DIRECTORIO DE PROEFESORES DEL CURSO: INTRODUCCION AL LENGUAJE DE PROGRAMACION COBOL.

17 DE MARZO AL 7 DE ABRIL DE 1992

ING. VICTOR M. LEYVA ALATRISTE (COORDINADOR)
SERVICIO POSTAL MEXICANO
GERENTE DE DESARROLLO POSTAL
NEZAHUALCOYOTL 109 PISO 8
COL. CENTRO
C.P. 06080 MEXICO, D.F.
TEL: 709-93-10

ING. ARTURO SANDOVAL ARRIAGA
ASESOR INDEPENDIENTE
RETORNO 59 # 40
COL. AVANTE
C.P. 04460 MEXICO, D.F.
TEL: 677-61-85
(DOMICILIO PARTICULAR)

ING. LAURA SANDOVAL MONTAÑO
CENTRO DE CALCULO
FACULTAD DE INGENIERIA, UNAM
JEFE DEPTO. FORMACION Y
ACTUALIZACION ACADEMICA
CIUDAD UNIVERSITARIA
MEXICO, D.F.
TEL: 550-57-34

E V A L U A C I O N D E L P E R S O N A L

D O C E N T E

CURSO: INTRODUCCION AL LENGUAJE DE PROGRAMACION COBOL , 1992.

FECHA: 17 de marzo al 7 de abril
martes y jueves de 17 a 21 hrs.
sabados de 10 a 14 hrs.

	C O N F E R E N C I S T A	DOMINIO DEL TEMA	EFICIENCIA EN EL USO DE AYUDAS AUDIOVISUALES	MANTENIMIENTO DEL INTERES COMUNICACION CON LOS ASISTENTES AMENIDAD, FACILIDAD DE EXPRESION	PUNTUALIDAD	PROMEDIO
1.-	ING. VICTOR M. LEYVA ALATRISTE					
2.-	ING. LAURA SANDOVAL MONTAÑO					
3.-	ING. ARTURO SANDOVAL ARRIAGA					
4.-						
5.-						
	EVALUACION TOTAL					

ESCALA DE EVALUACION: 1 A 10

CURSO: INTRODUCCION AL LENGUAJE
DE PROGRAMACION COBOL, 1992.

FECHA: 17 de marzo al 7 de abril
martes y jueves de 17 a 21 hrs.
sábados de 10 a 14 hrs.

	T E M A	ORGANIZACION Y DESARROLLO DEL TEMA	GRADO DE PROFUNDIDAD LOGRADO EN EL TEMA	GRADO DE ACTUALIZACION LOGRADO EN EL TEMA	UTILIDAD PRACTICA DEL TEMA	PROMEDIO
1.-	INTRODUCCION AL PROCESO					
	ELECTRONICO DE DATOS					
2.	INTRODUCCION A LA MICROCOMPUTACION					
3.-	INTRODUCCION AL LENGUAJE COBOL					
4.-	ELEMENTOS DEL LENGUAJE COBOL					
5.	EJEMPLOS DE APLICACION					
	EVALUACION TOTAL					

ESCALA DE EVALUACION: 1 A 10

EVALUACION DEL CURSO

	C O N C E P T O	
1.-	APLICACION INMEDIATA DE LOS CONCEPTOS EXPUESTOS	
2.-	CLARIDAD CON QUE SE EXPUSIERON LOS TEMAS	
3.-	GRADO DE ACTUALIZACION LOGRADO EN EL CURSO	
4.-	CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
5.-	CONTINUIDAD EN LOS TEMAS DEL CURSO	
6.-	CALIDAD DE LAS NOTAS DEL CURSO	
7.-	GRADO DE MOTIVACION LOGRADO EN EL CURSO	
	EVALUACION TOTAL	

ESCALA DE EVALUACION: 1 A 10

1.- ¿Qué le pareció el ambiente en la División de Educación Continua?

MUY AGRADABLE ( ) AGRADABLE ( ) DESAGRADABLE ( )

2.- Medio de comunicación por el que se enteró del curso:

PERIODICO EXCELSIOR ANUNCIO TITULADO DIVISION DE EDUCACION CONTINUA ( )

PERIODICO NOVEDADES ANUNCIO TITULADO DIVISION DE EDUCACION CONTINUA ( )

FOLLETO DEL CURSO ( )

CARTEL MENSUAL ( )

RADIO UNIVERSIDAD ( )

COMUNICACION CARTA, TELEFONO, VERBAL, ETC. ( )

REVISTAS TECNICAS ( )

FOLLETO ANUAL ( )

CARTELERA UNAM "LOS UNIVERSITARIOS HOY" ( )

GACETA UNAM ( )

3.- Medio de transporte utilizado para venir al Palacio de Minería:

AUTOMOVIL PARTICULAR ( )

METRO ( )

OTRO MEDIO ( )

4.- ¿Qué cambios haría en el programa para tratar de perfeccionar el curso?

_____

_____

_____

5.- ¿Recomendaría el curso a otras personas? ( ) SI ( ) NO

5.a. ¿Qué periódico lee con mayor frecuencia?

_____

_____

6.- ¿Qué cursos le gustaría que ofreciera la División de Educación Continua?

_____

_____

_____

7.- La coordinación académica fué:

EXCELENTE ☐ BUENA ☐ REGULAR ☐ MALA ☐

8.- Si está interesado en tomar algún curso INTENSIVO ¿Cuál es el horario más conveniente para usted?

LUNES A VIERNES DE 9 a 13 H. Y DE 14 A 18 H. (CON COMIDAD) ☐

LUNES A VIERNES DE 17 a 21 H. ☐

LUNES A MIERCOLES Y VIERNES DE 18 A 21 H. ☐

MARTES Y JUEVES DE 18 A 21 H. ☐

VIERNES DE 17 A 21 H. SABADOS DE 9 A 14 H. ☐

VIERNES DE 17 A 21 H. SABADOS DE 9 A 13 H. DE 14 A 18 H. ☐

OTRO ☐

9.- ¿Qué servicios adicionales desearía que tuviese la División de Educación Continua, para los asistentes?

_____

_____

_____

10.- Otras sugerencias:

_____

_____

_____

DIRECTORIO DE ASISTENTES AL CURSO: INTRODUCCION AL LENGUAJE DE PROGRAMACION COBOL

DEL 17 DE MARZO AL 7 DE ABRIL DE 1992

1.-OSCAR BAYARDI ORTIZ PETRICIOLI
SUPERVISOR DE SISTEMAS
ANTERA, S.A. DE C.V.

PITAGORAS NO. 577
NARVARTE BENITO JUAREZ
03020 MEXICO, D.F.
TEL. 543-4255

LUZ SAVIÑON 1363-A INT-1
NARVARTE BENITO JUAREZ
03020 MEXICO, D.F.
TEL. 543-4984

2.-ROGELIO DOMINGUEZ PACHECO
AUTORIZADOR DE OPERACIONES
BANCO DE MEXICO

AV. 5 DE MAYO NO. 2
CENTRO CUAUHTEMOC
06000 MEXICO, D.F.
TEL. 709-0044 EXT-2186

EDIF-67 ENT-L DEPTO-104
UNIDAD LINDAVISTA VALLEJO
MEXICO, D.F.
TEL. 5677-0194

3.-ALFONSO GALAN GARCIA
JEFE DE OFICINA DE INGENIERIA CIVIL
S. C. T.

PROVIDENCIA NO. 807, 3o. P.
DEL VALLE BENITO JUAREZ
03100 MEXICO, D.F.
TEL. 523-4651

GUSTAVO E. CAMPA NO. 58
VALLEJO
MEXICO, D.F.
TEL. 537-2743

4.-GABRIELA GARDUÑO BERMUDEZ
S. C. T.

5.-YARETSET GOMEZ ACOSTA
PERSONAL CALIF EN P.
BOTONES SUPERIORES

CALLE NORTE 45 NO. 633
INDUSTRIAL VALLEJO AZCAPOTZALCO
02300 MEXICO, D.F.

AV. MEXICO NO. 72
5A. SEC. JARDINES DE MORELOS
MEXICO, D.F.

6.-MERCEDES RAMIREZ MARQUEZ

S. C. T.

7.-JOSE REYES CONTRERAS
JEFE DE SISTEMAS
INDUSTRIAL PAPELERA NACIONAL, S.A.C.V.

CALZ. ROJO GOMEZ NO. 1201
IZTAPALAPA
MEXICO, D.F.
TEL. 6860022

ALFONSO CAPETILLO NO. 129
AMPL. SN PEDRO XALPA
MEXICO, D.F.
TEL. 352-2895

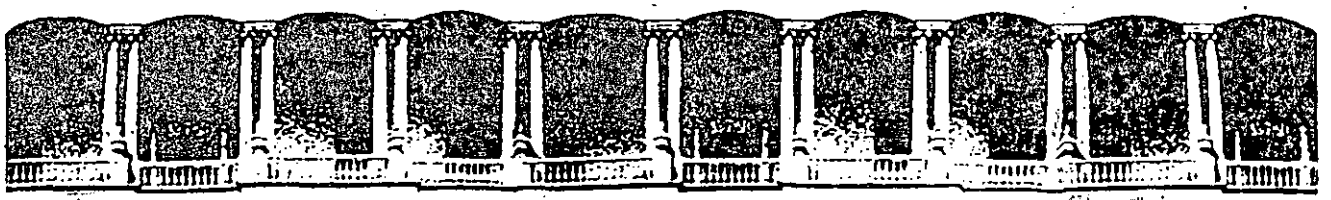
8.-RAQUEL ROCHA RODRIGUEZ

S. C. T.

9.-MAURICIO APOLONIO RODRIGUEZ GARCIA
JEFE DE DEPTO.
S. C. T.

CUAUHTEMOC NO. 614
SAN ANTONIO
MEXICO, D.F.
TEL. 519-2636

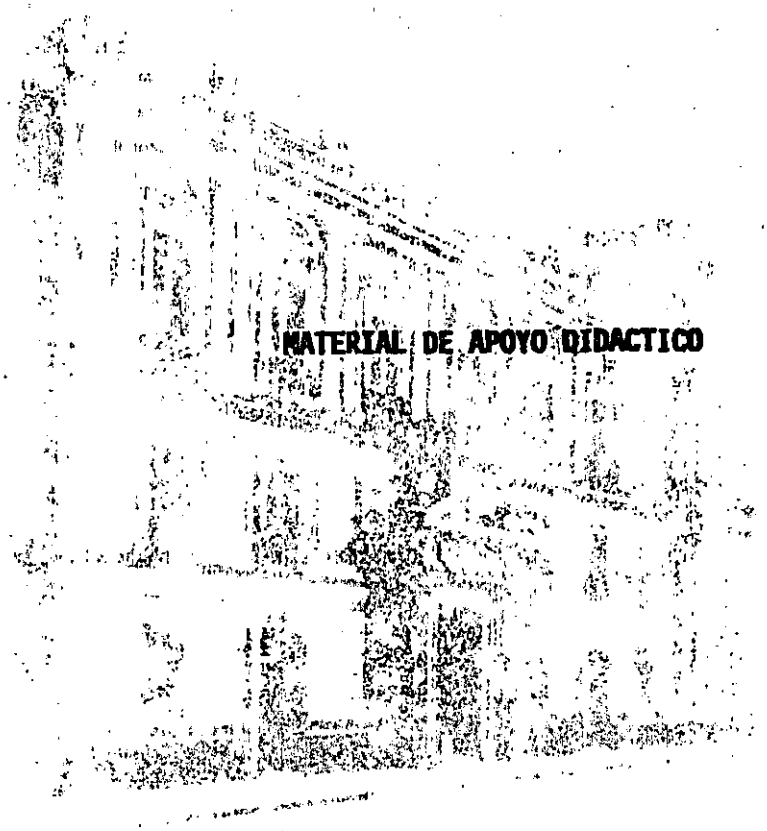
CIRCUITO RIO DEL ORO NO. 72 N
PASEOS DE CHURUBUSCO IZTAPALAPA
09030 MEXICO, D.F.
TEL. 654-4784



FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

INTRODUCCION AL LENGUAJE DE PROGRAMACION

C O B O L



MARZO-ABRIL, 1992.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. tel.: 521-40-20 Apdo. Postal M-2285

ADVERTENCIA

Estas notas de **COBOL** fueron elaboradas para cursos de **COBOL** impartidos en el Centro de Cálculo de la Facultad de Ingeniería y en ningún momento sustituyen libros sobre el lenguaje COBOL ya sea estandard o de algún equipo en especial, es por tanto necesario consultar libros de lenguaje **COBOL** y manuales de consulta, referencia o programación del equipo a emplear; estos estan discritos en la bibliografía. Aunque **COBOL** es un lenguaje estandard existen variaciones al momento de implementarse en algún sistema de cómputo, se recomienda consultar los manuales del equipo para identificar estas cambios.

CHAPTER 1

INTRODUCCION A COBOL

COBOL es un lenguaje de programación de computadoras para realizar procesos administrativos. Este lenguaje está implementado en la mayoría de los centros de cómputo, es por tanto, el lenguaje mayor difundido en el ambiente de cómputo.

1.1 HISTORIA DE COBOL

El lenguaje **COBOL** (COmmon Bussiness Oriented Lenguage) fué diseñado para usarse en procesos de información mercantiles.

Históricamente el lenguaje fué concebido por el Pentágono en mayo de 1959. En él intervinieron representantes del gobierno de los E.E.U.U., usuarios, fabricantes de equipo de cómputo decidiendo la creación de un lenguaje de alto nivel que pudiera procesar información mercantil.

La primera versión de **COBOL** apareció en diciembre de 1959. En 1961 se dió la versión **COBOL-61** el cual fijó las estructuras de las posteriores versiones. La siguiente versión apareció en 1965, pero no fué sino hasta agosto de 1968 cuando el lenguaje se estandarizó por el American National Standard Institute (ANSI). Esta versión estándar fué revisada en 1974 implementándosele mejoras que cada fabricante e instalación ha creido pertinentes.

1.2 MODULOS DE COBOL

Los modulos de **COBOL** son aquellos que componen al lenguaje **COBOL** y estos son mostrados en el apendice. La instalación que tenga el compilador COBOL cuenta siempre con el núcleo del lenguaje y unó o todos los demás modulos. Existen sistemas que adicionan una serie de rutinas o procesos para realizar diferentes operaciones, como son la interrelación con bases de datos, manejo de pantalla, etc., siendo estas operaciones no naturales de **COBOL**, por lo que, es necesario saber cuales son las instrucciones estandard del lenguaje y cuales son las que aporta el sistema. En estas notas se explicara solo las instrucciones del **COBOL** estandard.

1.3 CARACTERISTICAS

A continuación se presentan las características de **COBOL**:

1. Entendible. Ya que el lenguaje emplea oraciones, en inglés, que indican las operaciones a realizar.

2. Autodocumentado. Por la característica de codificación se elabora un documento que por si solo define lo que se está realizando.

3. Transportable. Esto significa que cualquier instalación de cómputo que cuente con el sistema Cobol podrá correr cualquier programa escrito en **COBOL**.

4. Orientado a los negocios. Por la gran cantidad de información manejada con referencia a la administración; este lenguaje fué orientado a emplearse en las empresas.

5. Lenguaje que requiere de un compilador que traduce del lenguaje **COBOL** a lenguaje de máquina de la computadora.

1.4 ELEMENTOS DE COBOL

Los elementos de **COBOL** son aquellos que define las características para programar en **COBOL**.

1.4.1 DESARROLLO DE SISTEMAS

El desarrollo de sistemas es importante al elaborar cualquier programa de **COBOL**.

1.4.2 LENGUAJES DE PROGRAMACION

Un lenguaje de programación se describe em terminos de sintaxis y semantica. La sintaxis se refiere a los termino que forman la estructura del lenguaje; la emantica define la forma correcta de definir las estructuras.

Aprender el lenguaje de programación **COBOL** es aprender las reglas de formación sintactica y semantica de las estructuras, de forma que el compilador **COBOL** traduzca a lenguaje de máquina.

1.4.3 CARACTERES

Los caracteres son los elementos primarios de un programa y son todos aquellos representados en la tabla de caracteres ASCII.

En el lenguaje **COBOL** los caracteres que se emplean son:

1. Caracteres numéricos : 0 a 9.

2. Caracteres alfabéticos : A a la Z y B (espacio en blanco).

3. Caracteres especiales : ().,"+-*/=$,;<>.

1.4.4 PALABRAS

Las palabras son cadenas de caracteres que definen un elemento del lenguaje **COBOL**.

Las palabras se forman con las siguientes reglas:

1. Longitud menor a 30 y mayor que cero.

2. Sólo pueden contener letras, números, guiones o subguiones.

3. Deben contener por lo menos una letra.

4. El guión o subguión no debe comenzar ni finalizar la palabra.

Las palabras por su definición se clasifican en:

1. Palabras reservadas. palabras que definen en forma general al lenguaje. (MOVE, ENVIRONMENT, ADD, DIVISION, etc.). Estas palabras no se pueden emplear para otro fin diferente para el cual estan ya definidas. La lista de las palabras reservadas se muestran en el apéndice.

2. Palabras del sistema. Palabras definidas por la máquina donde reside. (SYS$OUTPUT, SYS$INPUT, MTH$RANDOM, etc).

3. Palabras del usuario. Que son todas aquellas que emplea el usuario en la solución algoritmica de su problema (Hoja, número-de-cuenta, etc).

Las palabras por su función se dividen en:

1. Nombre de dato.

2. Nombre de grupo.

3. Nombre de registro.

4. Nombre de archivo.

5. Nombre de condición.

6. Claúsula.

7. Verbo.

8. Instrucción (Oración).

9. Nombre de procedimiento ó párrafo.

10. Nombre de sección.

11. Nombre de división.

La definición de nombres de dato, grupo, registro, archivo y condición se indicaran en los siguientes capitulos.

1.4.5 CLAUSULAS

Una cláusula es la parte de una instrucción donde se definen características de la propia instrucción o de los operandos que la conforman.

1.4.6 VERBOS

Las oraciones están compuestas con verbos, los cuales ordenan una acción. El lenguaje **COBOL** proporciona una serie de verbos cuya sintaxis y función se encuentra ya definida.

Algunos verbos operan con campos de información y otros definen el orden en que se ejecutarán las instrucciones.

Con los verbos se puede hacer:

1. Movimientos de información.
2. Operaciones aritméticas.
3. Operaciones de entrada y salida.
4. Comparaciones para tomas decisiones.
5. Repeticion de instrucciones.
6. Control de flujo.

1.4.7 ORACIONES

Las oraciones representan las instrucciones del programa, y siempre comienzan con un verbo. Las oraciones pueden terminar con un punto, pero en general se acostumbra terminarlas con punto si es que la sintaxis no marque lo contrario.

1.4.8 PARRAFOS

Las oraciones se agrupan en párrafos. Por lo que, el parrafo se compone de un nombre de párrafo, una serie de oraciones, donde la última termina con un punto.

1.4.9 SECCIONES

Los parrafos forman secciones.

1.4.10 DIVISIONES

Las secciones forman divisiones.

1.5 ESTRUCTURA DE UN PROGRAMA COBOL

Un programa **COBOL** esta estructurado en cuatro divisiones donde se define :

1. La identificación del programa.

2. La asociación al programa de los dispositivos externos por los cuales se obtendrán los datos de entrada y se emitirán los resultados.

3. La descripción de la información que se procesará.

4. Los algoritmos de procesamiento de la información.

En el lenguaje **COBOL**, a diferencia de otros lenguajes, es necesario entender la función de cada una de las divisiones para poder desarrollar algún programa.

1.6 CONVENCION AL DEFINIR LAS ESTRUCTURAS EN COBOL

La interpretación de los formatos que definen las instrucciones **COBOL** son :

1. Las palabras en **MAYUSCULAS RESALTADAS** ó subrayadas son palabras definidas por el lenguaje deben usarse al definir una instruccion.

2. Las palabras en MAYUSCULAS NO RESALTADAS son palabras definidas por el lenguaje pueden usarse si se desea al definir una instrucción y son para denotar mayor claridad.

3. Las palabras en minúsculas son las que definen una palabra **COBOL** del usuario y su valor depende de su propia definición.

4. Lo encerrado en {} implica que hay que escoger alguna de las palabras o frases definidas.

5. Lo encerrado en [ ] es opcional.

6. Los puntos ... indican que el concepto puede repetirse tantas veces sea necesario.

7. Las comas (,) y los puntos y coma (;) pueden omitirse en las declaraciones; no asi el punto (.).

1.7 REGLAS DE PUNTUACION

Toda palabra en **COBOL** se limita a la derecha por lo menos un espácio en blanco.

Dos o más nombres de dato escritos en una lista pueden separarse por un espacio en blanco, o por una coma seguida de un espacio en blanco.

1.8 CODIFICACION

Existen dos formas de codificar (escribir) programas **COBOL**:

1. Formato ANSI. Esta forma se desarrolló y usó desde el principio y se basa en la tarjeta perforada de 80 columnas. En ella se definen las siguientes columnas:

USO	COLUMNAS
. Hoja:	1-3
. Renglón:	4-6
. Carácter especial:	7
. Area A:	8-11
. Area B:	12-72
. Identificación:	73-80

La hoja, renglon e identificación son opcionales al codificar un programa **Cobol**.

2. Formato Terminal. Esta forma se introdujo al codificarse el programa directamente en la terminal.

USO	COLUMNAS
. Carácter especial:	1
. Area A:	1-4
. Area B:	5-256

El carácter especial siguiente:

1. * Comentario.

2. - Continuación de instrucción.

3. / Al imprimir un listado de un programa compilado, salta la impresion a la siguiente hoja.

Las palabras dentro de la codificación se dividen en:

1. Palabras que se colocan a partir del área A.

    1. Nombre de división.

    2. Nombre de sección.

    3. Nombre de párrafo.

    4. Definición de archivo.

    5. Definición de registro.

6. Definición de dato.

2. Palabras que se colocan a partir del área B.

1. Declaraciones.

2. Definición de grupo.

3. Definición de dato.

4. Instrucciones (Oraciones).

1.9 DIVISIONES DE LOS PROGRAMAS COBOL

Todos los programas en **COBOL** se estructuran en cuatro divisiones, las cuales van en el siguiente orden:

1. IDENTIFICATION DIVISION.

2. ENVIRONMENT DIVISION.

3. DATA DIVISION.

4. PROCEDURE DIVISION.

1.9.1 IDENTIFICATION DIVISION

Consta de unas cuantas líneas, donde se indica: el nombre del programa, el autor, la instalacción donde se ejecuta el programa, etc.

1.9.2 ENVIRONMENT DIVISION

En esta división se define :

1. Especificar las computadoras que se utilizan, tanto para la compilación del programa, como para su ejecución.

2. Asignar nombre a los dispositivos de entrada salida que se emplearán en el programa.

3. Definir algunas características de los archivos.

4. Dar nombre a ciertas funciones de algunos dispositivos.

1.9.3 DATA DIVISION

Se emplea para definir las características de la información que se manejará en el programa. En esta división se realizara:

1. Descripción de las características de los archivos, (etiquetas, atributos, nombre de los registros).

2. Descripción de los registros de entrada/salida, (campos en que se dividen los registros, características de los campos).

3. Descripción de los registros o campos auxiliares, (encabezados, contadores, registros intermedios).

1.9.4 PROCEDURE DIVISION

En esta división se define la forma en que se procesará la información que manejará el programa. La división esta constituida por parrafos. La ejecución del programa comienza desde el primer parrafo de esta división.

CHAPTER 2

DIVISIONES DE COBOL

Un programa escrito en **Cobol** se compone de cuatro divisiones.

2.1 IDENTIFICATION DIVISION

Consta de unas cuantas lineas, que identifican el programa. Nombre del programa, autor, instalacción utilizada, fecha de creación w compilación.

2.1.1 FORMATO IDENTIFICATION DIVISON

El formato de esta división se compone de los siguientes parrafos los cuales son en este orden:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      nombre-de-programa.
[ AUTHOR.          [ comentario . ] ]
[ INSTALLATION.    [ comentario . ] ]
[ DATE-WRITTEN.    [ comentario . ] ]
[ DATE-COMPILED.   [ comentario . ] ]
[ SECURITY.        [ comentario . ] ]
```

2.1.2 PROGRAM-ID

Parrafo que define el nombre lósico del programa; este nombre es referenciado al hacer uso de este proceso el sistema operativo. El nombre-de-programa este definido bajo las reglas de construccion de palabras **COBOL**.

2.1.3 AUTHOR

Parrafo que indica el programador o institución que realiza el programa. El comentario es desarrollado bajo las normas de documentación de la institución.

2.1.4 INSTALLATION

Parrafo que muestra el sistema de cómputo donde se implementa el programa.

2.1.5 DATE-WRITTEN

Parrafo que indica la fecha en que fué escrito el programa.

2.1.6 DATE-COMPILED

Parrafo que indica la fecha en que fué coompilado el programa, en algunos sistemas este comentario es puesto por el propio sistema y mostrado al imprimir un listado de compilación.

2.1.7 SECURITY

Parrafo que relaciona las caracteristicas del proceso, sus derechos sobre la elaboración y notas complementarias a la documentación.

2.2 ENVIRONMENT DIVISION

En la segunda división de un programa en **COBOL**. Se emplea para definir los dispositivos periféricos que se van a utilizar, asignandoles un nombre de archivo para su manejo. También para definir ciertas caracteristicas de los archivos, asi como para darles nombre a ciertas funciones de algunos dispositivos.

2.2.1 FORMATO ENVIRONMENT DIVISION

Esta división se compone de los siguientes secciones y parrafos:

```
ENVIRONMENT DIVISION.
[CONFIGURATION SECTION.
[SOURCE-COMPUTER.]    [Comentario.]
[OBJECT-COMPUTER.]    [Comentario.]
[SPECIAL-NAMES.]      [Nombres-especiales.]
[INPUT-OUTPUT SECTION.
[FILE-CONTROL.
                      [FILE-CONTROL-DESCRIPTION...]]

Nombres-especiales.

   Nombre IS Definición-de-funcion.

FILE-CONTROL-DESCRIPTION.

   SELECT Nombre-archivo-logico ASSIGN TO dispositivo-periferico.
```

2.2.2 CONFIGURATION SECTION

En esta sección se define la computadora donde se codificó el programa fuente y donde se encuentra el código ejecutable; así como de definir funciones para los dispositivos de entrada salida.

2.2.2.1 SOURCE-COMPUTER

En este párrafo se indica el nombre de la computadora o sistema de cómputo donde se codificó el fuente del programa.

2.2.2.2 OBJECT-COMPUTER

Párrafo que indica el nombre de la computadora o sistema de cómputo donde esta el código ejecutable del programa.

2.2.2.3 SPECIAL-NAMES

En este párrafo se definen los nombres por los cuales se indican funciones del sistema para los dispositivos de entrada salida.

2.2.3 INPUT-OUTPUT SECTION

En esta sección se declaran todos los dispositivos conectados al proceso por medio de archivos.

Los párrafos de esta sección se describirán al momento de definir conceptos relacionados con estos párrafos.

2.3 DATA DIVISION

La tercera división de un programa **COBOL** es la DATA DIVISION. Permite la descripción de las características de los archivos de entrada y/o salida, tales son:

1. Etiquetas.

2. Tamaño de bloque.

3. Tamaño de registro.

4. Nombre, tamaño y tipo de los campos en que está dividido un registro.

5. Descripción de registros o campos auxiliares como son contadores, encabezados, lineas de detalle etc.

2.3.1 FORMATO DATA DIVISION

La DATA DIVISION se compone de los siguientes secciones y parrafos:

```
DATA DIVISION.
[FILE SECTION.
     [FILE-DESCRIPTION
     [RECORD-DESCRIPTION] ...] ...]
[WORKING-STORAGE SECTION.
     [ELEMENTARY-DATA-DESCRIPTION] ...
     [RECORD-DESCRIPTION] ... ]

FILE-DESCRIPTION.

FD Nombre-de-archivo
  LABEL RECORD ARE { OMITTED  }
                   { STANDARD }
  DATA RECORD IS Nombre-de-registro.

RECORD-DESCRIPTION.

01 Nombre-de-registro.
  DEFINICION DE GRUPOS O DATO.

DEFINICION DE GRUPO.

[NUMERO DE NIVEL <> 01] NOMBRE DE GRUPO.
  DEFINICION DE GRUPOS O DATO.

ELEMENTARY-DATA-DESCRIPTION.

77 DEFINICION DE DATO.

DEFINICION DE DATO.
   { NOMBRE DE DATO  } { PICTURE } { A } CLAUSULAS.
   {    FILLER       } {     PIC } { X }
                                   { 9 }
```

Todos los conceptos de esta división se describiran al momento de definir conceptos relacionados con esta división.

2.4 PROCEDURE DIVISION.

En esta división se define la forma en que se procesará la información que manejará el programa. La división esta constituida por párrafos. La ejecución del programa comienza desde el primer párrafo de esta division.

2.4.1 FORMATO PROCEDURE DIVISION

La división de procedimientos se compone de los siguientes secciones y parrafos:

```
PROCEDURE DIVISION.
[Nombre de sección SECTION.]
{Nombre de párrafo.
    [instruccion] ... } }...
```

2.4.2 ALGORITMO DE PROCESO

El algoritmo de proceso de la información la componen una serie de instrucciones denominadas oraciones, que se organizan en forma lógica para obtener los resultados deseados.

2.4.3 DEFINICION DE PARRAFO

La procedure division contendrá por lo menos un párrafo. Este párrafo es nombrado con un nombre de párrafo cuya construcción sigue las normas de la definición de nombres **COBOL**, es recomendable definir nombres de acuerdo a la función que realiza.

CHAPTER 3

DEFINICION Y MANEJO DE INFORMACION

La definición y el manejo de la información dentro del proceso COBOL es una de las partes más importantes ya que se involucran el contenido de la informacion en la solución algoritmica del problema u objetivo.

3.1 DEFINICION DE INFORMACION

La definición de información tiene como objetivo de describir todos y cada uno de los datos a emplear, los cuales describen caracteristicas de la información.

3.1.1 DESCRIPCION DE LA INFORMACION

Para definir la información en un programa **COBOL**, tanto de entrada, como de salida e intermedia se emplean los conceptos de campo de dato, registro y archivo.

3.1.2 CAMPOS DE DATOS

Los campos de datos definen áreas de almacenamiento de información en el programa. A cada campo de dato habrá que definirle el número de caracteres que puede contener (longitud de campo) y el tipo de información que podrá almacenar (sea alfabético, numérico ó alfanumérico). Por ejemplo, el campo para almacenar el número de cuenta de un alumno de la UNAM, tendrá una longitud de 8 caracteres numéricos.

3.1.3 NOMBRE DE DATO

Es el nombre asociado a un campo de información, para hacerle referencia dentro del programa **COBOL**. Un nombre de dato es un nombre simbolico que direcciona localidades de memoria donde se guardan valores que varian a lo largo del proceso. Estos nombres se asocia con el tipo de dato a guardar y su longitud.

3.1.4 REGISTRO

Un registro es un conjunto de campos de información que se agrupan y se referencian en conjunto con un nombre (nombre de registro).

3.1.5 ARCHIVO

Un archivo es un conjunto de registros que tienen el mismo tipo de información. A los archivos se les asigna un nombre para hacerles referencia dentro del programa (Nombre de archivo).

Los archivos siempre se relacionan con dispositivos externos al programa, conocidos como equipos periféricos.

A través de los archivos se hace la entrada y salida de la información de cualquier programa **COBOL**.

3.2 REPRESENTACION DE DATOS

Los datos se representan todos y cada uno por localidades de memória y estos son declarados en la DATA DIVISION.

3.2.1 INTRODUCCION

La DATA DIVISION, es la parte donde se declaran todas y cada una de las variables y constantes que se usarán en el proceso algoritmico de solución. Ya sean provenientes del exterior por medio de archivos, entrada-salida estándar, o locales al proceso.

Se define como dato al área donde se guarda información que puede ser o no modificada (constantes y variables). La agrupación de datos se llama grupo y al conjunto de grupos define un registro.

La descripcion-de-dato en **COBOL** esta dada por :

```
{Nombre-de-dato} {PICTURE} [IS] definición-de-dato
{  FILLER      } {  PIC  }
```

3.2.2 CONSTRUCCION DE IDENTIFICADORES

El identificador ó nombre-de-dato está regido por la definición de palabras en **COBOL**.

Esta definición de nombre-de-dato debe de ir acorde al concepto que representa (líneas, nombre-archivo-maestro, balance, pasos, etc.). Si dentro de la definición de datos éste no se quiere nombrar se utiliza la palabra FILLER.

3.2.3 DEFINICION DE DATO

La clausula PICTURE sirve para definir el dato en tipo (numérico [9], alfabético [A], ó alfanumérico [X]) y el tamaño (de caracteres). Si es un campo numerico, sirve para indicar si tiene signo(+ ó -) y si tiene decimal. En la WORKING-STORAGE SECTION permite editar la información para su impresión.

La definición de registro esta dado por :

tipo-de-dato (longitud) [calificadores]

El tipo-de-dato esta dado por la naturaleza del dato y son:

1. Para datos numéricos 9.

2. Para datos alfabéticos A.

3. Para datos alfanuméricos X.

La longitud esta dada por la cantidad de caracteres representados por el número asociado a la longitud; o por la cantidad de 9's, A's o X's definidos. Un dato numérico en forma estandard contiene hasta 18 dígitos, y un dato alfabético o alfanumérico contiene hasta 255 caracteres.

3.2.4 NIVEL 77

El nivel 77 nos indica que el dato representa una variable que tomará valores durante el transcurso del proceso. Estos datos se definen con el nivel 77, el nombre del dato, (Nunca se usa el nombre FILLER), tipo y longitud (tamaño).

```
A    B
77   DESCRIPCION-DE-DATO.
```

La característica de estos datos es que son fijos y no pueden subdividirse.

Ejemplos.

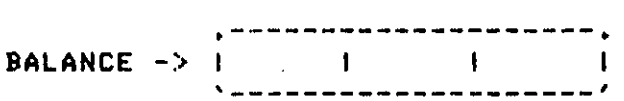
```
77  CONT-HOJA PICTURE 9(02).
77  LINEA-AUX PICTURE X(132).
77  MULTIPLICANDO-AUX PICTURE 9(03).
```

Al definir datos se indica la cantidad y tipo de caracteres asociados así si se define :

```
77 BALANCE PICTURE IS 9(03)
```

se requieren localidades de memoria referenciadas por el nombre simbolico BALANCE cuyo contenido son caracteres numéricos en un total de tres. De esta forma se define el rango de valores que podra contener la área de memoria es de 0 a 999.

Area de memoria de longitud 3 de tipo numérico.



3.3 INSTRUCCIONES

Las instrucciones son operaciones que involucran un verbo y operadores para realizar acciones para la solución algoritmica del problema.

3.3.1 INSTRUCCION STOP RUN

La instrucción STOP RUN es la instrucción que pone fin al proceso. Esta instrucción va en el lugar donde el algoritmo marca el final del proceso.

```
PROCEDURE DIVISION.
INICIO.
    ...
    STOP RUN.
```

3.3.2 INSTRUCCION ACCEPT

La instrucción ACCEPT toma el valor del dato de la entrada (teclado), este movimiento de información se ajusta a las caracteristicas del equipo, en forma general se hace por movimientos alfanuméricos, los cuales indica que el dato de entrada siempre es alfanumérico, (independiente del tipo de dato). Muchos equipos realizan una conversion de tipos ajustandose la información de entrada al tipo de dato leido. Es por esto que un blanco no es cero al momento de leerse. En algunas ocaciones estos movimientos no son marcados como error por el sistema, pero al definirse una operación aritmética con datos donde se encuentre blancos u otros caracteres no numéricos es donde indica el error.

```
PROCEDURE DIVISION.
INICIO.
    ...
    ACCEPT OPCION.
    ...
    ACCEPT NO-CTA.
    STOP RUN.
```

3.3.3 INSTRUCCION DISPLAY

La instrucción DISPLAY despliega el contenido de la variable a la salida del sistema, en forma general la salida se indica en pantalla.

Algunos sistemas de cómputo se usa la instrucción DISPLAY como modo de comunicación con el operador.

En implementaciones de **COBOL** en sistemas pequeños y medianos las instrucción DISPLAY y ACCEPT son usuadas con manejo de pantalla en procesos de captura.

```
PROCEDURE DIVISION.
INICIO.
    ...
    DISPLAY 'DAME LA OPCION:'.
    ACCEPT OPCION.
    ...
    DISPLAY 'NUMERO DE CUENTA:'.
    ACCEPT NO-CTA.
    ...
    DISPLAY 'EL REGISTRO CONTIENE LA INFORMACION:' REG-NOMINA.
    STOP RUN.
```

CHAPTER 4

DEFINICION DE REGISTROS Y MOVIMIENTO DE INFORMACION

Los registros son la estructura de información mayor empleada dentro del manejo de información. El registro siempre esta asociado a la agrupación de datos, y la entrada y salida de información de un archivo.

4.1 DEFINICION DE REGISTRO

El registro es la unidad basica para el proceso de informacion. Un registro esta definido por un número de nivel 01 y diferentes grupos de datos.

4.1.1 NIVELES 01..49

Los niveles 01 al 49 sirven para definir registros, datos agrupados y datos en una estructura que se denomina jerarquica, en donde se indica la asociación de datos bajo un nombre ya sea de grupo o registro. Los registros se definen en la DATA DIVISION. Los registros que son de entrada al sistema se definen en la FILE SECTION, y los propios del programa en la WORKING-STORAGE SECTION.

4.1.2 REPRESENTACION JERARQUICA DE LOS DATOS (REGISTROS)

```
01  NOMBRE-DE-REGISTRO.
 < número de nivel > NOMBRE-DE-GRUPO.
     < número de nivel > DESCRIPCION-DE-DATO.
```

La definición de registros se hace con la finalidad de tener un conjunto de campos relacionados para ser manejados como una entidad propia. El registro se define con un nivel 01 en el área A y un nombre de registro. Este nombre de registro está asociado a la descripción de archivos si es que se usa en un archivo; en un registro de encabezado, subencabezado o detalle.

La asignación de números de niveles se hace dependiendo del usuario. Es conveniente establecer una regla y es la de usar sólo números impares para definir el registro, grupo o dato. Por regla de **COBOL** el registro está definido por un nivel 01. Los grupos o datos por niveles 03, 05, etc. según sea necesario (el máximo nivel para la descripción de grupos o datos es 49). La asignación de números impares tiene por objeto permtir, dado el caso, la agrupación de datos.

La definición del tipo y longitud de dato sólo es puesta en la definición de dato, no así en la definición de registro o grupo subdividido, a menos que la definición de dato sea la propia definición de registro.

```
DATA DIVISION.
...
01  REGISTRO-MAESTRO.
    03 NOMBRE PIC X(30).
    03 DIRECCION.
       05 CALLE PIC X(10).
       05 NUMERO PIC X(04).
       05 COLONIA PIC IS X(20).
       05 CIUDAD PIC IS X(20).
       05 CODIGO-POSTAL PIC IS 9(06).
    03 SEXO PIC X(01).
    03 ESTADO-CIVIL PIC X(01).
```

Los datos propios del proceso, esto es, aquellos que no provienen del exterior por medio de archivos, deben ser declarados en la WORKING-STORAGE SECTION de la DATA DIVISION.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77  HOJAS PIC 9(02).
...
01  FIN-DE-ARCHIVO PIC X(02).
```

Cuando alguna área de memoria no se desea nombrarla, ya sea por no ser relevante su definición ó por constituirse por una constante es marcada con la palabra FILLER.

Ejemplo.

```
01  Registro-Entrada.
    03 Nombre-Empleado PICTURE X(32).
    03 FILLER          PICTURE X(28).
    03 Sueldo          PICTURE 9(06).
    03 FILLER          PICTURE X(12).
```

4.1.3 DEFINICION DE TIPOS (V, S, P)

Dentro de los tipos de datos numéricos se pueden emplear los siguientes caracteres:

1. S para definir que el número es signado.

2. V para definir que el dato es no entero o real (punto virtual).

3. P para definir que el valor es escalado a una potencia de 10.

Estos caracteres no son contabilizados dentro del tamaño del campo a menos que se especifique lo contrario. La definición de numeros signados es importante cuando se requiere de datos que definan cantidades numericas con signo. El punto virtual define cantidades numericas con parte entera y parte decimal. Los datos con valor escalado a potencias de 10 es empleado al utilizar valores en miles o millones y donde solo se representan las cifras significativas pero internamente e maneja a estas escalas.

4.1.4 CLAUSULA VALUE

Una de las clausulas más importantes es la clausula VALUE donde a la variable o constante se le asigna un valor en el tiempo de compilación. Estos valores iniciales estan dados por literales numéricas, alfanuméricas o por constantes figurativas. Estas inicializaciones solo son validas en la Working-Storage Section.

```
VALUE is {literal-numérica  }
         {literal-nonumérica}
         {constante-figurativa}
```

4.1.4.1 LITERALES

La literal es un elemento del lenguaje **COBOL**, que representa un valor constante (alfabético, alfanumérico o numérico) dentro de un programa. Existen dos tipos de literales, que son: las literales numéricas y las literales no numéricas.

4.1.4.2 LITERALES NUMERICAS

La literal numérica se divide en entera y no entera o real. La literal entera es aquella que está compuesta sólo de dígitos y los signos + ó -; y la literal numérica no entera por una parte entera, punto y una parte decimal; el punto no puede iniciar ni terminar la literal. El número máximo de dígitos de una literal numérica entera es de 23.

4.1.4.3 LITERALES NO NUMERICAS

La literal no numérica es un elemento formado por cualquiera de los caracteres que maneja el lenguaje **COBOL** incluyendo el espacio en blanco. Estas literales deben encerrarse entre comillas (') y el valor de la literal no numerica los constituyen los caracteres dentro de las comillas; Cualquier par de comillas ('') representa una simple comilla, lo que permite incluirse en las literales no numericas. El numero de caracteres no debe exceder de 256 caracteres.

Ejemplos.

```
DATA DIVISION.
     ...
     03 BALANCE PIC 9(02)P(03)V9(02) VALUE 0.
     06 SUMA-ACUMULADA PIC S9(10)V9(02) VALUE 0.
     05 FILLER PIC X(07) VALUE 'HOJA : '.
     06 CONTADOR-HOJAS PIC 9(03) VALUE 0.
     06 ASTERISCO PIC X(01) VALUE ''.
     04 FILLER PIC X(14) VALUE 'NOMINITA, S.A.'
```

4.1.4.4 CONSTANTES FIGURATIVAS

La constante figurativa es una palabra **COBOL** predefinida. Estas constantes se pueden emplear en cualquier parte donde se pueda emplear una literal. Son constantes figurativas:

1. Zero, zeros, zeroes. (Representan el valor de cero).

2. Space, spaces. (Representan al espacio).

3. Quote, quotes. (Representan a la comilla).

4. High-value, high-values. (define el mayor valor).

5. Low-value, low-values. (define el menor valor).

6. All <literal no numerica>. (Representa la repetición de la literal no numérica).

Ejemplos :

```
DATA DIVISION.
...
     01 CONTADOR-LINEAS PIC 9(02) VALUE ZEROES.
     03 FILLER          PIC X(10) VALUE ALL "*".
     03 BALANCE         PIC 9(02)P(03)V9(02) ZEROES.
     06 SUMA-ACUMULADA  PIC S9(10)V9(02) VALUE ZEROES.
     05 FILLER          PIC X(40) VALUE SPACES.
     06 CONTADOR-HOJAS  PIC 9(03) VALUE 0.
     06 ASTERISCOS      PIC X(132) VALUE ALL "*".
```

4.2 INSTRUCCION MOVE

La instruccion MOVE realiza el movimiento de información entre localidades de memoria referenciadas por un nombre de dáto, grupo o registro.

4.2.1 ASIGNACION DE DATOS

La modificación de valores dentro de variables se hace por medio de la asignación de valores a los nombres de datos. Esta asignación se hace por medio de la instrucción move.

```
MOVE {identificador} TO identificador [, identificador] ...
     { literal     }
```

Ejemplos :

```
Procedure division.
Parrafo-uno.
    MOVE 25.85 TO precio.
    MOVE zeroes TO total, subtotal.
    MOVE spaces TO registro.
    MOVE "Ejemplo" TO encabezado.
```

El movimiento de información entre datos enteros se realiza de derecha a izquierda, rellenando de ceros si el campo receptor es mayor o truncando si es menor. Para datos no enteros (numéricos con punto virtual) la parte entera se realiza de derecha a izquierda y la parte fraccionada de izquierda a derecha. Para datos alfanuméricos el movimiento de información se realiza de izquierda a derecha truncando si el campo receptor es de longitud

menor o rellenando de espacios si el campo es mayor.

Cualquier instrucción que como operación final sea la de mover un resultado a una variable contiene clausulas que checa el error de tamaño (ON SIZE ERROR) para datos numéricos ó (ON OVERFLOW) para datos alfabéticos ó alfanuméricos; no así la instrucción MOVE por lo tanto se perderan dígitos o caracteres al mover información.

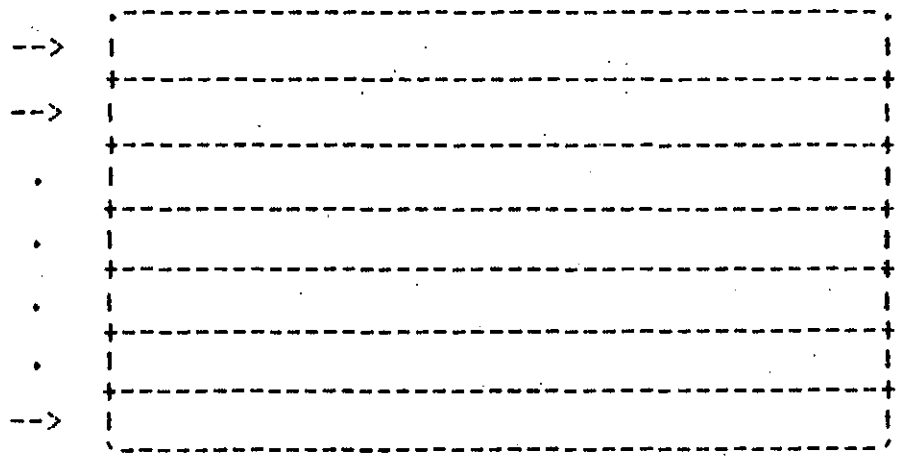
CHAPTER 5

MANEJO DE ARCHIVOS SECUENCIALES

Un archivo es un conjunto de registros asociados en un dispositivo fisico (cinta, disco, tarjetas, etc.) el cual entran al proceso. De forma general el archivo contiene información definida de cierta forma donde se define información para representar caracteristicas unicas.

Un archivo secuencial es aquel donde el acceso a cualquier registro es necesario pasar por los registros anteriores.

ARCHIVO SECUENCIAL



Los datos son referenciados en operaciones de entrada salida en un archivo secuencial de acuerdo a su posición física.

5.1 ESQUEMA GENERAL PARA EL MANEJO DE ARCHIVOS SECUENCIALES

La estructura en COBOL para el manejo de archivos secuenciales es la siguiente.

ESQUEMA GENERAL PARA EL MANEJO DE ARCHIVOS SECUENCIALES

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
    SELECT archivo-logico
           ASSIGN To archivo-fisico
           [; ORGANIZATION Is SEQUENTIAL ]
           [; ACCESS Mode Is SEQUENTIAL  ]
           [; File STATUS Is dato-x      ] .
DATA DIVISION.
FILE SECTION.
FD   archivo-logico

     [ BLOCK  Contains numero { RECORDS    } ]
     [                        { CHARACTERS } ]

     [ RECORD Contains numero { CHARACTERS } ]

     LABEL { RECORD   Is } STANDARD
           { RECORDS Are }

     [DATA RECORD Is   registro-archivo ].

01 registro-archivo.
   03 ...
PROCEDURE DIVISION.
parrafo-inicial.
     ...

          { INPUT  }
          { OUTPUT }
     OPEN { I-O    } archivo-logico
          { EXTEND }

     CLOSE archivo-logico

     READ archivo-logico [Record] [ INTO registro-x ]
          [; At END instrucción ]

     WRITE registro-archivo [FROM registro-x].

     REWRITE registro-archivo [FROM registro-x].
```

5.2 ENVIRONMENT DIVISION

En esta division se define el archivo logico el cual es manejado durante el proceso a un archivo fisico; esta definición depende del equipo de computo donde reside el proceso, y es en esta division donde se modifican los programas para la asignación de dispositivos fisico. Cuando un programa se pasa de un equipo a otro es en esta division donde se realizan los cambios adecuados para su correcto funcionamiento. Por lo anterior se dice que el lenguaje **COBOL** es estandard.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
    SELECT archivo-logico
           ASSIGN To archivo-fisico
           [ ORGANIZATION Is SEQUENTIAL ]
           [ ACCESS Mode Is SEQUENTIAL  ]
           [ File STATUS Is dato-x      ] .
```

5.2.1 INPUT-OUTPUT SECTION

Esta seccion se define la entrada y salida del proceso.

5.2.1.1 FILE-CONTROL

En este parrafo se definen los atributos y el control de los archivos.

5.2.1.1.1 INSTRUCCION SELECT

En esta instrucción se define la liga del archivo logico con archivo fisico. El nombre de archivo-logico es el nombre por el cual se identifica al archivo en el programa. Este nombre de archivo es usuado en instrucciones de entrada salida.

Existe una instruccion SELECT por cada archivo utilizado en el proceso.

5.2.1.1.1.1 ASSIGN

La clausula ASSIGN define el dispositivo al que se referira el archivo lógico. Esta definición es una liga entre el archivo logico y el archivo fisico.

5.2.1.1.1.2 ORGANIZATION

La clausula ORGANIZATION indica el tipo de organización del archivo, si esta clausula se omite, la organización del archivo se considera secuencial.

5.2.1.1.1.3 ACCESS

La clausula ACCESS MODE especifica el modo de acceso al archivo, si esta se omite se considera de tipo secuencial.

5.2.1.1.1.4 FILE STATUS

Esta clausula indica la variable la cual contendra el valor alfanumerico del resultado de cualquier operacion de entrada/salida a este archivo. La variable esta definida en la WORKING-STORAGE SECTION de tipo alfanumerico de tamaño dos. Los códigos estan definidos por el sistema, en forma general el codigo "00" es para definir una operación correcta de entrada salida.

5.3 DATA DIVISION

La definición de los registros asociados a los archivos se definen en esta division.

```
DATA DIVISION.
FILE SECTION.
FD   archivo-logico

     [ BLOCK  Contains numero { RECORDS    } ]
     [                        { CHARACTERS } ]

     [ RECORD Contains numero { CHARACTERS } ]

     LABEL { RECORD   Is } STANDARD
           { RECORDS Are }

     [DATA RECORD Is   registro-archivo ].

01 registro-archivo.
   03 ...
```

5.3.1 FILE SECTION

Esta sección define los archivos lógicos del proceso.

5.3.1.1 PARRAFO FD

Este parrafo indica las caracteristicas del archivo. Existen tantos FD's como SELECT's de la ENVIRONMENT DIVISION.

5.3.1.1.1 BLOCK CONTAINS

Esta clausula define la cantidad de registros o caracteres que van a ser transferidos al realizar operaciones de entrada salida. Esta cantidad esta de acuerdo a las caracteristicas del archivo fisico, a la cantidad de registros del archivo y al tipo de dispositivo. Si esta clausula no e indicada se considera el block contains de un registro.

5.3.1.1.2 RECORD CONTAINS

El número de esta clausula es igual a la cantidad de caracteres que define el registro.

5.3.1.1.3 LABEL RECORDS

Esta clausula define que los registros estan etiquetados conforme a la definisión del sistema, si esta clausula no puede omitirse.

5.3.1.1.4 DATA RECORD

En esta clausula se define el registro asociado a esta definición de archivo.

5.3.1.2 DEFINICION DE REGISTRO

Para cada archivo se le define un registro (con nivel 01) asociado, este registro indica los datos que se relacionaran en alguna operación de entrada salida.

5.4 PROCEDURE DIVISION

En esta division se realizan todas las operaciones sobre los archivos secuenciales.

5.4.1 INSTRUCCION OPEN

La instrucción OPEN abre un archivo.

```
        { INPUT  }
        { OUTPUT }
   OPEN { I-O    } archivo-losico
        { EXTEND }
```

El archivo se abre para leer (INPUT), escribir (OUTPUT), agregar información (EXTEND) o para actualizar información (I-O). El archivo siempre debe estar abierto al realizarse cualquier tipo de operación. Según sea el modo de abrir el archivo se pueden realizar operaciones de entrada salida:

Instrucción	Modo de abrir el archivo			
	INPUT	OUTPUT	I-O	EXTEND
READ	O		O	
WRITE		O		O
REWRITE			O	

5.4.2 INSTRUCCION CLOSE

La instrucción cierra un archivo.

```
CLOSE archivo-losico
```

Cuando se cierra un archivo ya no se puede leer o escribir sobre el archivo. Se pierde la liga entre el nombre lógico y físico.

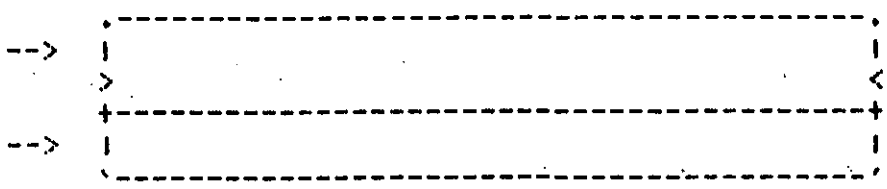
5.4.3 INSTRUCCION READ

La instrucción READ lee del archivo un registro.

```
READ archivo-losico [Record] [ INTO registro-x ]
     [! At END instrucción ]
```

La clausula AT END se activa cuando no se encontra otro registro del archivo. Si se usa la clausula INTO la información se depositara en el registro-x.

ARCHIVO SECUENCIAL



5.4.4 INSTRUCCION WRITE

La instrucción WRITE escribe un registro en el archivo.

```
WRITE registro-archivo [FROM registro-x].
```

La instrucción indica que la referencia al archivo se realiza por medio del registro asociado a ese archivo, a diferencia del READ que la instrucción marca que la lectura se hace por el nombre del archivo-logico.

Si se usa la clausula FROM la información viene del registro indicado. Esta clausula es igual si moviéramos primero del registro-x a registro-archivo y despues grabamos el registro-archivo.

5.4.5 INSTRUCCION REWRITE

La instrucción REWRITE reemplaza un registro por uno nuevo en el archivo (actualización de registro).

```
REWRITE registro-archivo [FROM registro-x].
```

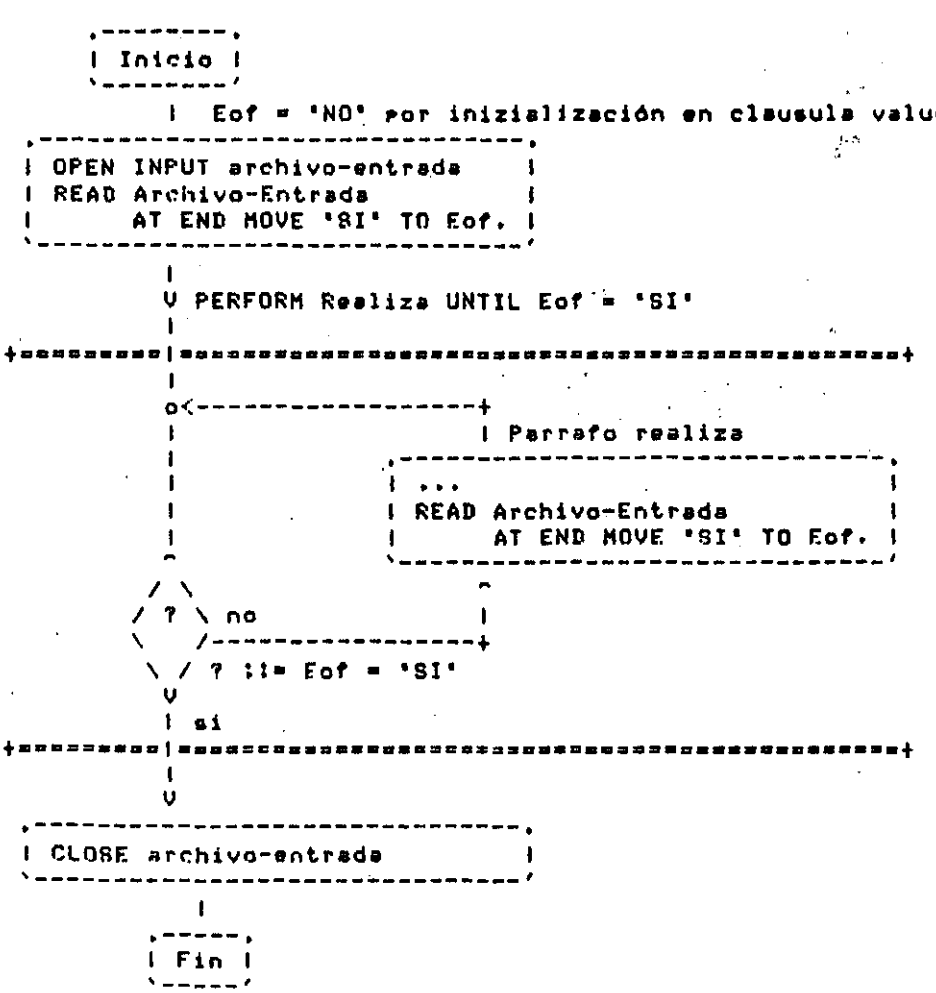
La instrucción indica que la referencia al archivo se realiza por medio del registro asociado a ese archivo, a diferencia del READ que la instrucción marca que la lectura se hace por el nombre del archivo-logico.

Si se usa la clausula FROM la información viene del registro indicado. Esta clausula es igual si moviéramos primero del registro-x a registro-archivo y despues grabamos el registro-archivo.

```
PROCEDURE DIVISION

 IDENTIFICATION DIVISION.
 PROGRAM-ID.    CONTROL.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE CONTROL.
      ASSIGN Archivo-Entrada ASSIGN To DISK.
 DATA DIVISION.
 FILE SECTION.
 FD  Archivo-Entrada
     LABEL Records are STANDARD
     DATE Record is Registro-entrada.
 01  Registro-Entrada.
     03 ...
 WORKING-STORAGE SECTION.
 77 Eof PICTURE Is X(02) VALUE 'NO'
 PROCEDURE DIVISION.
 Inicio.
       OPEN INPUT Archivo-Entrada.
       READ Archivo-Entrada AT END MOVE 'SI' TO Eof.
       PERFORM Realiza UNTIL Eof = 'SI'.
       CLOSE Archivo-Entrada.
       STOP RUN.
 Realiza.
       ...
       READ Archivo-Entrada AT END MOVE 'SI' TO Eof.
```


PROCEDURE DIVISION



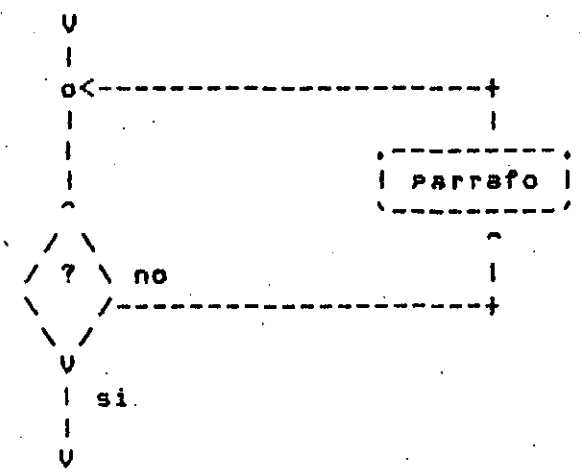
Para reescribir un registro se debe de abrir el archivo con opción I-O, leer el registro con la instruccion READ, alterar el contenido del registro y reescribirlo con está instrucción.

5.4.6 INSTRUCCION PERFORM UNTIL

La instrucción PERFORM sirve para transferir o ejecutar una instrucción o conJunto de instrucciones bajo cierta condición de terminación; al finalizar su acción el proceso continua en la siguiente instrucción del programa.

La instrucción PERFORM en el formato UNTIL lleva el control de proceso en la ejecución de un parrafo hasta definirse una condición por el UNTIL, esta condición es evaluada antes de ejecutar el parrafo; bajo la siguiente estructura.

Estructura de repetición (PERFORM UNTIL).



5.4.6.1 OPERADOR DE RELACION =

El operador = es verdadero cuando los operandos son iguales, en el manejo de archivos se usa cuando el indicador de fin de archivo es activado por la clausula AT END y el control de proceso termina cuando se refiere a esa condicion.

5.4.7 CONTROL DE FLUJO

Una de las estructuras de control de flujo para archivos es aquella donde se define una secuencia de instrucciones bajo el control de los registros del archivo (DO WHILE Hay-Datos). Esta estructura se basa en la clausula AT END de la instrucción READ, de la instruccion de control PERFORM UNTIL y de la relacion =.

CHAPTER 6

CONTROL DE FLUJO

El control de flujo es el mecanismo por el cual los algorimos definen la solución de los problemas de cómputo. El control de flujo depende de ciertas condiciones definidas en el contenido de la información en los nombres de dato, grupo o registro.

6.1 INSTRUCCIONES DE CONTROL

Son instrucciones de control aquellas que dependiendo de alguna condicion transfieren la secuencia de ejecución a cierto punto del programa **COBOL**. Con respecto a la programación estructurada, se reconocen como estructuras de control la estructura de IF y la de DOWHILE. Estas estructuras dependiendo de su acoplamiento y variación se convierten en DOUNTIL y CASE.

COBOL tiene construidas la estructura de IF y una de repeticion llamada PERFORM UNTIL (Que se describio en el capitulo anterior). Además cuenta de una estructura de repetición de una cierta cantidad (PERFORM TIMES).

6.2 INSTRUCCION PERFORM [PARRAFO]

La instrucción PERFORM indica que el control vaya al parrafo descrito y al terminarse el parrafo vuelva a la siguiente instruccion.

```
PROCEDURE DIVISION.
INICIO.
    ...
    PERFORM BLANQUEA-REG.
    ...
BLANQUEA-REG.
    MOVE SPACES TO REG.
```

6.3 INSTRUCCION PERFORM TIMES

La instrucción PERFORM indica que el control vaya al parrafo descrito y al terminarse el parrafo vuelva a la siguiente instrucción despues de pasar tantas veces como el valor asociado a TIMES indique.

```
PROCEDURE DIVISION.
INICIO.
    ...
    PERFORM LINEAS-BLANCAS 24 TIMES.
    ...
LINEAS-BLANCAS.
    MOVE SPACES TO REG-SAL.
    DISPLAY REG-SAL.
```

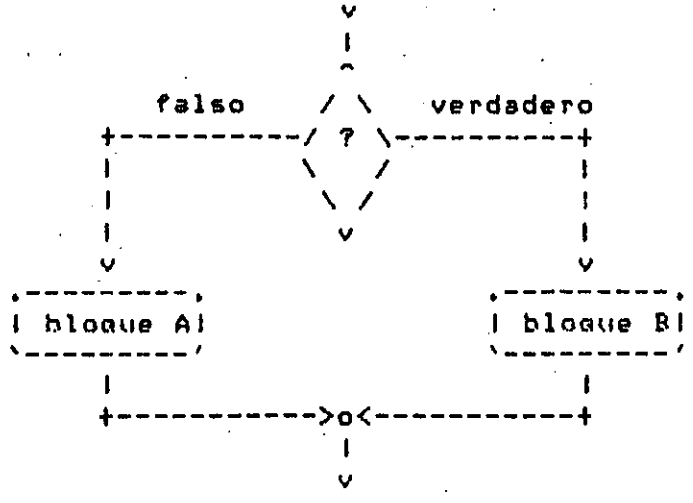
6.4 INSTRUCCION IF

La instrucción IF es la base para el control de flujo. Esta instrucción lleva el control hacia una serie de instrucciones cuando la condición es verdadera y a otra si es falsa. Estas condiciones pueden ser simples o complejas. La condición simple esta definida por algunos de las condiciones descritas en los parrafos siguientes. La condición compleja está compuesta de dos o mas condiciones simples relacionadas por los operadores logicos AND ó OR.

6.4.1 ESTRUCTURA IF

La programación estructurada contempla la estructura de desición (IF) bajo la siguiente figura:

Estructura de desición (IF THEN ELSE).



Al evaluarse l condicón (?)! si es verdadera se ejecuta el bloque B y si es falsa se ejecuta el bloque A.

6.4.2 FORMATO INSTRUCION IF

La instrucción IF tiene la siguiente estructura:

```
IF condición
   { instruccion ... }
   { NEXT SENTENCE   }
ELSE
   { instruccion ... }
   { NEXT SENTENCE   }.
```

Cuando la condición es verdadera ejecuta las instrucciones que se encuentran entre la siguiente del IF y antes del ELSE, al terminar con la última instruccion el flujo continua en la siguiente instruccion despues del punto que cierra el IF! si la condición es falsa se ejecutaran las instrucciones que estan despues del ELSE hasta la última que termina con punto. Cuando se emplea la clausula NEXT SENTENCE solo es puesta en algún bloque del IF. Es conveniente invocar parrafos (con PERFORM) los cuales contiene instrucciones que por su definicion tienen clausulas con invocación a otras instrucciones para evitar colocar puntos que terminarian el IF.

Ejemplo :

```
MOVE 'SI' TO Continua
ACCEPT Capturar
IF Capturar = 'SI'
   PERFORM Captura UNTIL Continua = 'NO'
ELSE
   NEXT SENTENCE.
DISPLAY 'Fin de captura.'
```

6.5 CONDICIONES DE RELACION

Las condiciones de relación comparan dos operandos para definir un estado logico verdadero o falso dependiendo del operador de relación.

6.5.1 OPERADORES DE RELACION

Los operadores de relación evaluan los operandos a comparar y regresan un valor verdadero o falso dependiendo del operador.

```
                    {IS [NOT] GREATER THAN}
{identificador-1}   {IS [NOT] LESS THAN   } {identificador-2}
{literal-1       } {IS [NOT] EQUAL TO     } {literal-2       }
{exp-arit-1      } {IS [NOT] >            } {exp-arit-2      }
                   {IS [NOT] <            }
                   {IS [NOT] =            }
```

6.6 CONDICIONES DE CLASE

Las condiciones de clase nos dice si el operando es del tipo que se indica y define un estado logico verdadero o falso dependiendo del operador de clase.

6.6.1 OPERADOR DE CLASE

Los operadores-de-clase determinan el tipo de dato.

```
                    {IS [NOT] NUMERIC}
{identificador-1} {IS [NOT] ALFANUMERIC}
```

6.7 CONDICIONES DE SIGNO

Las condiciones de signo indica si el operando es positivo, negativo o cero y define un estado logico verdadero o falso dependiendo del operador de signo.

6.7.1 OPERADOR DE SIGNO

Los operadores de signo indican si el dato numérico es zero, positivo o negativo.

```
                          {IS [NOT] POSITIVE}
{ identificador-1       } {IS [NOT] NEGATIVE}
{ expresion-aritmetica  } {IS [NOT] ZERO    }
```

6.8 CONDICIONES LOGICAS.

Las condiciones logicas indica la condición o la relación de dos condiciones se cumple la relación logica y define un estado logico verdadero o falso dependiendo del operador logico.

6.8.1 OPERADORES LOGICOS

Los operadores logicos relacionan condiciones booleanas evaluadas.

```
                [NOT] condición
   condición {AND [NOT] condición }
   condición {OR  [NOT] condición }
```

6.9 NIVEL 88

Este nivel es empleado para definir variables logicas asociadas a datos donde se evaluan al tomar valor el dato, la venaja de su uso radica en la simplificación de instrucciones y definiciones de condicion. Uno de sus empleos es en la validación de informacion, por su valor o rango de valores. Este tipo de variables cuando definen rango de valores desarrollan de una forma igual a una condición compuesta con operadores lógicos OR.

6.9.1 NOMBRES CONDICIONALES DE DATOS

Los nombres condicionales se usan para definir lógicamente el valor verdadero del contenido de una variable. Esto es empleado en el control de flujo del proceso y generalmenete cuando se valida la información.

Ejemplo :

```
DATA DIVISION.
...
01  FIN-DE-ARCHIVO PIC X(02) VALUE 'NO'.
    88 FIN-DE-ARCHIVO-OK VALUE 'SI'.
...
    03 CARRERA-DE-INGENIERIA PIC 9(02).
       88 CARRERA-CORRECTA VALUE 21 THRU 29, 31, 32.
       88 CARRERA-INGENIERIA-CIVIL VALUE 21.
       ...
       88 CARRERA-INGENIERIA-COMPUTACION VALUE 32.
       ...
PROCEDURE DIVISION.
INICIO.
    ...
    PERFORM LEE-ARCHIVO.
    PERFORM EMITE-REPORTE UNTIL FIN-DE-ARCHIVO-OK.
    ...
    STOP RUN
LEE-ARCHIVO.
    READ ARCHIVO-DATOS
         AT END MOVE 'SI' TO FIN-DE-ARCHIVO.
EMITE-REPORTE.
    ...
    IF CARRERA-CORRECTA
       PERFORM REPORTE-SEGUN-CARRERA
    ELSE
       PERFORM ERROR-EN-CARRERA.
    ...
```

```
REPORTE-SEGUN-CARRERA.
    ...
    IF CARRERA-INGENIERIA-CIVIL
       PERFORM REPORTE-CIVILES.
    ELSE
       IF CARRERA-INGENIERA-GEOLOGA
          ...
          ELSE IF CARRERA-INGENIERIA-COMPUTACION
                  PERFORM REPORTE-COMPUTACION.
```

Este nivel define variables logicas que asociadas a un dato toman valor verdadero o falso al tomar valor el dato. La variable asociada al nivel 88 toma el valor de verdadero al tomar el dato el valor especificado por el nivel.

```
01  FIN-DE-ARCHIVO PIC X(02) VALUE 'NO'.
    88 FIN-DE-ARCHIVO-OK VALUE 'SI'.
```

6.10 VARIACIONES A LA INSTRUCCION IF

La instrucción IF puede contener otras instrucciones IF como componentes del bloque THEN o ELSE (IF anidaddos). Al emplear instrucciones IF dentro de otras instrucciones IF es necesario saber donde esta la influencia de cada instrucción IF, para ello es usuada la clausula NEXT SENTENCE para rellenar los bloques que no se empleen para claridad de la instruccion y control de cada instrucción IF.

. Ejemplo :

```
  IF CARRERA-INGENIERIA-CIVIL
     IF ALUMNO-CORRECTO
        PERFORM EVALUA-AVANCE
     ELSE
        PERFORM REPORTA-ERROR
   ELSE
     IF CARRERA-INGENIERIA-GEOLOGA
        ...
        ELSE
            IF CARRERA-INGENIERIA-COMPUTACION
               PERFORM REPORTE-COMPUTACION.
```

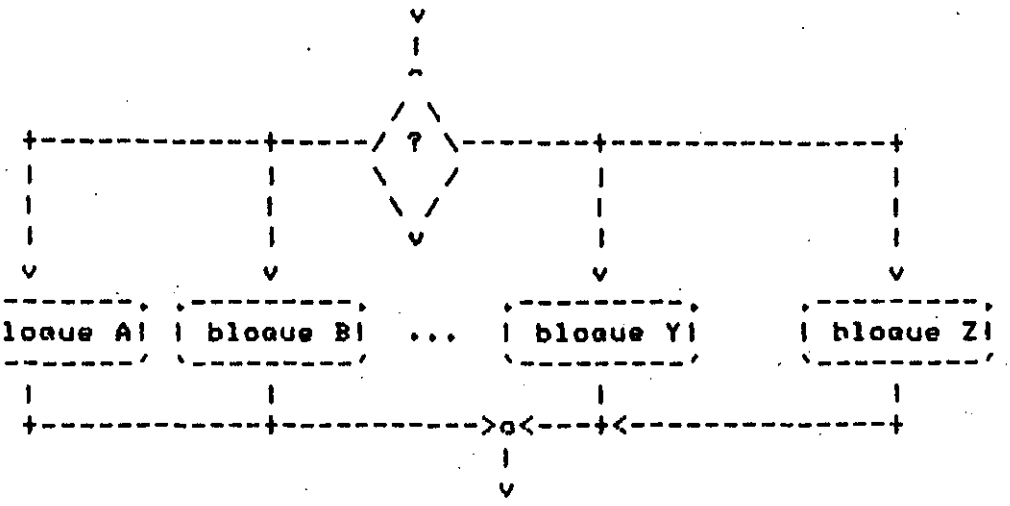
6.11 CONSTRUCCION DE ESTRUCTURA CASE

La estructura CASE es una agrupacion de varios IF's en cascada los cuales definen de una serie de condiciones un bloque a ejecutar.

6.11.1 ESTRUCTURA CASE

La estructura CASE sigue la siguiente figura:

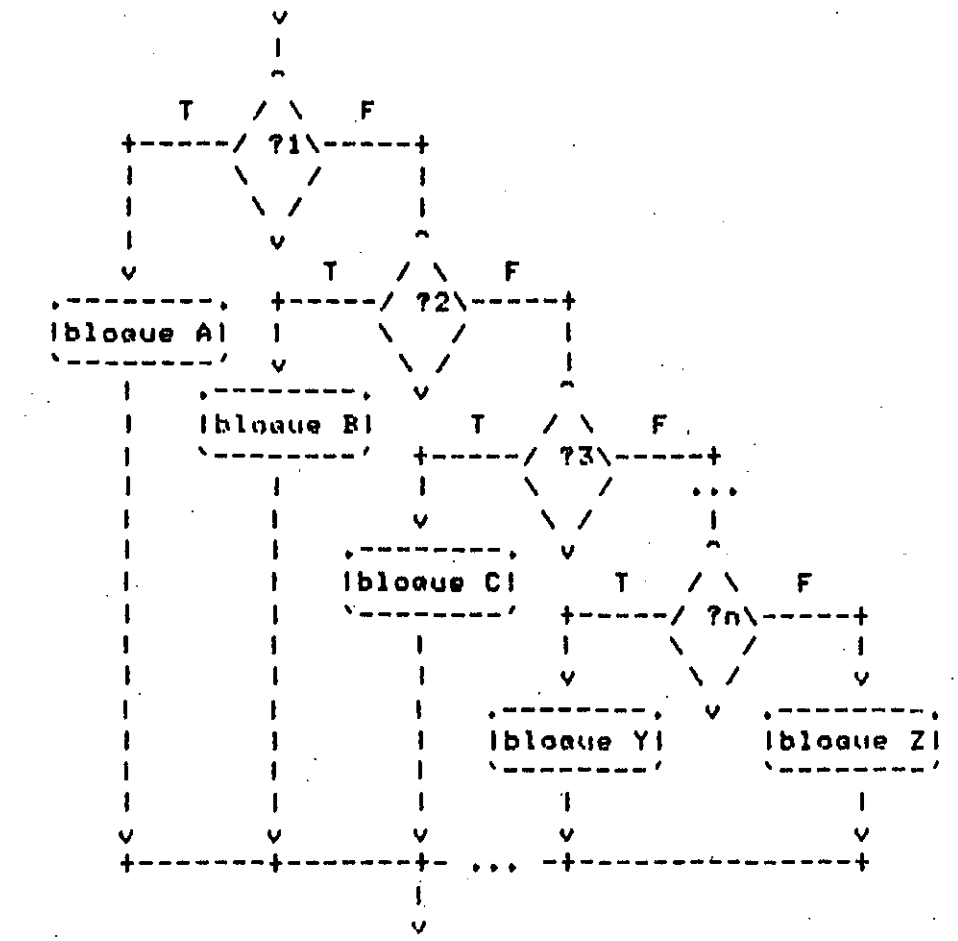
Estructura de desición (CASE).



Dependiendo de la condición (?) se ejecutara el bloque A, B, ... ó Y; en esta estructura se tiene una sección (Bloque Z) que se ejecutara cuando no se cumpla ninguna de las condiciones (OTHERWISE ó ELSE).

Como se indico esta estructura CASE es implementada con IF's anidados de la forma siguiente:

Estructura de desición CASE implementada con IF's anidados.



6.11.2 CASE EN COBOL

La estructura case es implementada en conol con IF's anidados de la siguiente forma:

Estructura case con IF's en **COBOL**

```
IF condición-1
   PERFORM BLOQUE-A
ELSE
   IF condición-2
      PERFORM BLOQUE B
   ELSE
      ...
      IF condición-N
         PERFORM BLOQUE-Y
      ELSE
         PERFORM BLOQUE-Z.
```

CONSTRUCCION DE ESTRUCTURA CASE

Ejemplo :

```
IF CARRERA-INGENIERIA-CIVIL
   PERFORM REPORTA-ALUMNO-CIVIL
 ELSE
   IF CARRERA-INGENIERIA-GEOLOGA
      PERFORM REPORTA-GEOLOGA
      ...
      ELSE
          IF CARRERA-INGENIERIA-COMPUTACION
             PERFORM REPORTE-COMPUTACION
          ELSE
             PERFORM ERROR-EN-CARRERA.
```

CHAPTER 7

INSTRUCCIONES ARITMETICAS

Los procesos de cómputo realizan cálculos matemáticos en la solución de ciertos partes de la programación. Estos cálculos van desde una simple suma hasta complejas evaluaciones.

7.1 INTRODUCCION

Las instrucciones aritmeticas son descritas y empleadas en la PROCEDURE DIVISION por ser está la division donde se realizan instrucciones ejecutables. Estas instrucciones requieren de dos operandos; en algunos formatos uno de los operandos es un nombre de dato donde se deposita el resultado de la operación aritmética. Casi todas las operaciones hacen uso de una ó más clausulas.

7.1.1 CLAUSULA ROUNDED

Esta clausula siempre esta descrita a un nombre de dato donde se deposita el valor final, a este valor final es redondeado a la cifra de menor significancia; esto es al valor final se le suma un medio de la menor cifra significativa.

7.1.2 CLAUSULA ON SIZE ERROR

Esta clausula indica que si el valor final de la operación realizada es mayor en tamaño a la definicion del nombre de dato donde se guardara el resultado entonces se ejecute la instrucción que describe esta clausula, de form general esta instruccion es un PERFORM donde el control va a un parrafo donde se realizaran las instrucciones adecuadas para controlar el error de tamaño.

7.2 INSTRUCCIONES ARITMETICAS

En todo proceso computacional se realizan operaciones aritmeticas, en COBOL se definen las instrucciones aritmeticas en forma general a las de realización basica, (suma, resta, multiplicación y division), asi como se define la instrucción que relaciona formulas (COMPUTE) que si se realizan con funciones basicas se necesitan de variables intermedias y instrucciones aritmeticas basicas.

7.2.1 INSTRUCCION ADD

La operación mas realizada es la suma, en esta instrucción se suman dos operandos o un operando y la suma parcial de dos o mas operandos.

7.2.1.1 VERBO ADD

El verbo ADD realiza la operación de suma bajo los siguientes formatos:

Formato 1:

```
     {identificador-1}[, {identificador-2}] ...
ADD  {literal-1      }[, { literal-2}]   } ] ...
     TO identificador-m    [ROUNDED]
        [,identificador-n [ROUNDED] ...
        [; ON SIZE ERROR instrucción]
```

Si se tiene la instrucción.

```
ADD A B TO C
```

el resultado de A + B + C y A + B + D es guardado en C y D respectivamente.

Formato 2:

```
     {identificador-1}[,{identificador-2}] ...
ADD  {literal-1      }[,{ literal-2}]   } ] ...
     GIVING identificador-m    [ROUNDED]
            [,identificador-n [ROUNDED] ...
            [; ON SIZE ERROR instrucción]
```

Si la instrucción es :

```
ADD A B GIVING C D
```

el resultado de A + B es guardado en C y D.

Las palabras TO y GIVING no pueden usarse ambas en la misma oración.

Para poder usar este verbo en una oración es necesario contar por lo menos, con dos operandos numéricos: el o los sumandos y el operando sobre el que se almacenará el resultado.

Si la opción escogida usa la palabra TO: se sumarán al valor original los valores a sumar.

Si la opción escogida usa la palabra GIVING: el resultado de la suma sustituirá el contenido original del campo.

Ejemplo:

```
ADD Area TO Manejo.
```

Si en el resultado de la operación el número de dígitos decimales excede al espacio disponible en el campo asignado al resultado, éste se truncará, a menos que se use seguido de la palabra ROUNDED (que redondeará la cifra al espacio especificado).

Ejemplo:

```
ADD Area TO Manejo ROUNDED
```

Si el número total de cifras de la parte entera excede al espacio disponible en el campo-dato asignado al resultado, ocurrirá un truncamiento de los dígitos más significativos. La opción ON SIZE ERROR permite manejar esta condición.

Ejemplo:

```
ADD Area, Manejo, Caja GIVING Total ROUNDED,
    ON SIZE ERROR PERFORM  Rutina-O.
```

Cuando se desean sumar dos o más cantidades y hacer algún otro campo de datos igual a la suma, se usa GIVING.

7.2.2 INSTRUCCION SUBTRACT.

Esta instrucción realiza restas de operandos, o resta de un operando y la suma de varios operandos.

7.2.2.1 VERBO SUBTRACT

El verbo SUBTRACT realiza la operación de resta bajo los siguientes formatos:

FORMATO 1:

```
          {identificador-1}[,{identificador-2}] ...
SUBTRACT {literal-1      }[,{ literal-2}]   } ...
          FROM identificador-m   [ROUNDED]
               [,identificador-n [ROUNDED]] ...
          [; ON SIZE ERROR instrucción]
```

En la instrucción:

```
SUBTRACT A B FROM C D
```

el resultado de C - (A + B) y D - (A + B) es guardado en C y D respectivamente.

FORMATO 2:

```
          {identificador-1}[,{identificador-2}] ...
SUBTRACT {literal-1      }[,{ literal-2}]   } ...
          FROM   {identificador}
                 {literal      }
          GIVING identificador-m   [ROUNDED]
                 [,identificador-n [ROUNDED]] ...
          [; ON SIZE ERROR instrucción]
```

En la instrucción:

```
SUBTRACT A B FROM C GIVING D E
```

el resultado de C - (A + B) es guardado en D y E.

Si se desea restar un campo de datos de otro y guardar la diferencia en un tercer campo, debe usarse la opción GIVING.

Todos los operandos antes de la palabra FROM son sumados y el resultado de esta suma es restado del operando de la derecha de la palabra FROM.

Ejemplo:

```
SUBTRACT A,B, FROM C GIVING D.
```

	A	B	C	D
ANTES	1	3	5	29
DESPUES	1	3	5	01

Si no se usa GIVING, el resultado de la diferencia reemplazará al minuendo del campo que sigue a la palabra FROM.

Ejemplo :

```
SUBTRACT AREA-1, AREA-2, AREA-3 FROM AREA-4 ROUNDED,
         ON SIZE ERROR PERFORM RUTINA-3.
```

7.2.3 INSTRUCCION MULTIPLY

La suma repetida de un operando es una multiplicacion, en COBOL la operación multiplicación puede involucrar al operando como variable que contendra al resultado de la multiplicacion. La instrucción MULTIPLY calcula el producto de pares de números y guarda el resultado en un dato especifico.

7.2.3.1 VERBO MULTIPLY

El verbo MULTIPLY realiza la operación de multiplicación bajo los siguientes formatos:

FORMATO 1:

```
          {identificador-1}
MULTIPLY  {literal-1      }
          BY identificador-m [ROUNDED]
             [;identificador-n [ROUNDED] ...
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

```
MULTIPLY A BY B C
```

el resultado de B x C y C x A es guardado en B y C respectivamente.

FORMATO 2:

```
            {identificador-1}
 MULTIPLY   {literal-1       }
            {identificador-1}
             BY {literal-1       }
             GIVING identificador-m   [ROUNDED]
                    [,identificador-n [ROUNDED] ...
                    [; ON SIZE ERROR instrucción]
```

En la instrucción:

MULTIPLY A BY B GIVING C D

el resultado de B X A es guardado en C y D.

NOTA: Si no se usa GIVING, será reemplazado el producto en el multiplicando. (segundo factor)

Este verbo aritmético multiplica dos cantidades y guarda el producto en la segunda o tercera variable.

Ejemplo:

MULTIPLY Piezas BY Costo GIVING Precio

	PIEZAS	COSTO	PRECIO
ANTES	23	147	4444444
DESPUES	23	147	0003381

7.2.4 INSTRUCCION DIVIDE

El numero de veces que un valor contiene otro se le llama division, en varias ocaciones este no es un numero exacto de veces por lo que se usan datos numericos no enteros. La instrucción DIVIDE divide un número entre otro y guarda el cociente y residuo en nombres de dato.

7.2.4.1 VERBO DIVIDE

El verbo DIVIDE realiza la operación de división bajo los siguientes formatos:

FORMATO 1:

```
        {identificador-1}
DIVIDE  {literal-1      }
        INTO identificador-m    [ROUNDED]
             [,identificador-n [ROUNDED] ...
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

DIVIDE A INTO B C

El cociente de B/A y C/A es guardado en B y C respectivamente.

FORMATO 2:

```
        {identificador-1}
DIVIDE  {literal-1      }
        INTO    {identificador-m}
                {literal-1      }
        GIVING identificador-m    [ROUNDED]
             [,identificador-n [ROUNDED] ...
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

DIVIDE A INTO B GIVING C D

el cociente B/A es guardado en C y D.

FORMATO 3:

```
        {identificador-1}
DIVIDE  {literal-1      }
        BY      {identificador-m}
                {literal-1      }
        GIVING identificador-m    [ROUNDED]
             [,identificador-n [ROUNDED] ...
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

DIVIDE A BY B GIVING C D

el cociente A/B es guardado en C y D.

FORMATO 4:

```
        {identificador-1}
DIVIDE  {literal-1      }
        INTO    {identificador-m}
                {literal-1      }
        GIVING identificador-m   [ROUNDED]
        REMAINDER identificador-r
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

```
DIVIDE A INTO B GIVING C REMANIDER D
```

el cociente B/A es guardado en C y el residuo en D.

FORMATO 5:

```
        {identificador-1}
DIVIDE  {literal-1      }
        BY      {identificador-m}
                {literal-1      }
        GIVING identificador-m   [ROUNDED]
        REMAINDER identificador-r
             [; ON SIZE ERROR instrucción]
```

En la instrucción:

```
DIVIDE A BY B GIVING C REMANIDER D
```

el cociente A/B es guardado en C y el residuo en D.

NOTA: Si no se usa GIVING, será reemplazado el cociente en el dividendo por lo tanto, este último operando no podrá ser una literal.

Ejemplo:

```
DIVIDE A INTO B GIVING C ROUNDED,
       ON SIZE ERROR PERFORM RUTINA-4
(Divide B entre A, dando C redondeado, en caso de error
 de tamaño ir a rutina-4).
```

Existe un error cuando el divisior es igual a cero.

7.2.5 INSTRUCCION COMPUTE.

Cuando se requiere de evaluar una formula es preferible utilizar la instrucción COMPUTE el cual agrupa varias operaciones basicas.

7.2.5.1 VERBO COMPUTE

El verbo COMPUTE realiza la operación de evaluación de formulas bajo el siguiente formato:

```
COMPUTE nom-dato-1 [ROUNDED] [,nom-dato-2 [ROUNDED] ...
        = formula
        [;ON SIZE ERROR instrucción]
```

Este verbo ofrece una forma mucho más compacta de especificar operaciones aritméticas.

En la expresión aritmetica:

1. Se pueden usar las constantes figurativas ZERO, ZEROS, ZEROES, una literal numérica o un nombre de dato numérico.

2. Los nombres de dato y las literales son relacionadas por operadores aritmeticos.

3. Una expresión aritmetica puede ser presedida por el operador unuario (-).

4. Las expresiones aritmeticas pueden ser agrupadas por parentesis.

Ejemplo:

```
COMPUTE X = A + (B * C)
```

ó también:

```
COMPUTE X ROUNDED = A + (B * C) ON SIZE ERROR
        PERFORM RUTINA-1
```

7.2.5.2 OPERADORES ARITMETICOS BINARIOS

Los operadores aritmeticos es una convesion para definir operaciones aritmeticas con dos operandos son:

1. + Suma.
2. - Resta.
3. * Multiplicacion.
4. / Division.
5. ** Potenciación.

El valor exponente en la potenciación debe ser una cantidad numerica entera.

7.2.5.3 OPERADORES ARITMETICOS UNUARIOS

Los operadores aritmeticos es una convesion para definir operaciones aritmeticas con dos operandos son:

1. + Es identico a multiplicar por +1 la expresión aritmatica.
2. - Es identico a multiplicar por -1 la expresión aritmetica.

Las expresiones aritméticas siguen las reglas de jerarquia y ejecución son las mismas que el álgebra común.

1. Evaluación de parentesis.
2. Operadores unuarios.

3. Multiplicación y división.

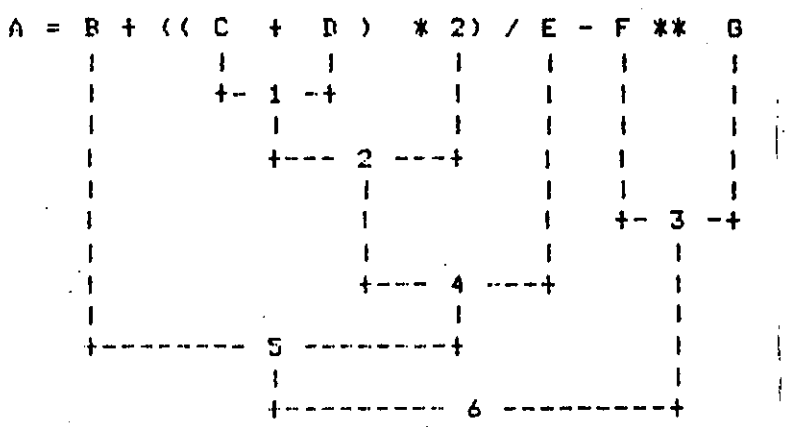
4. Suma y resta.

Si se tienen operaciones con mismo nivel de presedencia estas se evaluaran de izquierda a derecha.

En la expresión:

A = B + ((C+D)*2)/E-F**G

la evaluación es:



CHAPTER 8

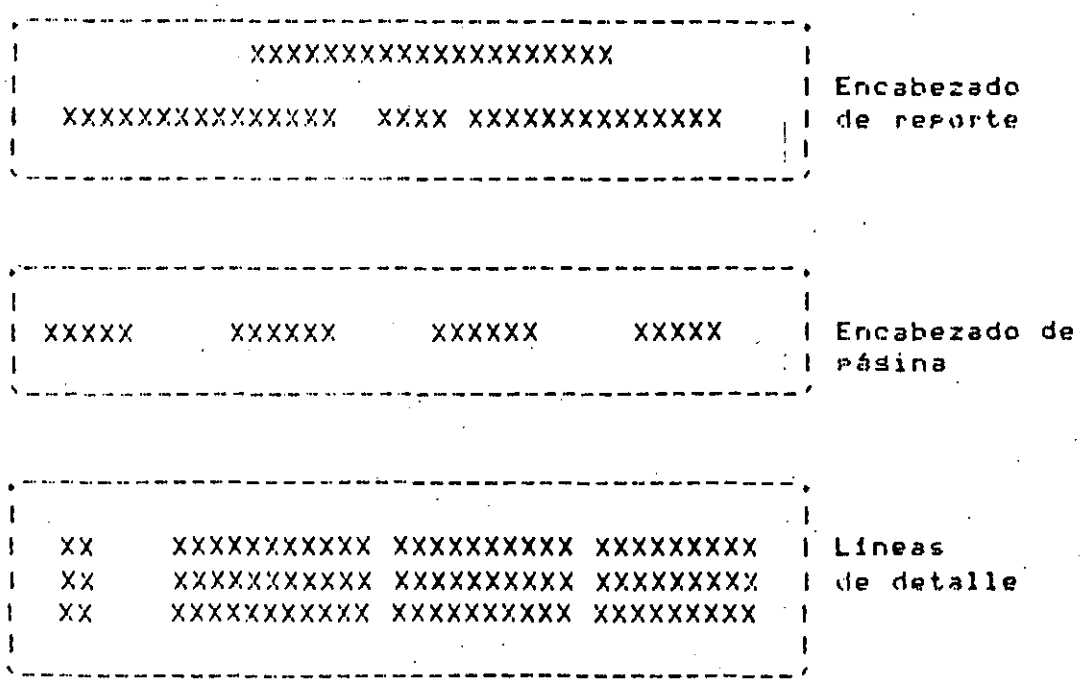
REPORTES

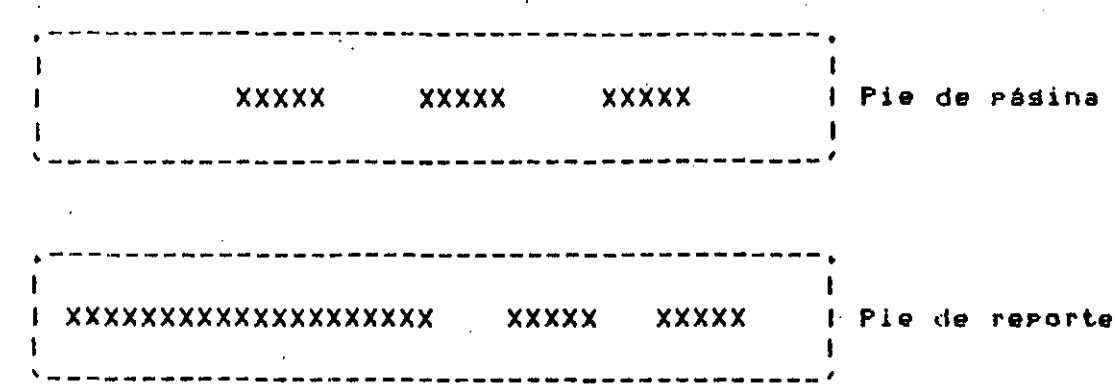
Los reportes son la salida normal de todo sistema de cómputo. Un reporte es información formateada en papel, pantalla, microfilm, etc. de forma que pueda ser interpretada por los usuarios de sistemas de cómputo.

8.1 FORMACION DEL REPORTE

Formar el reporte es diseñar la salida de información. Al presentar un reporte para su analisis o estudio es necesario formar una imagen de la información facil de leer y analizar.

La estructura general de un reporte es:





El encabezado de reporte es el principio de un reporte y generalmente tiene titulos, fecha e información referente al tipo de reporte.

El cuerpo del reporte consiste en una o más páginas. Cada página depende de las condiciones físicas del dispositivo que contendra el reporte. Una página se divide en: encabezado de página, líneas de detalle y pie de página. El encabezado de página incluye los definición del reporte así como la información de la numeración de hoja y conceptos por columna. Las líneas de detalle muestran la información del reporte en uno ó más formatos. En el pie de página se muestran datos que expliquen los conceptos descritos en la página, la descripción de claves utilizadas, sumas parciales por hoja o avance y numeración alterna de las hojas.

El pie de reporte es la parte final del reporte y se muestran las conclusiones finales con referencia a la información mostrada.

El diseño de salida de información es la técnica de usar hojas de imagen de salida y cálculos de campo, para representar encabezados, subencabezados y líneas de detalle.

8.2 ARCHIVOS DE IMPRESION

La estructura en COBOL para el manejo de archivos de impresión es casi idéntica para el manejo de archivos secuenciales y su estructura es la siguiente:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
    SELECT archivo-logico
           ASSIGN To dispositivo-de-impresión ó archivo-fisico
           [; File STATUS Is dato-x      ] .
```

```
DATA DIVISION.
FILE SECTION.
FD   archivo-logico

     LABEL { RECORD   Is } OMITTED
           { RECORDS Are }

     [DATA RECORD Is   registro-archivo ].

01 registro-archivo ...


PROCEDURE DIVISION.
...
     OPEN { OUTPUT } archivo-logico

     WRITE nombre-de-registro [ FROM registro ]
           [ {BEFORE}           { Exp-arit LINE } ]
           [ {AFTER } ADVANCING { Num-ent  LINES} ]
           [                    { PAGE          } ]

     CLOSE archivo-logico
```

8.3 DEFINICION DE ARCHIVO DE REPORTES

La definición de un archivo de reporte es identica a la definición de un archivo secuencial con la diferencia en que el registro asociado solo es un registro alfanumerico, sin división de datos. Estos archivos son asignados al dispositivo de impresion.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
    SELECT archivo-logico
           ASSIGN To dispositivo-de-impresión ó archivo-fisico
           [; File STATUS Is dato-x      ] .


DATA DIVISION.
FILE SECTION.
FD   archivo-logico

     LABEL { RECORD   Is } OMITTED
           { RECORDS Are }

     [DATA RECORD Is   registro-archivo ].

01 registro-archivo ...
...
```

Ejemplo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     ...
     SELECT ARCHIVO-IMPRESION  ASSIGN TO PRINTER.
DATA DIVISION.
..
FD ARCHIVO-IMPRESION
   LABEL RECORDS ARE OMITTED
   DATA RECORD IS REGISTRO-IMPRESION.
01 REGISTRO-IMPRESION PICTURE IS X(132).
```

El tamaño del registro depende del dispositivo de impresión y del tipo de reporte. El registro es de tipo alfanumérico porque es en este registro asociado al archivo de impresión y el registro destino de todos los movimientos de registros de encabezado, subencabezado, detalle, etc. Todos estos registros son descritos en la WORKING-STORAGE SECTION de la DATA DIVISION.

8.4 REGISTRO DE REPORTE

Se llaman registros de reporte los registros de encabezado y pie de reporte; los registros de encabezado y pie de página y los registros de detalle. Estos registros son descritos en la WORKING-STORAGE SECTION y formados con contantes alfanumericas y edición de datos.

8.5 EDICION DE DATOS

Si se representa la información en papel tal como se describe en la computadora se verian todos los caracteres que la describen. La edición de salida de información forma a los datos de impresión una forma losible de interpretar.

Los caracteres de edición son caracteres que definen las caracteristicas de salida de los datos, los cuales definen formas de impresión de valores. Estos caracteres son descritos en registros en la WORKING-STORAGE SECTION.

La función de los datos editados involucra el cambio de forma de los datos. Por ejemplo, se pueden suprimir los ceros no significativos de los datos numericos (los que se encuentran al principio del dato numérico), el uso de comas para la separación de cifras para una lectura lesible de las cantidades numéricas, la insercion de caracteres para denotar cantidades contables.

8.5.1 CARACTERES DE EDICION

Los caracteres de edición son el tipo de caracteres que pueden usarse junto a la cláusula picture para definir caracteristica de salida de la información.

1. Z Indica supresión de ceros no significativos en un campo numerico. Se contabiliza en el tamaño del campo (se coloca a la izquierda). El caracter Z es usado para reemplazar ceros al principio de una cantidad numerica, esto es se pondra un blanco de izquierda a derecha hasta encontrar cualquier digito diferente de cero.

2. 0 (Cero) Indica que se coloquen caracteres ceros, sin que se pierdan caracteres del dato fuente. Se contabiliza en el tamaño del campo.

3. . (Punto) Indica que se coloque un punto decimal, tomando en cuenta la colocación del punto virtual. Al moverse un dato este se alinea con respecto a la parte entera y decimal de la información.

4. , (Coma) Indica que se colocuen comas en el lugar correspondiente, sin que se pierdan caracteres del campo fuente. Se contabiliza en el tamaño del campo. Este caracter es empleado para separar cifras como en las cantidades contables (unidades, miles, millones). Al usuarse junto con el caracter flotante ($, -, +) , el caracter coma (,) es reemplazado por este caracter al indicarse tenerse digitos no significativos.

5. $ Indica que se debe poner un signo $ en el campo numerico editado. Se coloca a la izquierda y se contabiliza en el tamaño del campo. Cuando se usa el caracter $ este se coloca en la posición indicada. Este signo solo puede ser descrito en digitos significativos. El signo $ se dice que flota, cuando se ponen dos o mas signos $ juntos y sustituyen de izquierda a derecha los digitos no significativos por blancos hasta terminar su influencia o encontrarse con el siguiente digito significativo.

6. - (Signo menos) Indica que se ponga un signo menos (-) si el valor del campo es negativo, y si no se pondrá un espacio en blanco. Se puede colocar a la izquierda o derecha del dato y se contabiliza en el tamaño del campo. Este simbolo al igual que signo $ flota si se le coloca a la izquierda del campo.

7. + (Signo más) Funciona igual que el signo menos. Si el valor del campo es negativo se inserta el signo menos. En algunos sistemas de computo se indica con un simbolo + las cantidades positivas o blanco en otros sistemas.

8. CR Indica que se coloque este par de letras a la derecha del campo, siempre que el valor sea negativo, sino dejará espacio en blanco.

9. DB Indica que se coloque este par de letras a la derecha del campo, siempre que el valor sea negativo, sino dejará espacio en blanco.

10. * (Asterisco) Proteccion. Indica que se coloquen asteriscos suprimiendo ceros no significativos.

11. B Indica que se coloque espacios en blanco, sin que se pierdan caracteres del campo fuente. Se contabiliza en el tamaño del campo. Este caracter es empleado para saltar preformas descritas en los reportes o para separar partes de una información.

12. / Indica que se coloque un slash. Se contabiliza en el tamaño del campo. Este caracter es empleado para separar partes de la información.

Los caracteres de edición son asociados a registros que definen encabezados y detalles los cuales son definidos en la data división por medio de la working-storage section.

Ejemplo:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  ENCABEZADO.
    03 FILLER PIC X(10) VALUE SPACES.
    03 FILLER PIC X(27) VALUE 'DEPARTAMENTO DE MATEMATICAS'.
    03 FILLER PIC X(10) VALUE SPACES.
    03 FILLER PIC X(09) VALUE 'PAGINA : '.
    03 PAGINA-SALIDA PIC Z(03).
```

```
01  DETALLE-SALIDA.
    03 FILLER PIC X(03) VALUE SPACES.
    03 NUM-CTA-SAL PIC 9(08).
    03 FILLER PIC X(02) VALUE SPACES.
    03 NOM-SAL PIC X(40).
    03 FILLER PIC X(20) VALUE SPACES.
    03 CAL-FINAL-SAL PIC 9(03).9(02).
```

8.6 INSTRUCCION ACCEPT

La instrucción ACCEPT introduce la fecha, dia y hora a datos numericos o alfanumericos dependiendo del concepto y del formato empleado por el sistema de computo. Estos datos son empleados en los reportes para indicar fecha, dia y hora de elaboración.

```
                              { DATE }
ACCEPT identificador FROM     { DAY  }
                              { TIME }
```

Asi la instrucción:

ACCEPT Fecha-e FROM DATE

guarda la fecha en el nombre de dato Fecha-e con el formato yymmdd; donde yy los dos digitos de la derecha del año, mm el mes y dd el dia.

En la instrucción:

ACCEPT Dia-e FROM DAY

guarda el dia en el nombre de dato Dia-e con el formato yyddd; donde yy representa los digitos derechos del año y ddd el número de dia del año en cuestion.

La instrucción:

ACCEPT Hora-e FROM TIME

guarda la hora en el nombre de dato Hora-e con el formato hhmmsstt; donde hh representa las horas que han transcurrido desde la medianoche contada como cero, mm los minutos, ss los segundos y tt las decimas de segundo.

Para todos estos ejemplos son en una forma especifica, para su sistema consulte el manual de referencia del equipo.

8.7 INSTRUCION WRITE (CON CONTROL DE LINEA Y PAGINA)

La instrucción WRITE con control de línea y página es utilizada para paginar e imprimir líneas en blanco antes o después de escribir los registro de salida.

```
WRITE nombre-de-registro [ FROM registro ]
      [                        {                  } ]
      [ {BEFORE}               { Exp-arit LINE    } ]
      [ {AFTER } ADVANCING     { Num-ent  LINES   } ]
      [                        { PAGE             } ]
```

La clausula FROM registro especifica el registro a imprimir y es identico a mover el registro al registro de salida y luego imprimir en el archivo de salida. La clausula BEFORE y AFTER ADVANCING especifican la acción a realizar antes o despues de escribir respectivamente. El avance puede ser por una línea (1 LINE), varias líneas (n LINES) o salto a la siguiente página. (PAGE).

Ejemplo:

```
PROCEDURE DIVISION.
...
     OPEN OUTPUT ARCHIVO-IMPRESION.
     ACCEPT FECHA FORM DATE.
     MOVE FECHA TO FECHA-SAL.
     PERFORM ENCABEZADO.
     PERFORM LEE-ESCRIBE UNTIL EOF.
     ...
     CLOSE ARCHIVO-IMPRESION.
ENCABEZADO.
     WRITE REGISTRO-IMPRESION FROM ENCABEZADO
           AFTER ADVANCING PAGE.
     WRITE REGISTRO-IMPRESION FROM SUBENCABEZADO
           AFTER ADVANCING 2 LINES.
LEE-ESCRIBE.
     ...
     IF LINEAS > 58
        PERFORM ENCABEZADO.
     WRITE REGISTRO-IMPRESION FROM REGISTRO-DETALLE
           AFTER ADVANCING 1 LINE.
     ...
```

Este tipo de elaboración de reportes necesita el control de impresión de líneas de página para su control; la impresión se lleva tambien bajo un control de la información, asi, si la impresión se lleva por concepto al cambio de cada concepto se deben realizar las acciones encaminadas para su definición.

CHAPTER 9

CARACTERISTICAS DE LOS DATOS

A los datos se les define características relacionadas con algunas de sus propiedades en la DATA DIVISION. Estas características representan cualidades de la información ya sea de forma interna o en su manejo.

9.1 BLANK WHEN ZERO

Se usa esta cláusula para denotar que si el dato tiene como valor cero en la impresión se denotará con blancos independientemente del formato de edición numérico empleado.

Ejemplo:

```
DATA DIVISION.
     ...
     03 RESULTADO PIC ZZ9.99 BLANK WHEN ZERO.
```

9.2 CURRENCY AND DECIMAL-POINT

Con esta cláusula se dice que signo reemplaza al dolar ($) y al punto decimal en los formatos numéricos.

Ejemplo : $2345.45 como F3345,45

Se debe seguir la siguiente forma para su definición.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL NAMES.
        CURRENCY SIGN IS 'F'
        DECIMAL POINT IS COMMA.
```

9.3 USAGE IS { COMPUTACIONAL, COMP, DISPLAY, INDEX }

La representación de los datos numéricos puede ser de diferentes maneras. Esta representación depende del tipo de máquina. En forma estándar se tienen dos tipos que son:

- DISPLAY es del modo carácter (opción por omisión).
- COMPUTACIONAL es el modo numérico.

Ejemplo :

```
DATA DIVISION.
...
      03 ESTADISTICAS  PIC 99 USAGE COMPUTACIONAL.
      03 SUELDO PIC 99 USAGE DISPLAY.
```

9.4 { SYNCHRONIZED, SYNC } { LEFT, RIGTH }

Cuando se usa la cláusula **USAGE** se incrementa la eficiencia del programa, para completar esta efectividad, debe ser usada la cláusula **SYNCHRONIZED**.

Ejemplo :

```
DATA DIVISION.
      ...
      03 VALOR-INVENTARIO PIC 9(02)
         USAGE COMPUTACIONAL SYNCHRONIZED LEFT.
```

9.5 { JUSTIFIED, JUST } RIGHT

Esta opción usada para campos alfabéticos o alfanuméricos solamente es empleada para definir el modo de inserción de información al campo, e indica que la inserción de contenido se efectuará de derecha a izquierda, contrariamente a como se efectua.

Ejemplo :

```
DATA DIVISION.
      ...
      03 ATENCION-USUARIO PIC X(20) JUSTIFIED RIGTH.
```

9.6 SIGN IS { LEADING, TRAILING } SEPARATE CHARACTER

Esta opción se usa para indicar que el signo del número es un carácter que va al principio (LEADING) o al final (TRAILING) del número.

Ejemplo :

```
DATA DIVISION.
     ...
     03 SUMA-ACUMULADA PICTURE IS S9(07)
        SIGN IS LEADING SEPARATE CHARACTER.
```

9.7 RENOMBRADO DE AREA DE DATOS

El renombrar el área de datos se utiliza para proporcionar al proceso la capacidad de reagrupar información de los datos. A excepción del REDEFINES (Se vera en un punto posterior) que redefine un área, este renombramiento se realiza por posición de la información.

Como se notará en el ejemplo siguiente, el área de información comprendida entre apellidos y pago-educacion, se llama también ficha-pago-hacienda. Además, se emplea un número de nivel nuevo (66) y la cláusula RENAMES.

Ejemplo :

```
DATA DIVISION.
...
01 REGISTRO-DE-IMPUESTOS.
   05 NUMERO-DE-SOCIO PIC X(09).
   05 NOMBRE.
      09 PRIMER-NOMBRE PIC X(10).
      09 SEGUNDO-NOMBRE PIC X(10).
      09 APELLIDOS PIC X(25).
   05 IMPUESTOS.
      09 PAGO-PERSONAL PIC 9(06).
      09 PAGO-EDUCACION PIC 9(06).
      09 PAGO-IMSS PIC 9(06).
      09 PAGO-PREDIO-CASA PIC 9(06).
   66 FICHA-PAGO-HACIENDA
      RENAMES APELLIDOS THRU PAGO-EDUCACION.
   ...
Procedure division.
Inicio.
   ...
   Read causantes-empresa into registro-de-impuestos at end ... .
   Write pagos-a-hacienda from ficha-pago-hacienda.
   ...
   Stop run.
```

9.8 MULTIDEFINICION DE NOMBRE DE DATOS

La multiple definición de nombres de datos, o el uso de nombre de datos repetidos solo es permitido en nombres de datos ó datos agrupados bajo un nombre de grupo, y sirve para optimizar el movimiento de datos. (No asi en nombres de registros ni nivel 77, 66 ni 88).

Al definir nombres repetidos existen tres instrucciones que relacionan este tipo de definicion, mover información (MOVE CORRESPONDING), sumar (ADD CORRESPONDING) y resta (SUBTRACT CORRESPONDING); si se desea manejar algun campo de un registro o grupo con nombre identico en otro registro este es nombrado en función del registro especificao en la forma:

nombre-de-dato IN registro

ó

nombre-de-dato OFF registro

el modo de invocar depende del sistema de cómputo.

Ejemplo:

```
DATA DIVISION.
01 reg-aux.
   02 nombre pic x(32).
   02 carrera pic x(02).
   ...
01 reg-ant.
   02 nombre pic x(32).
   02 carrera pic x(02).
   ...
PROCEDURE DIVISION.
...
   ...
   IF carrera IN reg-ant <> carrera IN reg-aux
      ...
```

9.8.1 MOVE CORRESPONDING

Se ocupa para mover un registro (emisor) a otro (receptor), los subniveles receptores que se modificarán son únicamente los que tengan el mismo nombre, los que no cumplan esto quedarán con la misma información.

```
MOVE { CORRESPONDING } identificador-1 TO identificador-2
     { CORR           }
```

Ejemplo :

```
DATA DIVISION.
...
01 REGISTRO-MAESTRO.
   05 NOMBRE PIC X(40).
   05 DIRECCION.
      09 CALLE PIC X(10).
      ...
      09 CODIGO-POSTAL PIC 9(06).
01 REGISTRO-AUXILIAR.
   05 NOMBRE PIC X(40).
   05 DIRECCION.
      09 CALLE PIC X(10).
      ...
      09 CODIGO-POSTAL PIC 9(06).
    ...
PROCEDURE DIVISION.
INICIO.
    ...
    MOVE CORRESPONDING REGISTRO-MAESTRO TO REGISTRO-AUXILIAR.
    ...
    STOP RUN.
```

Es identico si se tuvieran las siguientes instrucciones:

```
MOVE NOMBRE IN REGISTRO-MAESTRO
     TO NOMBRE IN REGISTRO-AUXILIAR.
MOVE DIRECCION IN REGISTRO-MAESTRO
     TO DIRECCION IN REGISTRO-AUXILIAR.
...
```

9.8.2 ADD CORRESPONDING

La suma con nombres identicos tiene el formato:

```
ADD { CORRESPONDING } identificador-1 TO identificador-2
    { CORR          }

[ROUNDED] [ON SIZE ERROR instrucción]
```

Sea la instrucción:

```
ADD CORRESPONDING A TO B
```

con la siguiente estructura de dato

```
DATA DIVISION.
...
01 A.
   02 STOCK         PIC 9(05).
   02 AMOUNT        PIC 9(08).
   02 UNIT-PRICE-A  PIC 9(05).
   02 S-CODE        PIC X(06).
01 B.
   02 S-CODE        PIC X(06).
   02 UNIT-PRICE-B  PIC 9(05).
   02 STOCK         PIC 9(05).
   02 AMOUNT        PIC 9(08).
```

los elementos subordinados al registro A son sumados a los de B y el resultado guardado en B y es identico a:

```
ADD STOCK  OF A TO STOCK  OF B
ADD AMOUNT OF A TO AMOUNT OF B
```

9.8.3 SUBTRACT CORRESPONDING

La suma con nombres identicos tiene el formato:

```
SUBTRACT { CORRESPONDING } identificador-1 FROM
         { CORR          }

identificador-2 [ROUNDED] [ON SIZE ERROR instrucción]
```

Sea la instrucción:

```
SUBTRACT CORRESPONDING A FROM B
```

con la anterior estructura de dato.

los elementos subordinados al registro A son restados a los de B y el resultado guardado en B y es identico a:

```
SUBTRACT STOCK  OF A FROM STOCK  OF B
SUBTRACT AMOUNT OF A FROM AMOUNT OF B
```

9.9 CLAUSULA REDEFINES

La redefinición de área de datos es usada cuando se quiere hacer referencia a una misma área de información. Esto es permite que una misma área de memoria (registro, grupo o campo) se pueda referenciar con más de un nombre. Al usarse la cláusula REDEFINES presenta restricciones mostradas en el apendice. En forma general presenta el siguiente formato:

```
<Nivel> Nombre-de-dato REDEFINES Nombre-de-dato-a-redefinir.
```

Ejemplo :

```
DATA DIVISION.
     ...
     03 NUMERO-DE-CUENTA PIC 9(08).
     03 NUMERO-CUENTA REDEFINES NUMERO-DE-CUENTA PIC X(09).
     ...

PROCEDURE DIVISION.
INICIO.
    ...
    ACCEPT NUMERO-CUENTA.
    IF NUMERO-CUENTA IS NUMERIC
       PERFORM NUMERO-DE-CUENTA-NUMERICO
    ELSE
       PERFORM NUMERO-DE-CUENTA-NO-NUMERICO.
    ...
    STOP RUN.
```

Esta clausula es usuada en forma general a la definicion de tablas constantes como nombres de estados, codigos, puestos, etc. que se analizaran en el capitulo de tablas.

CHAPTER 10

MANEJO DE TABLAS DE UNA DIMENSION

Una tabla es la colección de datos relacionados. Ejemplos de tablas son: la definicion de impuestos, las nombres empleados para diferentes claves (estados, producto, etc.). El manejo de tablas es el empleo de las tablas para el manejo en los algoritmos.

10.1 DEFINICION DE TABLAS (INTRODUCCION)

Una tabla es un conjunto de valores almacenados en posiciones consecutivas y con un solo nombre de dato. Para hacer referencia a un elemento de la tabla, se da el nombre y el subindice que identifica la posición de ese elemento en particular. Los elementos de una tabla unidimensional se enumeran secuencialmente 1,2,3... hasta el último.

La clausula OCCURS permite definir tablas y tiene el siguiente formato fundamental:

```
<nivel> nombre-de-tabla OCCURS n TIMES
        PICTURE tipo(longitud).
```

Ejemplo:

```
DATA DIVISION.
...
01 tabla-ingreso-estado.
   02 ingreso-estado OCCURS 32 TIMES PICTURE 9(06)V9(02).
```

10.2 MANEJO DE TABLAS (INTRODUCCION)

El manejo de las tablas se hace por medio de indices donde el indice indica el lugar donde se encuentra la información deseada. En una tabla de una dimension se necesita solo de un indice.

De la definicion anterior relacionando el indice con alguno de los estados de la republica mexicana (1 Ags, 2 ..., 32 Zas) tenemos que

ingreso-estado(1)

denota en el valor asociado el ingreso del estado de Aguascalientes y

ingreso-estado(32)

denota el valor asociado el ingreso del estado de Zacatecas.

y uno de sus usos seria:

```
DATA DIVISION.
...
01 tabla-ingreso-estado.
   02 ingreso-estado OCCURS 32 TIMES PICTURE 9(06)V9(02).
01 estado pic 9(02).
   88 estado-ok value 1 thru 32.
PROCEDURE DIVISION.
...
   ...
   DISPLAY 'Clave estado : '
   ACCEPT estado.
   PERFORM checa-estado THRU estado-ok.
   DISPLAY 'El ingreso del estado ' estado ' es '
           ingreso-estado(estado).
   ...
```

10.3 DEFINICION DE TABLAS CON INFORMACION CONSTANTE

Algunos algoritmos definen tablas con valores constantes como son, los nombres de los dias de la semana, nombres de los estados, etc. Estas tablas son definidas en **COBOL** haciendo uso de la clausula REDEFINES y la definición de tablas.

DEFINICION DE TABLAS CON INFORMACION CONSTANTE

Ejemplo:

```
DATA DIVISION.
...
WORKING-STORAGE SECTION.
01 DEFINICION-DIAS.
   02 FILLER PICTURE X(10) VALUE "LUNES".
   02 FILLER PICTURE X(10) VALUE "MARTES".
   02 FILLER PICTURE X(10) VALUE "MIERCOLES".
   02 FILLER PICTURE X(10) VALUE "JUEVES".
   02 FILLER PICTURE X(10) VALUE "VIERNES".
   02 FILLER PICTURE X(10) VALUE "SABADO".
   02 FILLER PICTURE X(10) VALUE "DOMINGO".
01 TABLA-DIAS REDEFINES DEFINICION-DIAS.
   02 DIA    PICTURE X(10) OCCURS 7 TIMES.
```

Al evaluar una tabla en una instrucción como:

```
MOVE Dia(numero) TO dia-salida
```

Mueve el valor de la tabla dia indicada por el valor de número al campo dia-salida.

10.4 INSTRUCCION PERFORM VARYING

La instrucción PERFORM indica que el control vaya al parrafo descrito y al terminarse el parrafo vuelva a la siguiente instrucción despues de al cumplirse la condición de salida, esta variación de la instrucción PERFORM lleva un valor que se incrementa en cierto valor que fue inicializado antes de entrar a iterar y termina al cumplirse una condición.

```
PERFORM procedimiento THRU procedimiento
        VARYING identificador
        FROM valor-inicial
        BY   valor-incremental
        UNTIL condicion-de-salida.
```

Por ejemplo en el llenado de una tabla desde un archivo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
     SELECT ARCH-ENT ASSIGN ...
...
DATA DIVISION.
FD ARCH-ENT ...
01 REG-ENT.
   02  VALOR PIC 9(03).
   ...
...
WORKING-STORAGE SECTION.
01 TABLA-X.
   02 TABLA OCCURS 50 TIMES PIC X(10)
01 CONTADORES.
   02 CONT-I PIC 9(02).
01 EOF PIC X(02) VALUE 'NO'.
   88 EOF-OK VALUE 'SI'
   ...
PROCEDURE DIVISION.
...
   ...
   OPEN INPUT ARCH-ENT.
   READ ARCH-ENT AT END MOVE 'SI' TO EOF.
   PERFORM LLENA-TABLA VARYING CONT-I FROM 1 BY 1 UNTIL
           CONT-I > 50 OR EOF-OK.
   ...
LLENA-TABLA.
   MOVE VALOR TO TABLA(CONT-I).
   READ ARCH-ENT AT END MOVE 'SI' TO EOF.
```

10.5 DEFINICION DE TABLAS FIJAS, VARIABLES Y CON INDICE

La clausula OCCURS indica el numero de elementos de una tabla con un identici formato; en esta definición se indica si esta referenciada por un indice usado en el proceso de manejo y busqueda de elementos en la tabla (cuyo proceso se realiza en la PROCEDUE DIVISION).

Formato 1:

```
numero-de-nivel nombre-de-dato OCCURS n TIMES
                [ INDEXED By {nombre-de-indice}...]
```

En este primer formato se definen tablas con longitud fija.

Ejemplo:

Sea una tabla de salarios por mes de un empleado.

```
.--------.-------------------------------------------------.
|No. Emp |       Salarios    por        mes                |
|        |   |   |   |   |   |   |   |   |   |   |   |   |
`--------+-------------------------------------------------'
         ENE FEB MAR ABR MAY JUN JUL AGS SEP OCT NOV DIC
```

Su definición seria:

```
DATA DIVISION.
...
01 REG-SALARIO.
   02 NO-EMP            PIC 9(06).
   02 SALARIO-MES       OCCURS 12 TIMES PIC 9(06)V9(02).
```

Formato 2:

```
numero-de-nivel nombre-de-dato OCCURS n TO m TIMES
                DEPENDING On nombre-de-dato-2
                [ INDEXED By (nombre-de-indice)...]
```

En este formato se define una tabla variable donde el tamano varia de N a M; N representa el minimo número de ocurrencias y M el maximo número; la longitud de la tabla la define la variable nombre-de-dato-2.

Ejemplo:

Se define una tabla variable donde se definen materias con calificacion de un alumno referenciado por su número de cuenta.

```
.-----------+---------------+---     ...      ---.
|No. Cuenta|Numero Materias|Materia-calificacion|
`-----------+---------------+---     ...      ---'
```

Su definición seria:

```
DATA DIVISION.
...
01 Registro-materias.
   02 Numero-de-cuenta    pic 9(08).
   02 Numero-de-materias  pic 9(07).
   02 materias            occurs 1 to 7 times
                          depending on numero-de-materias.
      03 materia          pic 9(04).
      03 calificacion     pic x(01).
```

La clausula INDEXED BY define el nombre-de-indice; este indice el cual guarda el apuntador hacia un elemento de la tabla, es declarado automáticamente.

Asi en las definiciones anteriores podemos asociales un indice con:

```
DATA DIVISION.
...
01 REG-SALARIO.
   02 NO-EMP              PIC 9(06).
   02 SALARIO-MES         OCCURS 12 TIMES
                          INDEXED BY X1
                          PIC 9(06)V9(02).
01 Registro-materias.
   02 Numero-de-cuenta    pic 9(08).
   02 Numero-de-materias  pic 9(07).
   02 materias            occurs 1 to 7 times
                          depending on numero-de-materias
                          indexed by x2.
      03 materia          pic 9(04).
      03 calificacion     pic x(01).
```

10.5.1 MANEJO DE TABLAS

Los elementos de las tablas pureden referenciarse por medio de un apuntador o un indice.

Cuando se referencia con un apuntador este puede ser una literal numérica o un nombre-de-dato. Este valor debe ser mayo o igual a 1 y menor o igual al tamaño maximo de la tabla.

Para las definiciones:

```
Salario-mes(1)
```

nos referimos al salario del empleado en el mes de enero, y en:

```
Salario-mes(N)
```

nos referimos al salario del empleado de un mes cualquiera y que es definido por el valor de N [Si N es igual a 12, nos referimos al salario del mes de diciembre].

Si hacemos uso de indices estos estan descritos al momento de declararse y su su uso esta definido por:

Salario-mes(X1)

el valor de X1 asignado por la instrucción SET y la referencia de la tabla.

10.5.1.1 SET

La instrucción SET se utiliza para manipular el índice asignado a una tabla (OCCURS INDEXED BY). El índice de una tabla sólo se puede operar por medio de la instrucción SET. Se puede incrementar, decrementar o igualarse a una constante, variable u otro índice.

Formato 1.

```
      { identificador-1 }          { identificador-2 }
SET   {                 } ... TO   { nombre-indice-2 }
      { nombre-indice-1 }          { entero-1        }
```

En este formato se le da valor al nombre-de-indice o algun nombre de dato bajo las siguientes reglas:

campo receptor	campo emisor			
	Nombre-indice	nombre-dato	literal	
nombre-indice	valido	valido	valido	
nombre-dato	valido	no valido	no valido	

En la instrucción:

SET X1 TO N

El valor de N es puesto en X1.

En la instrucción:

SET N TO X1

El valor de X1 es salvado en la variable N.

Formato 2.

```
                                { UP BY   } { identificador-3 }
SET  { nombre-indice-3 } ...
                                { DOWN BY } { entero-2        }
```

Esta instruccion incrementa o decrementa el valor del nombre-de-indice en lo indicado por una constante o un nombre-de-dato.

En la instrucción:

```
SET X1 UP BY 3
```

el valor de X1 es aumentado en tres unidades.

y en la instrucción

```
SET X1 DOWN BY 5
```

el valor del indice X1 se decrementa en 5 unidades.

10.5.1.2 MANEJO DE TABLAS CON INDICES

Los indices pueden ser manejador para referenciar informacion en forma directa o relativa.

Ejemplo:

Manejo de indice en forma directa.

```
PROCEDURE DIVISION.
...
  MOVE ZERO TO SALARIO-ANUAL
  MOVE X1 TO 1
  PERFORM SUMA-SALARIO UNTIL X1 > 12.
  ...
SUMA-SALARIO.
  ADD SALARIO-MES(X1) TO SALARIO-ANUAL.
  SET X1 UP BY 1.
```

Ejemplo:

Manejo de indice en forma relativa.

```
PROCEDURE DIVISION.
...
  MOVE ZERO TO SALARIO-ANUAL
  MOVE X1 TO 1
  PERFORM SUMA-SALARIO UNTIL X1 > 12.
  ...
SUMA-SALARIO.
  ADD SALARIO-MES(X1)
      SALARIO-MES(X1 + 1)
      SALARIO-MES(X1 + 2) TO SALARIO-ANUAL.
  SET X1 UP BY 3.
```

10.6 BUSQUEDAS

Al tenerse una tabla se requiere de buscar información. Se entiende por búsqueda el proceso mediante el cual se recorre una tabla con objeto de identificar el, o los elementos, que cumplen con alguna característica.

Un tipo de búsqueda es la búsqueda lineal donde se barre la tabla elemento por elemento hasta encontrase la información o no encontrase por salirse de la tabla, este tipo de busqueda se realiza en una tabla ordenada o desordenada. Otro tipo de busqueda es la busqueda binaria la cual necesita que la tabla este ordenada por alguna condición llave.

10.6.1 BUSQUEDA DE INFORMACION LINEAL

La forma manual de buscar información en una tabla en forma lineal es haciendo uso de control de flujo de proceso de la siguiente forma:

Ejemplo:

Buscar un salario (monto) cualquiera en la tabla de salarios del empleado.

```
PROCEDURE DIVISION.
...
   SET X1 TO 1.
   DISPLAY 'Salario a buscar'
   ACCEPT Salario.
   PERFORM Busca-salario
           UNTIL Salario-mes(X1) = salario OR
                 X1 > 12.
   IF X1 > 12
      DISPLAY 'No se encontro el salario pedido'
   ELSE
      SET N TO X1
      DISPLAY 'El salario corresponde al ' N ' mes'.
   ...
BUSCA-SALARIO.
   SET X1 UP BY 1.
```

10.6.2 BUSQUEDA LINEAL

La busqueda lineal indica que se recorrera toda la tabla hasta encontrar la información deseada o referenciar un indice mayor al tamaño de la tabla.

```
                                  { nombre-indice-1 }
SEARCH identificador-1 [VARYING { identificador-2 }
       [AT END instruccion-imperativa-1] ...
       WHEN condición { instrucción-imperativa-2 }
                      { NEXT SENTENCE            }
```

La sintaxis indica que se busca en la tabla (identificador-1) el elemento que cumpla con la condición que aparece después del WHEN. La tabla sobre la cual se realiza la búsqueda debe tener asociado un indice, pués es en éste donde se almacena el elemento de la tabla que cumple con la condición.

Utilizando esta instrucción para el ejemplo anterior:

Ejemplo:

Buscar un salario (monto) cualquiera en la tabla de salarios del empleado.

```
PROCEDURE DIVISION.
...
   SET X1 TO 1.
   DISPLAY 'Salario a buscar'
   ACCEPT Salario.
   SEARCH Salario-mes
          VARYING X1
          AT END DISPLAY 'No se encontro el salario pedido'
          WHEN Salario-mes(X1) = salario
               SET N TO X1
               DISPLAY 'El salario corresponde al ' N ' mes'.
```

10.6.3 BUSQUEDA BINARIA

La busqueda binaria considera una tabla ordenada ya sea de forma ascendente o descendente, albuscar una información esta tabla es partida a la mitad y se compara este valor con el valor buscado, si los valores son iguales estonces termina la busqueda, si no son igunles entonces se toma como limite este elemento y el antepenultatimo elemento referenciado hacia adelante o atras dependiendo de la comparación y se vuelve a buscar; si la tabla es de un solo elemento esta tabla ya no se puede dividir y termina la busqueda por lo que el elemento a buscar no se encuentra en la tabla.

Ejemplo:

Buscar el elemento 10 de la siguiente tabla en busqueda binaria.

Tabla.

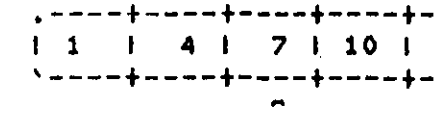
```
.----+----+----+----+----+----+----+----+----.
| 1  |  4 |  7 | 10 | 14 | 20 | 35 | 50 | 80 |
`----+----+----+----+----+----+----+----+----'
```

primer proceso

```
.----+----+----+----+----+----+----+----+----.
| 1  |  4 |  7 | 10 | 14 | 20 | 35 | 50 | 80 |
`----+----+----+----+----+----+----+----+----'
                     ^
                  Mitad de la tabla.
```

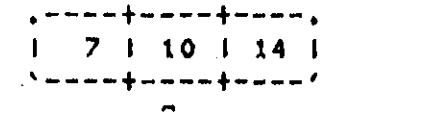
Es el elemento buscado?
NO : Definir la siguiente tabla y repetir el proceso.

segundo proceso



Mitad de la tabla.
Es el elemento buscado?
NO : Definir la siguiente tabla y repetir el proceso.

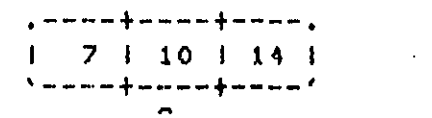
tercer proceso



Mitad de la tabla.
Es el elemento buscado?
SI : Fin de la busqueda.

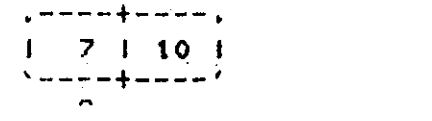
Si el elemento a buscar fuera el 9 se tendria:

tercer proceso



Mitad de la tabla.
Es el elemento buscado?
NO : Definir la siguiente tabla y repetir el proceso.

cuarto proceso



Mitad de la tabla.
Es el elemento buscado?
NO : Definir la siguiente tabla y repetir el proceso.
Como ya no se puede definir otra tabla entonces termina
la busqueda y no se encontro el elemento.

La busqueda binaria en **COBOL** considera la siguiente declaración con los formatos:

Formato 1:

```
numero-de-nivel nombre-de-dato OCCURS n TIMES
          { { ASCENDING  }  KEY IS nombre-de-dato ... }
          { { DESCENDING }                            }
          [ INDEXED Bw {nombre-de-indice}...]
```

Formato 2:

```
numero-de-nivel nombre-de-dato OCCURS n TO m TIMES
          DEPENDING On nombre-de-dato
          { { ASCENDING  }  KEY IS nombre-de-dato ... }
          { { DESCENDING }                            }
          [ INDEXED Bw {nombre-de-indice}...]
```

donde la clausula ASCENDING/DESCENDING KEY determina en que forma esta ordenada la información y por cual llave.

Ejemplo :

Se definen los codigos y nombres de las delegaciones del Distrito Federal.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
...
01  DEF-COD-DEG.
    03 FILLER PIC X(34) VALUE '01DELEGACION ALVARO OBREGON'.
    ...
    03 FILLER PIC X(34) VALUE '16DELEGACION XOCHIMILCO'.
01  TAB-DEF-COD-DEG REDEFINES DEF-COD-DEG.
    02 DELEGACIONES OCCURS 16 TIMES
                    ASCENDING KEY NO-DELEG
                    INDEXED BY NUM-DEG.
       03 NO-DELEG  PIC 9(02).
       03 NOM-DELEG PIC X(32).
```

La instrucción de busqueda binaria tiene el formato:

```
SEARCH ALL identificador-1
       [AT END instruccion-imperativa-1]
       WHEN condición [AND condicion] ...
            { instruccion-imperativa-2 }
            { NEXT SENTENCE            }
```

Las condiciones siempre son operaciones de igualdad.

Ejemplo:

Buscar el nombre de una delegación cuyo codigo se tiene en la variable cod-deg.

```
PROCEDURE DIVISION.
...
  SEARCH ALL DELEGACIONES
         AT END DISPLAY 'NO SE ENCONTRO LA DELEGACION'
                        ' CON CODIGO ' COD-DEG
         WHEN NO-DELEG(NUM-DELEG) = COD-DEG
              DISPLAY  'LA DELEGACION CON CODIGO ' COD-DEF
                       ' ES ' NOM-DELEG(NUM-DELEG).
...
```

CHAPTER 11

MANEJO DE CADENA DE CARACTERES

El manejo de cadena de caracteres al igual que el manejo de cantidades numericas es una de las actividades que se realizan bajo varios procesos para definir datos. Este manejo determina las caracteristicas de los caracteres asi como su uso.

11.1 DEFINICION DE CADENA DE CARACTERES

Una cadena de caracteres es un vector (tabla de una dimension) donde se guardan caracteres.

11.2 INSTRUCCION INSPECT

La instrucción inspect tiene dos actividades que son: las de revisar el contenido de la cadena de caracteres y la de reemplazar información de la cadena.

11.2.1 VERBO INSPECT

La verbo INSPECT realiza las funciones de contar y reemplazar caracteres en un strins.

11.2.2 CONTAR CARACTERES

Para contar cuantos caracteres se encuentran en un strins se tiene el formato:

```
INSPECT identificador-1 TALLYING
   {                    { { ALL       } { identificador-3 } }
   { identificador-2 FOR { { LEADING   } { literal-1       } }
   {                    {       CHARACTERS                 }

   [ {   BEFORE }          { identificador-4 } ]     }     }
   [ {   ------ } INITIAL  { literal-2       } ] ... } ... }
   [ {   AFTER  }                              ]     }     }
```

Este formato indica que la cadena de caracteres identificador-1 es analizada para contar en el identificador-2 todos (ALL) los caracteres iguales a identificador-3 o literal-1; o los caracteres que sean iguales al identificador-3 o literal-1 y que este uno tras de otro al principio (LEADING) de la cadena; o todos los caracteres (CHARACTERS); que este antes (BEFORE) o despues (AFTER) del identificador-4 o la literal-2.

Hay que tener cuidado de saber que valor tiene el identificador-2 antes de la instrucción ya que la instruccion suma 1 por cada caracter analizado y cumple con la condición definida; siempre es preferible mover el valor de zero a este identificador.

Si se desea saber cuantos signos de '$' se encuentran en la cadena TEST poniendo este valor en la variable numérica COUNT-A la instruccion se escribiria:

```
INSPECT TEST TALLYING COUNT-A FOR ALL '$'
```

Si queremos saber cuantos ceros existen a la izquierda del punto decimal y cuantos a la derecha en los contadores COUNT-A y COUNT-B de la cadena TEST, se escribiria:

```
INSPECT TEST TALLYING
        COUNT-A FOR ALL '0' BEFORE INITIAL '.'
        COUNT-B FOR ALL '0' AFTER  INITIAL '.'
```

De la cadena de caracteres TEST y sabiendo que contiene un nombre de persona justificado a la derecha, cual es la longitud del nombre; este valor lo tendremos en la variable COUNT-A haciendo uso de la instrucción:

```
INSPECT TEST TALLYING COUNT-A
        FOR CHARACTERS AFTER INITIAL ' '
```

11.2.3 . REEMPLAZAR CARACTERES

Para remplazar unos caracteres por otros se tiene el formato:

```
INSPECT identificador-1 REPLACING
  {{ CHARACTERS BY { identificador-5}                          }
  {{               { literal-3      }                          }

  {{ { ALL      } { identificador-6 }    { identificador-7 }}
  {{ { LEADING } { literal-4        } BY { literal-5       }}
  {{ { FIRST    } { --------------- }    { --------------- }}

   [ {   BEFORE }          { identificador-8 } ]     }
   [ {   ------ } INITIAL  { literal-6       } ] ... } ...
   [ {   AFTER  }                              ]     }
```

El formato indica que se reemplazara en la cadena de caracteres identificador-1 todos los caracteres (CHARACTERES BY) por el identificador-5 o la literal-3; o todos los caracteres (ALL) ; o los caracteres que esten al principio de la cadena (LEADING), o el primer caracter (FIRST) de lacadena que sean iguales a identificador-6 o literal-4 por el identificador-7 o literal-5; que este antes (BEFORE) o despues (AFTER) del identificador-8 o la literal-6.

El reemplazo de todos los espacios que estan al principio de de la cadena TEST por ceros se realiza con la instrucción:

```
INSPECT REPLACING LEADING ' ' BY '0'.
```

El reemplazo de todos los blancos por ceros en la cadena TEST es con la instrucción:

```
INSPECT REPLACING ALL ' ' BY '0'.
```

El reemplazo del primer cero por un signo '+' se realiza con la instrucción:

```
INSPECT REPLACING FIRST '0' BY '+'.
```

11.2.4 CONTAR Y REEMPLAZAR CARACTERES

Para contar y reemplazar caratcteres en un string se tiene el siguiente formato:

```
INSPECT identificador-1 TALLYING

   {                        { { ALL       } { identificador-3 } }
   { identificador-2 FOR    { { LEADING   } { literal-1       } }
   {                        {       CHARACTERS                  }

   [ {   BEFORE } INITIAL     { identificador-4 } ]     }     }
   [ {   ------ }             { literal-2       } ] ... } ... }
   [ {   AFTER  }                                 ]     }     }

  REPLACING

  {{ CHARACTERS BY { identificador-5}                          }
  {{               { literal-3      }                          }

  {{ { ALL      } { identificador-6 }    { identificador-7 }}
  {{ { LEADING  } { literal-4       } BY { literal-5       }}
  {{ { FIRST    } { --------------- }    { --------------- }}

   [ {   BEFORE } INITIAL     { identificador-8 } ]     }
   [ {   ------ }             { literal-6       } ] ... } ...
   [ {   AFTER  }                                 ]     }
```

Este formato es una conjunción de los anteriores, por lo que, en la cadena de caracteres identificador-1 es analizada para contar en el identificador-2 todos (ALL) los caracteres iguales a identificador-3 o literal-1; o los caracteres que sean iguales al identificador-3 o literal-1 y que este uno tras de otro al principio (LEADING) de la cadena; o todos los caracteres (CHARACTERS); que este antes (BEFORE) o despues (AFTER) del identificador-4 o la literal-2; y se reeplazara en la cadena de caracteres identificador-1 todos los caracteres (CHARACTERES BY) por el identificador-5 o la literal-3; o todos los caracteres (ALL) , o los caracteres que esten al principio de la cadena (LEADING), o el primer caracter (FIRST) de lacadena que sean iguales a identificador-6 o literal-4 por el identificador-7 o literal-5; que este antes (BEFORE) o despues (AFTER) del identificador-8 o la literal-6.

Contar la cantidad de signos '$' en la variable COUNT-A y reemplazar todos los caracteres '$' por '*' despues del primer '$' que se encuentran en la cadena de carateres TEST se realiza con la instruccion:

```
INSPECT TALLIYNG  COUNT-A FOR ALL '$'
        REPLACING ALL '$' BY '*' AFTER INITIAL '$'
```

El numero en formato libre contenido en la variable alfanumerica ABC de seis posiciones y que tiene blancos al principio de la cadena y un signo, se desea saber si es un número y luego convertirlo a numero haciendo uso de la instrucción move a la variable numerica S-ABC cuya definición es S9(06), entonces un segmento de programa seria:

```
PROCEDURE DIVISION.
...
     MOVE ZERO TO PLUS-SIGN, MINUS-SIGN.
     INSPECT ABC
             TALLYING  PLUS-SIGN  FOR ALL '+'
                       MINUS-SIGN FOR ALL '-'
             REPLACING LEADING SPACE BY ZERO
                       FIRST '+' BY ZERO
                       FIRST '-' BY ZERO.
     IF ABC IN NOT NUMERIC
        PERFORM ERROR-EN-VALOR-NUMERICO
     ELSE
        MOVE ABC TO S-ABC
        IF MINUS-SIGN NOT EQUAL 6
           MULTIPLY -1 BY S-ABC.
```

Sea la cadena definida bajo el nombre DIRECCION definida como X(20) y valor 'TOKYO MEXICO SANLUIS' y el FREQ una variable numerica con valor igual a cero, si se tienen la siguientes instrucciones el valor de FREQ y el de DIRECCION tienen los siguientes valores:

```
INSPECT DIRECCION TALLYING FREQ FOR LEADING 'TOKYO'
   <FREQ = 1   DIRECCION = 'TOKYO MEXICO SANLUIS' >

INSPECT DIRECCION TALLYING FREQ FOR ALL SPACE
   <FREQ = 3   DIRECCION = 'TOKYO MEXICO SANLUIS'>

INSPECT DIRECCION TALLYING FREQ
        FOR CHARACTERS BEFORE INITIAL SPACE
   <FREQ = 8   DIRECCION = 'TOKYO MEXICO SANLUIS'>

INSPECT DIRECCION TALLYING FREQ
        FOR CHARACTERS BEFORE INITIAL SPACE
   <FREQ = 8   DIRECCION = 'TOKYO MEXICO SANLUIS'>

INSPECT DIRECCION TALLYING FREQ
        FOR CHARACTERS AFTER INITIAL SPACE
   <FREQ = 22   DIRECCION = 'TOKYO MEXICO SANLUIS'>
```

```
INSPECT DIRECCION REPLACING CHARACTERS BY '*'
        BEFORE INITIAL SPACE
   <FREQ = 8   DIRECCION = '***** MEXICO SANLUIS'>

INSPECT DIRECCION REPLACING LEADING '*****' BY 'OSAKA'
   <FREQ = 8   DIRECCION = 'OSAKA MEXICO SANLUIS'>

INSPECT DIRECCION REPLACING FIRST SPACE BY '-'
   <FREQ = 8   DIRECCION = 'OSAKA-MEXICO SANLUIS'>

INSPECT DIRECCION REPLACING FIRST SPACE BY '-'
   <FREQ = 8   DIRECCION = 'OSAKA-MEXICO-SANLUIS'>
```

11.3 INSTRUCCION STRING

La instrucción STRING concatena dos o mas cadenas de caracteres en una cadena de caracter.

11.3.1 VERBO STRING

El verbo STRING concatena dos o más cadenas de caracteres en una cadena de caracteres, la isntruccion tiene el siguiente formato.

```
STRING { { identificador-1 }
       { { literal-1       }

       DELIMITED By { identificador-2 } }
                    { literal-2       } } ...
                    { SIZE            } }

       INTO identificador-3

       [WITH POINTER identificador-4]

       [ON OVERFLOW  instrucción]
```

Se concatenaran las cadenas de caracter ya sea en el identificador-1 o literal-1 delimitados por el caracter o cadena definida por el identificador-2 o la literal-2 o por su tamaño definido (SIZE) en la cadena de caracteres identificador-3 llevando como apuntador la variable numérica identificador-4, en caso de apuntar a un elemento mayor a la longitud de la cadena se ejecutara la instrucción definida en la clausula ON OVERFLOW.

Consideremos que se tiene un número de telefono (TELEFONO) con el formato 'x-xx-xx-xx' si se desea pasar a una variable de tamaño alfanumerica de tamaño 7 (TELEF) sin los '-' se tiene los siguientes estructuras:

```
DATA DIVISION.
...
01 TELEFONO.
   02 CLAVE-CIUDAD PIC 9(01).
   02 FILLER PIC X(01) VALUE '-'.
   02 CLAVE-SECTOR PIC 9(02).
   02 FILLER PIC X(01) VALUE '-'.
   02 CLAVE-ZONA PIC 9(02).
   02 FILLER PIC X(01) VALUE '-'.
   02 CLAVE-PUERTO PIC 9(02).
01 TELEF PIC X(07).
```

y la instrucción:

```
PROCEDURE DIVISION.
...
   STRING CLAVE-SECTOR DELIMITED BY SIZE
          CLAVE-ZONA   DELIMITED BY SIZE
          CLAVE-PUERTO DELIMITED BY SIZE
          INTO TELEF
```

Si se inserta en una variable (EMPLEADO) a partir del vigesimo caracter verificando que no exista una asignación a otro elemento mayor a la definición:

```
PROCEDURE DIVISION.
...
   MOVE 20 TO APUNTADOR.
   STRING CLAVE-SECTOR DELIMITED BY SIZE
          CLAVE-ZONA   DELIMITED BY SIZE
          CLAVE-PUERTO DELIMITED BY SIZE
          INTO EMPLEADO
          WITH POINTER APUNTADOR
          ON OVERFLOW  PERFORM ERROR-FORMA-EMPLEADO.
   ...
ERROR-FORMA-EMPLEADO.
   DISPLAY 'Error al formar la cadena EMPLEADO'.
```

11.4 INSTRUCCION UNSTRING

Esta instrucción separa de una cadena de caracteres las subcadenas que lo componen ya sea por su tamaño o un delimitador.

11.4.1 VERBO UNSTRING

El verbo UNSTRING considera una cadena de caracteres fuente y varias cadenas de caracteres destino, puede analizarse a partir de un elemento bajo el siguiente formato:

```
UNSTRING identificador-1
         { DELIMITED BY [ ALL ] { identificador-2 }
         {                      { literal-1       }

           { OR [ ALL ] { identificador-3 } }       }
           {            { literal-2       } } ...  } ...

         INTO { identificador-4
              [, DELIMITER IN identificador-5 ]
              [, COUNT     IN identificador-6 ] } ...

         [ WITH POINTER identificador-7]
         [ TALLYING     identificador-8]
         [ ON OVERFLOW  instrucción    ]
```

Esta instrucción mueve las subcadenas de caracteres del identificador-1 delimitadas (DELIMITED BY ALL) por el identificador-2 o la literal-1 ó (OR ALL) el identificador-3 o la literal-2 en cada una de las variables alfanuméricas identificador-4 donde en la variable alfanumérica identificador-5 se guardara el delimitador activo de la cadena (DELIMITER IN) y el número de caracteres transferidos en la variable numérica identificador-6 (COUNT IN) a partir del caracter marcado por la variable numérica identificador-7 (WITH POINTER); al efectuar los movimientos se contaran cuantos campos fueron llenados (identificadores 4's) en la variable numérica identificador-8; si existe un movimiento a algun lugar mayor a la definición de tamaños de las cadenas se ejecutara la instrucción definida en la clausula ON OVERFLOW.

Si se tienen las siguietes estructuras:

```
DATA DIVISION.
...
01 REGISTRO-EMISOR    PIC X(132).
01 REGISTRO-EMPLEADO.
   03 NOMBRE          PIC X(32).
   03 DIRECCION       PIC X(32).
   03 COLONIA         PIC X(32).
   03 CODIGO-POSTAL   PIC X(06).
01 CONTADORES.
   03 NO-NOMBRE       PIC 9(02).
   03 NO-DIRECCION    PIC 9(02).
   03 NO-COLONIA      PIC 9(02).
   03 NO-CODIGO       PIC 9(02).
01 DELIMITADORES.
   03 DEL-NOMBRE      PIC X(01).
   03 DEL-DIRECCION   PIC X(01).
   03 DEL-COLONIA     PIC X(01).
   03 DEL-CODIGO      PIC X(01).
```

y el valor de la variable REGISTRO-EMISOR es

'JOSE RAMIREZ*CALLE 15 NO. 60*SAN PEDRO DE LOS PINOS@03100@'

para separa los datos e introducirlos en REGISTRO-EMPLEADO se realiza la siguiente instrucción:

```
PROCEDURE DIVISION.
...
  UNSTRING REGISTRO-EMISOR
           DELIMITED BY '*' OR '@'
           INTO NOMBRE     COUNT IN NO-NOMBRE
                           DELIMITER IN DEL-NOMBRE
                DIRECCION  COUNT IN NO-DIRECCION
                           DELIMITER IN DEL-DIRECCION
                COLONIA    COUNT IN NO-COLONIA
                           DELIMITER IN DEL-COLONIA
                CODIGO-POSTAL
                           COUNT IN NO-CODIGO
                           DELIMITER IN DEL-CODIGO
                TALLYING IN NO-CAMPOS
                ON OVERFLOW PERFORM ERROR-TRANSFERENCIA.
  ...
 ERROR-TRANSFERENCIA.
       DISPLAY 'ERROR AL SEPARAR LOS DATOS DE ENTRADA'.
```

los valores de las variables seran:

```
NOMBRE         = 'JOSE RAMIREZ'
NO-NOMBRE      = 12
DEL-NOMBRE     = '*'
DIRECCION      = 'CALLE 15 NO. 60'
NO-DIRECCION   = 15
DEL-DIRECCION  = '*'
COLONIA        = 'SAN PEDRO DE LOS PINOS'
NO-COLONIA     = 22
DEL-COLONIA    = '@'
CODIGO-POSTAL  = '03100'
NO-CODIGO      = 5
DEL-CODIGO     = '@'
```

CHAPTER 12

ORDENAMIENTO DE DATOS

Muchos procesos requieren que la información este ordenada por alguna condición de dato. El ordenamiento de información en cobol puede realizarse en vectores o archivos. La instrucciones de ordenamiento en **COBOL** son sobre archivo.

12.1 ORDENAR INFORMACION

Es muchos procesos de información es necesario ordenar la información para que el proceso algoritmico sea correcto, cuantas veces se plantea el objetivo de emitir reportes con tal o cual orden (ordenado por: nombre, edad, sexo-nombre) o es necesario ordenarlo para totalizar cierto grupo de datos (departamento, cliente, sucursal etc.). Ordenar la información es realizar un proceso de acomodaiento de los datos dependiendo del concepto y los datos.

Por ejemplo: se tiene la información de los clientes de cierta compañia de seguros que en forma reducida se tiene sus nombres, edad y sexo.

Sea una muestra de esa información:

NOMBRE	EDAD	SEXO
JOSE MORALES ESCALERA	20	MASCULINO
MARIA SANCHEZ AGUILERA	23	FEMENINO
JOSE MIGUEL LOPEZ	28	MASCULINO
JUANITA LOPEZ ESCOBEDO	34	FEMENINO
LAURA ORTEGA RAMIREZ	10	FEMENINO

La información ordenada ascendente por nombre es:

NOMBRE	EDAD	SEXO
JOSE MIGUEL LOPEZ	28	MASCULINO
JOSE MORALES ESCALERA	20	MASCULINO
JUANITA LOPEZ ESCOBEDO	34	FEMENINO
LAURA ORTEGA RAMIREZ	10	FEMENINO
MARIA SANCHEZ AGUILERA	23	FEMENINO

La información ordenada ascendente por sexo-nombre es:

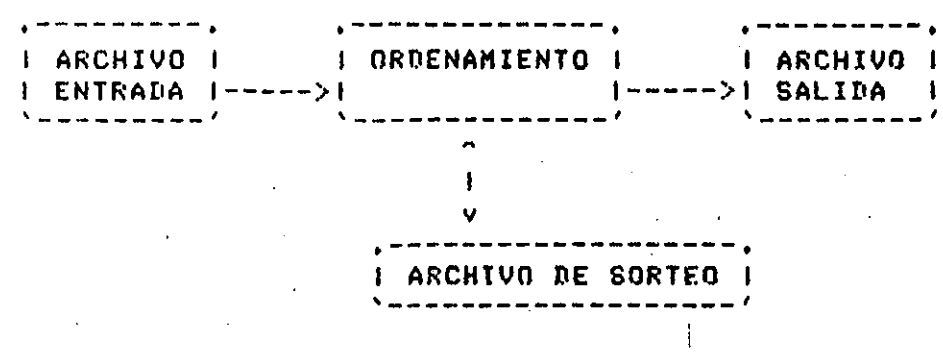
NOMBRE	EDAD	SEXO
JUANITA LOPEZ ESCOBEDO	34	FEMENINO
LAURA ORTEGA RAMIREZ	10	FEMENINO
MARIA SANCHEZ AGUILERA	23	FEMENINO
JOSE MIGUEL LOPEZ	28	MASCULINO
JOSE MORALES ESCALERA	20	MASCULINO

La información ordenada descendente por edad es:

NOMBRE	EDAD	SEXO
JUANITA LOPEZ ESCOBEDO	34	FEMENINO
JOSE MIGUEL LOPEZ	28	MASCULINO
MARIA SANCHEZ AGUILERA	23	FEMENINO
JOSE MORALES ESCALERA	20	MASCULINO
LAURA ORTEGA RAMIREZ	10	FEMENINO

12.2 ORDENACION EN COBOL

Desde el propio lenguaje **COBOL** se puede llevar a cabo el ordenamiento de los registros de un archivo. Para poder llevar a cabo el ordenamiento se requiere especificar un archivo intermedio.



Este archivo intermedio debe estar descrito en la ENVIRONMENT DIVISION por medio de la instrucción SELECT y su definición de archivo y registro en la DATA DIVISION en la sección FILE SECTION por medio del descriptor de archivo SD.

Existen dos modalidades de uso de la instrucción SORT.

1. SORT USING-GIVING. Con este formato uno o más archivos de entrada desordenados resultan en un archivo de salida ordenado. El ordenamiento se puede llevar a cabo ascendente o descendentemente de acuerdo a las llaves especificadas.

2. SORT INPUT OUTPUT PROCEDURE. Este formato define en el archivo de sorteo los registros a ordenar ascendiendolos de los archivos de entrada, así como tambien se puede selecionar del archivo de sorteo algunos registros especificos; este tipo de proceso permite el ahorro de espacio y tiempo al ordenar solo la información necesaria. El archivo se ordenara de acuerdo a las condiciones de ordenamiento y las llaves especificas.

Aunado a los dos anteriores formatos se pueden combinar la entrada de uno y la salida del otro o viceversa.

12.3 DEFINICION DE ARCHIVOS DE SORTEO

La definición de archivos para sorteo se lleva de la misma manera que la definición de archivos secuenciales, con la diferencia que no se difine un parrafo FD por cada archivo de sorteo sino un parrafo SD.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     ...
     SELECT archivo-losico ASSIGN To archivo-fisico.
DATA DIVISION.
FILE SECTION.
..
SD   archivo-losico
     [ DATA ( RECORD Is, RECORDS Are )  registro-sorteo.
01   registro-sorteo.
     03 llave-1 pic x(10).
     03 llave-2 pic 9(02).
     ...
     03 llave-n pic a(20).
```

Las llaves de ordenamiento estan definidas en tamaño y tipo en el registro asociado al archivo de sorteo (llave-1, llave-2, ...).

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
     SELECT ARCH-TYX ASSIGN TO DISK.
     SELECT ARCH-TYY ASSIGN TO DISK.
     SELECT ARCH-SRT ASSIGN TO DISK.
DATA DIVISION.
FILE CONTROL.
FD ARCH-TYX
   LABEL RECORD ARE STANDARD.
01 REG-X.
   03 NOMBRE-X PIC X(32).
   03 EDAD-X   PIC 9(02).
   03 SEXO-X   PIC X(10).
FD ARCH-TYY
   LABEL RECORD ARE STANDARD.
01 REG-Y.
   03 NOMBRE-Y PIC X(32).
   03 EDAD-Y   PIC 9(02).
   03 SEXO-Y   PIC X(10).
FD ARCH-SRT
   DATA RECORD IS REG-S.
01 REG-S.
   03 NOMBRE-S PIC X(32).
   03 EDAD-S   PIC 9(02).
   03 SEXO-S   PIC X(10).
```

12.4 INSTRUCCION SORT USING GIVING

La instrucción SORT con la clausula USING GIVING realiza sorteo de toda la información en el archivo de sorteo.

```
 SORT archivo-logico
      ON { ASCENDING  } KEY llave-x [, llave-y ]...
         { DESCENDING }
      [ ON { ASCENDING  } KEY llave-x [, llave-y ]... ]...
      [    { DESCENDING }                             ]
     USING  archivo-entrada-x [, archivo-entrada-y ] ...
     GIVING archivo-salida.
```

En esta instrucción el contenido del archivo-entrada-x, el archivo-entrada-y etc. son grabados en el archivo de sorteo, ordenandolo conforme al tipo de sorteo (ASCENDING/DESCENDING) de las llaves especificadas (llave-x, ..); para luego vaciar el archivo ordenado en el archivo de salida.

Ejemplo.

Ordenar por nombre ascendente:

```
PROCEDURE DIVISION.
...
SORT ARCH-SRT
    ON ASCENDING KEY NOMBRE-S
    USING ARCH-TYX GIVING ARCH-TTY.
```

En muchas ocaciones el archivo de salida es igual al archivo de salida, configurandose dependiendo del sistema de computo en que se ejecute el programa.

Ejemplo.

Ordenar por nombre ascendente:

```
PROCEDURE DIVISION.
...
SORT ARCH-SRT
    ON ASCENDING KEY NOMBRE-S
    USING ARCH-TYX GIVING ARCH-TTX.
```

12.5 INSTRUCCION SORT INPUT PROCEDURE OUTPUT PROCEDURE

La instrucción SORT con la clausula INPUT PROCEDURE OUTPUT PROCEDURE realiza sorteo de información que es introducida al archivo de sorteo con la instrucción RELEASE y extraida con la instrucción RETURN, el formato de la instrucción RELEASE es identico a WRITE del archivo secuencial y el de la instrucción RETURN es identico a READ del de un archivo secuencial. Cuando se hace uso de este tipo de instrucción es necesario de definir secciones, los cuales cominenzan con la definición de sección (Nombre SECTION), deben tener por lo menos dos parrafos, donde el ultimo contiene solo la instrucción EXIT.

```
SORT archivo-logico
    ON { ASCENDING  } KEY llave-x [, llave-y ] ...
       { DESCENDING }
    [ ON { ASCENDING  } KEY llave-x [, llave-y ]... ]...
    [    { DESCENDING }                             ]
    INPUT  PROCEDURE IS nombre-de-seccion-a
                        [ { THROUGH } nombre-de-b ]
                        [ {   THRU  }             ]
    OUTPUT PROCEDURE IS nombre-de-seccion-x
                        [ { THROUGH } nombre-de-y ]
                        [ {   THRU  }             ]
```

```
nombre-de-seccion-s SECTION.
parrafo-inicial-s.
     ...
     RELEASE registro-sorteo FROM registro-x.
     ...
párrafo final-s.
     EXIT.
nombre-de-seccion-x SECTION.
parrafo-inicial-x.
     ...
     RETURN archivo-sorteo Record
            [ Into  registro-x ] At END instruccion.
     ...
parrafo-final-x.
     EXIT.
```

Ejemplo:

Sortear solo los clientes de sexo = 'masculino', sortearlos por nombre y extraer los que tienen hasta una edad de 20 años:

```
PROCEDURE DIVISION.
...
SORT ARCH-SRT
    ON ASCENDING KEY NOMBRE-S
    INPUT PROCEDURE 01-MASC-ENT THRU 01-MASC-SAL
    OUTPUT PROCEDURE 01-EDAD-ENT THRU 01-EDAD-SAL.


01-MASC-ENT SECTION.
02-ESC-MASC.
        OPEN INPUT ARCH-TTX.
        MOVE 'NO' TO EOF.
        READ ARCH-TTX AT END MOVE 'SI' TO EOF.
        PERFORM 03-GRABA-MASC UNTIL EOF = 'SI'.
        CLOSE ARCH-TTX.
        GOTO 02-MASC-ENT.
03-GRABA-MASC.
        IF SEXO-X = 'MASCULINO'
           RELEASE REG-S FROM REG-X.
        READ ARCH-TTX AT END MOVE 'SI' TO EOF.

02-MASC-ENT.
        EXIT.

01-MASC-SAL SECTION.
02-MASC-SAL.
        EXIT.
```

INSTRUCCION SORT INPUT PROCEDURE OUTPUT PROCEDURE

```
01-EDAD-ENT SECTION.
02-ESC-EDAD.
        OPEN OUTPUT ARCH-TTY.
        MOVE 'NO' TO EOF.
        RETURN ARCH-SRT AT END MOVE 'SI' TO EOF.
        PERFORM 03-GRABA-EDAD UNTIL EOF = 'SI'.
        CLOSE ARCH-TTY.
        GOTO 02-EDAD-ENT.
03-GRABA-EDAD.
        IF EDAD-S < 21
           WRITE REG-Y FROM REG-S.
        RETURN ARCH-SRT AT END MOVE 'SI' TO EOF.

02-EDAD-ENT.
        EXIT

01-EDAD-SAL SECTION.
02-EDAD-SAL.
        EXIT
```

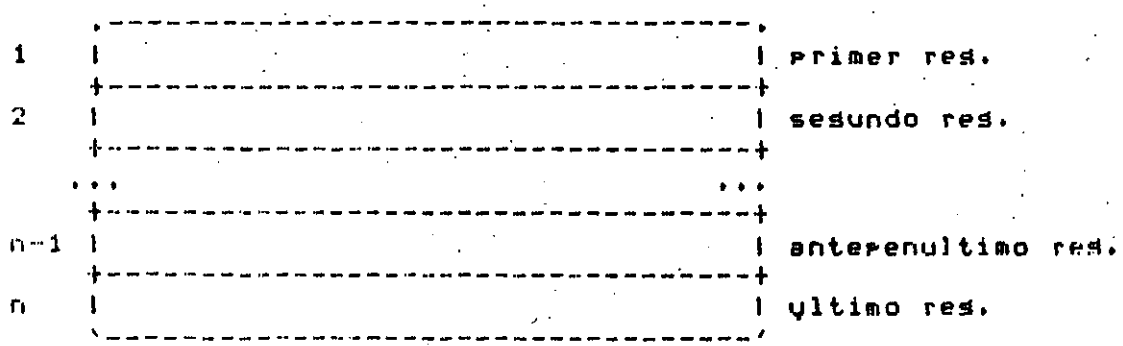
CHAPTER 13

ARCHIVOS RELATIVOS

Los archivos relativos son utilizados para referenciar registros identificados por su número de posición dentro del archivo. Estos archivos son empleados cuando se conoce de la información datos que identifiquen al registro dentro del archivo por su posición.

13.1 ARCHIVOS RELATIVOS

Un archivo relativo es aquel archivo el cual para referirse a un registro se hace por medio del numero de colocación del registro en el archivo. Un archivo relativo puede accesarse secuencialmente o en forma relativa.



Los archivos usa las direcciones asignadas a cada registro al momento de usarse el archivo, estas direcciones son valores numéricos de 1 al máximo número asignado por el control de archivos. Cada dirección son direccionadas por la llave relativa.

Un archivo de organización relativo puede ser manejado con un acceso secuencial, relativo o dinámico (secuencia y/o relativo).

13.2 ESTRUCTURA DE UN ARCHIVO RELATIVO

El manejo de archivos relativos considera la siguiente estructura general:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    ARCHIVOSREL.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
     SELECT archivo-logico ASSIGN To archivo-fisico
            ORGANIZATION Is RELATIVE
            [; ACCESS Mode Is
            { (SEQUENTIAL)                                    }]
            { { RANDOM  } RELATIVE Key is nombre-dato-1}]
            { { DYNAMIC }                                     }]
            [ FILE STATUS is nombre-dato-2 ]

DATA DIVISION.
FILE SECTION.
FD archivo-logico
   [DATA RECORD IS registro-archivo]
01 registro-archivo.
   03 ...
WORKING STORAGE SECTION.
01 REGISTROS-RELATIVOS.
   03 NOMBRE-DATO-1        PIC 9(06).
   03 NOMBRE-DATO-2        PIC X(02).

PROCEDURE DIVISION.
parrafo-inicial.
    ...
    MOVE valor TO nombre-dato-1
         { INPUT  }
    OPEN { OUTPUT } archivo-logico
         {  I-O   }
    READ archivo-logico [NEXT] Record [ INTO registro-x ]
                       [; At END instrucción ]
    READ archivo-logico Record [ INTO registro-x ]
                       [; INVALID KEY instrucción ]
    WRITE registro-archivo FROM registro-x
                       [; INVALID Key instrucción ]
    REWRITE registro-archivo FROM registro-x
                       [; INVALID Key instrucción ]
    DELETE archio-logico RECORD
                       [; INVALID Key instrucción ]
    START archivo-logico [KEY { IS EQUAL TO } nombre-dato-1
                              { IS =        }
                       [; INVALID Key instrucción ]
    CLOSE archivo-logico
```

13.3 DEFINICION DE ARCHIVO.

Un archivo relativo se identifica de manera similar al archivo secuencial; se le indicará su forma de organización y su modo de acceso, así como algunas características y su estructura de datos en el registro asociado. Se debe de definir un dato en la WORKING-STORAGE de tipo numerico donde se tendra el valor del numero de registro a accesar.

13.3.1 DECLARACION DE ARCHIVO

La declaración de archivo se realiza en la ENVIRONMENT DIVISION donde se describe la organización y el modo de acceso; así como su asignación al medio exterior.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE CONTROL.
     SELECT archivo-logico ASSIGN To archivo-fisico
            ORGANIZATION Is RELATIVE
            [; ACCESS Mode Is
            { (SEQUENTIAL)                                  }]
            { { RANDOM  } RELATIVE Key is nombre-dato-1)     ]
            { { DYNAMIC }                                   }]
            [ FILE STATUS is nombre-dato-2 ]
```

El archivo-logico es asignado a su definición como archivo-fisico y este nombre depende del sistema de cómputo; se define su organización (ORGANIZATION) relativa y modo de acceso (ACCESS MODE IS), si no se especifica el modo de acceso se considera secuencial (SEQUENTIAL), la llave de acceso esta descrita por el nombre-dato-1 de tipo numérico y el estado al realizar una operación de entrada/salida a este archivo en la variable alfanumerica nombre-dato-2; estas variables son descritas en la WORKING-STORAGE SECTION.

Para el acceso directo (RANDOM) ó secuencial-directo (DYNAMIC) es obligatorio definir la llave (RELATIVE KEY IS) nombre-dato-1.

El nombre-dato-2 de la clausula FILE STATUS IS tiene el valor de la operación de archivo. Si tiene el valor de '00' se tiene exito en la operación de archivo; si tiene cualquier otro valor se detecto un error en la operación de archivo y es necesario realizar algunas instrucciones de control.

13.3.2 DEFINICION DE ARCHIVO Y REGISTRO

Las caracteristicas del archivo asi como sus registro son definidos en la DATA DIVISION.

```
DATA DIVISION.
FILE SECTION.
FD archivo-logico
   ...
   [DATA RECORD IS registro-archivo]
01 registro-archivo.
   03 ...
WORKING STORAGE SECTION.
01 REGISTROS-RELATIVOS.
   03 NOMBRE-DATO-1        PIC 9(06).
   03 NOMBRE-DATO-2        PIC X(02).
```

En la DATA DIVISION se definen caracteristicas referentes al archivo relativo, asi como el registro asociado, el registro de llave y de estado son definidos en la WORKING-STORAGE SECTION.

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT ARCH-REL ASSIGN TO DISK
            ORGANIZATION IS RELATIVE
            ACCESS MODE IS DYNAMIC
                   RELATIVE KEY IS KEY-REL
           FILE STATUS IS STA-REL.
```

```
DATA DIVISION.
FILE SECTION.
FD ARCH-REL
   DATA RECORD IS REG-REL.
01 REG-REL.
   03 NOMBRE-R PIC X(32).
   03 EDAD-R   PIC 9(02).
   03 SEXO-R   PIC X(10).
WORKING-STORAGE SECTION.
01 REGISTROS-REL.
   03 KEY-REL  PIC 9(06).
   03 STA-REL  PIC X(02).
```

13.4 MANEJO DE ARCHIVOS RELATIVOS

El manejo de los archivos relativos es identico al de los archivos secuenciales con la siguiente caracteristica distintiva que es el numero de registro a accesar, por lo tanto esta variable asociada esta evaluada con el numero de registro deseado, las instrucciones de lectura y escritura esta acompañadas de una clausula de INVALID KEY para definir un proceso al detectarse

error en el acceso.

13.4.1 ABRIR UN ARCHIVO RELATIVO

Un archivo relativo se abre para leer y/o escribir.

La instrucción OPEN abre un archivo bajo el siguiente formato:

```
      { INPUT  }
OPEN  { OUTPUT } archivo-logico
      { I-O    }
```

Las instrucciones de entrada salida dependen del modo de abrir el archivo y de accesar.

MODO DE ACCESO	INSTRUCCION	MODO DE ABRIR EL ARCHIVO		
		INPUT	OUTPUT	I-O
SECUENCIAL	READ	0		0
	WRITE		0	
	REWRITE			0
	DELETE			0
	START	0		0
RANDOM	READ	0		0
	WRITE		0	0
	REWRITE			0
	DELETE			0
	START			
DYNAMIC	READ	0		0
	WRITE		0	0
	REWRITE			0
	DELETE			0
	START	0		0

13.4.2 CERRAR UN ARCHIVO RELATIVO

Cuando no se desea utilizar más el archivo relativo es cerrado con la instrucción:

```
CLOSE archivo-logico
```

Al cerrarse el archivo ya no se puede hacer ninguna operación de archivo.

13.4.3 LECTURA DE UN ARCHIVO RELATIVO

Para leer un archivo relativo este debe ser abierto para leer (INPUT) o leer y escribir (I-O). Dependiendo del modo de acceso se tienen dos tipos de lectura.

13.4.3.1 LECTURA SECUENCIAL

Para leer secuencialmente un archivo este es cuando el modo de acceso es secuencial (ACCESS MODE IS SEQUENTIAL) y abierto para leer INPUT ó de modo dinamico (ACCESS MODE IS DYNAMIC) y abieto para lectura (INPUT) o lectura escritura (I-O) para realizar esta última asignación se debe tener ya posecionado el registro desde el cual se leera secuencial (START). Este tipo de lectura tiene el formato:

```
READ archivo-logico [NEXT] Record [ INTO registro-x ]
                    [; At END instrucción ]
```

Como se notara tiene el mismo formato que la lectura de un archivo secuencial con la clausula de detección de fin de archivo (AT END).

13.4.3.2 LECTURA RELATIVA

Para leer de forma relativa un archivo este es cuando el modo de acceso es relativo (ACCESS MODE IS RANDOM y abierto para leer INPUT ó de modo dinamico (ACCESS MODE IS DYNAMIC) y abierto para lectura (INPUT) o lectura escritura (I-O). Este tipo de lectura tiene el formato:

```
READ archivo-logico Record [ INTO registro-x ]
                    [; INVALID KEY instrucción ]
```

Para realizar esta función debe de evaluarse el numero de registro a leer; si el registro no existe entonces se activa la clausula INVALID KEY ejecutandose la instrucción asociada.

13.4.4 ESCRIBIR UN REGISTRO

La entrada de información a un archivo relativo es una de las funciones cuando se abre para escribir (OUTPUT) para cualquier tipo de acceso. El formato es el siguiente:

```
WRITE registro-archivo FROM registro-x
                    [; INVALID Key instrucción ]
```

Antes de grabar se debe evaluar el registro donde se va efectuar la operación de archivo. Cuando se quiere grabar un registro que ya existe se ejecutara la instrucción activada por la clausula de invasión de llave (INVALID KEY).

13.4.5 ACTUALIZAR UN REGISTRO

Sobre escribir en un registro que ya tiene información en un archivo relativo se realiza con la siguiente instrucción:

```
REWRITE registro-archivo FROM registro-x
                    [; INVALID Key instrucción ]
```

Para actualizar un registro se debe abrir como entrada salida (I-O). Si el modo de acceso es secuencial primero debe leerse el registro a modificar. Si el modo de acceso es relativo o dinamico se debe evaluar primero la llave de acceso antes de realizar la operación de actualización.

Se ejecutara la instrucción descrita por la clausula INVALID KEY si existe un error al momento de realizar la operación o el registro no existe.

13.4.6 BORRAR UN REGISTRO

Borrar un registro de un archivo relativo es quitar la información que se encuentra en el archivo; con el siguiente formato:

```
DELETE archio-losico RECORD
                    [; INVALID Key instrucción ]
```

Para borrar un registro se debe abrir como entrada salida (I-O). Si el modo de acceso es secuencial primero debe leerse el registro a modificar. Si el modo de acceso es relativo o dinamico se debe evaluar primero la llave de acceso antes de realizar la operación de actualización.

Se ejecutara la instrucción descrita por la clausula INVALID KEY si existe un error al momento de realizar la operación o el registro no existe.

13.4.7 POSECIONARSE EN UN REGISTRO

Cuando se lee secuncialmente un archivo cuando es abierto para entrada salida (I-O) y modo relativo (RELATIVE) o dinamico (DYNAMIC) se debe posecionar del registro a partir de donde se va a leer; bajo el siguiente formato:

```
START archivo-logico [KEY { IS EQUAL TO } nombre-dato-1
                          { IS =        }
                     [; INVALID Key instrucción ]
```

El apuntador al registro se posecionará donde marca el valor depositado en nombre-dato-1 que es la llave del archivo relativo; por lo que debe estar evaluada antes de realizar la operación.

Se ejecutara la instrucción descrita por la clausula INVALID KEY si existe un error al momento de realizar la operación o el registro no existe.

Ejemplo:

```
PROCEDURE DIVISION.
...
    OPEN I-O ARCH-REL.
    DISPLAY 'REGISTRO A OPERAR : '
    ACCEPT KEY-REL
    READ ARCH-REL
         INVALID KEY DISPLAY 'NO EXISTE EL REGISTRO '
    IF STA-REL = '00'
       DISPLAY 'Los valores del registro son: '
       DISPLAY ' NOMBRE : ' NOMBRE-REL
       DISPLAY ' EDAD   : ' EDAD-REL
       DISPLAY ' SEXO   : ' SEXO-REL.
    ...
```

APPENDIX A

BIBLIOGRAFIA DE COBOL

1. INFORMATION SYSTEMS THROUGH COBOL.

   A. S. PHILIPPAKIS.

   LEORNAD J. KAZNIER.

   MC. GRAW HILL.

2. COBOL ESTRUCTURADO. CON APLICACION A LOS NEGOCIOS.

   A. S. PHILIPPAKIS.

   LEORNAD J. KAZNIER.

   MC. GRAW HILL.

3. ADVANCED COBOL.

   A. S. PHILIPPAKIS.

   LEORNAD J. KAZNIER.

   MC. GRAW HILL.

APPENDIX B

MODULOS DE COBOL.

Los módulos principales que componen el lenguaje COBOL son :

1. Núcleo.
2. Manejo de tablas.
3. Manejo de archivos secuenciales.
4. Manejo de archivos relativos.
5. Manejo de archivos indexados.
6. Clasificación e intercalación de información.
7. Generador de reportes.
8. Segmentación.
9. Librerías.
10. Depurador.
11. Comunicación entre programas.
12. Comunicación entre sistemas de cómputo.

4. MANUAL DE REFERENCIA DE COBOL.

5. MANUAL DE CONSULTA DE COBOL.

6. MANUALES DE PROGRAMACION DE COBOL.

7. TARJETA DE CONSULTA RAPIDA DE COBOL.

APPENDIX C

DESARROLLO DE SISTEMAS

PROGRAMACION.

En la elaboración de cualquier programa en **COBOL** es necesario definir:

1. El objetivo.

2. Los datos necesarios. (Entradas).

3. Los datos de salida. (Resultados).

4. Los dispositivos de entrada, salida y almacenamiento.

5. El método de solución.

En la definición de la información que se procesará se especifican las características, de tipo (alfabético, numérico ó alfanumérico) y tamaño (cuantos caracteres máximo puede tener un dato).

En todo programa existe cierta información cuyo contenido es definido dentro del programa (tales como los encabezados de los reportes que se emitiran) y otra información cuyo contenido varía (más no su característica) y es proporcionada al momento de la ejecución del programa (por ejemplo, los nombres de los empleados al elaborar una nómina). En este último caso los programas son prácticamente independientes del contenido de la información a procesar

11. B Inserción de espacios o blancos.

12. / Inserción de slash para separación de datos.

Caracteres de puntuación.

1. , Coma.

2. ; Punto y coma.

3. . Punto.

4. : Dos puntos.

5. " Comillas.

6. (Paréntesis izquierdo.

7.) Paréntesis derecho.

8. b Espacio o blanco.

9. ' Apóstrofe.

10.] Paréntesis cuadrado derecho.

11. [Paréntesis cuadrado izquierdo.

12. / Diagonal (Slash).

Operadores aritmeticos.

1. + Suma.

2. - Resta.

3. * Multiplicación.

4. / División.

5. ** Exponenciación.

Caracteres de relación.

1. > Mayor que.

2. < Menor que.

3. = Igual.

APPENDIX E

PALABRAS RESERVADAS DE COBOL

Las palabras reservadas son definidas por el lenguaje COBOL para ser utilizadas en las instrucciones, estas palabras no pueden ser usadas como nombres de dato por el usuario.

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALL
ALPHABETIC
ALSO
ALTER
ALTERNATE
AND
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
BEFORE
BLANK
BLOCK
BOTTOM
BY
CALL
CANCEL
CD
CF
CH
CHARACTER
CHARACTERS
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMUNICATION
COMP
COMPUTATIONAL
COMPUTE
CONFIGURATION
CONTAINS
CONTROL
CONTROLS
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DE
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISPLAY
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC
EGI
ELSE
EMI

ENABLE
END
END-OF-PAGE
ENTER
ENVIRONMENT
EOP
EQUAL
ERROR ESI
EVERY
EXCEPTION
EXIT
EXTEND
FD
FILE
FILE-CONTROL
FILLER
FINAL
FIRST
FOOTING
FOR
FROM
GENERATE
GIVING
GO
GREATER
GROUP
HEADING
HIGH-VALUE
HIGH-VALUES
I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEXED
INDICATE
INITIAL
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS
JUST
JUSTIFIED
KEY
LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES
MEMORY
MERGE
MESSAGE
MODE
MODULES
MOVE
MULTIPLE
MULTIPLY
NATIVE
NEGATIVE
NEXT
NO
NOT
NUMBER
NUMERIC
OBJECT-COMPUTER
OCCURS
OF
OFF
OMMITED
ON
OPEN
OPTIONAL
OR
ORGANIZATION
OUTPUT
OVERFLOW
PAGE
PAGE-COUNTER
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PROCEDURE
PROCEDURES
PROCEED
PROGRAM
PROGRAM-ID
QUEUE
QUOTE
QUOTES
RANDOM
RD
READ
RECEIVE
RECORD
RECORDS
REDEFINES
REEL
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN
SAME

SD	STRING	UNSTRING
SEARCH	SUB-QUEUE-1	UNTIL
SECTION	SUB-QUEUE-2	UP
SECURITY	SUB-QUEUE-3	UPON
SEGMENT	SUBTRACT	USAGE
SEGMENT-LIMITED	SUM	USE
SELECT	SUPPRESS	USING
SEND	SYMBOLIC	VALUE
SENTENCE	SYNC	VALUES
SEPARATE	SYNCHRONIZED	VARYING
SEQUENCE	TABLE	WHEN
SEQUENTIAL	TALLYING	WITH
SET	TAPE	WORDS
SIGN	TERMINAL	WORKINGS-STORAGE
SIZE	TERMINATE	WRITE
SORT	TEXT	ZERO
SORT-MERGE	THAN	ZEROES
SOURCE	THROUGH	ZEROS
SOURCE-COMPUTER	THRU	+
SPACE	TIME	-
SPACES	TIMES	
SPECIAL-NAMES	TO	/
STANDARD	TOP	
STANDARD-1	TRAILING	>
START	TYPE	<
STATUS	UNIT	=
STOP		

APPENDIX F

PROGRAMA

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    HOLA.
       AUTHOR.        CURSO COBOL DECFI.
       INSTALLATION.  DECFI.
       DATE-WRITTEN.  26 MAYO 1987.
       DATE-COMPILED. 26 MAYO 1987.
       SECURITY.      NINGUNA.
      *
      *   Introduccion a COBOL, el programa muestra las divisiones
      * e instrucciones basicas.
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-PC.
       OBJECT-COMPUTER. IBM-PC.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       PROCEDURE DIVISION.
       INICIO.
           DISPLAY 'HOLA AMIGUITOS .... ESTO ES COBOL....'.
           STOP RUN.
```

APPENDIX G

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    SUMA.
 AUTHOR.        CURSO COBOL DECFI.
 INSTALLATION.  DECFI.
 DATE-WRITTEN.  26 MAYO 1987.
 DATE-COMPILED. 26 MAYO 1987.
 SECURITY.      NINGUNA.
*
*    Programa de introduccion a COBOL, realiza la suma de dos
* cantidades introducidas por teclado.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77   DATO-UNO  PICTURE IS 9(02).
 77   DATO-DOS  PICTURE IS 9(02).
 77   SUMA      PICTURE IS 9(03).
 PROCEDURE DIVISION.
 INICIO.
*
*   Captura los sumandos.
*
     DISPLAY 'SUMANDO UNO '.
     ACCEPT DATO-UNO.
     DISPLAY 'SUMANDO DOS '.
     ACCEPT DATO-DOS.
*
*    Realiza la suma.
*
     ADD DATO-UNO DATO-DOS GIVING SUMA.
*
*    Muestra los sumandos y el resultado de la suma.
*
     DISPLAY 'LA SUMA DE ' DATO-UNO ' Y ' DATO-DOS ' ES ' SUMA.
     STOP RUN.
```

APPENDIX H

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    ETIQUETA.
 AUTHOR.        CURSO COBOL DECFI.
 INSTALLATION.  DECFI.
 DATE-WRITTEN.  26 MAYO 1987.
 DATE-COMPILED. 26 MAYO 1987.
 SECURITY.      NINGUNA.
*
*   Captura de los datos para hacer una etiqueta.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77   NOMBRE PICTURE X(32).
 77   CALLE  PICTURE X(32).
 77   CIUDAD PICTURE X(32).
 PROCEDURE DIVISION.
 INICIO.
*
* Captura de datos de la etiqueta.
*
     DISPLAY "NOMBRE DEL SOCIO".
     ACCEPT NOMBRE.
     DISPLAY "CALLE Y COLONIA DONDE VIVE".
     ACCEPT CALLE.
     DISPLAY "CIUDAD, ESTADO Y CODIGO POSTAL".
     ACCEPT CIUDAD.
*
* Forma la etiqueta con los datos capturados.
*
     DISPLAY "+----------------------------------+"
     DISPLAY "!                                  !"
     DISPLAY "! " NOMBRE " !"
     DISPLAY "!                                  !"
     DISPLAY "! " CALLE  " !"
     DISPLAY "!                                  !"
     DISPLAY "! " CIUDAD " !"
```

```
DISPLAY '|                                  |'
DISPLAY '+----------------------------------+'
STOP RUN.
```

APPENDIX I

MOVIMIENTO DE INFORMACION

Caracteristicas en el movimiento de información.

1. La asisnación del identificador-1 (o la literal) al identificador-2, identificador-3, etc. borra el valor de éstos colocando en ellos el valor del identificador-1.

2. El movimiento de cantidades no numericas se realiza de izquierda a derecha.

3. El movimiento de cantidades numericas enteras se realiza de derecha a izquierda.

4. El movimiento de cantidades numericas no-enteras se realiza justificando al punto virtual.

5. Si al realizar los movimientos el campo receptor es menor que el emisor se perderan cifras, de lo contrario se llenara de ceros para los numericos y blancos para los no numericos.

6. Si el destino es numerico editado el fuente no puede ser HIGH-VALUE, LOW-VALUE, SPACE o QUOTE.

7. Si el origen es de tipo ALL y el destino es USAGE solo se respetara una ocurrencia.

8. No se permite que el operando sea una área de índice.

9. Si se involucra un elemento de una tabla primero está es evaluada.

10. Si el destino es no signado solo se movera el valor absoluto.

11. A menos que se especifique JUSTIFY RIGTH el movimiento entre no numericas sera de izquierda a derecha.

Caracteristicas en el movimiento de información en la instrucción MOVE CORRESPONDING.

1. Los valores a mover deberan comprender a nombres de registro o grupo.

2. Se mueven todos los campos dentro del nombre emisor que tengan el mismo nombre-de-dato dentro del nombre receptor tal y como si estuvieramos escrito una serie de movimientos sencillos.

APPENDIX J

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    ETIQUETAS.
 AUTHOR.        CURSO COBOL DECFI.
 INSTALLATION.  DECFI.
 DATE-WRITTEN.  26 MAYO 1987.
 DATE-COMPILED. 26 MAYO 1987.
 SECURITY.      NINGUNA.
*
*   Formacion de etiquetas a partir de datos que estan en
* el archivo SOCIOS.DAT.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
      SELECT CARDS-IN ASSIGN TO DISK.
 DATA DIVISION.
 FILE SECTION.
 FD   CARDS-IN
      LABEL RECORDS ARE STANDARD
      VALUE OF FILE-ID IS "SOCIOS.DAT"
      DATA  RECORD IS CARD.
 01   CARD.
      02  NOMBRE PICTURE X(26).
      02  CALLE  PICTURE X(27).
      02  CIUDAD PICTURE X(27).
 WORKING-STORAGE SECTION.
 01   END-OF-DATA-INDICATOR PICTURE IS X(03).
 PROCEDURE DIVISION.
 MAIN-LOGIC.
*
*    Abrir el archivo para leer los datos que formaran
* la etiqueta.
*
      OPEN INPUT CARDS-IN.
*
*    Se formaran etiquetas hasta el fin de archivo.
*
```

```
           MOVE 'NO' TO END-OF-DATA-INDICATOR.
           PERFORM READ-A-CARD.
           PERFORM READ-PRINT
                   UNTIL END-OF-DATA-INDICATOR IS EQUAL TO 'YES'.
      *
      *    Cerrar el archivo de datos y fin de programa.
      *
           CLOSE CARDS-IN.
           STOP RUN.
       READ-A-CARD.
      *
      *    Lectura del siguiente registro, si ya no existe otro
      * registro la variable END-OF-DATA-INDICATOR tendra el valor
      * de 'YES'.
      *
           READ CARDS-IN RECORD
                AT END MOVE 'YES' TO END-OF-DATA-INDICATOR.
       READ-PRINT.
      *
      *    Forma la etiqueta.
      *
           DISPLAY ' '
           DISPLAY '+------------------------------+'
           DISPLAY '!                              !'
           DISPLAY '! ' NOMBRE '  !'
           DISPLAY '!                              !'
           DISPLAY '! ' CALLE ' !'
           DISPLAY '!                              !'
           DISPLAY '! ' CIUDAD ' !'
           DISPLAY '!                              !'
           DISPLAY '+------------------------------+'
           DISPLAY ' '
           PERFORM READ-A-CARD.
```

APPENDIX K

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     CREA-SOCIOS.
 AUTHOR.         CURSO COBOL DECFI.
 INSTALLATION.   DECFI.
 DATE-WRITTEN.   26 MAYO 1987.
 DATE-COMPILED.  26 MAYO 1987.
 SECURITY.       NINGUNA.
*
*    Captura de datos para crear etiquetas, muestra la
* la etiqueta y graba los datos en un archivo.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
      SELECT CARDS ASSIGN TO DISK.
 DATA DIVISION.
 FILE SECTION.
 FD   CARDS
      LABEL RECORDS ARE STANDARD
      VALUE OF FILE-ID IS "SOCIOS.DAT"
      DATA  RECORD IS CARD.
 01   CARD.
      02  NOMBRE PICTURE X(26).
      02  CALLE  PICTURE X(27).
      02  CIUDAD PICTURE X(27).
 WORKING-STORAGE SECTION.
 01   MAS-INFORMACION PICTURE IS X(02).
 PROCEDURE DIVISION.
 INICIO.
*
*    Abre el archivo donde se graberan los datos.
*
     OPEN OUTPUT CARDS.
*
*   Lee y graba los datos hasta finalizar con la variable
* de control MAS-INFORMACION.
*
```

```
           MOVE 'SI' TO MAS-INFORMACION.
           PERFORM LEE-CONTROL.
           PERFORM LEE-Y-GRABA
                   UNTIL MAS-INFORMACION IS EQUAL TO 'NO'.
      *
      *    Cierra el archivo de datos y fin de programa.
      *
           CLOSE CARDS.
           STOP RUN.
       LEE-CONTROL.
           DISPLAY 'INTRODUCCION DE DATOS (SI, NO) [SI]'
           ACCEPT MAS-INFORMACION.
       LEE-Y-GRABA.
      *
      *    Captura los datos de la etiqueta.
      *
           DISPLAY 'NOMBRE DEL SOCIO'.
           ACCEPT NOMBRE.
           DISPLAY 'CALLE Y COLONIA DONDE VIVE'.
           ACCEPT CALLE.
           DISPLAY 'CIUDAD, ESTADO Y CODIGO POSTAL'.
           ACCEPT CIUDAD.
      *
      *    Muestra la forma de la etiqueta.
      *
           DISPLAY '+------------------------------+'
           DISPLAY '!                              !'
           DISPLAY '! ' NOMBRE '  !'
           DISPLAY '!                              !'
           DISPLAY '! ' CALLE  ' !'
           DISPLAY '!                              !'
           DISPLAY '! ' CIUDAD ' !'
           DISPLAY '!                              !'
           DISPLAY '+------------------------------+'
      *
      *    Graba los datos en el archivo CARDS.
      *
           WRITE CARD.
           PERFORM LEE-CONTROL.
```

APPENDIX L

EJEMPLO DE ARCHIVOS SECUENCIALES

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SECUENCIAL.
AUTHOR.     CECAFI.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  VAX-11-780.
OBJECT-COMPUTER.  VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT ARCHIVO-1 ASSIGN TO 'ENTRADA1.DAT'
          ORGANIZATION IS SEQUENTIAL.
          ACCESS MODE  IS SEQUENTIAL.
   SELECT ARCHIVO-2 ASSIGN TO 'ENTRADA2.DAT'
          ORGANIZATION IS SEQUENTIAL
          ACCESS MODE  IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  ARCHIVO-1
   RECORD CONTAINS 40 CHARACTERS
   DATA RECORD IS REG-1.
01  REG-1.
   03 CARR-1                                PIC 9(02).
   03 NUMCTA-1                              PIC 9(08).
   03 NOMBRE-1                              PIC X(30).
FD  ARCHIVO-2
   RECORD CONTAINS 10 CHARACTERS
   DATA RECORD IS REG-2.
01 REG-2.
   03 CARR-2                                PIC 9(02).
   03 NUMCTA-2                              PIC 9(08).
WORKING-STORAGE SECTION.
77  SUMA                                     PIC 9(03) VALUE ZEROES.
77  HAY-DATOS                                PIC A(02) VALUE 'SI'.
PROCEDURE DIVISION.
INICIO.
   PERFORM ABRE-ARCHIVOS.
   READ ARCHIVO-1 AT END
        DISPLAY 'No tiene registros el archivo 1'
        MOVE 'NO' TO HAY-DATOS.
   PERFORM LECTURA-1 UNTIL HAY-DATOS = 'NO'.
```

```
    DISPLAY 'NUMERO DE REGISTROS DEL ARCHIVO 1 = ' SUMA.
    MOVE ZEROES TO SUMA.
    MOVE 'SI' TO HAY-DATOS.
    READ ARCHIVO-2 AT END
         DISPLAY 'No tiene registros el archivo 2'
         MOVE 'NO' TO HAY-DATOS.
    PERFORM LECTURA-2 UNTIL HAY-DATOS = 'NO'.
    DISPLAY 'NUMERO DE REGISTROS DEL  ARCHIVO 2 = ' SUMA.
    PERFORM CIERRA-ARCHIVOS.
    STOP RUN.
ABRE-ARCHIVOS.
    OPEN INPUT ARCHIVO-1
               ARCHIVO-2.
LECTURA-1.
    ADD 1 TO SUMA.
    READ ARCHIVO-1 AT END
         DISPLAY 'SE ACABO EL ARCHIVO 1'
         MOVE 'ARCHIVO-1' TO ARCHIVO
         MOVE 'NO' TO HAY-DATOS.
LECTURA-2.
    ADD 1 TO SUMA.
    READ ARCHIVO-2 AT END
         DISPLAY 'SE ACABO EL ARCHIVO 2'
         MOVE 'ARCHIVO-2' TO ARCHIVO
         MOVE 'NO' TO HAY-DATOS.
CIERRA-ARCHIVOS.
    CLOSE ARCHIVO-1
          ARCHIVO-2.
```

APPENDIX M

EJEMPLO DE ARCHIVOS SECUENCIALES

```
*-----------------------------------------------------
*                    IDENTIFICATION DIVISION         -
*-----------------------------------------------------
IDENTIFICATION DIVISION.
PROGRAM-ID.          ARCHIVOS-SECUENCIALES.
AUTHOR.              ASA-JMS.
INSTALLATION.        CECAFI.
DATE-WRITTEN.        13-MAR-87.
DATE-COMPILED.       13-MAR-87.
*-----------------------------------------------------
*                    ENVIRONMENT  DIVISION           -
*-----------------------------------------------------
ENVIRONMENT DIVISION.
*---------------------------------------
*       CONFIGURATION  SECTION          -
*---------------------------------------
CONFIGURATION SECTION.
SOURCE-COMPUTER.     VAX-11-780.
OBJECT-COMPUTER.     VAX-11-780.
*---------------------------------------
*       INPUT-OUTPUT SECTION            -
*---------------------------------------
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT    ALUMNOS     ASSIGN  TO  'ALUMNOS.DAT'
                          ORGANIZATION IS SEQUENTIAL
                          ACCESS MODE  IS SEQUENTIAL.
    SELECT    REPORTE    ASSIGN  TO   'REPORTEALUMNOS.LIS'
                          ORGANIZATION IS SEQUENTIAL
                          ACCESS MODE  IS SEQUENTIAL.
*-----------------------------------------------------
*                    DATA  DIVISION                  -
*-----------------------------------------------------
DATA DIVISION.
*---------------------------------------
*       FILE SECTION                    -
*---------------------------------------
FILE SECTION.
FD  ALUMNOS
```

CECAFI

```
    RECORD CONTAINS 45 CHARACTERS
    DATA RECORD IS REG-ALUMNOS.
01  REG-ALUMNOS.
    03 CARR-ALUMNOS                    PIC 9(02).
    03 NUMCTA-ALUMNOS                  PIC 9(08).
    03 NOMBRE-ALUMNOS                  PIC X(30).
    03 PROM-ALUMNOS                    PIC 9(02)V9(02).
FD  REPORTE
    RECORD CONTAINS 132 CHARACTERS
    DATA RECORD IS REG-REPORTE.
01  REG-REPORTE                        PIC X(132).
*-----------------------------------------
*       WORKING-STORAGE SECTION
*-----------------------------------------
WORKING-STORAGE SECTION.
01  ENCABEZADO.
    03 FILLER                      PIC X(15) VALUE SPACES.
    03 FILLER                      PIC X(08) VALUE 'CARRERA:'.
    03 FILLER                      PIC X(13) VALUE SPACES.
    03 FILLER                      PIC X(10) VALUE 'NUM. CTA.:'.
    03 FILLER                      PIC X(27) VALUE SPACES.
    03 FILLER                      PIC X(07) VALUE 'NOMBRE:'.
    03 FILLER                      PIC X(27) VALUE SPACES.
    03 FILLER                      PIC X(09) VALUE 'PROMEDIO:'.
01  REGISTRO-DETALLADO.
    03 FILLER                      PIC X(18) VALUE SPACES.
    03 CARR-DET                    PIC X(02).
    03 FILLER                      PIC X(17) VALUE SPACES.
    03 NUMCTA-DET                  PIC X(08).
    03 FILLER                      PIC X(17) VALUE SPACES.
    03 NOMBRE-DET                  PIC X(30).
    03 FILLER                      PIC X(17) VALUE SPACES.
    03 PROM-DET                    PIC 9(02).9(02).
77  RAYITAS                        PIC X(132) VALUE ALL '-'.
77  FIN-DE-ARCHIVO                 PIC A(02)  VALUE 'NO'.
*-------------------------------------------------------------
*                    PROCEDURE DIVISION
*-------------------------------------------------------------
PROCEDURE DIVISION.
INICIA.
    PERFORM PARRAFO-INICIAL.
    PERFORM PARRAFO-PRINCIPAL
            UNTIL (FIN-DE-ARCHIVO EQUAL TO 'SI').
    PERFORM PARRAFO-FINAL.
    STOP RUN.
PARRAFO-INICIAL.
    OPEN INPUT  ALUMNOS
         OUTPUT REPORTE.
    WRITE REG-REPORTE FROM ENCABEZADO.
    WRITE REG-REPORTE FROM RAYITAS.
    PERFORM LEE-ARCHIVO.
PARRAFO-PRINCIPAL.
    MOVE CARR-ALUMNOS TO CARR-DET.
    MOVE NUMCTA-ALUMNOS TO NUMCTA-DET.
    MOVE NOMBRE-ALUMNOS TO NOMBRE-DET.
    MOVE PROM-ALUMNOS TO PROM-DET.
```

```
    WRITE REG-REPORTE FROM REGISTRO-DETALLADO.
    PERFORM LEE-ARCHIVO.
PARRAFO-FINAL.
    CLOSE  ALUMNOS
           REPORTE.
LEE-ARCHIVO.
    READ  ALUMNOS
          AT END  DISPLAY 'FIN DEL ARCHIVO ALUMNOS.DAT'
                  MOVE 'SI' TO FIN-DE-ARCHIVO.
```

APPENDIX N

PROGRAMA

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    VALIDA-INFORMACION.
       AUTHOR.        CURSO COBOL DECFI.
       INSTALLATION.  DECFI.
       DATE-WRITTEN.  26 MAYO 1987.
       DATE-COMPILED. 26 MAYO 1987.
       SECURITY.      NINGUNA.
      *
      *    Creacion de un archivo de datos con referencia a los nombres
      * del archivo SOCIOS.DAT; esta informacion es validada por tipo
      * y ranso de valores.
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-PC.
       OBJECT-COMPUTER. IBM-PC.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT CARDS-IN  ASSIGN TO DISK.
            SELECT CARDS-OUT ASSIGN TO DISK.
       DATA DIVISION.
       FILE SECTION.
       FD   CARDS-IN
            LABEL RECORDS ARE STANDARD
            VALUE OF FILE-ID IS 'SOCIOS.DAT'
            DATA  RECORD IS CARD-IN.
       01   CARD-IN.
            02  NOMBRE PICTURE X(26).
            02  FILLER PICTURE X(54).
       FD   CARDS-OUT
            LABEL RECORD ARE STANDARD
            VALUE OF FILE-ID IS 'CUOTAS.DAT'.
            DATA RECORD IS CARD-OUT.
       01   CARD-OUT.
            02 NOMBRE-OUT     PICTURE X(26).
            02 SEXO-OUT       PICTURE X(01).
            02 TELEFONO-OUT   PICTURE X(10).
            02 RFC-OUT.
               03 RFC-L-OUT   PICTURE X(04).
               03 RFC-N-OUT   PICTURE 9(06).
```

```
           02 TIPO-SOCIO-OUT PICTURE 9(01).
           02 CUOTA-OUT      PICTURE 9(06).
           02 FILLER         PICTURE X(26).
       WORKING-STORAGE SECTION.
       77   END-OF-DATA-INDICATOR PICTURE IS X(03).
       77   OK-INFO-DEF          PICTURE IS X(02) VALUE 'NO'.
            88 OK-INFO  VALUE 'OK'.
       01   CARD-KBR.
            02 SEXO             PICTURE X(01).
               88 SEXO-OK VALUE 'F' 'f' 'M' 'm'.
            02 TELEFONO         PICTURE X(10).
            02 RFC.
               03 RFC-LETRAS    PICTURE X(04).
               03 RFC-NUMEROS   PICTURE X(06).
            02 TIPO-SOCIO-X     PICTURE X(01).
            02 TIPO-SOCIO-N     PICTURE 9(01).
               88 TIPO-SOCIO-OK VALUE 1 THRU 5.
            02 CUOTA-X          PICTURE X(06).
            02 CUOTA-N          PICTURE 9(06).
               88 CUOTA-OK VALUE 10000 THRU 500000.
       PROCEDURE DIVISION.
       MAIN-LOGIC.
      *
      *    Abrir el archivo para leer los nombres y otro archivo donde se
      * pondra la informacion capturada.
      *
           OPEN INPUT  CARDS-IN.
           OPEN OUTPUT CARDS-OUT.
      *
      *    Se pediran datos hasta el fin de archivo.
      *
           MOVE 'NO' TO END-OF-DATA-INDICATOR.
           PERFORM READ-A-CARD.
           PERFORM READ-ACCEPT-WRITE
                   UNTIL END-OF-DATA-INDICATOR IS EQUAL TO 'YES'.
      *
      *    Cerrar los archivos de datos y fin de programa.
      *
           CLOSE CARDS-IN.
           CLOSE CARDS-OUT.
           STOP RUN.
       READ-A-CARD.
      *
      *    Lectura del siguiente registro, si ya no existe otro
      * registro la variable END-OF-DATA-INDICATOR tendra el valor
      * de 'YES'.
      *
           READ CARDS-IN RECORD
                AT END MOVE 'YES' TO END-OF-DATA-INDICATOR.
       READ-ACCEPT-WRITE.
      *
      *    Muestra el nombre, captura la informacion, la valida y
      * graba en un archivo de salida.
      *
           DISPLAY ' '
           DISPLAY '+---------------------------+'
```

```
           DISPLAY '!                              !'
           DISPLAY '! ' NOMBRE '  !'
           DISPLAY '!                              !'
           DISPLAY '+------------------------------+'
           DISPLAY ' '
           PERFORM ACCEPT-AND-CHECK-INFORMATION.
           WRITE CARD-OUT.
           PERFORM READ-A-CARD.
       ACCEPT-AND-CHECK-INFORMATION.
      *
      *    Entrada de informacion y su validacion.
      *
           MOVE 'NO' TO OK-INFO-DEF.
           PERFORM ACCEPT-AND-CHECK-SEXO UNTIL OK-INFO.
           MOVE 'NO' TO OK-INFO-DEF.
           PERFORM ACCEPT-AND-CHECK-RFC UNTIL OK-INFO.
           MOVE 'NO' TO OK-INFO-DEF.
           PERFORM ACCEPT-AND-CHECK-TELEFONO UNTIL OK-INFO.
           MOVE 'NO' TO OK-INFO-DEF.
           PERFORM ACCEPT-AND-CHECK-TIPO-SOCIO UNTIL OK-INFO.
           MOVE 'NO' TO OK-INFO-DEF.
           PERFORM ACCEPT-AND-CHECK-CUOTA UNTIL OK-INFO.
      *
      *    Arma el registro a grabar.
      *
           MOVE NOMBRE       TO NOMBRE-OUT.
           MOVE SEXO         TO SEXO-OUT.
           MOVE RFC          TO RFC-OUT.
           MOVE TELEFONO     TO TELEFONO-OUT.
           MOVE TIPO-SOCIO-N TO TIPO-SOCIO-OUT.
           MOVE CUOTA-N      TO CUOTA-OUT.
       ACCEPT-AND-CHECK-SEXO.
           DISPLAY ' '.
           DISPLAY 'CLAVE SEXO'.
           ACCEPT SEXO.
           IF SEXO NOT ALPHABETIC
              DISPLAY 'ERROR EN TIPO DE DATO EN LA CLAVE DE SEXO.'
           ELSE
              IF NOT SEXO-OK
                 DISPLAY 'ERROR EN VALOR DE DATO, '
                         'DEBE SER F, f, M o m.'
              ELSE
                 MOVE 'OK' TO OK-INFO-DEF.
       ACCEPT-AND-CHECK-RFC.
           DISPLAY ' '.
           DISPLAY 'VALOR DEL RFC'.
           ACCEPT RFC.
           IF RFC-LETRAS NOT ALPHABETIC
              DISPLAY 'ERROR EN LA PARTE DE LETRAS DEL RFC.'
           ELSE
              IF RFC-NUMEROS NOT NUMERIC
                 DISPLAY 'ERROR EN LA PARTE NUMERICA DEL RFC.'
              ELSE
                 MOVE 'OK' TO OK-INFO-DEF.
       ACCEPT-AND-CHECK-TELEFONO.
           DISPLAY ' '.
```

```
           DISPLAY 'NUMERO TELEFONICO'.
           ACCEPT TELEFONO.
           IF TELEFONO NOT NUMERIC
              DISPLAY 'EL NUMERO TELEFONICO '
                      'DEBE CONTENER SOLO NUMEROS.'
           ELSE
              MOVE 'OK' TO OK-INFO-DEF.
       ACCEPT-AND-CHECK-TIPO-SOCIO.
           DISPLAY ' '.
           DISPLAY 'TIPO DE SOCIO'.
           ACCEPT TIPO-SOCIO-X
           IF TIPO-SOCIO-X NOT NUMERIC
              DISPLAY 'EL TIPO DE SOCIO DEBE SER NUMERICO.'
           ELSE
              MOVE TIPO-SOCIO-X TO TIPO-SOCIO-N
              IF NOT TIPO-SOCIO-OK
                 DISPLAY 'TIPO DE SOCIO FUERA DE RANGO, '
                         'CONSULTAR TABLA DE VALORES.'
              ELSE
                 MOVE 'OK' TO OK-INFO-DEF.
       ACCEPT-AND-CHECK-CUOTA.
           DISPLAY ' '.
           DISPLAY 'CUOTA DEL SOCIO'.
           ACCEPT CUOTA-X.
           IF CUOTA-X IS NOT NUMERIC
              DISPLAY 'LA CUOTA DEBE TENER '
                      'SOLO CARACTERES NUMERICOS.'
           ELSE
              MOVE CUOTA-X TO CUOTA-N
              IF CUOTA-N IS ZERO
                 DISPLAY 'LA CUOTA DEBE SER MAYOR A ZERO.'
              ELSE
                 IF NOT CUOTA-OK
                    DISPLAY 'LA CUOTA ESTA FUERA DE RANGO, '
                            'CONSULTAR TABLA DE CUOTAS.'
                 ELSE
                    MOVE 'OK' TO OK-INFO-DEF.
```

APPENDIX O

DEFINICION DE TIPO DE DATOS

Tipo de carácter	Símbolo	Uso
Definición de tipo de campo	9	Campo numérico
	A	Campo alfabético
	X	Campo alfanumérico
Definición de tipo numérico especial	V	Punto virtual
	P	Ajuste potencial
	S	Inclusión de signo
Caracteres de edición	$	Signo dollar
	Z	Supresión de ceros
	*	Protección de campos
	.	Ajuste e inserción de punto.
	,	Inserción de coma
	0	Inserción de ceros
	/	Inserción de slash
	+	Signo + evaluado
	-	Signo - evaluado
	DB	Debe evaluado
	CR	Crédito evaluado
	B	Inserción de blanco

Evaluación de caracteres de edición.

carácter de edición	Valor positivo	Valor negativo
+	+	-
-		-
db		db
cr		cr

Tipo de datos en **COBOL**.

Numéricos	Este tipo de datos hace combinación de los caracteres 9 V P S
Alfabéticos	Descrito sólo por el carácter A
Alfanuméricos	Descrito sólo por el carácter X
Numéricos editados	Hace uso de los caracteres 9 V P Z B / 0 , . + - CR DB $ *
Alfabéticos editados	Hace uso de los caracteres A B
Alfanuméricos editados	Hace uso de los caracteres A X 9 B 0 /

APPENDIX P

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    ETIQUETAS.
 AUTHOR.        CURSO COBOL DECFI.
 INSTALLATION.  DECFI.
 DATE-WRITTEN.  26 MAYO 1987.
 DATE-COMPILED. 26 MAYO 1987.
 SECURITY.      NINGUNA.
*
*   Formacion de etiquetas en un archivo a
* partir de datos del archivo SOCIOS.DAT.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
      SELECT CARDS-IN ASSIGN TO DISK.
      SELECT IMPRESORA ASSIGN TO PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD   CARDS-IN
      LABEL RECORDS ARE STANDARD
      VALUE OF FILE-ID IS 'SOCIOS.DAT'
      DATA  RECORD IS CARD.
 01   CARD.
      02  NOMBRE PICTURE X(26).
      02  CALLE  PICTURE X(27).
      02  CIUDAD PICTURE X(27).
 FD   IMPRESORA
      LABEL RECORDS ARE OMITTED
      DATA RECORD IS LINEA.
 01   LINEA PICTURE IS X(80).
 WORKING-STORAGE SECTION.
 01   END-OF-DATA-INDICATOR PICTURE IS X(03).
 01   MARCO-1 PICTURE X(31) VALUE
      '+-----------------------------+'.
 01   MARCO-2 PICTURE X(31) VALUE
      '!                             !'.
 01   NOMBRE-S.
```

```
           03 FILLER PICTURE X(02) VALUE '! '.
           03 NOMBRE-SAL    PIC X(26).
           03 FILLER PICTURE X(03) VALUE '  !'.
       01  CALLE-S.
           03 FILLER PICTURE X(02) VALUE '! '.
           03 CALLE-SAL     PIC X(27).
           03 FILLER PICTURE X(02) VALUE ' !'.
       01  CIUDAD-S.
           03 FILLER PICTURE X(02) VALUE '! '.
           03 CIUDAD-SAL    PIC X(27).
           03 FILLER PICTURE X(02) VALUE ' !'.
       PROCEDURE DIVISION.
       MAIN-LOGIC.
      *
      *    Abrir el archivo para leer los datos que formaran
      * la etiqueta y el archivo donde se guardaran las etiquetas.
      *
           OPEN INPUT CARDS-IN.
           OPEN OUTPUT IMPRESORA.
      *
      *    Se formaran etiquetas hasta el fin de archivo.
      *
           MOVE 'NO' TO END-OF-DATA-INDICATOR.
           PERFORM READ-A-CARD.
           PERFORM READ-PRINT
                   UNTIL END-OF-DATA-INDICATOR IS EQUAL TO 'YES'.
      *
      *    Cerrar el archivo de datos e impresion y fin de programa.
      *
           CLOSE CARDS-IN.
           CLOSE IMPRESORA.
           STOP RUN.
       READ-A-CARD.
      *
      *    Lectura del siguiente registro, si ya no existe otro
      * registro la variable END-OF-DATA-INDICATOR tendra el valor
      * de 'YES'.
      *
           READ CARDS-IN RECORD
                AT END MOVE 'YES' TO END-OF-DATA-INDICATOR.
       READ-PRINT.
      *
      *    Forma la etiqueta y se graba en el archivo de etiquetas.
      *
           MOVE SPACE TO LINEA.
           WRITE LINEA AFTER ADVANCING 2 LINE.
           WRITE LINEA FROM MARCO-1 AFTER ADVANCING 1 LINE.
           WRITE LINEA FROM MARCO-2 AFTER ADVANCING 1 LINE.
           MOVE NOMBRE TO NOMBRE-SAL.
           WRITE LINEA FROM NOMBRE-S AFTER ADVANCING 1 LINE.
           WRITE LINEA FROM MARCO-2 AFTER ADVANCING 1 LINE.
           MOVE CALLE TO CALLE-SAL.
           WRITE LINEA FROM CALLE-S AFTER ADVANCING 1 LINE.
           WRITE LINEA FROM MARCO-2 AFTER ADVANCING 1 LINE.
           MOVE CIUDAD TO CIUDAD-SAL.
           WRITE LINEA FROM CIUDAD-S AFTER ADVANCING 1 LINE.
```

```
           WRITE LINEA FROM MARCO-2 AFTER ADVANCING 1 LINE.
           WRITE LINEA FROM MARCO-1 AFTER ADVANCING 1 LINE.
           PERFORM READ-A-CARD.
```

APPENDIX D

PROGRAMA

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    REPORTE.
       AUTHOR.        CURSO COBOL DECFI.
       INSTALLATION.  DECFI.
       DATE-WRITTEN.  26 MAYO 1987.
       DATE-COMPILED. 26 MAYO 1987.
       SECURITY.      NINGUNA.
      *
      *    Emite un reporte de las cuotas de los socios.
      *
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-PC.
       OBJECT-COMPUTER. IBM-PC.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT CARDS     ASSIGN TO DISK.
            SELECT IMPRESORA ASSIGN TO PRINTER.
       DATA DIVISION.
       FILE SECTION.
       FD   CARDS
            LABEL RECORD ARE STANDARD
            VALUE OF FILE-ID IS 'CUOTAS.DAT'
            DATA RECORD IS CARD.
       01   CARD.
            02 NOMBRE          PICTURE X(26).
            02 SEXO            PICTURE X(01).
            02 TELEFONO        PICTURE X(10).
            02 RFC.
               03 RFC-L        PICTURE X(04).
               03 RFC-N        PICTURE 9(06).
            02 TIPO-SOCIO      PICTURE 9(01).
               88 SOCIO-1      VALUE 1.
               88 SOCIO-2      VALUE 2.
               88 SOCIO-3      VALUE 3.
               88 SOCIO-4      VALUE 4.
               88 SOCIO-5      VALUE 5.
            02 CUOTA           PICTURE 9(06).
            02 FILLER          PICTURE X(26).
       FD   IMPRESORA
```

```
       LABEL RECORDS ARE OMITTED
       DATA RECORD IS PRINT-LINE.

   01  PRINT-LINE PICTURE IS X(80).
   WORKING-STORAGE SECTION.
   77  END-OF-DATA-INDICATOR PICTURE IS X(03).
   77  HOJA                  PICTURE IS 9(04) VALUE ZEROES.
   77  LINEA                 PICTURE IS 9(02) VALUE ZEROES.
   77  FECHA                 PICTURE IS 9(06) VALUE ZEROES.
   77  SUM-CUOTA             PICTURE IS 9(09) VALUE ZEROES.
   77  SUM-PAGO              PICTURE IS 9(09) VALUE ZEROES.
   77  SUM-COBRO             PICTURE IS 9(09) VALUE ZEROES.
   77  CUOTA-SOCIO           PICTURE IS 9(06) VALUE ZEROES.
   77  PAGO                  PICTURE IS 9(06) VALUE ZEROES.
   77  COBRO                 PICTURE IS 9(06) VALUE ZEROES.
   01  HORA.
       03 HH PICTURE IS 9(02).
       03 MM PICTURE IS 9(02).
       03 SS PICTURE IS 9(02).
       03 FF PICTURE IS 9(02).
   01  ENCABEZADO-1.
       02 FILLER          PICTURE X(07) VALUE 'FECHA: '.
       02 FECHA-SAL       PICTURE 9(02)/9(02)/9(02).
       02 FILLER          PICTURE X(14) VALUE SPACES.
       02 FILLER          PICTURE X(22) VALUE
          'FACULTAD DE INGENIERIA'.
       02 FILLER          PICTURE X(18) VALUE SPACES.
       02 FILLER          PICTURE X(05) VALUE 'HOJA:'.
       02 HOJA-SAL        PICTURE Z(04).
       02 FILLER          PICTURE X(01) VALUE SPACE.
   01  ENCABEZADO-2.
       02 FILLER          PICTURE X(07) VALUE 'HORA : '.
       02 HORA-SAL.
          03 H-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE ':'.
          03 M-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE ':'.
          03 S-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE '.'.
          03 F-S          PICTURE 9(02).
       02 FILLER          PICTURE X(06) VALUE SPACE.
       02 FILLER          PICTURE X(30) VALUE
          'DIVISION DE EDUCACION CONTINUA'.
       02 FILLER          PICTURE X(24) VALUE SPACES.
   01  ENCABEZADO-3.
       02 FILLER          PICTURE X(28) VALUE SPACES.
       02 FILLER          PICTURE X(18) VALUE
          'RELACION DE CUOTAS'.
       02 FILLER          PICTURE X(34) VALUE SPACES.
   01  ENCABEZADO-4.
       02 FILLER          PICTURE X(12) VALUE SPACES.
       02 FILLER          PICTURE X(06) VALUE 'NOMBRE'.
       02 FILLER          PICTURE X(12) VALUE SPACES.
       02 FILLER          PICTURE X(02) VALUE 'TS'.
       02 FILLER          PICTURE X(06) VALUE SPACES.
       02 FILLER          PICTURE X(06) VALUE 'CUOTA'.
```

```
           02 FILLER          PICTURE X(11) VALUE SPACES.
           02 FILLER          PICTURE X(04) VALUE 'PAGO'.
           02 FILLER          PICTURE X(10) VALUE SPACES.
           02 FILLER          PICTURE X(05) VALUE 'COBRO'.
           02 FILLER          PICTURE X(05) VALUE SPACES.
       01  ENCABEZADO-5.
           02 FILLER          PICTURE X(02) VALUE SPACES.
           02 FILLER          PICTURE X(26) VALUE ALL '='.
           02 FILLER          PICTURE X(02) VALUE SPACES.
           02 FILLER          PICTURE X(02) VALUE ALL '='.
           02 FILLER          PICTURE X(03) VALUE SPACES.
           02 FILLER          PICTURE X(11) VALUE ALL '='.
           02 FILLER          PICTURE X(04) VALUE SPACES.
           02 FILLER          PICTURE X(11) VALUE ALL '='.
           02 FILLER          PICTURE X(04) VALUE SPACES.
           02 FILLER          PICTURE X(11) VALUE ALL '='.
           02 FILLER          PICTURE X(04) VALUE SPACES.

       01  DETALLE.
           02 FILLER          PICTURE X(02) VALUE SPACES.
           02 NOMBRE-SAL      PICTURE X(26).
           02 FILLER          PICTURE X(03) VALUE SPACES.
           02 TIPO-SOCIO-SAL  PICTURE 9(01).
           02 FILLER          PICTURE X(03) VALUE SPACES.
           02 CUOTA-SAL       PICTURE $Z(03),Z(03).Z(02).
           02 FILLER          PICTURE X(04) VALUE SPACES.
           02 PAGO-SAL        PICTURE $Z(03),Z(03).Z(02).
           02 FILLER          PICTURE X(04) VALUE SPACES.
           02 COBRO-SAL       PICTURE $Z(03),Z(03).Z(02).
           02 FILLER          PICTURE X(04) VALUE SPACES.
       01  SUM-CUOTA-SAL.
           02 FILLER          PICTURE X(20) VALUE
              'TOTAL DE CUOTAS   : '.
           02 SUM-CUOTA-S     PICTURE $Z(03),Z(03),Z(03).00.
           02 FILLER          PICTURE X(45) VALUE SPACES.
       01  SUM-PAGO-SAL.
           02 FILLER          PICTURE X(20) VALUE
              'COBROS EFECTUADOS : '.
           02 SUM-PAGO-S      PICTURE $Z(03),Z(03),Z(03).00.
           02 FILLER          PICTURE X(45) VALUE SPACES.

       01  SUM-COBRO-SAL.
           02 FILLER          PICTURE X(20) VALUE
              'CARTERA A COBRAR  : '.
           02 SUM-COBRO-S     PICTURE $Z(03),Z(03),Z(03).00.
           02 FILLER          PICTURE X(45) VALUE SPACES.
       PROCEDURE DIVISION.
       MAIN-LOGIC.
      *
      *    Abrir el archivo de cuotas y el de impresion.
      *

           OPEN INPUT  CARDS.
           OPEN OUTPUT IMPRESORA.
      *
      *    Se lee la fecha y hora de reporte del sistema.
```

```
*
     ACCEPT FECHA FROM DATE.
     ACCEPT HORA  FROM TIME.
*
*    Se evalua los datos de fecha del encabezado 1
* y hora del encabezado 2.
*
     MOVE FECHA TO FECHA-SAL.
     MOVE HH TO H-S.
     MOVE MM TO M-S.
     MOVE SS TO S-S.
     MOVE FF TO F-S.
*
*    Se imprime el primer encabezado y subencabezado.
*

     PERFORM IMPRIME-ENCABEZADO.
     PERFORM IMPRIME-SUBENCABEZADO.
*
*    Se imprimiran los registros detalle llevando un
* control de hoja.
*
     MOVE 'NO' TO END-OF-DATA-INDICATOR.
     PERFORM READ-A-CARD.
     PERFORM READ-PRINT
             UNTIL END-OF-DATA-INDICATOR IS EQUAL TO 'YES'.
*
*    Indica las sumatorias con respecto al estado de
* cuotas.
     PERFORM RESULTADOS-CUOTAS.
*
*    Cerrar los archivos de datos y fin de programa.
*
     CLOSE CARDS.
     CLOSE IMPRESORA.
     STOP RUN.
 READ-A-CARD.
*
*    Lectura del siguiente registro, si ya no existe otro
* registro la variable END-OF-DATA-INDICATOR tendra el valor
* de 'YES'.
*
     READ CARDS RECORD
          AT END MOVE 'YES' TO END-OF-DATA-INDICATOR.
 IMPRIME-ENCABEZADO.
*
*    Impresion de los registros de encabezado para cada
* pagina.
*
     ADD 1 TO HOJA.
     MOVE  HOJA TO HOJA-SAL.
     WRITE PRINT-LINE FROM ENCABEZADO-1 AFTER ADVANCING PAGE.
     WRITE PRINT-LINE FROM ENCABEZADO-2 AFTER ADVANCING 1 LINE.
     WRITE PRINT-LINE FROM ENCABEZADO-3 AFTER ADVANCING 1 LINE.
     MOVE 4 TO LINEA.
 IMPRIME-SUBENCABEZADO.
```

```
      *
      *    Impresion de las lineas de subencabezado.
      *
           WRITE PRINT-LINE FROM ENCABEZADO-4 AFTER ADVANCING 2 LINE.
           WRITE PRINT-LINE FROM ENCABEZADO-5 AFTER ADVANCING 1 LINE.
           ADD 3 TO LINEA.
       READ-PRINT.
      *
      *    Imprime el reporte requerido.
      *
      *
      *    Control de pagina.
      *
           IF LINEA IS GREATER 60
              PERFORM IMPRIME-ENCABEZADO
              PERFORM IMPRIME-SUBENCABEZADO.
      *
      *    Definicion de cuotas, pagos y cobros.
      *
           MOVE ZEROES TO CUOTA-SOCIO, PAGO, COBRO.
           IF SOCIO-1
             MOVE 200000 TO CUOTA-SOCIO
           ELSE
             IF SOCIO-2
                MOVE 300000 TO CUOTA-SOCIO
             ELSE
                IF SOCIO-3
                   MOVE 400000 TO CUOTA-SOCIO
                ELSE
                   IF SOCIO-4
                      MOVE 450000 TO CUOTA-SOCIO
                   ELSE
                      MOVE 500000 TO CUOTA-SOCIO.
           MOVE CUOTA TO PAGO.
           SUBTRACT PAGO FROM CUOTA-SOCIO GIVING COBRO.
           ADD CUOTA-SOCIO TO SUM-CUOTA.
           ADD PAGO TO SUM-PAGO.
           ADD COBRO TO SUM-COBRO.
      *
      *   Definicion del registro de detalle y su impresion.
      *
           MOVE NOMBRE TO NOMBRE-SAL.
           MOVE TIPO-SOCIO TO TIPO-SOCIO-SAL.
           MOVE CUOTA-SOCIO TO CUOTA-SAL.
           MOVE PAGO TO PAGO-SAL.
           MOVE COBRO TO COBRO-SAL.
           WRITE PRINT-LINE FROM DETALLE AFTER ADVANCING 1 LINE.
           ADD 1 TO LINEA.
           PERFORM READ-A-CARD.
       RESULTADOS-CUOTAS.
      *
      *    Control de pagina.
      *
           IF LINEA IS GREATER 60
              PERFORM IMPRIME-ENCABEZADO.
      *
```

```
*    Impresion de resultados.
*
     MOVE SUM-CUOTA TO SUM-CUOTA-S.
     MOVE SUM-PAGO  TO SUM-PAGO-S.
     MOVE SUM-COBRO TO SUM-COBRO-S.
     WRITE PRINT-LINE FROM SUM-CUOTA-SAL AFTER ADVANCING 4 LINES.
     WRITE PRINT-LINE FROM SUM-PAGO-SAL  AFTER ADVANCING 1 LINE.
     WRITE PRINT-LINE FROM SUM-COBRO-SAL AFTER ADVANCING 1 LINE.
```

APPENDIX R

USO DE LA CLAUSULA REDEFINES

La clausula REDEFINES permite definir la misma área de almacenamiento con diferentes nombres y tipo de datos.

1. No se pueden redefinir áreas a nivel 01 de la FILE SECTION.

2. La redefinición de registros se lleva a cabo con la multidefinición de registros denotada por la clausula DATA RECORDS.

3. Se pueden redefinir todo o parte de un registro en la WORKING-STORAGE SECTION.

4. El área a redefinir debe declararse en la definición de dato anterior a la redefinición.

5. El área a redefinir y el dato que define deben estar al mismo nivel y tener la misma longitud.

6. Se puede redefinir mas de una vez el área. No se puede redefinir datos con nivel 66 u 88.

7. Si se redefine más de una vez una área esta será referenciada siempre al mismo nombre y de forma seguida.

8. La clausula VALUE solo es permitida en el campo que originalmente se redefine.

9. Un registro 01 que emplea la clausula OCCURS no debe contener la clausula REDEFINES.

10. Se pueden redefinir niveles 88 en los datos redefinidos.

APPENDIX S

USO DE LA CLAUSULA OCCURS

La clausula OCCURS permite definir tablas. Siguiendo las siguientes reglas:

1. Los indices deben ser números enteros positivos.

2. El primer indice debe ser menor al segundo indice.

3. Los indices llave deben contener valores entre el indice uno y el indice dos.

4. Los campos que se usen como indices deben contener valores entre en indice uno y el indice dos. Estos indices son encerrados entre parentesis despues del nombre y definen el elemento de la tabla.

5. La clausula OCCURS no debe emplearse en campos que tengan los numeros de nivel 77, 66 y 88.

6. La clausula OCCURS puede emplearse en cualquier sección de la data division.

7. Los indices no deben emplearse al hacer uso del verbo SEARCH.

8. Cuando se definen tablas de tamaño variable el tamaño final debe estar entre los limites de la tabla.

9. La opción KEY indica que la tabla se encuentra ordenada del modo especificada.

10. La clausula INDEXED BY sirve para asociarle uno o mas indices a la tabla.

11. Estas dos últimas clausulas son requereidas al usarse el verbo SEARCH.

12. Los nombres de dato llave son campos nombrados en la definición de la tabla.

13. Cuando se usar varios campos llave estos se describen por orden de prioridad de ordenamiento de la tabla.

APPENDIX T

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    TABLAS.
 AUTHOR.        CURSO COBOL DECFI.
 INSTALLATION.  DECFI.
 DATE-WRITTEN.  26 MAYO 1987.
 DATE-COMPILED. 26 MAYO 1987.
 SECURITY.      NINGUNA.
*
*    Emite un reporte de las cuotas de los socios.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT CARDS     ASSIGN TO DISK.
     SELECT IMPRESORA ASSIGN TO PRINTER.
 DATA DIVISION.
 FILE SECTION.
 FD   CARDS
      LABEL RECORD ARE STANDARD
      VALUE OF FILE-ID IS 'CUOTAS.DAT'
      DATA RECORD IS CARD.
 01   CARD.
      02 NOMBRE          PICTURE X(26).
      02 SEXO            PICTURE X(01).
      02 TELEFONO        PICTURE X(10).
      02 RFC.
         03 RFC-L        PICTURE X(04).
         03 RFC-N        PICTURE 9(06).
      02 TIPO-SOCIO      PICTURE 9(01).
         88 SOCIO-1      VALUE 1.
         88 SOCIO-2      VALUE 2.
         88 SOCIO-3      VALUE 3.
         88 SOCIO-4      VALUE 4.
         88 SOCIO-5      VALUE 5.
      02 CUOTA           PICTURE 9(06).
      02 FILLER          PICTURE X(26).
 FD   IMPRESORA
```

```
       LABEL RECORDS ARE OMITTED
       DATA RECORD IS PRINT-LINE.

   01  PRINT-LINE PICTURE IS X(80).
   WORKING-STORAGE SECTION.
   77  END-OF-DATA-INDICATOR PICTURE IS X(03).
   77  HOJA                  PICTURE IS 9(04) VALUE ZEROES.
   77  LINEA                 PICTURE IS 9(02) VALUE ZEROES.
   77  FECHA                 PICTURE IS 9(06) VALUE ZEROES.
   01  HORA.
       03 HH PICTURE IS 9(02).
       03 MM PICTURE IS 9(02).
       03 SS PICTURE IS 9(02).
       03 FF PICTURE IS 9(02).
   01  ENCABEZADO-1.
       02 FILLER          PICTURE X(07) VALUE 'FECHA: '.
       02 FECHA-SAL       PICTURE 9(02)/9(02)/9(02).
       02 FILLER          PICTURE X(14) VALUE SPACES.
       02 FILLER          PICTURE X(22) VALUE
          'FACULTAD DE INGENIERIA'.
       02 FILLER          PICTURE X(18) VALUE SPACES.
       02 FILLER          PICTURE X(05) VALUE 'HOJA:'.
       02 HOJA-SAL        PICTURE Z(04).
       02 FILLER          PICTURE X(01) VALUE SPACE.
   01  ENCABEZADO-2.
       02 FILLER          PICTURE X(07) VALUE 'HORA : '.
       02 HORA-SAL.
          03 H-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE ':'.
          03 M-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE ':'.
          03 S-S          PICTURE 9(02).
          03 FILLER       PICTURE X(01) VALUE '.'.
          03 F-S          PICTURE 9(02).
       02 FILLER          PICTURE X(06) VALUE SPACE.
       02 FILLER          PICTURE X(30) VALUE
          'DIVISION DE EDUCACION CONTINUA'.
       02 FILLER          PICTURE X(24) VALUE SPACES.
   01  ENCABEZADO-3.
       02 FILLER          PICTURE X(28) VALUE SPACES.
       02 FILLER          PICTURE X(20) VALUE
          'DIRECTORIO DE SOCIOS'.
       02 FILLER          PICTURE X(32) VALUE SPACES.
   01  ENCABEZADO-4.
       02 FILLER          PICTURE X(22) VALUE SPACES.
       02 FILLER          PICTURE X(06) VALUE 'NOMBRE'.
       02 FILLER          PICTURE X(12) VALUE SPACES.
       02 FILLER          PICTURE X(10) VALUE 'TIPO SOCIO'.
       02 FILLER          PICTURE X(06) VALUE SPACES.
       02 FILLER          PICTURE X(06) VALUE 'CUOTA'.
       02 FILLER          PICTURE X(19) VALUE SPACES.
   01  ENCABEZADO-5.
       02 FILLER          PICTURE X(12) VALUE SPACES.
       02 FILLER          PICTURE X(26) VALUE ALL '='.
       02 FILLER          PICTURE X(02) VALUE SPACES.
       02 FILLER          PICTURE X(10) VALUE ALL '='.
```

```
           02 FILLER          PICTURE X(03) VALUE SPACES.
           02 FILLER          PICTURE X(11) VALUE ALL '='.
           02 FILLER          PICTURE X(16) VALUE SPACES.

       01  DETALLE.
           02 FILLER          PICTURE X(12) VALUE SPACES.
           02 NOMBRE-SAL      PICTURE X(26).
           02 FILLER          PICTURE X(02) VALUE SPACES.
           02 TIPO-SOCIO-SAL  PICTURE X(10).
           02 FILLER          PICTURE X(03) VALUE SPACES.
           02 CUOTA-SAL       PICTURE $Z(03),Z(03).Z(02).
           02 FILLER          PICTURE X(36) VALUE SPACES.
       01  DEF-SOCIOS.
           02 FILLER          PICTURE X(10) VALUE 'VITALICIO '.
           02 FILLER          PICTURE X(10) VALUE 'HONORARIO '.
           02 FILLER          PICTURE X(10) VALUE 'CONSEJERO '.
           02 FILLER          PICTURE X(10) VALUE 'DONADOR   '.
           02 FILLER          PICTURE X(10) VALUE 'NORMAL    '.
       01  TA-SOCIOS REDEFINES DEF-SOCIOS.
           02 TAB-SOCIOS      PICTURE X(10) OCCURS 5 TIMES.
       PROCEDURE DIVISION.
       MAIN-LOGIC.
      *
      *    Abrir el archivo de cuotas y el de impresion.
      *

           OPEN INPUT  CARDS.
           OPEN OUTPUT IMPRESORA.
      *
      *    Se lee la fecha y hora de reporte del sistema.
      *
           ACCEPT FECHA FROM DATE.
           ACCEPT HORA  FROM TIME.
      *
      *    Se evalua los datos de fecha del encabezado 1
      * y hora del encabezado 2.
      *
           MOVE FECHA TO FECHA-SAL.
           MOVE HH TO H-S.
           MOVE MM TO M-S.
           MOVE SS TO S-S.
           MOVE FF TO F-S.
      *
      *    Se imprime el primer encabezado y subencabezado.
      *

           PERFORM IMPRIME-ENCABEZADO.
           PERFORM IMPRIME-SUBENCABEZADO.
      *
      *    Se imprimiran los registros detalle llevando un
      * control de hoja.
      *
           MOVE 'NO' TO END-OF-DATA-INDICATOR.
           PERFORM READ-A-CARD.
           PERFORM READ-PRINT
                   UNTIL END-OF-DATA-INDICATOR IS EQUAL TO 'YES'.
```

```
*
*    Cerrar los archivos de datos y fin de programa.
*
     CLOSE CARDS.
     CLOSE IMPRESORA.
     STOP RUN.
 READ-A-CARD.
*
*    Lectura del siguiente registro, si ya no existe otro
* registro la variable END-OF-DATA-INDICATOR tendra el valor
* de 'YES'.
*
     READ CARDS RECORD
          AT END MOVE 'YES' TO END-OF-DATA-INDICATOR.
 IMPRIME-ENCABEZADO.
*
*    Impresion de los registros de encabezado para cada
* pagina.
*
     ADD 1 TO HOJA.
     MOVE  HOJA TO HOJA-SAL.
     WRITE PRINT-LINE FROM ENCABEZADO-1 AFTER ADVANCING PAGE.
     WRITE PRINT-LINE FROM ENCABEZADO-2 AFTER ADVANCING 1 LINE.
     WRITE PRINT-LINE FROM ENCABEZADO-3 AFTER ADVANCING 1 LINE.
     MOVE 4 TO LINEA.
 IMPRIME-SUBENCABEZADO.
*
*    Impresion de las lineas de subencabezado.
*
     WRITE PRINT-LINE FROM ENCABEZADO-4 AFTER ADVANCING 2 LINE.
     WRITE PRINT-LINE FROM ENCABEZADO-5 AFTER ADVANCING 1 LINE.
     ADD 3 TO LINEA.
 READ-PRINT.
*
*    Imprime el reporte requerido.
*
*
*    Control de pagina.
*
     IF LINEA IS GREATER 60
        PERFORM IMPRIME-ENCABEZADO
        PERFORM IMPRIME-SUBENCABEZADO.
*
*   Definicion del registro de detalle y su impresion.
*
     MOVE NOMBRE TO NOMBRE-SAL.
     MOVE TAB-SOCIOS(TIPO-SOCIO) TO TIPO-SOCIO-SAL.
     MOVE CUOTA TO CUOTA-SAL.
     WRITE PRINT-LINE FROM DETALLE AFTER ADVANCING 1 LINE.
     ADD 1 TO LINEA.
     PERFORM READ-A-CARD.
```

APPENDIX U

EJEMPLO DE BUSQUEDAS

El siguiente ejemplo ilustra la manera en que se pueden hacer búsquedas sobre una tabla. La tabla (llamada TABLA en el programa) posee dos llaves de ordenamiento (código y nombre) y un índice (I). El índice almacenará la posición del elemento buscado (después del SEARCH). El programa pregunta un número de código a buscar. En caso de ser exitosa la búsqueda se despliega el nombre de la estación, de lo contrario se despliega un mensaje de error.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    BUSQUEDAS.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  CODIGO-LECTURA  PIC 9(03).
01  TABLA-1.
    02  FILLER PIC X(18) VALUE '105QUERETARO'.
    02  FILLER PIC X(18) VALUE '106CELAYA'.
    02  FILLER PIC X(18) VALUE '107IRAPUATO'.
    02  FILLER PIC X(18) VALUE '109SALAMANCA'.
    02  FILLER PIC X(18) VALUE '110AGUASCALIENTES'.
    02  FILLER PIC X(18) VALUE '200AGUASCALIENTES'.
    02  FILLER PIC X(18) VALUE '203AGUASCALIENTES'.
    02  FILLER PIC X(19) VALUE '204LAGOS'.
    02  FILLER PIC X(18) VALUE '209ZACATECAS'.
    02  FILLER PIC X(18) VALUE '301FRESNILLO'.
    02  FILLER PIC X(18) VALUE '302CHIHUAHUA'.
    02  FILLER PIC X(18) VALUE '308JUAREZ'.
01  TABLA     02  ESTACION OCCURS 12 TIMES
        ASCENDING KEY CODIGO
        ASCENDING KEY NOMBRE INDEXED BY I.
        03  CODIGO    PIC 9(03).
        03  NOMBRE    PIC X(15).
PROCEDURE DIVISION.
PRINCIPAL.
    DISPLAY 'Que codigo deseas buscar 9(03) ' WITH NO ADVANCING.
    ACCEPT CODIGO-LECTURA.
    PERFORM BUSQUEDA-SECUENCIAL.
    PERFORM BUSQUEDA-BINARIA.
    STOP RUN.
BUSQUEDA-SECUENCIAL.
    SET I TO 1.
    SEARCH ESTACION
          AT END DISPLAY
```

APPENDIX U

EJERCICIO CON LA INSTRUCCION INSPECT

	nom-campo-datos antes	nom-campo-datos después	literal-1
INSPECT nom-campo-datos TALLYING literal-1 FOR LEADING 'L'	LULU	LULU	1
	HOLA	HOLA	0
	LLORA	LLORA	2
INSPECT nom-campo-datos TALLYING literal-1 FOR ALL 'L'	LULU	LULU	2
	HOLA	HOLA	1
	LLORA	LLORA	2
INSPECT nom-campo-datos REPLACING ALL 'M' BY 'P'	MAMA	PAPA	-
	MAMUT	PAPUT	2
INSPECT nom-campo-datos TALLYING literal-1 FOR CHARACTERS AFTER 'S' REPLACING ALL 'I' BY 'O'	MISSISSIPI	MOSSOSSOPO	7
	HISTORIA	HOSTOROA	5
INSPECT nom-campo-datos REPLACING ALL 'I' BY 'O' ALL 'S' BY 'Z' AFTER 'MIS'	MISSISSIPI	MOSZOZZOPO	-
	HEMISFERIO	HEMOSFEROO	-


```
           'No existe estacion con codigo ' CODIGO-LECTURA
         WHEN CODIGO(I) = CODIGO-LECTURA PERFORM  EXITO.
BUSQUEDA-BINARIA.
    SEARCH ALL ESTACION
         AT END DISPLAY
           'No existe estacion con codigo ' CODIGO-LECTURA
         WHEN NOMBRE(I) = 'AGUASCALIENTES' AND
              CODIGO(I) = CODIGO-LECTURA   PERFORM  EXITO.
EXITO.
    DISPLAY CODIGO(I) '     ' NOMBRE(I).
```

APPENDIX W

PROGRAMA

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     PRUEBA.
AUTHOR.         DECFI.
INSTALLATION.   IBM-PC.
DATE-WRITTEN.   10 JUNIO 87.
DATE-COMPILED.  10 JUNIO 87.
SECURITY.       NINGUNA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  VAX-11-780.
OBJECT-COMPUTER.  VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATOS   ASSIGN DISK.
    SELECT NOMYRFC ASSIGN DISK.
DATA DIVISION.
FILE SECTION.
FD  DATOS
    RECORD CONTAINS 36 CHARACTERS
    LABEL RECORD IS STANDARD
    VALUE OF ID IS 'DATOS.DAT'
    DATA RECORD IS REGISTRO-DATOS.
01  REGISTRO-DATOS.
    03 NOMBRE                    PIC X(30).
    03 DIA                       PIC 99.
    03 MES                       PIC 99.
    03 ANO                       PIC 99.
FD  NOMYRFC
    RECORD CONTAINS 40 CHARACTERS
    LABEL RECORD IS STANDARD
    VALUE OF ID IS 'NOMYRFC.DAT'
    DATA RECORD IS REGISTRO-NOMYRFC.
01  REGISTRO-NOMYRFC.
    03 NOMBRE.
       05 ACOMODADO         OCCURS 30 TIMES PIC X(01).
    03 LETRAS                    PIC A(04).
    03 ANO                       PIC 99.
    03 MES                       PIC 99.
    03 DIA                       PIC 99.
WORKING-STORAGE SECTION.
```

```
77  BLANCOS                     PIC 99 VALUE ZEROES.
77  PUNTO-Y-COMA-1              PIC 99 VALUE ZEROES.
77  PUNTO-Y-COMA-2              PIC 99 VALUE ZEROES.
77  ENCONTRAR                   PIC X(02) VALUE 'NO'.
    88 YA-LO-ENCONTRE           VALUE 'SI'.
77  I                           PIC 99 VALUE ZEROES.
77  J                           PIC 99 VALUE ZEROES.
77  DATOS-STATUS                PIC X(02) VALUE 'NO'.
    88 FIN-ARCHIVO-DATOS        VALUE 'SI'.
77  VOCALES                     PIC A VALUE 'B'.
    88 VOCAL                    VALUE 'A', 'E', 'I', 'O', 'U'.
01  NOMBRES.
    03 INDICE OCCURS 30 TIMES PIC X(01).
01  LETRAS-RFC.
    03 INDICE-LETRA OCCURS 4 TIMES PIC A(01).
PROCEDURE DIVISION.
INICIO.
    PERFORM PROCESO-INICIAL.
    PERFORM PROCESO-PRINCIPAL UNTIL FIN-ARCHIVO-DATOS.
    PERFORM PROCESO-FINAL.
    STOP RUN.
PROCESO-INICIAL.
    OPEN INPUT  DATOS.
    OPEN OUTPUT NOMYRFC.
PROCESO-FINAL.
    CLOSE DATOS.
    CLOSE NOMYRFC.
PROCESO-PRINCIPAL.
    READ DATOS AT END
         MOVE 'SI' TO DATOS-STATUS.
    MOVE ZEROES TO PUNTO-Y-COMA-1
                   PUNTO-Y-COMA-2
                   BLANCOS
                   J.
    MOVE 1 TO I
    MOVE 'NO' TO ENCONTRAR.
    IF NOT FIN-ARCHIVO-DATOS
       PERFORM INSPECCIONA THRU IMPRIME.
INSPECCIONA.
    MOVE NOMBRE IN REGISTRO-DATOS TO NOMBRES.
    MOVE INDICE(1) TO INDICE-LETRA(1).
    PERFORM ENCUENTRA-VOCAL UNTIL YA-LO-ENCONTRE.
    MOVE VOCALES TO INDICE-LETRA(2).
    INSPECT NOMBRES REPLACING ALL ' ' BY '' BEFORE INITIAL ';'.
    INSPECT NOMBRES TALLYING PUNTO-Y-COMA-1 FOR CHARACTERS
            BEFORE INITIAL ';'
    ADD 2 TO PUNTO-Y-COMA-1
    MOVE INDICE(PUNTO-Y-COMA-1) TO INDICE-LETRA(3).
    INSPECT NOMBRES REPLACING FIRST ';' BY ''.
    INSPECT NOMBRES REPLACING ALL ' ' BY '' BEFORE INITIAL ';'.
    INSPECT NOMBRES TALLYING PUNTO-Y-COMA-2 FOR CHARACTERS
            BEFORE INITIAL ';' REPLACING FIRST ';' BY ''.
    ADD 2 TO PUNTO-Y-COMA-2
    MOVE INDICE(PUNTO-Y-COMA-2) TO INDICE-LETRA(4).
    MOVE CORR REGISTRO-DATOS TO REGISTRO-NOMYRFC
    MOVE LETRAS-RFC TO LETRAS IN REGISTRO-NOMYRFC
```

```
           INSPECT NOMBRES TALLYING BLANCOS FOR CHARACTERS
                   BEFORE INITIAL ' '.
           ADD 2 TO BLANCOS.
           IF BLANCOS > 30
              MOVE 30 TO BLANCOS.
           PERFORM ACOMODA-NOMBRE VARYING I FROM PUNTO-Y-COMA-2 BY 1
                   UNTIL I = BLANCOS.
           PERFORM ACOMODA-NOMBRE VARYING I FROM 1 BY 1
                   UNTIL I = PUNTO-Y-COMA-2.
           INSPECT NOMBRE IN REGISTRO-NOMYRFC REPLACING ALL ' ' BY ' '.
       IMPRIME.
           WRITE REGISTRO-NOMYRFC.
       ENCUENTRA-VOCAL.
           ADD 1 TO I.
           MOVE INDICE(I) TO VOCALES.
           IF VOCAL
              MOVE 'SI' TO ENCONTRAR.
       ACOMODA-NOMBRE.
           ADD 1 TO J.
           MOVE INDICE(I) TO ACOMODADO(J).
```

APPENDIX X

PROGRAMA

```
identification division.
*****************************************************************************
program-id.    string01.
*****************************************************************************
*
* Este programa usa la instruccion unstring para transferir un dato capturado
* en terminal y puesto en un archivo de salida. Tambien es usado la
* instruccion string para la transferencia de datos
*
*****************************************************************************
environment division.
*****************************************************************************
input-output section.
file-control.

     select name-master assign to 'namemst.dat',
     select print-file  assign to 'mstlst.lis'.
data division.
*****************************************************************************
file section.
fd name-master
   label records are standard.
01 name-record.
   03 n-number      pic x(05).
   03 n-name        pic x(25).
   03 n-address-1   pic x(25).
   03 n-address-2   pic x(25).
   03 n-city        pic x(20).
   03 n-state       pic x(02).
   03 n-zip         pic x(05).
fd print-file
   label records are omitted.
01 print-record     pic x(132).
working-storage section.
01 error-sw         pic x(01) value 'n'.
01 current-date.
   03 c-yy          pic 9(02).
   03 c-mm          pic 9(02).
   03 c-dd          pic 9(02).
01 the-counters usage comp.
   03 page-ct       pic 9(03) value zeroes.
   03 line-ct       pic 9(03) value zeroes.
```

```
   03 name-ct      pic 9(03) value zeroes.
   03 mss-sub      pic 9(02) value zeroes.
   03 sub-compare  pic 9(02) value zeroes.
01 terminal-input  pic x(113).
*****************************************************************************
*
* El registro terminal-input(113) contiene en longitud al registro
* name-record(107) mas 6 delimitadores que se introducen.
*
*****************************************************************************
01 head-a.
   03 filler       pic x(35) value spaces.
   03 filler       pic x(16) value 'name master list'.
   03 filler       pic x(20) value spaces.
   03 filler       pic x(06) value 'as of '.
   03 h-date.
      05 h-mm      pic z(02).
      05 filler    pic x(01) value '/'.
      05 h-dd      pic z(02).
      05 filler    pic x(01) value '/'.
      05 h-yy      pic z(02).
   03 filler       pic x(31) value spaces.
   03 filler       pic x(05) value 'page '.
   03 h-page-ts    pic z(02)9(01).
01 head-b.
   03 filler       pic x(13) value '      name  '.
   03 filler       pic x(20) value spaces.
   03 filler       pic x(13) value '   address   '.
   03 filler       pic x(12) value spaces.
   03 filler       pic x(08) value '  city  '.
   03 filler       pic x(12) value spaces.
   03 filler       pic x(15) value '  state   zip  '.
   03 filler       pic x(03) value spaces.
procedure division.
*****************************************************************************
000-begin.
    open output name-master.
    open output print-file.
    accept current-date from date.
    move c-mm to h-mm.
    move c-dd to h-dd.
    move c-yy to h-yy.
010-print-headings.
    add 1 to page-ct.
    move page-ct to h-page-ts.
    write print-record from head-a after page.
    write print-record from head-b after advancing 2 lines.
    move spaces to print-record.
    write print-record after advancing 1 line.
    move 4 to line-ct.
020-get-terminal-input.
    display 'enter data in this format : [End 99999 in number]'.
    display 'number/name/address-1/address-2/city/state/zip'.
    accept  terminal-input.
    unstring terminal-input
             delimited by '/'
```

```
           into n-number
                n-name count in name-ct
                n-address-1
                n-address-2
                n-city
                n-state
                n-zip.
    if n-number = '99999' then
       go to 999-eoj.
030-validate-data.
    if n-number is not numeric then
       move 'y' to error-sw
       display 'non-numeric number'.
    if n-name = spaces then
       move 'y' to error-sw
       display 'name is missing'.
    if name-ct greater than 25 then
       move 'y' to error-sw
       display 'name is greater then 25 characters'.
    if n-address-1 = spaces then
       move 'y' to error-sw
       display 'address-1 missing'.
    if n-address-2 = spaces then
       move 'y' to error-sw
       display 'address-2 missing'.
    if n-city = spaces then
       move 'y' to error-sw
       display 'city missing'.
    if n-state = spaces then
       move 'y' to error-sw
       display 'state missing'.
    if n-zip is not numeric then
       move 'y' to error-sw
       display 'zip code is non-numeric'.
    if error-sw = 'y' then
       move 'n' to error-sw
       display 'transaction rejected ' terminal-input
       go to 020-get-terminal-input.
040-build-line-1.
    if line-ct greater than 56 then
       perform 010-print-headings.
    string '   ' n-name '   ' n-address-2 '   ' n-city '   ' n-state
           '   ' n-zip  delimited by size into print-record.
    write print-record after advancing 1 lines.
    add 1 to line-ct.
    write name-record.
    move spaces to name-record.
    move spaces to print-record.
    move zeroes to name-ct.
    go to 020-get-terminal-input.
999-eoj.
    close name-master.
    close print-file.
    display 'end of program '.
    stop run.
```

APPENDIX Y

EJEMPLO DE ORDENAMIENTO

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    ORDENADOR.
ENVIRONMENT    DIVISION.
CONFIGURATION  SECTION.
SOURCE-COMPUTER.       VAX-11-780.
OBJECT-COMPUTER.       VAX-11-780.
INPUT-OUTPUT   SECTION.
FILE-CONTROL.
     SELECT ARCH-ENT   ASSIGN TO 'AORDENAR.DAT'.
     SELECT ARCH-ORD   ASSIGN TO 'ORDENADO.DAT'.
     SELECT ARCH-INT   ASSIGN TO 'INTER.DAT'.
DATA DIVISION.
FILE SECTION.
FD   ARCH-ENT.
01   REG-ENT           PIC X(13).
FD   ARCH-ORD.
01   REG-ORD           PIC X(13).
SD   ARCH-INT.
01   REG-INTER.
     02  FILLER             PIC X(04).
     02  CODIGO-CLIENTE     PIC X(03).
     02  TOTAL-VENTAS       PIC 9(06).
PROCEDURE DIVISION.
ORDENA.
     SORT  ARCH-INT
           ON ASCENDING  KEY CODIGO-CLIENTE
           DESCENDING KEY TOTAL-VENTAS
           USING   ARCH-ENT
           GIVING  ARCH-ORD.
     STOP RUN.
```

APPENDIX Z

ORDENAMIENTO EMPLEANDO INPUT-OUTPUT PROCEDURE

```
IDENTIFICATION DIVISION.
PROGRAM-ID. INPUT-OUTPUT-PROCEDURE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT ENTRADA ASSIGN TO 'CLIENTES.DAT'.
     SELECT SALIDA  ASSIGN TO 'SALIDA.DAT'.
     SELECT SORTEO  ASSIGN TO 'SORTEO.SOR'.
DATA DIVISION.
FILE SECTION.
FD   ENTRADA
     RECORD CONTAINS 67 CHARACTERS
     DATA RECORDS ARE REG-ENT-1, REG-ENT-2.
01 REG-ENT-1.
     03 CLAVE-ENT                 PIC 9(04).
     03 NOMBRE-ENT                PIC X(32).
     03 DIRECCION-ENT             PIC X(30).
     03 CREDITO-ENT               PIC 9(01).
01 REG-ENT-2.
     03 CRED-MAX-ENT              PIC 9(06)V9(02).
     03 FILLER                    PIC X(59).
FD SALIDA
     RECORD CONTAINS 67 CHARACTERS
     DATA RECORDS ARE REG-SAL-1, REG-SAL-2.
01 REG-SAL-1.
     03 CLAVE-SAL                 PIC 9(04).
     03 NOMBRE-SAL                PIC X(32).
     03 DIRECCION-SAL             PIC X(30).
     03 CREDITO-SAL               PIC 9(01).
01 REG-SAL-2.
     03 CRED-MAX-SAL              PIC 9(06)V9(02).
     03 FILLER                    PIC X(59).
SD SORTEO
     RECORD CONTAINS 75 CHARACTERS
     DATA RECORD IS REG-SOR.
01 REG-SOR.
     03 CLAVE-SOR                 PIC 9(04).
     03 NOMBRE-SOR                PIC X(32).
     03 DIRECCION-SOR             PIC X(30).
```

```
     03 CREDITO-SOR                PIC 9(01).
     03 CRED-MAX-SOR               PIC 9(08).
WORKING-STORAGE SECTION.
77   TALLY                        PIC 9(02) VALUE ZEROES.
77   HAY-DATOS-ENT                PIC X(02) VALUE 'SI'.
     88  HAY-DATOS-ENT-OK                   VALUE 'NO'.
77   HAY-DATOS-SAL                PIC X(02) VALUE 'SI'.
     88  HAY-DATOS-SAL-OK                   VALUE 'NO'.
PROCEDURE DIVISION.
PRINCIPAL SECTION.
INICIO.
     PERFORM ABRE-ARCHIVOS.
     SORT SORTEO ON ASCENDING KEY CLAVE-SOR
          INPUT PROCEDURE ENTRADA-SORTEO
          OUTPUT PROCEDURE SALIDA-SORTEO.
     PERFORM CIERRA-ARCHIVOS.
     STOP RUN.
ABRE-ARCHIVOS SECTION.
ABRE.
     OPEN INPUT ENTRADA
          OUTPUT SALIDA.
ENTRADA-SORTEO SECTION.
ENTRADA-SORT.
     READ ENTRADA AT END
          DISPLAY 'ERROR FATAL .... NO HAY DATOS DE ENTRADA'
          MOVE 'NO' TO HAY-DATOS-ENT.
     PERFORM LEE-ENTRADA UNTIL HAY-DATOS-ENT-OK.
LEE-ENTRADA SECTION.
LEE.
     MOVE CLAVE-ENT TO CLAVE-SOR
     MOVE NOMBRE-ENT TO NOMBRE-SOR
     MOVE DIRECCION-ENT TO DIRECCION-SOR
     MOVE CREDITO-ENT TO CREDITO-SOR
     IF CREDITO-ENT IS EQUAL 1
         READ ENTRADA AT END
             MOVE 'NO' TO HAY-DATOS-ENT
             END-READ
         MOVE CRED-MAX-ENT TO CRED-MAX-SOR
     ELSE
         MOVE ZEROES TO CRED-MAX-SOR
     RELEASE REG-SOR
     READ ENTRADA AT END
             MOVE 'NO' TO HAY-DATOS-ENT.
SALIDA-SORTEO SECTION.
SALIDA-SORT.
    RETURN SORTEO AT END MOVE 'NO' TO HAY-DATOS-SAL.
    PERFORM IMP-SALIDA UNTIL HAY-DATOS-SAL-OK.
IMP-SALIDA SECTION.
IMP.
    INSPECT NOMBRE-SOR TALLYING TALLY FOR LEADING 'A'
    IF TALLY IS NOT EQUAL ZERO
        MOVE CLAVE-SOR TO CLAVE-SAL
        MOVE NOMBRE-SOR TO NOMBRE-SAL
        MOVE DIRECCION-SOR TO DIRECCION-SAL
        MOVE CREDITO-SOR TO CREDITO-SAL
        WRITE REG-SAL-1
```

```
        IF CREDITO-SOR IS EQUAL 1
             MOVE CRED-MAX-SOR TO CRED-MAX-SAL
             WRITE REG-SAL-2.
    RETURN SORTEO AT END
        MOVE 'NO' TO HAY-DATOS-SAL.
CIERRA-ARCHIVOS SECTION.
CIERRA.
    CLOSE ENTRADA SALIDA.
```

APPENDIX AA

PROGRAMA

```
      IDENTIFICATION DIVISION.
      PROGRAM-ID.     SORTEO.
      AUTHOR.         CURSO COBOL DECFI.
      INSTALLATION.   DECFI.
      DATE-WRITTEN.   26 MAYO 1987.
      DATE-COMPILED.  26 MAYO 1987.
      SECURITY.       NINGUNA.
     *
     *    Sortear un archivo de datos por nombre.
     *
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER. IBM-PC.
      OBJECT-COMPUTER. IBM-PC.
      INPUT-OUTPUT SECTION.
      FILE-CONTROL.
           SELECT CARDS          ASSIGN TO DISK.
           SELECT SORTING-FILE   ASSIGN TO DISK.
      DATA DIVISION.
      FILE SECTION.
      FD   CARDS
           LABEL RECORD ARE STANDARD
           VALUE OF FILE-ID IS NOMBRE-DE-ARCHIVO
           DATA RECORD IS CARD.
      01   CARD.
           02 NOMBRE         PICTURE X(26).
           02 FILLER         PICTURE X(54).
      SD   SORTING-FILE
           DATA RECORD IS SORT-CARD.
      01   SORT-CARD.
           02 NOMBRE-SORT    PICTURE X(26).
           02 FILLER         PICTURE X(54).
      WORKING-STORAGE SECTION.
      01   NOMBRE-DE-ARCHIVO PICTURE IS X(13).
      01   TIPO-SORT         PICTURE IS X(01) VALUE SPACE.
           88 TIPO-SORT-OK   VALUE 'A', 'a', 'D', 'd'.
           88 TIPO-SORT-A    VALUE 'A', 'a'.
           88 TIPO-SORT-D    VALUE 'D', 'd'.
      PROCEDURE DIVISION.
      MAIN-LOGIC.
```

```
*
*    Pide el archivo a sortear y lo ordena por el nombre en
* forma indicada.
*
     DISPLAY 'NOMBRE DEL ARCHIVO A SORTEAR'
     ACCEPT NOMBRE-DE-ARCHIVO.
     PERFORM PIDE-OPCION-DE-SORTEO UNTIL TIPO-SORT-OK.
     IF TIPO-SORT-A
        SORT SORTING-FILE  ON ASCENDING KEY NOMBRE-SORT
             USING CARDS GIVING CARDS
     ELSE
        SORT SORTING-FILE  ON DESCENDING KEY NOMBRE-SORT
             USING CARDS GIVING CARDS.
     STOP RUN.
 PIDE-OPCION-DE-SORTEO.
*
*    Pide la opcion de sorteo.
*
     DISPLAY 'TIPO DE ORDENAMIENTO (ASCENDENTE/DESCENDENTE) [A/D]'
     ACCEPT TIPO-SORT.
```

APPENDIX AB

EJEMPLO DE ARCHIVOS RELATIVOS

```
IDENTIFICATION DIVISION.
PROGRAM-ID.       ARCHIVO-RELATIVO.
AUTHOR.           CECAFI.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT LISTA ASSIGN TO DISK
           ORGANIZATION IS RELATIVE
           ACCESS  MODE IS DYNAMIC
           RELATIVE KEY IS R-CLAVE
           FILE STATUS IS FS-LISTA-X.
DATA DIVISION.
FILE SECTION.
FD LISTA
    RECORD CONTAINS  32 CHARACTERS
    LABEL  RECORD IS STANDARD
    VALUE  OF ID IS 'RLISTA.DAT'
    DATA RECORD IS REGISTRO-LISTA.
01  REGISTRO-LISTA.
    03 NOMBRE                  PICTURE IS X(32).
WORKING-STORAGE SECTION.
77  TOTAL-DE-LISTA PICTURE IS 9(02).
77  FINAL-DE-DATOS PICTURE X(01).
    88 FIN-DE-DATOS VALUE 'N' 'n'.
77  HAY-ERROR PICTURE IS X(01).
    88 ERROR-OK VALUE 'S' 's'.
77  OPCION PICTURE IS X(01).
    88 OPCION-SI VALUE 'S' 's'.
    88 TERMINAR  VALUE 'F' 'f'.
77  FS-LISTA-X PICTURE IS X(02).
77  R-CLAVE    PICTURE IS 9(03).
PROCEDURE DIVISION.
INICIO.
    PERFORM INICIO-PROCESO.
    PERFORM PROCESO UNTIL TERMINAR.
    STOP RUN.
```

```
INICIO-PROCESO.
    MOVE ' ' TO OPCION.
    MOVE ZEROES TO TOTAL-DE-LISTA.
    OPEN I-O LISTA.
PROCESO.
    DISPLAY ' '.
    DISPLAY 'OPCIONES :  '.
    DISPLAY ' '.
    DISPLAY 'A: ALTAS.'.
    DISPLAY 'B: BAJAS.'.
    DISPLAY 'C: CONSULTA.'.
    DISPLAY 'M: MODIFICACION.'.
    DISPLAY 'F: FIN.'
    DISPLAY ' '
    ACCEPT OPCION.
    MOVE ' ' TO FINAL-DE-DATOS.
    IF OPCION EQUAL 'A' OR EQUAL 'a'
        PERFORM PROCESO-ALTAS
                UNTIL FIN-DE-DATOS
    ELSE
        IF OPCION EQUAL 'B' OR EQUAL 'b'
            PERFORM PROCESO-BAJAS
                    UNTIL FIN-DE-DATOS
        ELSE
            IF OPCION EQUAL 'C' OR EQUAL 'c'
                PERFORM PROCESO-CONSULTA
                        UNTIL FIN-DE-DATOS
            ELSE
                IF OPCION EQUAL 'M' OR EQUAL 'm'
                    PERFORM PROCESO-MODIFICACIONES
                            UNTIL FIN-DE-DATOS.
PROCESO-ALTAS.
    MOVE 'S' TO FINAL-DE-DATOS.
    PERFORM LEE-DATOS.
LEE-DATOS.
    DISPLAY ' '
    DISPLAY 'CAPTURA DE INFORMACION (s/n) ? '.
    ACCEPT FINAL-DE-DATOS.
    IF FIN-DE-DATOS
       NEXT SENTENCE
    ELSE
       PERFORM PROCESA-ALTAS.
PROCESA-ALTAS.
    PERFORM LEE-CLAVE.
    MOVE ' ' TO HAY-ERROR.
    READ LISTA RECORD
         INVALID KEY MOVE 'S' TO HAY-ERROR.
    IF ERROR-OK
       PERFORM LEE-DATOS-LISTA
       WRITE REGISTRO-LISTA INVALID KEY
             DISPLAY 'WRITE : REGISTRO EXISTENTE.'
    ELSE
       DISPLAY ' '
       DISPLAY 'REGISTRO EXISTENTE.'
       PERFORM ESCRIBE-DATOS-LISTA.
LEE-DATOS-LISTA.
```

```
    DISPLAY ' '.
    DISPLAY 'NOMBRE ? '
    ACCEPT NOMBRE.
LEE-CLAVE.
    DISPLAY ' '
    DISPLAY 'NUMERO DE REGISTRO 9(03)'.
    ACCEPT R-CLAVE.
PROCESO-BAJAS.
    MOVE 'S' TO FINAL-DE-DATOS.
    PERFORM BORRA-DATOS UNTIL FIN-DE-DATOS.
BORRA-DATOS.
    DISPLAY ' '
    DISPLAY 'BORRADO DE INFORMACION (S/N) ? '.
    ACCEPT FINAL-DE-DATOS.
    IF FIN-DE-DATOS
       NEXT SENTENCE
    ELSE
       PERFORM BAJAS-DATOS-LISTA.
BAJAS-DATOS-LISTA.
    PERFORM LEE-CLAVE.
    MOVE ' ' TO HAY-ERROR.
    READ LISTA RECORD
       INVALID KEY MOVE 'S' TO HAY-ERROR.
    IF ERROR-OK
       DISPLAY ' '
       DISPLAY 'NO EXISTE EL REGISTRO'
    ELSE
       PERFORM ESCRIBE-DATOS-LISTA
       PERFORM SE-BORRA.
SE-BORRA.
    DISPLAY ' '.
    DISPLAY 'SE DESEA BORRAR INFORMACION (S/N) ?'.
    ACCEPT  OPCION.
    IF OPCION-SI
       DELETE LISTA RECORD
              INVALID KEY DISPLAY 'NO SE PUEDE BORRAR.'.
ESCRIBE-DATOS-LISTA.
    DISPLAY ' '.
    DISPLAY 'INFORMACION DEL REGISTRO LISTA.'
    DISPLAY ' '.
    DISPLAY 'CLAVE : ' R-CLAVE.
    DISPLAY 'NOMBRE : ' NOMBRE.
PROCESO-CONSULTA.
    MOVE 'S' TO FINAL-DE-DATOS.
    PERFORM CONSULTA-DATOS UNTIL FIN-DE-DATOS.
CONSULTA-DATOS.
    DISPLAY ' '
    DISPLAY 'CONSULTA DE INFORMACION (S/N) ? '.
    ACCEPT FINAL-DE-DATOS.
    IF FIN-DE-DATOS
      NEXT SENTENCE
    ELSE
       PERFORM CONSULTA-DATOS-LISTA.
CONSULTA-DATOS-LISTA.
    PERFORM LEE-CLAVE.
    MOVE ' ' TO HAY-ERROR.
```

```
    READ LISTA RECORD
         INVALID KEY MOVE 'S' TO HAY-ERROR.
    IF ERROR-OK
       DISPLAY ' '
       DISPLAY 'NO EXISTE EL REGISTRO'
    ELSE
       PERFORM ESCRIBE-DATOS-LISTA.
PROCESO-MODIFICACIONES.
    MOVE 'S' TO FINAL-DE-DATOS.
    PERFORM MODIFICA-DATOS UNTIL FIN-DE-DATOS.
MODIFICA-DATOS.
    DISPLAY ' '
    DISPLAY 'MODIFICACION DE INFORMACION (S/N) ? '.
    ACCEPT FINAL-DE-DATOS.
    IF FIN-DE-DATOS
       NEXT SENTENCE
    ELSE
       PERFORM MODIFICA-DATOS-LISTA.
MODIFICA-DATOS-LISTA.
    PERFORM LEE-CLAVE.
    MOVE ' ' TO HAY-ERROR.
    READ LISTA RECORD
         INVALID KEY MOVE 'S' TO HAY-ERROR.
    IF ERROR-OK
       DISPLAY ' '
       DISPLAY 'NO EXISTE EL REGISTRO'
    ELSE
       PERFORM ESCRIBE-DATOS-LISTA
       PERFORM SE-CAMBIA.
SE-CAMBIA.
    DISPLAY ' '.
    DISPLAY 'SE CAMBIA LA INFORMACION (S/N) ? '.
    ACCEPT  OPCION.
    IF OPCION-SI
       PERFORM LEE-DATOS-LISTA
       REWRITE REGISTRO-LISTA
              INVALID KEY DISPLAY 'NO SE PUEDE MODIFICAR'.
```

APPENDIX AC

PROGRAMA

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.       ARCHIVO-RELATIVO.
 AUTHOR.           CURSO COBOL DECFI.
 INSTALLATION.     DECFI.
 DATE-WRITTEN.     1 JUNIO 1987.
 DATE-COMPILED.    1 JUNIO 1987.
 SECURITY.         NINGUNA.
*
*    Manejo de archivos relativos.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-PC.
 OBJECT-COMPUTER. IBM-PC.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
 SELECT LISTA ASSIGN TO DISK
        ORGANIZATION IS RELATIVE
        ACCESS  MODE IS DYNAMIC
        RELATIVE KEY IS R-CLAVE
        FILE STATUS IS FS-LISTA-X.
 DATA DIVISION.
 FILE SECTION.
 FD  LISTA
     RECORD CONTAINS  32 CHARACTERS
     LABEL  RECORD IS STANDARD
     VALUE  OF FILE-ID IS 'RLISTA.DAT'
     DATA RECORD IS REGISTRO-LISTA.
 01  REGISTRO-LISTA.
     03 NOMBRE                  PICTURE IS X(32).
 WORKING-STORAGE SECTION.
 77  TOTAL-DE-LISTA PICTURE IS 9(02).
 77  FINAL-DE-DATOS PICTURE X(01).
     88 FIN-DE-DATOS VALUE 'N' 'n'.
 77  HAY-ERROR PICTURE IS X(01).
     88 ERROR-OK VALUE 'S' 's'.
 77  OPCION PICTURE IS X(01).
     88 OPCION-SI VALUE 'S' 's'.
     88 TERMINAR  VALUE 'F' 'f'.
 77  FS-LISTA-X PICTURE IS X(02).
```

```
       77  R-CLAVE    PICTURE IS 9(03).
       PROCEDURE DIVISION.
       INICIO.
      *
      *    Proceso bajo control por menu de opciones.
      *
           PERFORM INICIO-PROCESO.
           PERFORM PROCESO UNTIL TERMINAR.
           PERFORM FIN-PROCESO.
           STOP RUN.
       INICIO-PROCESO.
      *
      *    Define las condiciones iniciales del proceso.
      *
           PERFORM REVISA-ARCHIVO.
           MOVE ' ' TO OPCION.
           MOVE ZEROES TO TOTAL-DE-LISTA.
           OPEN I-O LISTA.
       REVISA-ARCHIVO.
      *
      *    El  archivo  a  usar  debe  de existir en el disco de
      * otro  modo  no  se  podra accesar la informacion. Por lo
      * tanto debe revisarse su existencia con FS-LISTA-X, si el
      * valor es '00' es que  el  archivo  ya existe en el disco
      * si es igual a '30' no existe el archivo.
      *
           OPEN I-O LISTA.
           IF FS-LISTA-X = '30'
              CLOSE LISTA
              OPEN OUTPUT LISTA
              CLOSE LISTA
           ELSE
              CLOSE LISTA.
       PROCESO.
      *
      *    Proceso controlado por este menu de opciones sobre
      *  el archivo relativo.
      *
           DISPLAY ' '.
           DISPLAY 'OPCIONES :  '.
           DISPLAY ' '.
           DISPLAY 'A: ALTAS.'.
           DISPLAY 'B: BAJAS.'.
           DISPLAY 'C: CONSULTA.'.
           DISPLAY 'M: MODIFICACION.'.
           DISPLAY 'F: FIN.'
           DISPLAY ' '
           ACCEPT OPCION.
           MOVE ' ' TO FINAL-DE-DATOS.
           IF OPCION EQUAL 'A' OR EQUAL 'a'
              PERFORM PROCESO-ALTAS UNTIL FIN-DE-DATOS
           ELSE
              IF OPCION EQUAL 'B' OR EQUAL 'b'
                 PERFORM PROCESO-BAJAS UNTIL FIN-DE-DATOS
              ELSE
                 IF OPCION EQUAL 'C' OR EQUAL 'c'
```

```
                PERFORM PROCESO-CONSULTA UNTIL FIN-DE-DATOS
             ELSE
                IF OPCION EQUAL 'M' OR EQUAL 'm'
                   PERFORM PROCESO-MODIFICACIONES
                           UNTIL FIN-DE-DATOS
                ELSE
                   IF TERMINAR
                      DISPLAY 'FIN DE PROCESO'
                   ELSE
                      DISPLAY 'ERROR EN OPCION.'.
 FIN-PROCESO.
*
*    Cerrar el archivo relativo.
*
     CLOSE LISTA.
 PROCESO-ALTAS.
*
*    Proceso de altas al archivo relativo.
*
     MOVE 'S' TO FINAL-DE-DATOS.
     PERFORM LEE-DATOS.
 LEE-DATOS.
*
*    Lectura de datos para dar de alta.
*
     DISPLAY ' '
     DISPLAY 'CAPTURA DE INFORMACION (S/N) [S]?'.
     ACCEPT FINAL-DE-DATOS.
     IF FIN-DE-DATOS
        NEXT SENTENCE
     ELSE
        PERFORM PROCESA-ALTAS.
 PROCESA-ALTAS.
*
*    Se da de alta la informacion.
*
     PERFORM LEE-CLAVE.
     MOVE ' ' TO HAY-ERROR.
     READ LISTA RECORD
          INVALID KEY MOVE 'S' TO HAY-ERROR.
     IF ERROR-OK
        PERFORM LEE-DATOS-LISTA
        WRITE REGISTRO-LISTA INVALID KEY
              DISPLAY 'WRITE : REGISTRO EXISTENTE.'
     ELSE
        DISPLAY ' '
        DISPLAY 'REGISTRO EXISTENTE.'
        PERFORM ESCRIBE-DATOS-LISTA.
 LEE-DATOS-LISTA.
*
*    Informacion requerida para el archivo.
*
     DISPLAY ' '.
     DISPLAY 'NOMBRE ? '
     ACCEPT NOMBRE.
 LEE-CLAVE.
```

```
*
*    Se define la clave del archivo relativo, esta clave
* vale de 000 a 999.
*
     DISPLAY ' '
     DISPLAY 'NUMERO DE REGISTRO 9(03)'.
     ACCEPT R-CLAVE.
*
* Proceso de bajas.
*
 PROCESO-BAJAS.
     MOVE 'S' TO FINAL-DE-DATOS.
     PERFORM BORRA-DATOS UNTIL FIN-DE-DATOS.
 BORRA-DATOS.
*
*    Informacion necesaria para el proceso de bajas.
*
     DISPLAY ' '
     DISPLAY 'BORRADO DE INFORMACION (S/N) [S]? '.
     ACCEPT FINAL-DE-DATOS.
     IF FIN-DE-DATOS
        NEXT SENTENCE
     ELSE
        PERFORM BAJAS-DATOS-LISTA.
 BAJAS-DATOS-LISTA.
*
*    Se define la informacion para dar de baja, con control
* de informacion.
*
     PERFORM LEE-CLAVE.
     MOVE ' ' TO HAY-ERROR.
     READ LISTA RECORD
         INVALID KEY MOVE 'S' TO HAY-ERROR.
     IF ERROR-OK
        DISPLAY ' '
        DISPLAY 'NO EXISTE EL REGISTRO'
     ELSE
        PERFORM ESCRIBE-DATOS-LISTA
        PERFORM SE-BORRA.
 SE-BORRA.
*
*    Proceso de verificacion para el borrado de informacion.
*
     DISPLAY ' '.
     DISPLAY 'SE DESEA BORRAR INFORMACION (S/N) [N]?'.
     ACCEPT  OPCION.
     IF OPCION-SI
        DELETE LISTA RECORD
               INVALID KEY DISPLAY 'NO SE PUEDE BORRAR.'.
 ESCRIBE-DATOS-LISTA.
*
*    Desplegado de informacion que se encuentra en el archivo.
*
     DISPLAY ' '.
     DISPLAY 'INFORMACION DEL REGISTRO LISTA.'
     DISPLAY ' '.
```

```
           DISPLAY 'CLAVE  : ' R-CLAVE.
           DISPLAY 'NOMBRE : ' NOMBRE.
      *
      * Proceso de consulta del archivo relativo.
      *
       PROCESO-CONSULTA.
           MOVE 'S' TO FINAL-DE-DATOS.
           PERFORM CONSULTA-DATOS UNTIL FIN-DE-DATOS.
       CONSULTA-DATOS.
      *
      *    Consulta de informacion del archivo relativo.
      *
           DISPLAY ' '
           DISPLAY 'CONSULTA DE INFORMACION (S/N) [S]? '.
           ACCEPT FINAL-DE-DATOS.
           IF FIN-DE-DATOS
              NEXT SENTENCE
           ELSE
              PERFORM CONSULTA-DATOS-LISTA.
       CONSULTA-DATOS-LISTA.
      *
      *    Lectura de clave relativa y verificacion de informacion.
      *
           PERFORM LEE-CLAVE.
           MOVE ' ' TO HAY-ERROR.
           READ LISTA RECORD
                INVALID KEY MOVE 'S' TO HAY-ERROR.
           IF ERROR-OK
              DISPLAY ' '
              DISPLAY 'NO EXISTE EL REGISTRO'
           ELSE
              PERFORM ESCRIBE-DATOS-LISTA.
      *
      * Proceso de modificacion.
      *
       PROCESO-MODIFICACIONES.
           MOVE 'S' TO FINAL-DE-DATOS.
           PERFORM MODIFICA-DATOS UNTIL FIN-DE-DATOS.
       MODIFICA-DATOS.
      *
      *    Modificacion al archivo relativo.
      *
           DISPLAY ' '
           DISPLAY 'MODIFICACION DE INFORMACION (S/N) [S]?'.
           ACCEPT FINAL-DE-DATOS.
           IF FIN-DE-DATOS
              NEXT SENTENCE
           ELSE
              PERFORM MODIFICA-DATOS-LISTA.
       MODIFICA-DATOS-LISTA.
      *
      *    Definicion de clave e informacion a modificar.
      *
           PERFORM LEE-CLAVE.
           MOVE ' ' TO HAY-ERROR.
           READ LISTA RECORD
```

```
            INVALID KEY MOVE 'S' TO HAY-ERROR.
        IF ERROR-OK
           DISPLAY ' '
           DISPLAY 'NO EXISTE EL REGISTRO'
        ELSE
           PERFORM ESCRIBE-DATOS-LISTA
           PERFORM SE-CAMBIA.
     SE-CAMBIA.
     *
     *    Verificacion de modificacion de informacion.
     *
        DISPLAY ' '.
        DISPLAY 'SE CAMBIA LA INFORMACION (S/N) [N]?'.
        ACCEPT  OPCION.
        IF OPCION-SI
           PERFORM LEE-DATOS-LISTA
           REWRITE REGISTRO-LISTA
                   INVALID KEY DISPLAY 'NO SE PUEDE MODIFICAR'.
```