004020             IF CONTALIN = 26
004030                  WRITE IMPRESO FROM CABECERA AFTER PAG-NUEVA
004040                  MOVE ZERO  TO CONTALIN
004045             ELSE   NEXT SENTENCE.
004050             MOVE NOMBRE  (ALFA)  TO NOMBRIM
004060             MOVE VENTAS  (ALFA)  TO VENTIM
004070           WRITE IMPRESO FROM LINDETAL AFTER 2.
```

La misión del campo-de-índice, que en este ejemplo se ha denominado INDICE, es almacenar el valor del nombre-de-índice ALFA, es decir, 46, para posteriormente intervenir en una declaración PERFORM con la condición de relación: UNTIL ALFA = INDICE, lo cual tiene la ventaja de que, por estar los dos campos almacenados internamente con el mismo formato (binario puro), no necesitan convertirse a un mismo formato para poder efectuar la comparación, con lo que se gana en rapidez de ejecución.

- La cláusula USAGE IS INDEX puede describirse con cualquier número de nivel.

- A un campo-de-índice sólo puede hacerse referencia, dentro de una declaración SEARCH, SET o en una declaración que contenga una condición de relación.

3.3.3.- Declaración SEARCH

FUNCION: Llevar a cabo la búsqueda en forma secuencial de una tabla, hasta que el contenido de un elemento de la misma satisfaga una determinada condición.

FORMATO:

```
SEARCH identificador-1 [ VARYING { nombre-de-índice  } ]
                                  { identificador-2  }
  [AT END declaración-imperativa-1]
    WHEN condición-1 { declaración-imperativa-2 }
                     { NEXT SENTENCE            }
  [WHEN condición-2 { declaración-imperativa-3 } ]...
                    { NEXT SENTENCE            }
```

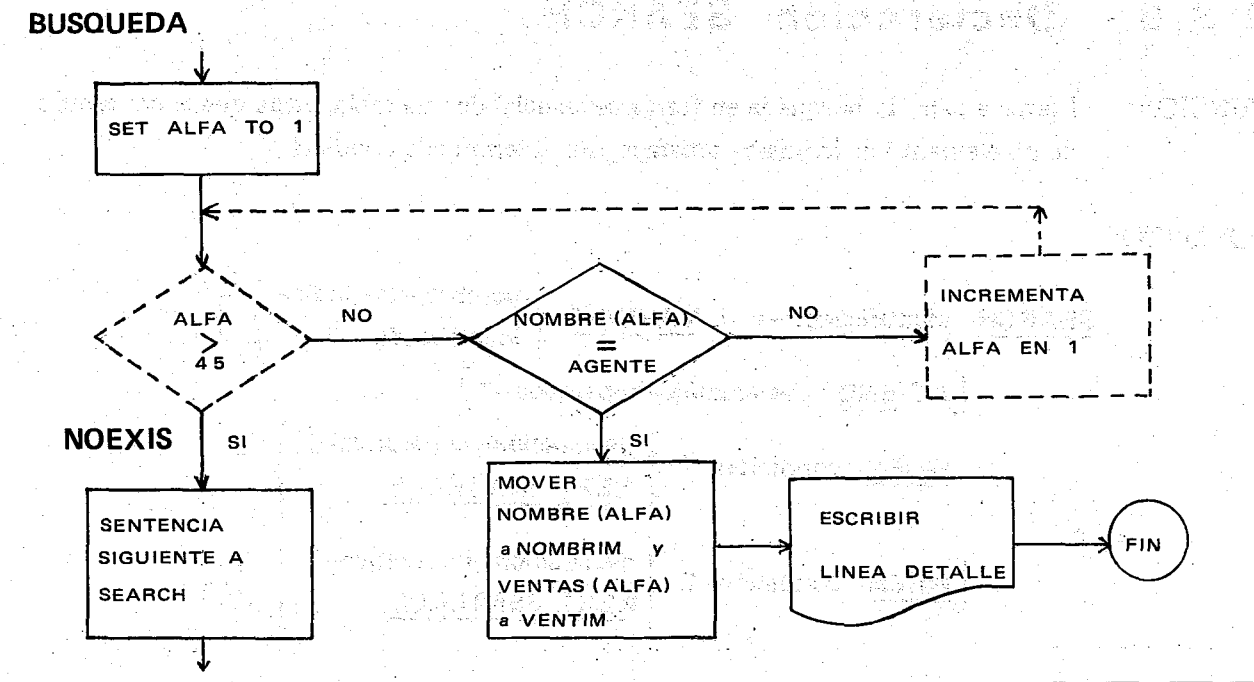
Es recomendable la búsqueda secuencial empleando, para referirse a los elementos de la tabla, la indexación en vez de la subindicación, por realizarse aquella de forma más rápida. Ello es debido a que empleando subindicación, el Sistema, previa a cada búsqueda, debe hacer la operación de conversión del valor del subíndice a formato binario puro; mientras que empleando la indexación no es necesaria dicha conversión, ya que el valor de los nombres-de-índice se almacenan automáticamente en la memoria en dicho formato.

Para poder efectuar la búsqueda secuencial empleando la declaración SEARCH, es necesario que la tabla esté indexada por un nombre-de-índice (línea 002055). Sin embargo, no es necesario que esté ordenada, y por tanto puede prescindirse de especificar las opciones ASCENDING o DESCENDING.

La declaración SEARCH del ejemplo siguiente, efectúa la búsqueda secuencial en TABLA2, del elemento que contiene al agente GOMEZ MANRIQUE JUAN:

```
002020     77  AGENTE      PIC X(35)
002030                     VALUE 'GOMEZ MANRIQUE JUAN                  '.
002040     01  TABLA2.
002050         02 ELEMEN    OCCURS 45
002055                     ASCENDING KEY NOMBRE    INDEXED BY ALFA.
002060             03 NOMBRE     PIC X(35).
002070             03 VENTAS     PIC 9(4).
......     ..........................................................
004100     BUSQUEDA.
004110             SET ALFA TO 1.
004120             SEARCH ELEMEN AT END GO TO NOEXIS
004130                 WHEN NOMBRE (ALFA) = AGENTE
004140                     MOVE NOMBRE (ALFA) TO NOMBRIM
004150                     MOVE VENTAS (ALFA) TO VENTIM
004160                     WRITE IMPRESO FROM LINDETAL AFTER PAG-NUEVA
004170                     GO TO FIN.
004180     NOEXIS.     DISPLAY  AGENTE  'NO EXISTE EN LA TABLA'.
```

El siguiente ordinograma facilita la comprensión de la forma como el ordenador ejecuta la búsqueda secuencial de TABLA2.



NOTA: Las líneas de guiones indican las acciones que automáticamente ejecuta el compilador sin que el programador tenga que especificarlas dentro del programa.

Antes de efectuar la búsqueda en la tabla, es preciso inicializar el nombre–de–índice ALFA, por medio de la declaración SET, con un valor, que será el número del elemento de la tabla en que comienza dicha búsqueda. En este ejemplo, la declaración SEARCH comienza la búsqueda en el primer elemento, ya que ALFA se inicializó con el valor uno, probando la condición que sigue a WHEN, es decir, si NOMBRE (ALFA) = AGENTE; caso de que la prueba resulte ser cierta, se ejecuta la declaración o las declaraciones que siguen a dicha condición hasta el primer punto (líneas 004140 a 004170). Si la prueba resulta ser falsa, se incrementa, automáticamente, en una unidad el nombre–de–índice ALFA, para repetir el ciclo citado con el segundo elemento.

Si hecha la prueba de dicha condición con todos los elementos de la tabla, no se encuentra a alguno de ellos que la cumpla, quiere decir que el valor buscado no existe en ningún elemento de la tabla, por lo que se da por finalizada la búsqueda. En este momento la secuencia de ejecución del programa pasaría a ejecutar la declaración o declaraciones que siguen a AT END hasta encontrar la palabra WHEN, omitiría las declaraciones que siguen a dicha palabra hasta encontrar el primer punto, continuando en la primera declaración a partir de él, es decir, continuaría la ejecución del programa en el procedimiento NOEXIS (línea 004180).

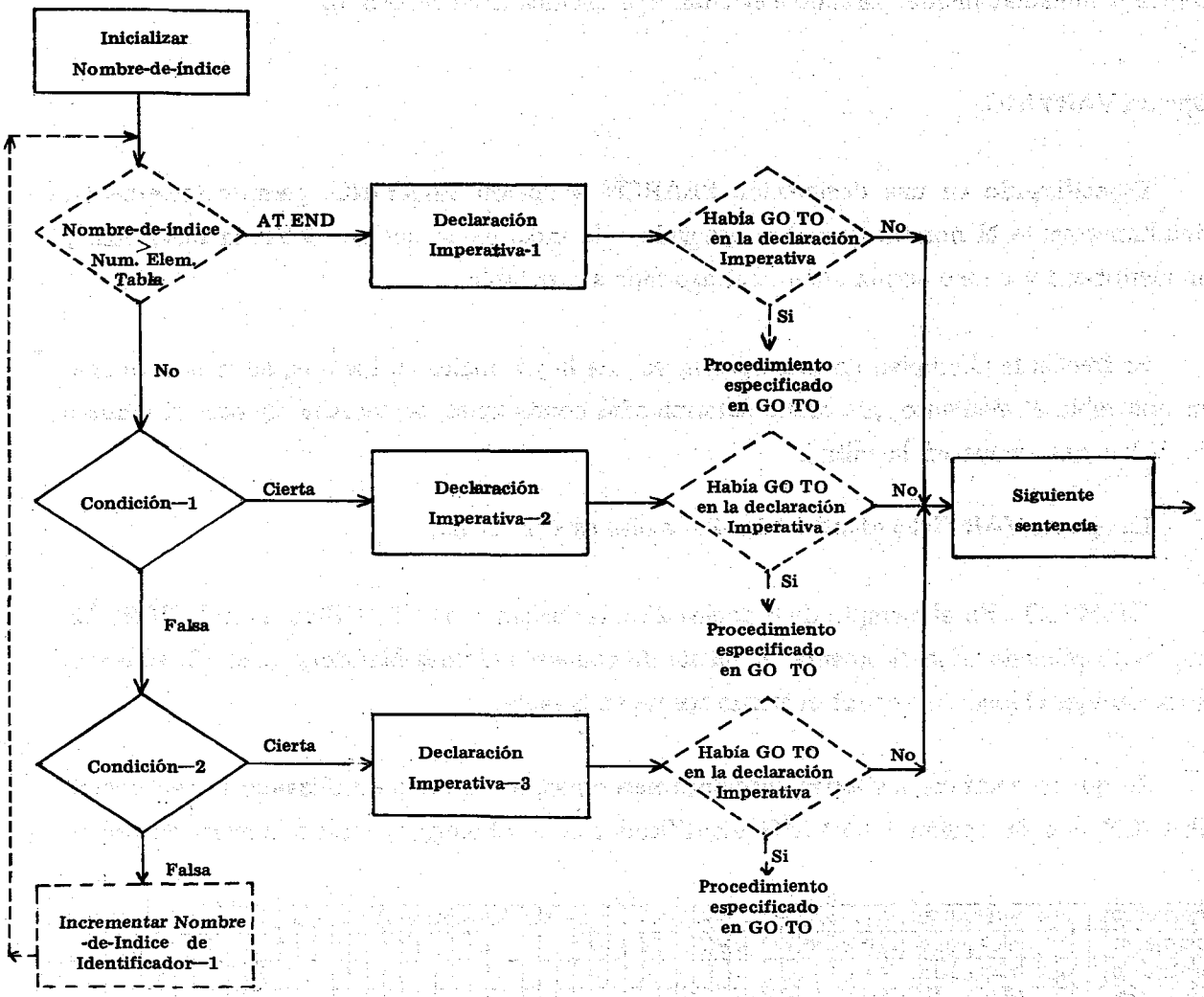
Si por cualquier motivo, antes de ejecutar la declaración SEARCH, se ha iniciado ALFA con un valor mayor que el número de elementos de que consta la tabla, no se realiza ninguna búsqueda y la secuencia de ejecución del programa pasaría a ejecutar la siguiente sentencia a SEARCH, es decir, bifurcaría a NOEXIS, que es donde comienza la siguiente sentencia.

Relacionando el formato de la declaración SEARCH con el ejemplo de la página 273:

– Indentificador (ELEMEN) es el nombre genérico con el que se denominan todos los elementos de la tabla. No puede describirse con nivel 01.

- Nombre–de–índice, ALFA, debe especificarse a continuación de INDEXED BY. No puede iniciarse con un valor cero o negativo.
- Condición–1, NOMBRE(ALFA)= AGENTE, puede ser de cualquier tipo de los descritos en la declaración IF.
- Puede especificarse una o varias palabras WHEN seguidas de una condición simple o compuesta.

La figura siguiente muestra los pasos que realiza el ordenador al ejecutar una declaración SEARCH, en la que se han especificado dos palabras WHEN, seguidas cada una de una condición:



Las líneas de guiones indican las acciones que ejecuta automáticamente el compilador sin tener el programador que codificar dentro del programa.

Las acciones que lleva a cabo el Sistema para ejecutar la declaración SEARCH en la que se han especificado dos palabras WHEN son las siguientes:

a) Se prueba cada condición o condiciones de la opción WHEN en el mismo orden de su escritura.
b) Si no se ha satisfecho ninguna de las condiciones, se incrementa el nombre–de–índice de identificador–1 para poder hacer referencia al siguiente elemento de la tabla y se repite el paso a).

c) Si después de la evaluación, se ha satisfecho una de las condiciones que siguen a WHEN, la búsqueda concluye, pasando a ejecutar la declaración imperativa asociada con tal condición. El nombre–de–índice, en ese momento, contendrá el número del elemento de la tabla que satisfizo la condición.

d) Si se alcanza el final de la tabla sin haber satisfecho ninguna condición de las que siguen a WHEN, la búsqueda concluye y, por tanto, si se ha especificado la opción AT END, se ejecuta la declaración imperativa que sigue a dicha opción. Si se omite AT END, la secuencia de ejecución del programa pasa a la sentencia siguiente.

Puede suceder que al comienzo de la búsqueda, el valor del nombre–de–índice que indexa a identificador–1 sea superior al número máximo de elementos de la tabla, en cuyo caso la búsqueda concluye inmediatamente, pasando a ejecutar lo especificado en el paso d).

Opción VARYING.

Especificando en una declaración SEARCH la opción VARYING, permite incrementar simultáneamente el nombre-de-índice asociado a la tabla en la que se realiza la búsqueda, y un identificador u otro nombre-de-índice asociado a otra tabla.

Es frecuente el empleo de esta opción, ya que hay trabajos en los que, además de buscar en una tabla al elemento que reuna determinadas condiciones, se necesita conocer el número de orden que ocupa en la misma.

La opción **VARYING identificador-2** resuelve este problema.

EJEMPLO. En el ejemplo de la página 273, la declaración SEARCH busca en la TABLA2, que está ordenada alfabéticamente, al agente de comercio Gomez Manrique Juan, sin embargo, no se obtiene el lugar que ocupa el citado agente en la tabla.

Se quiere codificar las especificaciones necesarias, a fin de que utilizando la declaración SEARCH con la opción VARYING identificador-2, se obtenga el citado número de orden.

```
002020    01 AGENTE PIC X(35) VALUE 'GOMEZ MANRIQUE JUAN                  ».
002021    01 ELEMEN-NUM  PIC 99.
002040    01 TABLA2.
002050       02 ELEMEN  OCCURS 45  INDEXED BY ALFA.
002060          03 NOMBRE      PIC X(35).
002070          03 VENTAS      PIC 9(4).
.......  ..........................................................
004098       MOVE 1 TO ELEMEN-NUM
004099       SET ALFA TO 1.
004100 BUSQUEDA.
004110       SEARCH  ELEMEN  VARYING ELEMEN-NUM
004120          AT END  GO TO NOEXIS
004130          WHEN NOMBRE (ALFA) = AGENTE
004131               DISPLAY AGENTE 'OCUPA LA POSICION ' ELEMEN-NUM
004140               MOVE NOMBRE (ALFA) TO NOMBRIM
.......  ..........................................................
004180 NOEXIS.
```

Es también frecuente el empleo de la opción **VARYING nombre-de-índice,** cuando se trata de incrementar simultáneamente dicho índice, perteneciente a una tabla distinta de aquella en la que se esta realizando la búsqueda, y el índice de la tabla en la que se realiza dicha búsqueda.

EJEMPLO. Se dispone de dos tablas cuyo contenido es el siguiente:

En la primera de ellas, se encuentran los nombres de los diferentes modelos de automóviles que fabrica la empresa; y en la segunda tabla el precio de cada uno de los modelos que figuran en la tabla primera.

Ambas tablas estan constituidas de tal forma que la posición que ocupan los diferentes modelos en una tabla se corresponde con la posición que ocupan los precios en la otra tabla.

Llevar a cabo la codificación necesaria para conocer cuál es el precio que corresponde al modelo: AUTI - 1200TURBO.

```
010   01 TABLA-MODELOS.
020      02 MODELOS-AUTO  OCCURS 50  INDEXED  BY INDICE1.
030         03 MODELO      PIC X(14).
040   01 TABLA-PRECIOS.
050      02 PRECIO-MODELOS OCCURS 50 INDEXED  BY INDICE2.
060         03 PRECIO      PIC 9(7).
....  ..........................................................
200   BUSQUEDA.
210      SET  INDICE1, INDICE2  TO 1
220      SEARCH  MODELOS-AUTO   VARYING INDICE2
230         AT END  DISPLAY 'NO EXISTE EL MODELO BUSCADO'
240         WHEN  MODELO (INDICE1) = 'AUTI-1200TURBO'
250            DISPLAY 'PRECIO DEL MODELO = '   PRECIO (INDICE2).
```

TABLAS DEFINIDAS CON MAS DE UN INDICE.

Las tablas que hasta ahora se han utilizado en los diferentes programas estaban definidas con un solo índice, que el Sistema empleaba para recorrer los diferentes elementos de la tabla, a fin de investigar su contenido; sin embargo, en la definición de una tabla, pueden utilizarse varios índices. En este caso es necesario conocer cuál de ellos es el que va a utilizar el Sistema para la búsqueda en la tabla.

El empleo de uno u otro índice para realizar la búsqueda está en función de que se especifique o no, en la declaración SEARCH, la opción VARYING, es decir:

a) Si no se especifica la opción VARYING, la búsqueda en la tabla se realizará utilizando el primer índice de los que figuran en la cláusula INDEXED BY.

b) Si se especifica la opción VARYING junto con el nombre de uno de los índices, que figuran en la cláusula INDEXED, éste será el que utilice para realizar la búsqueda.

Puede suceder que en la opción VARYING se especifique un índice distinto de los que figuran en la definición de la tabla objeto de la búsqueda, por ejemplo, el índice de otra tabla.

En este caso la búsqueda en dicha tabla se realiza, como el apartado a), utilizando el primer índice que sigue a INDEXED BY.

EJEMPLO. Escribir declaraciones SEARCH con/sin opción VARYING, que permitan investigar en la tabla FAB–DIARIA, a fin de conocer, si a partir del elemento 125 de dicha tabla existe un campo AUTOS cuyo valor sea igual a cero.

```
010  01  TABLA-FABRICACION.
020      02  FAB-DIARIA  OCCURS 1080 INDEXED BY IND1, IND2, IND3.
030          03 AUTOS      PIC 9(3).
040          03 CAMIONES   PIC 99.
050          .....................................
060      SET IND1 TO 125
070      SEARCH FAB-DIARIA
090          AT END PERFORM FIN
100          WHEN AUTOS (IND1) = 0
110          ----------------------

130      SET IND3 TO 125
140      SEARCH  FAB-DIARIA   VARYING IND3
150          AT END PERFORM FIN
160          WHEN AUTOS (IND3) = 0
170          ----------------------
```

La declaración SEARCH de la línea 070 deberá utilizar IND1 de acuerdo con el apartado a), mientras que la SEARCH de la línea 140 deberá utilizar IND3 conforme con el apartado b).

3.4.- Búsqueda BINARIA (Con índices)

El método de búsqueda binaria empleando indexación es el más utilizado, ya que con este método la mayor parte de las inscripciones que hace el programador cuando emplea la subindicación para: inicializar los subíndices, irles incrementando, controlarles para que no superen el valor del número de elementos de la tabla, etc., no es necesario especificarlas dentro del programa, por ser acciones que realiza el Sistema de forma automática, es decir, sin que se le indique, y siempre que para efectuar el proceso de búsqueda binaria se emplee la declaración SEARCH ALL.

3.4.1.- Declaración SEARCH ALL

FUNCION: Permitir la búsqueda binaria en una tabla descrita con la opción INDEXED BY.

FORMATO:

SEARCH ALL identificador–1 [**AT END** declaración-imperativa–1]

WHEN condición { declaración–imperativa–2 / **NEXT SENTENCE** }

Esta declaración efectúa el proceso de búsqueda en una tabla, utilizando el mismo principio que se describió en el método de búsqueda binaria empleando subindicación, pero con la ventaja de que las direcciones de los elementos de la tabla a los que se ha de acceder hasta localizar al que contenga un valor determinado, las calcula el Sistema automáticamente, por lo que el programador no tiene necesidad de ocuparse de la validez de los valores dados a los nombres–de–índice, y ni siquiera tener que inicializar con un valor dicho nombre–de–índice con la declaración SET, como se hacía en la búsqueda secuencial con la declaración SEARCH simple.

Cuanto se especificó para Identificador–1 y AT END en la declaración SEARCH, es válido para SEARCH ALL . El resto de los componentes de esta declaración presenta las siguientes diferencias:

- Sólo puede especificarse una palabra WHEN con una condición, la cual únicamente puede ser del tipo de relación: EQUAL TO o compuesta enlazada por el operador lógico AND.
- Los elementos de la tabla deben estar ordenados en forma ascendente o descendente por el contenido del campo objeto de la búsqueda (opción ASCENDING o DESCENDING).

La siguiente declaración SEARCH ALL realiza la búsqueda, en forma binaria, del elemento de la TABLA2 que contiene al agente de ventas GOMEZ MANRIQUE JUAN.

```
002020     77   AGENTE          PIC X(35)
002030                         VALUE 'GOMEZ MANRIQUE JUAN                    '.
002040     01   TABLA2.
002050          02   ELEMEN     OCCURS 45
002055                          ASCENDING KEY NOMBRE     INDEXED BY ALFA.
002060               03   NOMBRE          PIC X(35).
002070               03   VENTAS          PIC 9(4).
.......  ..............................................................
004080*  BUSQUEDA EMPLEANDO SEARCH ALL DEL ELEMENTO DE LA TABLA2
004090*       QUE CONTIENE AL AGENTE  GOMEZ MANRIQUE JUAN.
004100   BUSQUEDA.
004110                    SEARCH ALL ELEMEN  AT END GO TO NOEXIS
004120                         WHEN NOMBRE (ALFA) = AGENTE
004130                              MOVE NOMBRE (ALFA) TO NOMBRIM
004140                              MOVE VENTAS (ALFA) TO VENTIM
004150                              WRITE IMPRESO FROM LINDETAL AFTER PAG-NUEVA
004160                              GO TO  FIN.
004170   NOEXIS.
004175                    DISPLAY  AGENTE  'NO EXISTE EN LA TABLA'.
```

El agente de ventas es localizado cuando se cumple la condición: NOMBRE (ALFA) = AGENTE, en cuyo momento se trasferirá el nombre del agente (línea 004130) y la venta efec - tuada (línea 004140) a los campos del registro LINDETAL, y de ahí al área de salida IMPRESO para imprimirla al comienzo de una página nueva. Caso de que no existiese en la tabla un elemento que contuviera el nombre del citado agente, se cumpliría la condición AT END, por lo que pasaría a ejecutar la declaración GO TO NOEXIS.

Las declaraciones contenidas en el procedimiento NOEXIS harían imprimir el mensaje: GOMEZ MANRIQUE JUAN NO EXISTE EN LA TABLA, y finalizarían la ejecución.

Ver ejercicio número 15 de la pág. 284, en el que se emplean las declaraciones SEARCH y SEARCH ALL para realizar la búsqueda en una tabla de dos niveles.

4.- EJERCICIOS

1.- La declaración de la línea 270 es incorrecta. Explicar el motivo.

```
130   77 ALFA    PIC 9(5)   VALUE ZEROS.
140   01 REGISTRO.
150      02 ANUAL    OCCURS 4 TIMES.
160         03 MENSUAL  OCCURS 12 TIMES.
170            04 DIARIA  OCCURS 30 TIMES.
180               05 MODELO    PIC X(6).
190               05 NUMERO    PIC X(4).
200               05 CANTIDAD PIC 9(3).
. . . . . .
270   ADD CANTIDAD (3, 6)  TO  ALFA
```

2.- A la vista de la siguiente descripción del REGISTRO–VENTA:

```
010   01 REGISTRO-VENTA.
020      02  A   OCCURS  5  TIMES.
030         03   B     PIC  9(3).
040         03   C     PIC  X(6).
050         03   D     OCCURS  4  TIMES.
060            05   E      PIC  9(4).
070            05   F      OCCURS  7  TIMES.
080               06   G      PIC  XX.
090               06   H      PIC  9(3).
```

a) Representar gráficamente como estarían dispuestos los campos.

b) ¿Qué número de veces estan contenidos los campos E y H en REGISTRO–VENTA?

c) ¿Cuál de las siguientes inscripciones sería la correcta para referirse al elemento E que ocupa el tercer lugar dentro del campo D y éste a su vez el cuarto lugar dentro del campo A?

1.- E (1, 2)
2.- E (4, 3)
3.- E (3, 4)
4.- Ninguna de las soluciones es correcta.

d) Se quiere imprimir el contenido del campo G que ocupa el lugar 125 dentro del REGISTRO–VENTA ¿qué declaración la llevaría a cabo? Encontrar un algoritmo con el que calcular los tres subíndices que son necesarios para poder referirse al citado campo.

e) ¿Qué número de orden ocupa en la tabla el elemento H (4, 2, 3) ? Encontrar la fórmula a aplicar para calcular dicho número.

3.- TABLA–PRODUCCION

	FACTORIA (1)																FACTORIA (2)			FACTORIA (6)					
	IDENTIFI–CACION		PRODUCCION														IDENTIFI–CACION			PRODUCCION					
				ANUAL			ANUAL			ANUAL			ANUAL			ANUAL				NUAL		ANUAL			
	NOMBRE–FAC	DIRECCION	MODELO	CAMION	TRACTOR	TURISMO	CAMION	TRACTOR	TURISMO	CAMION	TRACTOR	TURISMO	CAMION	TRACTOR	TURISMO	CAMION	TRACTOR	TURISMO	NOMBRE–FAC	DIRECCION		TRACTOR	TURISMO	CAMION	TRACTOR	TURISMO
PIC	X(6)	X(12)	X(7)	99	99	999	99	99	999	99	99	999	99	99	999	99	99	999	X(6)	X(12)	X	99	999	99	99	999

Teniendo en cuenta que FACTORIA se repite seis veces, hacer la descripción de TABLA–PRODUCCION.

4.– ¿Es equivalente la descripción de las líneas 002030 a 002070 que se hizo en el programa DESGLOSE (pág. 249) a las que figuran a continuación ?

```
002030   01  BILLETES.
002035       02  FILLER      PIC 9(4)    VALUE 5000.
002036       02  FILLER      PIC 9(4)    VALUE 1000.
002037       02  FILLER      PIC 9(4)    VALUE 0500.
002038       02  FILLER      PIC 9(4)    VALUE 0100.
002039       02  FILLER      PIC 9(4)    VALUE 0050.
002040       02  FILLER      PIC 9(4)    VALUE 0025.
002045       02  FILLER      PIC 9(4)    VALUE 0005.
002050       02  FILLER      PIC 9(4)    VALUE 0001.
002060   01  BILLET  REDEFINES  BILLETES.
002070       02  BILLETE  OCCURS 8  PIC 9(4).
```

5.– En el programa DESGLOSE se empleó la declaración: MOVE ZEROS TO CONTADORES para llenar simultáneamente con ceros la tabla denominada CONTADORES, sin embargo, ahora también se quiere llenar dicha tabla con ceros, pero transfiriéndolos a cada elemento en forma sucesiva. Codificar las declaraciones que ejecuten dicho trabajo.

6.– Si en el programa DESGLOSE, las tablas denominadas CONTADORES y BILLET se hubieran descrito de la forma siguiente:

```
002010   01  CONTADORES.
002020       02  CONTADOR  OCCURS 8 TIMES  PIC 9(6)  INDEXED BY ALFA.
.....  .....  .....  .....  .....  .....  .....  .....  .....  .....
002060   01  BILLET  REDEFINES  BILLETES.
002070       02  BILLETE  OCCURS 8 TIMES  PIC 9(4)  INDEXED BY BETA.
```

¿Sería necesario modificar alguna declaración en la PROCEDURE DIVISION para llevar a cabo el trabajo de desglose de moneda?

7.– Aprovechando que han sido indexadas las tablas: CONTADORES y BILLET, según se ha descrito en la pregunta anterior, codificar la PROCEDURE DIVISION de forma que para el manejo de dichas tablas se emplee el método de Indexación.

8.– Si en el programa BUSECUEN (pág. 256) se suprime la descripción del campo ALFA (línea 001170) y se describe REGISTRO–AUTOS de la forma siguiente:

```
.....  .....  .....  .....  .....  .....  .....  .....  .....  .....
001120   01  REGISTRO–AUTOS.
001130       02  NOMBRE  PIC A(6).
001140       02  MODELO  PIC X(8).
001150       02  DIARIA  PIC 999 OCCURS 1080 INDEXED BY ALFA.
```

¿Qué declaraciones de la PROCEDURE DIVISION sería necesario modificar, como consecuencia de las variaciones introducidas, para obtener los mismos resultados?

9.– En algunas declaraciones SET de la descripción siguiente se han cometido errores. Explicar el motivo.

```
010    01 TABLA-INTERESES.
020       02 INTERES OCCURS 10 PIC 99V99 INDEXED BY ALFA.
030       .....
040    WORKING-STORAGE SECTION.
050       77 BETA USAGE INDEX.
060       77 GAMMA PIC 99 VALUE 12.
070    PROCEDURE DIVISION.
080       .....
090       SET GAMMA TO BETA
100       SET BETA TO 1
110       SET ALFA TO ZERO
120
```

10.– Hacer la codificación de la tabla cuya disposición gráfica y contenido de los elementos que la componen es:

	TABLA–PRODUCCION																					
	MENSUAL (1)													MENSUAL (2)				MENSUAL (12)				
		DIARIA 1		DIARIA 2		DIARIA 3		DIARIA 4		DIAR 5		DIARIA 20			DIARIA 1			DIARIA 19		DIARIA 20		
	MES	DIA	CANTI	DIA	CANTI	DIA	CANTI	DIA	CANTI	DIA	CANTI	DIA	CANTI	MES	DIA	CANTI	CANTI	DIA	CANTI	DIA	CANTI	
CONTENIDO	01	28	450	05	375	25	312	07	257	21	98	16	055	02	23	520	089	23	080	26	075	

teniendo en cuenta que los elementos de dicha tabla se disponen de la forma siguiente:

a) Ordenados en forma ascendente por el número del mes. Según la figura:01 para Enero, 02 para Febrero, etc, y así sucesivamente para todos los meses del año de producción.

b) Los veinte campos que siguen al número del mes se subdividen en otros dos, siendo el contenido de uno de ellos el número de automóviles fabricados durante cada uno de los veinte días del mes, estando ordenados según su valor, en forma descendente. El otro campo contiene el número del día en que se llevó a cabo dicha producción.

c) El campo CANTI va acompañado de PICTURE 999.

11.– En las descripciones siguientes:

```
130    01 TABLA-VENTAS.
140       02 VENTAS OCCURS 20 ASCENDING CANTI INDEXED BY ALFA.
150          03 CODIGO PIC 9(5).
160          03 CANTI PIC 9(4).
170       .....
180    MOVER. SET ALFA TO 1.
190    BUSCAR. SEARCH ALL VENTAS AT END GO TO FIN
200                    WHEN CANTI (ALFA) = 125
210                    DISPLAY CODIGO (ALFA) GO TO FIN.
220       .....
260    FIN. .....
```

sobra la línea 180, ya que no es necesario antes de comenzar la búsqueda en la tabla VENTAS indicar al compilador cuál es el número del elemento por el que ha de comenzar dicha búsqueda:

CIERTO FALSO

12.– En la fase de compilación de las siguientes inscripciones, el compilador ha detectado la existencia de un error cometido por el programador en la descripción de la tabla, por el cual no puede llevar a cabo la ejecución de la declaración SEARCH ALL. Explicar en qué consiste dicho error.

```
0100
0200    01 TABLA.
0300       02 VENTAS OCCURS 10 INDEXED BY ALFA.
0400          03 CODIGO PIC 9(5).
0500          03 CANTI  PIC 9(6).
0600    ..........................
0700    BUSCAR. SEARCH ALL VENTAS
0800                   WHEN CODIGO (ALFA) = 01250
0900                   DISPLAY VENTAS (ALFA).
1000    FIN. ..........................
```

13.– Comparando las descripciones que figuran a continuación, con las hechas en el programa ORBUSBIN (pág. 268 y 269) el programador ha cometido dos errores. ¿Cuál es el motivo?

```
1200    ..........................
1300    01 TABLA2.
1400       02 ELEMEN OCCURS 45 ASCENDING KEY IS NOMBRE INDEXED ALFA.
1500          03 NOMBRE PIC X(35).
1600          03 VENTAS PIC 9(4).
1700    ..........................
1800    BUSCAR. SEARCH ALL ELEMEN
1900                   WHEN VENTAS (ALFA) > 1250
2000                   MOVE VENTAS (ALFA) TO VENTIM.
```

14.– Teniendo en cuenta cuanto se especifica en las líneas 002020 a la 002070 del ejemplo de la pág. 276 ¿Cuál de las dos descripciones siguientes es otra solución del citado ejemplo? Indicar cuál es el motivo por el que una de dichas descripciones no es una solución.

a)

```
004099          SET ALFA TO 1.
004100    BUSQUEDA.
004110          SEARCH ELEMEN AT END GO TO NOEXIS
004120                 WHEN NOMBRE (ALFA) = AGENTE
004130                 DISPLAY ALFA 'ES LA POSICION QUE OCUPA' AGENTE
......                 ..........................
004180    NOEXIS.
```

b)

```
004099          SET ALFA TO 1.
004100    BUSQUEDA.
004110          SEARCH ELEMEN AT END GO TO NOEXIS
004120                 WHEN NOMBRE (ALFA) = AGENTE
004125                 SET ELEMEN-NUM TO ALFA
004130                 DISPLAY ELEMEN-NUM 'ES LA POSICION QUE OCUPA'
004135                                    AGENTE
......                 ..........................
004180    NOEXIS.
```

15.– La Sección de Estadística de RICOSA dispone de una cinta magnética, grabada con la estadística de las ventas mensuales de automóviles y camiones habidas durante el año 1981 en cada una de las 50 provincias.

Los datos están dispuestos formando la siguiente tabla:

TABLA VENTAS

nombre provincia	enero autos	enero cam.	febrero autos	febrero cam.			diciembre autos	diciembre cam.			
ALAVA	2315	178	4360	210	3061	102	2808	125	3735	356	ALBACETE	0406	036	19

Es decir, ordenados alfabéticamente por el nombre de las provincias, y dentro de cada provincia las ventas estan ordenadas por la secuencia de los meses: Enero, Febrero, etc.

Teniendo en cuenta que la descripción de la tabla sería:

```
010  01  TABLA-VENTAS.
020      02  PROVINCIA    OCCURS  50
030                        ASCENDING KEY  NOMBREPROV
040                        INDEXED BY  INDICE1.
050          03  NOMBREPROV      PIC X(16).
060          03  VENTASM OCCURS  12
070                        INDEXED BY  INDICE2.
080              04  AUTOS       PIC 9(4).
090              04  CAMIONES PIC 9(3).
```

Se quiere conocer si las ventas en la provincia de VALLADOLID en uno cualquiera de los doce meses reunen las siguientes condiciones : El número de autos vendidos superior a 1.500, y el número de camiones superior a 200.

Utilizar la declaración SEARCH ALL para localizar la provincia, y la declaración SEARCH para investigar en VENTASM si existe un elemento que reune las condiciones citadas.

16.– El procedimiento seguido por el programador en la ordenación alfabética de TABLA2, del programa BUSBINAR (pág. 263, líneas 003160 a 003250), descrito en la pág. 264 tiene la ventaja de su simplicidad. Sin embargo, no es el metodo optimo, ya que muchas de las comparaciones que se llevan a cabo entre los elementos de dicha tabla son innecesarias, pues se realizan 44 (nº de elementos–1) x 44 (recorridos) = 1936 comparaciones, cuando habrían sido, como máximo, suficientes con: 44 + 43 + 42 + ...+ 3+2+1 = 990. Este procedimiento tampoco contempla la posibilidad de que, casualmente, en alguno de los 44 recorridos puedan encontrarse ordenados los elementos de la citada tabla, ya que en este caso no sería necesario continuar la ordenación.

Por tanto, se quiere codificar una solución que optimice la ordenación de TABLA2, evitando llevar a cabo comparaciones innecesarias.

17.– Codificar la solución dada al ejercicio anterior de forma que se pueda incluir dentro del programa ORBUSBIN (pág. 268 y 269), empleando para el manejo de las tablas el metodo de indexación.

CAPITULO VII

TECNICAS DE PROGRAMACION DE DISPOSITIVOS DE ACCESO DIRECTO

1.- INTRODUCCION

En los programas ejemplo que hasta ahora se han expuesto, la lectura y/o escritura de archivos se ha llevado a cabo por medio de lectoras de fichas, unidades de cinta magnética, impresoras, etc. Estos dispositivos resultan útiles para efectuar determinados tipos de trabajos, en los que no se requiere grandes volúmenes de entrada de datos o escritura de resultados, o si intervienen grandes volúmenes, que la forma de procesar los archivos sea secuencialmente.

Existen otros tipos de trabajos en los que se requiere una respuesta inmediata del ordenador a la solicitud de una información, o procesar la información a medida que se va produciendo a fin de ir actualizando los archivos sin tener que esperar a que exista un determinado número de transaciones o modificaciones para llevar a cabo dicho proceso; es el caso, por ejemplo, de aplicaciones bancarias, reserva de billetes en líneas aéreas, control de existencias en almacenes, etc., en las que se necesita una respuesta inmediata a la solicitud de una información como: ¿Cuál es el saldo de una cuenta corriente? ¿Hay plazas para un determinado vuelo? ¿Hay existencias de una determinada pieza? etc., y en caso de respuesta afirmativa, habrá que actualizar inmediatamente el archivo con las variaciones que se produzcan para poder contestar a preguntas posteriores. En estos tipos de trabajo puede ser incluso necesario la ampliación del archivo sin tener que copiar de nuevo los registros que estaban grabados como sucedía con los archivos en cinta que se han empleado.

Por tanto, estos trabajos difícilmente podrían llevarse a cabo con los dispositivos hasta ahora empleados, bien porque las respuestas normalmente no estarían servidas en el momento solicitado, bien porque el número de consultas a realizar sería ínfimo al tener que efectuar la búsqueda de la información en forma secuencial hasta lograr localizar la información solicitada, o bien porque requerirían una reorganización total del archivo aunque sólo fuera para incluir un nuevo registro.

Por otra parte, es cada día mayor el número de aplicaciones que se mecanizan para las que se requiere reunan además de las citadas características, la posibilidad de almacenar mayor cantidad de información dentro de cada soporte. Estos problemas van a encontrar solución con la utilización de nuevos dispositivos que se describirán a lo largo de este capítulo.

2.- DISPOSITIVOS DE ACCESO DIRECTO

La aparición y posterior perfeccionamiento de los soportes magnéticos, discos, tambores, etc., así como de los dispositivos encargados de su grabación y lectura permiten poder almacenar en un espacio reducido gran cantidad de información, a la vez que sirven de medio para su rápida localización sea cual fuera su situación dentro del soporte magnético, es decir, con una pequeña exploración del archivo se localiza la información contenida en cualquier registro sin tener que recorrer secuencialmente todo el archivo. A este tipo de localización se denomina Acceso Directo.

A las máquinas que se emplean para la grabación y posterior localización de la información dentro del soporte, se denominan Dispositivos de Acceso Directo.

Actualmente, las unidades de disco son los dispositivos de acceso directo más empleados por razón de su economía, capacidad de almacenamiento, rapidez en la grabación y en la recuperación de la información, etc.

3.- METODOS DE ORGANIZACION

Para hacer posible el acceso directo es preciso crear el archivo en una forma lógica, siguiendo alguno de los métodos de organización de archivos comúnmente empleados por la mayor parte de los diversos tipos de ordenadores.

Dichos métodos son los siguientes:

- **ORGANIZACION SECUENCIAL**
- **ORGANIZACION SECUENCIAL INDEXADA**
- **ORGANIZACION DIRECTA**

Cada uno de estos métodos tiene sus ventajas e inconvenientes, de los que se hará una breve enumeración a lo largo de este Capítulo, a fin de que el analista o el programador sea capaz de elegir aquél que haga más eficaz la ejecución del trabajo.

4.- MODOS DE ACCESO

Los dispositivos de almacenamiento en discos magnéticos permiten, tanto en la creación como en los procesos posteriores con el archivo, el empleo de los dos modos de acceso: Secuencial y Directo (ver pág. 89). Sin embargo, dicho empleo está sujeto a determinadas limitaciones según el método de organización utilizado (ver cuadro resumen de modos de acceso).

Si la creación del archivo se va a realizar con organización secuencial o secuencial indexada debe especificarse en el programa: ACCESS IS SEQUENTIAL, con lo cual los registros se

grabarán en secuencia física, es decir, a partir de la primera pista del almacenamiento asignado para el archivo se irán grabando los registros y, no comenzará la grabación en una nueva pista sin antes estar completa la pista anterior.

Si la creación del archivo se realiza con organización directa se debe especificar en el programa: ACCESS IS RANDOM. En este caso, cada registro se irá grabando en el almacenamiento en forma aleatoria en aquella pista cuyo número se obtiene por la aplicación de un algoritmo a la clave del registro. Por tanto, puede suceder, por ejemplo, que el primer registro a grabar, lo haga en la pista número siete, el segundo registro en la pista número tres, etc. El especificar este tipo de acceso no excluye la posibilidad de que el algoritmo a aplicar sitúe los registros en el almacenamiento en secuencia física como en la organización secuencial.

Una vez creado el archivo, habrá que realizar con él procesos posteriores para los que requerirá la obtención o recuperación de registros desde el almacenamiento e introducirlos en la memoria del ordenador.

Según el tipo de organización con el cual fue creado el archivo, así será el modo de acceso a emplear:

- A un archivo creado con organización secuencial sólo puede accederse secuencialmente. Este acceso permitirá la obtención de los registros en la secuencia física en que se encuentran grabados en el almacenamiento.

- A un archivo creado con organización secuencial indexada o directa se puede elegir entre los dos modos de acceso para la recuperación de los registros:

  - Especificando ACCESS IS SEQUENTIAL los registros se recuperarán del almacenamiento en la secuencia física en la que se hallen grabados, es decir, no se comenzará la recuperación de registros de una pista sin antes haber recuperado completamente los de la pista anterior.

  - Especificando ACCESS IS RANDOM los registros se recuperan sin sujetarse a secuencia alguna, es decir, en el orden en que estime más conveniente el programador.

CUADRO RESUMEN DE MODOS DE ACCESO

ORGANIZACION	CREACION	RECUPERACION
SECUENCIAL	SEQUENTIAL	SEQUENTIAL
SECUENCIAL INDEXADA	SEQUENTIAL	SEQUENTIAL o RANDOM
DIRECTA	RANDOM	SEQUENTIAL o RANDOM

5.- ORGANIZACION SECUENCIAL

En un archivo que se ha de organizar secuencialmente los registros se situan dentro del discpack (paquete de discos) en posiciones físicas sucesivas, es decir, unos detrás de otros, llenando pista tras pista y cilindro tras cilindro hasta completar su creación.

La organización secuencial de los registros no lleva implícito que éstos se graben ordenados en forma ascendente o descendente conforme al contenido de un determinado campo llamado Clave; no obstante, es conveniente dicha ordenación para poder llevar a cabo, de forma eficaz, la realización de posteriores trabajos.

5.1.- CREACION DEL ARCHIVO

El programa siguiente lleva a cabo la creación de un archivo con organización secuencial a partir del archivo en fichas utilizado en el programa NOMINA.

```
001010  IDENTIFICATION DIVISION.
001020  PROGRAM-ID.  CREASECU.
001030* PROGRAMA PARA CREAR EN UN DISCO UN ARCHIVO SECUENCIAL.
001040  ENVIRONMENT DIVISION.
001050    INPUT-OUTPUT SECTION.
001060      FILE-CONTROL.
001070        SELECT ARCHIVO-OPERARIOS  ASSIGN SYS004-UR-2540R-S.
001080        SELECT ARCHIDISC          ASSIGN SYS005-DA-3340-S
001085                                  ACCESS SEQUENTIAL.
001090  DATA DIVISION.
001100    FILE SECTION.
001110      FD  ARCHIVO-OPERARIOS     LABEL RECORD OMITTED.
001120          01 FICHAOPE              PIC X(80).
001130      FD  ARCHIDISC               LABEL RECORD STANDARD.
001140          01 REGISDISC             PIC X(80).
001145    WORKING-STORAGE SECTION.
001150      01  HAY-REGISTROS    PIC XX     VALUE 'SI'.
001155  PROCEDURE DIVISION.
001156    ORDEN-DE-PROCESO.
001160        OPEN  INPUT  ARCHIVO-OPERARIOS
001162              OUTPUT ARCHIDISC.
001165        READ  ARCHIVO-OPERARIOS
001167              AT END  MOVE  'NO'  TO  HAY-REGISTROS.
001168        PERFORM  GRABAR-LEER  UNTIL HAY-REGISTROS = 'NO'
001170        CLOSE  ARCHIDISC
001172               ARCHIVO-OPERARIOS
001173        STOP RUN.
001174    GRABAR-LEER.
001180        WRITE  REGISDISC
001182               INVALID KEY  DISPLAY  'NO CABEN TODOS LOS REGISTROS EN
001184-                                    'EL ARCHIVO'
001190                                    STOP RUN.
001200        READ  ARCHIVO-OPERARIOS
001210              AT END  MOVE  'NO'  TO  HAY-REGISTROS.
```

A la vista de la codificación del programa este método de organización presenta muy leves diferencias con respecto a la codificación necesaria para crear un archivo en cinta, ya que practicamente se emplean las mismas cláusulas. Dichas diferencias son las siguientes:

Línea 001080, se debe especificar –DA– por ser de acceso directo el dispositivo empleado en la grabación; –3340– es el número de modelo de dicho dispositivo.

Línea 001130, el empleo de dispositivos de acceso directo obliga a crear los archivos con etiquetas, bien mediante la cláusula LABEL RECORD STANDARD, o bien si el programador crea una etiqueta propia, la cláusula LABEL RECORD nombre-de-la-etiqueta.

En la línea 001180 se ha utilizado el Formato 3 de la declaración WRITE. En este caso la función de la INVALID KEY es proporcionar el medio de conocer si todos los registros del ARCHIVO–OPERARIOS han podido ser grabados dentro del número de cilindros que el programador ha reservado en el discpack (21). Es decir, especificando la opción INVALID KEY en la declaración WRITE, obliga al Sistema a que antes de escribir cada registro, compruebe si puede grabarlo, caso afirmativo, lo graba, y en caso contrario, imprime el mensaje 'NO CABEN ....' y finaliza la ejecución del programa, en cuyo caso habría que iniciar de nuevo el trabajo reservando mayor número de cilindros.

(21) Hay que tener en cuenta que dentro del discpack pueden grabarse uno o varios archivos, por lo que el programador por medio de fichas de control debe indicar al Sistema cuántos cilindros va a ocupar el archivo que ha de crear, así como el número del cilindro en que debe comenzar dicha grabación. Por tanto, si por error asigna menor número de cilindros de los necesarios y no caben todos los registros,daría lugar a que el trabajo terminara anormalmente.

5.2.- PROCESO DEL ARCHIVO

Una vez creado el archivo, queda en disposición de poder ser procesado. El tipo de proceso del archivo debe ser secuencial, ya que por estar organizado secuencialmente, sólo se permite el acceso secuencial para la lectura o recuperación de los registros.

El siguiente ejemplo muestra una aplicación en la que resulta eficaz utilizar un archivo creado con organización secuencial, ya que van a ser procesados todos los registros del archivo en el orden que fueron grabados.

EJEMPLO.

Hacer las modificaciones necesarias en el programa NOMINA para llegar a la confección de la nómina, utilizando como archivo de entrada el archivo en disco creado en el ejemplo anterior (programa CREASECU).

```
Página  Linea  COL A  B
1  3  4  6  7  8  12  15  20  25  30  35  40  45  50  55  60  65  70
001200    SELECT ARCHIVO-OPERARIOS ASSIGN SYS007-DA-3340-S.
002070    LABEL RECORD IS STANDARD
```

Por tanto, a excepción de dos inscripciones que se han modificado, el resto de las inscripciones del programa NOMINA son válidas, ya que, aunque durante la creación del archivo se le haya denominado: ARCHIDISC, y al registro: REGISDIC, no es obligatorio que en la codificación de un programa que realice otro trabajo, se describan éstos con los mismos nombres que cuando fueron creados.

La organización secuencial de un archivo grabado con dispositivos de acceso directo presenta los mismos inconvenientes que un archivo grabado con una unidad de cinta magnética, es decir:

- Tener que crear de nuevo el archivo si se desea actualizar, incorporar, etc., algún registro al archivo.
- No poder acceder directamente a cualquier registro del archivo sin antes pasar por los registros que le precedan (21).
- Para ser rentable el trabajo a realizar, se necesita una previa clasificación de todas las transacciones que se han de realizar con los registros del archivo en el orden en que éstos están grabados, ya que si se procesan en cualquier orden daría lugar a un consumo excesivo de tiempo del ordenador.

---

(21) En algunos Sistemas, la organización secuencial permite no tener que crear de nuevo el archivo cuando: se quiera incorporar un nuevo registro y siempre que se haga a continuación de los ya grabados, o se quiera actualizar un registro, en cuyo caso se lee el registro, se actualiza y posteriormente se vuelve a grabar en el mismo lugar de donde se extrajo.

6.- ORGANIZACION SECUENCIAL INDEXADA

Este método de organización presenta diferencias con respecto al anterior método, ya que los registros se sitúan dentro del archivo en orden secuencial ascendente por el contenido de un campo del registro denominado clave, por ejemplo: el número de una cuenta corriente, el número del documento nacional de identidad, etc., y como diferencia esencial, aquella que hace que simultáneamente a la grabación de los registros, el Sistema cree los Indices (registros cuyo contenido es la clave del último registro grabado en cada pista o en cada cilindro, según que el índice sea de pistas o de cilindros, junto con la dirección física donde se graba) a fin de que en posteriores procesos del archivo pueda ser rápidamente localizado cualquier registro. Por tanto, si el Sistema lleva el control de la creación de índices a la vez que carga los registros dentro del archivo, no es necesario que el programador conozca con detalle cómo lo lleva a cabo, limitándose a realizar la codificación del programa que cree el archivo, así como indicar al Sistema el número de cilindros del discpack que ha reservado para dicha creación. No obstante, es conveniente tener conocimientos acerca de los conceptos siguientes:

- La posición de los índices y registros dentro de los cilindros que el programador ha reservado en el discpack, depende del Sistema de ordenador empleado. Un reparto normal en cada cilindro de los que componen el archivo, se muestra en la figura 23 en la que se puede distinguir tres partes:

  . Area de índices
  . Area de datos y
  . Area de overflow (excedentes)

- En la primera pista de cada cilindro el Sistema crea un índice de pista, constituido por registros, cuyo contenido es el número de pista y la clave del registro que ocupa el último lugar en esa pista de datos. Por tanto, por cada pista de datos se graba uno de dichos registros.

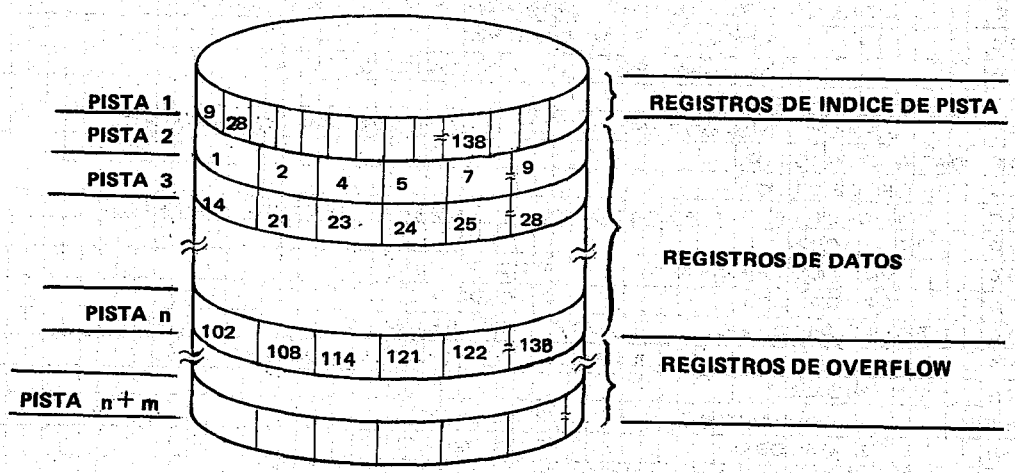


Figura 23

- En las siguientes pistas del cilindro y en número determinado que especifica el programador al codificar el programa, se sitúan los registros de datos constituidos por la clave y los datos en orden ascendente de clave. El conjunto de estas pistas recibe el nombre de área de datos.

- Las restantes pistas de cada cilindro se reservan para poder, una vez creado el archivo, añadir nuevos registros en un proceso posterior. Estas pistas reciben el nombre de excedentarias o área de OVERFLOW. Opcionalmente, el programador puede incrementar este área, con un área adicional independiente; en este caso dicha área servirá para añadir nuevos registros al archivo cuando el área de OVERFLOW de uno cualquiera de los cilindros esté llena; se ubica normalmente en los cilindros que siguen a los asignados al archivo.

- El Sistema también crea un índice de cilindros cuyo contenido son pequeños registros conteniendo cada uno la clave del registro de clave más alta que haya grabado en cada cilindro de los que componen el archivo. El programador únicamente debe preocuparse de indicar mediante una ficha de control, el número del cilindro en que el Sistema ha de crear el Indice de cilindros.

6.1.- CREACION DEL ARCHIVO

El siguiente ejemplo crea un archivo en disco con organización secuencial indexada utilizando como archivo de entrada el archivo en fichas empleado en la confección de la Nómina.

```
001010   IDENTIFICATION DIVISION.
001020    PROGRAM-ID. CREAINDX.
001030    REMARKS. CREACION DE UN ARCHIVO CON ORGANIZACION SECUENCIAL
001040                        INDEXADA A PARTIR DE OTRO ARCHIVO EN FICHAS.
001050   ENVIRONMENT DIVISION.
001060    INPUT-OUTPUT SECTION.
001070     FILE-CONTROL.
001080         SELECT ARCHIFICH  ASSIGN SYS005-UR-2540R-S.
001090         SELECT ARCHIDISC  ASSIGN SYS006-DA-3340-I
001100                        ACCESS SEQUENTIAL
001110                        RECORD KEY IS CODIDISCO.
001120     I-O-CONTROL.
001130         APPLY CYL-OVERFLOW OF 4 TRACKS ON ARCHIDISC.
001140   DATA DIVISION.
001150    FILE SECTION.
001160     FD ARCHIFICH RECORDING F  LABEL RECORD OMITTED.
001170     01 FICHA.
001180         02 CODIFICHA          PIC X(5).
001190         02 DATOSF             PIC X(75).
001200     FD ARCHIDISC      BLOCK  10 RECORDS
001210                       LABEL  RECORD  STANDARD.
001220     01 DISCO.
001230         02 CODIDISCO          PIC X(5).
001240         02 DATOSD             PIC X(115).
002010    WORKING-STORAGE SECTION.
002020     77 COMPARADOR             PIC X(5).
```

```
002030  PROCEDURE DIVISION.
002040    ABRIR.  OPEN INPUT ARCHIFICH   OUTPUT ARCHIDISC.
002050            DISPLAY 'RELACION DE REGISTROS A ANALIZAR POR POSIBLE ERR
002060-                   'OR'.
002070    LEER.   READ ARCHIFICH  AT END  GO TO FIN.
002080            WRITE DISCO FROM FICHA  INVALID KEY GO TO RUTINA-ERROR.
002090            MOVE CODIFICHA TO COMPARADOR
002100            GO TO LEER.
002110    RUTINA-ERROR.
002120            IF COMPARADOR = CODIDISCO
002130                 DISPLAY  FICHA  '  =  CODIGO DUPLICADO'
002140                 ELSE DISPLAY FICHA '  =  CODIGO NO ESTA EN SECUENCIA'.
002150            GO TO LEER.
002160    FIN.    CLOSE ARCHIFICH  ARCHIDISC
002170            STOP RUN.
```

Las declaraciones que llevarán a cabo la creación del archivo ARCHIDISC son en esencia las siguientes: abrir el archivo de entrada y de salida (línea 002040), leer ficha a ficha, es decir,los registros que constituyen el archivo de entrada, comprobar, previa a la grabación de cada registro, la validez de su clave o código (línea 002080); en caso de clave válida se graba y en caso contrario no se graba y se investiga el motivo de su invalidez (línea 002110 a 002150).

En la línea 001090, dentro de la cláusula ASSIGN, se indica al ordenador mediante la letra –I– que el método de organización que ha de seguir para la creación del archivo es el Indexado.

En otros Sistemas se emplea la cláusula: ORGANIZATION IS INDEXED.

En la línea 001110 se incluye la cláusula RECORD KEY cuya función es especificar al Sistema el nombre del campo de identificación del registro, es decir, aquel que va a contener la clave de los registros del archivo que se va a crear.

El formato de esta cláusula es:

RECORD KEY IS nombre–de–datos.

En el ejemplo, nombre–de–datos es CODIDISCO, el cual ocupa la primera posición dentro del registro (línea 001230) aunque ello no es preceptivo, pudiendo ocupar cualquier posición. Su longitud no debe exceder de 255 caracteres.

Con la cláusula RECORD KEY, el Sistema crea un campo interno donde irá introduciendo las claves de los registros, para antes de grabarlos efectuar una previa comprobación de que existe secuencia correcta en dichas claves.

En la línea 001130 se especifica la cláusula APPLY cuya función es indicar al Sistema que reserve en cada cilindro, de los que van a constituir el archivo, un número determinado de pistas para los registros que se han de añadir una vez creado.

Su formato es:

APPLY CYL-OVERFLOW OF entero TRACKS ON nombre-de-archivo -1

[nombre-de-archivo-2] . . .

El valor máximo de entero depende del tipo de dispositivo que se emplee, por ser diferente en cada dispositivo el número de pistas que componen cada cilindro.

El valor mínimo de entero es cero, con lo cual no se reserva ninguna pista para posteriores adiciones de registros.

En algún tipo de ordenador si se omite esta cláusula, el Sistema reserva un 20 por ciento de las pistas de cada cilindro para posteriores adiciones.

En el ejemplo, con la especificación de la cláusula APPLY, se indica al Sistema que reserve en cada cilindro de los que van a componer el archivo, cuatro pistas para añadir nuevos registros en posteriores procesos.

El número de pistas que el programador reserva para incluir nuevos registros depende del número de adiciones que prevea; así, si prevee que va a ser un número reducido, con reservar una pista en cada cilindro, sería suficiente.

En la línea 002080 se describe la declaración WRITE con la opción INVALID KEY para verificar la validez de las claves, ya que en el proceso de creación de un archivo indexado, una clave se considera válida cuando no está fuera de secuencia, ni duplicada.

Es decir, en un archivo que se va a crear:

- Los registros deben grabarse en secuencia ascendente de clave.

- No pueden grabarse registros que contengan la misma clave.

Ello lleva consigo que antes de ejecutarse el programa, deben ordenarse las fichas del archivo ARCHIFICH en secuencia ascendente de clave y eliminar aquéllas que estén repetidas.

El Sistema lleva a cabo la grabación de un nuevo registro de la forma siguiente:

Mueve a CODIDISCO el valor de la clave del registro que ha de grabar, así como a DATOSD los datos restantes del registro utilizando la opción FROM de la declaración WRITE.

Compara el contenido de CODIDISCO con el contenido del campo interno en el que conserva guardada la clave del registro grabado anteriormente; caso de que exista secuencia ascendente graba dicho registro, y en caso contrario la opción INVALID (clave inválida) impedirá dicha grabación, y la ejecución del programa bifuracará hacia el procedimiento especificado por el programador a continuación de INVALID, es decir, a RUTINA–ERROR (líneas 002120 a 002150).

En este procedimiento el programador ha descrito una forma de conocer el motivo por el que no se ha grabado un registro en el archivo: bien por haberse detectado un registro con clave duplicada (línea 002130) o bien por estar la clave fuera de secuencia (línea 002140). Efectivamente, cada vez que se graba un registro el programador (línea 002090) manda guardar la clave en el campo COMPARADOR (el cual nada tiene que ver con el que crea internamente el Sistema) y regresa a leer un nuevo registro (línea 002100). Cuando las claves sean inválidas, si al comparar el valor que contiene el campo COMPARADOR con la clave del registro que quería grabar (línea 002120), de dicha comparación resulta una igualdad, quiere decir que existen dos registros con clave duplicada o, en caso contrario, que la clave está fuera de secuencia.

Si al final de la ejecución del programa existiera una serie de registros que no se han grabado por ser sus claves inválidas , el programador debe investigar la causa de dicha invalidez, ya que podía suceder que fueran registros que había que grabar en el archivo y no se ha llevado a cabo por un simple error humano, por ejemplo, si en la transcripción de su clave hubiera resultado una duplicidad o bien por una mala ordenación de las fichas alguna estuviera fuera de secuencia; en cualquiera de los dos casos se corrige el error y se vuelve a ejecutar el programa; repitiendose esta operación tantas veces como sean necesarias hasta que no exista error alguno.

El programador podía haber dejado de especificar DISPLAY a continuación de INVALID, así como las descripciones de la línea 002110 a la 002150 y haber especificado GO TO FIN, pero para ello se requiere tener la certeza de que los registros que se van a grabar estén dispuestos en el archivo de entrada en orden ascendente y no haya claves duplicadas, ya que de ocurrir cualquiera de los casos la creación del archivo terminaría anormalmente sin poder conocer el programador cuál de las dos causas fue la que lo motivó.

Como resumen, el resultado de la ejecución del programa CREAINDX, es la creación de un archivo constituido por un índice de cilindros, un índice de pista en cada cilindro de los que componen el archivo, una serie de pistas llenas de registros colocados en orden secuencial ascendente de clave, y unas pistas de overflow vacías que podrán ser utilizadas para añadir, en posteriores trabajos, nuevos registros al archivo.

6.2- PROCESO DEL ARCHIVO

Una vez creado el archivo con organización secuencial indexada ya puede ser utilizado en la ejecución de trabajos que requieran un proceso que realice cualquiera de las siguientes acciones:

- Lectura (recuperación)
- Incorporación (adición) y
- Actualización de registros.

Estas acciones pueden llevarse a cabo en cualquiera de las dos formas de proceso:

- Secuencial y
- Al azar

La elección de una u otra forma de proceso depende del trabajo a realizar, ya que si de lo que se trata, por ejemplo, es de consultar en un archivo las existencias de determinadas piezas de repuesto de un almacén; consultar y/o actualizar una cuenta corriente desde una cualquiera de las sucursales de un banco, etc., el proceso al azar sería el más adecuado, ya que permitiría cumplir con las principales exigencias que requieren estos tipos de trabajos: rapidez en el tratamiento de las transacciones, y llevarlo a cabo en el orden en que éstas se producen.

Sin embargo, cuando el tipo de trabajo no requiera una rápida respuesta a una consulta, y el porcentaje de registros del archivo que han de ser tratados no sea muy elevado, puede ser aconsejable el proceso secuencial. En estos casos, se reunen las transacciones y se lleva a cabo con ellas una previa clasificación en el orden en que se encuentran en el archivo que se ha de utilizar.

A continuación se presentan una serie de programas en donde cada uno de ellos realiza una de las siguientes acciones: lectura, recuperación, o actualización. No obstante, en la página 324 se muestra un ejercicio en el que se combinan todas estas acciones en un mismo programa.

6.2.1.- LECTURA (Recuperación) SECUENCIAL

La lectura secuencial de archivos indexados es semejante a la de cualquier archivo en cinta, ya que el dispositivo de lectura se posiciona al comienzo del archivo y va leyendo los registros hasta llegar al final.

El siguiente ejemplo muestra el programa con el que se confecciona un listado de todo el archivo creado por el programa CREAINDX (pág. 292), contenido en el discpack 3340. En dicho listado se quiere imprimir el código del operario, apellidos y nombre, y el número de hijos.

```
001010  ID DIVISION.
001020  PROGRAM-ID. RECUSECU.
001030  REMARKS. PROGRAMA EJEMPLO DE LECTURA SECUENCIAL DE UN ARCHIVO
001040                   SECUENCIAL INDEXADO.
001050  ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070  FILE-CONTROL.
001080          SELECT ARCHIDISC  ASSIGN SYS006-DA-3340-I
001090                   ACCESS IS SEQUENTIAL                    <----
001100                   RECORD KEY CLAVED.                      <----
001110          SELECT IMPRESORA  ASSIGN SYS005-UR-1403-S.
001120  DATA DIVISION.
001130  FILE SECTION.
001140  FD  ARCHIDISC  BLOCK 10 RECORDS LABEL RECORD STANDARD.
001150  01  REGISDISC.
001160      02 CLAVED      PIC 9(5).
001170      02 APE-NOM     PIC X(35).
001180      02 FILLER      PIC X(10).
001190      02 NUM-HIJOS   PIC 99.
001200      02 FILLER      PIC X(68).
001210  FD  IMPRESORA   LABEL RECORD OMITTED.
001220  01  IMPRESO        PIC X(133).
001230  WORKING-STORAGE SECTION.
001240    01 SALIDA.
001250      02 FILLER      PIC X(40) VALUE SPACES.
002010      02 CLAVE       PIC 9(5).
002020      02 FILLER      PIC X(6)  VALUE SPACES.
002030      02 APE-NO      PIC X(35).
002040      02 FILLER      PIC X(6)  VALUE SPACES.
002050      02 NUM-HIJO    PIC 99.
002060  PROCEDURE DIVISION.
002070  ABRE.  OPEN INPUT ARCHIDISC  OUTPUT IMPRESORA.          <----
002080  LEER.  READ ARCHIDISC AT END GO TO FIN.                 <----
002090         MOVE CLAVED TO CLAVE MOVE APE-NOM TO APE-NO
002100         MOVE NUM-HIJOS TO NUM-HIJO
002110         MOVE SALIDA TO IMPRESO
002120         WRITE IMPRESO AFTER 2 GO TO LEER.
002130  FIN.   CLOSE ARCHIDISC IMPRESORA  STOP RUN.
```

NOTA: Tanto en éste como en los siguientes ejemplos, se señalan con una flecha aquellas cláusulas y declaraciones que puede ser necesario especificarlas con diferente formato, según el tipo de proceso a realizar y forma de acceso al archivo. El propósito es advertir al programador donde prestar más atención al escribir el programa, por ser la causa donde se origina el mayor número de errores.

La simple inspección del programa, hace que no necesite explicación adicional exhaustiva, ya que todas las declaraciones y cláusulas utilizadas se han descrito anteriormente. Así, la declaración READ (línea 002080) comienza la lectura del archivo a partir del primer registro y continúa hasta detectar el final; siendo la declaración WRITE (línea 002110) la que imprime los resultados deseados.

Una de las ventajas que tienen los dispositivos de acceso directo sobre las unidades de cinta es que por ser móviles sus cabezas de lectura / escritura, pueden situarse de forma casi instantánea en cualquier lugar del archivo. Esto permite comenzar la lectura del mismo a partir de cualquier registro, y continuarla secuencialmente hasta llegar al final del archivo.

Por tanto, si en el programa RECUSECU se quisiera iniciar la lectura del archivo ARCHIDISC a partir de un registro que no fuera el primero, sino por aquel que contuviera, por ejemplo, la clave 00182, habría que modificar dicho programa, intercalando en él las siguientes cláusulas y declaraciones.

```
001095                    NOMINAL KEY NOMIN
001230 WORKING-STORAGE SECTION.
001235   77 NOMIN              PIC 9(5) VALUE 00182.
002060 PROCEDURE DIVISION.
002075 INICIO.
002076                START ARCHIDISC  INVALID KEY  DISPLAY NOMIN ' = CLAVE IN
002077-               'VALIDA'  GO TO FIN.
```

Es conveniente, aunque no necesaria, la utilización de la declaración START cuyo formato es:

START nombre-de-archivo [INVALID KEY declaración imperativa]

La función de esta declaración es obligar a iniciar el proceso a partir de la clave especificada. Dicha declaración debe ejecutarse después de la declaración OPEN y antes de la READ. Si no existe en el archivo un registro con una clave igual a la clave en la que se quiere comenzar la lectura, sería una clave inválida, en cuyo caso pasaría el control a la declaración imperativa que sigue a INVALID, es decir, DISPLAY NOMIN'..... finalizando a continuación el proceso.

El empleo de la declaración START obliga a utilizar la cláusula NOMINAL (23) cuyo formato es:

NOMINAL KEY IS nombre-de-datos

La función de esta cláusula es permitir al Sistema localizar la posición física de un registro que tenga la misma clave que el contenido previamente colocado dentro de nombre-de-datos. Nombre-de-datos es un campo descrito dentro de la WORKING-STORAGE SECTION.

Así, en el ejemplo se ha introducido en el campo NOMIN la clave 00182, que es la que se quiere localizar en el archivo. La ejecución de la declaración START, obliga al Sistema a investigar dentro de cada registro en el campo de datos que se especifica en RECORD KEY, es decir, en CLAVED, una clave igual a la contenida en el campo NOMIN. Cuando exista coincidencia quiere decir que ha sido localizada y las cabezas de lectura están preparadas para proceder a la lectura del registro, la cual se llevará a cabo con la declaración READ (línea 002080).

La cláusula NOMINAL debe también emplearse cuando para recuperar, actualizar o añadir registros a un archivo indexado se hace al azar, es decir, siempre que se especifique ACCESS IS RANDOM (ver programas ejemplo en las páginas 300 a 304).

---

(23) En otros sistemas en vez de NOMINAL se denomina SYMBOLIC, siendo su función equivalente.

6.2.2.–ACTUALIZACION SECUENCIAL

El siguiente ejemplo muestra la forma de actualizar en forma secuencial una serie de registros del archivo indexado.

Los nuevos datos se encuentran en fichas, los cuales sustituirán a los ya existentes en el archivo indexado.

```
001010  ID   DIVISION.
001020   PROGRAM-ID.  ACTUASEC.
001030   REMARKS.  PROGRAMA PARA ACTUALIZAR SECUENCIALMENTE UN ARCHIVO
001040                    SECUENCIAL INDEXADO.
001050  ENVIRONMENT DIVISION.
001060    INPUT-OUTPUT SECTION.
001070     FILE-CONTROL.
001080              SELECT  ARCHIDISC  ASSIGN SYS006-DA-3340-I
001090                      ACCESS SEQUENTIAL                    <-------
001100                      RECORD KEY CLAVED.                   <-------
001110              SELECT  FICHAS  ASSIGN  SYS005-UR-2540R-S.
001120  DATA DIVISION.
001130    FILE SECTION.
001140     FD    ARCHIDISC  LABEL RECORD STANDARD BLOCK 10 RECORDS.
001150     01    REGISDIC.
001160              02  CLAVED      PIC 9(5).
001170              02  DATOSD      PIC X(115).
001180     FD    FICHAS   LABEL RECORD OMITTED.
001190     01    FICHA.
001200              02  CLAVEF      PIC 9(5).
001210              02  DATOSF      PIC X(75).
002010  PROCEDURE DIVISION.
002020    ABRE.  OPEN I-O ARCHIDISC    INPUT  FICHAS.             <-------
002030    LEEF.  READ FICHAS AT END  GO TO FIN.
002040    LEED.  READ ARCHIDISC AT END GO TO FIN.                 <-------
002050           IF CLAVED = CLAVEF MOVE DATOSF TO DATOSD
002060           ELSE GO TO LEED.
002070    GRABA.  REWRITE REGISDISC INVALID KEY GO TO FIN.  GO TO LEEF.  <---
002080    FIN.   CLOSE ARCHIDISC  FICHAS    STOP RUN.
```

Esta actualización consta en líneas generales de las siguientes operaciones: leer un registro para introducirlo en la memoria, sustituir los datos antiguos por los nuevos y volver a grabar dicho registro en el mismo lugar que ocupaba en el archivo.

Así, por medio de la declaración READ (línea 002040), lee e introduce en la memoria un registro. Si no es de los registros a actualizar se vuelve a leer un nuevo registro del archivo; caso contrario, se actualiza el registro (línea 002050) y se vuelve a grabar de nuevo en la misma posición física de la que se recuperó.

Esta acción la lleva a cabo la declaración REWRITE cuyo formato es el siguiente:

REWRITE nombre–de–registro [**FROM** identificador]

[**INVALID** KEY declaración imperativa]

La función de la opción INVALID es impedir que se vuelva a grabar (REWRITE) en el archivo, un registro que no haya sido previamente leído mediante una declaración READ. En este caso, no graba el registro y continúa la ejecución del programa en la declaración imperativa que sigue a INVALID; sin embargo, si el programador no hubiera especificado la opción INVALID el Sistema no sabría que acción realizar dando lugar a que el programa finalizara anormalmente.

En el programa ACTUASEC, aunque se especifica la opción INVALID (línea 002070) no es necesaria, ya que nunca se dará el caso de intentar reescribir en ARCHIDISC un registro que no haya sido previamente leído, pues dicha función la realiza la declaración READ (línea 002040).

Al igual que en la recuperación secuencial, debe el programador antes de ejecutar el programa, ordenar el archivo en fichas en secuencia ascendente por CLAVEF.

El archivo ARCHIDISC debe abrirse con la opción I–O, por utilizarse como archivo de entrada/salida.

6.2.3.– LECTURA (Recuperación) AL AZAR

El siguiente programa muestra la forma de recuperar al azar los registros del archivo indexado. Las claves de los registros a recuperar se encuentran en cualquier orden en un archivo en fichas. Los registros que se recuperen, se grabarán en un archivo en cinta.

```
Página  Línea  FD  A       B
1   3   4   6   7  8      12   15   20   25   30   35   40   45   50   55   60   65   70
001010    ID DIVISION.
001020    PROGRAM-ID. RECUAZAR.
001030    REMARKS. RECUPERACION AL AZAR DE REGISTROS DE UN ARCHIVO SECUENCI
001040                     AL INDEXADO.
001050    ENVIRONMENT DIVISION.
001060     INPUT-OUTPUT SECTION.
001070    FILE-CONTROL.
001080              SELECT ARCHIDISC ASSIGN SYS006-DA-3340-I
001090                     ACCESS IS RANDOM                          <------
001100                     RECORD KEY CLAVED                         <------
001110                     NOMINAL KEY NOMIN.                        <------
001120              SELECT FICHAS  ASSIGN SYS005-UR-2540R-S
001130              SELECT CINTA   ASSIGN SYS004-UT-2400-S.
001140    DATA DIVISION.
001150    FILE SECTION.
001160    FD  ARCHIDISC  LABEL RECORD STANDARD BLOCK 10 RECORDS.
001170    01  REGIS.
001180        02 CLAVED      PIC 9(5).
001190        02 DATOSD      PIC X(115).
001200    FD  FICHAS  LABEL RECORD OMITTED.
001210    01  FICHA.
001220        02 CODIGO      PIC 9(5).
001230        02 FILLER      PIC X(75).
```

```
002010  FD  CINTA   LABEL RECORD OMITTED.
002020  01  CINT        PIC X(120).
002030  WORKING-STORAGE SECTION.
002040  77  NOMIN      PIC 9(5).
002050  PROCEDURE DIVISION.
002060  ABRE.  OPEN INPUT  ARCHIDISC  FICHAS  OUTPUT  CINTA.
002070  LEE.   READ FICHAS AT END GO TO FIN.
002080         MOVE CODIGO TO NOMIN
002090  ------>READ ARCHIDISC  INVALID KEY DISPLAY CODIGO ' = CLAVE INVALI
002100  -        'DA'    GO TO LEE.
002110         MOVE REGIS  TO CINT
002120         WRITE CINT      GO TO LEE.
002130  FIN.   CLOSE ARCHIDISC  FICHAS  CINTA
002140         STOP RUN.
```

Los pasos necesarios para recuperar un registro al azar son:

El programador, previa a la declaración READ (línea 002090) para la lectura y localización de un registro, introduce en el campo que se especifica en la NOMINAL KEY: NOMIN (línea 002080) la clave del registro a localizar.

A continuación, por la declaración READ citada, el Sistema realiza automáticamente las siguientes funciones:

a) Investiga en el índice de cilindros la primera clave que sea igual o mayor a la del registro que se quiere localizar, con lo que obtiene el cilindro en que se encuentra dicho registro.

b) Investiga el índice de pista para este cilindro, hasta encontrar una clave igual o mayor a la del registro a recuperar, por ella conoce el Sistema en qué pista se encuentra el registro.

c) Investiga en dicha pista hasta que localiza el registro y lo introduce en la memoria.

Puede suceder, también, que el Sistema no localice el registro porque no exista en el archivo ninguno que contenga la citada clave, en cuyo caso, habrá que indicar al Sistema, qué acción deberá realizar. Dicha acción es la declaración imperativa que figura a continuación de INVALID.

En el ejemplo, el programador ha indicado la acción: DISPLAY CODIGO... GO TO LEE, es decir, que el Sistema escriba por la impresora aquellas claves, contenidas en el archivo FICHAS, que no se hallen en ningún registro del archivo ARCHIDISC, y regresar, posteriormente, al procedimiento LEE para continuar el proceso.

6. 2. 4.– GRABACION (Adiciones) AL AZAR

En la adición de nuevos registros a un archivo indexado pueden ocurrir los dos casos siguientes:

- Que la clave del registro a grabar esté comprendida entre las claves de dos registros ya grabados, o
- Que la clave del registro a grabar sea superior a la clave más alta grabada en el archivo.

En el primer caso, el registro se graba en el área de datos del cilindro,en la posición secuencial correcta dentro de la pista correspondiente, es decir, en aquel lugar que le corresponda por secuencia ascendente de clave, desplazándose los registros que le siguen en dicha pista una posición. El último registro desplazado de esa pista pasará al área de overflow de ese cilindro siempre que exista espacio en ella. Si la adición debe hacerse entre el último registro de una pista y el primero de la siguiente se graba en esta pista, pasando el último de élla al área de overflow; pero si la adición ha de hacerse entre el último registro de una pista y un registro situado con anterioridad en el área de overflow, se graba en dicha área. El Sistema lleva a cabo de esta forma la grabación, debido a que si los registros a intercalar se colocaran en secuencia numérica correcta, la adición de un nuevo registro supondría el desplazamiento de todos los registros que tuviesen claves más altas que las del registro a añadir.

Puede suceder que la adición de un número determinado de registros en un cilindro produzca el llenado de su área de overflow, en este caso el nuevo registro a añadir se situará en el área de excedentes independiente si el programador la ha reservado (mediante una ficha de control); caso de no haber previsto su reserva, no será posible grabar el nuevo registro, lo cual lleva consigo tener que reorganizar todo el archivo, es decir, efectuar una nueva creación.

En el segundo caso, el nuevo registro se graba en la última pista de datos que contenga registros; si dicha pista está llena,el registro se graba en el área overflow.

Esta es a grandes rasgos la forma con que lleva a cabo el Sistema la grabación de nuevos registros, en la cual, por no intervenir en estas acciones el programador, y no ser general para todos los Sistemas no hace necesaria una minuciosa descripción.

EJEMPLO.

El siguiente programa muestra la forma de incorporar al azar registros al archivo indexado.

Los nuevos registros se encuentran en fichas y no están ordenados por su clave.

```
001010 IDENTIFICATION DIVISION.
001020  PROGRAM-ID. ADICAZAR.
001030  REMARKS. ADICIONES A UN ARCHIVO INDEXADO A PARTIR DE
001040           OTRO ARCHIVO EN FICHAS.
001050 ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070   FILE-CONTROL.
001080       SELECT ARCHIFICH  ASSIGN SYS005-UR-2540R-S.
001090       SELECT ARCHIDISC  ASSIGN SYS006-DA-3340-I
001100                         ACCESS RANDOM                    <------
001110                         NOMINAL KEY NOMINA               <------
001120                         RECORD KEY IS CODIDISCO.         <------
001130 DATA DIVISION.
001140  FILE SECTION.
001150   FD ARCHIFICH RECORDING F  LABEL RECORD OMITTED.
001160   01 FICHA.
001170      02 CODIFICHA          PIC X(5).
001180      02 DATOSF             PIC X(75).
001190   FD ARCHIDISC       BLOCK  10 RECORDS
001200                      LABEL  RECORD  STANDARD.
001210   01 DISCO.
001220      02 CODIDISCO          PIC X(5).
001230      02 DATOSD             PIC X(115).
002010  WORKING-STORAGE SECTION.
002020   77 NOMINA                PIC X(5).
002030 PROCEDURE DIVISION.
002040  ABRIR. OPEN INPUT ARCHIFICH  I-O  ARCHIDISC.          <------
002050  LEER.  READ ARCHIFICH  AT END  GO TO FIN.
002060         MOVE CODIFICHA TO CODIDISCO    NOMINA
002070         MOVE DATOSF  TO  DATOSD
002080         WRITE DISCO INVALID KEY GO TO RUTINA-ERROR.      <------
002090         GO TO LEER.
002100  RUTINA-ERROR.
002110                 DISPLAY CODIFICHA  ' = CODIGO DUPLICADO'
002120         GO TO LEER.
002130  FIN.   CLOSE ARCHIFICH  ARCHIDISC
002140         STOP RUN.
```

La cláusula ACCESS IS RANDOM es necesaria tanto para grabar al azar nuevos registros como para actualizarlos. Al especificar esta cláusula obliga a especificar también la cláusula NOMINAL KEY.

Por ser una grabación al azar debe abrirse el archivo con la opción I–O.

Durante el proceso, puede suceder que alguna de las claves de los nuevos registros a añadir se encuentre ya grabada en el archivo, es decir, sea una clave inválida, en este caso la opción INVALID KEY de la declaración WRITE impide la escritura de dicho registro y ejecuta la declaración que sigue a INVALID. También impediría dicha cláusula la escritura de un registro cuando se encuentren llenas el área de overflow del cilindro donde debía grabarse dicho registro y el área de excedentes independiente.

6.2.5.– ACTUALIZACION AL AZAR

Con el siguiente programa se actualizan los sueldos de los operarios en un número determinado de registros contenidos en el archivo indexado. Las modificaciones a introducir se encuentran en fichas, no estando ordenadas en secuencia de clave (código del operario).

```
000100 ID DIVISION.
000200 PROGRAM-ID. ACTUAZAR.
000300 REMARKS. PROGRAMA PARA ACTUALIZAR AL AZAR UN ARCHIVO SECUENCIAL
000400                 INDEXADO.
000500 ENVIRONMENT DIVISION.
000600 INPUT-OUTPUT SECTION.
000700 FILE-CONTROL.
000800     SELECT ARCHIDISC  ASSIGN SYS006-DA-3340-I
000900                 ACCESS IS RANDOM                          <----
001000                 NOMINAL KEY  NOMINA                       <----
001100                 RECORD KEY  CLAVE.                        <----
001200     SELECT FICHAS     ASSIGN  SYS005-UR-2540R-S.
001300 DATA DIVISION.
001400 FILE SECTION.
001500 FD  ARCHIDISC  LABEL RECORD STANDARD BLOCK 10 RECORDS.
001600 01  REGISD.
001700     02 CLAVE          PIC 9(5).
001800     02 FILLER         PIC X(53).
001900     02 SUELDO         PIC 9(5).
002000     02 FILLER         PIC X(57).
002100 FD  FICHAS  LABEL RECORD OMITTED.
002200 01  FICHA.
002300     02 CLAVEF         PIC 9(5).
002400     02 SUELDOF        PIC 9(5).
002500     02 FILLER         PIC X(70).
002010 WORKING-STORAGE SECTION.
002020 77  NOMINA     PIC 9(5).
002030 PROCEDURE DIVISION.
002040 ABRE.  OPEN I-O ARCHIDISC  INPUT FICHAS.                  <----
002050 LEE.   READ FICHAS  AT END  GO TO FIN.
002060       MOVE CLAVEF TO NOMINA
002070 ----->  READ ARCHIDISC  INVALID  KEY DISPLAY  CLAVEF  '= NO EXISTE
002080 -             'UN REGISTRO CON ESTA CLAVE'  GO TO LEE.
002100       MOVE SUELDOF TO SUELDO
002110 ----->  REWRITE REGISD  INVALID KEY  DISPLAY CLAVE  '= CLAVE INVALI
002120 -             'DA'. GO TO LEE.
002130 FIN.   CLOSE ARCHIDISC  FICHAS  STOP RUN.
```

Se podía haber especificado la declaración REWRITE sin la opción INVALID KEY, ya que de existir en el archivo en fichas una clave que no se encontrara en el archivo indexado, resultaría ser una clave inválida, la cual sería detectada por la declaración READ (línea 002070).

7.- ORGANIZACION DIRECTA

Al describir la organización secuencial indexada, se vió que no era función del programador llevar el control de la dirección o posición que ha de ocupar cada registro en el espacio de almacenamiento reservado para la creación del archivo, ya que dicha función la realizaba automáticamente el Sistema Operativo. Sin embargo, en la organización directa el programador es responsable de la lógica y de la programación necesaria para proporcionar al Sistema una dirección para cada registro, donde situarlo físicamente dentro del espacio de almacenamiento reservado para el archivo.

Para cumplir esta función el programador utiliza dentro del programa un ALGORITMO con el que establece una relación matemática entre la clave de cada registro y la dirección que ha de ocupar en el archivo (ver algoritmos de las páginas 308, 315 y 320), es decir, aplicando una fórmula matemática a la clave de cada registro, se obtiene la dirección que el mismo ha de ocupar dentro del archivo.

Sin embargo, no existe un algoritmo único que pueda utilizarse en la creación de cualquier archivo, y por tanto es muy importante elegir el más adecuado, ya que una mala elección podía dar lugar a que:

- Existan pistas totalmente llenas y otras que no contengan ningún registro.
- Se obtenga la misma dirección donde grabar registros cuya clave es diferente, debiendo, en ese caso, asignar nuevas direcciones a dichos registros.

Por tanto, el algoritmo que se elija debe tender a:

- Distribuir uniformemente los registros a lo largo del archivo para un mejor aprovechamiento del almacenamiento reservado.
- Reducir al mínimo posible la obtención de direcciones iguales para diferentes registros, a fin de reducir el tiempo de proceso necesario tanto para su creación como para su localización.

Es importante resaltar que así como en la creación de un archivo con organización indexada el Sistema Operativo evitaba la grabación de aquellos registros que contuvieran la misma clave, sin embargo, con la organización directa, el programador es el encargado de programar la forma de evitar dicha grabación; caso de no hacerlo, y, por error, existen dos o más registros con clave repetida, dichos registros se grabarán en el archivo.

Como resumen se puede decir que la programación en este tipo de organización resulta ser más compleja, y requiere más tiempo la puesta a punto para cada programa, ya que por existir más intervención humana está más expuesta a errores. Por tanto, el empleo de la organización directa normalmente debe utilizarse para aquellos archivos cuyas características no hagan aconsejable el empleo de la organización secuencial o la secuencial indexada.

7.1.- METODOS DE DIRECCIONAMIENTO

El programador al poder decidir sin limitaciones la dirección donde ha de grabarse cada registro dentro del archivo o donde posteriormente ha de ser localizado, es el responsable de elegir el método de direccionamiento más adecuado que proporcione dichas direcciones teniendo en cuenta:

- El tipo de las claves de los registros –numéricas o alfanuméricas–.
- Mayor o menor número de claves utilizadas dentro del rango asignado (24).
- Futuro empleo que se vaya a hacer del archivo (añadir, actualizar, etc.,).

Estas consideraciones resultan ser problemas cuya solución se hallará al hacer la descripción de los métodos de direccionamiento:

- DIRECTO.
- INDIRECTO.

Cada uno de estos métodos puede utilizar para posicionar o localizar los registros dentro del archivo:

- Direcciones relativas de pista (25).
- Direcciones reales (absolutas) de pista.

---

(24) Rango de claves, son los límites inferior y superior entre los cuales el programador puede elegir cualquier clave para asignarla a cada registro.

(25) Dentro de un volumen de discos (discpack) pueden grabarse varios archivos, por lo que sólamente uno de ellos comenzará su grabación en la primera pista del citado volumen. Por este motivo, el algoritmo que se especifique dentro de cada programa debe ser independiente del número de la pista en la que se ha decidido comience la grabación del archivo. Es decir, el número que se obtenga por la aplicación del algoritmo a la clave de cada registro (Dirección Relativa) habrá que incrementarlo al número de la pista en la que se ha decidido comience la grabación del archivo, siendo el resultado (Dirección Real) el número de la pista donde ha de grabarse el registro. Asi, por ejemplo, si al aplicar el algoritmo a la clave de un registro se obtiene el número 2.150 (Dirección relativa de la pista), y el programador decidió comenzar la grabación del archivo a partir de la pista 840 del discpack, quiere decir que dicho registro se grabará en la pista 2.150 + 840 = 2.990 (Dirección real).

Este cálculo puede el Sistema realizarlo de dos formas:

- La primera consiste en especificar dentro del programa, únicamente el algoritmo (ver pág. 308, línea 002080), y el Sistema obtiene la dirección real: Sumando al número de la pista en la que ha de comenzar la grabacion del archivo (contenido en una ficha de control) el obtenido por el algoritmo.
- La segunda consiste en especificar dentro del programa el algoritmo y una serie de declaraciones en las que se indican al Sistema las operaciones que debe realizar para obtener la dirección real (ver pág. 321, líneas 003170 a la 003190).

Existen Sistemas para los que no es necesario indicarles donde debe comenzar la grabación del archivo, ya que ellos se encargan de gestionar dicho comienzo, teniendo en cuenta las pistas libres que queden en el volumen de discos.

7.1.1.- DIRECCIONAMIENTO DIRECTO

Direccionamiento directo es aquél en el que cada clave posible del archivo se convierte en una DIRECCION UNICA, lo cual implica:

a) Que las claves de los registros sean numéricas.

b) Que el algoritmo elegido no direccione (envíe) hacia una determinada pista más del número de registros que pueda contener.

c) Que el programador debe reservar en el discpack espacio para todas las claves posibles, las utilice o no, dentro del rango asignado, es decir, si el rango de claves que puede utilizar el programador está acotado entre la 00001 y 20500, debe reservar espacio en el discpack para almacenar 20500 registros, aunque en el momento de la creación del archivo solo se graben 15000, y el resto pueda servir como espacio reservado para añadir nuevos registros en procesos posteriores.

Por tanto, se puede deducir que el direccionamiento directo se aplica principalmente cuando:

- Los registros contienen claves numéricas.
- Existe un porcentaje muy alto de claves utilizadas dentro del rango asignado.

7.1.1.1.- CREACION DEL ARCHIVO

El programa siguiente, muestra la creación de un archivo empleando direccionamiento directo.

- El archivo a crear ARCHIDIS se grabará con la información contenida en los registros del archivo secuencial indexado ARCHINDX, el cual fue creado en el programa CREAINDX (pág. 292).
- La longitud de los registros en el archivo que se va a crear será de 165 posiciones.
- Se supone que el rango de las claves va de la 00001 a la 10000; existiendo 1500 claves que no han sido utilizadas.

La solución del problema va a consistir en líneas generales, en leer cada uno de los registros del archivo indexado, aplicar un algoritmo para calcular el número relativo de la pista donde se ha de grabar cada registro, y por último, la grabación de los mismos.

```
Página Línea  S A B
1  3   4  6  7 8  12   15   20   25   30   35   40   45   50   55   60   65   70
001 010   ID DIVISION.
001 020   PROGRAM-ID. DDIRECTO.
001 030   REMARKS. CREACION ARCHIVO POR DIRECCIONAMIENTO DIRECTO CON DIREC
001 040            CION RELATIVA DE PISTA A PARTIR DE OTRO ARCHIVO EN DISCO.
001 050   ENVIRONMENT DIVISION.
001 060    INPUT-OUTPUT SECTION.
001 070     FILE-CONTROL.
001 080          SELECT ARCHIDIS ASSIGN SYS005-DA-3340-D
001 090                 ACCESS IS RANDOM                          <------
001 100                 ACTUAL KEY IS ACTUALI.                    <------
001 110          SELECT ARCHINDX ASSIGN SYS006-DA-3340-I
001 120                 RECORD KEY CLAVEI.
001 130   DATA DIVISION.
001 140    FILE SECTION.
001 150     FD ARCHIDIS RECORDING F LABEL RECORD STANDARD.
001 160     01 DISCO.
001 170          02 CLAVED      PIC 9(5).
001 180          02 DATOSD      PIC X(160).
001 190     FD ARCHINDX RECORDING F LABEL RECORD STANDARD
001 200                 BLOCK 10 RECORDS.
001 210     01 DISCIN.
001 220          02 CLAVEI      PIC 9(5).
001 230          02 DATOSI      PIC X(115).
002 010   WORKING-STORAGE SECTION.
002 020    01 ACTUALI.                                            <------
002 030          02 PISTA       PIC S9(8) COMP.
002 040          02 CLAVEREG    PIC 9(5).
002 050   PROCEDURE DIVISION.
002 060    ABRE. OPEN INPUT ARCHINDX OUTPUT ARCHIDIS.
002 070    LEER. READ ARCHINDX AT END GO TO FIN.
002 080          DIVIDE 20 INTO CLAVEI GIVING PISTA.
002 090          MOVE CLAVEI TO CLAVEREG CLAVED
002 100          MOVE DATOSI TO DATOSD
002 110          WRITE DISCO INVALID KEY GO TO RUTINA-ERROR. GO TO LEER.
002 120    RUTINA-ERROR. DISPLAY CLAVEI 'NO HAY PISTAS SUFICIENTES'.
002 130    FIN.  CLOSE ARCHIDIS ARCHINDX
002 140          STOP RUN.
```

El trabajo previo del programador a la escritura del programa es elegir el algoritmo que ha de emplear para convertir cada clave en una dirección única.

En este programa se ha elegido (línea 002080) un algoritmo de acuerdo con las características del archivo a crear (clave de los registros numéricas y utilización de un porcentaje muy alto de claves) consistente en:

- Dividir la clave de cada registro por el número de registros que pueden grabarse en cada pista (26).
- El cociente obtenido indica al Sistema la dirección relativa de la pista.

En la línea 001080 se especifica –D– para indicar al Sistema que el tipo de organización con la que se va a crear el archivo es: DIRECTA.

---

(26) El número de registros que pueden grabarse en cada pista depende de la longitud del registro y de la longitud de la pista. En cada manual que describe las características del dispositivo de acceso directo que se utilice, se muestran unas tablas en las que se refleja el número de registros que caben en cada pista, según la longitud de los mismos.

En la línea 001100 se especifica la cláusula ACTUAL KEY cuya función es proporcionar al Sistema una dirección donde grabar un registro lógico dentro de un dispositivo de almacenamiento de acceso directo.

Su formato es:

ACTUAL KEY IS nombre–de–datos

Nombre–de–datos (ACTUAL I) es un campo compuesto, conteniendo la siguiente información:

a) Identificador de pista (PISTA, línea 002030), el cual indica la dirección relativa de pista.

b) Identificador de registro (CLAVEREG, línea 002040), cuyo contenido es la clave del registro que sirve para distinguirlo de los demás registros grabados en el archivo.

El campo denominado PISTA debe describirse siempre con PICTURE 9(8) y USAGE COMP por exigirlo así el Sistema, sin embargo, CLAVEREG debe describirse con la misma PICTURE que la clave de cada registro. Ambos campos se han descrito dentro de la WORKING-STORAGE SECTION.

Antes de la grabación de cada registro debe transferirse a dichos campos la información correspondiente. Efectivamente, después de leer un registro del archivo de entrada (línea 002070) se aplica a la clave del registro el algoritmo elegido, por el cual divide dicha clave por 20 (número de registros que caben en cada pista) guardando el cociente en el campo PISTA; al campo CLAVEREG se le transfiere la clave del registro leído por medio de la declaración MOVE de la línea 002090. Una vez completa la información de la ACTUAL KEY sólo queda transferir toda la información contenida en el registro leído al registro que se ha de grabar, lo cual se realiza por las declaraciones MOVE de las líneas 002090 y 002100.

En este momento, el Sistema está en disposición de grabar el registro, lo cual ejecuta mediante la declaración WRITE (línea 002110). Sin embargo, puede suceder que al intentar la grabación de un registro, ésta no pueda llevarse a cabo por los motivos siguientes:

- El programador ha reservado para el archivo menor número de las pistas necesarias para grabar todos los registros, o que

- El número calculado para ser utilizado en el algoritmo, a fin de dividirlo por la clave de cada registro, sea superior al número de registros que es capaz de contener cada pista.

En el primer caso, tan pronto se llenen las pistas reservadas para el archivo, el intento de grabar un nuevo registro hace que se cumpla la condición INVALID KEY, en cuyo caso, no graba el registro y ejecuta la declaración: GO TO RUTINA-ERROR, finalizando anormalmente la ejecución del programa.

Tampoco se grabaría si ocurriera el segundo caso, ya que daría lugar a que se intentara la grabación de un nuevo registro en una pista ya completa, lo cual haría cumplir la condición INVALID KEY, realizando las mismas acciones que en el párrafo anterior.

Si, como es lógico, en la creación del archivo no se quieren grabar registros con la misma clave, es necesario que el programador las compruebe o verifique, a fin de eliminar o corregir aquellas que esten duplicadas, ya que en la organización directa el Sistema no detecta esta duplicidad.

Así mismo si se quieren situar los registros dentro del archivo siguiendo una determinada secuencia, se debe, antes de iniciar la creación del mismo, ordenarlos en dicha secuencia, ya que el Sistema tampoco realiza verificación alguna en la ordenación de los registros.

No existen estos problemas con la ejecución del programa DDIRECTO, ya que se ha creado un archivo que por ser copia del archivo secuencial indexado ARCHINDX:

- No hay posibilidad de que existan registros con claves repetidas, y

- Los registros están colocados en secuencia ascendente de clave a partir de la primera pista; si bien y debido a que existen 1500 claves sin utilizar puede que alguna de las pistas estén llenas y otras que sólo contengan alguno o ningún registro.

Un archivo creado con direccionamiento directo presenta como ventajas:

- Poder localizar cualquier registro en el archivo con una sola búsqueda, ya que a cada registro le corresponde una dirección única, lo cual reduce, al mínimo, el tiempo necesario cuando dicha localización se realiza al azar.

- Permite un proceso secuencial, ya que los registros como consecuencia del algoritmo aplicado se encuentran grabados en el archivo en secuencia ascendente de clave.

Una posible desventaja podía ser que el programador al tener que reservar en el almacenamiento (discpack), espacio para todas las claves posibles dentro del rango que tenga asignado, las utilice o no, si el número de claves utilizadas fuera muy reducido se estaría haciendo un mal empleo de dicho almacenamiento. Esta desventaja dejaría de ser, si se tuviera previsto para posteriores procesos añadir nuevos registros al archivo con las claves que no fueron utilizadas durante su creación.

7.1.1.2.– ACTUALIZACION DEL ARCHIVO

El siguiente programa actualiza el archivo creado con el programa anterior (DDIRECTO), con el contenido de los registros grabados en una cinta magnética.

La actualización va a consistir en:

- Añadir nuevos registros (dar de alta) que se encuentren en el archivo en cinta pero no en el archivo en disco, o en
- Sustituir los datos antiguos de los registros grabados en el disco por los datos de los grabados en la cinta.

Para reconocer a cuál de los dos casos se refiere, cada registro de la cinta lleva grabado en la última posición las cifras 1 ó 2 respectivamente.

```
001010  ID DIVISION.
001020  PROGRAM-ID. ACTUDIRE.
001030  REMARKS. ACTUALIZA UN ARCHIVO CREADO CON DIRECCIONAMIENTO DIREC-
001040           TO CON LAS TRANSACCIONES CONTENIDAS EN UN ARCHIVO EN CINTA.
001050  ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070  FILE-CONTROL.
001080      SELECT ARCHIDIS  ASSIGN SYS005-DA-3340-D
001090                       ACTUAL KEY IS ACTUALY
001100                       ACCESS IS RANDOM.
001110      SELECT ARCHICIN  ASSIGN SYS006-UT-2400-S.
001120  DATA DIVISION.
001130  FILE SECTION.
001140  FD ARCHICIN  RECORDING F  LABEL RECORD OMITTED.
001150  01 CINTA.
001160     02 CLAVEC     PIC 9(5).
001170     02 DATOSC     PIC X(160).
001180     02 CLAV       PIC 9.
001190  FD ARCHIDIS  RECORDING F  LABEL RECORD STANDARD.
001200  01 DISCO.
001210     02 CLAVED     PIC 9(5).
001220     02 DATOSD     PIC X(160).
002010  WORKING-STORAGE SECTION.
002020  01 ACTUALY.
002030     02 PISTA      PIC S9(8) COMP.
002040     02 CLAVEREG   PIC 9(5).
002050  PROCEDURE DIVISION.
002060  ABRE. OPEN INPUT ARCHICIN  I-O ARCHIDIS.
002070  LEER. READ ARCHICIN AT END  GO TO FIN.
002080        DIVIDE 20 INTO CLAVEC  GIVING PISTA
002090        GO TO ALTA  ACTUALIZA  DEPENDING ON CLAV.
002100  ALTA. MOVE CLAVEC TO  CLAVED  CLAVEREG
002110        MOVE DATOSC TO  DATOSD
002120        WRITE DISCO  INVALID KEY GO TO FIN.
002130        GO TO LEER.
002140  ACTUALIZA.
002150        MOVE CLAVEC  TO CLAVEREG
002160        READ ARCHIDIS  INVALID KEY  GO TO FIN.
002170        MOVE DATOSC TO DATOSD
002180        WRITE DISCO  INVALID KEY  GO TO FIN.  GO TO LEER.
002190  FIN.  CLOSE ARCHIDIS  ARCHICIN
002200        STOP RUN.
```

Observando el programa, hay que hacer las siguientes precisiones:

La cláusula ACTUAL KEY proporciona al Sistema una dirección que utilizará, unas veces para localizar un registro en el archivo ARCHIDIS cuando haya que modificar sus datos, y otras para indicar la posición donde ha de grabarse un nuevo registro. Tanto para localizar un registro como para grabar uno nuevo debe transferirse previamente a los campos PISTA y CLAVEREG que componen dicha cláusula, la información correspondiente (línea 002080, 002100 y 002150).

El algoritmo a emplear para obtener el valor del campo PISTA debe ser el mismo que el utilizado en la creación del archivo (línea 002080).

El archivo ARCHIDIS debe abrirse con la opción I—O por ser un archivo de entrada/salida, ya que para actualizar un registro debe primero localizarse en el archivo, en cuyo caso, está haciendo de archivo de entrada, se introduce en la memoria, se modifican sus datos y al volverlo a escribir de nuevo en el lugar de donde se extrajo, está haciendo de archivo de salida.

Al intentar añadir un nuevo registro al archivo ARCHIDIS, por medio de la declaración WRITE (línea 002120), podía cumplirse la condición INVALID KEY, cuando:

- El número de pista obtenido por el algoritmo, donde se debía añadir un nuevo registro, quede fuera de los límites del almacenamiento reservado al archivo. En cuyo caso no se grabaría el registro y terminaría anormalmente el programa.

Al intentar localizar un registro por medio de la declaración READ (línea 002160) para ser actualizado podía cumplirse la condición INVALID KEY:

- Cuando al investigar el Sistema en la dirección de la pista obtenida por el algoritmo, no encuentre en ella la clave de registro que intenta localizar, o

- Cuando la pista obtenida por el algoritmo quede fuera de los límites del archivo.

Si se cumple cualquiera de los dos casos al no localizar el registro bifurcaría a FIN, terminando anormalmente el programa.

La declaración WRITE (línea 002180) podía haberse sustituido por REWRITE siempre que en la línea 001080 se hubiera sustituído —D— por —W—.

7.1.2.- DIRECCIONAMIENTO INDIRECTO

Con el direccionamiento directo, el programador debía reservar un lugar en el almacenamiento en disco por cada clave o código posible comprendidos dentro de un determinado rango, estuvieran o no asignados a un registro; y con el algoritmo que se aplicaba a cada clave se obtenía una dirección única de grabación para cada registro.

Sin embargo, existen numerosos casos en que por diferentes motivos no es recomendable o no es posible el empleo del direccionamiento directo para organizar un archivo.

No es recomendable el direccionamiento directo, por ejemplo, cuando el porcentaje de claves utilizadas para el conjunto de los registros que se han de grabar sea reducido en comparación con el número de claves comprendidas dentro del rango asignado , ya que ello obligaría a reservar gran cantidad de espacio de almacenamiento del que la mayor parte no sería utilizado. Es el caso de un archivo cuyo rango de claves estuviera comprendido entre la 1AA000 y la 5MB425, de las que únicamente se utilizaran 1000 claves.

No es posible el direccionamiento directo cuando al aplicar el algoritmo a la clave no se obtenga una dirección única para grabar cada registro, es decir, cuando se producen iguales direcciones de grabación para claves diferentes, caso muy frecuente cuando se emplean claves alfanuméricas para los registros. En este caso, cuando de la aplicación del algorimo se obtengan iguales direcciones de grabación, originará el llenado de unas pistas, otras quedarán parcialmente llenas, y otras que no contendrán un sólo registro, lo cual daría lugar a que un registro que hubiera de grabarse en una determinada pista que se encuentra llena, no podría llevarse a cabo, debiendolo hacer, bien en la primera pista sucesiva donde encuentre espacio o en las áreas de overflow (según se describe en páginas siguientes). A estos registros que no pueden grabarse en la dirección obtenida se les denomina SINONIMOS excedentes.

El direccionamiento indirecto trata de resolver estos problemas empleando un algoritmo con el que, a ser posible, las direcciones que se obtengan para grabar todo el archivo, aunque no sean únicas, se puedan condensar o concentrar en el menor espacio de almacenamiento posible, es decir, un algoritmo que reuna las siguientes características:

- Produzca un mínimo de sinónimos excedentes, ya que tanto para la creación como para la localización de un registro se necesita de un tiempo adicional al empleado para un registro normal.

- Distribuya uniformemente las direcciones a lo largo de todo el espacio reservado, a fin de economizar almacenamiento.

A lo largo de la descripción del método de direccionamiento indirecto se muestran una serie de ejemplos para la creación, actualización, etc., de archivos en los que para obtener la dirección en que se ha de grabar cada registro se emplean: unos algoritmos que proporcionan el número de la pista (Dirección Relativa), y otros que porporcionan el número de la pista y del cilindro (Dirección Real).

7.1.2.1.– CREACION DEL ARCHIVO (Dirección Relativa de Pista)

El siguiente programa muestra la creación de un archivo utilizando direccionamiento indirecto. Los registros que han de formar el nuevo archivo se toman de otro archivo grabado, también en disco, con organización secuencial.

El número de registros a grabar es de 1600, siendo su clave: alfanumérica de seis caracteres.

```
Página  Línea  Cont  A       B
1  3    4  6   7     8   12  15  20  25  30  35  40  45  50  55  60  65  70
000100 10  ID DIVISION.
000102 20  PROGRAM-ID. DINDIREC.
000103 30  REMARKS. CREACION ARCHIVO POR DIRECCIONAMIENTO INDIRECTO CON DI-
000104 40                   RECCION RELATIVA DE PISTA A PARTIR DE ARCHIVO EN DISCO.
000105 50  ENVIRONMENT DIVISION.
000106 60  INPUT-OUTPUT SECTION.
000107 70  FILE-CONTROL.
000108 80      SELECT ARCHIDIS  ASSIGN SYS005-DA-3340-D
000109 90                   ACCESS RANDOM
000110 00                   ACTUAL KEY IS ACTUALY.
000111 10      SELECT ARCHISEC  ASSIGN SYS006-DA-3340-S.
000112 20  DATA DIVISION.
000113 30  FILE SECTION.
000114 40      FD ARCHIDIS  RECORDING F  LABEL RECORD STANDARD.
000115 50      01 DISCO.
000116 60          02 CLAVED        PIC X(6).
000117 70          02 DATOSD        PIC X(160).
000118 80      FD ARCHISEC  RECORDING F  LABEL RECORD STANDARD.
000119 90      01 DISC.
000120 00          02 CLAVEC        PIC X(6).
000121 10          02 DATOSC        PIC X(160).
000122 20  WORKING-STORAGE SECTION.
000123 30  77 CLAVE                  PIC S9(6) COMP.
000124 40  77 COCIENTE               PIC 9(5).
000125 50  01 ACTUALY.
000201 10      02 PISTA             PIC S9(8) COMP.
000202 20      02 CLAVEREG          PIC X(6).
000203 30  01 CAMPO-ERRORES.
000204 40      02 FILLER            PIC 99 VALUE ZERO.
000205 50      02 OCT-ERROR         PIC 9  VALUE ZERO.
000206 60      02 FILLER            PIC 9(5) VALUE ZERO.
000207 70  PROCEDURE DIVISION.
000208 80  DECLARATIVES.
000209 90      ERRORES SECTION.
000210 00          USE AFTER STANDARD ERROR PROCEDURE ON ARCHIDIS
000211 10              GIVING CAMPO-ERRORES.
000212 20      RUTINA-ERROR.  IF OCT-ERROR = 1 GO TO SINONIMO
000213 30                  ELSE DISPLAY  CLAVEREG 'OTRO TIPO ERROR' GO TO FIN.
000214 40      SINONIMO.
000215 50          IF PISTA < 96    ADD 1 TO PISTA.  GO TO GRABAR.
000216 60  END DECLARATIVES.
000217 70  ABRE. OPEN INPUT ARCHISEC OUTPUT ARCHIDIS.
000218 80  LEER. READ ARCHISEC AT END GO TO FIN.
000219 90          MOVE CLAVEC TO CLAVE  CLAVEREG  CLAVED
000220 00          MOVE DATOSC TO DATOSD
000221 10          DIVIDE  CLAVE  BY  89  GIVING COCIENTE REMAINDER PISTA.
000222 20  GRABAR.
000223 30              WRITE DISCO  INVALID KEY  DISPLAY CLAVED 'NO HAY PISTAS SU
000224 40-             'FICIENTES' GO TO FIN.  GO TO LEER.
000225 50  FIN.  CLOSE ARCHIDIS  ARCHISEC  STOP RUN.
```

El trabajo preliminar del programador a la escritura del programa consiste en la elección del algoritmo apropiado. Es muy utilizado, por su sencillez, el denominado DIVISION/RESTO consistente en:

- Dividir la clave de cada registro por el número primo más próximo, por defecto, al número de pistas necesarias para grabar todos los registros incrementados en el veinte por ciento.
- El resto de la división proporciona la dirección relativa de la pista donde se ha de grabar el registro.

Así, en la línea 002210 se especifica el algoritmo utilizado, en el que el número primo 89 se obtiene de la forma siguiente:

- Número de registros de 166 caracteres que caben en cada pista de un dispositivo de acceso directo del modelo 3340 es igual a 20.
- Número de posiciones que se preveen son necesarias para grabar 1600 registros:

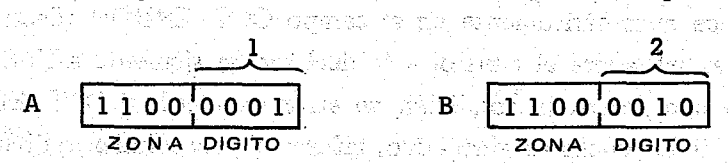
  Para reducir al mínimo los sinónimos excedentes, se asigna normalmente un 20 por ciento más del almacenamiento que sería necesario para poder grabar todos los registros sin que quedasen espacios libres en las pistas, es decir,

$$1600 + \frac{20}{100} \cdot 1600 = 1920$$

Por tanto, el número de pistas necesarias para grabar todo el archivo será: 1920 : 20 = 96.

- El número primo más próximo a 96 es 89.

Al igual que en la creación de los archivos con direccionamiento directo, se lee el archivo donde se encuentran los registros que han de grabarse en el archivo que se va a crear (línea 002180) y, a continuación, se transfiere a los campos CLAVEREG (línea 002190) la clave del registro, y a PISTA (línea 002210) el número de la pista que se obtenga de la aplicación del algoritmo, para lo cual, por ser alfanuméricas las claves de los registros, para poder efectuar la división de cada una de ellas por 89, es necesario desproveerlas previamente de la parte de zona, es decir, si la clave es 1AB045, una vez desprovista de la zona resultará el número 112045, ya que la configuración interna de las letras A y B son respectivamente:



En el programa, la eliminación de la parte de zona se realiza transfiriendo (línea 002190) la clave del registro que se acaba de leer: CLAVEC (línea 001160) al campo denominado CLAVE (línea 001230) descrito con PIC 9(6) y USAGE COMP.

Una vez completa la información y ejecutada la declaración MOVE (línea 002200), sería el momento en que debería llevarse a cabo la grabación del registro por medio de la declaración WRITE (línea 002230). Sin embargo, a lo largo de la ejecución del programa podía darse el caso en que el estado de ocupación de las pistas fuera el que muestra la figura 24, cuando un nuevo registro hubiera de grabarse en la pista N que se encuentra llena. En esta caso no podría grabarse en dicha pista y se intentaría su grabación en la pista siguiente; y así sucesivamente hasta encontrar una pista donde hubiera espacio para grabarlo. Observando dicha figura, debería realizarse su grabación en la pista N+2.

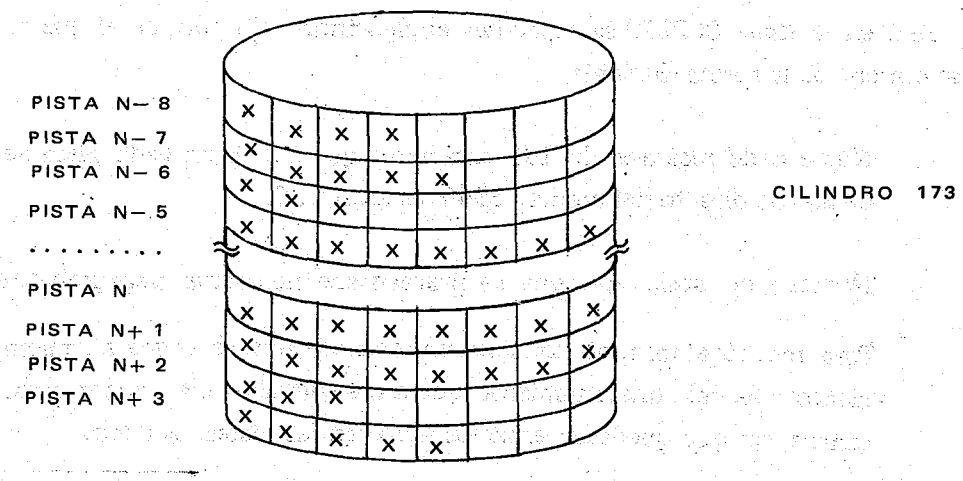


FIGURA 24

NOTA : La cruz (x) significa que en la posición donde se encuentra, está grabado un registro.

Para solucionar este problema, el programador debe indicar al Sistema las acciones que debe llevar a cabo antes de la grabación de cada registro en previsión de que se produzca este tipo de sinónimo que debía haberse grabado en una pista que en ese momento está llena. Las declaraciones comprendidas desde la línea 002080 a la 002160, van a indicar al Sistema que cualquier registro que no pueda grabarse en la pista obtenida por el algoritmo, se grabe en la pista sucesiva donde haya espacio disponible. Efectivamente, la declarativa USE AFTER STANDARD ERROR obliga al Sistema a comprobar antes de grabar cualquier registro (línea 002230) si existe espacio disponible en la dirección de pista obtenida por el algoritmo, caso afirmativo graba el registro, y en caso contrario, por no existir espacio se produce una condición de error, por lo que se introduce automáticamente en el campo OCT–ERROR (descrito en la línea Q02050) el valor 1 y se transfiere el control a la declaración siguiente a USE, es decir, a IF (línea 002120) donde comprueba la condición de error standard: si OCT–ERROR es igual a 1; como el resultado de la prueba es afirmativo, bifurca el procedimiento SINONIMO. Según se especifica en dicho procedimiento, si el contenido del campo PISTA (número de la pista donde se ha de grabar el registro) es menor de noventa y seis, incrementa en una unidad el valor de dicha pista y regresa al procedimiento GRABAR (línea 002220) para ver si puede grabarlo en la nueva pista, caso afirmativo lo hace y en caso contrario vuelve a incrementar en una unidad el valor del campo PISTA,repitiéndose el procedimiento descrito.

Si después de recorrer las pistas sucesivas para encontrar un espacio donde grabar el registro, se llegara al final de la pista 95 (para el archivo se han reservado 96 pistas, desde la 0 a la pista 95), al sumarle (línea 002150) una unidad al número de la pista y regresar a la declaración WRITE (línea 002230), resulta entonces que se estaría intentando grabar más allá de los límites asignados al archivo, con lo cual se cumpliría la condición INVALID, la cual impediría su grabación; imprimiéndose por la impresora la clave del registro que se quería grabar junto con el mensaje 'NO HAY PISTAS . . . . .' finalizando la ejecución del programa.

Por tanto, la sentencia USE tiene como función obligar al Sistema a ejecutar las rutinas de error automáticas, antes de la ejecución de determinados procedimientos especificados por el programador.

Su formato es:

```
USE AFTER STANDARD ERROR PROCEDURE
        ⎧ {nombre-de-archivo-1}...                                  ⎫
        ⎪  nombre-de-archivo-2 GIVING nombre-de-datos-1             ⎪
ON      ⎨  INPUT                                  [nombre-de-datos-2] ⎬
        ⎪  OUTPUT                                                   ⎪
        ⎩  I-O                                                      ⎭
```

Nombre-de-archivo, se refiere al archivo para el que se ejecutarán los procedimientos de manejo de errores especificados por el programador, cada vez que se produce un error por la ejecución de una declaración READ, WRITE o REWRITE para ese archivo. En el ejemplo se ejecutan los procedimientos descritos desde la línea 002120 a 002150 cada vez que no se pueda ejecutar la declaración WRITE (línea 002230) por no existir espacio disponible en una pista para grabar un registro.

Nombre-de-datos (en el ejemplo, CAMPO-ERRORES línea 002030) es un campo decimal de ocho octetos que debe describirse en la WORKING-STORAGE SECTION. Cada uno de los octetos sirve para indicar al Sistema diferentes condiciones de error. Para no hacer exhaustiva su descripción, sólo se indica que el tercer octeto lo emplea el Sistema para conocer la condición de error que se produce cuando el programador intenta grabar un registro en una pista que se encuentra llena, según se especificó en pág. 316 .

Podía haberse creado el archivo ARCHIDISC sin utilizar la sentencia USE. Sin embargo, tiene el inconveniente de que cada vez que hubiese que grabar un registro en una pista que estuviera llena habría que suspender el trabajo y, comenzarlo de nuevo utilizando en el algoritmo un número primo de valor superior. Así, se repetiría la operación probando con nuevos números primos hasta que pudieran grabarse todos los registros. Esto llevaría consigo el que se necesitaría más tiempo para la creación del archivo.

Otro inconveniente, que es consecuencia del anterior, es tener que reservar mayor número de pistas por ser mayor el número primo utilizado.

7.1.2.2.-ACTUALIZACION del ARCHIVO (Dirección Relativa de Pista)

El siguiente ejemplo muestra la forma de actualizar el archivo creado por el programa DINDIREC (pág. 314).

Los registros que contienen las modificaciones a introducir en el archivo ARCHIDIS, se encuentran grabados en una cinta magnética.

```
001010  ID DIVISION.
001020    PROGRAM-ID. ACTUINDI.
001030    REMARKS. ACTUALIZA UN ARCHIVO DIRECTO ORGANIZADO CON DIRECCIONA-
001040             MIENTO INDIRECTO A PARTIR DE UN ARCHIVO EN CINTA.
001050  ENVIRONMENT DIVISION.
001060    INPUT-OUTPUT SECTION.
001070      FILE-CONTROL.
001080          SELECT ARCHIDIS ASSIGN SYS005-DA-3340-D
001090                 ACCESS IS RANDOM
001100                 ACTUAL KEY IS ACTUALY.
001110          SELECT ARCHICIN ASSIGN SYS006-UT-2400-S.
001120  DATA DIVISION.
001130    FILE SECTION.
001140      FD  ARCHIDIS RECORDING F  LABEL RECORD STANDARD.
001150      01  DISCO.
001160          02  CLAVED      PIC X(6).
001170          02  FILLER      PIC X(12).
001180          02  CALLE       PIC X(25).
001190          02  FILLER      PIC X(10).
001200          02  DISTRITO    PIC 99.
001210          02  FILLER      PIC X(111).
002010      FD ARCHICIN  RECORDING F  LABEL RECORD OMITTED.
002020      01 CINTA.
002030         02 CLAVEC      PIC X(6).
002040         02 CALLEC      PIC X(25).
002050         02 DISTRIPOS   PIC 99.
002060    WORKING-STORAGE SECTION.
002070      77 COCIENTE       PIC 9(5).
002080      77 CLAVE          PIC S9(6) COMP.
002090      01 ACTUALY.
002100         02 PISTA       PIC S9(8) COMP.
002110         02 CLAVEREG    PIC X(6).
002120  PROCEDURE DIVISION.
002130  ABRE. OPEN INPUT ARCHICIN  I-O ARCHIDIS.
002140  LEER. READ ARCHICIN AT END  GO TO FIN.
002150        MOVE CLAVEC  TO CLAVE  CLAVEREG
002160        DIVIDE CLAVE  BY 89  GIVING COCIENTE REMAINDER PISTA.
002170  LEED. READ ARCHIDIS  INVALID KEY GO TO SINONIMO.
002180        MOVE CALLEC  TO CALLE  MOVE DISTRIPOS TO DISTRITO
002190        WRITE DISCO  INVALID KEY DISPLAY CLAVED 'CLAVE MODIFICADA'
002200        GO TO FIN.  GO TO LEER.
002210  SINONIMO.
002220        IF PISTA < 96 ADD 1 TO PISTA GO TO LEED  ELSE DISPLAY
002230        CLAVEREG 'NO EXISTE LA CLAVE EN EL ARCHIVO'  GO TO LEER.
002240  FIN.  CLOSE ARCHIDIS  ARCHICIN STOP  RUN.
```

La actualización no presenta prácticamente dificultad, realizándose de la forma siguiente:

Lee un registro de la cinta (línea 002140) e introduce en el campo ACTUALY por medio de las declaraciones de las líneas 002150 y 002160 el identificador de pista (PISTA) y el identificador de registro (CLAVEREG), con lo cual queda preparado el Sistema para realizar la búsqueda de un registro en la dirección obtenida. La búsqueda se realiza al ejecutar la declaración READ (línea 002170); si localiza al registro lo introduce en la memoria, lo actualiza con el

nuevo contenido de los campos CALLEC y DISTRIPOS y lo vuelve a escribir (línea 002190) en el mismo lugar de donde lo recuperó. Cada vez que se ejecuta la declaración READ sólo busca el registro en la pista obtenida por el algoritmo; si no lo encuentra, se cumple la condición INVALID KEY, con lo que pasa el control al procedimiento denominado SINONIMO (línea 002210) por el que incrementa en una unidad el número de pista para efectuar la búsqueda en la pista siguiente.

Esta operación se repite hasta la localización del registro o hasta que el valor del campo PISTA no sea menor que 96, en cuyo caso escribe por impresora (línea 002230) la clave que se está buscando junto con el mensaje 'NO EXISTE LA CLAVE . . . . .'; volviendo a leer un nuevo registro del archivo en cinta, repitiéndose con él todo el ciclo descrito.

7.1.2.3.– CREACION DEL ARCHIVO (Dirección Real de Pista)

Hay casos en la creación de archivos en que es difícil encontrar el algoritmo adecuado que distribuya uniformemente los registros a lo largo del espacio de almacenamiento reservado, dando lugar a situaciones en las que:

- Se encuentra llena la última pista reservada para el archivo, cuando un nuevo registro debía grabarse en ella, o

- Se distribuyen los registros con mayor frecuencia en determinadas pistas sucesivas pudiendo dar lugar a excesivos sinónimos para dichas pistas.

- En el primer caso, hay que comenzar de nuevo la creación del archivo, asignándo un número de pistas mayor. Sin embargo, ello podía no ser posible si no se dispone de espacio en el discpack.

- En el segundo caso, si bien no sería necesario comenzar de nuevo la creación del archivo, tiene el inconveniente de que por grabarse los sinónimos en pistas distintas a las que les corresponde, necesitaría de un tiempo de proceso superior para su creación; así, por ejemplo, si se encuentran llenas las pistas desde la 200 a la 320 y se produce un sinónimo que debía grabarse en la pista 201, habrá que intentar su grabación desde dicha pista hasta la 321 a fin de encontrar el espacio correspondiente.

Estos y otros casos pueden encontrar solución si se reservan áreas de overflow donde grabar los registros sinónimos.

Las áreas de overflow puede el programador reservarlas:

- En la última o últimas pistas de cada cilindro, en las que se grabarían los sinónimos excedentes de cada pista de ese cilindro, o

— En uno o varios cilindros situados a continuación de los reservados para el archivo. Los sinónimos que no puedan grabarse en la pista que les corresponda de un determinado cilindro, se intentará en la pista siguiente, así hasta llegar a la última pista de ese cilindro y caso de que no pueda grabarse en ella se hará en el cilindro de overflow.

En los siguientes ejemplos se ha optado por emplear este último tipo de área de overflow, lo cual exige al programador tener que proporcionar al Sistema el número de cilindro y de pista, es decir, la dirección real donde se ha de grabar cada registro.

La dirección real se obtendrá aplicando a la clave de cada registro el algoritmo y rutina siguiente:

— Se calcula la dirección relativa de pista (como hasta ahora se ha venido haciendo).

— Se suma a dicha dirección el número de la pista a partir del cual el programador decidió comenzar la grabación del archivo, con lo cual se obtiene el número absoluto o real de la pista.

— Dicho número se divide por el número de pistas que contiene cada cilindro, con lo que el cociente proporcionará el número del cilindro y el resto será el número de la pista dentro de ese cilindro donde se debería llevar a cabo la grabación del registro.

El siguiente programa muestra la creación de un archivo en disco a partir de los registros contenidos en un archivo en cinta empleando direcciones reales de pista.

Se reserva un cilindro para área de overflow.

```
000010  ID DIVISION.
000020  PROGRAM-ID. DIREREAL.
000030  REMARKS. CREACION ARCHIVO POR DIRECCIONAMIENTO INDIRECTO CON DI-
000040                   RECCION REAL DE   PISTA A PARTIR DE ARCHIVO EN CINTA.
000050  ENVIRONMENT DIVISION.
000060  INPUT-OUTPUT SECTION.
000070  FILE-CONTROL.
000080      SELECT ARCHIDIS  ASSIGN SYS005-DA-3340-A
000090                   ACCESS RANDOM
000100                   ACTUAL KEY IS ACTUALY.
000110      SELECT ARCHICIN  ASSIGN SYS006-UT-2400-S.
000120  DATA DIVISION.
000130  FILE SECTION.
000140      FD ARCHIDIS  RECORDING F  LABEL RECORD STANDARD.
000150      01 DISCO.
000160          02 CLAVED      PIC X(6).
000170          02 DATOSD      PIC X(160).
000180      FD ARCHICIN  RECORDING F  LABEL RECORD OMITTED.
000190      01 CINTA.
000200          02 CLAVEC      PIC X(6).
000210          02 DATOSC      PIC X(74).
000220  WORKING-STORAGE SECTION.
000230      77 CLAVE           PIC S9(6) COMP.
000240      77 COCIENTE        PIC 9(5).
000250      77 RESTO           PIC 9(4).
```

```
002010   01 DIRECCION.
002020        02 EXTENSION      PIC S999 COMP VALUE ZERO.
002030        02 CELDA          PIC S9 COMP   VALUE ZERO.
002040        02 CILINDRO       PIC S999 COMP.
002050        02 PISTA          PIC S999 COMP.
002060        02 REGISTRO       PIC X  VALUE LOW-VALUE.
002070        02 IDENTIF        PIC X(6).
002080   01 DIREC REDEFINES DIRECCION.
002090        02 FILLER         PIC X.
002100        02 ACTUALY        PIC X(14).
002110   01 CAMPO-ERRORES.
002120        02 FILLER         PIC 99 VALUE ZERO.
002130        02 OCT-ERROR      PIC 9  VALUE ZERO.
002140        02 FILLER         PIC 9(5) VALUE ZERO.
002150 PROCEDURE DIVISION.
002160   DECLARATIVES.
002170     ERRORES SECTION.
002180        USE AFTER STANDARD ERROR PROCEDURE ON ARCHIDIS
002190            GIVING CAMPO-ERRORES.
002200     RUTINA-ERROR.
002210        IF OCT-ERROR = 1 GO TO SINONIMO
002220            ELSE DISPLAY CLAVEC 'OTRO TIPO ERROR' GO TO FIN.
003010   SINONIMO.
003020        IF CILINDRO = 175 AND PISTA = 11 DISPLAY IDENTIF ' CILINDRO
003030            'DE OVERFLOW COMPLETO' GO TO FIN.
003040        IF CILINDRO = 175 ADD 1 TO PISTA GO TO GRABAR.
003050        IF PISTA = 11 GO TO CILINDRO-COMPLETO.
003060        ADD 1 TO PISTA
003070        GO TO GRABAR.
003080   CILINDRO-COMPLETO.
003090        MOVE 175 TO CILINDRO
003100        MOVE ZERO TO PISTA
003110        GO TO GRABAR.
003120   END DECLARATIVES.
003130   ABRIR. OPEN INPUT ARCHICIN OUTPUT ARCHIDIS.
003140   LEER.
003150        READ ARCHICIN AT END GO TO FIN.
003160        MOVE CLAVEC TO CLAVE
003170        DIVIDE CLAVE BY 89 GIVING COCIENTE REMAINDER RESTO
003180        ADD 2004 TO RESTO
003190        DIVIDE 12 INTO RESTO GIVING CILINDRO REMAINDER PISTA
003200        MOVE CLAVEC TO IDENTIF
003210        MOVE CINTA TO DISCO.
003220   GRABAR.
003230        WRITE DISCO INVALID KEY DISPLAY CLAVEC 'CLAVE INVALIDA'
003240        GO TO FIN. GO TO LEER.
003250   FIN. CLOSE ARCHIDIS ARCHICIN STOP RUN.
```

Estudiando el programa se observa que gran parte de las inscripciones son comunes con la creación del archivo en el programa DINDIREC, sin embargo, existen las diferencias siguientes:

En la cláusula SELECT debe especificarse: –A en vez de –D.

El artificio empleado para proporcionar al Sistema el campo ACTUALY de la cláusula ACTUAL KEY, exige la estructura mostrada de la línea 002010 a la 002100. En ella, a los campos denominados: CILINDRO (línea 002040), PISTA (línea 002050) e IDENTIF (línea 002070) se transferirán respectivamente el número de cilindro, número de pista y la clave del registro, que son los datos que proporcionan la dirección real de la pista donde se ha de grabar; el resto de los campos deben especificarse, por exigirlo el Sistema, como se hace en el programa; no describiendo su significado y empleo por ser innecesario.

La redefinición (línea 002080) es necesaria para eliminar un octeto del identificador de pista, constituído por los campos especificados de la línea 002020 a 002060, ya que el Sistema exige que la longitud total que ocupen en memoria los campos componentes de dicho Identificador sea de ocho posiciones, y tal como están descritos ocupan nueve (dos para EXTENSION, CELDA, CILINDRO y PISTA por ser su USAGE COMP y un octeto para el campo REGISTRO por ser su USAGE DISPLAY). Al eliminar el primer octeto, que no contiene información, queda el campo ACTUALY con la información necesaria para poder grabar el registro, es decir, en los ocho primeros octetos el identificador de pista y en los seis restantes el identificador o clave del registro.

El algoritmo empleado se especifica desde la línea 003170 a la 003190. La rutina con la que se obtiene la dirección real donde se grabará el registro, se describe en las líneas 003180 y 003190, en ella, el número 2004 es el de la pista a partir de la cual el programador decidió comenzar la grabación del archivo, y el número 12 es el de las pistas que contiene cada cilindro del disco del modelo elegido.

Se ha reservado para área de overflow el cilindro 175, ya que si el comienzo de la grabación del archivo debe hacerse a partir de la pista 2004 ó, lo que es lo mismo, a partir del cilindro 2004 : 12 = 167, al asignar al archivo 96 pistas equivalentes a 8 cilindros, quiere decir que el algoritmo proporcionará direcciones desde el cilindro 167 al 174, siendo al cilindro 175 donde han de enviarse los sinónimos excedentarios de cada uno de los ocho cilindros.

Los procedimientos especificados dentro de la declarativa USE indican el tratamiento de los sinónimos y obligan al Sistema a realizar las siguientes acciones:

- Siempre que se produzca un sinónimo que no pueda grabarse en el cilindro y pista direccionada, se producirá un error standard que pone automáticamente en el campo OCT-ERROR el valor 1, pasando el control del programa a RUTINA-ERROR, donde se intentará su grabación en las pistas sucesivas de ese cilindro (ADD 1 TO PISTA, línea 003060); caso de que llegue al final del cilindro sin poderlo grabar, deberá hacerlo en el área de overflow situada en el cilindro 175 y en la pista donde encuentre espacio, intentando su grabación a partir de la primera de dicho cilindro (líneas 003090 a la 003110). Si después de intentar su grabación en todas sus pistas, no encontrara espacio, sería debido a que el área de overflow estaría llena, en cuyo caso el trabajo terminaría anormalmente por quedar registros sin grabar. Ello supondría tener que comenzar de nuevo la creación del archivo, para lo cual, previamente, el programador deberá realizar alguna o algunas de las siguientes acciones:

  - Reservar más cilindros para el área de overflow
  - Reservar más pistas para el archivo, o
  - Utilizar un nuevo algoritmo, etc.

Es normal tener que efectuar varias pruebas con el programa hasta llegar a la creación definitiva del archivo, ya que es difícil para el programador preveer todas las posibles causas que pueden originar un final anormal en la creación del archivo.

7.1.2.4.- ACTUALIZACION del ARCHIVO (Dirección Real de Pista)

Una vez creado el archivo, ya puede ser utilizado para realizar cualquier trabajo que requiera la recuperación, incorporación, actualización, etc., de registros del archivo. En estos tipos de trabajo se hace necesaria la previa localización de cada registro, para lo cual se debe especificar en el programa el mismo algoritmo que el especificado durante la creación del archivo que proporcione el número del cilindro y pista donde se debe efectuar la búsqueda, así como indicar las acciones a realizar para el caso de que el registro a localizar sea un sinónimo y no se encuentre grabado en dicha pista.

Una de las acciones a realizar para localizar dichos sinónimos consiste en ir incrementando en una unidad el número de la pista direccionada por el algoritmo, para así continuar la búsqueda del registro en la pista siguiente hasta el final del cilindro; caso de no localizarlo en dicho cilindro, se indicarán en el programa las acciones a llevar a cabo para continuar la búsqueda en el cilindro de overflow. No obstante, existe la posibilidad de poder simplificar este tipo de acciones utilizando la cláusula APPLY EXTENDED–SEARCH, cuya función es extender automáticamente la búsqueda de un registro al cilindro completo, cuando no ha podido localizarse en la pista proporcionada por el algoritmo.

Su formato es:

APPLY EXTENDED-SEARCH ON nombre–de–archivo–1
[nombre–de–archivo–2] . . .

- Esta cláusula debe especificarse dentro del párrafo I–O–CONTROL.

- Nombre–de–archivo–1 es el nombre del archivo en el que ha de realizarse la búsqueda.

En un programa donde no se especifique dicha cláusula, cada declaración READ obliga a realizar la búsqueda del registro en el cilindro y pista obtenidos por el algoritmo; caso de no localizarlo a lo largo de dicha pista se cumple la condición INVALID KEY que obligará a ejecutar las acciones que se especifiquen en la declaración imperativa que sigue a INVALID.

Por el contrario, en un programa donde se especifique la cláusula APPLY EXTENDED–SEARCH, si un registro no se localiza en el cilindro y pista direccionada, continúa la búsqueda en las pistas siguientes de dicho cilindro; cumpliéndose la condición INVALID KEY de la declaración READ cuando no pueda localizarse el registro en ese cilindro, ejecutándose la declaración imperativa que sigue a INVALID (ver ejercicio número dos de este capítulo).

8.- EJERCICIOS

1.- Escribir un programa que sirva para mantener actualizado el archivo indexado creado con el programa CREAINDX (pág. 292),procesando al final de cada día todas las transacciones que se hayan producido.

El contenido de cada transacción ha sido perforado en una ficha, y el orden en que deben procesarse las transacciones es aquel en el que se han producido.

El contenido y formato de cada ficha es el siguiente:

COLUMNAS	CONTENIDO
De la 1 a la 5	CODIGO o CLAVE
De la 6 a la 79	DATOS VARIOS
En la 80	Cifra 1, 2 ó 3

Las acciones a reflejar en el programa son las siguientes:

- Si en la columna 80 de la ficha hay perforada la cifra 1, quiere decir que es un registro que hay que añadir al archivo indexado; dicha adición consistirá en grabar un nuevo registro con el contenido de la ficha.

- Si en la columna 80 hay perforada la cifra 2, significa que es un registro a dar de baja. La forma de dar de baja un registro, consistirá en sustituir todos los datos del registro cuya clave sea igual a la de la ficha, por guiones (-).

- Si hay perforada la cifra 3 en la columna 80, significa que hay que actualizar un registro del archivo indexado cuya clave sea igual a la de la ficha. La actualización consistirá en sustituir los datos en dicho registro por los nuevos datos contenidos en la ficha.

Se han verificado únicamente las columnas 1 a 5 y la 80 de cada ficha,a fin de comprobar si su contenido era numérico, por lo que el programador debe especificar en el programa las acciones a realizar por el ordenador ante la posibilidad de que:

- Aunque una clave sea numérica pueda ser inválida,o que

- En la columna 80 exista una cifra distinta de 1, 2 ó 3.

Si existe una clave inválida no se realizará la adición, baja o actualización de un registro debiendo imprimir un mensaje por la impresora (DISPLAY) en el que se refleje la clave inválida junto con el motivo que lo origina. La existencia de una clave inválida no debe ser causa para dar por finalizado el programa, sino que deberá continuar el proceso hasta finalizar las transacciones.

Si existe error en la columna 80 deberá imprimirse un mensaje de iguales características que en el caso anterior.

2.– Para actualizar el archivo directo creado por el programa DIREREAL (pág. 320) es necesario escribir un programa que procese todas las transacciones que diariamente se producen.

El contenido de cada transaccion constituye un registro grabado en una cinta cuyo formato es el siguiente:

POSICIONES	CONTENIDO
De la 1 a la 6	CLAVE
De la 7 a la 166	DATOS

El conjunto de estos registros constituyen el archivo en cinta cuyas características son:

– Existen grabadas etiquetas standard.

– Los registros están agrupados en bloques de 10, en el mismo orden en que se produjeron las transacciones.

El proceso consistirá en leer un registro del archivo en cinta, localizar en el archivo directo un registro con clave igual, y sustituir los datos que contenga este registro por los datos del registro de la cinta.

Por no haberse verificado la validez de las claves de las transacciones, puede suceder que alguna de ellas sea inválida por no encontrarse grabada dentro del archivo directo; en este caso el programador especificará en el programa una declaración DISPLAY que permita enviar por la consola un mensaje que escriba la clave inválida junto con la causa que lo motiva.

En el momento de producirse la primera clave inválida debe darse por finalizado el programa.

CAPITULO VIII

CARACTERISTICAS ESPECIALES

1.- CARACTERISTICA de CLASIFICACION (SORT)

Al hacer la descripción del programa NOMINA, se dijo que las fichas que contenían los datos de cada operario estaban ordenadas en secuencia ascendente por el contenido del campo CODIGO. Esta ordenación pudo hacerse, bien manualmente, o utilizando una máquina clasificadora; en cualquiera de los dos casos, resulta un proceso laborioso y expuesto a errores, razón por la que hay que recurrir a otros medios más efectivos y no originarios de error, aprovechando las posiblidades que brinda el ordenador para poder realizar cualquier tipo de clasificación.

Actualmente, las dos técnicas utilizadas para la realización de dicho tipo de trabajo son:

- Empleo de fichas de control, o
- Escritura de un programa en COBOL.

La elección de la técnica apropiada es responsabilidad del programador. La más sencilla es la primera, ya que cualquier clasificación se puede llevar a cabo empleando, simplemente, cinco o seis fichas de control, en las que hay que cumplimentar una serie de parámetros que proporcionen al Sistema la información necesaria para realizar dicha clasificación (27). Sin embargo, su empleo tiene limitaciones, ya que no es posible su uso cuando previa a la clasificación del archivo fuera necesario realizar un tratamiento de los registros (seleccionar, modificar datos, etc.).

Las primeras versiones del COBOL no permitían la realización de la clasificación de un archivo por medio de un programa, ya que carecía de las declaraciones que lo hicieran posible; sin embargo, las modernas versiones, con la aparición de la declaración SORT, incorporaron esta posibilidad. Su escritura dentro de un programa permite el que pueda llevarse a cabo, no sólo cualquier clasificación sino incluir procedimientos que realicen el tratamiento de los registros tanto antes como después de la clasificación.

---

(27) En este texto se omite la descripción de la información que debe incluirse en cada ficha de control por existir grandes diferencias según el Sistema Operativo empleado.

Es importante el conocimiento de ambas técnicas, pues su empleo se hace necesario en un alto porcentaje de los trabajos que se realizan en todo Centro de Proceso de Datos. Es muy frecuente, por ejemplo, un archivo que en su creación se ordenaron sus registros siguiendo la secuencia ascendente o descendente de una determinada clave, tenga que volver a clasificarse por el contenido de otra clave para poder realizar otro tipo de trabajo.

En este capítulo se presentan tres sencillos programas COBOL, que van desde la realización de una simple clasificación, hasta la ejecución de procedimientos que tratan los archivos antes y después de la clasificación.

1.1.– Declaración SORT

El primer programa realiza el trabajo siguiente: clasifica alfabéticamente el archivo de los operarios utilizado en la confección de la nómina (pág. 23), y lo graba en un nuevo archivo en un disco.

```
001010 ID DIVISION.
001020 PROGRAM-ID. SORT1.
001030 REMARKS. PROGRAMA EJEMPLO DE UN SORT PARA CREAR UN ARCHIVO
001040                  EN DISCO ORDENADO ALFABETICAMENTE.
001050 ENVIRONMENT DIVISION.
001060 INPUT-OUTPUT SECTION.
001070 FILE-CONTROL.
001080     SELECT ARCHIOPE   ASSIGN SYS004-UR-2540R-S.
001090     SELECT ARCHIDISC  ASSIGN SYS005-DA-3340-S.
001100     SELECT ARCHISORT  ASSIGN SYS001-DA-3340-S-SORTWK1.
001110 DATA DIVISION.
001120 FILE SECTION.
001130 FD  ARCHIOPE    LABEL RECORD OMITTED.
001140 01  FICHOPE     PIC X(80).
001150 FD  ARCHIDISC   LABEL RECORD STANDARD.
001160 01  DATOSD      PIC X(120).
001170 SD  ARCHISORT   LABEL RECORD STANDARD.
001180 01  CLASIF.
001190     02 FILLER   PIC X(5).
001200     02 APENOM   PIC X(35).
001210     02 FILLER   PIC X(40).
002010 PROCEDURE DIVISION.
002020 CLASIFICA.
002030     SORT ARCHISORT
002040          ASCENDING KEY APENOM
002050          USING ARCHIOPE
002060          GIVING ARCHIDISC.
002070     STOP RUN.
```

Para llegar a la solución del problema, el Sistema debe realizar en el orden en que se citan, las acciones siguientes:

- Leer todas las fichas que componen el archivo de operarios.
- Guardar su contenido en un archivo de trabajo.
- Clasificar dicho archivo en orden ascendente de apellidos y nombre.
- Grabar el archivo clasificado en otro archivo en disco.

La realización de dichas acciones las lleva a cabo la declaración SORT, para lo cual el programador debe proporcionar al Sistema, en una o varias Unidades de disco o cinta, el almacenamiento suficiente para llevar a cabo la ejecución del trabajo de clasificación. Dicho almacenamiento se denomina ARCHIVO de TRABAJO y del cual es preciso hacer su descripción dentro del programa como se realiza con cualquier otro archivo.

A continuación se describe el contenido de cada una de las DIVISIONES del programa.

En la ENVIRONMENT DIVISION y dentro de la cláusula SELECT se especifican los archivos:

- ARCHIOPE, archivo de entrada en fichas de donde se tomarán los registros a clasificar.
- ARCHISORT, archivo de trabajo reservado en un disco que utilizará el Sistema para realizar la clasificación (el ordenador en que se ha ejecutado el programa obliga a especificar la unidad simbólica SYS001 y el nombre SORTWK1).
- ARCHIDISC, archivo de salida en disco donde se grabarán los registros ya clasificados.

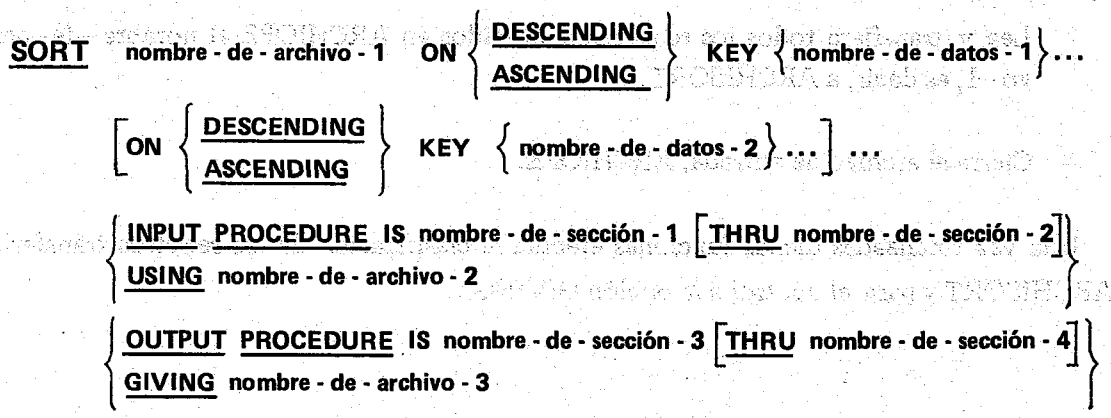
En la DATA DIVISION se hace la descripción de los archivos de entrada/salida y trabajo y la de cada uno de los registros asociados.

- El archivo de trabajo debe comenzar su descripción con: SD (SORT DESCRIPTION) (línea 001170) seguida de las demás cláusulas, y a continuación la descripción de los campos que van a formar parte del registro de clasificación, especificando con nombre-de-datos (no FILLER) a aquellos campos por los que se va a hacer la clasificación.

En este ejemplo la estructura adoptada por el registro de clasificación es la misma que la de los registros del archivo de entrada, ya que los campos que componen ambos registros son los mismos y conservan la misma posición (esto no es obligatorio, ya que el número y distribución de los campos del registro de clasificación puede ser diferente a los del registro del archivo de entrada, según se especifica en el ejemplo tercero, pág. 333).

En la PROCEDURE DIVISION se especifica la declaración SORT, cuya función es proporcionar al Sistema la información necesaria para controlar las operaciones de clasificación.

Su formato es:



La existencia en este formato de numerosas llaves quiere decir, que también serán numerosas las posibilidades por parte del programador de elección de la opción más adecuada para la realización del trabajo. Esta diversidad obliga, para no ser excesivamente teóricos, a fraccionar su descripción en éste y siguientes programas.

1.1.1.– Opciones ASCENDING y DESCENDING

Las opciones ASCENDING y DESCENDING sirven para indicar al Sistema que los registros que se van a clasificar lo van a hacer en secuencia ascendente o descendente por el contenido de uno o varios campos de datos.

- Nombre-de-datos-1, nombre-de-datos-2, son los nombres de los campos de datos por cuyo contenido se va a hacer la clasificación (APENOM).

- Hay que especificar, como mínimo, una opción: ASCENDING o DESCENDING. La clasificación, con cualquiera de las dos opciones, se realiza de acuerdo con el orden de valores establecido en la tabla de clasificación (pág. 184).

En el ejemplo, por haber elegido el programador la opción ASCENDING, quiere decir que la clasificación se hará en secuencia ascendente desde el valor inferior al valor superior del campo APENOM, es decir, orden alfabético.

1.1.2.– Opciones USING y GIVING

De las dos siguientes posibles elecciones que debe realizar en la declaración SORT, el programador ha optado por las opciones USING y GIVING.

Con la opción USING el compilador realiza automáticamente, sin que el programador tenga que especificarlas, las siguientes funciones:

- Abre el archivo especificado a continuación de USING: nombre–de–archivo–2, es decir, ARCHIOPE.

- Lee y transfiere todos los registros contenidos en ARCHIOPE al nombre–de–archivo–1, es decir, a ARCHISORT, y

- Cierra el archivo de entrada, ARCHIOPE.

Una vez finalizadas dichas funciones efectúa la clasificación de los registros transferidos a ARCHISORT y pasa el control a la opción GIVING.

Con la opción GIVING, el compilador realiza automáticamente las funciones siguientes:

- Abre nombre–de–archivo–3, es decir, el archivo de salida: ARCHIDISC.
- Transfiere y graba en ARCHIDISC los registros clasificados contenidos en el archivo de trabajo ARCHISORT, y
- Cierra el archivo de salida ARCHIDISC.

Realizadas estas acciones finaliza la ejecución del programa con la declaración STOP RUN.

Los archivos que figuren a continuación de USING y GIVING deben ser archivos secuenciales standard. Por tanto, ARCHIOPE y ARCHIDISC deben describirse con organización secuencial: –S (línea 001080 y 001090).

1.1.3.– Opción INPUT PROCEDURE

El siguiente programa ejemplo, utiliza el mismo archivo de entrada del programa SORT1, para crear un archivo en disco en el que se graben ordenados alfabéticamente aquellos operarios que reunan las siguientes condiciones:

- Con hijos y
- Sueldo no inferior a 20.000 pts.

```
001010  ID DIVISION.
001020    PROGRAM-ID.  SORT2.
001030*   PROGRAMA PARA CREAR UN ARCHIVO ORDENADO ALFABETICAMENTE DE
001040*   LOS OPERARIOS CON HIJOS Y SUELDO IGUAL O MAYOR DE 20000 PTS.
001060  ENVIRONMENT DIVISION.
001070    INPUT-OUTPUT SECTION.
001080      FILE-CONTROL.
001090          SELECT  ARCHIOPE     ASSIGN  SYS004-UR-2540R-S.
001100          SELECT  ARCHISORT    ASSIGN  SYS001-DA-3340-S-SORTWK1.
001110          SELECT  ARCHIDISC    ASSIGN  SYS005-DA-3340-S.
001120  DATA DIVISION.
001130    FILE SECTION.
001140      FD  ARCHIOPE       RECORDING F     LABEL RECORD  OMITTED.
001150      01  FICHOPE.
001160          02  FILLER                 PIC X(50).
001170          02  NUMHIJOS               PIC 99.
001180          02  FILLER                 PIC X(6).
001190          02  SUELDO                 PIC 9(5).
001200          02  FILLER                 PIC X(17).
002010      FD  ARCHIDISC      RECORDING F     LABEL RECORD  OMITTED.
002020      01  REGISD                      PIC X(80).
002030      SD  ARCHISORT      RECORDING  F
002035                         LABEL  RECORD  STANDARD
002036                         DATA   RECORD  CLASIF.
002040      01  CLASIF.
002050          02  FILLER                 PIC 9(5).
002060          02  APENOM                 PIC X(35).
002070          02  FILLER                 PIC X(40).
```

```
002080  PROCEDURE DIVISION.
002090    ORDENACION.
002100              SORT   ARCHISORT
002110                     ASCENDING KEY     APENOM
002120                     INPUT PROCEDURE   ENTRADA
002130                     GIVING   ARCHIDISC.
002135              STOP   RUN.
002140    ENTRADA SECTION.
002145      ABRIR.
002150              OPEN INPUT   ARCHIOPE.
002155      LEER.
002160              READ   ARCHIOPE   AT END   CLOSE ARCHIOPE    GO TO SALIDA.
002170                IF   NUMHIJOS = 0   OR   SUELDO < 20000    GO TO LEER
002175                ELSE    NEXT SENTENCE.
002180              RELEASE   CLASIF   FROM   FICHOPE.
002185              GO TO   LEER.
002190      SALIDA.         EXIT.
```

La diferencia principal con el programa SORT1 es el empleo de la opción INPUT PROCEDURE en vez de la opción USING.

La opción INPUT PROCEDURE indica al compilador que con cada registro del archivo de entrada ha de llevarse a cabo un procedimiento (proceso) previo a la clasificación, por ejemplo, seleccionar, modificar, etc., registros.

Nombre-de-sección-1 que figura en el formato de la declaración SORT a continuación de INPUT PROCEDURE es el nombre de dicho procedimiento. En el ejemplo, se denomina ENTRADA, comprende de la línea 002140 a la 002190, y en él se reflejan las acciones siguientes:

- Abre el archivo de entrada (línea 002150).
- Lee un registro de dicho archivo (línea 002160).
- Selecciona los registros que cumplan las condiciones del enunciado (línea 002170).
- Aquellos registros que cumplan dichas condiciones, los transfiere al archivo de trabajo ARCHISORT mediante la declaración RELEASE (línea 002180).

Finalizada la lectura y selección de todos los registros del archivo de entrada, el Sistema efectúa la clasificación de dichos registros, y pasa el control a la cláusula siguiente a INPUT PROCEDURE, es decir, a GIVING ARCHIDISC, ya que el funcionamiento de la opción INPUT es similar al de la declaración PERFORM.

La opción GIVING realiza las acciones descritas con el ejemplo SORT1, finalizando el programa al ejecutar la declaración STOP RUN.

1.1.3.1.- Declaración RELEASE

FUNCION: Transferir los registros al archivo de trabajo para ser clasificados.

FORMATO:

RELEASE nombre - de - registro - de - clasificación [FROM identificador]

Si se utiliza la opción **FROM identificador**, equivale a una declaración MOVE seguida de la declaración RELEASE. Es decir, la declaración RELEASE CLASIF FROM FICHOPE (línea 002180) equivale a:

```
002180     MOVE FICHOPE TO CLASIF
002181     RELEASE CLASIF
```

Prácticamente, la función de la declaración RELEASE es la misma que la de una declaracion WRITE, ya que realiza la función de grabar en el archivo de trabajo los registros que van a ser clasificados.

1.1.4.- Opción OUTPUT PROCEDURE

Con el programa SORT3 se quiere obtener un listado de todos aquellos operarios de la empresa que tengan hijos y su sueldo no sea inferior a 20.000 pts., siguiendo el criterio de ordenación siguiente:

- Clasificado alfabéticamente por provincias.
- Dentro de cada provincia figurarán los operarios nacidos en ella ordenados alfabéticamente por apellidos y nombre.
- Si en alguna de las provincias existieran operarios con igual nombre y apellidos se ordenarán en forma descendente por el CODIGO que tiene asignado cada operario.

El archivo de donde se tomarán los datos es el archivo de operarios utilizado en la confección de la nómina (pág. 23).

En el listado se deberá escribir el nombre de la provincia, y no el código de la misma como figura en el citado archivo.

Se perforarán diez fichas conteniendo cada una el nombre de cinco provincias (dieciseis columnas por provincia) y se introducirá dicho contenido en la memoria del ordenador mediante una declaración ACCEPT.

```
001010  ID DIVISION.
001020  PROGRAM-ID. SORT3.
001030  AUTHOR. C. DE LA FUENTE.
001040  REMARKS. PROGRAMA EJEMPLO DE EMPLEO DE UN SORT.
001050 ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070  FILE-CONTROL.
001080  SELECT ARCHIOPE ASSIGN SYS004-UR-2540R-S.
001090  SELECT ARCHIMPRE ASSIGN SYS005-UR-1403-S.
001100  SELECT ARCHISORT ASSIGN SYS001-DA-3340-S-SORTWK1.
001110 DATA DIVISION.
001120  FILE SECTION.
001130  FD ARCHIOPE LABEL RECORD OMITTED.
001140  01 FICHAOPE.
001150  02 CODIGO PIC X(5).
001160  02 APENOM PIC X(35).
001170  02 FILLER PIC 9(6).
001180  02 PROVINACI PIC 99.
```

```
001190     02 FILLER                PIC XX.
001200     02 NUMHIJOS              PIC 99.
001210     02 FILLER                PIC X(6).
001220     02 SUELDO                PIC 9(5).
001230     02 FILLER                PIC X(17).
002010   SD   ARCHISORT             RECORDING F
002015                              LABEL RECORD STANDARD.
002020   01  CLASIF.
002030       02 CODIGO              PIC X(5).
002040       02 APENOM              PIC X(35).
002050       02 PROVINACI           PIC 99.
002060       02 NUMHIJOS            PIC 99.
002070       02 SUELDO              PIC 9(5).
002080   FD   ARCHIMPRE             RECORDING F
002085                              LABEL RECORD OMITTED.
002090   01  IMPRESO.
002100       02 FILLER              PIC X.
002110       02 CODIGO              PIC X(5).
002120       02 FILLER              PIC X(4).
002130       02 APENOM              PIC X(35).
002140       02 FILLER              PIC X(8).
002150       02 NUMHIJOS            PIC 99.
002160       02 FILLER              PIC X(8).
002170       02 SUELDO              PIC 9(5).
002180       02 FILLER              PIC X(8).
002190       02 PROVINCIA           PIC X(16).
002200       02 FILLER              PIC X(41).
002210 WORKING-STORAGE SECTION.
002220   01 TABLAPROVIN.
002230       02 PROVIN    OCCURS 50      PIC X(16).
003010 PROCEDURE DIVISION.
003020 CLASIFICAR.
003030     SORT  ARCHISORT
003040           ASCENDING KEY  PROVINACI OF CLASIF  APENOM OF CLASIF
003050           DESCENDING KEY  CODIGO   OF CLASIF
003060           INPUT PROCEDURE  ENTRADA
003070           OUTPUT PROCEDURE  SALIDA.
003075     STOP RUN.
003080 ENTRADA SECTION.
003090     ACCEPT  TABLAPROVIN
003100     OPEN INPUT  ARCHIOPE.
003105 LEER.
003110     READ ARCHIOPE  AT END  CLOSE ARCHIOPE  GO TO  SAL1.
003120     IF  NUMHIJOS IN FICHAOPE = ZERO
003130       OR SUELDO IN FICHAOPE < 20000   GO TO  LEER
003135     ELSE  NEXT SENTENCE.
003140     MOVE CORRESPONDING  FICHAOPE  TO  CLASIF
003150     RELEASE  CLASIF
003155     GO TO  LEER.
003170 SAL1.   EXIT.
003180 SALIDA SECTION.
003190     OPEN  OUTPUT  ARCHIMPRE
003200     DISPLAY '                    RELACION ALFABETICA DE OPERARIOS
003210-                'DE RICOSA EN CADA PROVINCIA'
003220     DISPLAY 'CODIGO      APELLIDOS Y NOMBRE                    NU
003230-                'M.HIJOS    SUELDO    PROVINCIA NACIM.'.
003235     PROCESO.
003240     RETURN  ARCHISORT  AT END  CLOSE ARCHIMPRE  GO TO  SAL2.
003250     MOVE SPACES  TO  IMPRESO
003260     MOVE PROVIN (PROVINACI  OF  CLASIF)  TO  PROVINCIA
003270     MOVE CORRESPONDING  CLASIF  TO  IMPRESO
003280     WRITE IMPRESO  AFTER ADVANCING 2 LINES
003290     GO TO  PROCESO.
003300 SAL2.   EXIT.
```

Las principales características a resaltar en este programa son las siguientes:

a) La estructura del registro a clasificar es diferente a la de los registros del archivo de entrada.

b) La clasificación se va a realizar por el contenido de varios campos.

c) Se describe un procedimiento de entrada (INPUT PROCEDURE) para seleccionar los registros que reunan las condiciones citadas en el enunciado del problema,y un procedimiento de salida (OUTPUT PROCEDURE) para imprimirlos según un determinado formato a medida que se vayan devolviendo del archivo de trabajo, una vez clasificados.

Con respecto al apartado a),los registros del archivo de entrada ARCHIOPE son de mayor longitud y contienen mayor número de campos que los registros del archivo de clasificación ARCHISORT. Es necesario especificar en este registro aquellos campos que van a servir como campos de clave durante la clasificación, es decir, PROVINACI, APENOM y CODIGO, así como los que vayan a ser utilizados dentro de la PROCEDURE: NUMHIJOS y SUELDO.

Referente al apartado b), en las opciones ASCENDING y DESCENDING se especifican de izquierda a derecha y por orden de importancia decreciente los campos que intervienen en la clasificación. En el ejemplo, como la clasificación por el campo PROVINACI debe ser previa a la del campo APENOM, se refleja en el programa en ese orden.

Si, por ejemplo, la clasificación se quisiera hacer en orden descendente de PROVINACI y APENOM, y ascendente de CODIGO, habría que especificar:

```
003040           DESCENDING KEY PROVINACI OF CLASIF
003041                          APENOM OF CLASIF
003050           ASCENDING KEY CODIGO OF CLASIF
```

Se han calificado los nombres PROVINACI, APENOM, etc. con el nombre-de-datos CLASIF, para que el compilador pueda diferenciarlos de PROVINACI OF ARCHIOPE (línea 001180) y APENOM OF ARCHIOPE (línea 001160).

Con respecto al apartado c), el INPUT PROCEDURE, que comprende desde la línea 003080 a la línea 003170, es prácticamente copia del INPUT del programa SORT2, al que, únicamente, se ha añadido la declaración MOVE (línea 003140) y la declaración ACCEPT TABLAPROVIN (línea 003090), esta última con el fin de llenar la tabla creada en la WORKING–STORAGE (líneas 002220 y 002230) con el nombre de las cincuenta provincias, para así poder ir transfiriendolas al registro IMPRESO (línea 003260) y ser impresas con su nombre, ya que en el archivo de entrada ARCHIOPE dichas provincias figuran con código numérico.

La declaración ACCEPT aunque se ha incluido dentro de INPUT PROCEDURE no es preceptivo, ya que podía haberse hecho, por ejemplo, inmediatamente después de PROCEDURE DIVISION (línea 003010) o en otro lugar, con tal de que el nombre de cada provincia se encuentre dentro de la tabla antes de ejecutar la declaración de la línea 003260.

El OUTPUT PROCEDURE está constituido por la Sección denominada –SALIDA– en la que se describen las acciones a realizar con los registros del archivo de trabajo una vez clasificados, es decir, una vez finalizadas las acciones que se indican en la INPUT PROCEDURE, automáticamente son clasificados los registros que se han transferido al archivo de trabajo y pasa el control del programa a la ejecución de OUTPUT PROCEDURE.

El procedimiento SALIDA abre el archivo de salida ARCHIMPRE e imprime en él los encabezamientos que ha de llevar (líneas 003200 a la 003230), y a continuación ejecuta la declaración RETURN.

1.1.4.1.- Declaración RETURN

FUNCION: Recuperar del archivo de trabajo los registros que han sido clasificados.

FORMATO:

RETURN nombre–de–archivo–de–clasificación [INTO identificador]

AT END declaración–imperativa

Observando el formato se puede decir que una declaración RETURN, es prácticamente una declaración READ aplicada al archivo de trabajo para clasificación, por cuanto todas las referencias hechas para dicha declaración en la página 203 son válidas para RETURN.

```
003235     PROCESO.
003240         RETURN ARCHISORT AT END CLOSE ARCHIMPRE GO TO SAL2.
003250         MOVE SPACES TO IMPRESO
003260         MOVE PROVIN (PROVINACI OF CLASIF) TO PROVINCIA
003270         MOVE CORRESPONDING CLASIF TO IMPRESO
003280         WRITE IMPRESO AFTER ADVANCING 2 LINES
003290         GO TO PROCESO.
003300     SAL2.    EXIT.
```

Las declaraciones comprendidas entre las líneas 003250 y 003280, realizan las funciones de: transferir al registro de salida –IMPRESO– el contenido de los campos del registro de clasificación (línea 003270), de la tabla de provincias transfiere la provincia correspondiente (línea 003260) y, por último, imprime el registro (línea 003280).

Una vez recuperados todos los registros, ejecuta las declaraciones CLOSE ARCHIMPRE y EXIT, en cuyo momento realiza las acciones oportunas para dar por finalizada la declaración SORT y pasar el control a la declaración siguiente , es decir, a STOP RUN, ya que OUTPUT PROCEDURE tiene un funcionamiento similar a la PERFORM, con lo que finaliza la ejecución del programa.

2.- CARACTERISTICA DE ESCRITOR DE INFORMES

2.1.- GENERALIDADES

El Escritor de Informes es un medio que el lenguaje COBOL, a través de nuevas cláusulas y declaraciones, pone a disposición del programador para ser utilizado en aquellos trabajos que requieren la confección de informes impresos.

Si se tiene en cuenta que con las cláusulas y declaraciones que hasta ahora se han descrito, es suficiente para la realización de cualquier tipo de trabajo cuyo objeto sea la obtención de un informe impreso, parece indicar que no es necesario el estudio de aquéllas que constituyen el Escritor de Informes. Sin embargo, varias razones lo hacen aconsejable, ya que:

- Existen trabajos en los que empleando el Escritor de Informes se reduce el tiempo de proceso necesario para la confección del informe y

- Permite al programador mayor flexibilidad al poder elegir para cada trabajo el método de programación más adecuado.

El Escritor de Informes se utiliza principalmente en aquellos trabajos que requieren la confección de amplios informes en los que haya que imprimir un número elevado de páginas; se acompañen de cabeceras y pies en cada página; haya que imprimir líneas con el contenido de las cantidades parciales o totales que se vayan produciendo durante el proceso, etc.

La codificación de un programa que tenga como finalidad la obtención de un informe impreso utilizando el Escritor de Informes, presenta en alguna de las DIVISIONES diferencias notables con respecto a los programas que hasta ahora se han codificado. Efectivamente, la codificación de la IDENTIFICATION y ENVIRONMENT es invariable, mientras que en las otras dos DIVISIONES alguna de las diferencias más importantes son las siguientes:

- En la DATA DIVISION se utiliza una nueva sección: REPORT SECTION donde se describe el formato físico que va a tener cada página del informe, y se define el contenido (SOURCE), formato (LINE, COLUMN) y operaciones de suma (SUM) de cada una de las líneas que han de componer el informe.

- En la PROCEDURE DIVISION se utilizan declaraciones como INITIATE, GENERATE y TERMINATE con funciones análogas a las declaraciones WRITE, MOVE, ADD, etc.,.

Como resumen de todo lo expuesto se puede decir:

- Que cualquier informe impreso puede realizarse, bien codificando un programa en el que se haga uso del Escritor de Informes, o bien con el sistema que se ha venido empleando en los diversos programas que se han expuesto.

- Que el programador es el responsable de la elección del método más adecuado, teniendo siempre presente las ventajas e inconvenientes de cada uno de ellos, y

- Que se incrementa la codificación necesaria para la obtención del informe en la DATA DIVISION, a la vez que se reduce la de la PROCEDURE DIVISION.

2.2.- FORMATO de un INFORME IMPRESO

Un informe impreso está formado por un conjunto de líneas impresas. Dentro de un informe pueden existir uno o varios grupos de informe, recibiendo éstos diferente denominación según el lugar que ocupen dentro del mismo, frecuencia con que se realice su escritura y, su contenido.

El Escritor de Informes permite presentar un informe impreso estructurando los diferentes grupos conforme al siguiente formato:

CABECERA DE INFORME
(REPORT HEADING)

CUERPO DEL INFORME

- **CABECERA DE PAGINA**
  (PAGE HEADING)
- **CUERPO DE PAGINA**
  - **CABECERA DE CONTROL**
    (CONTROL HEADING)
  - **LINEAS DE DETALLE**
    (DETAIL)
  - **PIE DE CONTROL**
    (CONTROL FOOTING)
- **PIE DE PAGINA**
  (PAGE FOOTING)

PIE DEL INFORME
(REPORT FOOTING)

Las palabras inglesas que figuran debajo de cada grupo son los nombres con los que hay que especificarlos dentro del programa para poder ser reconocidos por el compilador. A fin de familiarizar al programador, se hará mención, normalmente, a dichos grupos con su nombre inglés.

Un modelo completo de informe impreso estaría formado por todos los grupos citados, pero ello no quiere decir que sea obligatorio especificar todos los grupos en cada informe, ya que para cada tipo de trabajo se utilizarán sólo aquéllos que sean necesarios, es decir, pueden existir informes que sólo contengan líneas de detalle (resultados), otros que contengan una cabecera de informe y líneas de detalle, otros una cabecera de página, líneas de detalle y un pié de informe, y así,las múltiples combinaciones posibles.

Para ir fijando conceptos acerca de la denominación, posición y contenido de cada grupo se utilizará el informe impreso que se muestra en las figuras 25, 26 y 27, obtenido por la ejecución del programa REPORIMP cuyo enunciado y codificación figuran en las págs. 341y342.

En la realización de cualquier trabajo, lo normal es que al programador se le proporcione, sobre hojas–formato de salida de impresora,el formato del informe en el que figure la posición y contenido de cada grupo, a fin de que, a partir de dicho formato,haga su codificación en el programa dentro de la REPORT SECTION (según se verá posteriormente). Pero puede suceder que le presenten las hojas de un informe impreso, como el obtenido por el programa REPORIMP, y tenga que deducir de qué grupo se trata, a fin de llevar a cabo la codificación. En este caso, no le será difícil reconocer de qué grupo se trata, si se tienen en cuenta las siguientes consideraciones:

- REPORT HEADING se imprime una sola vez en todo el informe y antes de la impresión de cualquier otro grupo. Puede ir sólo en una página o al comienzo de una página junto con otros grupos de informe.

- REPORT FOOTING se imprime una sola vez en todo el informe, bien sólo en una página o al final de la última página después de los restantes grupos.

- PAGE HEADING se imprime al comienzo de cada página del informe.

- PAGE FOOTING se imprime al final de cada página.

- DETAIL son las líneas de detalle o de resultados, van situadas entre la cabecera y pié de página y son fáciles de localizar por ser el número de ellas superior al de cualquier grupo.

- CONTROL HEADING se imprime inmediatamente antes de determinadas líneas de detalle que reunan ciertas condiciones, por ejemplo, en el informe citado, antes de imprimir el código y nombre de los operarios de cada provincia.

- CONTROL FOOTING se imprime después de las líneas de detalle que reunan determinadas condiciones; por ejemplo, cada vez que acaban de imprimirse los operarios pertenecientes a una determinada provincia.

Por tanto, examinando las figuras 25, 26 y 27, cada uno de los grupos de informe citados estarían constituidos por las siguientes líneas.

- Líneas 2 y 6 de la primera página – REPORT HEADING.

- Líneas 7 y 9 de la primera, segunda, etc., páginas – PAGE HEADING.

- Líneas 11, 12, 33 y 34 de la primera página, 28 y 29 de la segunda, etc., –CONTROL HEADING.

- Líneas 14, 16, 18, 20, etc., – DETAIL.

- Líneas 29, 30 y 31 de la primera página, etc., – CONTROL FOOTING.

- Línea 43 de la última página, a continuación del último CONTROL FOOTING el CONTROL FOOTING FINAL.

- Línea 60 de cada página – PAGE FOOTING.

- Líneas 61 y 63 de la última página – REPORT FOOTING.

Como comprobación de cuanto antecede, se especifica, dentro de la REPORT SECTION (líneas 002090 a la 004080) del citado programa, el nombre de cada grupo de informe precedido del nivel 01 y seguido de las líneas (LINE) que lo componen junto con su contenido (VALUE, SOURCE, etc) hasta encontrar el siguiente número de nivel 01.

PROGRAMA EJEMPLO UTILIZANDO EL ESCRITOR DE INFORMES.

El siguiente programa utiliza la característica del Escritor del Informe para producir el informe impreso cuyo formato se muestra en las figuras 25, 26 y 27. El archivo de entrada lo constituye las fichas de cada operario utilizadas para la confección de la nómina. Para poder realizar el trabajo se ha hecho una clasificación previa de dicho archivo por provincia de nacimiento, y dentro de cada provincia,por orden alfabético de apellidos y nombre.

```
001010  ID DIVISION.
001020    PROGRAM-ID.    REPORIMP.
001030*   PROGRAMA QUE CONFECCIONA UN INFORME UTILIZANDO EL ESCRITOR DE
001035*   INFORMES.
001040  ENVIRONMENT DIVISION.
001050    CONFIGURATION SECTION.
001060      SPECIAL-NAMES.
001065          DECIMAL-POINT IS COMMA.
001070    INPUT-OUTPUT SECTION.
001080      FILE-CONTROL.
001090          SELECT ARCHIFICHA      ASSIGN SYS004-UR-2540R-S.
001100          SELECT ARCHIMPRESO     ASSIGN SYS005-UR-1403-S.
001110  DATA DIVISION.
001120    FILE SECTION.
001130      FD  ARCHIFICHA          RECORDING F
001135                              LABEL RECORD OMITTED
001136                              DATA RECORD REGISOPE.
001140      01  REGISOPE.
001150          02  CODIGO         PIC X(5).
001160          02  APELLNOM       PIC X(35).
001170          02  FILLER         PIC X(6).
001180          02  PROVINACIM     PIC 99.
001190          02  FILLER         PIC X(32).
001200      FD  ARCHIMPRESO         RECORDING F
001205                              LABEL RECORD OMITTED
001210                              REPORT IS INFORIMPRE.
001220    WORKING-STORAGE SECTION.
001230      01  CONTAOPE           PIC 9(5)   VALUE 1.
001240      01  TABLAPROVIN.
001250          02  PROVINCIA      OCCURS 50  PIC X(15).
002005*
002010  REPORT SECTION.
002020    RD  INFORIMPRE
002030          CONTROLS ARE FINAL      PROVINACIM
002040          PAGE LIMIT IS 65 LINES
002050                HEADING 2
002060                FIRST DETAIL 11
002070                LAST DETAIL 58
002080                FOOTING 59.
002090      01  GRUPO1       TYPE REPORT HEADING.
002100          02  ALFA     LINE NUMBER 2
002110                        COLUMN NUMBER 30
002120                        PIC X(75)
002130                        VALUE 'EMPRESA RICOSA
002140-                             '          SECCION DE ESTADISTICA'.
```

```
002150     02  BETA  PIC  X(56)
002160                 COLUMN  39
002170                 LINE    6
002180                 VALUE 'ESTADISTICA DE OPERARIOS NACIDOS EN C
002190-                      'ADA PROVINCIA'.
002200   01  GRUPO2        TYPE  PAGE HEADING.
002210     02  CAB1        LINE 7.
002220           03  CAB11  COLUMN 105  PIC X(12)  VALUE 'PAGINA NUM. '.
002230           03  CAB12  COLUMN 117  PIC 9(4)   SOURCE PAGE-COUNTER.
002240     02  CAB2        LINE 9
002245                      COLUMN  30  PIC X(75)  VALUE ALL '='.
002250   01  GRUPO3        TYPE  CONTROL HEADING  PROVINACIM.
003010     02  ZETA        LINE NUMBER  PLUS 2.
003020           03  ZETA1  COLUMN  44  PIC X(33)
003030                      VALUE 'RELACION DE OPERARIOS NACIDOS EN '.
003040           03  ZETA2  COLUMN  77  PIC X(16)
003050                      SOURCE  PROVINCIA (PROVINACIM).
003060     02  LANDA  LINE  PLUS 1
003070                      COLUMN  44  PIC X(49)  VALUE ALL '-'.
003080   01  LINEADETALLE  TYPE  DETAIL.
003090     02  LINEA  LINE  PLUS 2.
003100           03  CAMPO1  COLUMN  44  PIC X(5)   SOURCE  CODIGO.
003115           03  CAMPO2  COLUMN  53  PIC X(35)  SOURCE  APELLNOM.
003120   01  CONTROL1      TYPE  CONTROL FOOTING PROVINACIM.
003130     02  CON1        LINE  PLUS 1
003135                      COLUMN  30  PIC X(75)  VALUE ALL '-'.
003140     02  CON2        LINE  PLUS 1.
003150           03  CON21  COLUMN  44  PIC X(31)
003160                      VALUE 'NUMERO DE OPERARIOS NACIDOS EN '.
003170           03  CON22  COLUMN  75  PIC X(16)
003180                      SOURCE  PROVINCIA (PROVINACIM).
003190           03  CON23  COLUMN  91  PIC X(3)   VALUE ' = '.
003200           03  CON24  COLUMN  94  PIC Z.ZZ9
003205                      SUM  CONTAOPE  UPON  LINEADETALLE.
003210     02  CON3        LINE  PLUS 1
003215                      COLUMN  30  PIC X(75)  VALUE ALL '*'.
003220   01  CONTROL2      TYPE  CONTROL FOOTING  FINAL.
003230     02  CONT1       LINE  PLUS 3.
003240           03  CONT11  COLUMN  44  PIC X(49)  VALUE 'TOTAL DE OPERA
003250-                     'RIOS RELACIONADOS EN EL INFORME = '.
004010           03  CONT12  COLUMN  93  PIC ZZ.ZZ9  SUM  CON24.
004020   01  PIEPAG        TYPE  PAGE FOOTING
004025                      LINE  NUMBER 60
004030                      COLUMN  10  PIC X(10)  VALUE ALL '-'.
004040   01  TYPE  REPORT FOOTING.
004050     02  LINE  PLUS 2
004055                      COLUMN  61  PIC X(15)  VALUE 'FIN DEL INFORME'.
004060     02  LINE  PLUS 2.
004070           03         COLUMN  61  PIC X(7)   VALUE 'RICOSA '.
004080           03         COLUMN  68  PIC X(8)   SOURCE  CURRENT-DATE.
004085
004090 PROCEDURE DIVISION.
004100 COMIENZO.
004110           ACCEPT  TABLAPROVIN
004120           OPEN  INPUT  ARCHIFICHA      OUTPUT  ARCHIMPRESO
004130           INITIATE  INFORIMPRE.
004140 LEER.
004150           READ  ARCHIFICHA  AT END  GO TO  FIN.
004160           GENERATE  LINEADETALLE
004170           GO TO  LEER.
004180 FIN.
004190           TERMINATE  INFORIMPRE
004200           CLOSE  ARCHIFICHA  ARCHIMPRESO
004210           STOP RUN.
```

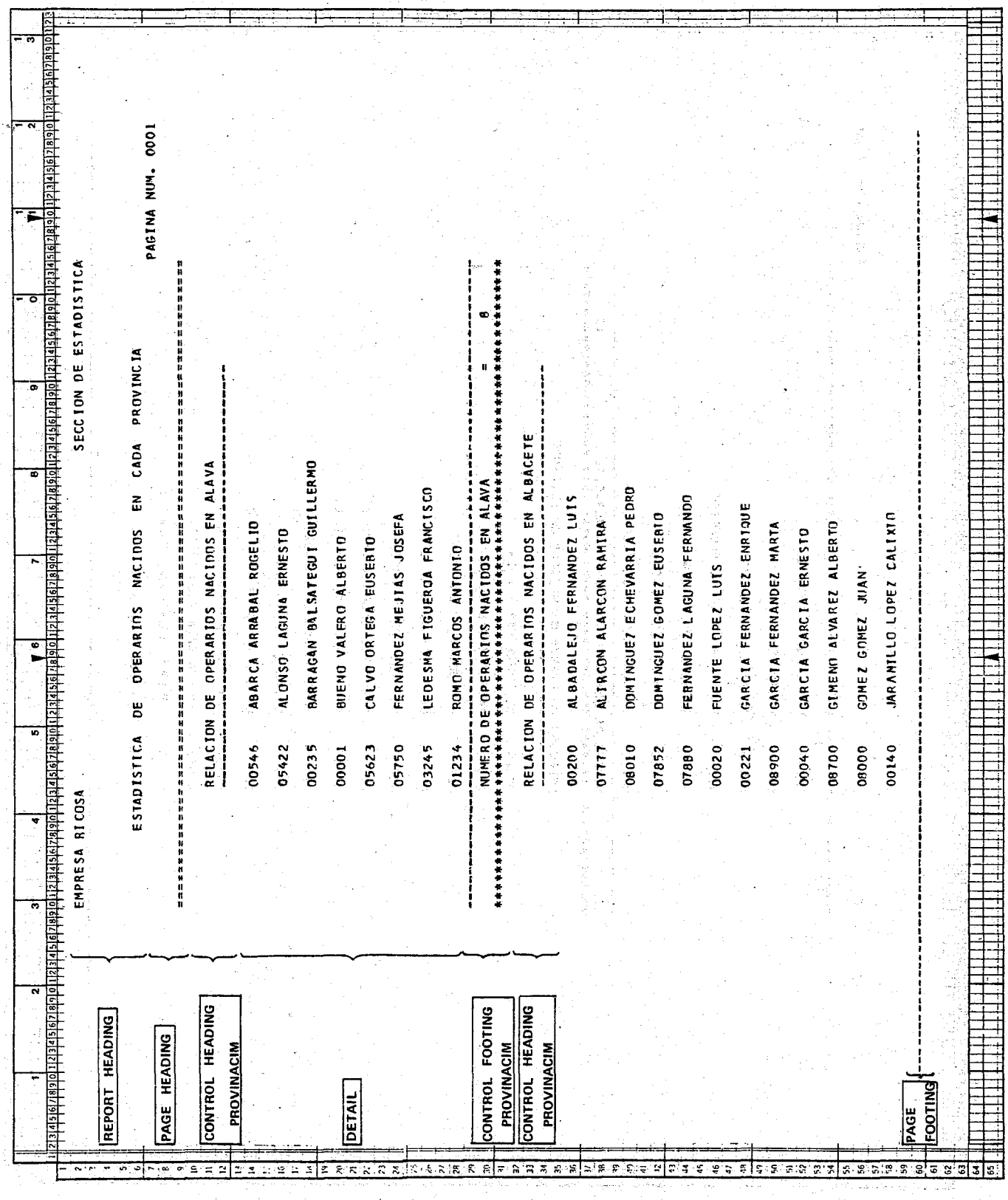
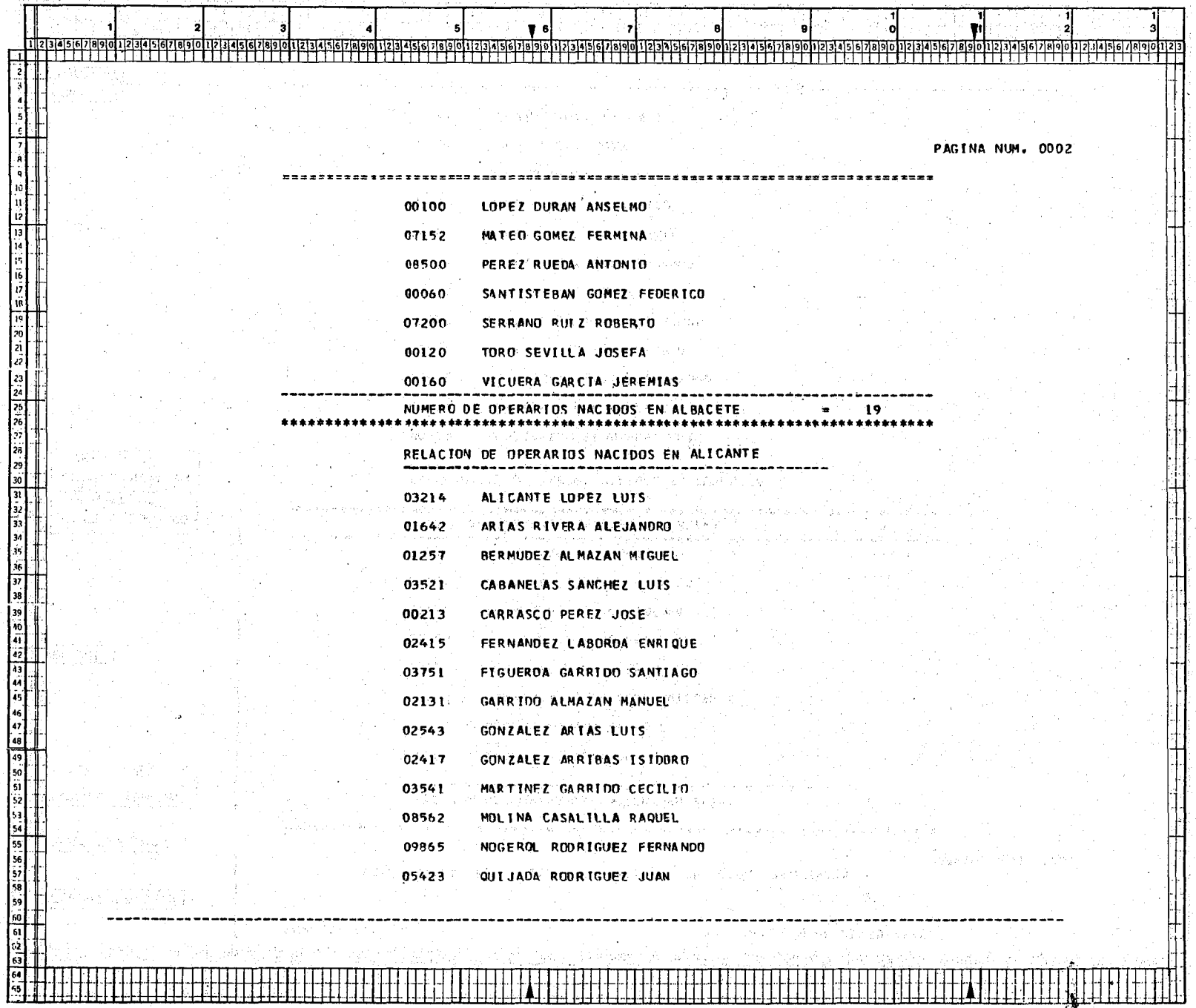

REPORT HEADING
EMPRESA RICOSA
SECCION DE ESTADISTICA
ESTADISTICA DE OPERARIOS NACIDOS EN CADA PROVINCIA
PAGINA NUM. 0001
PAGE HEADING
CONTROL HEADING PROVINACIM
RELACION DE OPERARIOS NACIDOS EN ALAVA
00546 ABARCA ARRABAL ROGELIO
05422 ALONSO LAGUNA ERNESTO
00235 BARRAGAN BALSATEGUI GUILLERMO
DETAIL
00001 BUENO VALERO ALBERTO
05623 CALVO ORTEGA EUSEBIO
05750 FERNANDEZ MEJIAS JOSEFA
03245 LEDESMA FIGUEROA FRANCISCO
01234 ROMO MARCOS ANTONIO
CONTROL FOOTING PROVINACIM
NUMERO DE OPERARIOS NACIDOS EN ALAVA = 6
CONTROL HEADING PROVINACIM
RELACION DE OPERARIOS NACIDOS EN ALBACETE
00200 ALBADALEJO FERNANDEZ LUIS
07777 ALARCON ALARCON RAMIRA
08010 DOMINGUEZ ECHEVARRIA PEDRO
07852 DOMINGUEZ GOMEZ EUSEBIO
07880 FERNANDEZ LAGUNA FERNANDO
00020 FUENTE LOPEZ LUIS
00221 GARCIA FERNANDEZ ENRIQUE
08900 GARCIA FERNANDEZ MARTA
00040 GARCIA GARCIA ERNESTO
08700 GIMENO ALVAREZ ALBERTO
08000 GOMEZ GOMEZ JUAN
00140 JARAMILLO LOPEZ CALIXTO
PAGE FOOTING

FIGURA 25



PAGINA NUM. 0002

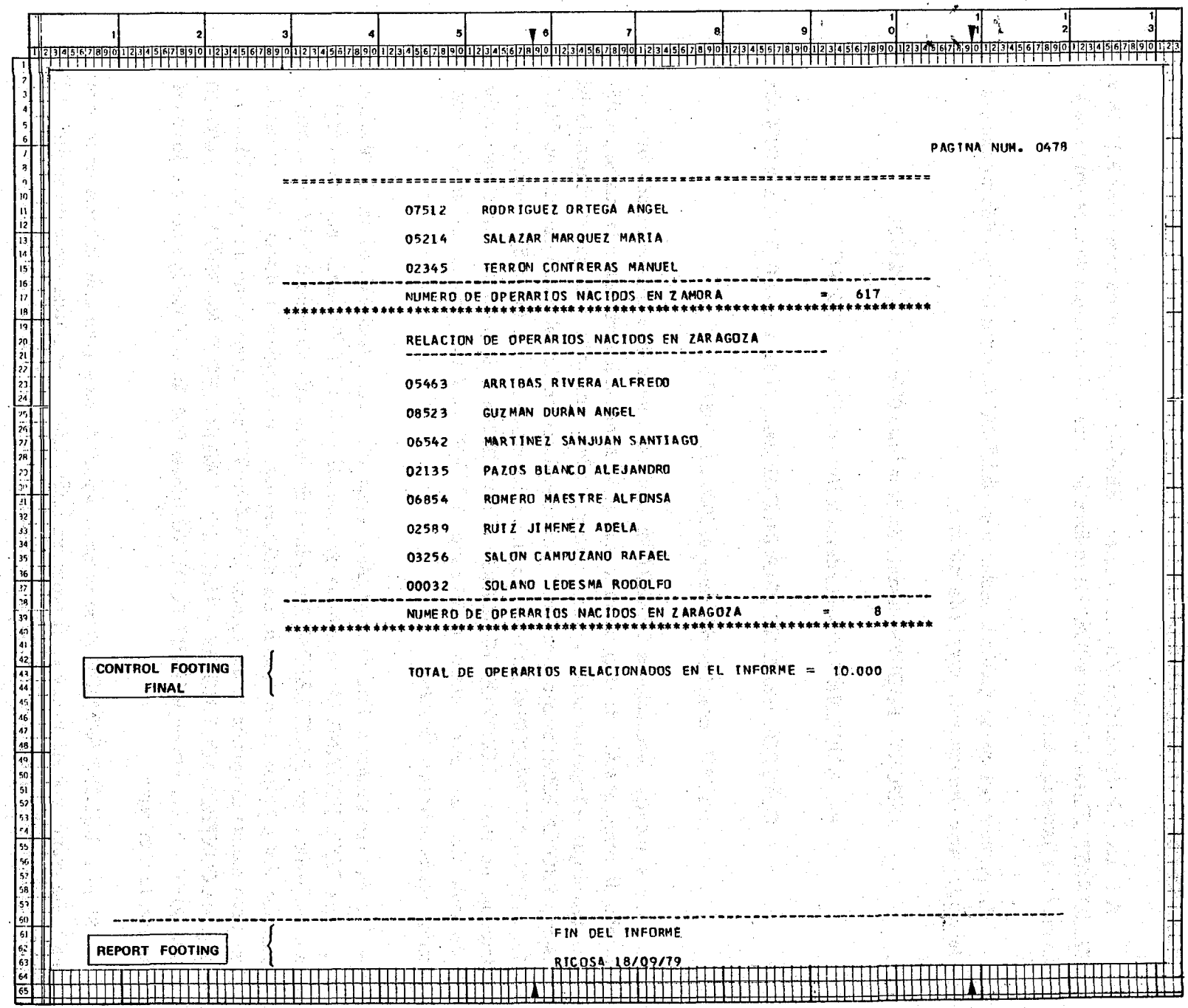
00100	LOPEZ DURAN ANSELMO
07152	MATEO GOMEZ FERMINA
08500	PEREZ RUEDA ANTONIO
00060	SANTISTEBAN GOMEZ FEDERICO
07200	SERRANO RUIZ ROBERTO
00120	TORO SEVILLA JOSEFA
00160	VICUERA GARCIA JEREMIAS

NUMERO DE OPERARIOS NACIDOS EN ALBACETE = 19

RELACION DE OPERARIOS NACIDOS EN ALICANTE

03214	ALICANTE LOPEZ LUIS
01642	ARIAS RIVERA ALEJANDRO
01257	BERMUDEZ ALMAZAN MIGUEL
03521	CABANELAS SANCHEZ LUIS
00213	CARRASCO PEREZ JOSE
02415	FERNANDEZ LABORDA ENRIQUE
03751	FIGUERDA GARRIDO SANTIAGO
02131	GARRIDO ALMAZAN MANUEL
02543	GONZALEZ ARIAS LUIS
02417	GONZALEZ ARRIBAS ISIDORO
03541	MARTINEZ GARRIDO CECILIO
08562	MOLINA CASALILLA RAQUEL
09865	NOGEROL RODRIGUEZ FERNANDO
05423	QUIJADA RODRIGUEZ JUAN

FIGURA 26



PAGINA NUM. 0478

07512 RODRIGUEZ ORTEGA ANGEL

05214 SALAZAR MARQUEZ MARIA

02345 TERRON CONTRERAS MANUEL

NUMERO DE OPERARIOS NACIDOS EN ZAMORA = 617

RELACION DE OPERARIOS NACIDOS EN ZARAGOZA

05463 ARRIBAS RIVERA ALFREDO

08523 GUZMAN DURAN ANGEL

06542 MARTINEZ SANJUAN SANTIAGO

02135 PAZOS BLANCO ALEJANDRO

06854 ROMERO MAESTRE ALFONSA

02589 RUIZ JIMENEZ ADELA

03256 SALON CAMPUZANO RAFAEL

00032 SOLANO LEDESMA RODOLFO

NUMERO DE OPERARIOS NACIDOS EN ZARAGOZA = 8

CONTROL FOOTING FINAL

TOTAL DE OPERARIOS RELACIONADOS EN EL INFORME = 10.000

REPORT FOOTING

FIN DEL INFORME

RICOSA 18/09/79

FIGURA 27

2. 3.– ESCRITURA del INFORME

El orden en que se imprime en el informe cada grupo de informe, es un orden preestablecido y controlado por el Escritor de Informes.

A la vista de la codificación del programa REPORIMP, las declaraciones GENERATE y TERMINATE (líneas 004160 y 004190) realizan la función de escritura de cada grupo de informe en la forma siguiente:

– La primera vez que se ejecuta la declaración GENERATE origina la impresión automática del contenido de los siguientes grupos de informe:

- REPORT HEADING (descrito desde la línea 002090 a la 002190).
- PAGE HEADING (descrito desde la línea 002200 a la 002245).
- CONTROL HEADING PROVINACIM (descrito de la línea 002250 a la 003070).
- DETAIL (descrito en la línea 003080 a la 003115).

– Las siguientes declaraciones GENERATE efectuan, antes de imprimirse cada línea de detalle, la comprobación de si el operario que se va a imprimir pertenece a la misma provincia que el que se imprimió anteriormente o, lo que es lo mismo, si ha cambiado el contenido del campo PROVINACIM; caso afirmativo, imprime el contenido del grupo de informe CONTROL FOOTING PROVINACIM (descrito en las líneas 003120 a la 003215), a continuación imprime un nuevo CONTROL HEADING PROVINACIM y la línea de detalle que estaba pendiente. Caso de que el contenido del campo PROVINACIM fuera igual al del anterior operario, escribe la línea de detalle.

– Cuando se alcanza el final de la página, y antes de imprimir la siguiente línea de detalle, se imprime PAGE FOOTING (descrito de la línea 004020 a la 004030), salta a una nueva página e imprime de nuevo PAGE HEADING y la línea de detalle que no pudo imprimir en la página anterior. Así prosigue la confección del informe hasta que imprime la última línea de detalle en cuyo momento ejecuta la declaración TERMINATE que obliga a imprimir, por última vez, el grupo CONTROL FOOTING PROVINACIM y CONTROL FOOTING FINAL (línea 003220 a la 004010) y, por último, REPORT FOOTING (líneas 004040 a 004080), concluyendo en ese momento el informe.

Si en la REPORT SECTION no se describe alguno de estos grupos de informe no se imprime dicho grupo; por ejemplo, si en el programa REPORIMP no se hubiera descrito el grupo REPORT HEADING no se hubieran impreso las líneas 2 y 6 del encabezamiento del informe.

2.4.- CONTROLES y RUPTURAS de CONTROL

Al especificar la cláusula CONTROL en el programa REPORIMP: CONTROLS ARE FINAL PROVINACIM (línea 002030) su función es indicar al Escritor de Informes que antes de imprimir cada línea de detalle debe controlar, o lo que es lo mismo, comprobar si el contenido del campo de datos PROVINACIM (línea 001180) que figura en cada ficha del archivo operarios, es el mismo o diferente que el del operario anterior que acaba de imprimirse; caso de ser el mismo se imprimirá una nueva línea de detalle, pero si es diferente quiere decir que se ha producido una RUPTURA DE CONTROL, en cuyo caso, antes de imprimir dicha línea de detalle el Escritor realizará la impresión del grupo de informe CONTROL FOOTING PROVINACIM, inmediatamente después el grupo CONTROL HEADING PROVINACIM y a continuación la línea de detalle. Estas acciones se repiten a lo largo de la confección del informe cada vez que cambie el contenido del campo PROVINACIM.

Al campo de datos PROVINACIM se le denomina campo de CONTROL.
La palabra FINAL aunque no es un campo de datos, el Escritor de Informes lo interpreta como un Control cuya ruptura se produce solamente al comienzo y fin del informe, lo cual obliga a imprimir el grupo CONTROL HEADING FINAL al comienzo de la escritura del informe y el CONTROL FOOTING FINAL al final de informe inmediatamente después de la impresión de la última línea de detalle.

2.5.- Cláusula REPORT

En la DATA DIVISION, dentro de la FILE SECTION, además de las cláusulas necesarias para la descripción de un archivo de salida por impresora, es necesario incluir la cláusula REPORT (línea 001210) al hacer la descripción del archivo que va a contener el informe.

FORMATO:

REPORT IS nombre-de-informe-1, nombre-de-informe-2 . . .

Esta cláusula servirá para dar un nombre al informe (INFORIMPRE), con el que el compilador reconocerá todas las referencias que se hagan al informe dentro del programa, por ejemplo, en las líneas 002020, 004130 y 004190.

```
001200     FD  ARCHIMPRESO RECORDING F  LABEL RECORD OMITTED.
001210     ------------>  REPORT IS INFORIMPRE.
001220   WORKING-STORAGE SECTION.
```

```
004090   PROCEDURE DIVISION.
004100     COMIENZO.
004110         ACCEPT TABLAPRDVIN
004120         OPEN INPUT ARCHIFICHA    OUTPUT ARCHIMPRESO
004130         INITIATE INFORIMPRE.     <-------
004140     LEER.
004150         READ ARCHIFICHA END GO TO FIN.
004160         GENERATE LINEADETALLE
004170         GO TO LEER.
004180     FIN.
004190         TERMINATE INFORIMPRE     <-------
004200         CLOSE ARCHIFICHA ARCHIMPRESO
004210         STOP RUN.
```

2.6.- REPORT SECTION

Dentro de la REPORT SECTION (líneas 002010 a 004080) se distinguen dos partes bien diferenciadas:

- La primera parte (líneas 002020 a 002080) comprende la descripción física del informe, en la que se refleja: el nombre del informe, los campos de datos que han de utilizarse como controles, y una serie de parámetros que servirán para fijar el número máximo de líneas que pueden imprimirse por página, así como la posición en que debe imprimirse cada grupo de informe dentro de cada página.

- La segunda parte (líneas 002090 a 004080) está constituida por una serie de entradas que incluyen, junto a los nombres de cada grupo de informe, la descripción de cada uno de los campos compuestos y elementales que componen dichos grupos.

ESTRUCTURA:

REPORT SECTION.

Entrada de descripción de informe

Entrada de descripción de grupos de informe

2.6.1.- ENTRADA de DESCRIPCION de INFORME

FORMATO:

REPORT SECTION.

RD nombre-de-informe
[Cláusula CODE]
[Cláusula CONTROL]
[Cláusula PAGE LIMIT]

RD es el indicador de nivel.

Nombre-de-informe es el nombre del informe que se especificó dentro de la cláusula REPORT al hacer la descripción del archivo impreso (INFORIMPRE, línea 001210).

```
002010     REPORT SECTION.
002020       RD  INFOR IMPRE
002030         CONTROLS  ARE FINAL  PROVINACIM
002040         PAGE LIMIT IS 65 LINES
002050             HEADING 2
```

La cláusula CODE es necesaria cuando se escribe más de un informe en un archivo (no se hace su descripción por salirse de los propósitos de este texto).

2.6.1.1.– Cláusula CONTROL

FUNCION: Indicar los identificadores (campos de datos) que se han de probar antes de imprimir un grupo de informe de detalle. Si el resultado de la prueba indica que el valor del identificador ha variado, origina una ruptura de control, debiendo realizar el Escritor de Informes determinadas acciones de acuerdo con lo que haya descrito el programador en la REPORT SECTION.

FORMATO:

```
{ CONTROL IS  }  { FINAL                                              }
{             }  { identificador-1 [identificador-2] ...              }
{ CONTROLS ARE}  { FINAL identificador-1 [identificador-2] ...        }
```

Identificador–1, identificador–2, ... son campos de datos descritos en la FILE o en la WORKING–STORAGE SECTION.

Así, en el programa REPORIMP, por haber descrito los grupos: CONTROL HEADING y CONTROL FOOTING PROVINACIM, cada vez que se produzca una ruptura de control por cambiar el nombre de la provincia de un operario respecto del operario anterior, las acciones que realizará el Escritor antes de imprimir la línea de detalle conteniendo a dicho operario serán: Impresión de grupos de informe CONTROL relacionados con PROVINACIM, y posteriormente impresión de la línea de detalle. Si hubiera omitido la descripción de alguno de estos grupos no se realizaría la impresión del mismo.

El orden en que debe especificarse cada identificador no puede ser caprichoso, sino que debe hacerse por niveles de mayor a menor importancia; así,por ejemplo, si en un informe se desea obtener una relación en la que se emplee una línea por cada agente y día conteniendo el nombre del agente junto con las ventas de automóviles que ha realizado en ese día,y que al final de cada día se imprima una línea en la que se refleje el total de ventas de todos los agentes en dicho día; otra línea cuando se llegue al final de cada mes reflejando las ventas del mes, otra al final de año,y otra al final de los cuatro años que se han estado vendiendo automóviles, deberá especificar una serie de campos de control en la cláusula CONTROL como:

CONTROLS ARE FINAL AÑO MES DIA

El Escritor de Informes antes de imprimir una línea de detalle, prueba todos los campos de control comenzando por el nivel de mayor importancia hasta llegar al de menor, es decir, en el ejemplo anterior, iría probando desde FINAL hasta llegar a DIA. Si se produce una ruptura de control en un determinado nivel, por ejemplo MES, lleva implícito también la ruptura de control en el nivel DIA. Así, en el programa REPORIMP, cuando se llega al final del archivo ARCHIFICHA se produce la ruptura de control FINAL, la cual lleva implícita la ruptura del control PROVINACIM, por lo que se escriben CONTROL FOOTING PROVINACIM y CONTROL FOOTING FINAL respectivamente.

De todo lo expuesto se deduce que cada vez que se produce una ruptura de control, antes de imprimir el grupo de detalle, se imprime siempre que se hayan descrito dentro de la REPORT SECTION los grupos siguientes:

- Pié de control de nivel inferior.
- Pié de control de nivel superior siguiente y sucesivos hasta imprimir el pié de control del nivel que produjo la ruptura de control.
- Encabezamiento de control que produjo la ruptura.
- Encabezamiento de control del nivel inferior siguiente y sucesivos hasta imprimir el control de menor nivel.

2.6.1.2.- Cláusula PAGE

FUNCION: Describir el formato físico de una página del informe.

FORMATO:

```
PAGE [ LIMIT IS    ]          { LINE  }
     [             ] entero-1 {       }
     [ LIMITS ARE  ]          { LINES }

[HEADING entero-2]
[FIRST DETAIL entero-3]
[LAST DETAIL entero-4]
[FOOTING entero-5]
```

Entero-1, entero-2, etc., constituyen una serie de parámetros que permiten flexibilizar la impresión de cada grupo de informe, ya que establecen los márgenes o límites dentro de los cuales debe imprimirse cada grupo. Por tanto, evitan la rigidez que supondría tener que imprimir cada grupo de informe siempre en una posición fija de cada página del informe.

```
002010     REPORT SECTION.
002020       RD  INFORIMPRE
002030           CONTROLS ARE FINAL     PROVINACIM
002040           PAGE  LIMIT  IS      65 LINES
002050                 HEADING          2
002060                 FIRST  DETAIL  11
002070                 LAST   DETAIL  58
002080                 FOOTING        59.
```

El significado de cada uno de los parámetros es el siguiente:

Entero-1: Indica el número de la última línea en la que puede escribirse dentro de cada página del informe.

Entero–2: Es el número de la primera línea donde se puede comenzar a imprimir los grupos REPORT HEADING/FOOTING y PAGE HEADING.

Entero–3: Es el número de la primera línea donde puede comenzar a imprimirse CONTROL HEADING, DETAIL y CONTROL FOOTING.

Entero–4: Es el número de la última línea a partir de la cual no puede imprimirse ningún grupo CONTROL HEADING ni DETAIL.

Entero–5: Es el número de la última línea en la que puede imprimirse un CONTROL FOOTING.

La figura 28 muestra junto con cada uno de los parámetros y grupos de informe, los límites entre los cuales puede realizarse su impresión dentro de cada página del informe.

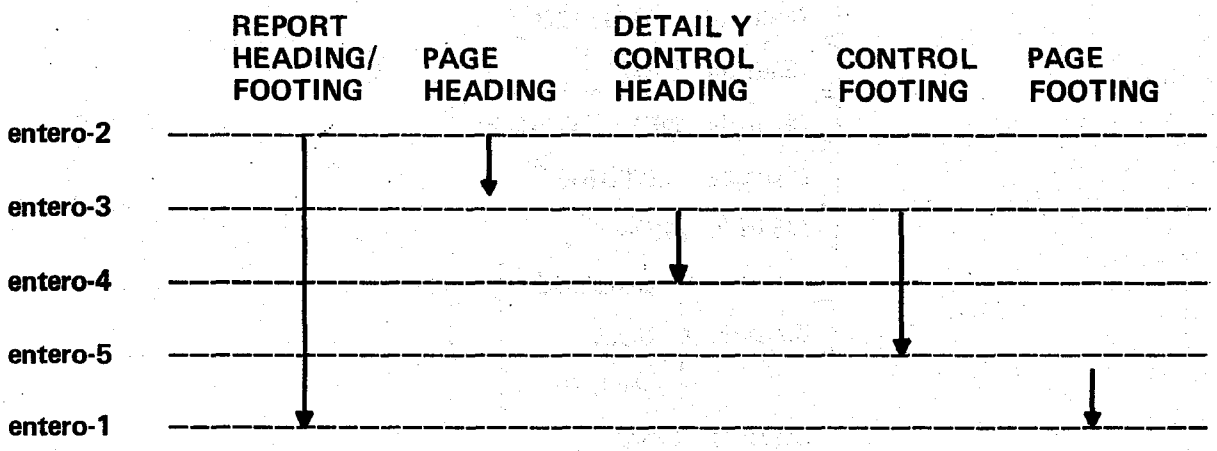


FIGURA 28

La interpretación de esta figura es la siguiente:

- Los grupos de informe REPORT HEADING y REPORT FOOTING puede extenderse su impresión desde la línea entero–2 hasta la línea entero–1 inclusive.

- El grupo de informe PAGE HEADING puede extenderse su impresión desde la línea entero–2 hasta la línea entero–3 menos 1 inclusive.

- Los grupos de informe CONTROL HEADING y DETAIL pueden extender su impresión desde la línea entero–3 a la línea entero–4 inclusive.

- Los grupos de informe CONTROL FOOTING pueden imprimirse desde la línea entero–3 hasta la línea entero–5 inclusive.

- El grupo de informe PAGE FOOTING puede imprimirse desde la línea entero–5 más 1 hasta la línea entero–1 inclusive.

2.6.2.– ENTRADA de DESCRIPCION de GRUPOS de INFORME

La descripción de un grupo de informe difiere muy poco de como se hacía la descripción de un registro de datos. No obstante, existe una diferencia esencial entre ambas descripciones, debida a que en la descripción del grupo de informe se incluyen nuevas cláusulas que permiten indicar dónde y cuándo hay que imprimir cada grupo.

FORMATO:

```
número–de–nivel [ nombre–de–datos–1 ]

                [ Cláusula BLANK WHEN ZERO ]
                [ Cláusula COLUMN ]
                [ Cláusula GROUP ]
                [ Cláusula JUSTIFIED ]
                [ Cláusula LINE ]
                [ Cláusula NEXT GROUP ]
                [ Cláusula PICTURE ]
                [ Cláusula RESET ]
                [ Cláusula { SOURCE } ]
                           { SUM    }
                           { VALUE  }
                [ Cláusula TYPE ]
                [ Cláusula USAGE ]
```

Nombre–de–datos–1 es opcional y sólo es obligatorio especificarlo cuando se haga referencia a él en la PROCEDURE DIVISION. Así, en el programa REPORIMP, únicamente sería necesario especificar LINEADETALLE (línea 003080) por hacerse referencia a él en la declaración GENERATE, y, por tanto, los nombres GRUPO1 (línea 002090), GRUPO2 (línea 002200), ALFA (línea 002100), BETA (línea 002150), ZETA (línea 003010), etc., no sería necesario especificarlos, según se hace en el grupo REPORT FOOTING (línea 004040).

```
003080     01 LINEADETALLE TYPE DETAIL.
003090        02 LINEA LINE PLUS 2.
003100           03 CAMPO1 COLUMN NUMBER 44 PIC X(5) SOURCE CODIGO.
```

```
004040     01 TYPE REPORT FOOTING.
004050        02 LINE PLUS 2 COLUMN 61 PIC X(15) VALUE 'FIN DEL INFORME'.
004060        02 LINE PLUS 2.
```

Excepto nombre–de–datos–1 que, cuando se especifica, ha de hacerse a continuación del número–de–nivel, todas las demás cláusulas que integran el formato pueden escribirse en cualquier orden (comparar líneas 002100 a 002140 con líneas 002150 a 002190).

```
002150                02  BETA    PIC X(56)
002160                            COLUMN  39
002170                            LINE    6
002180                            VALUE 'ESTADISTICA DE OPERARIOS NACIDOS EN C
002190-                           'ADA PROVINCIA'.
002200    01  GRUPO2               TYPE PAGE HEADING.
002210                02  CAB1     LINE 7.
002220                      03  CAB11   COLUMN 105  PIC X(12)  VALUE 'PAGINA NUM. '.
002230                      03  CAB12   COLUMN 117  PIC 9(4)   SOURCE PAGE-COUNTER.
002240                02  CAB2     LINE 9
002245                            COLUMN  30  PIC X(75)  VALUE ALL '='.
```

Cada grupo de informe puede estar constituido por campos compuestos y elementales o únicamente por campos elementales.

Según el número–de–nivel y las cláusulas que se utilicen del formato general, servirá para distinguir los grupos de informe de los campos compuestos y campos elementales.

Efectivamente:

a) Un grupo de informe debe llevar obligatoriamente el número de nivel 01 y la cláusula TYPE, y, opcionalmente, las cláusulas LINE, NEXT GROUP y USAGE.

b) Un campo compuesto, formado por varios campos elementales (CAB1, línea 002210), puede llevar cualquier número de nivel del 02 al 49 y la cláusula LINE. Opcionalmente la USAGE.

c) Un campo elemental puede llevar cualquier número de nivel del 02 al 49 y, por lo menos, deben estar presentes una cláusula COLUMN, una PICTURE y una de las cláusulas SOURCE, SUM o VALUE (CAB11 y CAB12, líneas 002220 y 002230). Si el campo elemental es el único que se va a imprimir en toda una línea debe llevar la cláusula LINE (ALFA, línea 002100). El resto de las cláusulas del formato general es opcional su empleo.

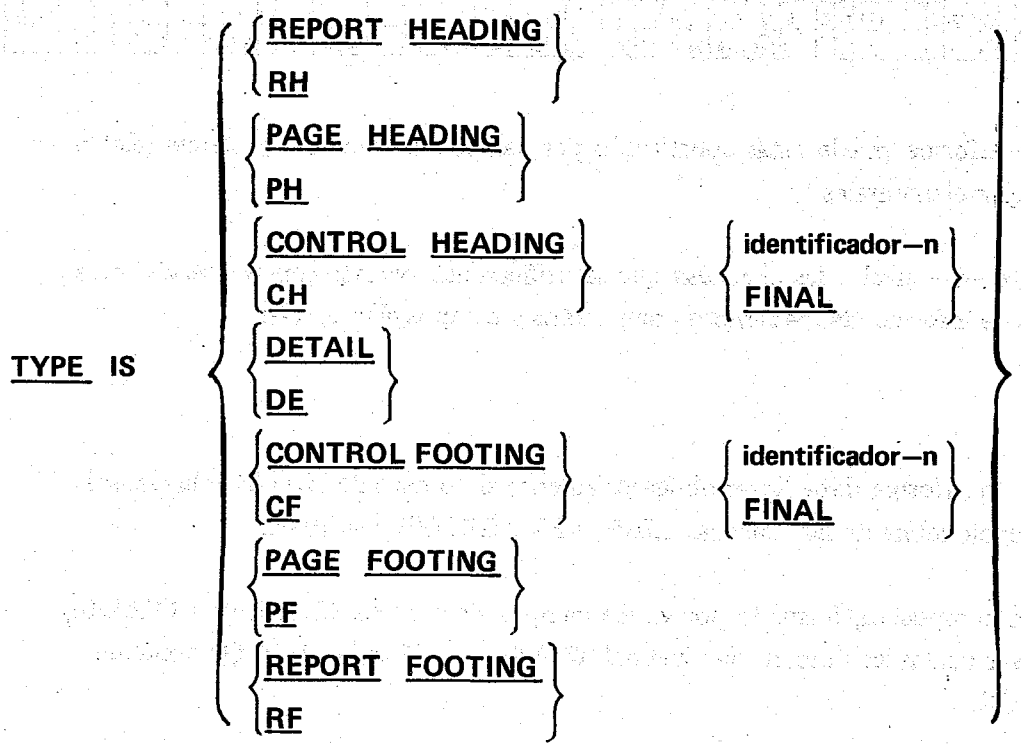
d) Si el grupo de informe está formado únicamente por un campo elemental (PIEPAG, línea 004020 y 004030) se utiliza conjuntamente cuanto se describe en los apartados a), b) y c).

Las cláusulas USAGE, BLANK WHEN ZERO, JUSTIFIED, PICTURE y VALUE tienen el mismo empleo que el especificado al hacer su descripción en la DATA DIVISION.

2.6.2.1.– Cláusula TYPE

FUNCION: Indicar el momento del proceso en que ha de imprimirse cada grupo de informe.

FORMATO:



Puede especificarse indistintamente cada grupo de informe por su nombre completo: REPORT HEADING,o por su forma abreviada RH.

Identificador–n significa que puede especificarse uno o varios identificadores para actuar de CONTROLES.

Dentro de la REPORT SECTION puede describirse un solo grupo de informe de cada uno de los tipos siguientes que se indican en el formato: REPORT HEADING, PAGE HEADING, DETAIL, CONTROL HEADING FINAL, CONTROL FOOTING FINAL, PAGE FOOTING y REPORT FOOTING, y un grupo CONTROL HEADING y CONTROL FOOTING por cada identificador (campo de control) que se especifique dentro de la cláusula CONTROL de la entrada de descripción de informe.

2.6.2.2.– Cláusula COLUMN

FUNCION: Indicar el número de la columna donde debe comenzar la escritura del contenido de un campo elemental dentro de cada línea que se haya de imprimir.

FORMATO:

COLUMN NUMBER IS entero

Entero es la posición donde se colocará el carácter extremo izquierdo del campo elemental.

Si en la descripción de un campo elemental se omite esta cláusula no se imprime el contenido de dicho campo.

2.6.2.3.- Cláusula SOURCE

FUNCION: Indicar el campo de datos que va a utilizarse como emisor.

FORMATO:

```
SOURCE IS { TALLY          }
          { identificador-1 }
```

Identificador-1 debe ser un campo de datos descrito en la FILE o en la WORKING-STORAGE SECTION.

Así, el especificar SOURCE CODIGO en CAMPO1 (línea 003100) y SOURCE APELLNOM en CAMPO2 (línea 003110), quiere decir que, antes de imprimir una línea de detalle, se moverá automáticamente a CAMPO1 y CAMPO2 el contenido que en ese momento se encuentre en los campos CODIGO (línea 001150) y APELLNOM (línea 001160), pertenecientes al último registro que se haya leído.

Los contadores internos como PAGE-COUNTER, CURRENT-DATE , TALLY, etc., pueden utilizarse en vez de un campo de datos (líneas 002230 y 004080).

CURRENT-DATE, que se especifica como SOURCE dentro de uno de los campos elementales que componen el grupo REPORT FOOTING, es un registro interno en el que se introduce la fecha del día cuando el operador pone en funcionamiento el ordenador.

2.6.2.4.- Cláusula SUM

FUNCION: Definir un contador que se incrementa automáticamente con la cantidad contenida en un campo de datos.

FORMATO:

```
SUM { TALLY          } [ { TALLY          } ] ... [ UPON nombre-de-datos ]
    { identificador-2 }   { identificador-3 }
```

- La cláusula SUM sólo puede especificarse en un campo elemental de un grupo de informe CONTROL FOOTING.

- Identificador–2, identificador–3, etc., deben ser campos de datos elementales de PICTURE numérica, pudiendo estar descritos en la FILE, WORKING–STORAGE, LINKAGE, o en otro grupo CONTROL FOOTING de menor nivel dentro de la REPORT SECTION.

En el programa REPORIMP se describen dos contadores denominados: CON24 (línea 003200) y CONT12 (línea 004010). Ambos contadores se inicializan automáticamente a cero al comienzo del programa, al ejecutar la declaración INITIATE.

El contador CON24 se va incrementando en la cantidad con que se ha inicializado el campo CONTAOPE (línea 001230) cada vez que se imprime una línea de detalle, es decir, se incrementa en una unidad por contener dicho campo la cláusula VALUE 1. Es necesario, especificar UPON LINEADETALLE, por ser CONTAOPE un campo de datos que no figura dentro de una cláusula SOURCE en el grupo de informe DETAIL: LINEADETALLE.

El contador CONT12 va incrementándose con los valores que contiene el contador CON24 en el momento de imprimirse cada CONTROL FOOTING PROVINACIM.

2.6.2.5.– CONTADORES

LINE–COUNTER al igual que PAGE–COUNTER son registros: contadores internos, que crea automáticamente el Escritor de Informes, sin necesidad de hacer su descripción en el programa, cuando el programador especifica la cláusula PAGE LIMIT.

LINE–COUNTER se utiliza para controlar la posición, dentro de cada página del informe, de cada línea de impresión, así como el salto de página.

Se inicializa con el valor cero al ejecutar la declaración INITIATE, y se va incrementando unas veces, restaurándose a cero otras, etc., basándose en los límites que imponen los parámetros de cláusula PAGE LIMIT y los que se especifican en las cláusulas LINE NUMBER y NEXT GROUP. De la combinación de dichos parámetros se irán obteniendo para LINE–COUNTER valores que indicarán el número de la línea, con respecto al comienzo de la página, donde ha de imprimirse cada línea del informe.

PAGE–COUNTER, que se especifica como SOURCE del campo CAB12 (línea 002230), tiene como objeto presentar el número de página en una línea del informe.

La declaración INITIATE inicializa dicho contador con el valor 1 y va incrementándose, automáticamente, en una unidad inmediatamente después de escribir un grupo de informe PAGE FOOTING y antes de escribir un PAGE HEADING. Si se desea inicializar PAGE–COUNTER con un valor distinto de 1, el programador lo puede hacer especificando, por ejemplo, una declaración MOVE 7 TO PAGE–COUNTER a continuación de la declaración INITIATE (línea 004130).

2.6.2.6.- Cláusula LINE

FUNCION: Indicar el número de línea absoluto, con respecto al comienzo de la página, de la línea a imprimir, o el número relativo con respecto a la línea anterior impresa.

FORMATO:

```
LINE NUMBER IS { entero-1          }
               { PLUS entero-2     }
               { NEXT PAGE         }
```

Por cada línea que haya de imprimirse de cada grupo de informe debe especificarse una cláusula LINE.

```
002090     01 GRUPO1 TYPE REPORT HEADING.
002100        02 ALFA  LINE NUMBER 2          <-------
002110                COLUMN NUMBER 30
```

LINE NUMBER IS entero-1 expresa una línea absoluta, e indica la línea fija de la página en la que ha de imprimirse. Por tanto, si LINE-COUNTER al comienzo del programa contiene el valor cero debido a la ejecución de la declaración INITIATE, pasa a contener el valor de entero-1, es decir, el valor 2, por especificar en ALFA (línea 002100) la cláusula LINE NUMBER 2, imprimiendo en esta línea el contenido de dicho campo; a continuación, por especificar en BETA la cláusula LINE NUMBER 6, pasa LINE-COUNTER a contener el valor 6, imprimiéndose el contenido de dicho campo en la línea 6. El siguiente grupo de informe: PAGE HEADING (línea 002200) se imprimirá en las líneas 7 y 9 por la misma razón.

LINE NUMBER PLUS entero-2 indica que hay que incrementar el contenido que exista en LINE-COUNTER en el valor de entero-2 para obtener el número de la siguiente línea a imprimir.

El grupo CONTROL HEADING PROVINACIM, que es el que debe imprimirse a continuación, contiene una LINE NUMBER PLUS 2, es decir, una LINE relativa, en cuyo caso esta cláusula incrementa el valor de LINE-COUNTER en dos unidades, lo cual quiere decir que deberá imprimir el contenido de los campos ZETA1 y ZETA2 en la línea 11.

La siguiente línea a imprimir se haría en la línea 12 con el contenido de LANDA, ya que por especificar LINE PLUS 1 incrementa LINE-COUNTER en una unidad.

El siguiente grupo a imprimir es el de detalle -DETAIL- lo cual se hará en la línea 14 por especificar LINE PLUS 2; a continuación una nueva línea de detalle en la línea 16 por la misma razón anterior; así continúa imprimiendo líneas de detalle hasta que LINE-COUNTER adquiere un valor superior al parámetro contenido en LAST DETAIL (línea 002070), es decir, mayor de 58, por lo que no imprimiría la línea de detalle, y sí el grupo de

informe PAGE FOOTING, el cual, por contener la cláusula LINE 60 (línea 004020), hace que el contador pase a valer 60, imprimiendo en dicha línea el pié de página. Finalizada la página, salta a una nueva, e inicializa automáticamente a cero LINE-COUNTER, comenzando de nuevo la impresión de PAGE HEADING en la línea 7 de dicha página.

Cuando tras la escritura de líneas de detalle se produzca la ruptura del control: PROVINACIM, se debe imprimir el CONTROL FOOTING PROVINACIM, el cual comenzará a imprimirse una línea más abajo de la última línea de detalle impresa, por especificar LINE PLUS 1 (línea 003130). Observando la figura 25, la primera ruptura de control se produce al imprimir la línea de detalle cuyo contenido es: 01234 ROMO MARCOS ANTONIO, por lo que comienza la impresión del grupo CONTROL FOOTING PROVINACIM una línea más abajo. A continuación debe imprimirse un nuevo CONTROL HEADING PROVINACIM, el cual comenzará dos líneas más abajo por haber especificado LINE PLUS 2 (línea 003010).

Cuando se llegue al final del informe se produce la ruptura de control FINAL, en cuyo caso, si LINE-COUNTER contiene el valor 37, imprimiría CONTROL FOOTING PROVINACIM en las líneas 38, 39 y 40, a continuación saltaría tres líneas e imprimiría CONTROL FOOTING FINAL. Por último, introduciría el valor 60 en LINE-COUNTER, por PIEPAG LINE 60 imprimiría el pié de página y en las líneas 61 y 63 imprimiría REPORT FOOTING.

La opción LINE NUMBER IS NEXT PAGE se describe en la pág. 374.

2.6.2.7.- Cláusula GROUP INDICATE

FUNCION: Indicar que el contenido del campo elemental se imprimirá únicamente en la primera aparición del campo después de cualquier ruptura de control, o, sin haber ruptura de control, cuando se comience a imprimir en una nueva página.

FORMATO:

GROUP INDICATE

Esta cláusula sólo puede especificarse dentro de un grupo de informe DETAIL al describir un campo elemental.

EJEMPLO:

El siguiente formato impreso muestra un listado de operarios por orden alfabético de provincias, en el que figura, además, el código y sueldo que perciben. Al final de cada provincia se imprime un CONTROL FOOTING en el que figuran los sueldos totales de dicha provincia y los que se van acumulando de las provincias precedentes.

RELACION DE OPERARIOS POR PROVINCIA Y SUELDO

PAGINA 1

	PROVINCIA	CODIGO	APELLIDOS Y NOMBRE	SUELDO	S. ACUMULADO
	ALAVA	0625	GARCIA GOMEZ LUIS	20.050	
	ALAVA	0832	GARCIA GOMEZ ALFONSO	32.750	
(A)	ALAVA	0912	ALVAREZ PAZ JOSE	29.450	
	ALAVA	1294	SAGASTA PERIS FELIPE	43.086	
			TOTALES	125.336	125.336
	ALBACETE	0097	SANCHEZ GONZALEZ ENRIQUE	29.448	
	ALBACETE	0177	ALBA LOPEZ TOMAS	37.724	
(B)	ALBACETE	7282	SEVILLA GOMEZ ALFREDO	35.789	
			TOTALES	102.961	228.297
	ALICANTE	0047	PEREZ PEREZ PEDRO	50.412	
(C)	ALICANTE	7222	LOPEZ ALVAREZ ANTONIO	28.329	

PAGINA 2

	PROVINCIA	CODIGO	APELLIDOS Y NOMBRE	SUELDO	S. ACUMULADO
(E)	ALICANTE	8142	SUAREZ LOPEZ EUSEBIO	32.333	
	ALICANTE	8237	GIL ANTUNEZ LUIS	42.825	
(D)	ALICANTE	8958	ESTEBAN MARTIN MARTA	48.229	
			TOTALES	202.128	430.425

Si en el grupo de informe DETAIL se hace la descripción siguiente:

```
010  01  LINEA  TYPE DETAIL.
020      02  LINE PLUS 2.
030          03  COLUMN 20   PIC X(16)
040                  SOURCE  PROVINACIM)  GROUP INDICATE.
050          03  COLUMN 37   PIC X(5)   SOURCE CODIGO.
060          03  COLUMN 46   PIC X(35)  SOURCE APELLNOM.
070          03  COLUMN 85   PIC ZZ.ZZ9 SOURCE SUELDO.
```

Por especificar la cláusula GROUP INDICATE se elimina el nombre de las provincias que figuran señaladas en (A) (B) (C) . . .; y cada vez que el impreso salta para imprimir en una nueva página, si el operario a imprimir pertenece a la misma provincia que el último que se ha impreso en la página anterior, imprime el nombre de dicha provincia (E).

2.6.2.8.– Cláusula RESET

FUNCION: Especificar el identificador de CONTROL que en una ruptura de control provoca la restauración a cero del contador SUM especificado en un campo elemental.

FORMATO:

RESET ON { **identificador** / **FINAL** }

Identificador es el nombre de un campo de datos descrito en una cláusula CONTROL de la entrada RD en la descripción de informe.

Cada vez que se produce una ruptura de control, los contadores asociados con el grupo de informe CONTROL FOOTING se restauran automáticamente a cero, a no ser que se especifique una cláusula RESET que indique que la restauración se ha de verificar cuando se produzca una ruptura de control en un nivel superior.

Así, en el ejemplo anterior para poder imprimir los totales que se vayan produciendo en cada una de las provincias habrá que especificar:

```
090  01 TYPE CONTROL FOOTING PROVINACIM.
100     02 LINE PLUS 2.
110        03 COLUMN 64 PIC X(7) VALUE 'TOTALES'.
120        03 CON1 COLUMN 82 PIC Z.ZZZ.ZZ9 SUM SUELDO.
130        03 CON2 COLUMN 98 PIC ZZ.ZZZ.ZZ9
140           SUM CON1 RESET ON FINAL.
```

Cuando se produce una ruptura de control por cambio de provincia, automáticamente se llevan a cabo las siguientes acciones:

- El contenido del contador CON2 se incrementa en el valor que en ese momento contenga CON1.
- Se imprime el contenido del grupo CONTROL FOOTING PROVINACIM.
- Se restaura a cero el contador CON1.

El contador CON2 no se restaura a cero hasta que se imprime el último CONTROL FOOTING PROVINACIM del informe, por haberlo descrito con la opción ON FINAL.

2.7.– DECLARACIONES de la PROCEDURE DIVISION

2.7.1.– Declaración INITIATE

FUNCION: Iniciar el proceso de confección de un informe.

FORMATO:

INITIATE nombre–de–informe–1 [nombre–de–informe–2] . . .

Nombre–de–informe–1, nombre–de–informe–2 . . . son los nombres descritos dentro de una entrada REPORT en la descripción del archivo de informe (línea 001210).

```
001200     FD  ARCHIMPRESO          RECORDING  F
001205                              LABEL RECORD  OMITTED
001210                              REPORT IS  INFORIMPRE.
001220   WORKING-STORAGE SECTION.
```

La declaración INITIATE realiza las funciones siguientes:

- Inicializa a cero todos los contadores que se especifiquen dentro de cada CONTROL FOOTING.
- Inicializa a cero los registros LINE–COUNTER y PRINT–SWITCH.
- Inicializa a 1 el registro PAGE–COUNTER.

No puede ejecutarse una segunda declaración INITIATE para un determinado nombre de informe, si antes no se ha ejecutado una declaración TERMINATE para dicho nombre de informe.

Por no ser función de esta declaración la apertura del archivo de informe, es necesario especificar una declaración OPEN para dicho archivo (línea 004120).

```
004090  PROCEDURE DIVISION.
004100    COMIENZO.
004110              ACCEPT  TABLAPROVIN
004120              OPEN INPUT  ARCIFICHA   OUTPUT ARCHIMPRESO
004130              INITIATE  INFORIMPRE.
004140    LEER.
```

2.7.2.- Declaración GENERATE

FUNCION: Confección del informe.

FORMATO:

GENERATE identificador

Identificador es el nombre de un grupo de informe DETAIL.

Se puede especificar en la PROCEDURE DIVISION tantas declaraciones GENERATE como grupos de informe DETAIL se hayan descrito en la REPORT SECTION. En los dos programas ejemplo sólo se ha utilizado una GENERATE por haberse especificado solamente un grupo de Informe DETAIL: LINEADETALLE.

Una declaración GENERATE realiza automáticamente las siguientes acciones:

- Transfiere los datos a los campos elementales descritos con la cláusula SOURCE.
- Acumula en los contadores SUM el contenido de los identificadores que se hayan especificado.
- Restaura dichos contadores.
- Edita el contenido de los campos elementales de cada grupo de informe.
- Controla PAGE-COUNTER y LINE-COUNTER.
- Ejecuta la rutina especificada en una declaración USE (ver pág. 375) antes de imprimir un determinado grupo de informe.

Las acciones que ejecuta la primera y sucesivas declaraciones GENERATE en la escritura del informe se describen en el apartado 2.3.

Cuando al ejecutar una declaración GENERATE se detecte una ruptura de control, se llevarán a cabo cuantas acciones se especifican en el último párrafo del apartado 2.6.1.1.

2.7.3.- Declaración TERMINATE

FUNCION: Finalizar el proceso de confección del informe.

FORMATO:

TERMINATE nombre-de-informe-1 [nombre-de-informe-2] ...

Nombre-de-informe-1, nombre-de-informe-2, ... son los nombres de los diferentes informes especificados en una entrada RD.

La ejecución de la declaración TERMINATE tiene lugar después de imprimir la última línea de detalle del informe. Efectivamente, en el programa REPORIMP el programador decidió que la ejecución de dicha declaración se lleve a cabo cuando finalice la lectura del archivo ARCHIFICHA, lo cual tiene lugar inmediatamente después de escribir la última línea de detalle del informe. En ese momento:

- Se imprimen todos los grupos de informe CONTROL FOOTING, asociados con el informe desde el nivel inferior hasta FINAL.

- Se imprime el grupo de informe REPORT FOOTING, finalizando la confección del informe.

Una declaración TERMINATE no cierra el archivo de informe, por lo que hay que especificar una declaración CLOSE (línea 004200) para dicho archivo.

```
004180     FIN.
004190         TERMINATE INFORIMPRE
004200         CLOSE  ARCIFICHA   ARCHIMPRESO
004210         STOP RUN.
```

2.8.– PROGRAMA EJEMPLO DE CONFECCION DE UNA NOMINA.

El programa que se muestra en las páginas 364 a la 366 realiza el trabajo de confección de la nómina de la empresa RICOSA, según el formato que se especifica en las figuras 29 a la 34.

Las diferencias con el trabajo que realiza el programa NOMINA (pág. 23) son las siguientes:

- En vez de ser una nómina general, en la que no se tiene en cuenta la factoría a la que pertenece cada operario, ésta es una nómina independiente para cada factoría.

- Al final de cada factoría se presentan los totales acumulados en cada concepto por todos los operarios de la misma.

- Al final del informe se muestra el total general de cada uno de los conceptos que componen la nómina de las seis factorías.

Para realizar este trabajo se supone que las fichas del archivo operarios han sido previamente clasificadas en orden alfabético de factorías.

Examinando la codificación del programa se puede observar que la descripción de archivos y registros es la misma que se hizo para el programa NOMINA, a excepción de ARCHIMPRE cuya descripción se hace en la REPORT SECTION.

Los campos descritos dentro de la WORKING–STORAGE SECTION, a excepción de los denominados: CONTINUA, CAMBIO y NOMBREF, tienen en este programa la misma función que en el programa NOMINA.

```
001010 ID   DIVISION.
001020   PROGRAM-ID. NOMIFACT.
001030   REMARKS. NOMINA DE RICOSA ORDENADA POR FACTORIAS.
001040 ENVIRONMENT DIVISION.
001050   CONFIGURATION SECTION.
001060     SPECIAL-NAMES.
001070         DECIMAL-POINT IS COMMA.
001080   INPUT-OUTPUT SECTION.
001090     FILE-CONTROL.
001100         SELECT ARCHIFICH   ASSIGN SYS005-UR-2540R-S   RESERVE 1.
001110         SELECT ARCHICIN    ASSIGN SYS006-UT-2400-S.
001120         SELECT ARCHIMPRE   ASSIGN SYS007-UR-1403-S    RESERVE 1.
001130 DATA DIVISION.
001140   FILE SECTION.
001150     FD   ARCHIFICH   RECORDING F   LABEL RECORD OMITTED.
001160     01   FICHAOPE.
001170         02   OPERARIO.
001180              03 CODIGO              PIC X(5).
001190              03 APENOMBRE           PIC X(35).
001200              03 FECHANACI           PIC 9(6).
001210              03 FILLER              PIC 99.
001220              03 ESTADO              PIC 9.
001230                 88 SOLTERO          VALUE 1.
001240                 88 CASADO           VALUE 2.
001250                 88 VIUDO            VALUE 3.
002010              03   FILLER            PIC A.
002020              03   NUM-HIJOS         PIC 99.
002030              03   FECHAINGRESO.
002040                   04   DIA           PIC 99.
002050                   04   MES           PIC 99.
002060                   04   ANNO          PIC 99.
002070              03   SUELDO            PIC 9(5).
002080              03   TIPODESCUEN       PIC 99.
002090              03   NUM-HORASEX       PIC 999.
002100         02   FACTORIA.
002110              03   NOMBREFAC         PIC A(6).
002120              03   FILLER            PIC 9(3).
002130         02   FILLER                 PIC X(3).
002140     FD   ARCHICIN    RECORDING F   LABEL RECORD OMITTED
002150                      BLOCK 10 RECORDS.
002160     01   REGISCIN.
002170          02   DATOSCIN             PIC X(80).
002180          02   SUELDOCIN            PIC 9(6).
002190     FD   ARCHIMPRE   LABEL RECORD OMITTED
002200                      REPORT IS IMPRESO.
002210   WORKING-STORAGE SECTION.
002220     77 CONTADOR                     PIC 9(5) VALUE 1.
002230     77 TOTALBRUTO                   PIC 9(6).
002240     77 TRIENIOS                     PIC 9(5).
002250     77 PLUSF                        PIC 9(4).
003010     77 HORASEX                      PIC 9(5).
003020     77 DESCU                        PIC 9(4).
003030     77 NUMTRIENIOS                  PIC 99.
003040     77 TOTALS                       PIC 9(6).
003050     77 CONTINUA                     PIC X(14).
003060     77 CAMBIO                       PIC 9        VALUE 1.
003080     77 NOMBREF                      PIC X(6) VALUE SPACES.
003090     01 FECHA.
003100          02   D                     PIC 99.
003110          02   FILLER                PIC X.
003120          02   M                     PIC 99.
003130          02   FILLER                PIC X.
003140          02   A                     PIC 99.
```

```
003150     REPORT SECTION.
003160       RD  IMPRESO
003170           CONTROLS    ARE    FINAL     NOMBREFAC
003180           PAGE        60     LINES
003190           HEADING 2
003200           FIRST DETAIL 8
003210           LAST  DETAIL 54
003220           FOOTING 55.
004010       01  TYPE  REPORT  HEADING     NEXT GROUP    NEXT PAGE.
004020           02  LINE 9          COLUMN  50      PIC X(30)
004030                               VALUE 'E M P R E S A          R I C O S A'.
004040           02  LINE 15         COLUMN  52      PIC X(27)
004050                               VALUE 'SECCION   DE   CONTABILIDAD'.
004060           02  LINE 24.
004070               03               COLUMN  37      PIC X(47)     VALUE 'NOMINA DE LA
004080-                                        'S SEIS FACTORIAS CORRESPONDIENTE A '.
004090               03   COLUMN 84  PIC X(8)   SOURCE FECHA.
004100       01  CABE-PAG  TYPE   PAGE HEADING.
004110           02  LINE 2.
004120               03  COLUMN  50  PIC X(22)   VALUE 'NOMINA DE LA FACTORIA '.
004130               03  COLUMN  72  PIC X(6)    SOURCE NOMBREFAC.
004140               03  COLUMN  80  PIC X(14)   SOURCE CONTINUA.
004150           02  LINE 5
004160               COLUMN 7   PIC X(120) VALUE 'CODIGO                APELLIDOS   Y
004170-                '   NOMBRE                        SUELDO     TRIENIOS     HORAS-E
004180-                'X              PLUSF        DESCUENTO        TOTAL'.
004190           02  LINE 6   COLUMN 7    PIC X(120)   VALUE ALL '='.
004200       01  LINEADETALLE   TYPE DE   LINE   PLUS 2.
004210           02  COLUMN   8    PIC X(5)    SOURCE   CODIGO.
004220           02  COLUMN  15    PIC X(35)   SOURCE   APENOMBRE.
004230           02  COLUMN  58    PIC ZZ.ZZ9  SOURCE   SUELDO.
004240           02  COLUMN  71    PIC ZZ.ZZ9  SOURCE   TRIENIOS.
004250           02  COLUMN  84    PIC ZZ.ZZ9  SOURCE   HORASEX.
006010           02  COLUMN  98    PIC Z.ZZ9   SOURCE   PLUSF.
005020           02  COLUMN 110    PIC ZZ.ZZ9  SOURCE   DESCU.
005030           02  COLUMN 121    PIC ZZZ.ZZ9   SOURCE   TOTALS.
005040       01   TYPE  CF   NOMBREFAC     NEXT GROUP NEXT PAGE.
005050           02   LINE   PLUS 1  COLUMN 8    PIC X(120)   VALUE ALL '-'.
005060           02   LINE   PLUS 1.
005070               03   COLUMN    25   PIC X(20)  VALUE 'TOTALES   ACUMULADOS'.
005080               03   CONTA1   COLUMN  54     PIC ZZ.ZZZ.ZZ9   SUM SUELDO.
005090               03   CONTA2   COLUMN  68     PIC  Z.ZZZ.ZZ9   SUM TRIENIOS.
005100               03   CONTA3   COLUMN  81     PIC  Z.ZZZ.ZZ9   SUM HORASEX.
005110               03   CONTA4   COLUMN  94     PIC  Z.ZZZ.ZZ9   SUM PLUSF.
005120               03   CONTA5   COLUMN 107     PIC  Z.ZZZ.ZZ9   SUM DESCU.
005130               03   CONTA6   COLUMN 117     PIC ZZZ.ZZZ.ZZ9  SUM TOTALS.
005140       01   TYPE  CF  FINAL   LINE NEXT PAGE.
005150           02       COLUMN  14     PIC X(37)
005160                    VALUE 'TOTAL ACUMULADO DE LAS SEIS FACTORIAS'.
005170           02       COLUMN  54     PIC ZZ.ZZZ.ZZ9     SUM CONTA1.
005180           02       COLUMN  68     PIC  Z.ZZZ.ZZ9     SUM CONTA2.
005190           02       COLUMN  81     PIC  Z.ZZZ.ZZ9     SUM CONTA3.
005200           02       COLUMN  94     PIC  Z.ZZZ.ZZ9     SUM CONTA4.
005210           02       COLUMN 107     PIC  Z.ZZZ.ZZ9     SUM CONTA5.
005220           02       COLUMN 119     PIC ZZZ.ZZZ.ZZ9    SUM CONTA6.
005230           02   LINE  PLUS  3.
005240           03       COLUMN  14  PIC X(24)   VALUE 'REGISTROS PROCESADOS = '.
005250           03       COLUMN  38  PIC ZZ..ZZ9     SUM CONTADOR UPON LINEADETALLE.
006010       01   TYPE   PF.
006020           02  LINE 56     COLUMN 8     PIC X(120)   VALUE ALL '*'.
006030           02  LINE 58.
006040               03   COLUMN  105   PIC X(12)   VALUE 'PAGINA NUM.. '.
006050               03   COLUMN  117   PIC 9(4)    SOURCE   PAGE-COUNTER.
006060       01   TYPE   RF    LINE   NEXT PAGE.
006070           02  LINE 28   COLUMN 50     PIC X(35)   VALUE ALL '*'.
006080           02  LINE 30   COLUMN 50     PIC X(35)   VALUE '*   F I N      DEL
006090-                                      'I N F O R M E   *'.
006100           02  LINE 32   COLUMN 50     PIC X(35)   VALUE ALL '*'.
```

```
000611 0  PROCEDURE DIVISION.
000612 0    DECLARATIVES.
000613 0      INFORME SECTION.
000614 0          USE BEFORE REPORTING  CABE-PAG.
000614 5      ALFA.
000615 0          IF  CAMBIO = 1     GO TO  BETA
000615 5          ELSE  GO TO  NOIMPRE.
000616 0      BETA.
000616 5          IF    NOMBREFAC = NOMBREF
000616 7                MOVE  '(CONTINUACION)' TO  CONTINUA
000616 8                GO TO  SAL
000617 0          ELSE MOVE  NOMBREFAC  TO  NOMBREF
000618 0               MOVE  SPACES  TO  CONTINUA
000618 5               GO TO  SAL.
000618 6      NOIMPRE.
000619 0          MOVE 1  TO  PRINT-SWITCH.
000620 0      SAL.        EXIT.
000621 0    END DECLARATIVES.
000622 0      PROCESO.
000623 0          ACCEPT  FECHA
000624 0          OPEN    INPUT  ARCHIFICH
000624 5                  OUTPUT  ARCHICIN    ARCHIMPRE.
000625 0          INITIATE  IMPRESO.
000701 0      LEER.
000701 5          READ ARCHIFICH  AT END    GO TO  FIN.
000702 0      CALCULO-TRIENIOS.
000703 0          IF  MES  OF  FECHAINGRESO < M
000704 0                COMPUTE  NUMTRIENIOS = (A - ANNO OF FECHAINGRESO) / 3
000704 5          ELSE
000705 0              COMPUTE NUMTRIENIOS = (A - ANNO OF FECHAINGRESO - 1) / 3.
000706 0          COMPUTE  TRIENIOS = NUMTRIENIOS * 1000.
000707 0      CALCULO-PLUSF.
000708 0          IF  CASADO    COMPUTE  PLUSF = (500 + NUM-HIJOS * 250)
000709 0          ELSE
000710 0                IF  VIUDO  COMPUTE PLUSF = NUMHIJOS * 250
000711 0                ELSE    MOVE 0  TO  PLUSF.
000711 5      CALCULO-NETO.
000712 0          MULTIPLY  NUM-HORASEX  BY  150  GIVING  HORASEX
000713 0          COMPUTE TOTALBRUTO = SUELDO + TRIENIOS + PLUSF + HORASEX
000714 0          COMPUTE  DESCU = (TOTALBRUTO * TIPODESCUEN) / 100
000715 0          SUBTRACT DESCU  FROM  TOTALBRUTO    GIVING  TOTALS.
000715 5      GRABARCINTA.
000716 0          MOVE TOTALS  TO  TOTALSUELDOCIN
000717 0          MOVE FICHAOPE  TO  DATOSCIN.
000718 0          WRITE  REGISCIN.
000719 0      IMPRIMIR.
000719 5          GENERATE  LINEADETALLE
000719 6          GO TO  LEER.
000720 0      FIN.
000720 5          MOVE  ZERO  TO  CAMBIO
000721 0          TERMINATE  IMPRESO
000721 5          CLOSE  ARCHIMPRE   ARCHIFICH
000722 0          STOP  RUN.
```

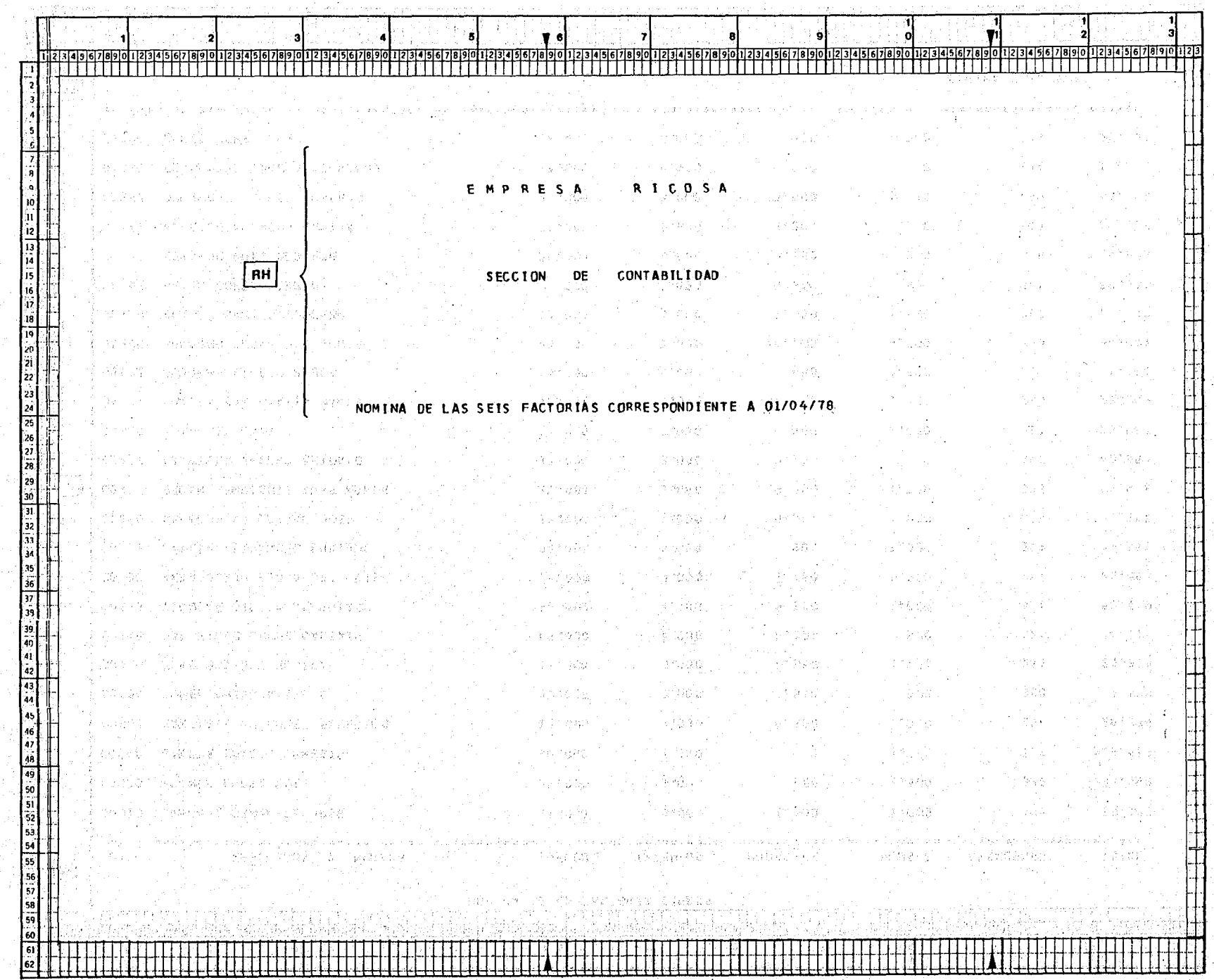
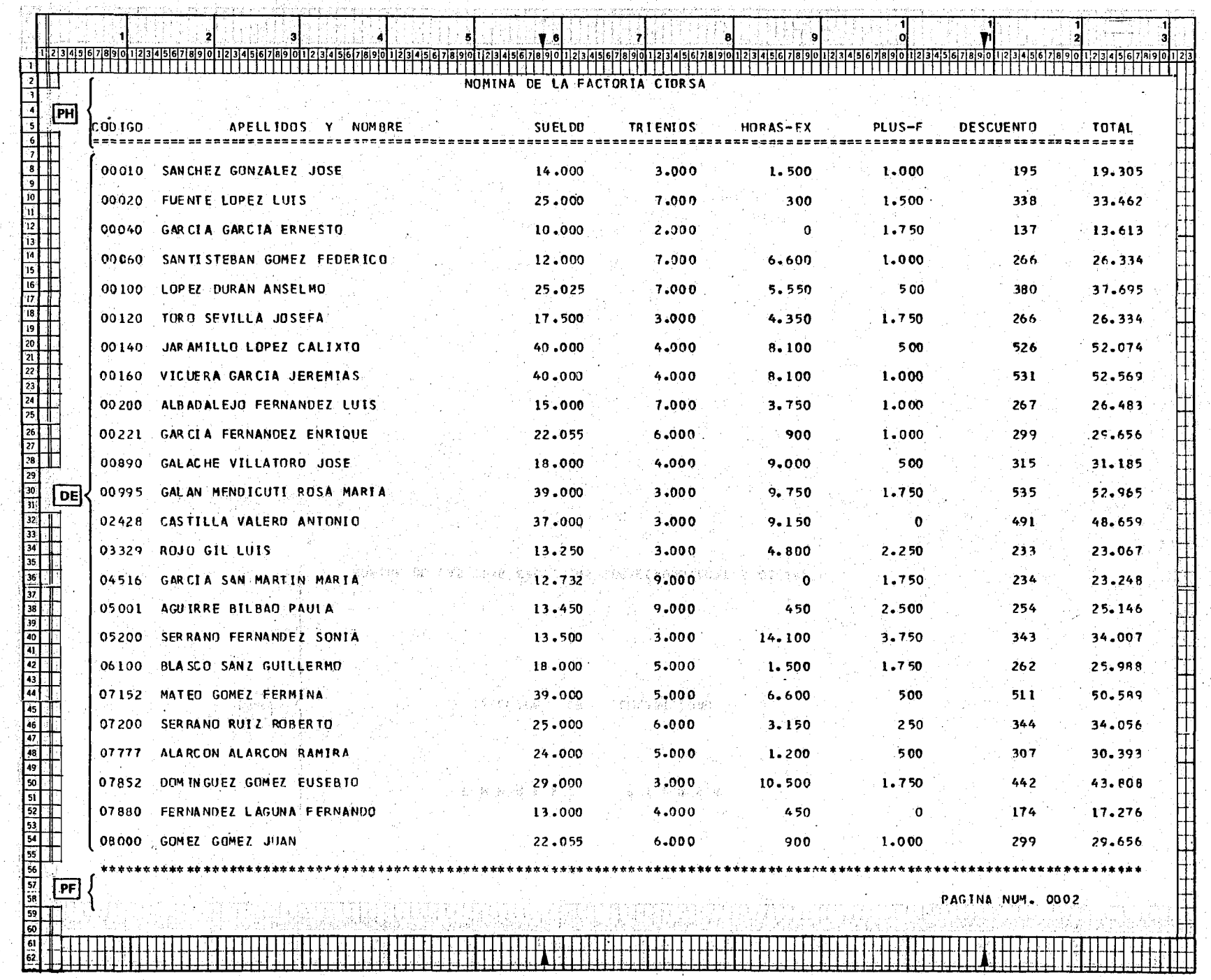

RH
EMPRESA RICOSA
SECCION DE CONTABILIDAD
NOMINA DE LAS SEIS FACTORIAS CORRESPONDIENTE A 01/04/78

FIGURA 29



NOMINA DE LA FACTORIA CIORSA

PH

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX	PLUS-F	DESCUENTO	TOTAL
00010	SANCHEZ GONZALEZ JOSE	14.000	3.000	1.500	1.000	195	19.305
00020	FUENTE LOPEZ LUIS	25.000	7.000	300	1.500	338	33.462
00040	GARCIA GARCIA ERNESTO	10.000	2.000	0	1.750	137	13.613
00060	SANTISTEBAN GOMEZ FEDERICO	12.000	7.000	6.600	1.000	266	26.334
00100	LOPEZ DURAN ANSELMO	25.025	7.000	5.550	500	380	37.695
00120	TORO SEVILLA JOSEFA	17.500	3.000	4.350	1.750	266	26.334
00140	JARAMILLO LOPEZ CALIXTO	40.000	4.000	8.100	500	526	52.074
00160	VICUERA GARCIA JEREMIAS	40.000	4.000	8.100	1.000	531	52.569
00200	ALBADALEJO FERNANDEZ LUIS	15.000	7.000	3.750	1.000	267	26.483
00221	GARCIA FERNANDEZ ENRIQUE	22.055	6.000	900	1.000	299	29.656
00890	GALACHE VILLATORO JOSE	18.000	4.000	9.000	500	315	31.185
00995	GALAN MENDICUTI ROSA MARIA	39.000	3.000	9.750	1.750	535	52.965
02428	CASTILLA VALERO ANTONIO	37.000	3.000	9.150	0	491	48.659
03329	ROJO GIL LUIS	13.250	3.000	4.800	2.250	233	23.067
04516	GARCIA SAN MARTIN MARIA	12.732	9.000	0	1.750	234	23.248
05001	AGUIRRE BILBAO PAULA	13.450	9.000	450	2.500	254	25.146
05200	SERRANO FERNANDEZ SONIA	13.500	3.000	14.100	3.750	343	34.007
06100	BLASCO SANZ GUILLERMO	18.000	5.000	1.500	1.750	262	25.988
07152	MATEO GOMEZ FERMINA	39.000	5.000	6.600	500	511	50.589
07200	SERRANO RUIZ ROBERTO	25.000	6.000	3.150	250	344	34.056
07777	ALARCON ALARCON RAMIRA	24.000	5.000	1.200	500	307	30.393
07852	DOMINGUEZ GOMEZ EUSEBIO	29.000	3.000	10.500	1.750	442	43.808
07880	FERNANDEZ LAGUNA FERNANDO	13.000	4.000	450	0	174	17.276
08000	GOMEZ GOMEZ JUAN	22.055	6.000	900	1.000	299	29.656

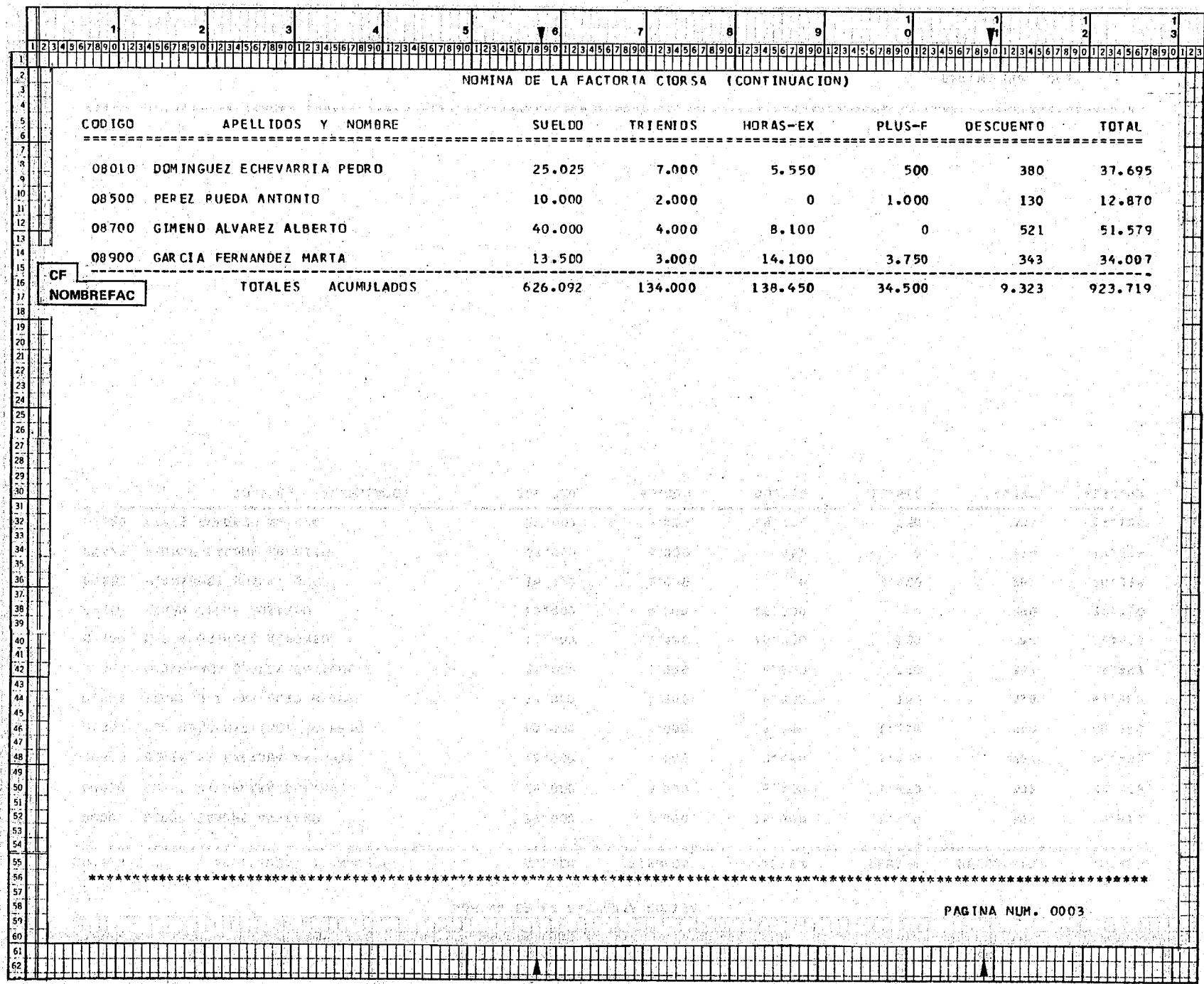
DE

****

PF

PAGINA NUM. 0002

FIGURA 30



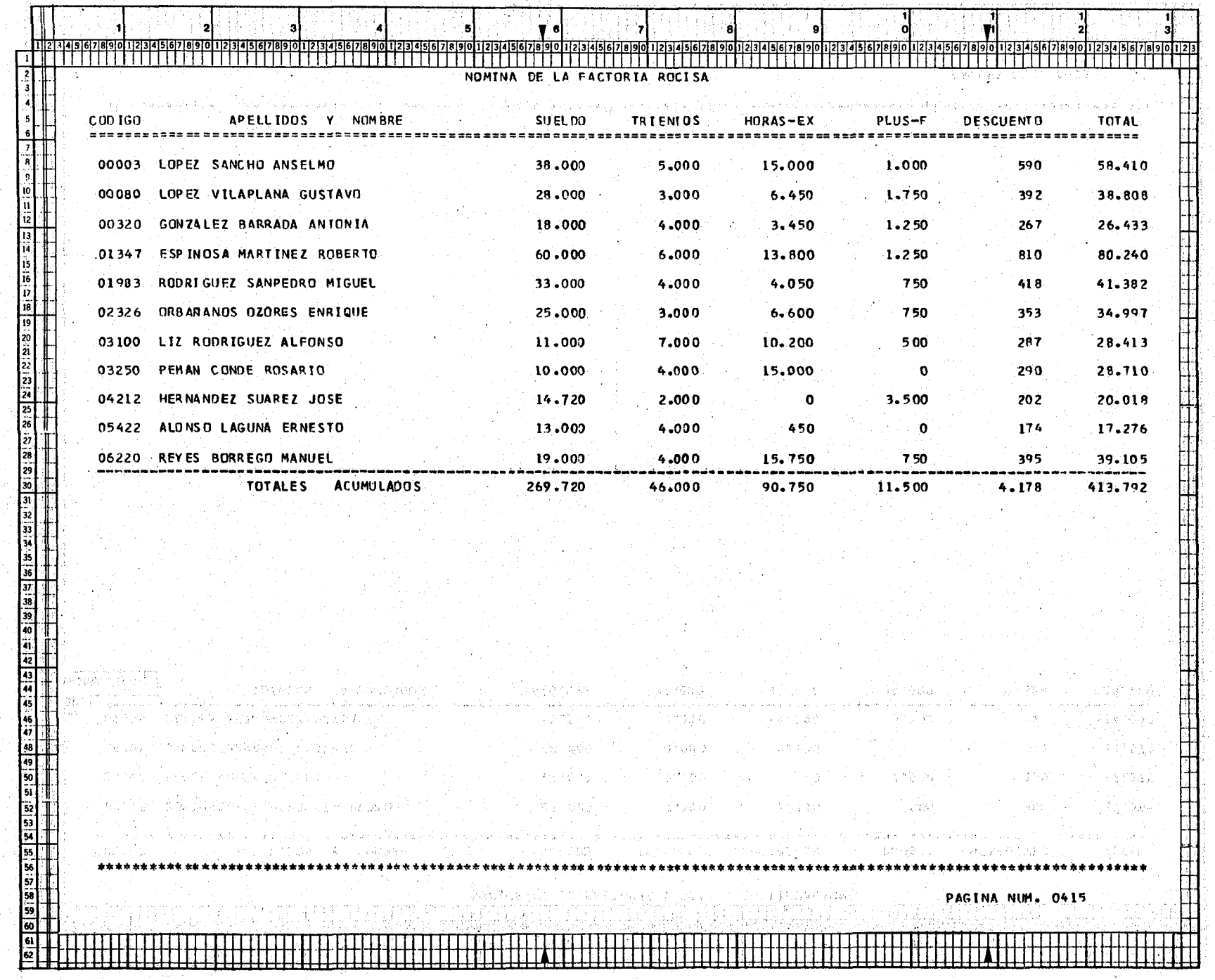
NOMINA DE LA FACTORIA CIORSA (CONTINUACION)

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX	PLUS-F	DESCUENTO	TOTAL
08010	DOMINGUEZ ECHEVARRIA PEDRO	25.025	7.000	5.550	500	380	37.695
08500	PEREZ PUEDA ANTONIO	10.000	2.000	0	1.000	130	12.870
08700	GIMENO ALVAREZ ALBERTO	40.000	4.000	8.100	0	521	51.579
08900	GARCIA FERNANDEZ MARTA	13.500	3.000	14.100	3.750	343	34.007
CF NOMBREFAC	TOTALES ACUMULADOS	626.092	134.000	138.450	34.500	9.323	923.719

****

PAGINA NUM. 0003

FIGURA 31



NOMINA DE LA FACTORIA ROCISA

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX	PLUS-F	DESCUENTO	TOTAL
00003	LOPEZ SANCHO ANSELMO	38.000	5.000	15.000	1.000	590	58.410
00080	LOPEZ VILAPLANA GUSTAVO	28.000	3.000	6.450	1.750	392	38.808
00320	GONZALEZ BARRADA ANTONIA	18.000	4.000	3.450	1.250	267	26.433
01347	ESPINOSA MARTINEZ ROBERTO	60.000	6.000	13.800	1.250	810	80.240
01983	RODRIGUEZ SANPEDRO MIGUEL	33.000	4.000	4.050	750	418	41.382
02326	ORBAÑANOS OZORES ENRIQUE	25.000	3.000	6.600	750	353	34.997
03100	LIZ RODRIGUEZ ALFONSO	11.000	7.000	10.200	500	287	28.413
03250	PEMAN CONDE ROSARIO	10.000	4.000	15.000	0	290	28.710
04212	HERNANDEZ SUAREZ JOSE	14.720	2.000	0	3.500	202	20.018
05422	ALONSO LAGUNA ERNESTO	13.000	4.000	450	0	174	17.276
06220	REYES BORREGO MANUEL	19.000	4.000	15.750	750	395	39.105
	TOTALES ACUMULADOS	269.720	46.000	90.750	11.500	4.178	413.792

****

PAGINA NUM. 0415

FIGURA 32

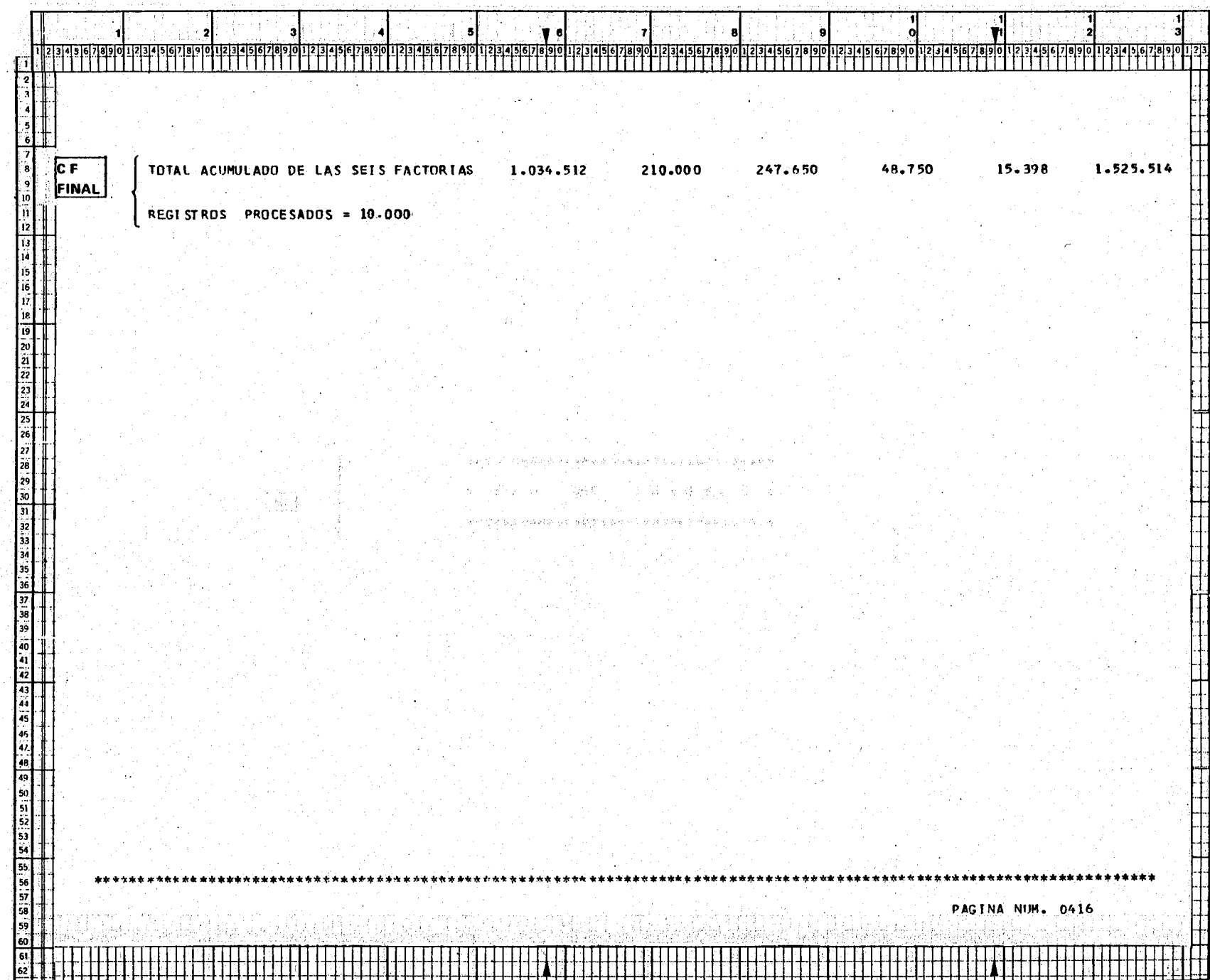

CF FINAL
TOTAL ACUMULADO DE LAS SEIS FACTORIAS 1.034.512 210.000 247.650 48.750 15.398 1.525.514
REGISTROS PROCESADOS = 10.000
PAGINA NUM. 0416

FIGURA 33

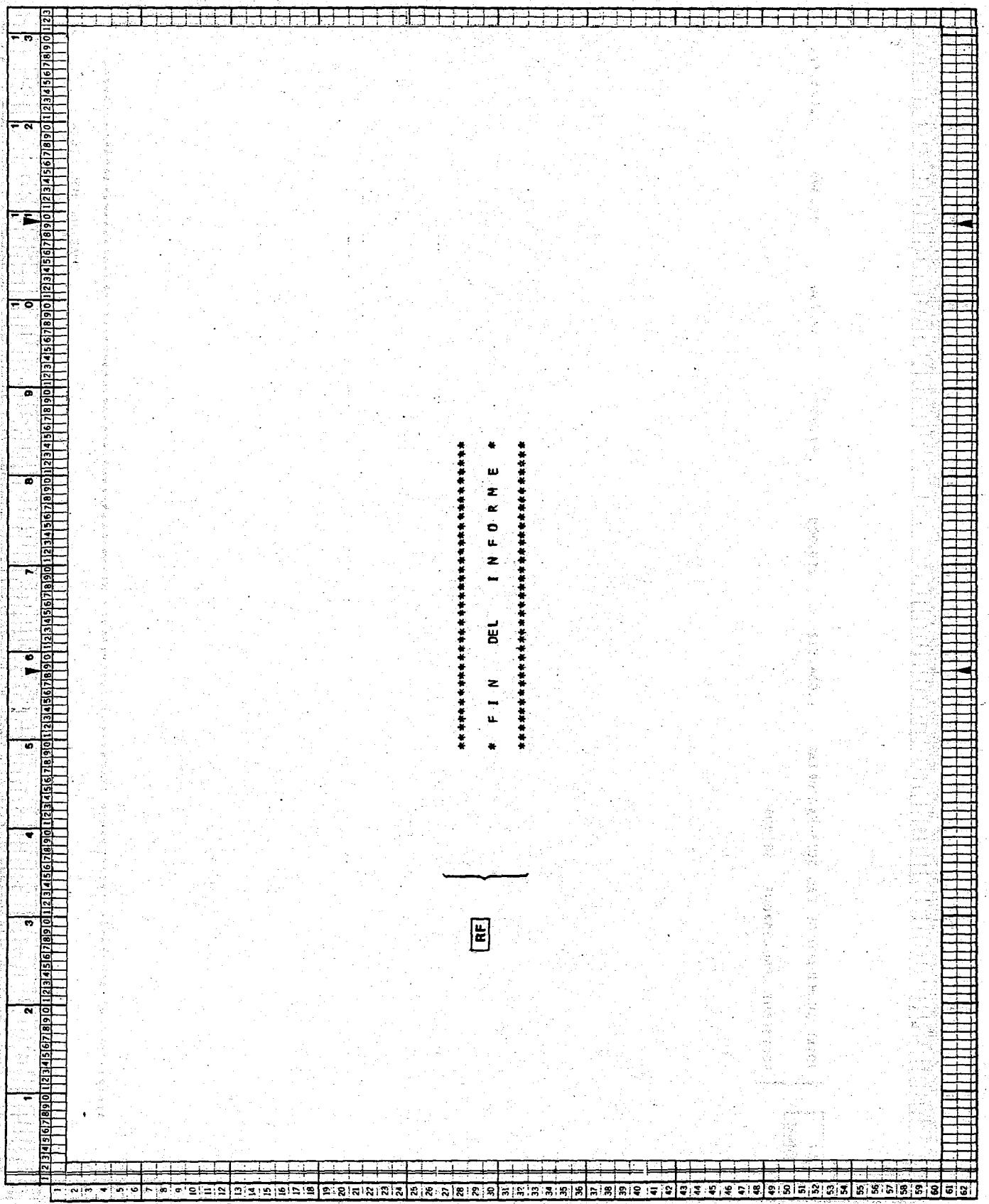


FIGURA 34

2.8.1.- Cláusula NEXT GROUP

FUNCION: Indicar el espaciado o el cambio de página que debe seguir a la última línea que se ha impreso del grupo de informe donde se ha especificado.

FORMATO:

```
NEXT GROUP IS  { entero-1
               { PLUS entero-2
               { NEXT PAGE
```

Esta cláusula no puede especificarse en grupos de informe: PAGE HEADING, PAGE FOOTING y REPORT FOOTING.

Especificando la opción NEXT PAGE en un grupo de informe REPORT HEADING (línea 004010), una vez finalizada la impresión de las líneas que componen dicho grupo, saltaría a la página siguiente donde imprimiría el siguiente grupo de informe. Por tanto, REPORT HEADING se imprimirá sólo en una página (fig. 29), haciéndolo el siguiente grupo de informe que por orden corresponde: PAGE HEADING, al comienzo de la página siguiente.

También se ha especificado esta opción en el grupo de informe CONTROL FOOTING NOMBREFAC (línea 005040), por lo que cada vez que haya una ruptura de control al llevar a cabo la impresión del último operario de cada factoría, se imprimirá el contenido de las dos líneas que componen dicho grupo (CF), no pudiéndose imprimir en esta página ninguna otra línea de detalle ni grupos CONTROL, debiéndolo hacer en la siguiente página.

Estas acciones se realizan así debido a que la opción NEXT PAGE fija a cero el valor de LINE-COUNTER cada vez que imprime la última línea del grupo de informe en que se especifique, lo que obliga a la impresora a saltar a la página siguiente previa impresión del grupo PAGE FOOTING si el programador lo ha descrito en la REPORT SECTION.

Si se especifica NEXT GROUP entero-1 en un grupo de informe, entero-1 indica el número de línea absoluta en que se fija LINE-COUNTER después de imprimir la última línea de dicho grupo, y por tanto será donde se imprima la primera línea del siguiente grupo de informe.

Si se especifica NEXT GROUP PLUS entero-2 en un grupo de informe, entero-2 indica un número de línea relativo que incrementa LINE-COUNTER en ese valor después de imprimir la última línea de dicho grupo.

Así, si en el grupo de informe DETAIL (línea 004200) se hubiera especificado NEXT GROUP PLUS 3 en vez de LINE PLUS 2, haría que una vez finalizada la escritura de una línea de detalle saltara tres líneas para efectuar la escritura de la siguiente línea de detalle.

En un grupo de informe REPORT HEADING no puede especificarse una cláusula NEXT GROUP absoluta o relativa que fije a LINE–COUNTER en un valor mayor a FIRST DETAIL entero–3 menos uno. Es decir, no sería correcto especificar en REPORT HEADING (línea 004010): NEXT GROUP 30, por ser el valor 30 superior al valor 7 (valor de entero–3 menos 1).

2.8.2.– Cláusula LINE (Continuación)

Otra cláusula utilizada en la REPORT SECTION es: LINE con la opción NEXT PAGE (línea 006060), su función es indicar que el grupo de informe ha de imprimirse en la página siguiente. Por tanto, por haberse especificado esta opción en el grupo REPORT FOOTING su contenido se imprimirá sólo en una página (fig. 34).

También se ha especificado esta opción en el grupo CONTROL FOOTING FINAL (línea 005140), pero ello no es necesario, ya que al imprimir el último CONTROL FOOTING NOMBREFAC correspondiente a la factoría ROCISA, el siguiente grupo a imprimir sería el citado CONTROL FOOTING FINAL, debiéndolo hacer en la página siguiente por así ordenarlo la cláusula NEXT GROUP NEXT PAGE de CONTROL FOOTING NOMBREFAC.

- LINE NUMBER IS NEXT PAGE sólo puede especificarse dentro de grupos: DETAIL, CONTROL HEADING, CONTROL FOOTING y REPORT FOOTING.
- Dentro de un grupo de informe una LINE NUMBER PLUS entero–2 no debe preceder a una LINE NUMBER entero–1.
- Las cláusulas LINE NUMBER entero–1 que se especifiquen dentro de un grupo de informe deben figurar en orden ascendente de valores (líneas 004020, 004040, 004060)

2.8.3.– Cláusula SUM (Continuación)

Los contadores o acumuladores que se especifican dentro del grupo CONTROL FOOTING NOMBREFAC: CONTA1, CONTA2, etc., se inicializan a cero al comienzo de la ejecución de programa, y van acumulando el contenido de los campos de datos que figuran a continuación de la cláusula SUM: SUELDO, TRIENIOS, etc., respectivamente, cada vez que se acaba de imprimir un grupo de informe DETAIL, por estar en este grupo descritos dichos campos con una cláusula SOURCE. Cada vez que se produzca una ruptura de control por cambiar NOMBREFAC, se imprime el contenido de dichos contadores y se restauran a cero para ir de nuevo acumulando el contenido del SUELDO, TRIENIOS, etc., de los operarios de la siguiente factoría.

En CONTROL FOOTING FINAL figuran otra serie de contadores (líneas 005170 a 005220) para los que el programador no ha especificado un nombre por no ser necesario. Su función es ir acumulando la cantidad que contengan los contadores: CONTA1, CONTA2, etc. en el momento antes de imprimir el grupo CONTROL FOOTING NOMBREFAC, es decir, por haber seis factorías se modificarán seis veces los valores de los contadores de CONTROL FOOTING FINAL. Existe una excepción en el contador que se describe en la línea 005250, el cual por especificar UPON LINEADETALLE,se va incrementando en el VALUE del campo CONTADOR (línea 002220) cada vez que se imprime el grupo DETAIL denominado LINEADETALLE.

Los nombres de los contadores CONTA1, CONTA2, etc., es necesario especificarlos en CONTROL FOOTING NOMBREFAC para poder definir los contadores que componen el CONTROL FINAL.

2.8.4.- Sentencia USE

FUNCION: Especificar una serie de declaraciones o procedimientos que deben ejecutarse inmediatamente antes de imprimir un determinado grupo de informe.

FORMATO:

USE BEFORE REPORTING nombre-de-datos

Nombre-de-datos es el nombre con que se ha descrito a un tipo de grupo de informe. En el programa NOMIFACT el tipo de grupo de informe es PAGE HEADING y el nombre con que se ha descrito es CABE-PAG (línea 004100).

El tipo de grupo de informe puede ser cualquiera,excepto un grupo DETAIL.

En el programa NOMIFACT se ha especificado la sentencia USE junto con los procedimientos que la componen: ALFA, BETA,NOIMPRE y SAL, los cuales se ejecutarán inmediatamente antes de imprimir el grupo de informe PAGE HEADING.

Por tanto, antes de imprimir el grupo de informe PAGE HEADING o,lo que es lo mismo, los encabezamientos de cada página, prueba por medio de la declaración IF del procedimiento ALFA (línea 006150) si el campo CAMBIO, descrito en línea 003060, contiene el valor 1, lo cual es cierto por haberse inicializado con dicho valor,y seguirá siendo cierto mientras en la PROCEDURE DIVISION no haya una declaración que modifique el valor de dicho campo, por lo que mientras el valor del campo CAMBIO sea igual a 1, ejecutará las declaraciones del

procedimiento BETA. En este procedimiento comprueba (declaración IF) si el nombre de la factoría a la que pertenece el operario que ha de imprimir en la primera línea de detalle de la nueva página es el mismo que la del operario que imprimió en la última línea de detalle de la página anterior, caso afirmativo, mueve al campo CONTINUA el contenido: (CONTINUACION) con lo que al realizar la impresión de dicho grupo de informe aparecerá: NOMINA DE LA FACTORIA... (CONTINUACION) (ver fig. 31); caso contrario, mueve blancos al campo CONTINUA, por lo que se imprimirán blancos en las 14 posiciones que ocupa dicho campo. La declaración MOVE NOMBREFAC TO NOMBREF es necesaria para guardar en este último campo el nombre de la nueva factoría, a fin de realizar nuevas comparaciones con la citada IF.

A la vista del formato previsto para el informe, se observa que las acciones a realizar, una vez que se imprima la última línea de detalle, son: imprimir el último grupo CONTROL FOOTING NOMBREFAC, PAGE FOOTING, saltar a una nueva página, donde no se imprimirá el grupo PAGE HEADING y sí el contenido del grupo CONTROL FOOTING FINAL, y en la página siguiente el grupo REPORT FOOTING.

Todas estas acciones se llevan a cabo de la forma siguiente:

Cuando se ha procesado el último registro del archivo de entrada –ARCHIFICH– pasa a leer la marca de fin de archivo, con lo que se cumple la condición AT END y obliga a bifurcar al procedimiento FIN (línea 007200) en el cual se mueve el valor cero al campo CAMBIO, y a continuación se ejecuta la declaración TERMINATE. Esta declaración hace que se impriman los grupos de informe CONTROL FOOTING NOMBREFAC y PAGE FOOTING, salte a una nueva página y, antes de imprimir un nuevo grupo PAGE HEADING, pase el control a la sentencia USE, donde ejecuta el procedimiento ALFA, en el cual el resultado de la comparación: IF CAMBIO = 1 es falso, por lo que bifurca al procedimiento NOIMPRE, donde mueve 1 al registro que genera el Escritor de Informes: PRINT–SWITCH. Este registro, que se inicializa automáticamente a cero al ejecutar la declaración INITIATE, cuando su contenido sea 1 a partir de ese momento suprime la impresión del grupo de informe especificado en la sentencia USE, es decir, CABE–PAG; por tanto, cuando después de ejecutar la citada declaración (línea 006190) se devuelve el control a la declaración TERMINATE para completar sus funciones, obliga a saltar a la página siguiente pero no imprime PAGE HEADING en esa página, ni en las sucesivas, imprimiendo, sin embargo, el CONTROL FOOTING FINAL y el REPORT FOOTING. Finaliza el programa con la ejecución de las declaraciones CLOSE ARCHIMPRE y STOP RUN.

Si en un programa quiere hacerse referencia, dentro de la PROCEDURE DIVISION, a uno o varios procedimientos de los que componen la sentencia USE, sólo puede llevarse a cabo por medio de una declaración PERFORM.

3.- EJERCICIOS

1.— La Sección de Estadística de la Empresa Ricosa solicita un informe impreso cuyo formato se muestra en las figuras 35 a la 39.

El archivo de entrada de donde se toman los datos para ser impresos en el informe, es el utilizado en la confección de la nómina (pág. 23). Las fichas que componen dicho archivo están ordenadas por el número del código de cada operario.

El trabajo a realizar consta de las siguientes fases:

- Leer cada registro del archivo de entrada y seleccionar aquellos registros en los que figuren operarios con hijos y sueldo no inferior a 20.000 pts.

- Clasificar los registros seleccionados en orden ascendente por el código de la provincia, y dentro de cada provincia por orden alfabético de apellidos y nombre. Caso de que en una provincia existieran dos operarios con los mismos apellidos y nombre se imprimirán en orden descendente por el código asignado.

- Imprimir los registros una vez clasificados según el formato citado.

Codificar el programa que pueda llevar a cabo dicho trabajo, utilizando las características REPORT y SORT.

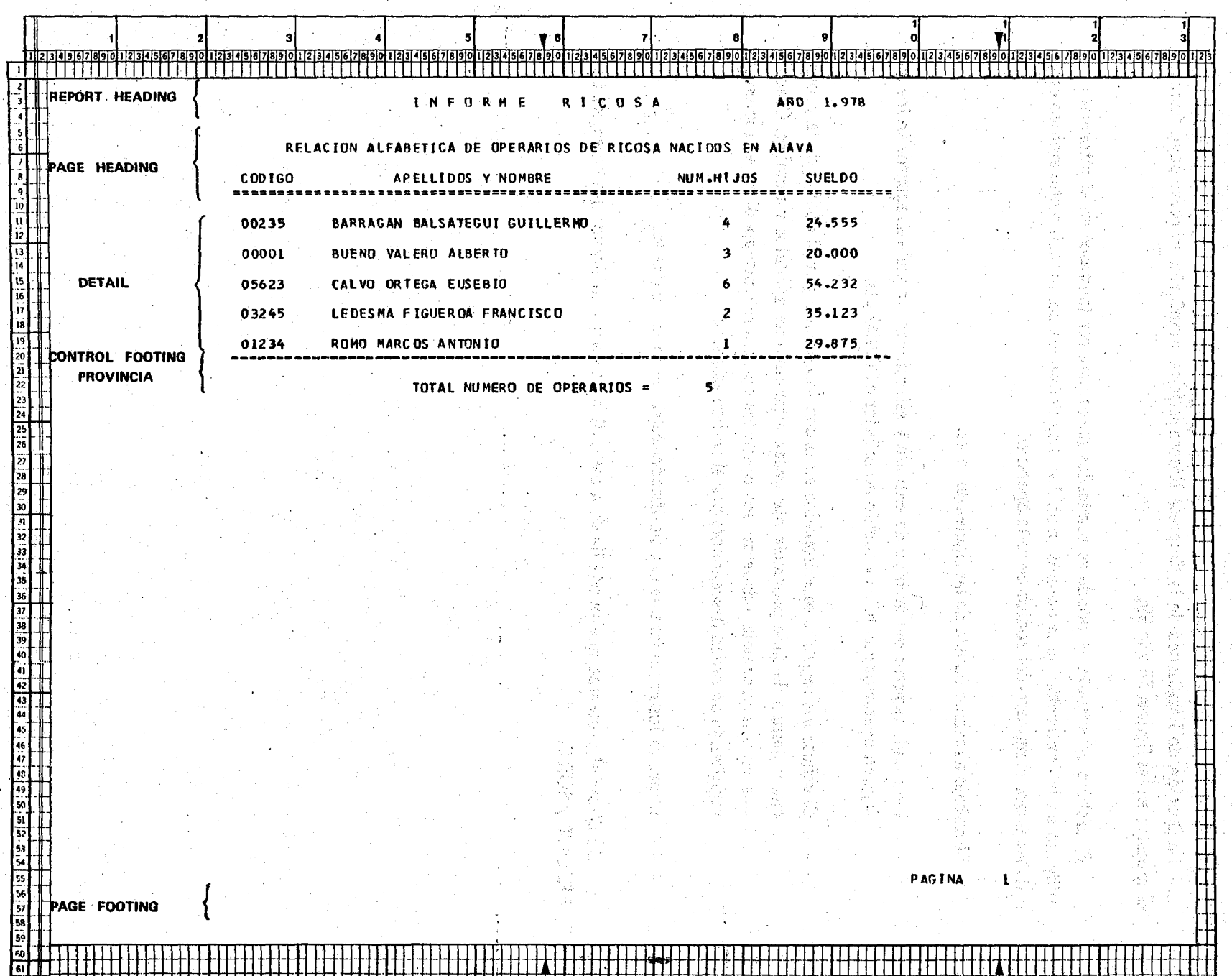
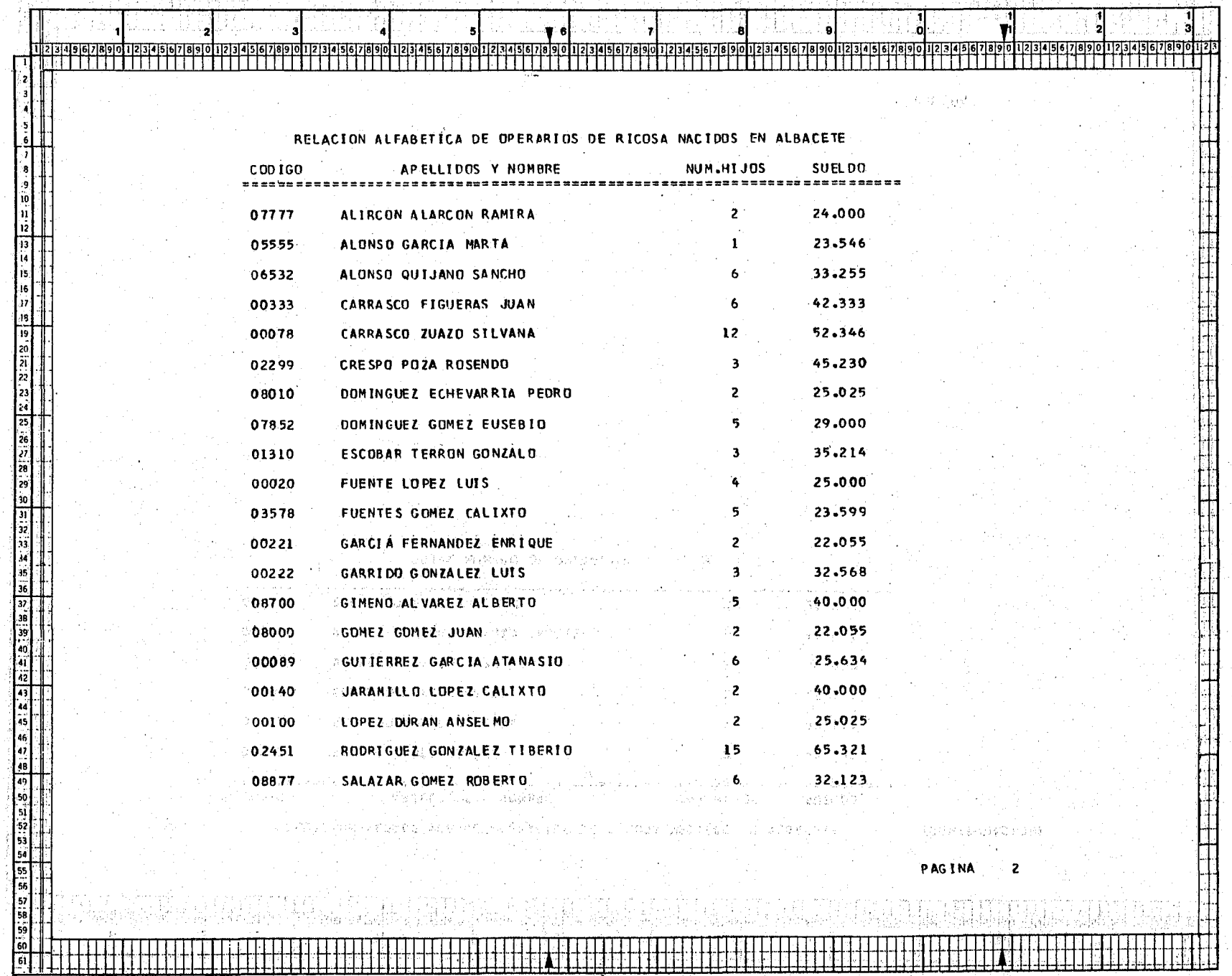

REPORT HEADING
I N F O R M E R I C O S A AÑO 1.978
PAGE HEADING
RELACION ALFABETICA DE OPERARIOS DE RICOSA NACIDOS EN ALAVA
CODIGO APELLIDOS Y NOMBRE NUM.HIJOS SUELDO
DETAIL
00235 BARRAGAN BALSATEGUI GUILLERMO 4 24.555
00001 BUENO VALERO ALBERTO 3 20.000
05623 CALVO ORTEGA EUSEBIO 6 54.232
03245 LEDESMA FIGUEROA FRANCISCO 2 35.123
01234 ROMO MARCOS ANTONIO 1 29.875
CONTROL FOOTING PROVINCIA
TOTAL NUMERO DE OPERARIOS = 5
PAGINA 1
PAGE FOOTING

FIGURA 35

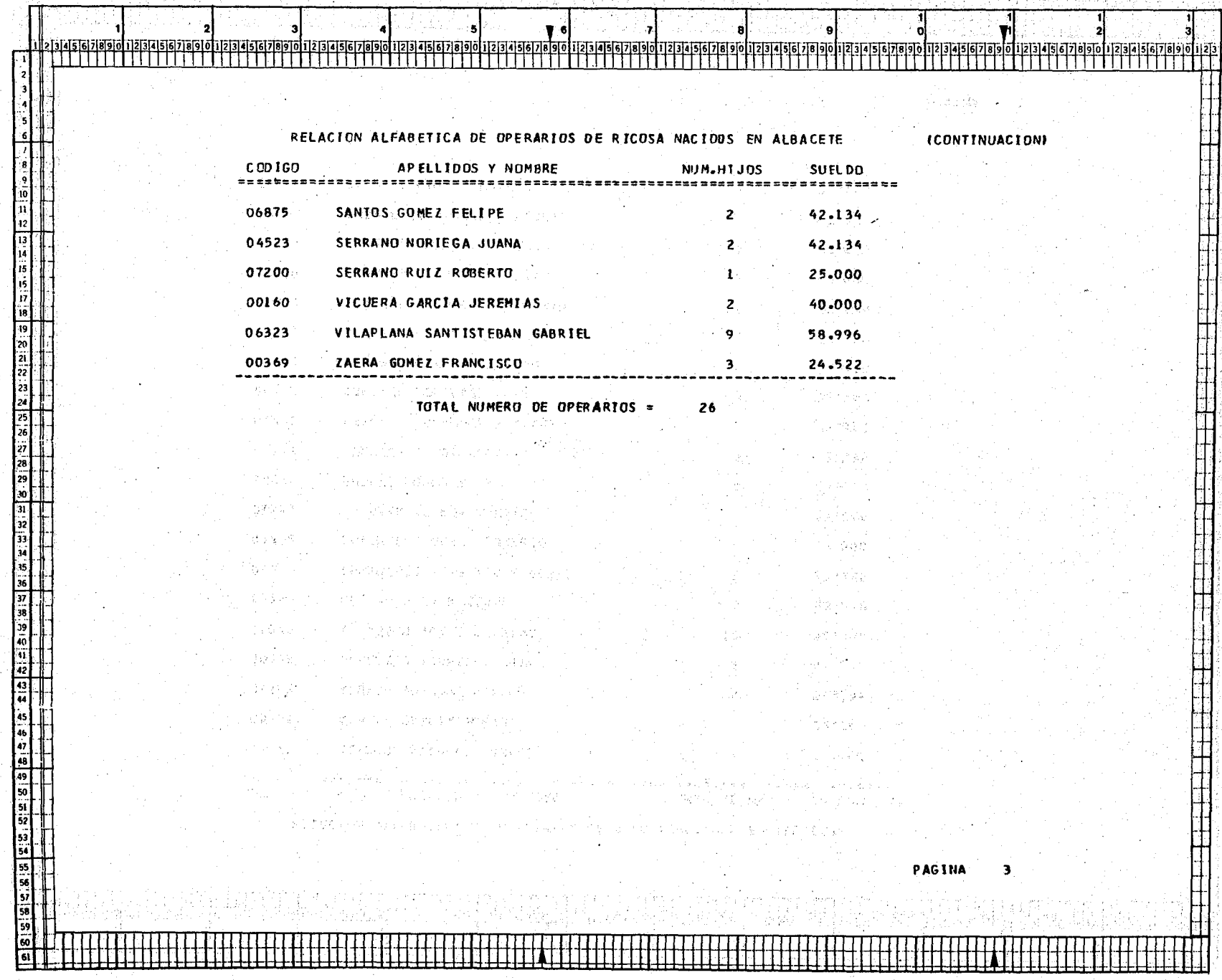


RELACION ALFABETICA DE OPERARIOS DE RICOSA NACIDOS EN ALBACETE

CODIGO	APELLIDOS Y NOMBRE	NUM.HIJOS	SUELDO
07777	ALIRCON ALARCON RAMIRA	2	24.000
05555	ALONSO GARCIA MARTA	1	23.546
06532	ALONSO QUIJANO SANCHO	6	33.255
00333	CARRASCO FIGUERAS JUAN	6	42.333
00078	CARRASCO ZUAZO SILVANA	12	52.346
02299	CRESPO POZA ROSENDO	3	45.230
08010	DOMINGUEZ ECHEVARRIA PEDRO	2	25.025
07852	DOMINGUEZ GOMEZ EUSEBIO	5	29.000
01310	ESCOBAR TERRON GONZALO	3	35.214
00020	FUENTE LOPEZ LUIS	4	25.000
03578	FUENTES GOMEZ CALIXTO	5	23.599
00221	GARCIA FERNANDEZ ENRIQUE	2	22.055
00222	GARRIDO GONZALEZ LUIS	3	32.568
08700	GIMENO ALVAREZ ALBERTO	5	40.000
08000	GOMEZ GOMEZ JUAN	2	22.055
00089	GUTIERREZ GARCIA ATANASIO	6	25.634
00140	JARAMILLO LOPEZ CALIXTO	2	40.000
00100	LOPEZ DURAN ANSELMO	2	25.025
02451	RODRIGUEZ GONZALEZ TIBERIO	15	65.321
08877	SALAZAR GOMEZ ROBERTO	6	32.123

PAGINA 2

FIGURA 36



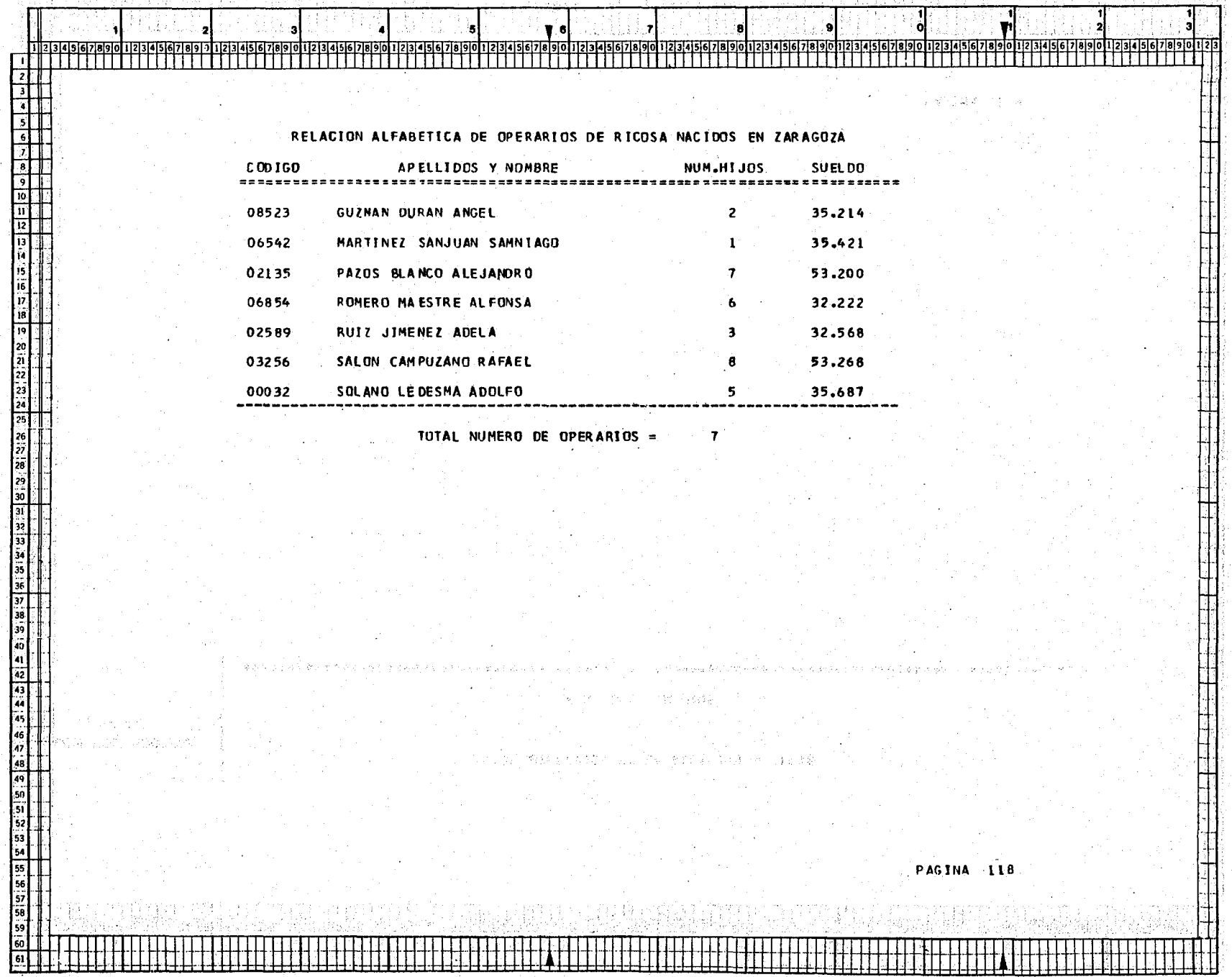
RELACION ALFABETICA DE OPERARIOS DE RICOSA NACIDOS EN ALBACETE (CONTINUACION)

CODIGO	APELLIDOS Y NOMBRE	NUM.HIJOS	SUELDO
06875	SANTOS GOMEZ FELIPE	2	42.134
04523	SERRANO NORIEGA JUANA	2	42.134
07200	SERRANO RUIZ ROBERTO	1	25.000
00160	VICUERA GARCIA JEREMIAS	2	40.000
06323	VILAPLANA SANTISTEBAN GABRIEL	9	58.996
00369	ZAERA GOMEZ FRANCISCO	3	24.522

TOTAL NUMERO DE OPERARIOS = 26

PAGINA 3

FIGURA 37



RELACION ALFABETICA DE OPERARIOS DE RICOSA NACIDOS EN ZARAGOZA

CODIGO	APELLIDOS Y NOMBRE	NUM.HIJOS	SUELDO
08523	GUZNAN DURAN ANGEL	2	35.214
06542	MARTINEZ SANJUAN SAMNIAGO	1	35.421
02135	PAZOS BLANCO ALEJANDRO	7	53.200
06854	ROMERO MAESTRE ALFONSA	6	32.222
02589	RUIZ JIMENEZ ADELA	3	32.568
03256	SALON CAMPUZANO RAFAEL	8	53.268
00032	SOLANO LEDESMA ADOLFO	5	35.687

TOTAL NUMERO DE OPERARIOS = 7

PAGINA 118

FIGURA 38

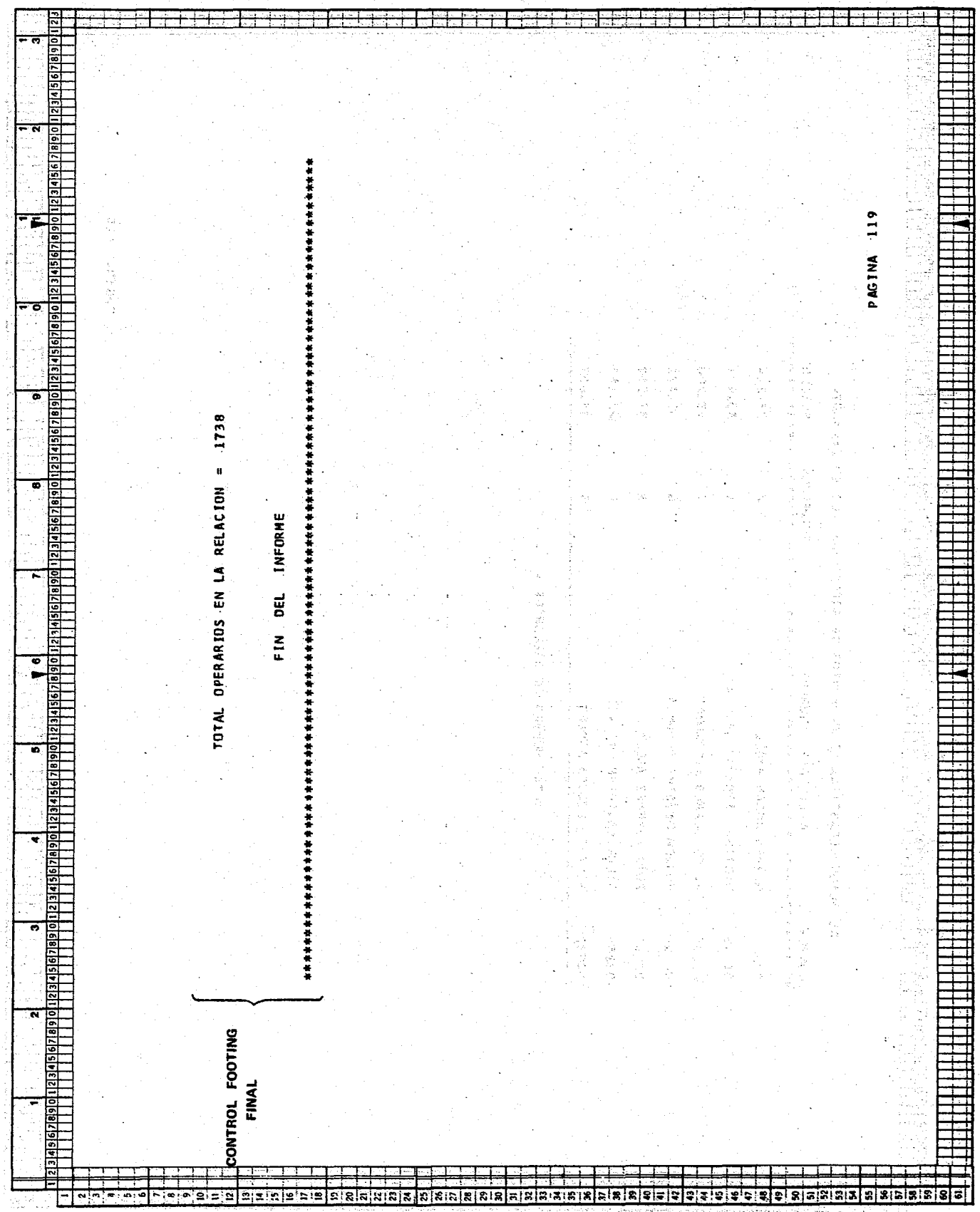

CONTROL FOOTING FINAL
TOTAL OPERARIOS EN LA RELACION = 1738
FIN DEL INFORME
****
PAGINA 119

FIGURA 39

CAPITULO IX

COMUNICACION ENTRE PROGRAMAS

1.- INTRODUCCION

En este capítulo se van a estudiar las características del lenguaje con las que poder establecer la comunicación entre varios programas, a fin de permitir que un programa escrito en COBOL pueda ordenar la ejecución de otros programas COBOL (28).

Hasta ahora se ha visto cómo cada trabajo era realizado por un solo programa, por ejemplo: la confección de la nómina la realizaba el programa NOMINA (pág. 23); el desglose de la moneda era realizado por el programa DESGLOSE (pág. 249), etc. Cada uno de estos programas era, normalmente, codificado y depurado por un solo programador.

Sin embargo, hay trabajos que requieren la confección de programas de tal extensión que sería casi imposible poder llevarla a cabo un solo programador en el plazo solicitado.

Una solución a este problema consistiría en que entre varios programadores confeccionaran el programa correspondiente, codificando cada uno una parte del mismo. Sin embargo, no es en la confección de un programa donde normalmente se emplea más tiempo, sino en su puesta a punto (compilación, depuración, etc.), incrementándose dicho tiempo considerablemente si el programa es extenso. Por tanto, llevando a cabo esta solución, probablemente no sería posible realizar el trabajo en el plazo solicitado.

Las especificaciones del lenguaje COBOL que se van a estudiar permiten una solución más rápida, que consiste en: dividir el trabajo en varias partes y asignar cada una a un programador, el cual confeccionaría UN PROGRAMA con el que realizaría dicha parte del trabajo.

Esta solución tiene la ventaja de que cada programa puede ser puesto a punto por el programador que lo confeccionó, y en forma simultánea con el resto de los programas. Esto supone un ahorro considerable de tiempo, a la vez que existe menor dificultad en la depuración de los programas por ser más reducidos. Una vez puestos a punto los diferentes programas se unirían para formar una unidad de proceso con la que llevar a cabo el mismo.

---

(28) No se estudia, aunque las características del lenguaje lo permiten, la comunicación entre un programa COBOL y programas escritos en otros lenguajes: FORTRAN, PL/1, etc.

Los programas que forman la unidad de proceso, no se procesan de forma independiente, es decir, de forma que los resultados obtenidos en la ejecución de un programa no influyan en los resultados que se obtengan con la ejecución del resto de los programas, sino que en el proceso de cada registro participan los programas que se han integrado dentro de dicha unidad de proceso. Para lo cual, es preciso incluir dentro de cada programa declaraciones de ENLACE a fin de indicar al Sistema el momento en que cada programa debe suspender su proceso para pasar el control a otro programa, y así sucesivamente hasta finalizar el proceso del registro. Esta operación se repetiría con todos los registros del archivo a procesar.

Hay también programas en los que, con el paso del tiempo y por haber variado las condiciones existentes cuando se confeccionaron, sería necesario introducir pequeñas modificaciones o confeccionar un programa nuevo. Sin embargo, si las nuevas condiciones son temporales, podría ser conveniente no introducir modificación alguna en el programa por si, pasadas esas circunstancias, hubiera que volver a ejecutar éste con su codificación inicial. En este caso, sería conveniente confeccionar un programa que ejecutado en unión con el primitivo realice la misma función que si se hubieran introducido las modificaciones en este último.

Los dos ejemplos que se describen a lo largo de este Capítulo sirven para aclarar conceptos a la vez que permiten seguir con facilidad el estudio de las especificaciones del lenguaje que hacen posible la solución de los problemas citados.

EJEMPLO 1º

La Sección de Contabilidad de la Empresa devuelve al Centro de Proceso de Datos la nómina confeccionada con fecha 01/04/78, por no estar conforme con los resultados de la misma.

Realizado el oportuno estudio en el citado Centro, se ha observado la existencia de los siguientes errores y omisiones:

a) En las fichas perforadas que contienen los datos de cada operario, figura el sueldo de los operarios comprendidos entre los códigos 00003 y 01000 ambos inclusive, así como el 01458 con 1.000 pts. más de las que les corresponde, mientras que el número de horas extras de estos operarios figura con 3 horas extras menos de las trabajadas.

b) No se ha tenido en cuenta que la paga del mes de Abril es extraordinaria, y por tanto, no se ha incluído la escritura de la cabecera "PAGA EXTRAORDINARIA", ni se ha incrementado el plus familiar en 6.000 pts. para los casados, 5.000 pts. para los viudos/as y 3.000 pts. para los solteros/as (según está convenido para este tipo de pagas).

Entre las diversas soluciones, se opta por aquélla que se cree puede ser la más rápida, y se adapta más a la problemática expuesta. Dicha solución consiste en:

- No modificar el contenido de las fichas erróneamente perforadas.
- No modificar ninguna declaración del programa NOMINA.

- Confeccionar un programa: NOMIN, que formando una unidad de proceso con el programa NOMINA, sirva para corregir los resultados obtenidos por éste e incorpore las omisiones citadas.

- Incluir en el programa NOMINA únicamente las declaraciones de enlace con el programa NOMIN.

SOLUCION :

PROGRAMA NOMINA

```
001010  IDENTIFICATION DIVISION.
001020    PROGRAM-ID.   NOMINA.
001030    AUTHOR. PROGRAMADOR C. DE LA FUENTE.
.....  ..........................................
001110  ENVIRONMENT DIVISION.
.....  ..........................................
002040  DATA DIVISION.
.....  ..........................................
006010  PROCEDURE DIVISION.
006020          OPEN   INPUT   ARCHIVO-OPERARIOS,
006030                 OUTPUT CINTA-NOMINA  ARCHI-IMPRESO,
006040          DISPLAY 'MARQUE LA FECHA DE HOY SEGUN FORMATO  DD/MM/AA'
006050                  UPON CONSOLE,
006060          ACCEPT  FECHA  FROM CONSOLE,
006070          STOP 'COMPROBAR SI EL PAPEL DE LA IMPRESORA ES DE FORMATO ESP
006080-                'ECIAL PARA NOMINA'.
006090  ESCRIBIR-1.
006100          WRITE IMPRESO FROM CABECERA-1 AFTER ADVANCING PAG-NUEVA
006110          DISPLAY '                                                  '
006120-                 '                                                  '.
006121* ************************************************
006122*LLAMADA AL PRIMER PUNTO DE ENTRADA DEL PROGRAMA NOMIN
006123*
006124          CALL  'CABECERA'.
006125*
006126*PUNTO DE RETORNO DESDE EL PROGRAMA NOMIN
006127* ************************************************
006130  ESCRIBIR-2.
006140          ADD  1  TO  NUM-HOJA
006150          WRITE  IMPRESO FROM CABECERA-2  AFTER 2
.....  ..........................................
007110  CALCULO-PLUSF.
007120          IF CASADO  COMPUTE PLUSF = (500 + NUM-HIJOS * 250)
007130             ELSE
007140                  IF VIUDO  COMPUTE PLUSF = NUM-HIJOS * 250
007150                     ELSE  MOVE 0 TO PLUSF.
007151* ************************************************
007152* **LLAMADA AL SEGUNDO PUNTO DE ENTRADA DEL PROGRAMA NOMIN**
007153*
007154          CALL 'PLUSF-ERRORFICHA' USING PLUSF, FICHA-OPERARIO.
007155*
007156* ****PUNTO DE RETORNO DESDE EL PROGRAMA NOMIN****
007157* ************************************************
007180  10.  MULTIPLY  NUM-HORASEX  BY  150  GIVING  HORASEX
007190          COMPUTE TOTALBRUTO =  SUELDO + TRIENIOS + PLUSF + HORASEX
007200          COMPUTE DESCU = (TOTALBRUTO * TIPODESCUEN) / 100
007210          SUBTRACT  DESCU FROM  TOTALBRUTO  GIVING  TOTALS.
008010  ALFA-1.
008020          MOVE CODIGO  TO CODIGO-S
.....  ..........................................
008180          GO TO LEER.
```

PROGRAMA NOMIN

PROGRAMA: NOMIN — SISTEMA — INSTRUCCIONES PARA PERFORACION — HOJA

PROGRAMADOR: N. RODRIGUEZ VERAO — FECHA: 5-8-1984 — SIGNO — PERFORACIONES

```
001010  IDENTIFICATION DIVISION.
001020      PROGRAM-ID. NOMIN.
001030 *    PROGRAMA PARA IMPRIMIR UNA CABECERA, RECTIFICAR RESULTADOS DE
001040 *       LA NOMINA DEBIDO A UN ERROR DE PERFORACION EN EL SUELDO Y HO-
001050 *       RAS EXTRAS ASIGNADAS A OPERARIOS CON CODIGO > 00002 Y < 01001
001060 *       Y EL 01458, E INCREMENTAR PLUS FAMILIAR A TODOS LOS OPERARIOS.
001070  ENVIRONMENT DIVISION.
001080  DATA DIVISION.
001090    WORKING-STORAGE SECTION.
001100      01  CABECERA-3.
001110          02  FILLER    PIC X(53) VALUE  SPACES.
001120          02  TITULO    PIC X(26) VALUE  '*** PAGA EXTRAORDINARIA ***'.
001130          02  FILLER    PIC X(54) VALUE  SPACES.
001140    LINKAGE SECTION.
001150      77  PLUSF                       PIC 9(4).
001160      01  FICHA-OPE.
001170          02  CODI                    PIC X(5).
001180          02  FILLER                  PIC X(43).
001190          02  ESTADO                  PIC 9.
001200              88  SOLTERO             VALUE 1.
001210              88  CASADO              VALUE 2.
001220              88  VIUDO               VALUE 3.
001230          02  FILLER                  PIC X(9).
001240          02  SUELDO                  PIC 9(5).
001250          02  FILLER                  PIC XX.
002010          02  NUM-HORASEX             PIC 999.
002020  PROCEDURE DIVISION.
002030 *  PUNTO DE ENTRADA, DECLARACIONES NECESARIAS PARA IMPRIMIR UNA
002040 *                NUEVA CABECERA Y PUNTO DE RETORNO AL PROGRAMA NOMINA.
002050  ENTRADA1.
002060              ENTRY 'CABECERA'.
002070                  DISPLAY CABECERA-3.
002080  RETORNO1.
002090              EXIT PROGRAM.
002100 *  PUNTO DE ENTRADA, DECLARACIONES NECESARIAS PARA INCREMENTAR EL
002110 *                PLUS FAMILIAR, CORREGIR EL VALOR DEL SUELDO Y NUMERO DE
002120 *                HORAS EXTRAS, Y PUNTO DE RETORNO AL PROGRAMA NOMINA.
002130  ENTRADA2.
002140              ENTRY 'PLUSF-ERRORFICHA' USING PLUSF, FICHA-OPE.
002150                  IF CASADO ADD 6000 TO PLUSF
002160                  ELSE
002170                      IF VIUDO ADD 5000 TO PLUSF
002180                      ELSE
002190                          ADD 3000 TO PLUSF.
002200                  IF CODI > 00002 AND < 01001 OR = 01458
002210                      SUBTRACT 1000 FROM SUELDO
002220                      ADD 3 TO NUM-HORASEX
002230                  ELSE NEXT SENTENCE.
002240  RETORNO2.
002250              EXIT PROGRAM.
```

DESCRIPCION DEL PROCESO

Comenzaría el proceso ejecutándose las declaraciones del programa NOMINA a partir de la primera (línea 006020) hasta la declaración CALL 'CABECERA' (línea 006124), en cuyo momento interrumpiría su proceso para pasar el control al programa NOMIN, entrando en él en aquel punto que contenga una declaración ENTRY que se denomine de la misma forma que la declaración CALL que lo llamó, es decir, en ENTRY 'CABECERA' (línea 002060); ejecuta la declaración DISPLAY 'CABECERA' (línea 002070) con la que escribe la cabecera 'PAGA EXTRAORDINARIA', y la declaración EXIT PROGRAM (línea 002090) con la que devuelve el control al programa NOMINA, para continuar su ejecución a partir de la declaración siguiente a la CALL (línea 006140).

Cuando se ejecute la declaración CALL 'PLUSF-ERRORFICHA' (línea 007154) vuelve a transferir el control al programa NOMIN, entrando en el mismo en la declaración ENTRY 'PLUSF-ERRORFICHA' (línea 002140) para ejecutar las declaraciones de la línea 002150 a la 002230, las cuales sirven: para incrementar el PLUS de cada operario en una determinada cantidad según su estado civil, y rectificar el valor del campo SUELDO decrementándolo en 1.000 pts. a determinados operarios. Realizadas estas operaciones y cuando ejecute la declaración EXIT (línea 002250) devuelve de nuevo el control al programa NOMINA entrando en él en la declaración de la línea 007180, donde ejecuta hasta la declaración GO TO LEER, que obliga a bifurcar al procedimiento LEER, a fin de leer un nuevo registro e iniciar su proceso.

Las transferencias de control e interacción entre los dos programas se muestra gráficamente en la figura 40.

Los resultados de la ejecución de ambos programas se muestran en la figura 41, en la que la nómina aparece ya corregida (comparar con los resultados de la pág. 49).

2.- PROGRAMA de LLAMADA y PROGRAMA LLAMADO

Un programa que contiene una declaración CALL, se dice que es un programa que LLAMA a otro. Se le denomina programa de LLAMADA. La finalidad de la llamada es la de transferirle el control durante el proceso.

Al programa que recibe el control se le denomina programa LLAMADO.

El programa llamado deberá contener: una declaración ENTRY a fin de establecer el punto de entrada a donde le es transferido el control, así como una declaración EXIT que devuelva el control al programa que le llamó una vez realizada la función encomendada.

Examinando el programa NOMINA (línea 006124 y 007154), se ve que es un programa que llama, mientras que el programa NOMIN (líneas 002060, 002090, 002140 y 002250) es un programa llamado.

PROGRAMA DE LLAMADA

```
001010      IDENTIFICATION DIVISION.
001020      PROGRAM-ID. NOMINA.
.....       .....
001110      ENVIRONMENT DIVISION.
.....       .....
002040      DATA DIVISION.
.....       .....
006010      PROCEDURE DIVISION.
.....       .....
006124          CALL ' CABECERA '
006140          .....
.....           .....
007154          CALL 'PLUSF-ERRORFICHA ' USING PLUSF, FICHA-OPERARIO.
007180          .....
.....           .....
008180          GO TO LEER.
```

PROGRAMA LLAMADO

```
001010      IDENTIFICATION DIVISION.
001020      PROGRAM-ID. NOMIN.
.....       .....
001070      ENVIRONMENT DIVISION.
001080      DATA DIVISION.
.....       .....
001140          LINKAGE SECTION.
.....           .....
002020      PROCEDURE DIVISION.
.....       .....
002060          ENTRY 'CABECERA '
.....           .....
002090          EXIT PROGRAM.
.....           .....
002140          ENTRY 'PLUSF-ERRORFICHA ' USING PLUSF, FICHA-OPE.
.....           .....
002250          EXIT PROGRAM.
```

FIGURA 40

NOMINA DE LA EMPRESA RICOSA CORRESPONDIENTE A 01/04/78

** PAGA EXTRAORDINARIA **

HOJA NUM. 0001

CODIGO	APELLIDOS Y NOMBRE	SUELDO	TRIENIOS	HORAS-EX.	PLUS-F.	DESCUEN.	TOTAL
00001	BUENO VALERO ALBERTO	20.000	5.000	3.750	3.000	4.445	*27.305
00002	GARCIA GOMEZ MARTA	25.000	4.000	4.500	5.500	5.460	*33.540
00003	LOPEZ SANCHO ANSELMO	37.000	5.000	15.450	6.000	1.269	*62.181
00005	FERNANDEZ GUTIERREZ SONIA	11.000	4.000	19.200	3.000	2.604	*34.596
00010	SANCHEZ GONZALEZ JOSE	13.000	3.000	1.950	6.000	479	*23.471
00025	SERRANO SANTIESTEBAN ROBERTO	28.000	3.000	10.950	7.750	994	*48.706
00028	GOMEZ DE LA IGLESIA MARIA	29.000	6.000	6.450	6.500	959	*46.991
00080	LOPEZ VILAPLANA GUSTAVO	27.000	3.000	6.900	6.750	873	*42.777
00125	VILAGRA RUIPEREZ MERCEDES	18.000	2.000	4.500	3.000	550	*26.950
00162	FERNANDEZ PONCE DE LEON EUSEBIO	14.000	7.000	4.200	3.000	564	*27.636
00320	GONZALEZ BARRADA ANTONIA	17.000	4.000	3.900	5.750	2.145	*28.505
00415	RIOBO MATEO CARLOS	24.000	6.000	3.600	5.250	5.439	*33.411
00585	FREIRE PEREZ ENCARNACION	39.000	2.000	4.050	3.000	6.727	*41.323
00890	GALACHE VILLATORO JOSE	17.000	4.000	9.450	6.500	2.586	*34.364
00995	GALAN MENDICUTI ROSA MARIA	38.000	3.000	10.200	6.750	1.159	*56.791
01000	IRAYZOZ BLASCO JOAQUIN	44.000	4.000	10.350	3.000	8.589	*52.761
01255	ESCOBAR GIMENO NIEVES	39.000	5.000	6.600	3.000	7.504	*46.096
01347	ESPINOSA MARTINEZ ROBERTO	60.000	6.000	13.800	6.250	1.721	*84.329
01458	ECHEVARRIA ALVAREZ EUSTAQUIO	39.000	4.000	8.550	3.000	1.091	*53.459
01983	RODRIGUEZ SANPEDRO MIGUEL	33.000	4.000	4.050	6.750	6.692	*41.108
01995	DOMINGUEZ DE LA GUARDIA FERNANDA	17.500	3.000	4.350	5.000	2.089	*27.761
02100	CABRERA BOSQUE NIEVES	19.800	3.000	4.500	3.000	2.121	*28.179
02325	SOBRINO RUIZ ANDRES	42.000	3.000	4.950	7.750	1.154	*56.546
02326	ORBANANOS OZORES ENRIQUE	25.000	3.000	6.600	5.750	5.649	*34.701

FIGURA 41

3.- ENLACE de PROGRAMAS

Cuando en la realización de un trabajo van a intervenir, dentro de una misma unidad de proceso, dos o más programas, es necesario el ENLACE o encadenamiento debido entre dichos programas, a fin de que puedan trabajar como si fueran un programa único.

El enlace entre programas puede establecerse por medio de las declaraciones:

- **CALL**
- **ENTRY**
- **EXIT**

Estas declaraciones hacen posible que durante el proceso se vaya transfiriendo el control del mismo entre los diferentes programas.

Cada una de estas declaraciones junto con la LINKAGE SECTION (necesaria igualmente para el enlace entre programas) se describen con detalle a continuación.

3.1.- PUNTO de ENTRADA

Se denomina así al lugar del programa donde se especifica una declaración CALL.

3.1.1.- Declaración CALL

FUNCION: Transferir el control de un programa a otro dentro de la misma unidad de proceso.

FORMATO:

CALL literal [**USING** identificador-1 [identificador-2] . . .]

Siempre que un programa llame a otro para transferirle el control durante el proceso, debe especificarse dentro de la PROCEDURE DIVISION una declaración CALL seguida de un literal. Dicho literal debe ser de tipo no numérico, y para su formación sigue las mismas reglas que para el nombre de programa.

Cuando el programa de llamada transfiere el control al programa llamado, el punto de entrada en este último, puede tener lugar:

- En la primera declaración de la PROCEDURE DIVISION, o
- En una declaración distinta de la primera.

En el primer caso, el contenido del literal puede ser el nombre del programa llamado, o cualquier otro nombre.

Efectivamente, teniendo en cuenta que la solución del problema se concibió de tal forma que una vez ejecutada la declaración DISPLAY (línea 006110) del programa NOMINA se pasara el control al programa NOMIN para entrar en la primera declaración de dicho programa a fin de poder imprimir la cabecera '** PAGA EXTRAORDINARIA **', el programador para llevarlo a cabo tiene dos opciones: especificar CALL "CABECERA" o especificar CALL 'NOMIN'.

Si especifica CALL 'CABECERA', es decir, un literal distinto del nombre del programa llamado, obliga a escribir en él una declaración ENTRY con un literal igual al de la declaración CALL: ENTRY 'CABECERA' (línea 002060). Sin embargo, si especifica CALL 'NOMIN' (nombre del programa llamado), el control se transfiere automáticamente al programa llamado entrando en él en la primera declaración que sigue a PROCEDURE DIVISION, no siendo necesaria la declaración ENTRY.

En el segundo caso, el literal que sigue a la declaración CALL en el programa de llamada debe ser igual al literal de la declaración ENTRY en la que se desee tenga lugar la entrada al programa llamado. Ver la igualdad entre el literal 'PLUSF–ERRORFICHA' de la línea 007154 en el programa NOMINA y el literal de la línea 002140 en el programa NOMIN.

Opción USING.

Una de las ventajas que proporciona la comunicación entre programas, es que tanto el programa de llamada como el llamado pueden referirse y utilizar los mismos campos de datos de la memoria.

Esto supone un ahorro de memoria, ya que, aunque haya que hacer en cada programa la descripción de los registros, campos, etc. que han de compartir, sin embargo, la reserva de memoria se produce únicamente por la descripción que se haga en el programa de llamada y en el llamado dentro de la FILE y WORKING–STORAGE SECTION. No se reserva memoria por la descripción de registros, campos, etc. dentro de la LINKAGE SECTION.

Se puede decir, que el programa de llamada pone a disposición o cede al programa llamado una serie de campos de la memoria para ser utilizados por éste.

Compartir los mismos campos de la memoria por diferentes programas obliga a:

- Especificar en el programa que llama la declaración CALL con la opción USING, en la que se enumeren los campos de datos que éste va a ceder al programa llamado, y

- Especificar en el programa llamado la declaración ENTRY con la opción USING, en la que se enumeren igualmente los campos puestos a disposición por el programa que llama.

Los campos de datos que figuren dentro de la opción USING de las declaraciones CALL y ENTRY, pueden denominarse de igual o distinta forma, es indiferente, lo importante es el ORDEN en que se especifiquen y que su descripción sea equivalente, ya que dentro de la memoria se establece entre ellos una correspondencia uno a uno, es decir, a la dirección de la memoria en la que se encuentra el primer campo que sigue a USING en el programa que llama, se corresponde con la dirección del primer campo que sigue a USING en el programa llamado; la dirección del segundo campo del programa que llama se corresponde con la dirección del segundo campo del programa llamado, y así sucesivamente.

Por tanto, es correcto cuando en el programa NOMINA se especifica:

```
007154    CALL 'PLUSF-ERRORFICHA' USING PLUSF, FICHA-OPERARIO.
```

y en el programa NOMIN:

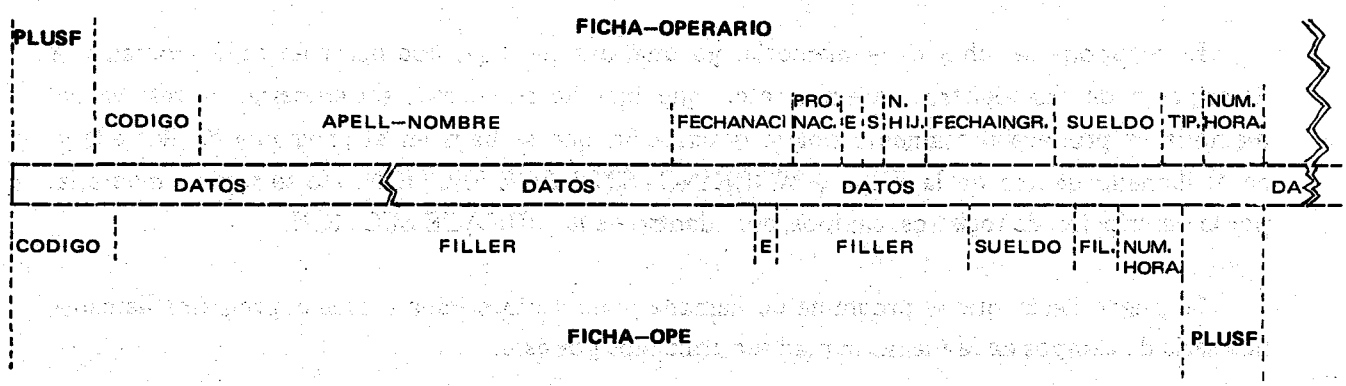
```
002140    ENTRY 'PLUSF-ERRORFICHA' USING PLUSF, FICHA-OPE.
```

ya que de esta forma se establece la correspondencia uno a uno entre los campos.

Sin embargo, no hubiera sido correcto especificar dentro de la declaración ENTRY del programa NOMIN:

```
002140    ENTRY 'PLUSF-ERRORFICHA' USING FICHA-OPE, PLUSF.
```

ya que establecería entre los campos que siguen a la opción USING de las declaraciones ENTRY y CALL citadas, la correspondencia siguiente:



lo cual quiere decir, que cuando se hiciera referencia al campo CODI (línea 001170) en el programa NOMIN, el Sistema tomaría el valor que en ese momento contuviera el campo PLUSF del programa NOMINA.

3.2.- PUNTO de ENTRADA

Se denomina así al lugar del programa donde se especifica una declaración ENTRY.

3.2.1.- Declaración ENTRY

FUNCION: Establecer un punto de entrada en un programa llamado.

FORMATO:

ENTRY literal [USING identificador-1 [identificador-2] ...]

Dentro de un programa pueden existir uno o varios puntos de entrada, denominándose cada uno con un nombre (literal) diferente.

Cuando el punto de entrada va a tener lugar en la primera declaración que sigue a la PROCEDURE DIVISION, se puede prescindir de ENTRY literal, y especificar, a continuación de la palabra DIVISION, la opción USING seguida del nombre de los campos de datos que le va a ceder el programa que le llama.

EJEMPLO. En el programa DESGLO2 (pág. 397) es correcto cuando se especifica:

```
002010 PROCEDURE DIVISION USING BILLET, TITULOS.
```

Sin embargo, no hubiera sido correcto:

```
002010 PROCEDURE DIVISION.
002011     ENTRY 'DESGLO2' USING BILLET, TITULOS.
```

Cuanto se especifica acerca de la opción USING en la declaración CALL, es aplicable a la declaración ENTRY.

3.3.- PUNTO de RETORNO

Un programa llamado debe contener, además del punto de entrada, el encadenamiento necesario para llevar a cabo la bifurcación de regreso al programa que lo llamó, a fin de introducirse en él en la declaración siguiente a la CALL que produjo la llamada.

La declaración que realiza este cometido es: EXIT PROGRAM cuya descripción se hizo en la pág. 232.

En un programa llamado debe existir, como mínimo, una declaración EXIT.

Se emplea en algunos Sistemas la declaración GOBACK, cuya función es equivalente a la que realiza la declaración EXIT.

3.4.- LINKAGE SECTION

FUNCION: Sirve para describir los campos de datos de la memoria puestos a disposición por otro programa.

FORMATO:

LINKAGE SECTION.

[Cláusulas de descripción de un campo de datos] . . .
[Cláusulas de descripción de un registro] . . .

La descripción de registros y campos se realiza utilizando las mismas cláusulas que si dicha descripción se hiciera en la WORKING–STORAGE SECTION, a excepción de la cláusula VALUE que tiene la misma limitación que si se describiera en la FILE SECTION; es decir, sólo puede usarse junto con los nombres de condición.

Por la descripción que se haga de registros y/o campos en esta Sección, el Sistema NO RESERVA memoria, pero sí permite poner a disposición del programador nombres de campos (identificadores) con los cuales poder REFERIRSE y, por tanto, ser utilizados aquellos campos de la memoria reservados por otros programas. Es decir, la descripción que se hace en la LINKAGE SECTION del programa llamado es prácticamente una REDEFINICION de los campos descritos en la FILE o WORKING del programa de llamada.

Así, por la descripción de FICHA-OPERARIO en el programa NOMINA, se han reservado campos de la memoria donde se irán guardando los datos contenidos en cada ficha. Parte de estos campos van a ser utilizados en el programa NOMIN, para lo cual es preciso hacer su descripción en la LINKAGE SECTION especificándose con igual o distinto nombre, es indiferente, pero siempre conservando la misma posición relativa dentro del registro:

```
001160    01  FICHA-OPE.
001170        02  CODI                    PIC X(5).
001180        02  FILLER                  PIC X(43).
001190        02  ESTADO                  PIC 9.
001200            88  SOLTERO               VALUE 1.
001210            88  CASADO                VALUE 2.
001220            88  VIUDO                 VALUE 3.
001230        02  FILLER                  PIC X(9).
001240        02  SUELDO                  PIC 9(5).
```

Efectivamente, los nombres FICHA–OPERARIO y CODIGO del programa NOMINA son diferentes a FICHA–OPE y CODI, ambos hacen referencia al mismo área de la memoria. Sin embargo, los campos ESTADO, SUELDO, etc., se denominan de igual forma en los dos programas.

Es fundamental describir los campos FILLER dentro de FICHA–OPE, a fin de que todos los campos que componen dicho registro conserven la misma posición relativa que en el registro FICHA–OPERARIO, según se dijo al describir la opción USING (pág. 391).

La LINKAGE SECTION es obligatoria cuando dentro de un programa figura una declaración ENTRY con la opción USING.

3.5.- CONSIDERACIONES GENERALES

En el apartado 2 se vió cómo un programa puede llamar a otro. Hay que añadir que es también posible que el programa llamado pueda a su vez llamar a otro programa.

Sin embargo, la llamada entre programas no puede realizarse de forma caprichosa, sino que han de tenerse en cuenta las consideraciones siguientes:

- El programa que comienza el proceso puede llamar a cualquier programa de los que forman la unidad de proceso, pero no puede ser llamado por el resto de ellos.

- Si un programa llama a otro, éste no puede llamar a quien lo llamó.

EJEMPLO:

Se supone que en la ejecución de un trabajo van a intervenir tres programas: ALFA, BETA y ZETA.

Se ha previsto que el proceso lo debe comenzar el programa ALFA, el cual llamará a ZETA para cederle el control, y una vez terminada su función devolver el control a ALFA para que éste llame a BETA. Una vez que BETA termine su función llamará a ZETA, etc.

A partir de este momento y por haber intervenido los tres programas en el proceso, las sucesivas llamadas entre ellos deberán ajustarse a las limitaciones que son consecuencia de las consideraciones citadas anteriormente, es decir:

- El programa ALFA podrá seguir llamando a BETA y a ZETA.

- El programa BETA podrá seguir llamando a ZETA, pero no a ALFA .

- El programa ZETA no podrá llamar a ALFA ni a BETA.

Por tanto, sólo se podrá incluir en el programa ALFA, declaraciones de enlace CALL; en el programa BETA las declaraciones CALL, ENTRY y EXIT, y en el programa ZETA las declaraciones ENTRY y EXIT.

Al programa ALFA que puede llamar y no ser llamado se le suele denominar programa PRINCIPAL, y a BETA y ZETA subprogramas.

El siguiente ejemplo muestra una serie de programas que llaman y programas que son llamados. Puede también servir de ejemplo de cómo un trabajo se puede dividir en varias partes, asignando cada parte a un programador para que codifique y depure el programa correspondiente. Es fácil comprender que lo que se persigue con este ejemplo, son fines didácticos, ya que el trabajo que realiza no sería, por su extensión, el más adecuado para dividirle en varias partes.

EJEMPLO 2º :

Se necesita realizar, con urgencia, el desglose de billetes y monedas de los sueldos netos resultantes de la confección de la nómina.

Para ello se reunen tres programadores y deciden repartirse el trabajo de la siguiente forma:

- El primer programador confeccionará un programa en el que describa dos tablas: una conteniendo el valor de los diferentes tipos de billetes y monedas existentes, y otra conteniendo los diferentes títulos que han de figurar en el informe impreso: BILLETES DE CINCO MIL PTS., BILLETES DE MIL PTS., ... etc.

- El segundo programador confeccionará un programa con el que se realice el desglose de los sueldos netos, utilizando como archivo de entrada el archivo en cinta obtenido durante la ejecución del programa NOMINA.

- El tercer programador confeccionará un programa con el que lleve a cabo la impresión de los diferentes titulares y resultados del desglose, conforme al formato de la pág. 253.

SOLUCION:

PROGRAMA DESGLO1

PROGRAMADOR C. de la FUENTE FECHA 14-9-1984 PERFORACIONES

```
001010  IDENTIFICATION DIVISION.
001020    PROGRAM-ID. DESGLO1.
001030    AUTHOR. C. DE LA FUENTE.
001040*   PROGRAMA PRINCIPAL QUE JUNTO CON LOS PROGRAMAS DESGLO2 Y
001050*       DESGLO3 REALIZAN EL TRABAJO DENOMINADO 'DESGLOSE DE MONEDA'.
001060  ENVIRONMENT DIVISION.
001070  DATA DIVISION.
001080    WORKING-STORAGE SECTION.
001090    01   BILLETES.
001100           02 FILLER PICTURE X(32)  VALUE '5000010000500010000500025000050
001110-                                          '001'.
001120    01   BILLET REDEFINES BILLETES.
001130           02 BILLETE OCCURS 8 PICTURE 9(4).
001140    01   TITULARES.
001150           02 FILLER PICTURE X(27) VALUE 'BILLETES DE CINCO MIL PTS.  '.
001160           02 FILLER PICTURE X(27) VALUE 'BILLETES DE MIL PTS.        '.
001170           02 FILLER PICTURE X(27) VALUE 'BILLETES DE QUINIENTAS PTS. '.
001180           02 FILLER PICTURE X(27) VALUE 'BILLETES DE CIEN PTS.       '.
001190           02 FILLER PICTURE X(27) VALUE 'MONEDAS DE CINCUENTA PTS.   '.
001200           02 FILLER PICTURE X(27) VALUE 'MONEDAS DE VEINTICINCO PTS. '.
001210           02 FILLER PICTURE X(27) VALUE 'MONEDAS DE CINCO PTS.       '.
001220           02 FILLER PICTURE X(27) VALUE 'MONEDAS DE UNA PTS.         '.
001230    01   TITULOS REDEFINES TITULARES.
001240           02 TITULO OCCURS 8 PICTURE X(27).
001250  PROCEDURE DIVISION.
002010  LLAMADA-A-DESGLO2.
002020           CALL 'DESGLO2' USING BILLET TITULOS.
002030  FIN.
002040           STOP RUN.
```

PROGRAMA DESGLO2

PROGRAMADOR: M.F. SACRISTAN — FECHA: 14-9-1984 — PERFORACIONES:

Página	Línea	Cont.	A	B
001	010			IDENTIFICATION DIVISION.
001	020			PROGRAM-ID. DESGLO2.
001	030	*		ESTE PROGRAMA ES LLAMADO POR EL PROGRAMA PRINCIPAL PARA HACER LA
001	040	*		RUTINA DE DESGLOSE DE MONEDA, Y A SU VEZ LLAMA AL PROGRAMA
001	050	*		DESGLO3 PARA QUE CONTINUE LA EJECUCION DEL TRABAJO.
001	060			ENVIRONMENT DIVISION.
001	070			INPUT-OUTPUT SECTION.
001	080			FILE-CONTROL.
001	090			SELECT CINTA ASSIGN SYS005-UT-2400-S.
001	100			DATA DIVISION.
001	110			FILE SECTION.
001	120			FD CINTA RECORDING F LABEL RECORD OMITTED BLOCK 10 RECORDS.
001	130			01 REGISCIN.
001	140			02 DATOS PIC X(80).
001	150			02 SUELDO PIC 9(6).
001	160			WORKING-STORAGE SECTION.
001	170			77 A PIC 9.
001	180			77 COCIENTE PIC 99.
001	190			01 CONTADORES.
001	200			02 CONTADOR OCCURS 8 PIC 9(6).
001	210			LINKAGE SECTION.
001	220			01 BILLET.
001	230			02 BILLETE OCCURS 8 PIC 9(4).
001	240			01 TITULOS.
001	250			02 TITULO OCCURS 8 PIC X(27).
002	010			PROCEDURE DIVISION USING BILLET TITULOS.
002	020			ABRIR.
002	030			OPEN INPUT CINTA.
002	040			MOVE ZEROS TO CONTADORES.
002	050			LEER.
002	060			READ CINTA AT END CLOSE CINTA GO TO LLAMADA-A-DESGLO3.
002	070			MOVE 1 TO A.
002	080			PERFORM DESGLOSAR 8 TIMES GO TO LEER.
002	090			DESGLOSAR.
002	100			DIVIDE BILLETE (A) INTO SUELDO
002	110			GIVING COCIENTE REMAINDER SUELDO
002	120			ADD COCIENTE TO CONTADOR (A).
002	130			ADD 1 TO A.
002	140			LLAMADA-A-DESGLO3.
002	150			CALL 'DESGLO3' USING A, CONTADORES, TITULOS.
002	160			RETORNO-A-DESGLO1.
002	170			EXIT PROGRAM.
	180			
	190			
	200			

PROGRAMA DESGLO3

PROGRAMADOR: NIEVES CABANELAS R. | FECHA: 14-9-1984 | PERFORACIONES

```
001010   IDENTIFICATION DIVISION.
001020     PROGRAM-ID. DESGLO3.
001030 *   ESTE PROGRAMA ES LLAMADO POR EL PROGRAMA DESGLO2 PARA REALIZAR
001040 *             LA IMPRESION DE LOS RESULTADOS DEL DESGLOSE DE MONEDA.
001050     ENVIRONMENT DIVISION.
001060       CONFIGURATION SECTION.
001070         SPECIAL-NAMES.
001080             DECIMAL-POINT IS COMMA
001090             C01   IS PAG-NUEVA.
001100       INPUT-OUTPUT SECTION.
001110         FILE-CONTROL.
001120             SELECT  IMPRESORA  ASSIGN SYS006-UR-1403-S.
001130     DATA DIVISION.
001140       FILE SECTION.
001150         FD   IMPRESORA   RECORDING F LABEL RECORD OMITTED.
001160         01   IMPRESO     PIC X(133).
001170       WORKING-STORAGE SECTION.
001180     01   CABECERA-A.
001190             02 FILLER     PIC X(30)   VALUE SPACES.
001200             02 FILLER     PIC X(76)   VALUE 'NUMERO DE BILLETES Y MONEDAS N
001210-                'ECESARIAS PARA PAGAR A LOS OPERARIOS DE RICOSA'.
001220     01   CABECERA-B.
001230             02 FILLER     PIC X(50)   VALUE SPACES.
001240             02 TITULIN    PIC X(30).
001250             02 TOTAL      PIC ZZZ.ZZ9.
002010             02 FILLER                 PIC X(46)   VALUE SPACES.
002020     LINKAGE SECTION.
002030     77   A                            PIC 9.
002040     01   TITULOS.
002050             02 TITULO     OCCURS 8    PIC X(27).
002060     01   CONTADORES.
002070             02 CONTADOR   OCCURS 8    PIC 9(6).
002080     PROCEDURE DIVISION USING A CONTADORES TITULOS.
002090               OPEN  OUTPUT  IMPRESORA.
002100               WRITE IMPRESO FROM CABECERA-A  AFTER  PAG-NUEVA
002110               MOVE  1  TO  A.
002120               PERFORM  IMPRIMIR-RESULTADOS  8  TIMES.
002130               CLOSE IMPRESORA.
002140     RETORNO-A-DESGLO2.
002150               EXIT PROGRAM.
002160     IMPRIMIR-RESULTADOS.
002170               MOVE TITULO (A) TO TITULIN
002180               MOVE CONTADOR (A)  TO  TOTAL
002190               WRITE IMPRESO  FROM  CABECERA-B  AFTER 2.
002200               ADD  1  TO  A.
```

DESCRIPCION DEL PROCESO.

Comienza el primero de los tres programas (DESGLO1) con la ejecución de la declaración CALL (línea 002020), la cual llama al programa DESGLO2,le transfiere el control y le cede para su empleo, si lo precisa, la tabla BILLET y TITULOS descritos en las líneas 001120 y 001230 respectivamente.

Toma el control el programa DESGLO2, comenzando su ejecución a partir de la primera declaración de la PROCEDURE DIVISION por haber especificado dentro de la citada declaración CALL el nombre del programa llamado. Ejecuta las declaraciones siguientes hasta la declaración de la línea 002130 con las cuales realiza el desglose de cada sueldo. Cuando finaliza el desglose de todos los sueldos ejecuta la declaración CALL'DESGLO3' (línea 002140) con la cual llama al programa DESGLO3, y le cede el control y el empleo de los campos A, CONTADORES y TITULOS.

Recibido el control de DESGLO2, el programa DESGLO3 comienza su ejecución a partir de la primera declaración, realizando con la declaraciones especificadas desde la línea 002090 a la línea 002130 la impresión de la cabecera, así como la de los ocho titulares con sus correspondientes cantidades. Cuando finaliza la impresión, ejecuta la declaración EXIT PROGRAM (línea 002150) la cual devuelve el control al programa DESGLO2 entrando en la declaración siguiente a la CALL que originó la llamada, es decir, en la declaración EXIT PROGRAM (línea 002170) la cual hace que devuelva a su vez el control al programa DESGLO1, entrando en la declaración STOP RUN, por la que finaliza el proceso.

Hay que hacer notar que aunque el programa DESGLO1 cede al programa DESGLO2 los campos BILLET y TITULOS, este programa utilizará para hacer el desglose solamente los campos que componen la tabla BILLET. Sin embargo, es necesario ceder también TITULOS y describirlos en la LINKAGE SECTION para que este programa pueda a su vez cederlos al programa DESGLO3, que sí los utilizará. El motivo de hacerlo así, es debido a que el programa DESGLO1 al no llamar al programa DESGLO3 no puede pasarle estos campos, y por tanto debe hacerlo a través del intermediario DESGLO2.

4.- EJERCICIOS (Repaso)

1.- Alguna de las siguientes afirmaciones NO es correcta:

a) No puede haber espacios en blanco dentro de un nombre creado.
b) Un guión en la columna 7 de la Hoja de Codificación sirve para indicar la continuación de una palabra COBOL.
c) El punto decimal, en un literal numérico de coma fija, puede estar en cualquier lugar del literal, excepto en el extremo izquierdo.
d) No es obligatorio llenar de inscripciones una línea de la hoja de codificación.

2.- Alguna de las siguientes afirmaciones ES correcta:

a) Deben escribirse dentro del margen B los números de nivel 02 al 49.
b) Los nombres-de-párrafo deben ir escritos solos en una línea.
c) La escritura en las columnas 73 a 80 es obligatoria.
d) El paréntesis izquierdo no debe ir seguido de un espacio, pero si precedido.

3.- Una de las siguientes palabras es un nombre creado por el programador:

a) PROGRAN-ID
b) DIVISION
c) FINAL
d) NO

4.- El literal '34521.45' es:

a) De coma fija
b) De coma flotante
c) No numérico
d) Numérico

5.- Las opciones OMITTED y STANDARD de la cláusula LABEL RECORD pueden especificarse indistintamente para archivos contenidos en:

a) Disco
b) Cinta magnética
c) Impreso
d) Fichas

6.- La cláusula VALUE no tiene limitaciones a su empleo cuando se utiliza en la:

a) FILE SECTION
b) LINKAGE SECTION
c) WORKING-STORAGE SECTION
d) REPORT SECTION

7.- En la descripción siguiente:

```
010 WORKING-STORAGE SECTION.
020   01  CAMPOA   PIC XXX   VALUE '0'.
030   01  CAMPOB   PIC 999   VALUE ZERO.
```

¿Qué valor inicial se situará dentro de CAMPOA y CAMPOB?

a) CAMPOA [0 0 0] CAMPOB [0 0 0]
b) CAMPOA [0 _ _] CAMPOB [_ _ 0]
c) CAMPOA [0 _ _] CAMPOB [0 _ _]
d) CAMPOA [0 _ _] CAMPOB [0 0 0]

8.– Si un campo de edición está descrito de la forma siguiente:

```
010 02 ALFA      PIC +ZZZ.99.
```

¿Cuál será la salida impresa si la cantidad a editar es –20?

a) | – | 0 | 2 | 0 | . | 0 | 0 |

b) | – | | 2 | 0 | . | 0 | 0 |

c) | + | 0 | 2 | 0 | . | 0 | 0 |

d) | | | | | – | 2 | 0 |

9.– Para editar la cantidad $3,452.36– ¿cuál de las siguientes PICTURES no sería correcta?

a) PIC $Z,ZZZ.ZZ+

b) PIC $$,$$9.99+

c) PIC $$Z,999.99–

d) PIC $,$$$.99–

10.– En la siguiente descripción:

```
010 .....
020   02  COCIENTE      PIC 99.
030   02  RESTO         PIC 99.
040 PROCEDURE DIVISION.
050 .....
060       DIVIDE 1000 INTO 50 GIVING COCIENTE REMAINDER RESTO
```

¿Cuál sería el contenido de los campos: COCIENTE y RESTO después de ejecutada la declaración DIVIDE?

a) COCIENTE | 0 | 0 | RESTO | 0 | 0 |

b) COCIENTE | 2 | 0 | RESTO | 0 | 0 |

c) COCIENTE | 0 | 0 | RESTO | 5 | 0 |

d) COCIENTE | 0 | 0 | RESTO | 2 | 0 |

11.– En la descripción siguiente:

```
080 .....
090   02  ALFA      PIC 999V9   VALUE 172.53.
100   02  BETA      PIC  9V999  VALUE   7.25.
110   02  OMEGA     PIC +++.9.
120 PROCEDURE DIVISION.
130 .....
140       ADD ALFA, BETA GIVING OMEGA ROUNDED
```

¿Cuál sería el contenido del campo OMEGA una vez ejecutada la declaración ADD?

a) | + | 7 | 9 | . | 8 |

b) | | 7 | 9 | . | 8 |

c) | 0 | 7 | 9 | . | 8 |

d) | 1 | 7 | 9 | . | 8 |

12.– En la descripción siguiente:

```
030 WORKING-STORAGE SECTION.
040 .....
050   02  ALFA      PIC X(4)  VALUE '2222'.
060   02  BETA      PIC X(4)  VALUE 'A99B'.
070   02  ZETA      PIC X(4)  VALUE SPACES.
080   02  SIGMA     PIC X(4)  VALUE ZEROS.
```

Si se efectúa una prueba de Relación entre los diferentes campos ¿cuál será el campo cuyo contenido es el menor?

a) ALFA
b) BETA
c) ZETA
d) SIGMA

13.– En la descripción siguiente:

```
050   02  ALFA        PIC X(5)   JUSTIFIED RIGHT.
060   02  BETA        PIC 9(3)V9 BLANK WHEN ZERO.
070 PROCEDURE DIVISION.
080 ..........................................
090       MOVE  ZEROS TO BETA
100       MOVE  'FIN' TO ALFA
```

¿Cuál sería el contenido del campo ALFA y BETA después de ejecutadas las declaraciones MOVE?

a) ALFA | | | F | I | N | BETA | 0 | 0 | 0 | · | 0 |
b) ALFA | | | F | I | N | BETA | | | | |
c) ALFA | | | F | I | N | BETA | | | | | |
d) ALFA | F | I | N | | | BETA | | | | |

14.– En la descripción siguiente:

```
120 01  CABECERA-1.
130       02  ALFA    PIC X(75)  VALUE SPACES.
140       02  BETA    PIC X(5).
150       ..........................
160 PROCEDURE DIVISION.
170       MOVE ALL '*' TO BETA
```

Después de ejecutada la declaración MOVE, el contenido del campo BETA sería:

a) | * | | | | |
b) | | | | | * |
c) | * | * | * | * | * |
d) | | | | | |

15.– En la descripción siguiente:

```
190 ..........................................
200   02  PAG-NUEVA          PIC 99   VALUE 12.
210 PROCEDURE DIVISION.
220 ..........................................
230       WRITE IMPRE AFTER ADVANCING PAG-NUEVA
```

La declaración WRITE obliga a imprimir el registro IMPRE después de saltar:

a) Doce líneas
b) Al comienzo de nueva página
c) Al canal 12
d) Doce líneas a partir del comienzo de la página

16.– En la descripción siguiente:

```
010 .....
020     03 ALFA      PIC 9  VALUE 3.
030     03 CAMPOA    PIC 9  VALUE 1.
040 PROCEDURE DIVISION.
050 .....
060 A.  GO TO M, N, P, Q DEPENDING ON ALFA.
070 M.  MOVE 7 TO CAMPOA STOP RUN.
080 N.  ADD 5 TO CAMPOA STOP RUN.
090 P.  MULTIPLY 5 BY CAMPOA STOP RUN.
100 Q.  SUBTRACT 1 FROM CAMPOA STOP RUN.
```

¿Hacia qué procedimiento bifurcaría la secuencia de ejecución del programa, cuando se lleve a cabo la declaración GO?

a) M
b) N
c) P
d) Q

17.– En la siguiente descripción:

```
120 01  BETA     PIC 99  VALUE ZEROS.
130 01  CAMPOA   PIC 99  VALUE 1.
140 PROCEDURE DIVISION.
150 .....
160     PERFORM CALCULO UNTIL BETA > 49. STOP RUN.
170 .....
180 CALCULO. ADD 7 TO BETA
190          MULTIPLY 2 BY CAMPOA.
```

¿Cuál sería el contenido de CAMPOA una vez finalizada la ejecución de la declaración PERFORM?

a) 32
b) 64
c) 56
d) 28

18.– En la descripción siguiente:

```
090 .....
100 02  BETA     PIC 9.
110 PROCEDURE DIVISION.
120 .....
130     PERFORM ALFA VARYING BETA FROM -8 BY 2 UNTIL BETA = 0
140     CLOSE ARCHIVO  STOP RUN.
150 ALFA.
160         MOVE CAMPOA  TO  CAMPOB .....
```

¿Cuántas veces se ejecutará el procedimiento ALFA?

a) 6
b) 1
c) 5
d) 0

19 .– Un subíndice conviene que este descrito con USAGE:

a) COMP
b) COMP–3
c) DISPLAY
d) COMP–1

20.– En la descripción siguiente:

```
180  01  TABLA-VENTAS.
190      02  VENTAS-ANUALES  OCCURS 5.
200          04  VENTAS-MENSUALES  OCCURS  12.
210              06  VENTAS-DIARIAS  OCCURS  30.
220                  07  MODELOA        PIC 999.
230                  07  MODELOB        PIC 999.
230                  07  MODELOC        PIC 99.
...                  ..  .....          ...  ...
```

¿Cuál de las siguientes inscripciones sería necesaria para visualizar por la consola, las ventas diarias del MODELOA que contiene el elemento que ocupa en la tabla el lugar 1.300?

a) DISPLAY MODELOA (4, 10, 8) UPON CONSOLE.
b) DISPLAY VENTAS–DIARIAS (5, 8, 10) UPON CONSOLE.
c) DISPLAY MODELOA (4, 8, 10) UPON CONSOLE.
d) DISPLAY VENTAS–DIARIAS (4, 10, 8) UPON CONSOLE.

21.– En una tabla que contiene 256 elementos ¿Cuál será el número máximo de búsquedas necesarias para localizar a cualquier elemento, si la búsqueda se hace en forma binaria?

a) 256
b) 8
c) 9
d) 255

22.– La declaración SEARCH ALL puede realizar la búsqueda en aquellas tablas que utilicen en su descripción la opción:

a) OCCURS (únicamente)
b) DEPENDING ON
c) OCCURS e INDEXED (únicamente)
d) ASCENDING y/o DESCENDING

23.– Con una de las siguientes declaraciones no es posible incrementar el valor contenido en un nombre–de–índice:

a) PERFORM
b) SET
c) ADD
d) SEARCH

24.– Para poder grabar (adiciones) registros al azar en un archivo secuencial indexado, una de las siguientes cláusulas o declaraciones no es necesaria. Decir cuál es:

a) ACCESS
b) NOMINAL
c) WRITE
d) REWRITE

APENDICES

APENDICE A

SOLUCIONES a los EJERCICIOS

CAPITULO I.

1.– D.
2.– A.
3.– B.
4.– C.
5.– D.
6.– 025, numero de Horas Extras.
7.– C.
8.– B.
9.– D.
10.– D.
11.– B y C.
12.– D.
13.– (A–D–1)/3.
14.– B.
15.– MOVE FICHA–OPERARIO TO DATOSCIN y
MOVE TOTALS TO TOTAL, TOTAL–SUELDOCIN.
16.– C.
17.– En la primera pagina: 23 líneas y 24 en las siguientes.
18.– FALSO.
19.– C.
20.– D.
21.– 133.
22.– SUELDO, CODIGO, NUMERO DE LA SECCION.
23.– D.

CAPITULO II.

1.– A.

2.– No, ya que PLUS es una palabra reservada y no puede utilizarse como nombre de un campo de datos.

3.– La respuesta A no es un nombre creado por llevar intercalado un espacio en blanco. La respuesta B, por ser una palabra reservada. La respuesta C, por llevar incorporada la barra (/). La respuesta D, por no poder ir un guión al final de una palabra.

4.– La respuesta A, no es correcta por llevar un punto al final del literal. La respuesta B, porque el signo debe ir delante del literal. La respuesta C, por dejar un espacio en blanco entre el signo y la primera cifra. La respuesta D, por no poder formar el signo $ parte de un literal.

5.– La respuesta B, por no estar encerrada entre apóstrofos.

6.– La respuesta C no es un nombre de datos,ya que no contiene una letra como mínimo, sin embargo, sí podía ser válido como nombre de procedimiento.

7.– A la respuesta B le sobra la O. La respuesta C debe escribirse ALL 'B'. La respuesta D debe escribirse HIGH–VALUE.

8.– 1000 es un literal, TRIENIOS y NUMTRIENIOS son nombres de datos.

9.– Cierto.

10.– Línea 001180 La palabra INPUT debe comenzar a partir de la columna 8.
Línea 001190 La palabra SELECT debe comenzar una columna más a la derecha, ya que después de un punto hay que dejar un espacio como mínimo.
Línea 001200 La palabra ALTERNATEAREA debe escribirse ALTERNATE AREA.
Línea 004010 FD debe comenzar en el Margen A (comienza en el margen B).
Línea 004040 01 debe comenzar en el Margen A (comienza en la columna 7).
Línea 004050 Después de la palabra SECTION debe ir un punto.
Líneas 006190 y 006200 Se ha omitido el carácter guión (–) en la columna 7.
Línea 007020 Después del carácter punto y coma (;) debe ir un espacio en blanco.
Línea 007070 Después del paréntesis izquierdo no debe existir ningún espacio en blanco, así como entre el carácter / y el 3 debe existir un espacio en blanco.
Línea 007080 El signo – comprendido entre FECHAINGRESO y 1 debe ir precedido y seguido de un espacio en blanco.

CAPITULO III.

1.– D. 2.– C. 3.– FALSO.

4.– Línea 001010 Debe especificar IDENTIFICATION.
Línea 001030 " " AUTHOR.
Línea 001050 " " DATE–WRITTEN.
Línea 001070 " " SECURITY.
Línea 001110 Después de la palabra DIVISION debe especificar un punto.
Línea 001120 Debe especificar SECTION.
Línea 001170 " " COMMA.
Línea 001180 " " INPUT.
Línea 001190 " " FILE–CONTROL.
Línea 001200 " " ASSIGN.
Línea 001220 " " ACCESS y SEQUENTIAL.
Línea 001250 " " ARCHI–IMPRESO.
Línea 001230 Sobra el punto especificado al final de la línea.

5.– Pueden suprimirse las líneas 001210, 001220 y 001240. También, con determinado tipo de compiladores, puede prescindirse del contenido de las líneas 001130 y 001140.

6.– Sí son correctas, ya que después de un nombre de párrafo (FILE–CONTROL) pueden hacerse inscripciones en la misma línea, así como que en una misma línea pueden describirse varias cláusulas (RESERVE, ACCESS, etc.).

7.–

```
PROGRAMA:                    FECHA          PERFORACIONES
Página Línea Cont A B
                                                                  73  80
                1   5   10  15  20  25  30  35  40  45  50  55  60  65  70
001010   ID DIVISION.
001020     PROGRAM-ID. ACTUALIZ.
001030     REMARKS. PROGRAMA PARA ACTUALIZAR EL ARCHIVO MAESTRO.
001040                      DE PIEZAS DE REPUESTO DE LA FACTORIA COIRSA.
001050
001060   ENVIRONMENT DIVISION.
001070     CONFIGURATION SECTION.
001080       SOURCE-COMPUTER. IBM-370.
001090       OBJECT-COMPUTER. IBM-370.
001100     INPUT-OUTPUT SECTION.
001110       FILE-CONTROL.
001120           SELECT ARCHI-MAES    ASSIGN SYS004-UT-2400-S  RESERVE NO.
001130           SELECT ARCHI-MODIFI  ASSIGN SYS005-UT-2400-S  RESERVE NO.
001140           SELECT ARCHI-ACTUAL  ASSIGN SYS006-UT-2400-S  RESERVE NO.
001150       I-O-CONTROL.
001160           RERUN ON SYS008-UT-2400-S EVERY 100 RECORDS OF ARCHI-ACTUAL.
```

8.– No es necesario, ya que normalmente las inscripciones que componen el párrafo SPECIAL–NAMES se emplean para controlar el salto y espaciado en la impresora, así como para hacer más legibles los resultados numéricos (edición) en el impreso etc., y en este ejercicio no se utilizan archivos impresos.

CAPITULO IV.

1.– D. 2.– CIERTO. 3.– CIERTO.

4.–

```
002090   FD  ARCHIVO-OPERARIOS  LABEL RECORD OMITTED.

002110   FD  CINTA-NOMINA  BLOCK 10 RECORDS
002120                     LABEL RECORD OMITTED.

002140   FD  ARCHI-IMPRESO LABEL RECORD OMITTED.
```

5.–

```
Página Línea Cont A B
                1   5   10  15  20  25  30  35  40  45  50  55  60  65
002010   DATA DIVISION.
002020     FILE SECTION.
002030       FD ARCHI-MAES  RECORDING F  BLOCK 15 RECORDS
002040                      LABEL RECORD STANDARD  DATA RECORD IS MAES.
002050         01 MAES.
002060            02 CODIGO     PIC 9(8).
002070            02 EXISTEN    PIC 9(10).
```

```
002080     FD ARCHI-MODIFI   RECORDING F    DATA RECORD IS MODIFI
002090                       LABEL RECORD OMITTED.
002100     01 MODIFI.
002110        02 CODI              PIC 9(8).
002120        02 VARIACION         PIC 9(10).
002130        02 CLAVE             PIC 9.
002140     FD ARCHI-ACTUAL   RECORDING F    LABEL RECORD STANDARD
002150                       DATA RECORD ACTUALI  BLOCK 15 RECORDS.
002160     01 ACTUALI.
002170        02 CODIG             PIC 9(8).
002180        02 EXISTACT          PIC 9(10).
002190  WORKING-STORAGE SECTION.
002200  77   CAMBIO1               PIC 9.
002210  77   CAMBIO2               PIC 9.
002220  77   CAMBIO3               PIC 9.
```

6.– No puede haber dos nombres–de–archivo que se denominen de la misma forma: ARCHI–CINTA.

7.– Sí es correcto, ya que el compilador puede diferenciar cada uno de los registros que se denominen FICHA, de la siguiente forma:
FICHA IN ARCHIOPE y FICHA OF ARCHIMPRE.

8.– D.

9.– 02 ALFA PIC S9999V999.

10.– B y C.

11.– 03 BETA PIC ZZZ.9DB.

12.– 03 ZETA PIC $$$,$$9.99+

13.– Línea 001090, USAGE COMP es incompatible con una PIC alfanumérica.
Línea 001130, se debe especificar S99.
Línea 001140, si en un campo se especifica PIC, no se debe especificar COMP–2, y a la inversa.
Línea 001160, es una redundancia especificar JUSTIFIED, ya que los campos numéricos se justifican automáticamente por la derecha sin especificar dicha cláusula.

14.– El campo PROVINCIA–NACIM debe llevar PICTURE 99 ó apóstrofos en las VALUES.

15.– Que el número de nivel que debe preceder a los nombres de condición es el 88.

16.– No, ya que la suma de la serie de caracteres de las PICTURES que siguen a los campos BE1 y BE 2 es menor que la que sigue a ALFA: $6 < 7$.

17.– Sí puede, si bien en este caso no es correcta la descripción de la tabla por no poder especificarse OCCURS con un nivel 01.

18.– En la WORKING–STORAGE SECTION no deben describirse archivos.

19.– No puede haber dos registros (nivel 01) que se denominen de la misma forma, ya que no existe posibilidad de calificación, por lo que el compilador no podrá diferenciar un registro de otro.

20.– Línea 002070 No se puede especificar BLOCK 0 (cero).
Línea 002090 El número de nivel que debe preceder al nombre de registro FICHA–OPERARIO es el 01.
Línea 002110 El valor 5 no debe estar encerrado entre paréntesis con espacios en blanco.
Línea 002130 FECHA-NACI no debe llevar PICTURE por ser campo compuesto.
Línea 002170 Sólo se emplea la cláusula VALUE dentro de la FILE SECTION con nombres––de–condición.
Línea 002180 No se ha especificado la cláusula PICTURE en el campo ESTADO.
Líneas 003180 a 003220 Falta la descripción de la cláusula LABEL, obligatoria en la descripción de todo archivo.
Línea 003240 No es correcto 9(80), ya que el campo DATOSCIN va a contener campos alfanuméricos,y aún en el caso de que sólo hubiera de contener números, el máximo valor que se podría especificar sería 9(18).
Línea 004010 Sobra el punto después de RECORDING IS F.
Línea 004040 El mismo error que en la línea 002170.
Línea 004051 El número de nivel 01 se debe especificar a continuación de los campos definidos con nivel 77.
Línea 004080 El punto final debe ir inmediatamente después del paréntesis derecho, y el número de nivel 88 no es válido.
Línea 005200 Después de la palabra reservada ALL debe ir ' '.
Línea 005230 Después de la palabra VALUE debe ir un literal o una constante figurativa, y no la palabra BLANCOS que nada significa.

CAPITULO V.

1.– La solución B no es correcta,ya que un nombre de procedimiento no puede terminar en un guión.

2.– En una declaración ADD sólo puede usarse TO o GIVING, nunca ambos.
La 2 es correcta y equivale a MOVE ZEROS TO ALFA
La palabra ROUNDED (caso de utilizarse) debe seguir al campo que se especifique después de la palabra COMPUTE,es decir, COMPUTE NETO ROUNDED = ...
En vez de FOR debe escribirse BY.
Es correcta la 5.

3.– CAMPOA permanecería con el valor 740,ya que no puede sumarse un literal no numérico a un campo de tipo numérico.

4.– Se transferirían ceros, ya que el resultado de multiplicar 5000 x 225 = 1125000 al no caber dentro del campo ALFA, no introduciría dicho resultado en ALFA y pasaría a ejecutar la declaración que sigue a ON SIZE ERROR, es decir, MOVE ZEROS TO ALFA.

5.– CAMPOB = | 0 | 7 | 3 | CAMPOC = | 0 | 2 | 5 |

6.– COMPUTE CONTADOR = CONTADOR + 1

7.– No son equivalentes. El valor de ALFA en la primera COMPUTE sería ALFA = $\frac{8^2}{4}$ x 2 = = 32 y en la segunda ALFA = $\frac{8^2}{4 \times 2}$ = 8;

8.– No está permitido el movimiento de un literal no numérico '1234' al campo numérico ALFA.

9.– Bifurcaría al procedimiento TOTALIZAR, ya que la prueba de la comparación da como resultado que BETA es mayor que ALFA.

10.– Ninguna es correcta, ya que cuando se emplea la opción NEXT SENTENCE debe figurar inmediatamente a continuación de la CONDICION que sigue a IF, o bien inmediatamente después de la palabra ELSE; es decir, no puede haber declaración alguna entre ELSE y NEXT SENTENCE, o entre la condición que sigue a IF y NEXT SENTENCE.

11.–

```
010          IF ALFA NOT GREATER THAN BETA
020            IF ZETA EQUAL TO SIGMA
030              IF OMEGA < GAMMA GO TO 1
040              ELSE GO TO 2.

060          IF ALFA NOT > BETA AND ZETA = SIGMA
070                          AND OMEGA < GAMMA
080            GO TO 1
090          ELSE GO TO 2.
```

12.– 1) Permitirían la ejecución del procedimiento CALCULO–PRIMA las declaraciones 1ª, 3ª y 4ª.

2) La segunda declaración no lo permite, ya que para la primera prueba tomaría el ordenador, al operador AND y las condiciones que le rodean, es decir:
EDAD = 25 AND EDAD EQUAL TO 50 , lo cual es imposible, ya que nunca puede ocurrir que EDAD sea igual a 25 y 50 a la vez.
La modificación consiste en añadir los paréntesis como sigue:

```
010          IF (EDAD > 25 OR EDAD = 25)
020          AND (EDAD = 50 OR EDAD < 50)
030          PERFORM CALCULO– PRIMA.
```

13.– Que sería suficiente con ser cierta la condición ANNOS = 30 AND MM NOT < MES para que se imprimiera el REGISTRO–OPERARIO, lo cual no es correcto, pues para imprimir dicho registro es necesario que cumpla también otras condiciones.

14.– Imprimiría una línea con el contenido del registro LINEA y a continuación avanzaría el papel verticalmente tres espacios.

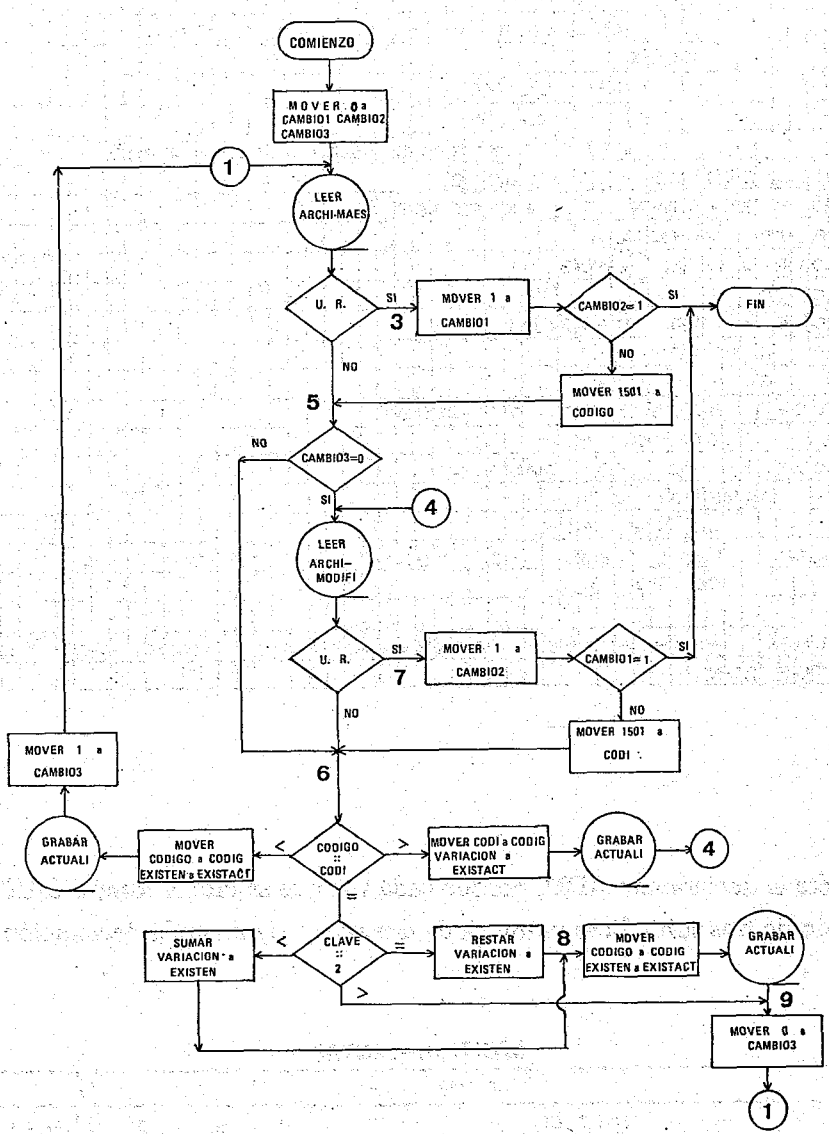
15.– a) Ejecutaría el procedimiento SUMAR ocho veces. Efectivamente,los valores sucesivos de BETA hubieran sido 1,4,7,10,13,16,19,22y25 si para este campo se hubieran reservado dos octetos, pero al reservarle un único octeto truncará la cifra de orden superior, por lo que los valores que irá tomando dicho campo serán: 1, 4, 7, 0, 3, 6, 9, 2 y 5 siendo con este último valor con el que cumple la condición, y por tanto deja de ejecutar el citado procedimiento.

b) ALFA = 90.

16.– Ocho veces con el primer y tercer PERFORM, y nueve veces con el segundo.

17.– Probablemente no se ejecutaría ninguna vez, ya que por no estar inicializado el campo BETA, sería casual que en los dos octetos de la memoria reservados para dicho campo, existiera un valor numérico en cuyo caso se ejecutaría tres veces el procedimiento.

18.– Para facilitar la comprensión de la PROCEDURE DIVISION se muestra a continuación el ordinograma de proceso :



La codificación sería la siguiente:

```
003010  PROCEDURE DIVISION.
003015    PROCESO.
003020          OPEN   INPUT ARCHI-MAES   ARCHI-MODIFI
003025                 OUTPUT  ARCHIACTUAL.
003030          MOVE   ZERO  TO  CAMBIO1    CAMBIO2    CAMBIO3.
003040     1.   READ   ARCHI-MAES  AT END  GO TO  3.
003045     5.   IF   CAMBIO3  NOT = 0    GO TO  6.
003050     4.   READ   ARCHI-MODIFI  AT END   GO TO  7.
003060     6.   IF   CODIGO > CODI
003065                 MOVE CODI  TO  CODIG
003070                 MOVE VARIACION  TO  EXISTACT
003075                 WRITE  ACTUALI
003080                 GO TO  4
003085          ELSE
003090                 IF   CODIGO < CODI
003095                      MOVE CODIGO  TO  CODIG
003100                      MOVE EXISTEN  TO  EXISTACT
003105                      WRITE  ACTUALI
003110                      MOVE 1  TO  CAMBIO3
003115                      GO TO  1
003120                 ELSE
003125                      IF   CLAVE < 2
003130                           ADD VARIACION  TO  EXISTEN
003135                      ELSE
003140                           IF   CLAVE > 2
003145                                GO TO  9
003150                           ELSE SUBTRACT VARIACION  FROM  EXISTEN.
003155     8.   MOVE CODIGO   TO   CODIG
003160          MOVE EXISTEN   TO EXISTACT
003165          WRITE  ACTUALI.
003170     9.   MOVE 0   TO   CAMBIO3
003175          GO TO  1.
003180     3.   MOVE 1   TO   CAMBIO1
003185          IF   CAMBIO2  =  1
003190               GO TO   FIN
003195          ELSE   MOVE 1501   TO   CODI
003200                 GO TO  5.
003205     7.   MOVE 1   TO   TO   CAMBIO2
003210          IF   CAMBIO1  =  1
003215               GO TO   FIN
003220          ELSE   MOVE 1501   TO   CODI
003225                 GO TO  6.
003230     FIN.
003235          CLOSE  ARCHI-MAES   ARCHI-ACTUAL   ARCHI-MODIFI
003240          STOP RUN.
```

CAPITULO VI.

1.– Es incorrecta la declaración ADD, porque cada vez que se cite el campo CANTIDAD, debe ir acompañado de tres subíndices y nunca de dos,como figura en la declaración.

2.– a)

REGISTRO–VENTAS

		A (1)																										A (2)		A (5)	
		D (1,1)																D (1, 2)				D (1, 4)							D (5, 4)		
B	C	E	F		F		F		F		F		F		F		E	F			F		F		B		F			
			G	H	G	H	G	H	G	H	G	H	G	H	G	H		G	H	G		H	G	H			G	H		
999	X(6)	9999	XX	999	XX	999	XX	999	XX	999	XX	999	XX	999	XX	999	9999	XX	999	XX		999	XX	999	999		XX	999		
PIC																															

b) El campo E está contenido: 5 x 4 = 20 veces; el campo H: 7 x 4 x 5 = 140.

c) E (4, 3)

d) El algoritmo para obtener los diferentes subíndices que acompañarán al elemento G es el siguiente:

Llamando N al número de orden que ocupa el elemento en la tabla, y b y c a los números que siguen a las cláusulas OCCURS del segundo y tercer nivel.

N | b . c [cociente por exceso]

−r A ⟶ Número del subíndice del primer nivel.

−r + b . c | c [cociente por exceso]

−r' B ⟶ Número del subíndice del segundo nivel.

−r' + c = C ⟶ Número del subíndice del tercer nivel.

De la aplicación del algoritmo se obtiene para G los siguientes subíndices:

```
125 | 28          −15 + 28 = 13 | 7          − 1 + 7 = [6]
−140  [5]         −14  [2]
 −15               −1
```

es decir, 5, 2, 6, y por tanto la declaración sería: DISPLAY G (5, 2, 6).

e) La fórmula a aplicar sería: (A−1) b . c + (B−1) c + C, donde el significado de cada letra es el mismo que en el apartado d).

Por tanto (4−1). 28 + (2−1) . 7 + 3 = 84 + 7 + 3 = 94

Es decir, el elemento H (4, 2, 3) ocupa el lugar 94 de todos los elementos H contenidos en la tabla.

3.−

```
130  01 TABLA-PRODUCCION.
140     02 FACTORIA OCCURS 6 TIMES.
150        03 IDENTIFICACION.
160           04 NOMBRE-FAC   PIC X(6).
170           04 DIRECCION    PIC X(12).
180        03 PRODUCCION.
190           04 MODELO       PIC X(7).
200           04 ANUAL        OCCURS 5 TIMES.
210              05 CAMION    PIC 99.
220              05 TRACTOR   PIC 99.
220              05 TURISMO   PIC 9(3).
```

4.− Es equivalente.

5.–

```
003070  PROCEDURE DIVISION.
003075    PROCESO.
003080          OPEN INPUT  FICHAS     OUTPUT IMPRESORA
003090          PERFORM LLENAR-TABLA VARYING A FROM 1 BY 1 UNTIL A = 9
003100          READ FICHAS AT END  MOVE 0 TO NUM-REGIS
.....           .....................................
004080    LLENAR-TABLA.
004090          MOVE 0 TO CONTADOR (A).
```

6.– No haría falta hacer modificación alguna, ya que una tabla indexada puede manejarse indistintamente empleando el índice que la ha indexado o un subíndice.

7.–

```
003070  PROCEDURE DIVISION.
003075    PROCESO.
003080          OPEN INPUT FICHAS     OUTPUT IMPRESORA
003090          MOVE ZEROS  TO  CONTADORES
003100          READ FICHAS  AT END   MOVE  0  TO  NUM-REGIS
003105          SET ALFA  TO  1
003110          PERFORM DESGLOSAR  UNTIL NUM-REGIS = 0
003120          PERFORM IMPRIMIR-CABECERA
003125          SET ALFA  TO  1
003130          PERFORM IMPRIMIR-TITULOS  VARYING A FROM 1 BY 1 UNTIL A = 9
003140          CLOSE  FICHAS    IMPRESORA
003150          STOP RUN.
003160    DESGLOSAR.
003170          PERFORM  DESGLOSE-PAGAS
003180                   VARYING  N  FROM  1  BY  1  UNTIL  N = 17
003190                   AFTER BETA  FROM  1  BY  1  UNTIL  BETA = 9.
003200          READ FICHAS  AT END  MOVE 0  TO  NUM-REGIS.
003210    DESGLOSE-PAGAS.
003220          DIVIDE  PAGA (N)  BY  BILLETE (BETA)
003230                  GIVING  COCIENTE  REMAINDER  PAGA (N)
003240          ADD COCIENTE  TO  CONTADOR (ALFA)
003250          SET ALFA  UP BY  1
003260          IF  ALFA = 9   SET ALFA  TO  1.
003270    IMPRIMIR-CABECERA.
004020          WRITE IMPRESO  FROM  CABECERA-A  AFTER  PAG-NUEVA.
004030    IMPRIMIR-TITULOS.
004040          MOVE TITULO (A)   TO  TITULIN
004050          MOVE CONTADOR (ALFA)  TO  TOTAL
004060          WRITE  IMPRESO  FROM  CABECERA-B  AFTER ADVANCING 2 LINES
004070          SET  ALFA  UP BY 1.
```

8.– No haría falta introducir ninguna modificación en la PROCEDURE DIVISION. Tampoco haría falta incializar el índice ALFA por medio de una declaración SET, ni ir variando su valor como sucedía con el subíndice,ya que de ello se encarga la PERFORM.

9.– Las declaraciones SET de las líneas 090 y 100 son incorrectas,ya que de acuerdo con el cuadro de transferencias de la pág. 270, no se permiten las transferencias que se pretenden hacer con dichas declaraciones.

10.–

```
050   01 TABLA-PRODUCCION.
060        02  MENSUAL    OCCURS 12  ASCENDING KEY IS MES.
070            03  MES      PIC 99.
080            03  DIARIA   OCCURS 20  DESCENDING KEY IS CANTI.
090                04  DIA     PIC 99.
100                04  CANTI  PIC 9(3).
```

11.– CIERTO.

12.– El error consiste en que al hacer la descripción de la TABLA no se ha especificado: ASCENDING KEY IS CODIGO.

13.– El primer error consiste en no ser correcto especificar: WHEN VENTAS (ALFA), ya que en vez de VENTAS debía haberse especificado NOMBRE, por estar la tabla ordenada en forma ascendente por dicho NOMBRE.
El segundo error consiste en haber especificado el símbolo > en vez del tipo de relación permitido: EQUAL TO, ó = .

14.– La descripción b) es correcta.
La descripción a) es incorrecta, ya que de acuerdo con el formato de la declaración DISPLAY no puede figurar un nombre–de–índice con dicha declaración. Por tanto no es correcto escribir DISPLAY ALFA, por ser ALFA un nombre–de–índice.

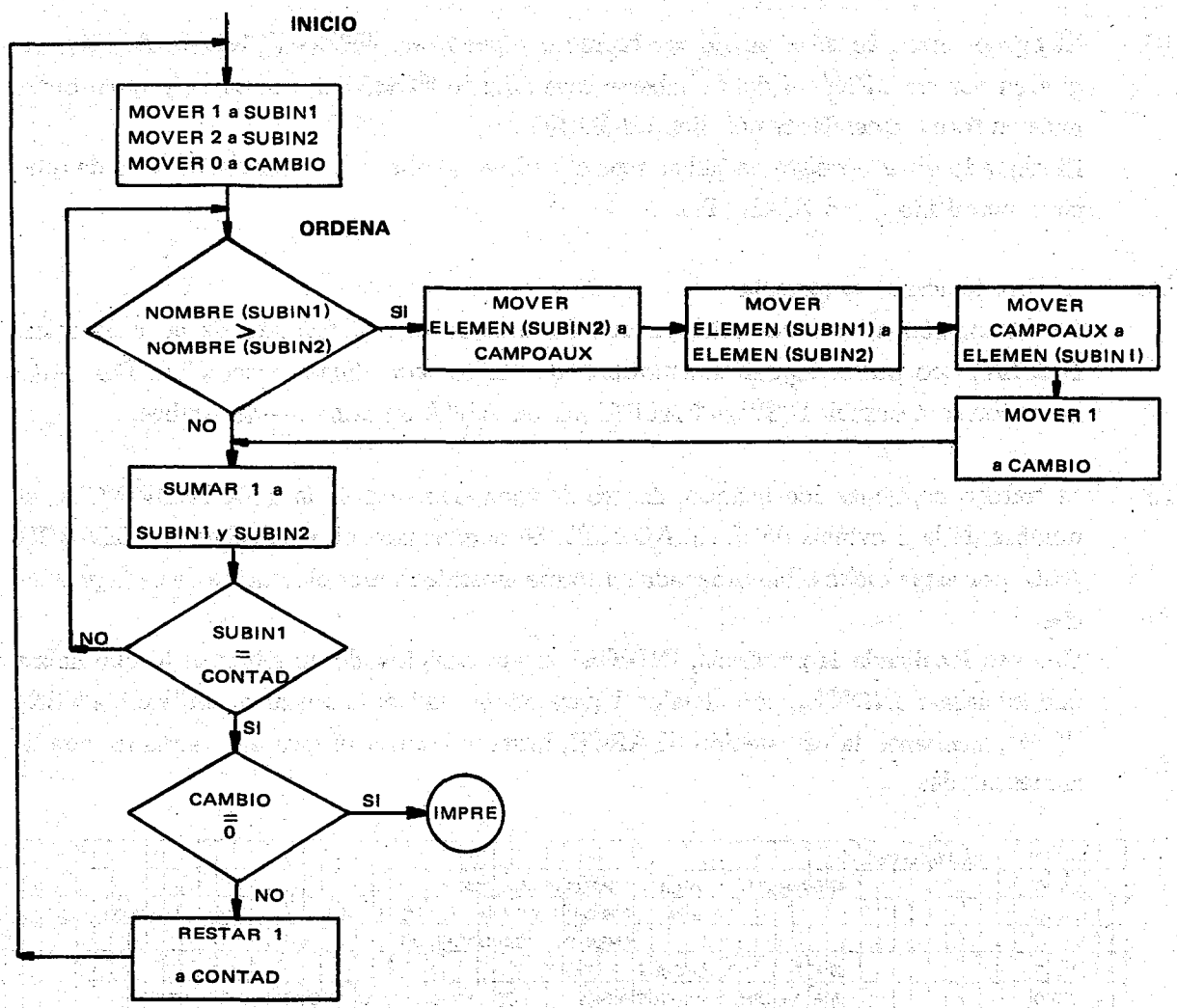
15.– El trabajo comienza localizando, dentro de cada elemento de la tabla PROVINCIA, el nombre de la provincia de VALLADOLID. Se emplea para ello la declaración SEARCH ALL por estar dicha tabla ordenada en forma ascendente por el nombre de cada provincia.
Una vez localizada la provincia, INDICE1 estará posicionado en ella, por lo que habrá que inicializar INDICE2 con el valor 1 para poder realizar la búsqueda en la tabla VENTASM, mediante la declaración SEARCH, hasta encontrar el mes que reuna las condiciones citadas.

```
010    BUSQUEDA.
020        SEARCH  ALL  PROVINCIA
030            WHEN  PROVINCIA (INDICE1) = 'VALLADOLID'
040                NEXT  SENTENCE.
050        SET  INDICE2  TO  1.
060        SEARCH  VENTASM  AT END  GO TO  NOEXIS
070            WHEN  AUTOS (INDICE1, INDICE2) > 1500
080                AND  CAMIONES (INDICE1, INDICE2) > 200
090                DISPLAY 'UN MES REUNE LAS CONDICIONES'.
```

16.–

```
002010   77  CONTAD      PIC 99  VALUE 45.
002011   77  CAMBIO      PIC 9.
.....
003150* ORDENACION ALFABETICA DE LA TABLA2
003160  INICIO.
003161          MOVE 1 TO SUBIN1
003162          MOVE 1 TO SUBIN2
003163          MOVE 0 TO CAMBIO.
003170  ORDENA.
003171          IF NOMBRE (SUBIN1) > NOMBRE (SUBIN2)
003180              MOVE ELEMEN (SUBIN2) TO CAMPOAUX
003190              MOVE ELEMEN (SUBIN1) TO ELEMEN (SUBIN2)
003200              MOVE CAMPOAUX TO ELEMEN (SUBIN1)
003201              MOVE 1 TO CAMBIO
003202          ELSE  NEXT SENTENCE.
003210          ADD 1 TO SUBIN1, SUBIN2
003220          IF  SUBIN1 = CONTAD
003230              IF CAMBIO = 0 GO TO IMPRE
003240              ELSE  SUBTRACT 1 FROM CONTAD
003250                    GO TO INICIO
003260          ELSE GO TO ORDENA.
004010* IMPRESION DE AGENTES Y CANTIDAD VENDIDA.
004020  IMPRE.
```

El siguiente ordinograma facilita la comprensión de la solución dada:



17.–

```
                                                                
 002031   77  CAMBIO          PIC 9.
 002032   77  CONTAD          PIC 99   VALUE 45.
                                                                
 003090* ORDENACION ALFABETICA DE LA TABLA2
 003100  INICIO.
 003110          MOVE ZERO  TO  CAMBIO
 003120          SET ALFA TO 1.
 003140  ORDENA.
 003150          IF  NOMBRE (ALFA) > NOMBRE (ALFA + 1)
 003160              MOVE ELEMEN (ALFA + 1) TO CAMPOAUX
 003170              MOVE ELEMEN (ALFA) TO ELEMEN (ALFA + 1)
 003171              MOVE CAMPOAUX TO ELEMEN (ALFA)
 003172              MOVE 1 TO CAMBIO
 003173          ELSE  NEXT SENTENCE.
 003174          SET ALFA UP BY 1
 003175          IF ALFA = 0
 003176              IF CAMBIO = 0  GO TO IMPRE
 003177              ELSE SUBTRACT 1 FROM CONTAD
 003178                   GO TO INICIO
 003179          ELSE GO TO ORDENA.
 003190* IMPRESION DE AGENTES Y CANTIDAD VENDIDA
 003200  IMPRE.
                                                                
```

1.-

```
000101 0  ID DIVISION.
000102 0  PROGRAM-ID. ACTUALIN.
000103 0  REMARKS. PROGRAMA PARA ACTUALIZAR AL AZAR EL ARCHIVO
000104 0                   CREADO CON ORGANIZACION SECUENCIAL INDEXADA.
000105 0  ENVIRONMENT DIVISION.
000106 0  INPUT-OUTPUT SECTION.
000107 0  FILE-CONTROL.
000108 0  SELECT ARCHIFICH  ASSIGN SYS005-UR-2540R-S
000109 0                    RESERVE 1 ALTERNATE AREA.
000110 0  SELECT ARCHIDISC  ASSIGN SYS006-DA-3340-I
000111 0                    ACCESS  RANDOM
000112 0                    RECORD KEY CODIDIS
000113 0                    NOMINAL KEY CODICLAVE.
000114 0  DATA DIVISION.
000115 0  FILE SECTION.
000116 0  FD ARCHIFICH  RECORDING F  LABEL RECORD OMITTED.
000117 0  01 FICHA.
000118 0  02 CODIFICH      PIC X(5).
000119 0  02 DATOSF        PIC X(74).
000120 0  02 CLAVEFICH     PIC 9.
000201 0  FD ARCHIDISC  BLOCK 10 RECORDS  LABEL RECORD STANDARD.
000202 0  01 DISCO.
000203 0  02 CODIDIS       PIC X(5).
000204 0  02 DATOSD        PIC X(115).
000205 0  WORKING-STORAGE SECTION.
000206 0  77 CODICLAVE     PIC X(5).
000207 0  PROCEDURE DIVISION.
000208 0  ABRIR. OPEN INPUT ARCHIFICH
000209 0                I-O   ARCHIDISC
000210 0                DISPLAY 'RELACION DE ERRORES EN EL ARCHIVO EN FICHAS'.
000211 0  LEER.  READ ARCHIFICH  AT END GO TO FIN.
000212 0                GO TO ALTA BAJA ACTUALIZAR DEPENDING ON CLAVEFICH.
000213 0                DISPLAY 'FICHA CON ERROR EN LA CLAVE DE LA COLUMNA 80 ='
000214 0                        FICHA
000215 0                GO TO LEER.
000215 5  ALTA.  MOVE CODIFICH TO CODICLAVE
000216 0                WRITE DISCO FROM FICHA  INVALID KEY  DISPLAY 'CODIGO DE
000217 0-               'L REGISTRO A DAR DE ALTA YA ESTA GRABADO EN EL ARCHIV
000218 0-               'O EN DISCO =' CODIFICH.
000219 0                GO TO LEER.
000301 0  BAJA.  MOVE CODIFICH  TO CODICLAVE
000302 0                READ ARCHIDISC  INVALID KEY  GO TO NOBAJA.
000303 0                MOVE  ALL '-' TO DATOSD
000304 0                REWRITE DISCO  INVALID  GO TO NOBAJA.
000305 0                GO TO LEER.
000306 0  ACTUALIZAR.
000307 0                MOVE CODIFICH  TO CODICLAVE
000308 0                READ ARCHIDISC INVALID KEY  GO TO NOACT.
000309 0                MOVE DATOSF  TO  DATOSD
000310 0                REWRITE DISCO  INVALID KEY  GO TO NOACT.
000311 0                GO TO LEER.
000312 0  NOBAJA. DISPLAY 'CODIGO DEL REGISTRO A DAR DE BAJA NO ESTA GRABA
000313 0-               'DO EN EL ARCHIVO EN DISCO =' CODIFICH
000314 0                GO TO LEER.
000315 0  NOACT.  DISPLAY 'CODIGO DEL REGISTRO A ACTUALIZAR NO ESTA GRABAD
000316 0-               'O EN EL ARCHIVO EN DISCO =' CODIFICH
000317 0                GO TO LEER.
000318 0  FIN.   CLOSE  ARCHIFICH  ARCHIDISC
000319 0                STOP RUN.
```

2.–

```
001010  ID DIVISION.
001020  PROGRAM-ID. ACTUREAL.
001030  REMARKS. ACTUALIZACION DE UN ARCHIVO EN DISCO EMPLEANDO DIRECCIO
001040                   NAMIENTO INDIRECTO CON DIRECCION REAL DE PISTA.
001050  ENVIRONMENT DIVISION.
001060  INPUT-OUTPUT SECTION.
001070  FILE-CONTROL.
001080      SELECT ARCHIDIS ASSIGN SYS005-DA-3340-A
001090                   ACTUAL KEY IS ACTUALY   ACCESS RANDOM.
001100      SELECT ARCHICIN ASSIGN SYS006-UT-2400-S.
001110  I-O-CONTROL.
001120      APPLY EXTENDED-SEARCH  ON ARCHIDIS.
001130  DATA DIVISION.
001140  FILE SECTION.
001150  FD ARCHIDIS  RECORDING F  LABEL RECORD STANDARD.
001160  01 DISCO.
001170      02 CLAVED        PIC X(6).
001180      02 DATOSD        PIC X(160).
001190  FD ARCHICIN RECORDING F BLOCK 10 RECORDS LABEL RECORD STANDARD.
001200  01 CINTA.
001210      02 CLAVEC        PIC X(6).
001220      02 DATOSC        PIC X(160).
002010  WORKING-STORAGE SECTION.
002020  77 CLAVE             PIC S9(6) COMP.
002030  77 COCIENTE          PIC 9(5).
002040  77 RESTO             PIC 9(4).
002050  01 DIRECCION.
002060      02 EXTENSION     PIC S999 COMP VALUE ZERO.
002070      02 CELDA         PIC S9 COMP   VALUE ZERO.
002080      02 CILINDRO      PIC S999 COMP.
002090      02 PISTA         PIC S99  COMP.
002100      02 REGISTRO      PIC X  VALUE LOW-VALUE.
002110      02 IDENTIF       PIC X(6).
002120  01 DIREC  REDEFINES  DIRECCION.
002130      02 FILLER        PIC X.
002140      02 ACTUALY       PIC X(14).
002150  PROCEDURE DIVISION.
002160  ABRIR. OPEN INPUT ARCHICIN  I-O ARCHIDIS.
002170  LEERCINTA.
002180      READ ARCHICIN AT END GO TO FIN.
002190      MOVE CLAVEC  TO  CLAVE
002200      DIVIDE CLAVE      BY 89 GIVING COCIENTE  REMAINDER RESTO
002210      ADD  2004 TO  RESTO
002220      DIVIDE 12  INTO RESTO GIVING CILINDRO REMAINDER  PISTA
002230      MOVE CLAVEC  TO  IDENTIF.
003010  LEERDISCO.
003020      READ ARCHIDIS INVALID KEY GO TO SINONIMO.
003030  ACTUALIZAR.   MOVE DATOSC  TO DATOSD
003040      WRITE ARCHIDIS INVALID KEY GO TO FIN.
003050      GO TO LEERCINTA.
003060  SINONIMO.
003070      MOVE 175 TO CILINDRO
003080      MOVE ZERO TO  PISTA
003090      READ ARCHIDIS INVALID KEY  DISPLAY CLAVEC 'CLAVE INVALIDA'
003100        GO TO FIN.
003110      PERFORM ACTUALIZAR.
003120  FIN.  CLOSE ARCHIDIS ARCHICIN   STOP RUN.
```

CAPITULO VIII

1.–

Hoja de Codificación en Cobol

PROGRAMADOR: | HOJA DE | SISTEMA | FECHA | INSTRUCCIONES PARA PERFORACION | SIGNO | PERFORACIONES | Identificación 73–80

PROGRAMA:

Página	Línea	C	A	B
001	010			ID DIVISION.
001	020			PROGRAM–ID. REPOSORT.
001	030			REMARKS. PROGRAMA EJEMPLO DE EMPLEO DE LA REPORT SECTION
001	040			Y DEL SORT.
001	050			ENVIRONMENT DIVISION.
001	060			CONFIGURATION SECTION.
001	070			SPECIAL–NAMES. DECIMAL POINT IS COMMA.
001	080			INPUT–OUTPUT SECTION.
001	090			FILE–CONTROL.
001	100			SELECT ARCHIOPE ASSIGN SYS004–UR–2540R–S.
001	110			SELECT ARCHISORT ASSIGN SYS001–DA–3340–S–SORTWK1.
001	120			SELECT ARCHIMPRE ASSIGN SYS005–UR–1403–S.
001	130			DATA DIVISION.
001	140			FILE SECTION.
001	150			FD ARCHIOPE LABEL RECORD OMITTED.
001	160			01 FICHOPE.
001	170			02 CODI PIC X(5).
001	180			02 APENOM PIC X(35).
001	190			02 FILLER PIC 9(6).
001	200			02 PROVINACI PIC 99.
001	210			02 FILLER PIC XX.
001	220			02 NUMHIJO PIC 99.
001	230			02 FILLER PIC X(6).
001	240			02 SUEL PIC 9(5).
001	250			02 FILLER PIC X(17).
002	010			SD ARCHISORT LABEL RECORD STANDARD.
002	020			01 CLASIF.
002	030			02 CODIGO PIC X(5).
002	040			02 APENOMBRE PIC X(35).
002	050			02 NUMHIJOS PIC 99.
002	060			02 SUELDO PIC 9(5).
002	070			02 PROVINCIA PIC 99.
002	080			FD ARCHIMPRE LABEL RECORD OMITTED
002	090			REPORT IMPRESO.
002	100			WORKING–STORAGE SECTION.
002	110			77 ANNO PIC 9999.
002	120			77 CONTINUA PIC X(14).
002	130			77 PROV PIC 99 VALUE ZEROS.
002	140			77 CAMBIO PIC 9 VALUE 1.
002	150			77 CONTO PIC 9(4) VALUE 1.
002	160			01 TABLAPROVIN.
002	170			02 PROVIN OCCURS 50 PIC X(16).

1.– (Continuación).

```
003010 REPORT SECTION.
003020   RD  IMPRESO  CONTROLS ARE  FINAL  PROVINCIA
003030                PAGE  61  LINES
003040                HEADING          3
003050                FIRST DETAIL    11
003060                LAST DETAIL     50
003070                FOOTING         54.
003080   01 TYPE REPORT HEADING.
003090      02 LINE 3.
003100         03 COLUMN  44  PIC X(46)
003110            VALUE 'INFORME RICOSA          AÑO '.
003120         03 COLUMN  90  PIC 9.999  SOURCE ANNO.
003130   01 CABECERA TYPE PAGE HEADING.
003140      02 LINE 6.
003150         03 COLUMN  30  PIC X(54)  VALUE 'RELACION ALFABETICA DE
003160-                                  'OPERARIOS DE RICOSA NACIDOS EN '.
003170         03 COLUMN  84  PIC X(16)  SOURCE PROVIN (PROVINCIA).
003180         03 COLUMN 101  PIC X(14)  SOURCE CONTINUA.
003190      02 LINE 8
003200         COLUMN  25  PIC X(92)
003210         VALUE 'CODIGO          APELLIDOS Y NOMBRE
003220-              '  NUM.HIJOS     SUELDO '.
003230      02 LINE 9
003235         COLUMN  24  PIC X(74)  VALUE ALL '='.
003240   01 LINEA-DETALLE TYPE DETAIL.
003250      02 LINE PLUS 2.
003260         03 COLUMN  25  PIC 9(5)  SOURCE CODIGO.
004010         03 COLUMN  35  PIC X(35)  SOURCE APENOMBRE.
004020         03 COLUMN  78  PIC Z9  SOURCE NUMHIJOS.
004030         03 COLUMN  88  PIC ZZ.ZZ9  SOURCE SUELDO.
004040   01 TYPE CONTROL FOOTING PROVINCIA
004050                          NEXT GROUP NEXT PAGE.
004060      02 LINE PLUS 1.
004070         03 COLUMN  24  PIC X(74)  VALUE ALL '-'.
004080      02 LINE PLUS 2.
004090         03 COLUMN  44  PIC X(27)
004095                       VALUE 'TOTAL NUMERO DE OPERARIOS ='.
004100         03 CONTADOR    PIC ZZ.ZZ9
004110                       COLUMN 72  SUM CONTO  UPON LINEA-DETALLE.
004120   01 TYPE CONTROL FOOTING FINAL.
004130      02 LINE 11.
004140         03 COLUMN  50  PIC X(32)
004150                       VALUE 'TOTAL OPERARIOS EN LA RELACION'.
004160         03 COM COLUMN  82  PIC ZZ.ZZ9  SUM CONTADOR.
004170      02 LINE PLUS 4
004180         COLUMN  60  PIC X(17)  VALUE 'FIN DEL INFORME'.
004190      02 LINE PLUS 2
004195         COLUMN  24  PIC X(93)  VALUE ALL '*'.
004200   01 TYPE PAGE FOOTING.
004210      02 LINE 56.
004220         03 COLUMN 100  PIC X(6)  VALUE 'PAGINA'.
004230         03 COLUMN 107  PIC ZZZ9  SOURCE PAGE-COUNTER.
```

1.– (Continuación)

```
005010  PROCEDURE DIVISION.
005020    DECLARATIVES.
005030      REP SECTION.
005040          USE BEFORE REPORTING CABECERA.
005045        ALFA.
005050          IF CAMBIO = 1 GO TO BETA
005055          ELSE GO TO NOIMP.
005057        BETA.
005060          IF PROVINCIA = PROV
005065            MOVE '(CONTINUACION)' TO CONTINUA
005070            GO TO SALI
005080          ELSE
005085            MOVE PROVINCIA TO PROV
005090            MOVE SPACES TO CONTINUA
005095            GO TO SALI.
005097        NOIMP.
005100          MOVE 1 TO PRINT-SWITCH.
005110        SALI. EXIT.
005120    END DECLARATIVES.
005130    SOR SECTION.
005140        SORT ARCHISORT
005150          ASCENDING KEY PROVINCIA APENOMBRE
005160          DESCENDIG KEY CODIGO
005170          INPUT PROCEDURE ENTRADA
005180          OUTPUT PROCEDURE SALIDA.
005185          STOP RUN.
005190    ENTRADA SECTION.
005200      ABRIR.
005205          OPEN INPUT ARCHIOPE
005210          ACCEPT TABLAPROVIN
005220          ACCEPT ANNO.
005225      LEER.
006010          READ ARCHIOPE AT END CLOSE ARCHIOPE GO TO SAL1.
006020          IF NUMHIJO = 0 OR SUEL < 20000 GO TO LEER.
006030          MOVE CODI TO CODIGO
006040          MOVE APENOM TO APENOMBRE
006045          MOVE NUMHIJO TO NUMHIJOS
006048          MOVE SUEL TO SUELDO
006050          MOVE PROVINACI TO PROVINCIA
006060          RELEASE CLASIF
006065          GO TO LEER.
006070      SAL1. EXIT.
006080    SALIDA SECTION.
006090          OPEN OUTPUT ARCHIMPRE
006100          INITIATE IMPRESO.
006110      RELEER.
006120          RETURN ARCHISORT AT END GO TO FINIMP.
006130          GENERATE LINEA-DETALLE
006140          GO TO RELEER.
006150      FINIMP.
006160          MOVE ZERO TO CAMBIO
006170          TERMINATE IMPRESO.
006180          CLOSE ARCHIMPRE.
006190      SAL2. EXIT.
```

CAPITULO IX

1.– c. (Párrafo 2, pág. 62).
2.– d. (Párrafo 6, pág. 71).
3.– a. Observar PROGRAN–ID; el resto son palabras reservadas.
4.– c. Por estar entre comillas (ver párrafo 2, pág. 63).
5.– b. (Párrafo 5, pág. 105).
6.– c.
7.– d. (Párrafo 1, pág. 138).
8.– b. (Línea 8, pág. 124).
9.– d. Ya que truncaría la cifra de los millares, es decir, la cifra 3.
10.– c. Por ser la división 50:1000.
11.– a. Ya que en una PICTURE con tres caracteres + sólo puede almacenarse un número con dos cifras enteras. Situandose el signo + delante, siempre que el número sea positivo (penúltimo párrafo pág. 121 y apartado 2 de la pág. 122).
12.– c. Ver tabla de clasificación (pág. 184).
13.– b.
14.– c.
15.– a.
16.– c. (Pág. 214).
17.– c. Los valores que va tomando BETA son 0, 7, 14, 28, 35, 42, 49 y cuando pasa a valer 56 se cumple la condición, por lo que no se ejecuta más veces el procedimiento CALCULO; por tanto se ha ejecutado dicho procedimiento 7 veces, durante las cuales el CAMPOA ha pasado por los siguientes valores 2, 4, 8, 16, 32, 64, 28 (trunca por ser PIC 99) y 56.
18.– b. Por no haber descrito el campo BETA con PIC S9, el valor con el que se inicializaría el campo BETA sería 8 en vez de –8, que al no cumplir la condición BETA = 0 obligaría a ejecutar el procedimiento ALFA una vez. Cuando se incremente el valor del campo BETA con el valor que sigue a BY, es decir, 2, BETA debería valer 10, pero como se ha definido con PIC 9 reservaría una sola posición de memoria y, por truncamiento, el valor de BETA sería 0; con lo que se cumple la condición y, por tanto, no se ejecutaría más veces dicho procedimiento.
19.– a.
20.– c.
21.– c. (Pág. 265).
22.– d. (Ultimo párrafo de la pág. 277).
23.– c.
24.– d.

APENDICE B

PALABRAS RESERVADAS COBOL

A

ACCEPT	Aceptar
ACCESS	Acceso
ACTUAL	Real
ADD	Sumar
ADVANCING	Avanzar
AFTER	Después de
ALL	Todo
ALPHABETIC	Alfabético
ALTER	Alterar
ALTERNATE	Alternativa
AND	Y
APPLY	Aplicar
ARE	Son
AREA	Area
AREAS	Areas
ASCENDING	Ascendente
ASSIGN	Asignar
AT	En, al
AT END	Al final
AUTHOR	Autor

B

BEFORE	Antes de
BEGINNING	Comienzo; cabecera
BLANK	Blanco
BLOCK	Bloque
BY	Por

C

CALL	Llamar (a un programa)
CANCEL	Cancelar
CF	Pie de control
CH	Encabezamiento de control
CHANGED	Cambiado
CHARACTER	Carácter
CHECKING	Comprobación; Verificación
CLOSE	Cerrar
COBOL	COBOL
CODE	Codigo
COLUMN	Columna
COMMA	Coma
COMPUTATIONAL (COMP)	Formato Binario
COMPUTATIONAL—1 (COMP—1)	Formato punto flotante interno corto
COMPUTATIONAL—2 (COMP—2)	Formato punto flotante interno ampliado
COMPUTATIONAL—3 (COMP—3)	Formato empaquetado
COMPUTE	Calcular
CONFIGURATION	Configuración
CONSOLE	Consola
CONTAINS	Contiene
CONTROL	Control
COPY	Copiar
CORRESPONDING (CORR)	Corresponde
COUNT	Cuenta
CURRENCY	Moneda
CURRENT—DATE	Fecha del día

D

DATA	Datos
DATE—COMPILED	Fecha de Compilación
DATE—WRITTEN	Fecha de Escritura
DE	Grupo de Detalle
DECIMAL—POINT	Punto Decimal
DECLARATIVES	Declarativas
DEPENDING	Dependiendo
DESCENDING	Descendente
DETAIL	Detalle
DIRECT	Directo
DIRECT—ACCESS	Acceso Directo
DISPLAY	Representar. Exhibir
DISPLAY—ST	Display—Libras Esterlinas
DIVIDE	Dividir
DIVISION	División
DOWN	Debajo

E

ELSE	En caso contrario
END	Fin
ENDING	Terminación
END—OF—PAGE (EOP)	Final de página
ENTER	Introducir, entrar
ENTRY	Inscripción, entrada
ENVIRONMENT	Características del Equipo
EQUAL (S)	Igual
ERROR	Error
EVERY	Cada, todos
EXAMINE	Examinar
EXHIBIT	Exhibir
EXIT	Salida
EXTENDED—SEARCH	Búsqueda ampliada

F

FD	Descripción de Archivo
FILE (S)	Archivo
FILE—CONTROL	Control de Archivo
FILLER	Relleno
FINAL	Final
FIRST	Primer, primero
FOOTING	Pie
FOR	Para
FORM—OVERFLOW	Capacidad excedida de formulario
FROM	Desde, de

G

GENERATE	Generar
GIVING	Dando como resultado
GO	Ir, bifurcar
GREATER	Mayor que
GROUP	Grupo

H

HEADING	Encabezamiento
HIGH—VALUE	Valor más elevado
HIGH—VALUES	Valores más elevados

I

IDENTIFICATION (ID)	Identificación
IF	Si (condicional)
IN	En
INCLUDE	Incluir
INDEX	Indice
INDEXED	Indexado
INDICATE	Indicar
INITIATE	Iniciar
INPUT	Entrada
INPUT—OUTPUT	Entrada—Salida
INSTALLATION	Instalación
INTO	En, dentro de, por
INVALID	Inválido
I—O	Entrada—Salida
I—O—CONTROL	Control de Entrada—Salida
IS	Es

J

JUSTIFIED (JUST)	Justificado, Alineado

K

KEY (S)	Clave

L

LABEL (S)	Etiqueta
LAST	Ultimo
LEADING	Que aparece al principio
LEFT	Izquierda

LESS . . . . . Menos, menor que
LIBRARY . . . . . Biblioteca
LIMIT (S) . . . . . Límite
LINE (S) . . . . . Líneas
LINKAGE . . . . . Enlace, unión
LOCK . . . . . Bloqueo
LOW—VALUE (S) . . . . . Valor más bajo

M

MEMORY . . . . . Memoria
MODE . . . . . Modalidad
MORE—LABELS . . . . . Más etiquetas
MOVE . . . . . Mover, transferir
MULTIPLE . . . . . Multiple
MULTIPLY . . . . . Multiplicar

N

NAMED . . . . . Denominado
NEGATIVE . . . . . Negativo
NEXT . . . . . Siguiente
NO . . . . . No
NOMINAL . . . . . Nominal
NOT . . . . . No
NOTE . . . . . Nota, Comentario
NUMBER . . . . . Número
NUMERIC . . . . . Numérico

O

OBJECT—COMPUTER . . . Ordenador—Objeto
OCCURS . . . . . Se presenta, ocurre
OF . . . . . De
OFF . . . . . Fuera, Desconectado
OMITTED . . . . . Omitido
ON . . . . . En, de
OPEN . . . . . Abrir
OR . . . . . O
ORGANIZATION . . . . . Organización
OTHERWISE . . . . . De lo contrario
OUTPUT . . . . . Salida
OVERFLOW . . . . . Exceder

P

PAGE . . . . . Página
PAGE—COUNTER . . . . . Contador de páginas
PERFORM . . . . . Ejecutar
PF . . . . . Pié de página
PH . . . . . Encabezamiento de página
PICTURE (PIC) . . . . . Descripción, formato, modelo
PLUS . . . . . Más
POSITIONING . . . . . Posición
POSITIVE . . . . . Positivo
PROCEDURE . . . . . Procedimiento
PROCEED . . . . . Proseguir
PROCESS (ING) . . . . . Procesar
PROGRAM . . . . . Programa
PROGRAM—ID . . . . . Identificación del programa
PROTECTION . . . . . Protección

Q

QUOTE . . . . . Comilla
QUOTES . . . . . Comillas

R

RANDOM . . . . . Aleatorio
RD . . . . . Descripción de Informe
READ . . . . . Leer
READY . . . . . Preparado, listo
RECORD (S) . . . . . Registro
RECORDING . . . . . Registrar
REDEFINES . . . . . Redefine
REEL . . . . . Carrete, rollo
RELEASE . . . . . Versión
REMAINDER . . . . . Resto
REMARKS . . . . . Observaciones
RENAMES . . . . . Volver a dar nombre
REPLACING . . . . . Sustitución, sustituir
REPORT (S) . . . . . Informe
RERUN . . . . . Nueva pasada
RESERVE . . . . . Reservar
RESET . . . . . Restaurar
RESTRICTED . . . . . Limitado, Restringido
RETURN . . . . . Retornar
REVERSED . . . . . Invertido
REWIND . . . . . Rebobinar
REWRITE . . . . . Volver a grabar
RF . . . . . Pié del Informe
RH . . . . . Encabezamiento del Informe
RIGHT . . . . . Derecha
ROUNDED . . . . . Redondeado
RUN . . . . . Ejecución de un programa

S

SAME . . . . . Igual, misma
SEARCH . . . . . Investigar, buscar
SECTION . . . . . Sección
SECURITY . . . . . Seguridad
SD . . . . . Descripción de clasificación
SELECT . . . . . Seleccionar
SENTENCE . . . . . Sentencia
SEPARATE . . . . . Separar
SEQUENTIAL . . . . . Secuencial
SET . . . . . Colocar
SIGN . . . . . Signo
SIZE . . . . . Tamaño, Longitud
SOURCE . . . . . Fuente
SOURCE—COMPUTER . . Ordenador—Fuente
SORT . . . . . Clasificar
SPACE . . . . . Espacio, blanco
SPACES . . . . . Espacios, blancos
SPECIAL—NAMES . . . . . Nombres especiales
STANDARD . . . . . Standard
START . . . . . Empezar
STOP . . . . . Parar, parada, fin
SUBTRACT . . . . . Restar
SUM . . . . . Sumar
SYMBOLIC . . . . . Simbólico
SYNCHRONIZED (SYNC). Sincronizado
SYSPUNCH . . . . . Salida de Perforación

T

TALLY . . . . . Item de datos binarios
TALLYING . . . . . Contando
TAPE . . . . . Cinta
TERMINATE . . . . . Terminar
THAN . . . . . Que
THEN . . . . . Entonces, luego
THRU . . . . . Hasta
TIME (S) . . . . . Veces
TO . . . . . A, hasta
TRACE . . . . . Seguimiento
TRACK—AREA . . . . . Area de Pista
TRACKS . . . . . Pistas
TRANSFORM . . . . . Transformar
TYPE . . . . . Tipo

U

UNIT . . . . . Unidad
UNIT—RECORD . . . . . Registro—Unitario
UNTIL . . . . . Hasta
UP . . . . . Sobre
UPON . . . . . Sobre
USAGE . . . . . Uso
USE . . . . . Utilizar
USING . . . . . Utilizar, utilizando
UTILITY . . . . . Utilidad

V

VALUE (S) . . . . . Valor
VARYING . . . . . Variando

W

WHEN . . . . . Cuando
WITH . . . . . Con
WITHOUT . . . . . Sin
WORKING—STORAGE . . Memoria de Trabajo
WRITE . . . . . Escribir, grabar
WRITE—ONLY . . . . . Escribir, grabar solamente

Z

ZERO . . . . . Cero
ZEROES . . . . . Ceros
ZEROS . . . . . Ceros

APENDICE C

RESUMEN DE FORMATOS (a)

IDENTIFICATION DIVISION

```
{ IDENTIFICATION DIVISION. }
{ ID DIVISION.             }
PROGRAM-ID. nombre-de-programa.
AUTHOR. [entrada-de-comentarios] ...
INSTALLATION. [entrada-de-comentarios] ...
DATE-WRITTEN. [entrada-de-comentarios] ...
DATE-COMPILED. [entrada-de-comentarios] ...
SECURITY. [entrada-de-comentarios] ...
REMARKS. [entrada-de-comentarios] ...
```

ENVIRONMENT DIVISION

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. nombre-de-ordenador.
OBJECT-COMPUTER. nombre-de-ordenador.
SPECIAL-NAMES. [nombre-de-función-1 IS nombre-mnemotécnico] ...
    [DECIMAL-POINT IS COMMA].
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    { SELECT [OPTIONAL] nombre-de-archivo
      ASSIGN TO [entero] nombre-de-equipo-1 [nombre-de-equipo-2] ... [FOR MULTIPLE { REEL } ]
                                                                                   { UNIT }
      RESERVE { entero } ALTERNATE [ AREA  ]
              { NO    }            [ AREAS ]
      ACCESS MODE IS { SEQUENTIAL }
                     { RANDOM     }
      ACTUAL KEY IS nombre-de-datos
      NOMINAL KEY IS nombre-de-datos
      RECORD KEY IS nombre-de-datos. } ...
I-O-CONTROL.
      RERUN ON nombre-de-sistema EVERY entero RECORDS OF nombre-de-archivo
      APPLY CYL-OVERFLOW OF entero TRACKS ON nombre-de-archivo-1 [nombre-de-archivo-2] ...
      APPLY EXTENDED-SEARCH ON nombre-de-archivo-1 [nombre-de-archivo-2] ...
      SAME [ RECORD ] AREA FOR nombre-de-archivo-1 { nombre-de-archivo-2 } ...
           [ SORT   ]
```

DATA DIVISION

```
DATA DIVISION
FILE SECTION
FD nombre-de-archivo
      BLOCK CONTAINS entero { CHARACTERS }
                            { RECORDS    }
      RECORD CONTAINS entero CHARACTERS
      RECORDING MODE IS modalidad
      LABEL { RECORD IS  } { OMITTED       }
            { RECORDS ARE } { STANDARD     }
                            { nombre-de-datos-1 }
      DATA { RECORD IS  } nombre-de-datos-1 [nombre-de-datos-2] ...
           { RECORDS ARE }
```

(a) En este resumen de formatos se incluye el formato específico de cada cláusula y declaración como si fuera necesaria su inclusión dentro de un programa COBOL, aunque muchas de ellas por ser optativas no será necesario incluirlas dentro de determinados programas.

DESCRIPCION DE REGISTRO

```
01-49 { nombre-de-datos-1 }
      { FILLER            }
      REDEFINES nombre-de-datos-2
      BLANK WHEN ZERO
      { JUSTIFIED } RIGHT
      { JUST      }
      { PICTURE   }
      { PIC       } IS serie-de-caracteres

      [USAGE IS] { INDEX                 }
                 { DISPLAY               }
                 { COMPUTATIONAL   }     }
                 { COMP            }     }
                 { COMPUTATIONAL-1 }     }
                 { COMP-1          }     }
                 { COMPUTATIONAL-2 }     }
                 { COMP-2          }     }
                 { COMPUTATIONAL-3 }     }
                 { COMP-3          }     }

88 nombre-de-condición { VALUE IS  } literal-1 [THRU literal-2] [ literal-3 [THRU literal-4]] ...
                       { VALUES ARE }

66 nombre-de-datos-1 RENAMES nombre-de-datos-2 [THRU nombre-de-datos-3]
```

NOTA: Los formatos de la cláusula OCCURS se incluyen con los formatos de la característica de manejo de tablas.

WORKING-STORAGE SECTION

77 nombre-de-datos-1

NOTA: A continuación se repiten todas las entradas de la Descripción de Registros.

LINKAGE SECTION

NOTA: Las mismas cláusulas que la WORKING-STORAGE SECTION.

PROCEDURE DIVISION

PROCEDURE DIVISION.

Declaración ACCEPT

```
ACCEPT identificador [FROM { SYSIPT                 }]
                           { CONSOLE                }
                           { nombre-mnemotécnico    }
```

Declaración ADD

Formato-1

```
ADD { Identificador-1 } [ Identificador-2 ] ... TO identificador-m [ROUNDED]
    { Literal-1       }   Literal-2 ]
[ identificador-n [ROUNDED] ] ... [ON SIZE ERROR declaración-imperativa]
```

Formato-2

```
ADD { Identificador-1 } { Identificador-2 } [ Identificador-3 ] ... GIVING identificador-m [ROUNDED]
    { Literal-1       } { Literal-2       }   Literal-3 ]
[ON SIZE ERROR declaración-imperativa]
```

Formato-3

```
ADD { CORRESPONDING } identificador-1 TO identificador-2 [ROUNDED]
    { CORR         }
    [ON SIZE ERROR declaración-imperativa]
```

Declaración ALTER

```
ALTER nombre-de-procedimiento-1 TO [PROCEED TO] nombre-de-procedimiento-2
    [ nombre-de-procedimiento-3 TO [PROCEED TO] nombre-de-procedimiento-4 ] ...
```

Declaración CALL

```
CALL literal [USING identificador-1 [identificador-2]... ]
```

Declaración CLOSE

```
CLOSE nombre-de-archivo-1 [ { REEL } [WITH { NO REWIND } ] ]
                          { UNIT }        { LOCK      }
    [ nombre-de-archivo-2 [ { REEL } [WITH { NO REWIND } ] ] ] ...
                          { UNIT }        { LOCK      }
```

Declaración COMPUTE

```
COMPUTE identificador-1 [ROUNDED] = { expresión-aritmética }
                                    { identificador-2      }
                                    { literal-1            }
    [ON SIZE ERROR declaración-imperativa]
```

DECLARATIVES SECTION

```
PROCEDURE DIVISION.
DECLARATIVES.
    { nombre-de-sección SECTION. Sentencia USE.
    { nombre-de-párrafo. {sentencia}... } ... } ...
END DECLARATIVES.
```

Declaración DISPLAY

```
DISPLAY { identificador-1 } [ identificador-2 ] ... [UPON { CONSOLE                 }]
        { literal-1       }   { literal-2       }         { SYSLST                  }
                                                          { nombre-mnemotécnico     }
```

Declaración DIVIDE

Formato-1

```
DIVIDE { identificador-1 } INTO identificador-2 [ROUNDED]
       { literal         }
    [ON SIZE ERROR declaración-imperativa]
```

Formato-2

```
DIVIDE { identificador } { INTO } { identificador-2 } GIVING identificador-3 [ROUNDED]
       { literal-1     } { BY   } { literal-2       }
    [REMAINDER identificador-4] [ON SIZE ERROR declaración-imperativa]
```

Declaración ENTER

```
ENTER nombre-de-lenguaje
```

Declaración ENTRY

```
ENTRY literal [USING identificador-1 [identificador-2] ... ]
```

Declaración EXAMINE

Formato-1

```
EXAMINE identificador TALLYING { UNTIL FIRST } carácter-1 [REPLACING BY carácter-2]
                               { ALL         }
                               { LEADING     }
```

Formato-2

```
EXAMINE identificador REPLACING { ALL         } carácter-1 BY carácter-2
                                { LEADING     }
                                { FIRST       }
                                { UNTIL FIRST }
```

Declaración EXIT

```
nombre-de-párrafo. EXIT [PROGRAM].
```

Declaración GO

Formato-1

```
GO TO[nombre-de-procedimiento-1]
```

Formato-2

```
GO TO nombre-de-procedimiento-1 [nombre-de-procedimiento-2] ...
      DEPENDING ON identificador
```

Declaración IF

```
IF condición [THEN] { declaración-1 ... } ELSE { declaración-2 ... }
                    { NEXT SENTENCE     }      { NEXT SENTENCE     }
```

Declaración MOVE

Formato-1

```
MOVE { identificador-1 } TO identificador-2 [identificador-3] ...
     { literal         }
```

Formato-2

```
MOVE { CORRESPONDING } identificador-1 TO identificador-2
     { CORR          }
```

Declaración MULTIPLY

Formato-1

```
MULTIPLY { identificador-1 } BY identificador-2 [ROUNDED]
         { literal         }
  [ON SIZE ERROR declaración-imperativa]
```

Formato-2

```
MULTIPLY { identificador-1 } BY { identificador-2 } GIVING identificador-3 [ROUNDED]
         { literal-1       }    { literal-2       }
  [ON SIZE ERROR declaración-imperativa]
```

Declaración NOTE

```
NOTE serie-de-caracteres
```

Declaración OPEN

```
OPEN [INPUT {nombre-de-archivo [WITH NO REWIND] }... ]
   [ {OUTPUT nombre-de-archivo [WITH NO REWIND] }... ]
   [ {I-O nombre- de-archivo }... ]
```

Declaración PERFORM

Formato-1

```
PERFORM nombre-de-procedimiento-1 [THRU nombre-de- procedimiento-2 ]
```

Formato-2

```
PERFORM nombre-de-procedimiento-1 [THRU nombre-de-procedimiento-2 ] { identificador } TIMES
                                                                    { entero       }
```

Formato-3

```
PERFORM nombre-de-procedimiento-1 [THRU nombre-de-procedimiento-2 ] UNTIL condición
```

Formato-4

```
PERFORM nombre-de-procedimiento-1 [THRU nombre-de-procedimiento-2]
    VARYING { nombre-de-índice-1 } FROM { nombre-de-índice-2 }
            { identificador-1    }      { literal-2       }
                                        { identificador-2 }
    BY      { literal-3          }
            { identificador-3    }
    UNTIL condición-1
  [ AFTER   { nombre-de-índice-4 } FROM { nombre-de-índice-5 }
            { identificador-4    }      { literal-5       }
                                        { idendificador-5 }
    BY      { literal-6          }
            { identificador-6    }
    UNTIL condición-2
  [ AFTER   { nombre-de-índice-7 } FROM { nombre-de-índice-8 }
            { identificador-7    }      { literal-8       }
                                        { identificador-8 }
    BY      { literal-9          }
            { identificador-9    }
    UNTIL condición-3 ]]
```

Declaración READ

```
READ nombre-de-archivo RECORD [INTO identificador] { AT END          } declaración-
                                                   { INVALID KEY     } imperativa
```

Declaración REWRITE

```
REWRITE nombre-de-registro [FROM identificador][INVALID KEY declaración-imperativa]
```

Declaración START

```
START nombre—de—archivo [INVALID KEY declaración imperativa]
```

Declaración STOP

```
STOP { RUN     }
     { literal }
```

Declaración SUBTRACT

Formato—1

```
SUBTRACT { identificador-1 } [ identificador-2 ]
         { literal-1       } [ literal-2       ]  ... FROM identificador-m [ROUNDED]
  [identificador- n [ROUNDED]] ... [ON SIZE ERROR declaración-imperativa]
```

Formato-2

```
SUBTRACT { identificador-1 } [ identificador-2 ] ... FROM { identificador-m }
         { literal-1       } [ literal-2       ]        { literal-m       }
  GIVING identificador-n [ROUNDED][ON SIZE ERROR declaración-imperativa]
```

Formato-3

```
SUBTRACT { CORRESPONDING } identificador-1 FROM identificador-2 [ROUNDED]
         { CORR          }
  [ON SIZE ERROR declaración-imperativa]
```

Declaración WRITE

Formato-1

```
WRITE nombre-de-registro [FROM identificador-1 ] [ { BEFORE } ]
                                                 { AFTER  }

  ADVANCING { identificador-2 LINES } ] [AT { END-OF-PAGE } declaración-imperativa]
            { entero LINES          }       { EOP        }
            { nombre-mnemotécnico   }
```

Formato-2

```
WRITE nombre-de-registro [ FROM identificador ] INVALID KEY declaración-imperativa
```

***Sentencia* USE**

Formato

```
USE AFTER STANDARD ERROR PROCEDURE

  ON { {nombre—de—archivo—1 } ...                                                  }
     { nombre—de—archivo—2 GIVING nombre—de—datos—1 [nombre—de—datos—2 ]           }
```

MANEJO DE TABLAS

Formatos de la DATA DIVISION

Cláusula OCCURS

Formato-1

```
OCCURS entero-2 TIMES [ { ASCENDING  } ] KEY IS nombre-de-datos-2 [nombre-de-datos-3]... ] ...
                      { DESCENDING }
    [INDEXED BY nombre-de-índice-1 [ nombre-de-índice-2] ... ]
```

Formato-2

```
OCCURS entero-1 TO entero-2 TIMES [DEPENDING ON nombre-de-datos-1] [ { ASCENDING  } ]
                                                                    { DESCENDING }
    KEY IS nombre-de-datos-2 [ nombre-de-datos-3] ... ]...
    [INDEXED BY nombre-de-índice-1 [nombre-de-índice-2] ... ]
```

Cláusula USAGE

```
[USAGE IS ] INDEX
```

Formatos de la PROCEDURE DIVISION

Declaración SEARCH

Formato-1

```
SEARCH identificador-1 [VARYING { nombre-de-índice-1 } ] [AT END declaración-imperativa-1 ]
                                { identificador-2    }
        WHEN condición-1 { declaración-imperativa-2 }
                         { NEXT SENTENCE            }
    [   WHEN condición-2 { declaración-imperativa-3 } ]...
                         { NEXT SENTENCE            }
```

Formato-2

```
SEARCH ALL identificador-1 [AT END declaración-imperativa— 1 ]
        WHEN condición-1 { declaración-imperativa-2 }
                         { NEXT SENTENCE            }
```

Declaración SET

Formato-1

```
SET { nombre-de-índice-1 [ nombre-de-índice-2]... } TO { nombre-de-índice-3 }
    { identificador-1    [ identificador-2 ] ...   }    { identificador-3   }
                                                        { literal-1         }
```

Formato-2

```
SET nombre-de-índice-4 [nombre-de-índice-5]... { UP BY   } { identificador-4 }
                                               { DOWN BY } { literal-2       }
```

CLASIFICACION (SORT)

Formato de la ENVIRONMENT DIVISION

```
FILE-CONTROL.

    SELECT nombre-de-archivo

        ASSIGN TO entero [nombre-de-equipo-1] [nombre-de-equipo-2] ...

        [FOR MULTIPLE { REEL }]
                      { UNIT }

        [RESERVE { entero-2 } ALTERNATE [ AREA  ]].
                 { NO       }           [ AREAS ]
```

Formato de la DATA DIVISION

DESCRIPCION DE ARCHIVOS

```
SD nombre-de-archivo-de-clasificación

    RECORDING MODE IS modalidad

    DATA { RECORD IS  } nombre-de-datos-1 nombre-de-datos-2 ...
         { RECORDS ARE}

    RECORD CONTAINS [entero-1 TO] entero-2 CHARACTERS

    LABEL { RECORD IS  } { STANDARD } .
          { RECORDS ARE} { OMITTED  }
```

Formatos de la PROCEDURE DIVISION

Declaración RELEASE

```
RELEASE nombre-de-registro-de-clasificación [FROM identificador]
```

Declaración RETURN

```
RETURN nombre-de-archivo-de-calificación RECORD [INTO identificador]
        AT END declaración-imperativa
```

Declaración SORT

```
SORT nombre-de-archivo-1 ON { DESCENDING } KEY {nombre-de-datos-1} ...
                            { ASCENDING  }

    [ON { DESCENDING } KEY {nombre-de-datos-2} ... ] ...
       { ASCENDING  }

   { INPUT PROCEDURE IS nombre-de-sección-1 [THRU nombre-de-sección-2] }
   { USING nombre-de-archivo-2                                         }

   { OUTPUT PROCEDURE IS nombre-de-sección-3 [THRU nombre-de-sección-4] }
   { GIVING nombre-de-archivo-3                                        }
```

ESCRITOR DE INFORMES

Formatos de la DATA DIVISION

FILE SECTION

Cláusula REPORT

```
{ REPORT IS  }
{ REPORTS ARE } nombre-de-informe-1 [nombre-de-informe-2] ...
```

REPORT SECTION

```
RD nombre-de-informe

    WITH CODE nombre-mnemotécnico

{ CONTROL IS  } { FINAL                                              }
{ CONTROLS ARE } { identificador-1 [identificador-2] ...             }
                { FINAL identificador-1 [identificador-2] ...        }

    PAGE [ LIMIT IS  ] entero-1 { LINE  }
         [ LIMITS ARE ]         { LINES }

    [HEADING entero-2]
    [FIRST DETAIL entero-3]
    [LAST DETAIL entero-4]
    [FOOTING entero-5]
```

ENTRADA DE DESCRIPCION DE GRUPO DE INFORME

Formato-1

```
01    nombre-de-datos-1

    LINE NUMBER IS { entero-1          }
                   { PLUS entero-2     }
                   { NEXT PAGE         }

    NEXT GROUP IS  { entero-1          }
                   { PLUS entero-2     }
                   { NEXT PAGE         }

                   { { REPORT HEADING }                                  }
                   { { RH             }                                  }
                   { { PAGE HEADING   }                                  }
                   { { PH             }                                  }
                   { { CONTROL HEADING } { identificador-n }             }
                   { { CH             } { FINAL            }             }
    TYPE IS        { { DETAIL         }                                  }
                   { { DE             }                                  }
                   { { CONTROL FOOTING } { identificador-n }             }
                   { { CF             } { FINAL            }             }
                   { { PAGE FOOTING   }                                  }
                   { { PF             }                                  }
                   { { REPORT FOOTING }                                  }
                   { { RF             }                                  }
```

Cláusula USAGE

Formato-2

nn **nombre-de-datos-1**

COLUMN NUMBER IS entero-1
GROUP INDICATE
Cláusula JUSTIFIED
Cláusula LINE — igual que en Formato-1
Cláusula PICTURE

RESET ON { **identificador-1** | **FINAL** }

SOURCE IS { **TALLY** | **identificador-2** }

SUM { **TALLY** | **identificador-3** } { **TALLY** | **identificador-4** } ... [**UPON nombre-de-datos**]

VALUE IS literal-1
Cláusula BLANK WHEN ZERO
Cláusula USAGE

FORMATOS DEL ESCRITOR DE INFORMES DE LA PROCEDURE DIVISION

Declaración GENERATE

GENERATE *identificador*

Declaración INITIATE

INITIATE **nombre-de-informe-1** [**nombre-de-informe-2**] ...

Declaración TERMINATE

TERMINATE **nombre-de-informe-1** [**nombre-de-informe-2**] ...

Sentencia USE

USE BEFORE REPORTING *identificador-1*

INDICE

Acceso
  modos de 286
  al azar 90
  secuencial 90
Actualización de un archivo
  directo 311, 318, 323
  indexado 304, 299
ALL 'literal' 65

Búsqueda en tablas
  binaria indexada 278
  binaria subindicada 257
  secuencial indexada 267
  secuencial subindicada 255

Calificación de los nombres de datos 111
Campo
  alfabético 114
  alfanúmerico 114
  de edición numérica 119
  numéricos de coma fija 115
  numéricos de coma flotante 117
Caracteres
  alfabéticos 55
  empleados en condiciones de relación 57
  empleados en edición 118
  empleados en expresiones aritméticas 57
  empleados en puntuación 56
  especiales 56
  numéricos 55
Clasificación de
  campos de datos114
  característica de, 327
Característica de
  clasificación 327
  escritor de informes 337
Cláusula
  ACCESS MODE 89
  ACTUAL KEY 309
  APPLY 294, 323
  ASSIGN 87
  BLANK WHEN ZERO 130
  BLOCK CONTAINS 101
  COLUMN 354
  CONTROL 349
  DATA RECORD 103
  GROUP INDICATE 358
  JUSTIFIED 129
  LABEL RECORDS 105
  LINE 357, 374
  NEXT GROUP 373
  NOMINAL KEY 298
  OCCURS 131
  PAGE 350
  PICTURE 112
  RECORD CONTAINS 103
  RECORD KEY 293
  RECORDING MODE 100
  REDEFINES 134
  RENAMES 142
  REPORT 347
  RERUN 92
  RESERVE 91
  RESET 360
  SAME 93
  SELECT 86
  SOURCE 355
  SUM 356, 374
  TYPE 354
  USAGE 125
  USAGE INDEX 272
  VALUE 137
Comunicación entre programas 383
Comparación de operandos
  no números 186
  numéricos 184
Compilación, pruebas 46
Condiciones compuestas 190
Condiciones de
  clase 183
  nombre de condición 183
  relación 183
  signo 183
CONFIGURATION SECTION 82
Consideraciones generales sobre escritura de
  DIVISIONES, SECCIONES, PARRAFOS 70
Constantes 60
  figurativas 63
Continuación de
  literales no numéricos 68
  palabras COBOL 68
Controles y rupturas de control 347
Creación de un archivo
  directo 307, 314, 319
  indexado 291
  secuencial 288

DATA DIVISION 97
Declaración
  ACCEPT 210
  ADD 156
  ALTER 215
  CALL 390
  CLOSE 209
  COMPUTE 165
  DISPLAY 212

DIVIDE 164
ENTRY 393
EXAMINE 175
EXIT 233
GO 213
GENERATE 362
IF 179
INITIATE 361
MOVE 169
MULTIPLY 163
NOTE 234
OPEN 201
PERFORM 217
READ 203
RELEASE 333
RETURN 336
REWRITE 299
SEARCH 273, 278
SET 270
SORT 328
START 298
STOP 200
SUBTRACT 162
TERMINATE 362
WRITE 205
Declaraciones
aritméticas 156
condicionales 154
imperativas 154
Declarativas
BEFORE REPORTING 375
ERROR PROCEDURE 317
Descripción de
archivos 99
registros 106
Dirección
relativa de pista 314
real de pista 319
Direccionamiento
directo 307
indirecto 313
Dispositivos de acceso directo 286

Edición con
inserción fija 119
inserción flotante 121
supresión de ceros 118
sustitución de ceros 118
Ejemplo de
búsqueda binaria en tablas (indexación) 268
búsqueda binaria en tablas (subindicación) 262
búsqueda secuencial en tablas (subindicación) 256
indexación 266
manejo de tablas (subindicación)248
un programa COBOL 23
un programa con IF compuestos 196
Ejercicios 51, 75, 95, 147, 235, 280
Ejecución, pruebas 46
Elementos del lenguaje COBOL 55
Enlace de programas 390
Entrada de descripción de
archivos 99
grupos de informe 352
informe 348
registros 106
ENVIRONMENT DIVISION 81
Estructura de la
DATA DIVISION 98
ENVIRONMENT DIVISION 81
IDENTIFICATION DIVISION 79
PROCEDURE DIVISION 155
Estructura de un Programa COBOL 77
Expresión aritmética 166

FD 99
FILE SECTION 99
FILLER 109
Formato
de un informe impreso 338
notación sobre él 72

HIGH–VALUE 65
Hoja de codificación 66

Identificador 110
IDENTIFICATION DIVISION 79
Indexación
directa 269
relativa 269
Tipos de 254
INPUT–OUTPUT SECTION 85

Jerarquías de IF 198
Juego de caracteres COBOL 55

LINE–COUNTER 356
Líneas
de comentarios 68
de continuación 70, 71
en blanco 71
LINKAGE SECTION 394
Literal
numérico de coma fija 61
numérico de coma flotante 62
Literales
no numéricos 63
numéricos 61
LOW–VALUE 65

Margen
A 69
B 69

Métodos para referenciar elementos de tablas 241
de acceso 286
de organización 287
Movimiento de tipo
no numérico 172
numérico 170

Nombres
creados 58
de condición 59, 139, 140
de datos 59, 109
especiales 59, 83, 206
Notación sobre formato de inscripciones 72
Números de
nivel 108
secuencia 66

Opción
ASCENDING 133, 330
AT END 203
COMPUTATIONAL 126
COMP–1 126
COMP–2 126
COMP–3 126
CORRESPONDING 161, 163, 173
DEPENDING 132, 214
DESCENDING 133, 330
DISPLAY 125
FROM 162, 205, 210
GIVING 159, 330
INDEXED 133, 269
INPUT PROCEDURE 331
INTO 165, 203
INVALID 204, 208
OUTPUT PROCEDURE 333
REMAINDER 165
REPLACING 176
ROUNDED 160
SIZE ERROR 160
USING 330
TALLYING 175
TO 158
VARYING 276

Organización
métodos de 286
directa 305
indexada 291
secuencial 288

PAGE–COUNTER 355, 356
Palabras
COBOL 57
reservadas 60
Párrafo 154
DATE–COMPILED 80
FILE–CONTROL 85
I–O–CONTROL 92
OBJECT–COMPUTER 82
SOURCE–COMPUTER 82
SPECIAL–NAMES 83
Procedimiento 154
PROCEDURE DIVISION 153
Programa
de llamada y llamado 387
fuente 44
objeto 46
Prueba de condiciones de
clase 188
nombre de condición 189
relación 183
signo 188
Punto de
entrada 393
llamada 390
regreso 393

QUOTE 65

Recuperación
al azar en un archivo indexado 296
secuencial en un archivo indexado 300
Registros especiales
LINE–COUNTER 356
PAGE–COUNTER 355, 356
Reglas de formación de los
literales no numéricos 63
literales numéricos de coma fija 61
literales numéricos de coma flotante 62
nombres creados 59
Reglas de edición 118
Reglas de puntuación 71
Relaciones entre VALUE, OCCURS y REDEFINES 141
REPORT SECTION 348

Sección 155
Sentencia 154
Sentencia USE
BEFORE REPORTING 375
ERROR PROCEDURE 317
Serie de caracteres 112
Símbolos utilizados en la cláusula PICTURE 112
SPACE 64
Subindicación 242
Subíndice 242

Tablas, descripción 132, 241
Truncamiento 159

Verbos
aritméticos 156
de bifurcación 213
de comparación 179
de detención 200
de entrada/salida 201
de movimiento 169

WORKING–STORAGE SECTION 144

ZERO 64