DIRECTORIO DE PROFESORES AL CURSO: LENGUAJE DE PROGRAMACION COBOL

DEL 6 DE SEPTIEMBRE AL 5 DE OCTUBRE DE 1985

1.- M. en I. JORGE ONTIVEROS JUNCO (COORDINADOR)
JEFE DEL DEPARTAMENTO DE PLANEACION
METODOS Y PROCEDIMIENTOS DEL CECAFI
FACULTAD DE INGENIERIA DE LA UNAM
MEXICO, D.F.
TEL. 550-52-15 EXT. 3734

2.- ING. VICTOR MANUEL LEYVA ALATRISTE
JEFE DEL DEPTO. DE INGENIERIA DE SISTEMAS
DIR. GRAL. DE CORREOS, S.C.T.
DIAGONAL SAN ANTONIO No. 1309-4°PISO
COL. NARVARTE
C.P. 03020
MEXICO, D.F.
TEL. 536-75-46

3.- ING. SOCRATES A. MUÑIZ ZAFRA
JEFE INTERINO DEL CENTRO DE CALCULO DE
LA FACULTAD DE INGENIERIA DE LA UNAM
MEXICO, D.F.
TEL. 550-57-34

4.- ING. GUSTAVO RODRIGUEZ ORTIZ
JEFE DEL DEPTO. DE SISTEMAS
DESARROLLO DE INGENIERIA, S.A.
PABLO DE LA LLAVE No. 110
COL. BOSQUES DE TETLAMEYA
MEXICO, D.F.
TEL.73-12-03

LENGUAJE DE PROGRAMACION COBOL 1985

VIERNES 6 DE SEPTIEMBRE JOJ Y SAMZ

Inauguración del Curso
Introducción a la Computación e Historia del PED
Partes de la Computadora

SABADO 7 DE SEPTIEMBRE SAMZ Y VMLA

Programación Estructurada
Introducción a la VAX
Práctica (teclear un ejemplo)

VIERNES 13 DE SEPTIEMBRE JOJ Y SAMZ

Historia del Cobol
Conceptos
Estructura de un Programa Cobol
Identification Division.

SABADO 14 DE SEPTIEMBRE JOJ Y GRO

Environment Division
Data Division
    Filler, campos A,X,9 y Niveles 01 y 02
Continuación de DCL y EDT.
    PRACTICA

VIERNES 20 DE SEPTIEMBRE VMLA Y GRO

Data Division y Procedure Division
    Working Storage Section, diseño de salida, value
    campos Z, signo y punto
    verbos
    párrafos
    Accept, Display y Move
    Redefines y Occurs

SABADO 21 DE SEPTIEMBRE VMLA Y GRO

    Open y Close
    Read y Write
    Operaciones aritméticas
    EJEMPLO
    PRACTICA

VIERNES 27 DE SEPTIEMBRE VMLA Y GRO

    IF
    Perform 1 y 2
    Read, Write y Rewrite

SABADO 28 DE SEPTIEMBRE VMLA Y GRO

SORT
Archivos secuenciales
opcionalmente SEARCH y archivos de acceso directo
PRACTICA

VIERNES 4 DE AGOSTO GRO, VMLA Y SAMZ

Desarrollo de Sistemas
Clausura Oficial del Curso

SABADO 5 DE AGOSTO GRO Y VMLA

Temas Diversos (dependiendo del avance del curso)
PRACTICA

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: LENGUAJE DE PROGRAMACION COBOL

FECHA: Del 6 de septiembre al 5 de octubre de 1985.

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	
ING. JORGE ONTIVEROS JUNCO					
ING. VICTOR MANUEL LEYVA ALATRISTE					
ING. SOCRATES ALBERTO MUÑIZ ZAFRA					
ING. GUSTAVO RODRIGUEZ ORTIZ					

ESCALA DE EVALUACION: 1 A 10

CURSO:

FECHA:

	TEMA	ORGANIZACION Y DESARROLLO DEL TEMA	GRADO DE PROFUNDIDAD LOGRADO EN EL TEMA	GRADO DE ACTUALIZACION LOGRADO EN EL TEMA	UTILIDAD PRACTICA DEL TEMA	
	Introducción a la Computadora e...					
	Programación Estructurada					
	Historía del Cobol					
	Environment Division					
	Data División y Procedure Division					
	Open y Close					
	Perform 1 y 2					
	Archivos secuenciales					
	Desarrollo de sistemas					
	Temas diversos					

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	

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?

☐ sí ☐ no

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?

_____

_____

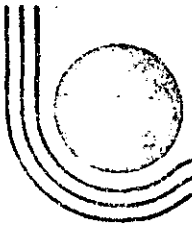
_____

10.- Otras sugerencias:

_____

_____

_____



APUNTES DEL CURSO : C O B O L

AGOSTO, 1985

COBOL

Common Business Oriented Lansuage

INTRODUCCION

El COBOL es un lenguaje que sirve para programar computadoras digitales, orientado al procesamiento de información para aplicaciones comerciales, diseñado especialmente para servir en la administración.

El lenguaje COBOL tiene las siguientes características:

Es un lenguaje de Alto Nivel: emplea terminos de la lengua inglesa, lo cual facilita la programación.

Es de Carácter Universal: la mayoría de las computadoras lo aceptan y es similar en todas ellas.

Es Autodocumentado: la extructura de los programas obligan a la documentación de los mismos.

Pertenece a los Lenguajes Compiladores: se requiere de un programa (compilador) que traduzca del lenguaje COBOL al lenguaje de máquina de la computadora.

CARACTERES

Los elementos básicos de cualquier lenguaje son los caracteres con los cuales éste se forma.

En el lenguaje COBOL los caracteres que se emplean son:

Caracteres Numericos : 0 a 9

Caracteres Alfabéticos: A a la Z

b (espacio en blanco)

Caracteres Especiales : -+/=$,;.:'*()[]

En algunos casos, en el lenguaje COBOL, ciertos caracteres tiene un significado especial, estos caracteres son:

CARACTERES DE EDICION:

B Inserción de espacios o blancos
Z Supresión de ceros
0 Inserción de ceros
+ Signo positivo
- Signo negativo
CR Crédito
DB Débito
* Protección de cheques
$ Signo de pesos
, Inserción de comas
. Inserción de puntos

CARACTERES DE PUNTUACION:

, Coma
; Punto y coma
: Dos puntos
. Punto
" Comillas
(Paréntesis izquierdo
) Paréntesis derecho
Ø Espacio o blanco
' Apostrofe
] Paréntesis cuadrado derecho
[Paréntesis cuadrado izquierdo
/ diagonal

OPERADORES ARITMETICOS

- Menos
+ Más
* Multiplicación
/ División
** Exponenciación

CARACTERES DE RELACION

> Mayor
< Menor
= Igual

PROGRAMACION

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

a) Los Resultado a obtenerse (Salidas).

b) Los Datos requeridos para obtener los resultados deseados (Entradas).

c) La forma en que se procesarán los datos (Algoritmos).

d) Los dispositivos externos al programa de los cuales se obtendrán los datos a procesarse (Dispositivos de entrada).

e) Los dispositivos externos al programa en los cuales se emitiran los resultados (Dispositivos de Salida).

f) Los dispositivos de almacenaminto de información, etcétera.

En la definición de la información que se procesará hay que especificar las caracteristicas de ésta, como es el tipo (alfabético, numérico, etc.), y el tamaño (cuantos caracteres máximo puede tener un dato, por ejemplo un nombre).

En todo programa existe cierta información cuyo contenido es definido dentro del programa (tales como los encabezados de los reportes que se emitirán) y otra información cuyo contenido (mas no sus caracteristicas) es proporcionado al momento de ejecución del programa (por ejemplo, los nombres de los empleados en la elaboración de una nomina). En este último caso los programas son practicamente independientes del contenido de la información a procesar.

ESTRUCTURA DE UN PROGRAMA

Todo programa en COBOL está estructurado en cuatro divisiones en donde se definen lo siguiente:

1) La identificación del programa

2) La asociación al programa de los dispositivos externos por los cuales se obtendran 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 antes de poder desarrollar algún programa.

DESCRIPCION DE LA INFORMACION

Para definir en un programa en COBOL, la información que se va a procesar, tanto de entrada, de salida, como intermedia, se emplean los conceptos Campos de Datos, los Registros y los Archivos.

CAMPOS DE DATOS

Los Campos de Datos definen áreas de almacenamiento de información en el programa. A cada campo de dato hay que definirle el número de caracteres que puede contener (longitud del campo) y el tipo de información que podrá almacenar (sea alfabético, numérico o alfanumérico).

Por ejemplo el ampo para almacenar el número de cuenta de un alumno de la UNAM, tendrá una longitud de 8 caracteres del tipo numérico.

NOMBRE-DE-DATO

Es el nombre asociado a un Campo de información, para hacerle referencia dentro de un programa en COBOL (ejemplo: NO-DE-CUENTA)

REGLAS PARA FORMAR LOS NOMBRES DE DATO

Las reglas para formar los datos de dato y en general para formar cualquier nombre que el programador tenga que definier dentro de un programa, son:

a) De 1 a 30 caracteres, elegidos de los siguientes:

   A a la Z (excepto Ñ) 0 al 9 - (guiones)

b) Deben empezar con un carácter alfabético (letra)

c) Los guiones (-) sólo podrán ir intermedios en el nombre.

   No pueden ir al principio ni al final del nombre.

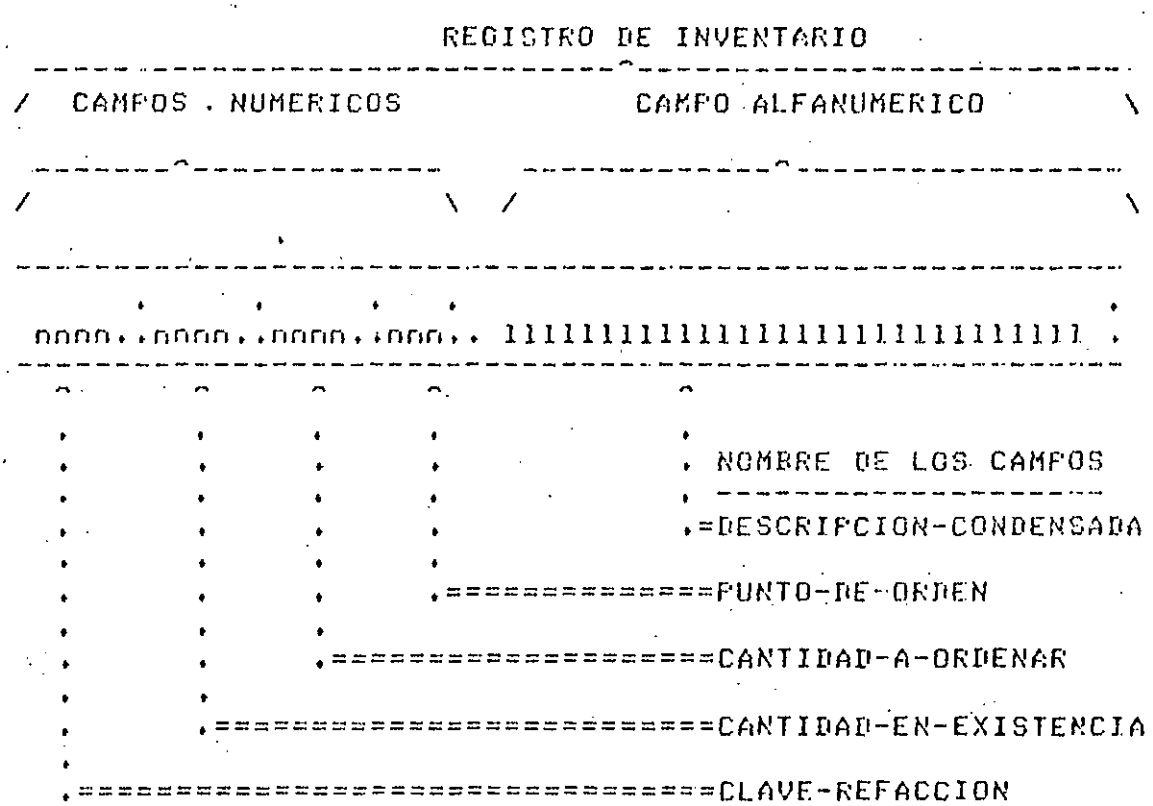
d) No se permiten espacios o blancos intermedios.

e) No deben emplearse palabras reservadas (ver más adelante) en la elaboración de los nombres.

REGISTRO

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

Ejemplo de un registro de inventario, identificado con el nombre de REGISTRO-DE-INVENTARIO:



Este ejemplo se codifica en lenguaje COBOL como sigue:

```
01  REGISTRO-DE-INVENTARIO.
    03 CLAVE-REFACCION          PIC 9 (4).
    03 CANTIDAD-EN-EXISTENCIA PIC 9 (4).
    03 CANTIDAD-A-ORDENAR      PIC 9 (4).
    03 PUNTO-DE-REORDEN        PIC 9 (3).
    03 DESCRIPCION-CONDENSADA PIC X (30).
```

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 perifericos (ejemplo: Lectora de Tarjetas, Impresoras, Cintas y Discos magnéticos, etc.).

A través de los Archivos se hace la Entrada (obtención de los datos a proceso) y la Salida (emisión de reportes y resultados) de la información de cualquier programa en COBOL.

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.

VERBOS.

Las oraciones están compuestas con Verbos, que como en todo lenguaje ordenan una acción. El COBOL proporciona una serie de Verbos, cuya sintaxis y función se encuentran ya definidos en el lenguaje (pertenecen a las palabras reservadas, ver mas adelante).

Algunos verbos operan con campos de información y otros definen el órden en que se ejecutarán las instrucciones. Con los Verbos se pueden hacer:

a) Operaciones Aritméticas

b) Movimientos de Información

c) Operaciones de Entrada y Salida

d) Comparaciines para tomar decisiones

e) Repeticiones de Instrucciones

f) Alteración del Flujo del Proceso

EJEMPLO:

MOVE CLAVE-ARTICULO TO CLAVE-SALIDA.

Esta instrucción indica que se mueva el contenido del Campo de Dato denominado CLAVE-ARTICULO al Campo de Dato denominado CLAVE-SALIDA.

ORACIONES.

Como se mencionó, las Oraciones representan las instrucciones del programa, y siempre comienzan con un Verbo. Las Oraciones se pueden o no terminar con un punto (Ver ejemplo anterior) pero por lo general se acostumbra terminarlas con punto. (Existen ciertos casos en donde no se deben terminar con punto, como se verá al describir el verbo IF).

PARRAFOS

En COBOL las Oraciones se agrupan en Párrafos. En otras palabras un párrafo es una agrupación de oraciones, terminadas con un punto, que se identifican con un nombre (Nombre de Párrafo).

Los párrafos permiten alterar el flujo de ejecución de las instrucciones o bien para ejecutar un grupo de instrucciones (una vez más veces) sin alterar al flujo de ejecución. Los párrafos se codifican en una sola línea y se terminan con un punto.

EJEMPLO:

```
INICIO.
     OPEN INPUT LECTORA OUTPUT IMPRESORA.
LECTORA.
     READ LECTORA AT END GOTO FIN.
     PERFORM PROCESO.
     GOTO LECTURA.
PROCESO.
FIN.
     CLOSE LECTORA IMPRESORA.
     STOP RUN.
```

TIPOS DE PALABRAS

En el lenguaje COBOL se emplean tres tipos de palabras, que son: Nombres o Sustantivos, Verbos y Palabras Reservadas.

NOMBRES O SUSTANTIVOS

Los nombres se dividen en dos grupos: los definidos por el programador y los proporcionados por el lenguaje COBOL (los cuales tienen ya una función y contenido definido).

Los nombres definidos por el programador se construyen con las mismas reglas empleadas para los Nombres de Datos. Estos se emplean para definir:

Nombres de Datos

Nombres de Registros

Nombres de Archivo

Nombres de Condición

Nombres de Procedimiento o Párrafo

Los nombres que proporciona el lenguaje COBOL son de:

Constantes figurativos

Registros especiales

PALABRAS RESERVADAS

Las palabras reservadas, son palabras en Inglés que tienen un significado ya determinado en el lenguaje COBOL. Estas palabras no se pueden emplear para otro fin diferente para el cual están ya definidas. En la siguiente hoja se proporciona una lista de las palabras reservadas del COBOL

REGLAS DE PUNTUACION

1. Toda palabra en COBOL se limita a la derecha por lo menos un espacio en blanco.

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

HOJA DE CODIFICACION

La forma inicial y la mas común de sumistrar un programa en COBOL a la computadora es empleando tarjetas perforadas (ACTUALMENTE EN DESUSO), las cuales están compuestas de 80 columnas y 12 renglones.

La codificación de los programas en COBOL requieren de un formato ya establecido, el cual se encuentra detallado en estas hojas.

Las hojas de Codificación están compuestas de 80 columnas y 25 renglones (o sea que representan hasta 25 líneas de código) y tienen el siguiente formato:

COLUMNAS	CONTENIDO
1-3	Número de página u hoja. Cada hoja se enumera de 1 en 1 comenzando en 1.
4-6	Número de renglón. Para cada hoja éste se inicia en 10 y se asciende en rangos de 10. El número de página y el número de renglón representan la secuencia de las líneas de código del programa. Ambas secuencias son opcionales.
7	Columna de indicaciones al compilador COBOL. - (guión) Continuación de títulos que no caben en un solo renglón. * (asterisco) Para poner comentarios al programa.
8	Margen A. El COBOL exige que ciertas partes de la codificación principien en esta columna, como son los nombres de las Divisiones, los párrafos, las secciones, etc.
12	Margen B. Los instrumentos (oraciones) y ciertas declaraciones de la información principian en este margen.
72	Hasta esta columna se puede codificar las declaraciones e instrucciones del programa.
73-80	Identificación del Programa. Es el nombre que el programador le asigna al programa, el cual puede estar formado de 1 a 8 caracteres. Esta identificación es opcional.

NOMENCLATURA

Un manual de Instrucciones para el lenguaje COBOL muestra al lector las características del lenguaje.

Este texto maneja ciertas reglas de interpretación las cuales son:

a) Palabras Mayúsculas Resaltadas: Deberán usarse como están escritas si queremos usar la función.

b) Palabras Mayúsculas no Resaltadas: Pueden usarse si se desea, sin embargo se emplean, para que las declaraciones que escribamos resulten más claras.

c) Palabras Escritas con Minúsculas: Son términos genéricos que indican la clase de palabra que al programador deberá suministrar en la posición mostrada en el formato.

d) Llaves: Cuando hay palabras o frases encerradas entre llaves {} habrá que escoger alguna de ellas.

e) Corchetes: Las palabras encerradas entre [ ] representan opciones.

Lo que contenga puede incluirse en la declaración, si se desea la opción, pero puede omitirse en caso contrario.

f) Puntos Suspensivos (...) : Indican que se puede repetir el concepto que les precede.

NOTA. Las comas (,) y los puntos y comas (;) pueden omitirse en las declaraciones, no así los puntos (.)

EJEMPLOS:

```
MULTIPLY Literal-1      BY identificador-2 [ROUNDED]
         identificador-1
    identificador-3  [ROUNDED]
[, identificador-4  [ROUNDED] ]...
```

DIVISION DE LOS PROGRAMAS EN COBOL

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

IDENTIFICATION DIVISION

ENVIRONMENT DIVISION

DATA DIVISION

PROCEDURE DIVISION

IDENTIFICATION DIVISION (DIVISION DE IDENTIFICACION)

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

ENVIRONMENT DIVISION (DIVISION DE EQUIPO)

Sirve para

- Especificar las computadoras que se utilizan, tanto para la compilación del programa, como para su ejecución.
- Asignar nombre a los dispositivos de Entrada/Salida que se emplearán en el programa.
- Definir algunas características de los archivos. (Secuencial, Random, etc.)
- Para darles un nombre a ciertas funciones de algunos dispositivos
  (ejemplo: Salto de hoja en una impresora).

DATA DIVISION (DIVISION DE LOS DATOS)

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

- Descripción de las características de los archivos de Entrada/Salida.
  - Etiquetas
  - Atributos
  - Nombres de los Registros
- Descripción de los Registros de Entrada/Salida
  - Campos en los que se dividen los registros
  - Características de los Campos (tipo y tamaño)
- Descripción de Registros o Campos Auxiliares, como son:
  - Encabezados
  - Contadores
  - Registros intermedios

PROCEDURE DIVISION (DIVISION DEL PROCESO)

En esta división se define la forma en que se procesará la información que manejará el programa. La División está constituida por oraciones agrupadas por párrafos. La ejecución del programa principia desde el primer párrafo de esta división.

1. IDENTIFICATION DIVISION.

Consta de unas cuantas líneas que identifican el programa:

Nombre del programa, autor, instalación utilizada, fecha de creación y descripción del programa.

La estructura de esta división es como sigue:

```
IDENTIFICATION DIVISION.
  PROGRAM-ID.      Comentario.
 [INSTALLATION.    Comentario.]
 [DATE-WRITTEN.    Comentario.]
 [DATE-COMPILED.   Comentario.]
 [SECURITY.        Comentario.]
 [REMARKS.         Comentario.]
```

2. ENVIRONMENT DIVISION.

Es la segunda división de un programa en COBOL. Se emplea para definir los dispositivos periféricos que se van a utilizar, asignándoles un nombre de archivo para su manejo. También para definir ciertas características de los archivos, así como para darles nombres a ciertas funciones de algunos dispositivos. La estructura de esta división es como sigue:

```
ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.

[SOURCE-COMPUTER. Comentario.]

[OBJECT-COMPUTER. Comentario.]

[SPECIAL-NAMES.   Nombres-Especiales.] ]

[INPUT-OUTPUT SECTION.

[FILE-CONTROL.

        FILE-CONTROL-DESCRIPTION  ...] ]
```

DESCRIPCION:

NOMBRES ESPECIALES:

Control-de-dispositivo-periférico IS Nombre-de-dato

Ejemplo:

```
C01 IS Salta-hoja.
```

FILE-CONTROL-DESCRIPTION.

```
SELECT   Nombre-de-archivo   ASSIGN   TO
           Dispositivo-periferico.
```

3. DATA DIVISION

La tercera parte de un programa COBOL es la DATA DIVISION, permite la descripción de las características de los archivos de entrada s/o salida, tales características son:

- ETIQUETAS
- TAMAÑO DEL BLOQUE
- TAMAÑO DE REGISTROS
- NOMBRE, TAMAÑO Y TIPO DE LOS CAMPOS EN QUE ESTA DIVIDIDO UN REGISTRO
- DESCRIPCION DE REGISTROS O CAMPOS AUXILIARES COMO SON: CONTADORES, ENCABEZADOS, REGISTROS INTERMEDIOS, ETC.

Esta compuesto de varias secciones, las más importantes son:

FILE SECTION Y WORKING-STORAGE SECTION.

La estructura es la siguiente:

```
DATA DIVISION.
[FILE SECTION.
        [FILE-DESCRIPTION
        [RECORD-DESCRIPTION]  ...]  ...]

[WORKING-STORAGE SECTION.
        [DESCRIPCION-DATO ELEMENTAL]
        [RECORD-AUXILIAR-DESCRIPTION]  ...]
```

DESCRIPCION:

FILE-DESCRIPTION:

```
FD   NOMBRE-DE-ARCHIVO

                        {OMITTED}
     LABEL RECORDS
                        {STANDARD}

     DATA RECORD IS     nombre-de-resistro
```

RECORD-DESCRIPTION:

```
                    {Nombre-de-dato}     {Picture}     {A}
Número-de-nivel     {FILLER         }                  {X}
                    {F              }    {PIC     }    {9}

                                                 {Literal  }
                                          VALUE
 0                                               {Constante}
```

A - Alfabéticos
X - Alfanuméricos
9 - Numéricos

La cláusula VALUE solo puede emplearse en la WORKING-STORAGE SECTION.

4. PROCEDURE DIVISION

```
PROCEDURE DIVISION.

[[nombre-de-sección SECTION.]

  nombre-de-parrafo.

[[nombre-de-parrafo.] ... [instrucción.] ...] ...] ...
```

LITERALES

Una 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.

LITERALES NUMERICAS.

Una literal Numérica, es un elemento formado con los dígitos 0 al 9, los signos + ó - y un punto decimal. De hecho es una cantidad numérica
ejemplo:

```
+601     -740     +38.35     .005
 601        0      74.78      643.
```

El número máximo de dígitos de una literal numérica es de 23.

LITERALES NO NUMERICAS

Una 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 numérica, los constituyen los caracteres dentro de las comillas. Cualquier par de comillas ('') representan una simple comilla, lo que permite incluirlas en las literales no numéricas. El número de caracteres no debe exceder los 256 caracteres.
Ejemplo:

```
'JOSE JOSE'     '357'     'BC.D'     $A7-$'

'B. W. ''BILL''  JONES '     *** HOLA ***
```

CONSTANTES FIGURATIVAS

Una constante figurativa es un elemento cuyo valor se encuentra definido en el lenguaje COBOL. Estas constantes se pueden emplear en cualquier parte donde se puede emplear una literal.

TIPOS:

A. ZERO, ZEROS, ZEROES

Representan el valor de cero, o uno ó más caracteres o (cero), dependiendo para lo que se le emplee.

B. SPACE, SPACES

Representan uno ó más espacios en blanco

C. QUOTE, QUOTES

Representan uno ó más del caracter " (comilla)

D. ALL literal-no-numérica

Representa la repetición de la literal no numérica, tantas veces como sea necesaria.

ejemplo:

ALL ABC = ABCABC ó ABCAB ó AB

Ejemplos:

```
MOVE SPACE TO LINEA
MULTIPLY  CONTADOR BY ZERO
IF CLAVE = ALL "*"
05 NUMERO-DE-HOJA  PIC 99 VALUE ZEROS.
```

CLAUSULA PICTURE

{PICTURE } CARACTERES PICTURE

{ PIC}

Sirve para indicar el tamaño (de caracteres) y el tipo (numérico, alfabético ó alfanumérico) de un campo. Si el campo es numérico, sirve para indicar si tiene signo (+ ó -) y si tiene decimal. En la WORKING-STORAGE SECTION permite editar la información para su impresión ($, *, Ø, o, etc).

CARACTERES PICTURE:

C(n) Indica que el caracteres 'C' se repite n veces (ejemplo: 9(5) = 99999).

9 Indica campo numérico
A Indica campo alfabético
X Indica campo alfanumérico
V Indica colocación de punto virtual (no se contabiliza en el tamaño del campo)
S Indica existencia de signo. (No se contabiliza en el tamaño del campo) .
P Sirve para indicar que el punto decimal está fuera del campo. (No se contabiliza en el tamaño del campo) (max. 18 en 1130)???

ejemplo:

PIC	TAMAÑO	CONTENIDO EN MEMORIA	VALOR
VPPP999	3	123	0.000123

CARACTERES PICTURE DE EDICION (sólo WORKING-STORAGE)

Z Indica supresión de ceros no sisnificativos en un campo numérico. Se contabiliza en el tamaño del campo (se colocan a la izquierda) ejemplo:

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
01234	ZZZZZ	Ø1234
00123	Z(4)9(3)	ØØØØ123
0000	Z(4)	ØØØØ

$ Indica que se debe poner un signo de $ en el campo numérico editado. Se coloca a la izquierda y se contabiliza en el tamaño del campo.

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	$9999	$1234
0023	$$99	Ø$23
23.50	$(2)ZZZ9.99	Ø$ØØ23.50
12	$(4)	Ø$12

- (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 debe colocar a la izquierda y se contabiliza en el tamaño del campo. Ejemplos:

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
-1234	---99	-1234
-5	-(3)9	ØØ-5
-5	-ZZ9	-ØØ5
-12	----	Ø-12

+ Funciona igual que el signo negativo. Si el valor del campo es negativo se inserta un signo menos.

0 (cero) indica que se coloquen caracteres ceros, sin que se pierdan caracteres del dato fuente. Se contabiliza en el tamaño del campo. Ejemplos:

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	990(2)99	120034
1234	9(4)0(4)	12340000
0012	Z(4)000	ØØ12000

B Indica que se coloquen espacios en blanco, sin que se pierdan caracteres del campo fuente. Ejemplos:

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	99BB99	12ØØ34
1234	9(4)B(2)	1234ØØ

, (Coma) indica que se coloquen comas en el lugar correspondiente, sin que se pierdan caracteres del campo fuente.

. (Punto) indica que se coloque un punto decimal, tomando en cuenta la colocación del punto virtual.
Sólo puede utilizarse en punto decimal.

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	$ZZ,Z(3).99	Ø$1,234.00
-1.23	++9.9(3)	Ø1.230

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

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	***9	1234
1234	*(3)4(9)	***1234
12.00	$*****.99	$***12.00

CR y DB Indica que se coloquen este par de letras a la derecha del campo, siempre que el valor sea negativo, sino dejará los espacios en blanco.

CAMPO FUENTE	PICTURE	RESULTADO IMPRESO
1234	9(4)CR	1234
-1234	9(4)DB	1234DB
-1234	9(4)CR	1234CR

VERBO MOVE.

```
        { registro-especial }
        { atributo          }
MOVE    { campo-de-dato-1   } TO campo-d-2 [,campo-d-3 ] ...
        { literal-1         }
```

instrucción muy común en el programa fuente, consiste en mover el contenido de campo-de-dato-1 o literal-1, al campo-de-dato-2, sin alterar el contenido del campo-de-dato-1.

Ejemplo:

```
MOVE  RFC  TO  RFC-SAL.
MOVE  ZERO TO  XX,YY,ZZ.
MOVE 'XYZ' TO  NOMBRE.
```

La operación MOVE depende del tipo de dato que se está moviendo.

MOVE ALFANUMERICO:

```
MOVE  --------------  TO ---------------
        campo emisor       campo destino
```

: toma el primer caracter izquierdo de la variable emisora y lo lleva a la primera posición de la variable destino; y así sucesivamente todos los demás caracteres, hasta que alguno se termine. Si el emisor tiene un tamaño menor al del destino, los campos faltantes de este último se llenan con blancos.

MOVE NUMERICO:

Campo emisor ._._._._._.

Campo destino .0._._._._.

: toma el primer caracter apartir del punto y mueve de derecha a izquierda.

Por lo tanto se concluye que si:

C. Emisor > C. Destino
- Se pierden cifras más significativas.

C. Emisor < C. Destino
- Llena de Ceros.

NOTA: Si el punto no está especificado lo toma por default como primera posición de la izquierda.

Ejemplo:

C. Receptor definido como 999V99

5 4 3 2 1 6 7 8 9 0

C. Emisor C. Receptor 3 2 1 0 0

C. Receptor después del movimiento.

C. Emisor definido como 99V999 2 1 0 0 0

C. Receptor definido como 999999 5 4 3 2 1

1 0 0 1 6 2 1 3 7 4 9 8 C. Receptor después del mov.

Emisor Receptor 0 0 0 0 1 0

Existen situaciones en las cuales hay que mover la mayor parte del contenido de un registro a campos de datos del mismo nombre en otro registro, siendo gravoso escribir todos los MOVE necesarios para tales casos.

La opción CORRESPONDING del verbo MOVE hace esto innecesario.

MOVE CORRESPONDING nombre-dato-1 TO nombre-dato-2.

Características:

- Ambos nombres de dato deben corresponder a niveles de grupo.
- Se mueven todos los campos dentro de nombre-dato-1 que tengan el mismo nombre que otro campo dentro de nombre-dato-2, tal y como si hubiéramos escrito una serie de MOVE sencillos.

Ejemplo:

MOVE A in Emisor TO A in Receptor

MOVE E in Emisor TO E in Receptor

MOVE C in Emisor to C in Receptor

MOVE D in Emisor to D in Receptor

MOVE F in Emisor to F in Receptor

A B C D E F G H Emisor

A D E B F G H C Receptor

Considerando que Emisor y Receptor son nombres de registros y tienen el formato mostrado; el efecto total de

MOVE CORRESPONDING Emisor TO Receptor

sería exactamente el mismo.

VERBO ADD.

```
     { identificador-1}  [ { identificador-2  }]
ADD  { literal-1       }  [,{ literal-2       }]   ...   TO

     identificador-m  [ROUNDED ]

   [, identificador-n  ROUNDED] ...

     ; ON SIZE ERROR instrucción ELSE instrucción
```

NOTA: El otro formato de la proposición ADD es exactamente igual, sólo cambiando la palabra TO por la palabra GIVING.

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

ADD

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:

El resultado de la suma reemplazará el contenido original del campo que le sigue a la palabra TO.

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, perderán los dígitos a menos que se use seguido de la palabra ROUNDED (que redondeará la cifra al espacio especificado).

Ejemplo:

ADD Area TO Manejo ROUNDED

VERBO MULTIPLY.

```
          {nombre-dato-1  }       {  nombre-de-dato-2}
MULTIPLY  { literal-1      } BY   { literal-2          }

[GIVING nombre-de-dato-3] [ROUNDED]

[; ON SIZE ERROR instrucción [ELSE instrucción]]
```

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

MULTIPLY.

Este verbo aritmético multiplica dos cantidades y hace una segunda o una tercera variable igual al producto.

Ejemplo:

```
MULTIPLY Piezas BY Costo GIVING Precio
         PIEZAS    COSTO     PRECIO
ANTES      23        147      4444444
DESPUES    23        147      0003381
```

VERBO DIVIDE.

```
       {  literal-1       }       {  literal-2       }
DIVIDE { identificador-1 } INTO { identificador-2 }

[GIVING identificador-3] [ROUNDED] [REMAINDER

                identificador-4 ROUNDED]

[;ON SIZE ERROR instrucción [ ELSE instrucción ]]
```

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 GO TO RUTINA-4

(Divide B entre A, dando C redondeado, en caso de error

de tamaño ir a rutina-4).
```

VERBO EXAMINE.

Formato 1:

```
                                                UNTIL FIRST
        EXAMINE nom-campo-datos TALLYING        ALL
                                                LEADING

                literal-1 [ REPLACING BY literal-2 ]
```

Formato 2:

```
                                                ALL
        EXAMINE nom-campo-datos REPLACING       LEADING
                                                UNTIL FIRST

                literal-1 BY literal-2
```

TALLYING , - CONTABILIZANDO EN TALLY
(campo numérico de PIC 99999).
ALL - TODOS
LEADING - AL PRINCIPIO (ENCABEZANDO)
UNTIL FIRST - HASTA EL PRIMER

REGLAS.

1. nom-campo-datos: Numérico, Alfabético o Alfanumérico
Con USAGE DISPLAY
(IMPLICITO O EXPLICITO).

2. literal-1 y literal-2: Debe de consistir de un solo caracter
(ejemplos: 1,"X") y debe de ser congruente con la clase del nombre-campo-datos.
Puede ser una constante figurativa (excepto ALL).

EXAMINE examina un campo de datos de izquierda a derecha del campo, realizando cualquiera de lo siguiente:

1. Contando el número de veces que aparece un caracter específico.
2. Reemplazando un caracter por otro.
3. Contando y reemplazando (incisos 1 y 2).

NOTA:

Para el conteo proporciona una variable llamada TALLY que puede contener valores de 0 a 99999. No debe de ser definida por el programa pero sí se puede utilizar.

EJEMPLOS	CAMPO-X ANTES	DESPUES	TALLY
EXAMINE CAMPO-X TALLYING ALL 1	101213	=	3
	230115	=	2
EXAMINE CAMPO-X TALLYING LEADING 1	121014	=	1
	402110	=	0
	111011	=	3
	111111	=	6
EXAMINE CAMPO-X TALLYING UNTIL FIRST 1	101315	=	0
	473110	=	3
EXAMINE CAMPO-X TALLYING ALL "A"	CASITA	=	2
EXAMINE CAMPO-X TALLYING LEADING SPACE	MATUYO	=	2
EXAMINE CAMPO-X TALLYING UNTIL FIRST ZERO	MAYO10	=	5

EXAMINE CAMPO-X REPLACING ALL 1 BY 0	121314	020304	NO
	783112	783002	USADO
EXAMINE CAMPO-X REPLACING LEADING 1 BY 0	131415	031415	*
	420117	420117	*
	111711	000711	*
	111111	000000	*
EXAMINE CAMPO-X REPLACING UNTIL FIRST 1 BY 0	123014	123014	*
	234156	000156	*
	743826	000000	*
EXAMINE CAMPO-X REPLACING FIRST 1 BY 0	123014	023014	*
	234156	234056	*
	743826	743526	*
EXAMINE CAMPO-X REPLACING SPACE BY ZERO	ØØ354	003540	*
EXAMINE CAMPO-X REPLACING "Ø" BY 0	Ø73Ø4Ø	073040	*
SON IGUALES			
EXAMINE CAMPO-X REPLACING "-" BY 0	7348-4	734804	*

6. Más de una redefinición a una área original es permitida. Las nuevas redefiniciones a una área original deberán hacerse a esta última únicamente (o sea que el nombre-de-dato-2 deberá ser el mismo en todas las nuevas redefiniciones del área que representa el nombre-de-dato-2).

7. La cláusula VALUE sólo es permitida en el campo que originalmente define el área y no es permitida en los campos que redefinen al campo en cuestión. (Con excepción de los niveles 88).

8. Un campo con nivel 01 que emplea la cláusula OCCURS no debe contener la cláusula REDEFINES.

EJEMPLOS:

```
01  REGISTRO-1.
    02  PARTE-UNO              PIC X(60).
    02  PARTE-DOS REDEFINES PARTE-UNO.
        03  XX                 PIC X(40).
        03  YY                 PIC 9(20).
    02  PARTE-TRES REDEFINES PARTE-UNO.
        03  ZZ                 PIC A(55).
        03  FILLER             PIC X(5).
01  TIPO-DE-ARTICULOS.
        03  ARTICULO-1         PIC XX VALUE "A"
        03  ARTICULO-2         PIC XX VALUE "B"
        03  ARTICULO-3         PIC XX VALUE "9"
01  TABLA-ARTICULOS REDEFINES TIPO-DE-AN
        03  ARTICULOS          PIC XX OCCURRS 3
```

CLAUSULA OCCURS

Esta Cláusula permite definir tablas. Una tabla es un conjunto de campos de información, donde cada campo tiene las mismas características (tamaño y formato).

El formato de esta Cláusula tienen dos formas:

FORMA 1:

```
{OC    }
{OCCURS}  entero-2  TIMES

[[{ASCENDING  }
[[{DESCENDING }  KEY IS nom-dato-2 [, nom-dato-3] ... ] ...]

[INDEXED BY  nombre-indice-1  [,nombre-indice-2] ...]
```

FORMA 2:

```
{OC     }
{OCCURS }  entero-1 TO  entero-2  TIMES

        [DEPENDING ON nombre-dato-1]

[ [ {ASCENDING }
[ [ {DESCENDING} KEY IS nom-dato-2 [,nom-dato-3] ...] ...]

[INDEXED BY  nombre-indice-1  [,nombre-indice-2 ] ...]
```

REGLAS:

1. El entero-1 y el entero-2 tienen que ser números enteros positivos.

2. El entero-1 debe ser menor que el entero-2

3. El contenido del nombre-dato-1 debe contener un valor entero positivo; este define el tamaño máximo de la tabla, pero no debe exceder el valor del entero-2.

4. El nombre de dato-2 puede ser el nombre del campo que contiene la cláusula OCCURS ó un campo en que se divide el primero.

5. El nombre-dato-3, etc. sólo puede ser un campo en que se divide el campo que emple la cláusula OCCURS.

6. La cláusula OCCURS no puede emplearse en campos que tengan los siguientes números de nivel: 77,66 y 88.

7. Para hacer referencia a un campo que emplea la cláusula OCCURS y a los campos en que se divida este primero, se hace utilizando indices (enteros, nombres de datos o nombres de indice). Los indices se encierran entre parentesis y deben seguir al nombre del campo.

8. La cláusula OCCURS se puede emplear en cualquier sección de la DATA DIVISION.

9. Los indices no deben utilizarse al emplear el verbo SEARCH

10. En la forma-1 el valor del entero-2 determina el tamaño maximo de la tabla.

11. En la forma-2, el valor del entero-1 define el tamaño minimo de la tabla y el entero-2 define el tamaño máximo de la tabla. El tamaño exacto de la tabla lo determina el valor del contenido del nombre-de-dato-1.

12. La opción KEY IS sirve para indicar que la tabla se encuentra organizada ascendente ó Descendente con respecto a los nombres de dato 2,3,etc. Estos se escriben en orden descendente de acuerdo a su prioridad.

13. La opción INDEXED BY sirve para asociarle uno ó más indices a la tabla. Estas dos últimas cláusulas son requeridas para emplear el verbo SEARCH.

EJEMPLOS	COBOL	CAMPO-X ANTES	DESPUES	Tally
		101213		= 3
	EXAMINE CAMPO-X TALLYING ALL 1	230115		= 2
		121014		= 1
		412110		= 0
	EXAMINE CAMPO-X TALLYING	111011		= 3
	LEADING 1	111111		= 6
	EXAMINE CAMPO-X TALLYING UNTIL	101315		= 0
	FIRST 1	473110		= 3
	EXAMINE CAMPO-X TALLYING ALL "A"	CASITA		= 2
	EXAMINE CAMPO-X TALLYING LEADING SPACE	##TUYO		= 2

EXAMINE CAMPO-X TALLYING UNTIL FIRST ZERO	MAYO 10		= 5
EXAMINE CAMPO-X REPLACING ALL 1 BY 0	121314 783112	020304 783002	NO USADO
EXAMINE CAMPO-X REPLACING LEADING 1 BY 0	131415 420117 111711 111111	031415 420117 000711 000000	
EXAMINE CAMPO-X REPLACING UNTIL FIRST 1 BY 0	123014 234156 743826	123014 000156 000000	
EXAMINE CAMPO-X REPLACING FIRST 1 BY 0	123014 234156 743826	023014 234056 743826	
EXAMINE CAMPO-X REPLACING SPACE BY ZERO	ØØ3540	003540	
EXAMINE CAMPO-X REPLACING *Ø BY 0 SON IGUALES	Ø73Ø4Ø	073040	
EXAMINE CAMPO-X REPLACING "-" BY 0	7348-4	734804	

LENGUAJE DE PROGRAMACION COBOL

CONCEPTO DE UNA COMPUTADORA

SEPTIEMBRE, 1985

UNAM

CONCEPTO DE UNA COMPUTADORA*

FACULTAD DE INGENIERIA

* TOMADO DE: "QUE HACEN LAS COMPUTADORAS Y COMO LO HACEN"

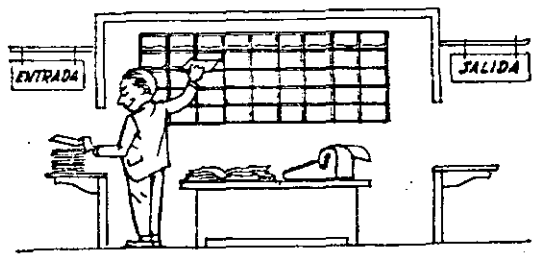
POR: I B M, Argentina.

C O N C E P T O D E C O M P U T A D O R A

El objeto de esta breve reseña sobre las computadoras electrónicas y sus múltiples aplicaciones al servicio del hombre, es transmitir al lector una completa visión de conjunto, mediante un lenguaje sencillo que permita comprender conceptualmente los temas tratados, sin necesidad de conocimientos previos en la materia.

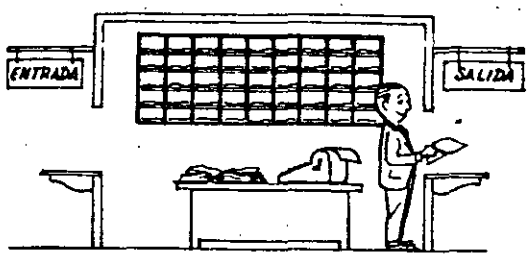
Esperamos que estas páginas, muy simples en apariencia pero con profundo contenido, permitan, a quienes las lean, ingresar al maravilloso mundo de las máquinas automáticas.

Este señor se llama Control. Trabaja en una pequeña habitación. Tiene a su disposición una máquina de calcular que suma, resta, multiplica y divide. Tiene también el señor Control un archivo parecido al casillero que existe en los trenes para clasificación postal.
Hay, además, en la habitación, dos ventanillas identificadas con sendos carteles: "Entrada" y "Salida".
El señor Control tiene un manual que le indica cómo debe desenvolverse con estos elementos, si alguien le pide que haga un trabajo.

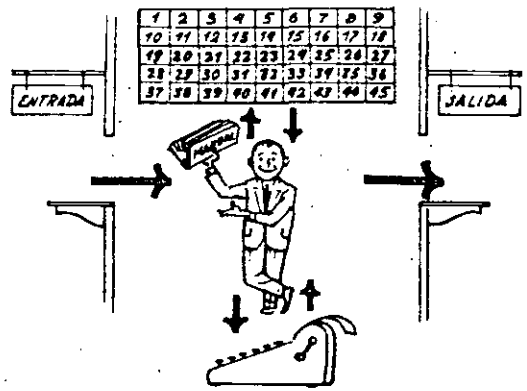


Una persona quiere saber el resultado de un complicado cálculo. Para ello, escribe ordenada, precisa y detalladamente, cada una de las operaciones que, en conjunto, integran ese cálculo, anota cada instrucción elemental en una hoja de papel y coloca todas las hojas en orden en la ventanilla "Entrada".
El señor Control, al ver las hojas, lee en su manual que debe tomar esas hojas con instrucciones, una por una, y colocarlas correlativamente en su archivo. Y así lo hace.

Una vez ubicadas todas las instrucciones en el archivo, el señor Control consulta nuevamente el manual. Allí se le indica que, a continuación, debe tomar la instrucción de la casilla 1 y ejecutarla, luego la de la casilla 2 y ejecutarla, y así sucesivamente hasta ejecutar la última instrucción. Algunas instrucciones indicarán que hay que sumar una cantidad a otra (instrucciones aritméticas); otras, que el señor Control debe ir a la ventanilla "Entrada" para buscar algún dato que intervenga en el cálculo -- (instrucciones de "entrada/salida"), dato que la persona que le formuló el problema habrá colocado ya en dicha ventanilla, en otra hoja de papel.
Finalmente, otras instrucciones indicarán que debe elegirse una de entre dos alternativas (instrucciones lógicas): por ejemplo, supongamos que una parte del cálculo - desde la instrucción que está en la casilla 5 del archivo hasta la que está en la casilla 9 debe ejecutarse 15 veces porque el cálculo así lo exige.
En tal caso, la instrucción que está en la casilla 10 indicará que, si los pasos 5 a 9 se han ejecutado menos de 15 veces, se debe volver al paso 5. Cuando se hayan realizado las 15 repeticiones y no antes, el señor Control seguirá con la instrucción de la casilla 11.



Después de ejecutar todas las instrucciones del archivo, haciendo con la máquina de calcular las operaciones en ellas indicadas, el señor Control entrega, a través de la ventanilla "Salida", los resultados obtenidos . . . y se sienta a esperar un nuevo trabajo.



Obsérvese que la actuación del señor Control es puramente mecánica: sólo sigue las indicaciones de su manual y cumple de acuerdo con ellas, las instrucciones que recibe a través de la ventanilla "Entrada" . Toma decisiones, pero solamente cuando se le señalan las alternativas que existen y con qué criterio debe elegir una de ellas.

El señor Control puede resolvernos cualquier problema, por complicado que éste sea. Pero para ello debemos indicarle paso a paso, en la forma más elemental y detallada, todo lo que debe hacer para resolverlo, sin olvidarnos absolutamente nada porque, en ese caso, el señor Control no sabría continuar por si mismo.

Haga el lector la prueba de formular un problema cualquiera de modo tal que una persona que no conozca nada acerca de ese problema, pueda resolverlo sin necesidad de hacer consultas. Verá que es una experiencia interesantísima.

El esquema que acabamos de representar mediante el señor Control y sus elementos de trabajo, corresponde exactamente al esquema de funcionamiento de una computadora electrónica.

A continuación presentaremos una breve descripción de los elementos de la computadora que corresponden a los elementos de trabajo del señor Control.

Las unidades de Entrada (representadas por la ventanilla "Entrada") : -

Son en la computadora, dispositivos capaces de leer información (Instrucciones o Datos) con el objeto de procesarla. Existen una gran variedad de elementos de entrada, entre los cuales tenemos:

Tarjetas de Cartulina y Cintas de Papel: Que son perforadas de manera que cada perforación representa un número, una letra ó un símbolo especial de acuerdo con un código predeterminado.

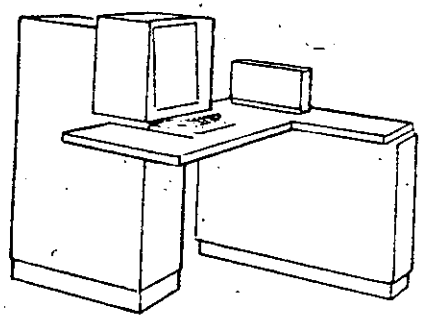
Cintas magnéticas: Conocidas como "memorias externas" tienen la ventaja de permitir almacenar la información en forma mas concentrada (a razón de 80 a 2400 caracteres por pulgada de longitud) y de ser más veloces, ya que pueden enviar o recibir información a la unidad de control a velocidades que van de 10,000 a 680,000 caracteres por segundo. Pueden llegar a tener hasta 730 m. de longitud.

Disco Magnético: También conocidos como "Memoria externa", en general tienen un diámetro aproximado de 30 cm. y pueden grabar hasta 400,000 letras, números, y caracteres especiales, formando palabras, cifras, ó registros completos. Se pueden grabar o leer a razón de 77,000 a 312,000 caracteres por segundo y su tiempo de acceso a un registro alcanza un promedio de 60 mili-segundos.

Una diferencia importante entre las cintas y los discos es la siguiente:
En las cintas los registros se graban o leen secuencialmente.
En los discos se tiene "Libre Acceso" a un registro cualquiera, en forma inmediata, pues cada registro se localiza por su posición física dentro del disco.

Lectora Optica de Caracteres Impresos: Puede leer un documento impreso por una máquina de escribir, o por una máquina de contabilidad o por la impresora de una computadora, a una velocidad de 30,000 caracteres por minuto.

Unidad de Representación Visual: Esta unidad de entrada/salida sirve para hacer consultas a la computadora, por medio de un teclado de máquina de escribir, y obtener la respuesta reflejada en una pequeña pantalla de televisión.
La imagen está formada por hasta 12 renglones de hasta 80 caracteres (letras, numeros, ó signos especiales) cada uno.



Vemos aquí otra Unidad de Representación Visual, más evolucionada que la anterior, la comunicación hombre-máquina puede establecerse en ella por medio de gráficas, es decir que la entrada y la salida de datos se hacen por medio de imágenes.

Cuenta esta unidad para ello con un dispositivo con forma de lápiz, que tiene en su punta una célula fotoeléctrica. Un delgado haz de luz parte en determinado momento de un punto de la pantalla y la recorre en forma de zig-zag. Si se apoya el "lápiz" en cualquier posición de la pantalla, su célula fotoeléctrica detectará en algún momento el haz de luz.

Por el tiempo transcurrido desde que el haz de luz comenzó su "barrido" hasta que fue detectado, la computadora determina en qué punto de la pantalla se encuentra apoyado el "lápiz" .

Como el barrido dura una fracción de segundo y se realizan muchos barridos por segundo, se puede "escribir" con el "lápiz" sobre la pantalla y el dibujo "ingresa" en la memoria de la computadora como una sucesión de puntos codificados.

La pantalla está imaginariamente dividida en 1.040.576 puntos, de manera que los trazos que se obtienen son prácticamente continuos.
Pueden dibujarse así curvas, estructuras, letras, números y cualquier tipo de gráfico, y esa información ingresa automáticamente a la computadora.

Por otra parte, los resultados obtenidos por la computadora son representados en la pantalla también como curva, letras, etc., bajo control del programa almacenado en la memoria.

Lectora Optica de Manuscritos: Salvo algunas pequeñas restricciones en cuanto al formato de los caracteres, esta unidad puede "leer" documentos escritos por cualquier persona y con cualquier ejemplo a una velocidad aproximada de 30,000 caracteres por minuto.

El registrador/analizador Fotográfico es una Unidad de Entrada/Salida de datos que realiza las siguientes funciones.

1) Registra los resultados de la computadora sobre microfotografías, mediante un tubo de rayos catódicos, que inciden sobre una película fotográfica, y cuyo haz electrónico actúa gobernado por el Programa Almacenado. La película se revela automáticamente dentro de la unidad y 48 segundos después está lista para ser proyectada.

2) Proyecta sobre una pantalla translúcida las microfotografías registradas.

3) Analiza imágenes reproducidas en negativo sobre película transparente, las digitaliza y las transmite a la Unidad Central de Procesamiento.

La película utilizada tiene 30.5 milímetros de ancho y 120 metros de longitud. La Entrada o Salida de imágenes puede consistir en letras, números, símbolos, dibujos, gráficas, mapas, curvas, etc. En una microfotografía de 30.5 mm X30.5 mm pueden registrarse hasta 30,600 letras y números, o hasta 16,777,216 puntos correspondientes a imágenes.

La velocidad de Registro/Análisis es de 40,000 letras, números y símbolos por segundo, o su equivalente si se trata de imágenes.

Máquina de Escribir (Teletipo).

Las unidades de almacenamiento o memorias (Representadas por el archivo del señor Control) permiten registrar las instrucciones y los datos para resolver un problema; entre estas se tienen:

Los Anillos Magnetizantes: Estos pueden magnetizarse en un sentido ó en otro "Recordando" así un 1 o un 0 respectivamente. Con 8 de éstos anillos se forma una posición de memoria, en la cual puede registrarse una letra, un dígito ó un carácter especial, según las distintas combinaciones de anillos "En 1" y "En 0", de acuerdo a un código predeterminado.

Las Memorias de Flip -Flops

Las Cintas Magnéticas

Los Discos Magnéticos

El dispositivo aritmético (representado por la máquina de cálcular) que realiza las cuatro operaciones aritméticas.

Las unidades de salida (representadas por la ventanilla "Salida" que pueden ser:

Impresoras

Máquinas de Escribir (Teletipos)

Grabadoras de Cintas Magnéticas

Grabadoras de Discos Magnéticos

Unidad de Representación Visual

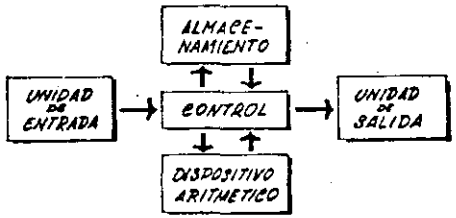
Registrador Analizador Fotográfico

Unidad de Respuesta Oral con la cual la Computadora puede hablar en todo el sentido de la palabra.

Contiene una Cinta magnetofónica en la cual un locutor ha grabado un diccionario de una gran variedad de palabras, en cualquier idioma.

Finalmente, un dispositivo electrónico de control (representado por el señor control) ayudado de un programa especial o sistema operativo (representado por el manual del señor Control), gobierna todas las operaciones de todas las unidades que componen la computadora.

Habiendo descrito las partes que componen la computadora podemos mostrar el siguiente esquema que la representa:



O en forma más resumida :

S i e n d o :

UNIDAD CENTRAL
- ALMACENAMIENTO
- CONTROL
- DISPOSITIVO ARITMETICO

Hemos hablado hasta este momento de la computadora electrónica desde el punto de vista conceptual. Durante las dos últimas décadas se han producido avances tecnológicos tan extraordinarios en materia de electrónica que la computadora ha sufrido enormes transformaciones. Veremos ahora cómo se ha ido modificando la idea original hasta llegar a los más modernos sistemas de procesamiento de datos.

Las primeras computadoras tenían circuitos con válvulas de vacío. Los tiempos de operación se median en ellas en milisegundos (milésimas de segundo). Cuando aparecieron los transistores, el diseño de los circuitos se mejoró notablemente y la duración de las operaciones en las computadoras que utilizaban esta "Tecnología de Estado Sólido" se midió en microsegundos (millonésimas de segundo) .

El hecho de que las nuevas máquinas fueran miles de veces más rápidas que las anteriores, trajo aparejada la creación de unidades de entrada, salida y memoria externa mucho más veloces.

La invención de un nuevo tipo de transistor ("chip") provocó una verdadera revolución en los circuitos electrónicos y sus procesos de fabricación. El nuevo elemento es tan pequeño que en un dedal de costura caben más de 50,000 chips. Debido a su tamaño, se les denomina circuitos microminiaturizados o microcircuitos. Los tiempos de operación se miden ahora en nanosegundos (milmillonésimas de segundo). Ha nacido en esta forma la tercera generación de computadoras, y las altas velocidades alcanzadas posibilitaron un nuevo enfoque en el diseño de los sistemas de procesamiento de datos.

Enunciaremos brevemente los adelantos que esta tercera generación ha introducido con respecto a la tecnología anterior :

. La computadora se autogobierna y trabaja sin detenerse, pasando de un trabajo a otro sin demora alguna.

. El Operador interviene sólo cuando algún problema excepcional ocurre. La comunicación entre hombre y máquina se realiza sólo sobre la base de "Informes por Excepción" .

. Si ocurre una falla en los circuitos o en la parte electromecánica la máquina realiza un autodiagnóstico e indica cuál es la anomalía.

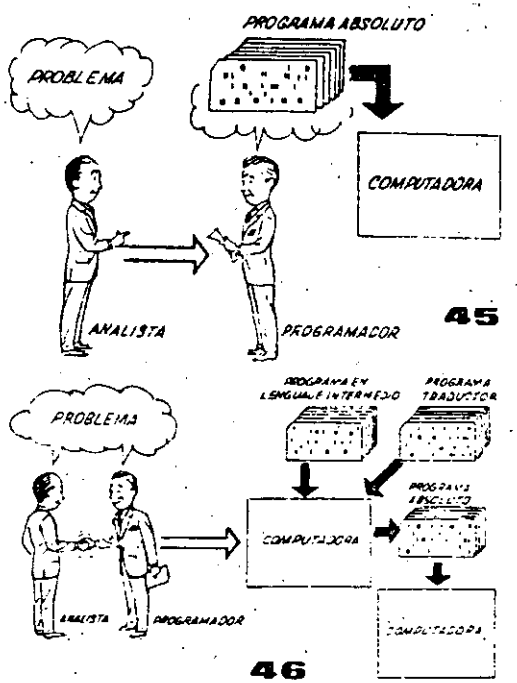
. La velocidad de Entrada-Proceso-Salida se ha incrementado extraordinariamente.

.Todas las operaciones del sistema se realizan en forma simultánea.

. Los lenguajes de programación han evolucionado de manera notable.

. El autocontrol y la autoverificación de operaciones han alcanzado niveles insospechados.

. Pueden realizarse, con máximo rendimiento, varios trabajos distintos simultáneamente. (Multiproceso).



Hasta ahora hemos visto muchas unidades que, en distintas combinaciones, configuran computadoras electrónicas para las más variadas aplicaciones. Ahora nos detendremos para analizar el manejo de dichos sistemas..

El Programa de Instrucciones almacenado en la Unidad Central de Procesamiento, consta de una secuencia de órdenes y comandos, expresados según una codificación especial denominada "Lenguaje Absoluto de Máquina"..Las primeras computadoras se "programaban" en este complejo lenguaje. Había entonces una enorme diferencia entre nuestro idioma y aquél según el cuál debíamos comunicarnos con la máquina. Esto obligaba a un gran esfuerzo común entre el analista que conocía el problema, y el programador que conocía la computadora, pues ambos hablaban del mismo proceso en distintos lenguajes.

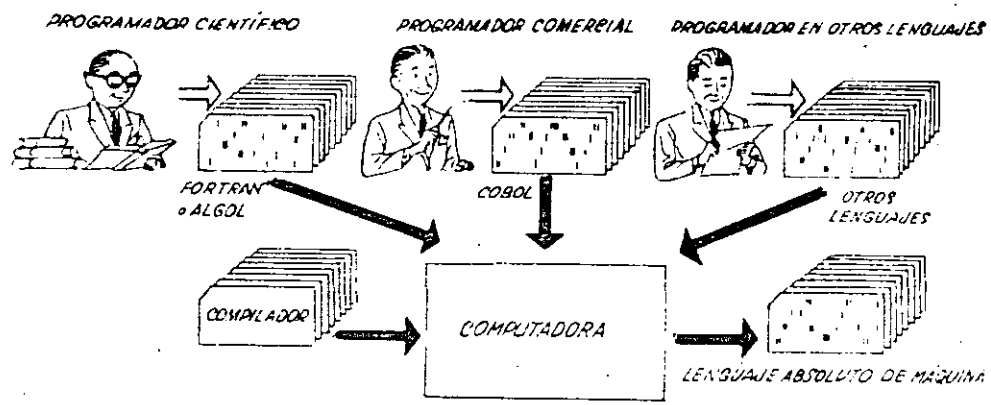
Se crearon, para solucionar el problema, lenguajes intermedios cada vez más parecidos a nuestro idioma. Es decir que cada nuevo lenguaje intermedio se acercaba más al problema y se alejaba más de la máquina. Para cada uno de estos lenguajes se creó un programa traductor llamado "Compaginador" o "Compilador", que tenía la misión de traducir el lenguaje intermedio al absoluto de máquina. Ahora, el analista y el programador "hablan un mismo idioma" :
ambos conocen el problema y la solución.

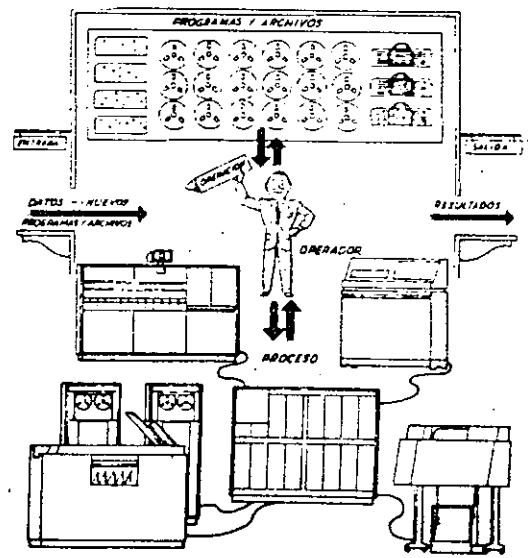
Pero la computadora seguía desarrollándose, y pronto los lenguajes intermedios fueron insuficientes para formular intrincados problemas científicos o comerciales. Nacieron, entonces, lenguajes especializados: dos de ellos, el FORTRAN y el ALGOL, permiten programar problemas científicos-técnicos utilizando una notación casi idéntica a la notación matemática común. El COBOL es un lenguaje comercial cuyas sentencias configuran oraciones y frases en forma tal que una persona que no sabe qué es una computadora, puede leer un programa y entender perfectamente qué es lo que hará la máquina cuando lo tenga almacenado.

Cada uno de estos lenguajes tiene un programa Compilador para cada tipo distinto de computadora capaz de procesarlo. Esto significa que un programador que sabe FORTRAN, por ejemplo, puede programar una computadora aún sin conocerla. Es decir que estos tres lenguajes constituyen un "esperanto" de las máquinas.

La tercera generación de computadoras permitió abordar complejos problemas que incluían, entre otros, aspectos comerciales y científicos.

Hemos llegado así a que la computadora nos "entienda", en lugar de que se limite a recibir órdenes en su idioma.





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

LENGUAJE DE PROGRAMACION COBOL

D C L RULES AND SYNTAX

SEPTIEMBRE, 1985

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtemoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

DCL RULES AND SYNTAX

INTRODUCTION

The main form of communication with the VMS operating system is through the Digital Command Language (DCL). DCL makes it fairly simple to tell VMS what you want to do. The DCL is based on the English language, which makes learning DCL much easier.

Like any language, DCL has its own form and syntax. This module discusses how to change English to DCL and DCL back to English. Translation is not as difficult as you might think. Once you understand the DCL rules and syntax, the language becomes relatively easy.

OBJECTIVES

1. To describe the purpose of a command language.
2. To select and define the parts of a DCL command line.

RESOURCES

1. VAX/VMS Primer
2. VAX/VMS Command Language User's Guide
3. VAX Software Handbook

PURPOSE OF THE DIGITAL COMMAND LANGUAGE (DCL)

The DIGITAL Command Language is the syntax (a set of rules that specifies what a correct statement is) that you use to tell VAX/VMS what you want it to do. Three goals were built into the design of DCL:

- Functional
- English-like
- Simple Commands

PARTS OF A DCL COMMAND

In English, each part of a sentence gives the reader a different piece of information. For example:

Carefully stop the train.

- What to do: stop
- How to apply the action: carefully
- Who or what to apply the action to: the train

The general form of a DCL command line is

verb/qualifier parameter

- The verb or action word comes first.
- Any qualifiers generally come immediately after the word to which they apply. The qualifier must be preceded by a slash (/).
- The parameter or who/what to do the action to, comes last. It must be separated from other words with one or more spaces.

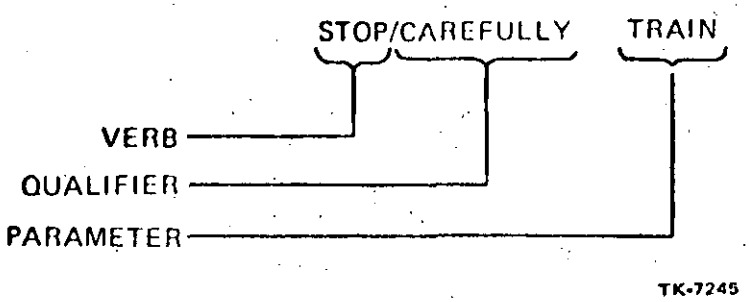


Figure 4-1 Parts of a DCL Command

Table 4-1 DCL Command Combinations

DCL Combination	Example
1 verb parameter	Tickle Annie
2 verb parameters	Tickle Fred,Annie
3 verb/qualifier parameter	Tickle/carefully Annie
4 verb parameter/qualifier	Tickle Annie/carefully
5 verb parameters/qualifier	Tickle Fred,Annie/carefully
6 verb/qualifier parameters	Tickle/carefully Luke,Jake

English translations of the Examples in Table 4-1 are as follows:

1. Tickle Annie.

2. Tickle Annie and Fred.

3. Carefully tickle Annie.

4. Tickle Annie carefully
   (Note: This command is equivalent to command number 3.)

5. Tickle Fred and then tickle Annie carefully.
   (In this command, the verb is applied to two different people. The qualifier allows the action to be taken differently for Annie than for Fred.)

6. Carefully tickle Luke and Jake.

DELIMITERS

Delimiters are punctuation marks so that the command is understandable to you and to the computer.

(a) TickleRay

(b) Tickle Ray

(c) Tickle carefully Ray

(d) Tickle/carefully Ray

Delimiters are important in understanding commands and instructions. As in English, sentences without delimiters are difficult to read.

For example:

Itishardtoreadsentencesiftheyarenotproperlyputtogether

The command line delimiter on most terminals is a single key, similar to the carriage return key on an electric typewriter.

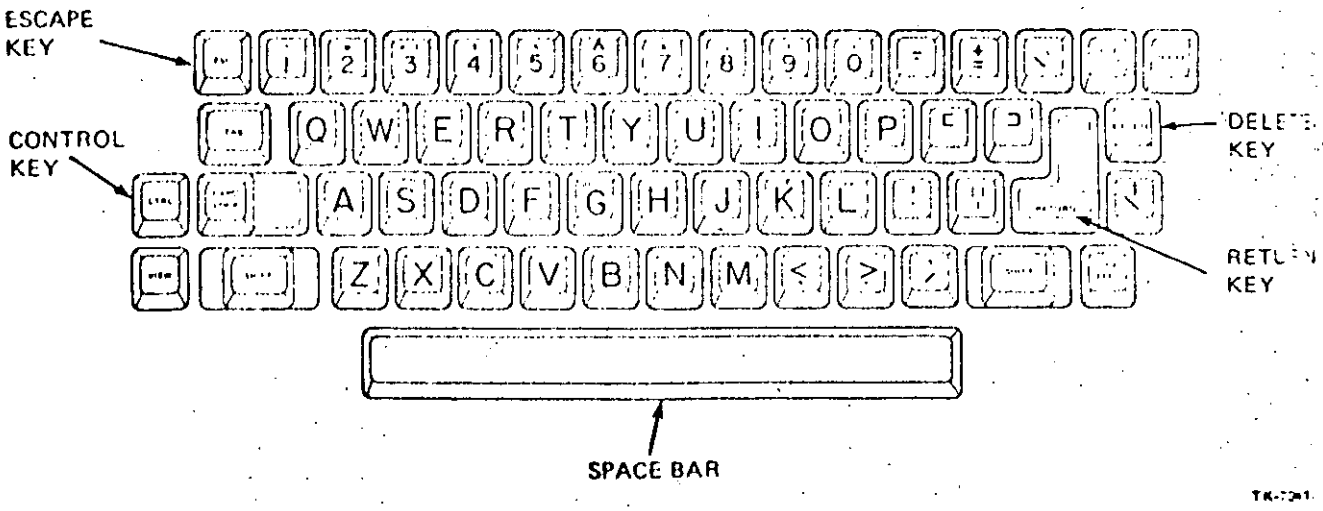


Figure 4-2 Standard DEC Keyboard

DEFAULTS

A default is information that is assumed unless otherwise indicated.

Table 4-1 Examples of DCL Command Defaults

DCL Command	Default	Resulting Command
GO	parameter=home	GO HOME
THROW BALL	verb qualifier=safely	THROW/SAFELY BALL
LIFT GODZILLA	parameter qualifier=slowly	LIFT GODZILLA/SLOWLY

PROMPTS

A prompt is one or more characters that appears on the terminal to indicate the program's readiness to receive a command.

NOTE

The interactive process is the only process type where a user is directly involved in entering the commands and waiting for each to finish.

The DCL Prompt

The DCL prompt is the dollar sign ($) followed by a space.

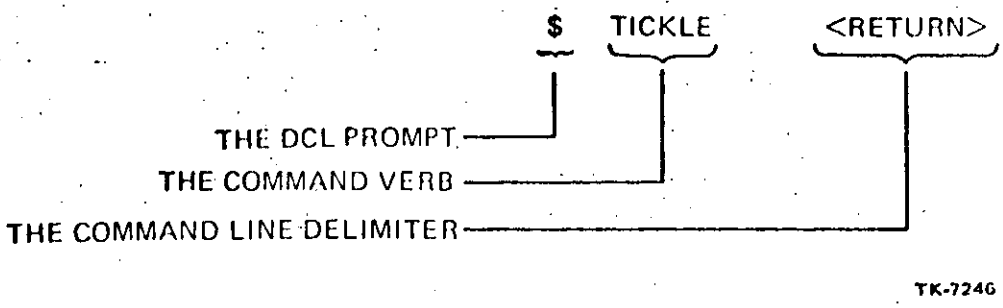


Figure 4-3 DCL Prompt and Command

Parameter Prompts

If an incomplete command is given to DCL, a parameter prompt will appear on the terminal. It reminds the user that more information is needed. For example,

```
$ TICKLE <RETURN>
```

shows a user issuing the command, TICKLE, but forgets to indicate who to tickle. DCL returns with a prompt indicating that more information is needed:

```
$_Who:  Ray
```

If there is a default parameter for the command TICKLE, no parameter prompt is given.

SUMMARY

In summary, to construct a command string:

- Wait for a prompt from the system. When the system is ready to receive a DCL command, it displays the dollar sign ($) at your terminal.
- Enter a command verb.
- If required or wanted, enter command qualifiers. Precede each command qualifier with a slash (/).
- If required or wanted, enter parameters. Separate the parameters from the verb and other parameters with one or more spaces.
- If required or wanted, enter parameter qualifiers. Again, precede each qualifier with a slash (/).
- Send the command to the system using the final delimiter (RETURN key).

Defaults are used for portions of the command line you do not choose to enter

If you forget what parameters to enter, hit the RETURN key after the verb and its qualifiers. The system will prompt you for the parameters it needs. If the verb has default parameters, no prompt will occur.

LENGUAJE DE PROGRAMACION COBOL

DCL COMMANDS

SEPTIEMBRE, 1985.

DCL COMMANDS

INTRODUCTION

For operators as well as users to manipulate and interact with the computer, it is necessary to use the correct language. This module explains the commands of DCL which perform specific operator-related duties.

Because DCL is flexible, several DCL commands accomplish the same thing. This module discusses the easiest and most direct DCL commands for the following categories:

- Information
- File control and manipulation
- Image and Process Control

OBJECTIVES

To select and invoke the proper DCL commands that:

- Return specific information
- Control and manipulate files
- Control images and processes

RESOURCES

1. VAX/VMS Command Language User's Guide
2. VAX Software Handbook

INTERACTION AND MANIPULATION

As an operator, it is necessary to communicate with VAX/VMS to perform the following functions:

- Retrieve information
- Provide new or updated information to VMS
- Create and remove files on the system
- Alter information within selected files

These functions are performed using DCL commands. Table 6-1 lists the DCL commands by group and their functions.

Table 6-1 DCL Commands

DCL Command and Group	Function
Informational	
HELP	Finds information about DCL commands.
SHOW	Obtains process or system information.
File Control/Manipulation	
DIRECTORY	Obtains a list of one or more files.
TYPE	Displays the contents of a file on the terminal.
PRINT	Prints the contents of the file on the lineprinter.
COPY	Creates a new file copy.
DELETE	Removes files from the system.
PURGE	Removes almost all copies of files.
Image and Process Control	
SET	Alters process, system or device information.
STOP	Stops a process or an image.
RUN	Activates a process or an image.

Informational Commands The following DCL commands provide information and assistance:

HELP - provides information about DCL commands and other topics.

SHOW - allows users to find information on their process or the operating system.

The HELP Command

Information in the form of help messages is available for most of the DCL commands and a few other selected topics. These help messages give:

- A brief description of the command
- The format of the command
- Parameters and qualifiers for which there is additional information.

To access these help messages and display them on your terminal, use the HELP command:

```
$ HELP
or
$ HELP topic
```

The help messages are organized by topic and subtopic; each help message may have subordinate help messages explaining various parameters and qualifiers. As you access help messages, you will be prompted for additional topics or subtopics. At each prompt, you may enter:

- The name of a topic or subtopic for more information and go to down a level to a subordinate help message
- ? to re-display current topic's help message
- <CR> to go up a level (exit at the top level)
- CTRL/Z to exit immediately

File Control/Manipulation Commands

An operator must be able to manipulate the files that exist on disk and tape volumes to:

- Obtain a list of file names.
- Examine the information in files.
- Obtain a printout of a file's contents.
- Remove files from system volumes.

Table 6-3 File Control/Manipulation Commands

Function	Command	Command Form
List file names in one or more directories.	DIRECTORY	$ DIRECTORY parameter $ DIRECTORY SYS$SYSTEM
Display contents of one or more files on the terminal	TYPE	$ TYPE parameter $ TYPE SYS$SYSTEM:STARTUP.COM
Print contents of one or more files on a line printer.	PRINT	$ PRINT parameter $ PRINT SYS$SYSTEM:STARTUP.COM
Copy old file to make a new file.	COPY	$ COPY old_name new_name $ COPY OLDFILE.DAT NEWFILE.DAT
Remove one or more files from a disk.	DELETE	$ DELETE parameter $ DELETE OLDFILE.DAT;*
Removes all but highest version(s) of selected file(s).	PURGE	$ PURGE parameter $ PURGE OLDFILE.DAT

The SHOW Command

SHOW displays current information about your process or the system or the terminal. The command form is:

SHOW parameter

Table 6-2 SHOW Command Parameters

Information	Command
List of active processes	SHOW SYSTEM
Date and Time	SHOW DAYTIME
Default device and directory	SHOW DEFAULT
Active logical names	SHOW LOGICAL
Process disk use	SHOW QUOTA
Process information	SHOW PROCESS
Process statistics	SHOW STATUS
Terminal characteristics	SHOW TERMINAL
Default file protection	SHOW PROTECTION

Table 6-5 Selected Qualifiers for the DIRECTORY Command

Qualifier	Result
None	Lists file names.
/PROTECTION	Lists file names and protection codes.
/SIZE	Lists file names and sizes in blocks.
/FULL	Lists file names and a full set of statistics for each file.

There are other qualifiers to the DIRECTORY command which provide selected portions from the /FULL qualifier display.

Listed below are selected parts from the output of a DIRECTORY/FULL command:

- Size:X/Y

  The amount of disk blocks used is X. The number of disk blocks allocated to the file is Y.

- Owner: [group_number,member_number]

  The owner number is the UIC number assigned to each user. The UIC is the only way to tell which files actually belong to which user.

- Created: Day-Month-Year

  The date the file is created.

- Revised: Day-Month-Year

  The revised date is when the file was last changed.

- File Protection

  The protection information indicates which UICs are allowed to read, write, execute, and delete the file.

The DIRECTORY Command

To display a list of file names that are on disk, use the DIRECTORY command. The form of the command is:

```
$ DIRECTORY parameter/qualifier
```

Table 6-4 DIRECTORY Command Parameters

Parameter	Example	Output
NONE		Lists file names on default device in the default directory.
Directory name	[STUDENT]	Lists file names in directory [STUDENT].
Name of file in default directory	FRED.DAT	Lists all files named FRED.DAT in default directory.
Name of file NOT in default directory	PHONEY.EXE	Lists error message stating "File not found".
Name of directory NOT on device	[FAKE]	Lists error message stating "Directory not found".
File name with wildcard	DEB.*	Lists any file in the default directory named DEB with any file type.
Ellipsis alone	[...]	Lists all file names in all subdirectories at all levels below default directory.
Ellipsis with directory name	[STUDENT...]	Lists all file names in all subdirectories at all levels below "STUDENT".
Wildcarded main directory	[*]	Lists all file names in all main level directories on default device.
Wildcarded main directory with file name	[*]FRED.DAT	Lists all files named FRED.DAT in all main level directories.
Wildcarded subdirectory	[STUDENT.*]	Lists all file names in all subdirectories below "STUDENT".

The COPY Command

To copy an existing file to a new file, use the COPY command. The COPY command form is:

```
$ COPY old_file_name new_file_name
```

For example:

```
$ COPY [MARSH]NAMES.DAT [MARSH]NEW.DAT
```

The COPY command is used to:

1. Obtain a copy of a file from another directory or subdirectory.
2. Create a copy of a file for other users to access without endangering the original file.
3. Create a copy of a file in a different directory.
4. Create a copy of a file whose information constantly changes.

The DELETE Command

The DELETE command allows you to get rid of files that are no longer needed. The DELETE command form is:

```
$ DELETE parameter
```

The parameter is a list of one or more file specifications. The file specifications must contain the version number of the file to be deleted. The only defaults for the DELETE command are the device and directory defaults.

If the verb qualifier /CONFIRM is added to the DELETE command, DCL asks the user permission to delete the file. If the response by the user is yes (Y), the file is deleted. If the response is no (N), the file is not deleted (see Example 6-6).

```
$ DELETE/CONFIRM FILE.DAT;2
USERDISK:[STUDENT]FILE.DAT;2, delete? (Y or N) :Y
$
```

The TYPE Command

To display the contents of a file on a terminal, use the TYPE command:

```
$ TYPE parameter
```

The parameter is made up of one or more file specifications. Like the DIRECTORY command, if the file specification is incomplete, VMS has a set of defaults to use.

Table 6-6 Defaults for the TYPE Command File Specification

Portion of File Specification	Default
Device	Present default device
Directory	Present default directory
Filename	No Default
File type	.LIS
Version	Highest version number

The PRINT Command

The PRINT command causes files to be printed on a line printer. The form of the command is:

```
$ PRINT parameter
```

The parameter is a list of one or more file specifications (separated by commas) to be printed.

The PRINT command has the same file specification defaults as the TYPE command (see Table 6-7).

Image and Process Control Commands

Many duties performed by an operator cannot be accomplished with one DCL command. Some duties need a program written specifically for that duty. Users and operators have several types of processes that can be used to work for them, including:

- Interactive
- Batch
- Detached
- Subprocess

Table 6-7 Image and Process Control Commands

Purpose	DCL Command
Alter process information	SET
Stop a process	STOP
Submit a batch process	SUBMIT
Activate an image	RUN
Create a subprocess	RUN/qualifier
Create a detached process	RUN/UIC=[uic]

The PURGE Command

The PURGE command removes all but the highest version number of each file specification given. The highest version numbered file is usually the most recent file. The PURGE command removes the older files, buts keeps the newest file for future use. The PURGE command form is:

```
$ PURGE parameter
```

The parameter is a list of one or more file specifications. If the parameter is not given (no file specifications), the PURGE command acts on all files in the default directory.

The /KEEP qualifier allows a user to specify how many versions of each file are kept. The default for the /KEEP qualifier is 1. If a user wants the two highest version numbered files kept, the qualifier /KEEP=2 is added to the PURGE command.

The STOP Command

The command to stop a process is STOP. The form of the command is:

```
$ STOP process_name
or
$ STOP/ID=pid
```

For example:

```
$ STOP MARSH
or
$ STOP/ID=002A0023
```

It is possible for two processes to have the same name if they have different group UIC numbers. To be sure the correct process is stopped, refer to it by the PID.

The SUBMIT Command

To submit a batch process for creation, use the SUBMIT command whose form is:

```
$ SUBMIT file_specification
```

The file specification is put on a list of files to be activated as batch processes. VMS allows only a few batch processes to run at one time. The files submitted are command procedures.

The SET Command

The SET command is used to change a characteristic (or value) of a process on VMS or the part of the system itself. SET requires a parameter (like the SHOW command). There is no default parameter value for the SET command.

Table 6-8 SET Command Parameters

Operation	DCL Command	Comment
Alter default device and/or directory	SET DEFAULT	The change is temporary; logging off or another SET DEFAULT resets it.
Alter protection of one or more files	SET PROTECTION	Must be owner of file or have SYSTEM UIC.
Alter your password	SET PASSWORD	Must be logged in and know old password.
Alter the terminal's characteristics	SET TERMINAL	Privilege required to use /PERMANENT.

The RUN Command

- Activates an image.
- Creates a subprocess.
- Creates a detached process.

Table 6-9 Using the RUN Command

	Necessary Qualifiers	Needed for Success
Image	NONE	READ or EXECUTE access to file
Subprocess	Any other than /DEBUG or /UIC	Same as image, plus sufficient quotas and limits.
Detached	/UIC	Same as image, plus DETACH privilege.

INTERPRETING DCL ERROR MESSAGES

A DCL error message is of the form:

```
%FACILITY-SEVERITY-CODE message
```

For example:

```
%DCL-W-IVVERB, unrecognized command
```

where

FACILITY indicates where the error comes from.

SEVERITY tells you how severe the error is. See Table 6-12 for message type and operator action.

CODE is an abbreviation of the message text that generally comes in the next line.

MESSAGE is a one line explanation of the problem.

Table 6-10 Severity Code Letters and Meanings

Code	Meaning	Operator Action
S	Success	Do nothing. The command works.
I	Information	Read the message for information. The command is all right.
W	Warning	The command may have performed some of the request; check the command and input for problems.
E	Error	Read the rest of the message.
F	Fatal Error	Same as error only more severe.

17

TERMINAL CONTROL CHARACTERS

In VMS, some of the terminal keys take on a special meaning when typed with the CTRL key.

Table 6-11 Terminal Control Key Functions

Function	Control Key
Interrupts a command or program.	CTRL/Y
Interrupts a command or program.	CTRL/C
Temporarily halts the flow of information to the terminal.	CTRL/S
Resumes the flow of information to the terminal.	CTRL/Q
Separates the terminal from the flow of information. The information keeps comming out, but the terminal does not put the output on the screen or paper.	CTRL/O
Reconnects the terminal to the flow of information.	CTRL/O

NOTE

If the terminal does not echo commands or words as you type them in, hit a CTRL/Q. This counteracts any CTRL/Ss you might have entered and forgotten about.

CURSOS: "INTRODUCCION AL SISTEMA VAX -11/780" Y
"OPERACION Y ADMINISTRACION DEL SISTEMA VAX / VMS"
DEL 29 DE ABRIL AL 13 DE MAYO.
MEXICO, D.F.

MODIFYNG THE DCL ENVIRONMENT

PROFESORES:

ING. EDUARDO S. JALLATH CORIA.
ING. ALEJANDRO JIMENEZ GARCIA.
ING. SOCRATES A. MUÑIZ ZAFRA.
ING. HUMBERTO SANCHEZ SANDOVAL.

MAYO DE 1985.

MODIFYING THE DCL ENVIRONMENT

INTRODUCTION

As an operator, you must become proficient with the DCL commands that control and interact with images, processes, and VMS itself. There are DCL commands that can be used to modify your DCL environment. These commands make DCL easier to use and you, as an operator, more efficient.

One method of changing your DCL environment is through what are known as Command Procedure files. Command procedure files contain a series of DCL commands. VMS can read the DCL commands from the files instead of from your terminal. By placing commonly-used DCL command sequences in these files, a user can more easily interact with VMS.

In the Files, Directories, and File Specifications module, logical names were substituted for portions of file specifications. This module discusses the DCL commands (ASSIGN and DEASSIGN) used to set up and get rid of logical names. By adding qualifiers to the ASSIGN command, logical names can be created for use by other processes. Logical names, as discussed in later modules, are a good way for an operator to assist and control VMS and user processes.

DCL symbols are also used to modify your DCL environment. DCL symbols are a series of characters that represent part or all of a DCL command. The DCL symbol selected is up to the user. This freedom of selection gives the user the ability to tailor the DCL command language to suit personal needs.

To accomplish some of the above tasks, it is necessary to create and modify certain kinds of files. To do this, you need to learn a system utility called the EDT editor. The EDT editor assists you in creating and modifying files that contain text. Text files, when created properly, can be used for command procedures, messages to other users, lists of tasks to accomplish, etc.

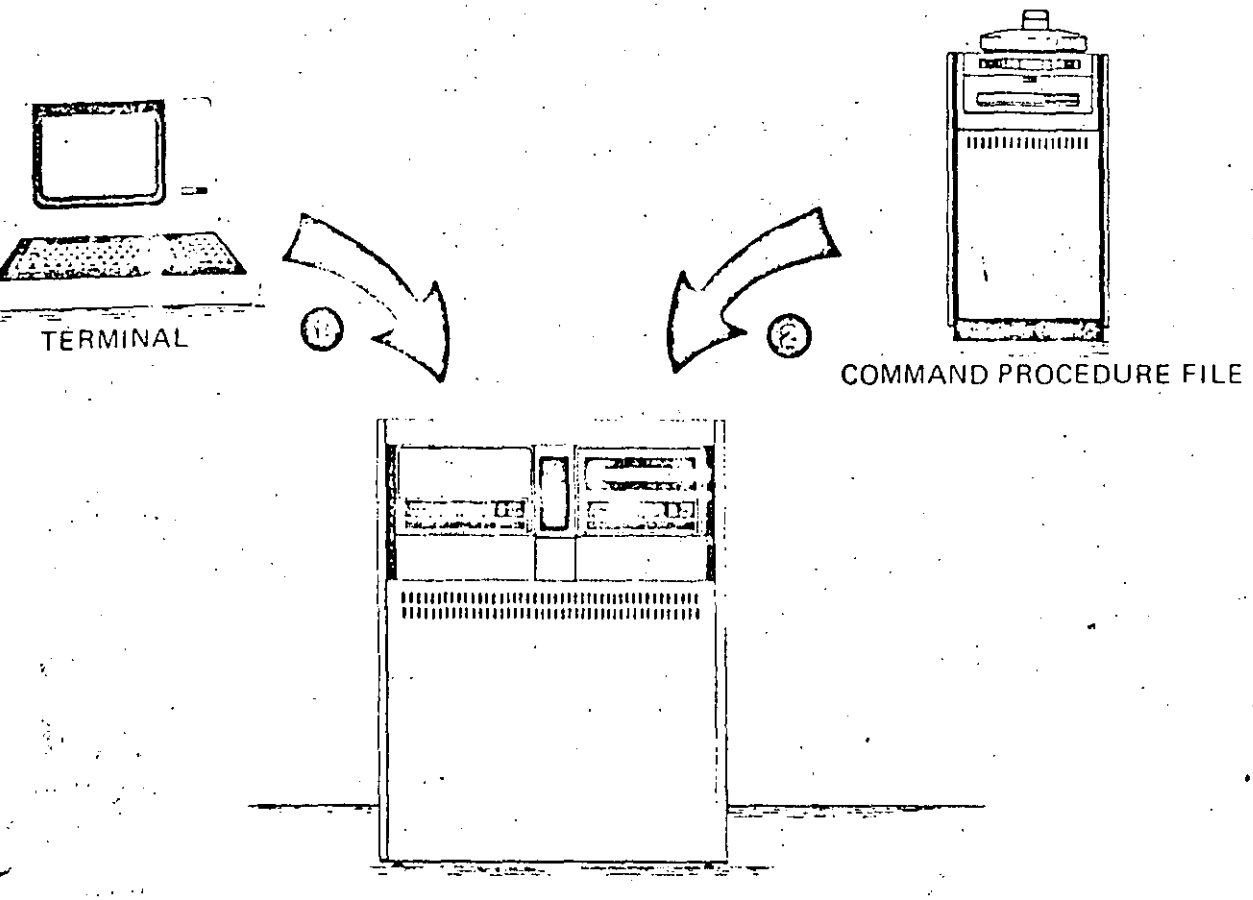
More importantly, the EDT editor helps you set up files necessary to operate the VAX/VMS system. The types of files used in system operation are presented throughout the course. The EDT editor enables you to examine, and if necessary, tailor the files for your job as a system operator.

MODIFYING YOUR DCL ENVIRONMENT

Table 7-1 Modifying DCL and Its Environment

Function	Method
Enter commands from a file instead of a terminal	Command Procedures
Create logical names	DCL command ASSIGN
Delete logical names	DCL command DEASSIGN
Shorten DCL commands or change DCL command verbs	DCL Symbols

COMMAND PROCEDURES



① DCL COMMANDS CAN BE ENTERED FROM THE TERMINAL

② BY ENTERING "@FILE SPECIFICATION", THE DCL COMMANDS CAN BE ENTERED FROM A COMMAND PROCEDURE FILE

Figure 7-1 Entering DCL Commands

- A command procedure is a file of DCL commands
- The file type defaults to .COM
- The DCL commands in a command procedure appear as they would be entered at a terminal
- A command procedure is activated by entering "@" and the file specification

```
COMMANDS.COM        (name of file)

$ SHOW PROCESS      (contents)
```

Example 7-1 Simple Command Procedure

To activate the above command procedure, type:

```
$ @COMMANDS.COM
```

```
INFO.COM                                (name of file)

$ SHOW PROCESS    !Comment one          (contents of file)
$! Comment two
$ SHOW LOGICAL
$ SHOW TIME
$   -   SHOW SYSTEM
```

Example 7-2 Command Procedure

- The $ should always be the first character in the first column of the command line.
- DCL commands in a command procedure should not be abbreviated.
- If you want to add comments to the file, place an exclamation mark in front of each comment.
- DCL commands are normally executed in the order they appear in the command procedure. However, this order can be changed by placing DCL logic commands in the file.

LOGICAL NAMES

Logical names can be used to remove any references to a specific device in DCL commands. They are set up by using the following DCL command:

```
$ ASSIGN   real_name   logical_name
```

where

real_name is the device or file specification desired.

logical_name is the name to be used in all DCL commands.

For example,

```
$ ASSIGN DBA1:  FRED

        <------------
```

To remove logical names after they have served their purpose, use:

```
$ DEASSIGN logical_name
```

For example,

```
$ DEASSIGN FRED
```

VMS uses a series of data structures called logical name tables to keep track of logical names. VMS uses three types of tables to hold all the logical names on the system:

- Process - Each process has its own logical name table. No process can read another process's logical name table.
- Group - Each group UIC has its own logical name table. Group tables are accessible only by users in each group and VMS.
- System - There is one system logical name table in VMS. It is accessible to all users as well as VMS.

To change entries in a group or system table requires a privilege for each type of logical name table. To change each table type (Process, Group, and System) requires the addition of qualifiers to the ASSIGN command.

Table 7-2 Logical Names

	Process	Group	System
DCL Command Qualifier	/PROCESS	/GROUP	/SYSTEM
Privilege needed to create or remove	NONE	GRPNAM	SYSNAM
Create Logical Name	ASSIGN/PROCESS	ASSIGN/GROUP	ASSIGN/SYSTEM
Find Logical Name	SHOW LOGICAL/PROCESS	SHOW LOGICAL/GROUP	SHOW LOGICAL/SYSTEM
Remove Logical Name	DEASSIGN/PROCESS	DEASSIGN/GROUP	DEASSIGN/SYSTEM

NOTE

When a user logs off the system, the Process logical names set up for that specific process are destroyed. However, the entries in the Group logical name tables and in the System logical name table are kept until specifically DEASSIGNed, or the computer is shut down.

SYMBOLS

Symbols can be thought of as command synonyms:

```
$ SYS=="SHOW SYSTEM"
```

where

SYS — is the symbol being created.

== — indicates that "SYS" is to mean the same as "SHOW SYSTEM".

" — indicates the beginning of the string to be replaced.

SHOW SYSTEM is the actual DCL command string.

" — indicates the end of the command string to be replaced.

Once the command SYS=="SHOW SYSTEM" is issued, the commands SYS and SHOW SYSTEM perform the same function.

Table 7-3 Local and Global Symbols

Function	Local Symbol	Global Symbol
Create a symbol	=	==
	$ SYS="SHOW SYSTEM	$ SYS=="SHOW SYSTEM"
Find a Symbol	SHOW SYMBOL/LOCAL	SHOW SYMBOL/GLOBAL
	$ SHOW SYMBOL/LOCAL SYS	$ SHOW SYMBOL/GLOBAL SYS
Remove a Symbol	DELETE/SYMBOL/LOCAL	DELETE/SYMBOL/GLOBAL
	$ DELETE/SYMBOL/LOCAL SYS	$ DELETE/SYMBOL/GLOBAL SYS
List all symbols	SHOW/SYMBOL/LOCAL/ALL	SHOW SYMBOL/GLOBAL/ALL
	$ SHOW SYMBOL/LOCAL/ALL	$ SHOW SYMBOL/LOCAL/ALL

Table 7-4 Symbols

Function	Symbol Creation	Example
Replace a DCL Verb	SCROLL=="TYPE"	$ SCROLL FRED.DAT
Replace a DCL Parameter	MINE==DBA2:[STUDENT]	$ DIRECTORY 'MINE' $ TYPE 'MINE'FRED.DAT
Replace a DCL Command Line	SYS=="SHOW SYSTEM"	$ SYS
Activate an Image	DO=="RUN file.exe"	$ DO
Activate a Command Procedure	LOOK=="@INFO.COM"	$ LOOK

How logical names and symbols are used is sometimes confusing. They both relate how DCL commands are specified; but, they differ in how they are applied. Table 7-5 compares logical names and symbols.

Table 7-5 Logical Names and Symbols

	Logical Names	Symbols
Function	Represents device, directory, and file specifications.	Represents commands or portions of command strings.
Use	Replaces part or all of a file specification.	Replaces part or all of a command. The symbol must be the first (left-most) part of the command line.
Created	$ ASSIGN command	Assignment statement (==)
Displayed	$ SHOW LOGICAL command	$ SHOW SYMBOL/GLOBAL command
Deleted	$ DEASSIGN command	$ DELETE/SYMBOL/GLOBAL command

CREATING AND MODIFYING FILES

Files are created and manipulated on VMS with system utilities called editors. VMS has several editors; one of the most commonly used is the EDT editor.

The EDT editor is run (activated) with the DCL command

```
$ EDIT file_spec
```

or

```
$ EDIT/EDT file_spec
```

The /EDT qualifier invokes the EDT editor. Other qualifiers invoke different editors.

NOTE

EDT looks for whatever file name is given. If the file you wish to edit exists, but you misspell the file name, EDT searches for the file by the misspelled name. The editor does not know that the file name requested is a misspelled existing file.

For example, to create a new file:

```
$ EDIT/EDT    PERSONAL.DAT
Input file does not exist
[EOB]
*
```

To edit an old file:

```
$ EDIT/EDT DBA2:[MARSH.OPER.STUDENT]FRED.DAT
1    first line of file
*
```

The EDT command form is:

```
*EDT-command parameter
```

where

*	is the EDT prompt
EDT-command	is a verb that tells the EDT editor what to do
parameter	for most commands is a line number or a range of lines numbers

The line numbers are assigned by EDT and can be changed when necessary. EDT assigns a number to each line in a file beginning at 1.

```
    INFO.COM

1   $ SHOW PROCESS   !First Comment
2   $! Second Comment
3   $ SHOW LOGICAL
4   $ SHOW TIME
5   $       SHOW SYSTEM
6   $
```

- A single line number is specified by simply giving the number (for example, 3 or 42)
- A range of line numbers is specified by giving the first and the last line numbers in the range, separated by a colon (for example, 1:5 or 24:36)
- The first line of a file may be specified by BEGIN
- The last line of a file may be specified by END

Table 7-6 EDT Editor Commands

Action	Command	One Line	Range of Lines
Display one or more lines of text	TYPE	*TYPE 1	*TYPE 1:4
Add text to a file at a specific line number	INSERT	*INSERT 3 new information <CTRL/Z>	
Delete one or more lines of text	DELETE	*DELETE 2	*DELETE 2:5
Exit from the editor saving all editing changes	EXIT	*EXIT	*EXIT
Exit from the editor NOT saving any editing changes	QUIT	*QUIT	*QUIT

Table 7-7 Additional EDT Commands

Action	Command and Parameter(s)	Examples
Copy one or more lines from one location to another in the same file	COPY range TO line_number	*COPY 1:2 TO 5 *COPY 4 TO 6
Move one or more lines from one location to another in the same file	MOVE range TO line_number	*MOVE 1:3 TO 6 *MOVE 4 TO 6
Substitute a set of one or more characters for a different set of one or more characters	SUB/set-1/set-2/range	*SUB/show/set/1:6 *SUB/show/set/3
Renumber the lines in the file (1,2,3,4, etc.)	RESEQUENCE	*RESEQUENCE
Copy of the entire contents of one file into the present edited file	INCLUDE file_name line#	*INCLUDE FRED.DAT 6

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

LENGUAJE DE PROGRAMACION

COBOL

PRESENTACION DE DATOS

AGOSTO, 1985

Palacio de Mineria Calle de Tacuba 5 primer piso Deleg. Cuauhtémoc 06000 México, D.F. Tel.: 521-40-20 Apdo. Postal M-2285

Representacion de datos.

Se define como dato al area donde se guarda informacion que puede ser o no modificada (Constantes y variables). La agrupacion de datos se llama grupo y la agrupacion de grupos definen un registro.

Cuando un dato es a la vez grupo y registro se le llama por este ultimo nombre REGISTRO.

La representacion de datos en COBOL esta dado por:

Numero de nivel Nombre-de-dato PICTURE IS tipo(longitud)

El nombre-de-datos esta regido por la definicion de palabras en COBOL.

Esta definicion de nombre-de-dato debe de ir de acorde al concepto que representa (lineas, nombre-archivo maestro, balance, pesos etc.), si dentro de la definicion de datos este no se quiere nombrar se utiliza la palabra FILLER.

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

- Para datos numericos 9.
- Para datos alfabeticos A.
- Para datos alfanumericos X.

Dentro de los datos numericos se pueden emplear los siguientes caracteres:

- Para definir que el numero es signado S.
- Para definir que el dato es no entero o real V (punto virtual).
- Para definir que el valor es escalado a una potencia de 10 P.

Ejemplos.

```
Data division.

    03 balance Pic 9(02)P(03)V9(02).
    04 nombre-cliente Pic X(40).
    06 suma-acumulada Pic S9(10)V9(02).
    05 filler Pic X(40).
```

Existen tipos de datos que se llaman editados, esto es porque son definidos para ser empleados para resultados de procesos. El tipo de datos esta denotado por los caracteres de edicion.

Caracteres de edicion		
	$	Signo doller
	Z	Supresion de ceros
	*	Proteccion de cheques
	.	Ajuste e insercion de punto
	,	Insercion de coma
	0	Insercion de ceros
	/	Insercion de slash

		Signo + evaluado
	DB	Debe evaluado
	CR	Credito evaluado
	B	Insercion de blanco

Ejemplos

Data division.

02 salario-resumen pic z(02)9(03).9(02).
02 balance pic $(02)z(03)9(03).9(02)cr.
03 peso pic $*(10).9(02).

En resumen los datos en cobol se definen segun su tipo, los cuales estan compuestos como indica la siguiente tabla.

Tipo de datos en cobol.

Numericos	Este tipo de datos hace combinacion de los caracteres 9 V P S
Alfabeticos	Descrito solo por el caracter A
Alfanumericos	Descrito solo por el caracter X
Numericos editados	Hace uso de los caracteres 9 V P Z B / 0 , . + - CR DB $ *
Alfabeticos editados	Hace uso de los caracteres A B
Alfanumericos editados	Hace uso de los caracteres A X 9 B 0 /

* Definicion basica de la representacion jerarquica de los datos.

< numero de nivel > nombre-de-registro.
< numero de nivel > nombre-de-grupo.
< numero de nivel > nombre-de-dato PICTURE IS tipo (longitud).

Ejemplo :

Data division.

01 registro-maestro.
05 Nombre pic x(30).
05 Direccion.
09 Calle pic x(10).
09 Numero pic 9(04).
09 Colonia pic is x(20).
09 Ciudad pic is x(20).
09 Codigo-postal pic is 9(06).

05 Estado-civil pic x(01).

* Clausulas referentes para determinar las caracteristicas de los datos.

- VALUE Is <literal-numerico, literal-nonumerico, constante-figurativa>.

Esta clausula es empleada para definir valores a los datos (ya sean constantes o variables), al momento de compilacion.

La literal no numerica se divide en entera y no entera o real. La literal entera es aquella que esta compuesta solo de digitos, y la no entera por digitos y un punto, el cual no puede iniciar ni terminar la literal.

La literal no numerica se refiere a la cadena de caracteres que se encuentra entre comillas (").

La constante figurativa es la palabra COBOL que esta predefinida, y son constantes figurativas:

- Zero, zeros, zeroes.
- Space, spaces.
- Quote, quotes.
- High-value, high-values.
- Low-value, low-values.
- All <literal no numerica de longitud 1>

Ejemplos :

```
Data division.

    ...

    01 contador-lineas pic 9(02) value zeroes.
    05 filler pic x(10) value all "*".
    04 filler pic x(14) value "Nominita, S.A."

    ...
```

- BLANK WHEN ZERO.

Se usa esta clausula para denotar que si el dato tiene como valor cero en la impresion se denotara con blancos independientemente del formato de edicion numerico empleado.

Ejemplo:

```
Data division.

    ...

    05 resultado pic zz9.99.BLANK WHEN ZERO.

    ...
```

- CURRENCY AND DECIMAL-POINT.

Con esta clausula se dice que sigma presenta el dolar ($) y el

punto decimal en los formatos numericos.
Ejemplo : $2345,45 como P2345,45

Se debe seguir la siguiente forma para su definicion.

Ejemplo :

Environment division.
Configuration section.
Special-names.
    Currency sign is 'P'
    Decimal point is comma.

USAGE IS (COMPUTACIONAL , COMP , DISPLAY , INDEX).

La representacion de los datos numericos puede ser de diferentes maneras. Esta representacion depende del tipo de maquina. En forma estandard se tienen dos tipos que son :

DISPLAY es del modo caracter y esta es el default.
COMPUTACIONAL es el modo numerico.

Ejemplo :

Data division.

03 estadisticas pic 99 USAGE COMPUTACIONAL.
03 sueldo pic 99 USAGE DISPLAY.

(SYNCHRONIZED, SYNC) (LEFT, RIGTH).

Cuando se usa la clausula USAGE se incrementa la eficiencia del programa pero no le da todo el trabajo. La calusula SYNCHRONIZED debe ser usada para completar esta efectividad. Esto es debido a las diferentes estructuras de las maquinas.

Ejemplo :

Data division.

03 valor-inventario pic 9(02) USAGE COMPUTACIONAL SYNCHRONIZED LEFT.

(JUSTIFIED, JUST) RIGHT.

Esta opcion usada para campos alfabeticos o alfanumericos solamente es empleada para definir el modo de insercion de informacion al campo indica que la insercion de contenido se efectuara de derecha a izquierda, contrariamente a como se efectue.

Ejemplo :

Data division.

- SIGN IS (LEADING, TRAILING) SEPARATE CHARACTER.

Esta opcion se usa para indicar que el signo del numero es un caracter que va al principio (LEADING) o al final (TRAILING) del numero.

Ejemplo :

```
Data division.
    ...
    03 suma-acumulada picture is S9(07)
-       SIGN IS LEADING SEPARATE CHARACTER.
    ...
```

Multidefinicion de nombre de datos.

La multiple definicion de nombres de datos, o el uso de nombre de datos repetidos solo es permitido en nombres de datos agrupados en grupos o registros, y esto se usa para optimizar el movimiento de datos.

Ejemplo :

```
Data Division.
  ...
01 registro-maestro.
   03 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.
```

Redefinicion en el nombre de area de datos.

La redefinicion de area de datos es usada cuando se quiere hacer referencia a una misma area de informacion. Hace uso de la clausula REDEFINES el cual presenta las siguientes restricciones.

- No se puede redefinir areas a nivel 01 de la File section.
- El area a redefinir debe declararse en la definicion de dato anterior a la redefinicion.
- El area a redefinir y el dato que define deben estar al mismo nivel y tener la misma longitud.
- No se puede redefinir datos con nivel 66 o 88.

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.
```

Renombrar el de area de datos.

Renombrar el area de datos es utilizado para proveer al proceso de la capacidad de "reagrupar" informacion de los datos a excepcion de el REDEFINES que redefine una area, este renombramiento se realiza con la posicion de la informacion.

Como se notara en el ejemplo el area de informacion comprendida entre apellidos y paso-educacion se llama tambien ficha-paso-hacienda, ademas de que se emplea un numero de nivel nuevo (66) y la clausula 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 paso-personal pic 9(06).
          09 paso-educacion pic 9(06).
```

```
      09 paso-imss pic 9(06).
      09 paso-precio-casa pic 9(06).
   66 ficha-paso-hacienda RENAMES apellidos THRU paso-educacion.


Procedure division.
Inicio.

    Read causantes-empresa into resistro-de-impuestos at end.
    Write pasos-a-hacienda from ficha-paso-hacienda.

    Stop run.
```

Nombres condicionales de datos.

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

Ejemplo :

```
Data division.


   01 fin-de-archivo pic x(02) value "no".
      88 fin-de-archivo-ok value "si".
   03 carrera-de-insenieria pic 9(02).
      88 carrera-correcta value 21 THRU 29, 31, 32.
      88 carrera-insenieria-civil value 21.

      88 carrera-insenieria-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-insenieria-civil
       Perform reporte-civiles.
    else
       if carrera-insenieria-[illegible]
```

```
elsif carrera=ingeniria-computacion
    Perform reporte-computacion.
```

Representacion de datos.

Se define como dato al area donde se guarda informacion que puede ser o no modificada (Constantes o variables). La agrupacion de datos se llama grupo y la agrupacion de grupos definen un registro.

Cuando un dato es a la vez grupo y registro se le llama por este ultimo nombre REGISTRO.

La representacion de datos en COBOL esta dado por:

Numero-de-nivel Nombre-de-dato PICTURE IS tipo (longitud).

El nombre-de-dato esta regido por la definicion de palabras en COBOL.

Esta definicion de nombre-de-dato debe de ir de acorde al concepto que representa (lineas , nombre-archivo-maestro , balance, pasos etc.), si dentro de la definicion de datos este no se quiere nombrar se utiliza la palabra FILLER.

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

La representacion de los datos numericos puede ser de diferentes maneras. Esta representacion depende del tipo de maquina. En forma estandard se tienen dos tipos que son:

- DISPLAY es el modo caracter y este es el default.
- COMPUTACIONAL es el modo numerico.

Ejemplo:

```
Data division.

    03 estadisticas pic 99 USAGE COMPUTACIONAL.
    03 sueldo pic 99 USAGE DISPLAY.
```

[SYNCHRONIZED, SYNC] [LEFT, RIGTH].

Cuando se usa la clausula USAGE se incrementa la eficiencia del programa, pero no la del todo el trabajo. La calusula SYNCHRONIZED debe ser usada para completar esta efectividad. Esto es debido a las diferentes estructuras de las maquinas.

Ejemplo :

```
data division.

    03 valorsinventario pic 9(02) USAGE COMPUTACIONAL SYNCHRONIZED LEFT.
```

Esta opcion usada para campos alfabeticos o alfanumericos solamente es empleada para definir el modo de insercion de informacion al campo e indica que la insercion de contenido se efectuara de derecha a izquierda, contrariamente a como se efectua.

Ejemplo :

Data division.

01 atencion-usuario Pic x(20) JUSTIFIED RIGTH.

SIGN IS (LEADING, TRAILING) SEPARATE CHARACTER.

Esta opcion se usa para indicar que el signo del numero es un caracter que va al principio (LEADING) o al final (TRAILING) del numero.

Ejemplo :

Data division.

```
    03 suma-acumulada picture is S9(07)
       SIGN IS LEADING SEPARATE CHARACTER.
```

Multidefinicion de nombre de datos

La multiple definicion de nombres de datos o el uso de nombre de datos repetidos solo es rermitido en nombres de datos asrusados en srupos o otros, y esto se usa rara ortimizar el movimiento de datos.

Ejemplo :

```
Data Division.

01 resistro-maestro.
   05 Nombre ric x(40).
   05 Direccion.
      09 Calle ric x(10).

      09 codiso-rostal ric 9(06).

01 resistro-auxiliar.
   05 Nombre ric x(40).
   05 Direccion.
      09 Calle ric x(10).

      09 codiso-rostal ric 9(06).


Procedure division.
 Inicio.

     Move corresronding resistro-maestro to resistro-auxiliar.

     Stor run.
```

Redefinicion en el nombre de area de datos

La redefinicion de area de datos es usada cuando se quiere hacer referencia a una misma area de informacion. Hace uso de la clausula REDEFINES el cual presenta las siguientes restricciones:

- No se puede redefinir areas a nivel 01 de la File section.
- El area a redefinir debe declararse en la definicion de dato anterior a la redefinicion.
- El area a redefinir y el dato que define deben estar al mismo nivel y tener la misma longitud.
- No se puede redefinir datos con nivel 66 o 88.

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.
```

Renombrar el de area de datos

Renombrar el area de datos es utilizado para proveer al proceso de la capacidad de reagrupar información de los datos, a excepción de el REDEFINES que redefine una area, este renombramiento se realiza por medio de la palabra RENAMES.

Como se notara en el ejemplo el area de información comprendida entre apellidos y paso-educacion se llama también ficha-paso-hacienda, además de que se emplea un numero de nivel nuevo (66) y la clausula 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 paso-personal pic 9(06).
          09 paso-educacion pic 9(06).
          09 paso-imss pic 9(06).
          09 paso-predio-casa pic 9(06).
       66 ficha-paso-hacienda RENAMES apellidos THRU paso-educacion.
.
.
.
Procedure division.
Inicio.
.
.
.
       Read causantes-empresa into registro-de-impuestos at end ...
       Write pasos-a-hacienda from ficha-paso-hacienda.
.
.
.
       Stop run.
```

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 generalmente 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.
```

El tipo esta dado por la naturaleza del datos son:

- Para datos numericos 9.
- Para datos alfabeticos A.
- Para datos alfanumericos X.

Dentro de los datos numericos se pueden emplear los siguientes caracteres:

- Para definir que el numero es signado S.
- Para definir que el dato es no entero o real V (punto virtual).
- Para definir que el valor es escalado a una potencia de 10 P.

Ejemplos.

Data division.

```
03 balance pic 9(02)P(03)V9(02).
01 nombre-cliente pic X(40).
06 suma-acumulada pic S9(10)V9(02).
05 filler pic x(40).
```

son definidos para ser empleados para resultados de procesos. El tipo de datos está denotado por los caracteres de edición.

Caracteres de edición	$	Signo dollar
	Z	Supresion 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	Credito evaluado
	B	Inserción de blanco

Ejemplos :

Data division.

```
02 salario-resumen Pic z(03)9(03).9(02).
02 balance Pic $(03)z(03)9(03).9(02)cr.
03 paso Pic $*(10).9(02).
```

En resumen los datos en cobol se definen segun su tipo, los cuales estan compuestos con indicadores la siguiente tabla.

Tipo de datos en cobol.

Numericos	Este tipo de datos hace combinacion de los caracteres 9 V P S
Alfabeticos	Descrito solo por el caracter A
Alfanumericos	Descrito solo por el caracter X
Numericos editados	Hace uso de los caracteres 9 V P Z B / 0 , . + - CR DB $ *
Alfbeticos editados	Hace uso de los caracteres A B
Alfanumericos editados	Hace uso de los caracteres A X 9 B 0 /

A B

numero de nivel > nombre-de-registro.

numero de nivel > nombre-de-grupo.

numero de nivel nombre-de-dato PICTURE IS tipo (longitud).

Ejemplo:

```
Data division.

...

01 registro-maestro.
   05 Nombre Pic X(30).
   05 Direccion.
      09 Calle Pic X(10).
      09 Numero Pic x(04).
      09 Colonia pic is x(20).
      09 Ciudad pic is x(20).
      09 Codigo-Postal pic is 9(06).
   05 Sexo pic x(01).
   05 Estado-civil pic x(01).

...
```

* Cláusulas referentes para determinar las características de los datos.

VALUE is <literal-numerica, literal-nonumerica, constante-figurativa>.

Esta clausula es empleada para definir valores a los datos (ya sean constantes o variables), al momento de compilacion.

La literal no numerica se divide en entera y no entera o real. La literal entera es aquella que esta compuesta solo de digitos, y la no entera por digitos y un punto, el cual no puede iniciar ni terminar la literal.

La literal no numérica se refiere a la cadena de caracteres que se encuentra entre comillas (").

La constante figurativa es la palabra COBOL que esta predefinida, y son constantes figurativas:

- Zero, zeros, zeroes.
- Space, spaces.
- Quote, quotes.
- High-value, high-values.
- Low-value, low-values.
- All <literal no numerica de longitud 1>.

Ejemplos :

Data division.

```
01 contador-lineas pic 9(02) value zeroes.
03 filler pic x(10) value all "*".
04 filler pic x(14) value "Nominitas S.A."
```

Se usa esta clausula para denotar que si el dato tiene como valor cero en la impresion se denotara con blancos independientemente del formato de edicion numerico empleado.

Ejemplo:

```
Data division.

    03 resultado pic zz9.99 BLANK WHEN ZERO.
```

Con esta clausula se dice que signo reemplaza al dolar ($) y al punto decimal en los formatos numericos.
Ejemplo : $2345.45 como F2345,45

Se debe seguir la siguiente forma para su definicion.

Ejemplo :

```
Environment division.
Configuration section.
Special-names.
    Currency sign is 'F'
    Decimal point is comma.
```

Data division.

La DATA DIVISION identifica el lugar donde se declaran los datos que se usaran en el proceso. En general consiste en dos secciones:

- FILE SECTION.
- WORKING-STORAGE SECTION.

En la FILE SECTION se definen los datos que provienen de almacenamiento externo (archivos).

En la WORKING-STORAGE SECTION se definen los datos de uso interno (variables locales), así como registros de encabezado y detalle que se emplearan al emitir reportes.

FILE SECTION

La FILE SECTION indica los datos que son introducidos al proceso por medio de archivos y debe existir asociacion entre la DATA DIVISION y la ENVIRONMENT DIVISION.

La FILE SECTION describe:

- El nombre de archivo.
- El nombre del registro asociado al archivo.
- Caracteristicas logicas-fisicas de los registros (etiquetas, numero de caracteres por registro, numero de registros por bloque, etc)
- La estructura jerarquica de los datos dentro del registro.
- Tipo y longitud de los datos.

Ejemplo :

```
Environment division.
Input-output section.
File-control.
    Select archivo-datos assign to "datos.dat".

Data division.
File section.
FD  archivo-datos
    record contains 80 characters
    label record are omitted
    data record is registro-datos.
01  registro-datos.
    05 nombre pic x(40).
    05 sexo pic 9(01).
       88 femenino value 0.
       88 masculino value 1.
    05 salario pic 9(06)V99.
    05 filler pic x(31).
```

FD significa FILE DESCRIPTION, y existen tantos fd's como select's.

Definicion Multiple de datos provenientes de un archivo.

Esta definicion Multiple de datos provenientes de un archivos es empleada cuando se definen en la estructura del archivo un registro principal y varios registro auxiliares, generalmente se usa cuando se tienen registros de ancho fija pequeña (80 columnas) y gran informacion a guardar. Para diferenciar un registro de otro se usa una indicacion en un campo del registro para saber que pertenecen a un grupo de informacion con una llave.

Ejemplo:

```
Environment division.
Input-output section.
File-control.
    Select informacion-alumnos assign to "alumnos.dat".

Data division.
File section.
Fd  informacion-alumnos

    data record are informacion-general, informacion-historial, ...

01  informacion-general.
    03 numero-de-cuenta pic x(08).
    05 tarjeta-numero   pic x(01).
    05 definicion-informacion.
       09 nombre pic x(40).
          ...

01  informacion-historial.
    03 numero-de-cuenta pic x(08).
    05 tarjeta-numero   pic x(01).
    05 definicion-historial.
       09 anio-de-ingreso pic 9(20).
          ...
```

WORKING-STORAGE SECTION.

La WORKING-STORAGE SECTION se definen datos y registros que se usan en el proceso y que no provienen de archivos. Estos elementos son llamados constantes y variables locales.

Tradicionalmente se han definido las constantes a un nivel 77 asi como variables de control (interruptores), y como registros los datos de encabezado, de detalle y auxiliares.

Ejemplo :

```
Data division.

Working-storage section.

77 pi pic 9(01),9(04) value 3.1416.
77 fin-de-archivo pic x(02) value "no".
   88 fin-de-archivo value 'si'.
77 contador-de-lineas value zeroes.
77 contador-de-hojas  value zeroes.
01 encabezado-general.
   05 filler pic x(10) value spaces.
   05 filler pic x(14) value 'Nominitar S.A.'.
   05 filler pic x(10) value spaces.
   05 hoja   pic z(04) value 0001.
   05 filler pic x(42) value spaces.
```

```
datdef9.idx     00
datdiv1.idx     00
datdiv10.idx    00
datdiv2.idx     00
datdef1.idx     00
datdef2.idx     00
datdef3.idx     00
datdef4.idx     00
datdef5.idx     00
datdef6.idx     00
datdef7.idx     00
datdef8.idx     00
datdef9.idx     00
datdef10.idx    00
datdef11.idx    00
datdef12.idx    00
datdef13.idx    00
datdef14.idx    00
datdef15.idx    00
datdef16.idx    00
datdef17.idx    00
```

Asumiendo que las los nombres de dato X, Y y Z inicialmente contienen los valores indicados en la primera li%'nea, indique para cada instruccion MOVE (si es posible) el valor que tienen estos nombres de datos despues de ejecutada la instruccio%'n. Instruccion Nombre de datos X Y Z valores iniciales 10 18 20 MOVE X TO Y MOVE X TO 15 MOVE 15 TO X MOVE X TO Y,Z MOVE ZEROES TO X, Y, Z Para cada uno de las siguietes instrucciones MOVE, indicar si la instruccion es valida o invalida en COBOL dependiendo del campo emisor y el campo receptor. Campo emisor. Campo receptor. Resultado. Alfabetico Nume%'rico Alfabetico Alfanume%'rico Nume%'rico Alfabetico Nume%'rico Alfanume%'rico Alfanume%'rico Alfabetico Alfanume%'rico Nume%'rico

en blanco.

Instruccion	valor x	valor y	valor z
valores iniciales	10	12	100
add x to y			
add x y sivins z			
add x y			
add z to y			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	20
subtract y from z			
subtract z from y			
subtract z from x sivins y			
subtract x from y sivins z			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
multiply y by z			
multiply y by z sivins x			
multiply z by y			

Instruccion	valor x	valor y	valor z
valores iniciales	10	12	100
divide x by z			
divide z into x			
divide z into x sivins y			

Instruccion	valor w	valor x	valor y	valor z
valores iniciales	15	10	12	100
add x to y				
add w x to y				
add 5, y sivins z				
add w, x, y				
add w, 3, x sivins z				
add w 3 x to z				
add z x to 3				
add x w sivins y				
add x to y rounded				
add x 12.456 to y rounded				
add 1000 12.3 x to y				
add 1000 12.4 x to y rounded on size error move zero to w				
add x y sivins z rounded w rounded				

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	20
subtract y from z			
subtract z from y			
subtract y z from x			
subtract 12 18 from x			
subtract y 25 from 100			
subtract z from x sivins y			
subtract y 25 from 100 sivins z			
subtract x 25.6 from y rounded			
subtract -12000.4 from y rounded			
on size error move +999.99 to z			
subtract 10 from z sivins x, y			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
multiply y by z			
multiply y by z sivins x			
multiply z by y			
multiple z by 20			
multiply z by 20 sivins x			
multiply 2.5 by x rounded			
multiply x by y rounded			
multiply 100.25 by x rounded on size error move zero to y			
multiply x by 10.2 sivins y z rounded			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
divide x by z			
divide z into x			
divide x into 100			
divide z into 100 sivins y			
divide 2 into z sivins x			
divide x by z sivins y			
divide 60 by y sivins x			
divide 12.2 into y rounded			
divide 12.2 into x y rounded			
divide x into 10 sivins y, z rounded			
divide x by 10 sivins x, z rounded			
divide 4.5 into x sivins y rounded			
on size error move zero to z			

Expresion aritmetica Cobol	Expresion algebraica
a + b / c	
(a + b) / c	
a + (b / c)	
((a + (b * c)) / d) ** 2	
((a + (b * c)) / d ** 2	
a + (b * c) / d ** 2	
a + b * c / d ** 2	

Instruccion	valor x	valor y	valor z
valores iniciales	10	12	100
add x to y			
add x y sivins z			
add x y			
add z to y			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	20
subtract y from z			
subtract z from y			
subtract z from x sivins y			
subtract x from y sivins z			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
multiply y by z			
multiply x by z sivins x			
multiply z by y			

Instruccion	valor x	valor y	valor z
valores iniciales	10	12	100
divide x by z			
divide z into x			
divide z into x sivins y			

Instruccion	valor w	valor x	valor y	valor z
valores iniciales	15	10	12	100
add x to y				
add w x to y				
add 5, y sivins z				
add w, x, y				
add w, 3, x sivins z				
add w 3 x to z				
add z x to 3				
add x w sivins y				
add x to y rounded				
add x 12.456 to y rounded				
add 1000 12.3 x to y				
add 1000 12.4 x to y rounded on size error move zero to w				
add x y sivins z rounded w rounded				

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	20
subtract y from z			

```
subtract z from y
subtract y z from x
subtract 12 13 from x
subtract y 25 from 100
subtract z from x sivins y
 obtract y 25 from 100 sivins z
subtract x 25.6 from y rounded
subtract -12000.4 from y rounded
   on size error move +999.99 to z
subtract 10 from z sivins x, y
```

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
multiply y by z			
multiply y by z sivins x			
multiply z by y			
multiply z by 20			
multiply z by 20 sivins x			
multiply 2.5 by x rounded			
multiply x by y rounded			
multiply 100.25 by x rounded on size error move zero to y			
multiply x by 10.2 sivins y z rounded			

Instruccion	valor x	valor y	valor z
valores iniciales	90	30	10
divide x by z			
divide z into x			
divide x into 100			
divide z into 100 sivins y			
divide 2 into z sivins x			
divide x by z sivins y			
divide 60 by y sivins x			
divide 12.2 into y rounded			
divide 12.2 into x y rounded			
divide x into 10 sivins y, z rounded			
divide x by 10 sivins x, z rounded			
divide 4.5 into x sivins y rounded on size error move zero to z			

Expresion aritmetica Cobol	Expresion algebraica
a + b / c	
(a + b) / c	
a + (b / c)	
((a + (b * b)) / d) ** 2	
(a + (b * c)) / d ** 2	
a + (b * c) / d ** 2	
b * c / d ** 2	

Sumario del move

Tipo de campo emisor	tipo de campo receptor			
	alfabetico	alfanumerico	alfanumerico editado	numerico entero	numerico no ente	numerico editado
alfabetico	si	si		no		
alfanumerico	si	si		si		
alfanumerico editado	si	si		no		
numerico entero	no	si		si		
numerico no entero	no	no		si		
numerico editado	no	si		no		

Ejercicios con move

Move	Campo emisor		Campo receptor
	999V99	01352	9999V999
	999V99	25235	999V99
	999V99	25235	99V99
	999V99	25235	99V9
	9999	1000	9999V99
	9999V	0001	999999
	999V99	12613	99999V9
	99999V99	1000225	9999V9
	AAAAAA	MCGRAW	AAAAA

Combinacion de campos emisores y receptores.

campo emisor	Campo receptor	Valido	No valido
Numerico	Numerico		
Numerico	Alfabetico		
Numerico	Alfanumerico		
Alfabetico	Numerico		
Alfabetico	Alfabetico		
Alfabetico	Alfanumerico		
Alfanumerico	Numerico		
Alfanumerico	Alfabetico		
Alfanumerico	Alfanumerico		

Move	Campo X	Campo Y	Campo Z
Valores iniciales	10	18	20
move x to y			
move x to 15			
move 15 to x			
move x to y, z			
move x to z, y			
move zeroes to x, y, z			

Tipo de caracteres que pueden usarse junto a la clausula picture.

Tipo de caracter	Sembolo	Uso
Definicion de tipo de campo	9	Campo numerico
	A	Campo alfabetico
	X	Campo alfanumerico
Definicion de tipo campo numerico especial	V	Punto virtual
	P	Ajuste potencial
	S	Inclusion de signo
Caracteres de edicion	$	Signo dollar
	Z	Supresion de ceros
	*	Proteccion de campos
	.	Ajuste e insercion de punto
	,	Insercion de coma
	0	Insercion de ceros
	/	Insercion de slash
	+	Signo + evaluado
	-	Signo - evaluado
	DB	Debe evaluado
	CR	Credito evaluado
	B	Insercion de blanco

Evaluacion de caracteres de edicion.

Tipo de caracter de edicion	Valor positivo	Valor negativo
+	+	-
-		-
db		db
cr		cr

Tipo de datos en cobol.

Numericos	Este tipo de datos hace combinacion de los caracteres 9 V P S
Alfabeticos	Descrito solo por el caracter A
Alfanumericos	Descrito solo por el caracter X
Numericos editados	Hace uso de los caracteres 9 V P Z B / 0 , . + - CR DB $ *
Alfabeticos editados	Hace uso de los caracteres A B
Alfanumericos editados	Hace uso de los caracteres A X 9 B 0 /

Ejercicios sobre descripcion de tipo de datos.

Descripcion	Valor numerico	Representacion en almacenamiento
02 Suma-prod pic is 9(06).	12,327	
03 Bodega picture is 9(04)	8,956	
02 Poblacion pic is 9(10)	1,563,813	
03 afiliacion pic is 9(04)	285	
02 horas-trabajadas pic 99v9	38.50	
03 paso-neto pic 9(04)v99	452.39	
03 Capacidad-total pic 999	555	
02 monto pic 99ppPV	12	
03 descuento pic vp(03)9(04)	1023	
07 Balance pic s9999v99	136.29	
02 Balance pic s9999v99	-1251.16	
04 Balance pic s9(04)v99	-0.10	
05 Balance pic 9(04)v99	-325.18	

Descripcion	Ejemplo	Representacion en almacenamiento
02 nombre-de-parte pic xxxxx	diodo	
02 nombre-de-parte pic x(05)	pija	
03 nombre-de-alumno x(20)	J. Anaya Sanchez	
04 codigo-mensaje x(30)	$_Define file :	
03 nombre-elemento pic a(10)	cuchara	
04 nombre-de-parte pic a(10)	Tornillo	

Descripcion	Valor numerico	Representacion en impresion
02 resultado pic $999.99	135.13	
2 resultado pic $9(05).99	200.0	
02 resultado pic $$99.99	12.49	
02 resultado pic $$$9.99	150.10	
02 resultado pic $$$$.99	0.15	
02 resultado pic $$$$.$$	0.0	
02 resultado pic $9,999.99	2,350.22	
02 resultado pic $9,999.99	150.31	
02 resultado pic $$,999.99	130.20	
02 resultado pic $$,$$$.99	25.40	
02 resultado pic $$,$$$.999	0.019	
02 resultado pic $$,$$$.$$$	0.009	
02 resultado pic $$,$$$.$$$	0.0	
02 resultado pic $$,$$9.999	2,210.2	
02 resultado pic $$,999.9	2,210.2	
02 resultado pic $$,999.9	2,210.256	
02 resultado pic $9,999.9999	23	
02 resultado pic $$,$$$,$$9	0.002	
02 resultado pic z99	25	
02 resultado pic zzz.99	25	
02 resultado pic zzz.99	0.10	
02 resultado pic zzz.zzz	0.256	
02 resultado pic zzz.zzz	0.0	
02 resultado pic $zzz.9	13.2	
2 resultado pic $z,zzz,zzz.cr	156,320.18	
02 resultado pic $z,zzz,zzz.z	3,156,344.18	
02 resultado pic $$,$$z.999	0.001	
02 resultado pic $***.99	256.10	
02 resultado pic $***.99	10.13	
03 resultado pic $***.99	0.15	
03 Balance pic +999.9	32.2	
03 Balance pic 999.9+	32.2	
03 Balance pic 999.9+	-32.2	
03 Balance pic ++9.9	-001.3	
03 Balance pic ++++.99	0.55	
03 Balance pic ++++.99	-0.55	
03 Balance pic ++++.++	0.01	
03 Balance pic ----.--	0.0	
03 Balance pic --99.99	-10.25	
03 Balance pic -999.99	100.25	
03 Balance pic $$$$.99-	-10.2	
03 Balance pic $$$$.99+	20.35	
03 Balance pic $999.99db	132.26	
03 Balance pic $999.99db	-132.26	
03 Balance pic $,$$9.99cr	-10.50	

Descripcion	Valor numerico	Representacion en almacenamiento
02 Suma-prod pic is 9(06).	12,327	
03 Bodega picture is 9(04)	8,956	
02 Poblacion pic is 9(10)	1,563,813	
03 afilacion pic is 9(04)	285	
02 horas-trabajadas pic 99v9	38.50	
03 pago-neto pic 9(04)v99	452.39	
03 Capacidad-total pic 999	555	
02 monto pic 99PPPV	12	
03 descuento pic VP(03)9(04)	1023	
07 Balance pic s9999v99	156.29	
02 Balance pic s9999v99	-1251.16	
04 Balance pic s9(04)v99	-0.10	
05 Balance pic 9(04)v99	-325.18	

Descripcion	Ejemplo	Representacion en almacenamiento
02 nombre-de-parte pic xxxxx	diodo	
02 nombre-de-parte pic x(05)	pija	
03 nombre-de-alumno x(20)	J. Anaya Sanchez	
04 codigo-mensaje x(30)	$_Define file :	
03 nombre-elemento pic a(10)	cuchara	
04 nombre-de-parte pic a(10)	Tornillo	

Descripcion	Valor numerico	Representacion en impresion
02 resultado pic $999.99	125.13	
02 resultado pic $9(05).99	200.0	
02 resultado pic $$99.99	12.49	
02 resultado pic $$$9.99	150.10	
02 resultado pic $$$$.99	0.15	
02 resultado pic $$$$.$$	0.0	
02 resultado pic $9,999.99	2,350.22	
02 resultado pic $9,999.99	150.31	
02 resultado pic $$,999.99	130.20	
02 resultado pic $$,$$$.99	25.40	
02 resultado pic $$,$$$.999	0.019	
02 resultado pic $$,$$$.$$$	0.009	
02 resultado pic $$.$$$.$$$	0.0	
02 resultado pic $$,$$9.999	2,210.2	
02 resultado pic $$,999.9	2,210.2	
02 resultado pic $$,999.9	2,210.256	
02 resultado pic $9,999,9999	23	
02 resultado pic $$,$$$,$$9	0.002	
02 resultado pic z99	25	
02 resultado pic zzz.99	25	

```
02 resultado pic zzz.99                    0.10
02 resultado pic zzz-zzz                   0.256
02 resultado pic zzz.zzz                   0.0
02 resultado pic $zzz.9                   13.2
02 resultado pic $z,zzz,zzz,zz   154,320.18
02 resultado pic $z,zzz,zzz,z  3,156,344.18
02 resultado pic $$,$$z.999                0.001
-------------------------------------------------
02 resultado pic $***.99                 256.18
03 resultado pic $***.99                  10.13
03 resultado pic $***.99                   0.15
-------------------------------------------------
03 Balance pic +999.9                     32.2
03 Balance pic 999.9+                     32.2
03 Balance pic 999.9+                    -32.2
03 Balance pic ++9.9                     -001.3
03 Balance pic +++9.99                     0.55
03 Balance pic +++9.99                    -0.55
03 Balance pic ++++.++                     0.01
03 Balance pic ----.--                     0.0
03 Balance pic --99.99                   -10.25
03 Balance pic -999.99                   100.25
03 Balance pic $$$$.99-                  -10.2
03 Balance pic $$$$.99+                   20.38
03 Balance pic $999.99db                -132.26
03 Balance pic $999.99db                -132.26
03 Balance pic $.$$9.99cr                -10.50
```

Cuestionario.

1. Mencione el nombre de las cuatro divisiones en orden.
2. Para que sirve la séptima posición del formato ansi de codificación cobol.
3. Que son las áreas A y B.
4. De que forma son denotadas las palabras reservadas en el formato de instrucciones cobol.
5. Que son las constantes figurativas.
6. Explique los siguientes conceptos. Programa fuente. Programa objeto. Compilador.
7. Cual es el parrafo en donde se pone el nombre del programa.
8. Que longitud pueden tener los nombres definidos en cobol.
9. Que propositos persigue la IDENTIFICATION DIVISION.
10. De las cuatro divisiones, en cual nos referimos a las particularidad del centro de computo y al equipo empleado.
11. Mencione las dos secciones de la ENVIRONMENT DIVISION.
12. Que función desempeña la DATA DIVISION.
13. Que significa FD.
14. Mencione dos de las secciones de la DATA DIVISION.
15. En que circustancias se emplea el termino de resistro.
16. En que sección de la DATA DIVISION puede usarse la clausula value para denotar caracteristicas de evaluación en compilación de los datos.
17. Que es la representación jerarquica de los datos.
18. Diga cuales son los tipos principales para denotar datos.
19. Que es el punto virtual.
20. De que forma se realiza el movimiento de información entre datos numericos.
21. Caracter de edición referente a la supreción de ceros.

22. Caracter que define la correcta posición del punto decimal en la definición de un dato.

23. De que forma se realiza el movimiento de información entre datos alfabeticos y alfanumericos.

24. Cual es la diferencia entre el caracter V y el caracter . (punto) en la definición de datos.

25. Por que usa la inserción de comas.

26. Como se evalua los caracteres + , - , cr y db.

27. Por que se usa el caracter de protección de campos.

28. Que clausula define que si el campo numerico es cero, este sea representado con espacios.

29. Como podemos cambiar el sentido del movimiento entre campos alfabeticos y alfanumericos.

30. Porque se permite la multiple definición de nombres de dato, cuales son sus restricciones.

31. Porque un dato se redefine o una area se renombra, cuales son las diferencias y cuales son sus restricciones.

32. Que son los nombre condicionales, como se definen y cuando se emplean.

EVALUACION DE COBOL BASICO.

TOMANDO COMO EJEMPLO EL PROGRAMA DE LA NOMINITA, RESPONDA LAS SIGUIENTES PREGUNTAS E INDIQUE EJEMPLOS.

1. CUALES SON LAS ACCIONES NECESARIAS PARA EL FUNCIONAMIENTO DEL PROGRAMA.

2. INDIQUE EL OBJETIVO DE LAS DIVISIONES Y SECCIONES.

3. CUAL ES LA DIFERENCIA ENTRE UN DATO ELEMENTAL Y UN REGISTRO.

4. PARA QUE SIRVEN LOS NUMEROS DE NIVEL, LAS CONSTANTES FIGURATIVAS Y LOS CARACTERES DE EDICION.

5. QUE Y CUALES SON LAS FIGURAS LOGICAS DE LA PROGRAMACION ESTRUCTURADA.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                 NOMINITA.
AUTHOR.                     JORGE-VALERIO
INSTALLATION.               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PUBLICA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO MENCIONANDO LA FUENTE.
ENVIRONMENT DIVISION.
CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATOS-ENTRADA ASSIGN TO "PERSONAL.DAT".
    SELECT DATOS-SALIDA  ASSIGN TO "LISTA.LIS".
DATA DIVISION.
FILE SECTION.
FD  DATOS-ENTRADA
    RECORD CONTAINS 39 CHARACTERS
    DATA RECORD IS DATOS-FD.
01  DATOS-FD.
    03    NOMBRE-DATOS-FD                  PIC X(30).
    03    SUELDO-DIARIO-DATOS-FD           PIC 9(04)V9(02).
    03    DIAS-TRABAJADOS-DATOS-FD         PIC 9(02).
    03    SEXO-DATOS-FD                    PIC X(01).
FD  DATOS-SALIDA
    RECORD CONTAINS 132 CHARACTERS
    DATA RECORD IS LINEA-FD.
01  LINEA-FD                              PIC X(132).
WORKING-STORAGE SECTION.
77   HAY-MAS-DATOS-WS                PIC X(02)          VALUE "SI".
77   SUELDO-MENSUAL-WS               PIC 9(06)V9(02).
77   SOBRE-SUELDO-WS                 PIC 9(06)V9(02).
77   NUMERO-LINEAS-WS                PIC 9(04)          VALUE 60.
01   ENCABEZADO-WS.
     03 FILLER                       PIC X(50)          VALUE SPACES.
     03 FILLER                       PIC X(32)          VALUE
     "EMPRESA DEL LIBRE COMERCIO S.A. ".
     03 FILLER                       PIC X(50)          VALUE SPACES.
01   PROCESO-WS.
     03 FILLER                       PIC X(34)          VALUE SPACES.
     03 FILLER                       PIC X(64)          VALUE
     "SUELDOS PERCIBIDOS EN LA SEMANA DEL 2 AL 7 DE NOVIEMBRE DE 1983.".
     03 FILLER                       PIC X(34)          VALUE SPACES.
01   TITULO-WS.
     03 FILLER                       PIC X(41)          VALUE SPACES.
     03 FILLER                       PIC X(06)          VALUE "NOMBRE".
     03 FILLER                       PIC X(30)          VALUE SPACES.
     03 FILLER                       PIC X(14)          VALUE "SUELDO MENSUAL".
     03 FILLER                       PIC X(41)          VALUE SPACES.
01   DETALLE-WS.
     03 FILLER                       PIC X(37)          VALUE SPACES.
     03 NOMBRE-DETALLE-WS            PIC X(30).
     03 FILLER                       PIC X(11)          VALUE SPACES.
     03 SUELDO-MENSUAL-DETALLE-WS    PIC $ZZZZ9.99.
     03 FILLER                       PIC X(45)          VALUE SPACES.
PROCEDURE DIVISION.
INICIO.
    PERFORM PRIP-ARCHIVOS.
```

```
    PERFORM CERRAR-ARCHIVOS.
    PERFORM FIN-DE-PROGAMA.
ABRIR-ARCHIVOS.
    OPEN    INPUT       DATOS-ENTRADA
            OUTPUT      DATOS-SALIDA.
PROCESO.
    PERFORM LECTURA
            WITH TEST BEFORE UNTIL NOT HAY-MAS-DATOS-WS IS EQUAL "SI".
LECTURA.
    READ DATOS-ENTRADA
         AT END MOVE "NO" TO HAY-MAS-DATOS-WS.
    IF HAY-MAS-DATOS-WS EQUAL "SI" THEN
       PERFORM CALCULOS
       PERFORM ESCRITURA
    ELSE
       NEXT SENTENCE.
*   ENDIF
CALCULOS.
    MULTIPLY        SUELDO-DIARIO-DATOS-FD
          BY        DIAS-TRABAJADOS-DATOS-FD
          GIVING    SUELDO-MENSUAL-WS.
    IF SEXO-DATOS-FD = "M" THEN
       PERFORM CINCO-PORCIENTO
    ELSE
       PERFORM DIEZ-PORCIENTO
    END-IF.
CINCO-PORCIENTO.
    MULTIPLY        0.05
          BY        SUELDO-MENSUAL-WS
          GIVING    SOBRE-SUELDO-WS.
    ADD SOBRE-SUELDO-WS TO SUELDO-MENSUAL-WS.
DIEZ-PORCIENTO.
    MULTIPLY        0.10
          BY        SUELDO-MENSUAL-WS
          GIVING    SOBRE-SUELDO-WS.
    ADD SOBRE-SUELDO-WS TO SUELDO-MENSUAL-WS.
ESCRITURA.
    MOVE NOMBRE-DATOS-FD TO NOMBRE-DETALLE-WS.
    MOVE SUELDO-MENSUAL-WS TO SUELDO-MENSUAL-DETALLE-WS.
    IF NUMERO-LINEAS-WS > 57 THEN
       MOVE 4 TO NUMERO-LINEAS-WS
       WRITE LINEA-FD FROM ENCABEZADO-WS AFTER PAGE
       WRITE LINEA-FD FROM PROCESO-WS AFTER 2
       WRITE LINEA-FD FROM TITULO-WS AFTER 2
       WRITE LINEA-FD FROM DETALLE-WS AFTER 2
    ELSE
       ADD 2 TO NUMERO-LINEAS-WS
       WRITE LINEA-FD FROM DETALLE-WS AFTER 1
    END-IF.
CERRAR-ARCHIVOS.
    CLOSE   DATOS-ENTRADA
            DATOS-SALIDA.
FIN-DE-PROGAMA.
    STOP RUN.
```

LENGUAJE DE PROGRAMACION COBOL

GUIA DE ACCESO AL SISTEMA VAX 11-780

SEPTIEMBRE,1985

Palacio de Minería Calle de Tacuba 5 primer piso Deleg. Cuauhtemoc 06000 México, D.F., Tel.: 521-40-20 Apdo. Postal M-2285

GUIA DE ACCESO AL SISTEMA VAX 11-780

1

ENTRADA AL SISTEMA VAX11-780

1.1. QUE ES EL SISTEMA VAX 11/780 DEL CENTRO DE CALCULO.

La VAX-11/780 es una super minicomputadora de 32 bits por palabra de memoria, fabricada por Digital Equipment Corporation(DEC) , la cual forma parte de la serie de m'quinas m's populares utilizadas actualmente en la

INTRODUCCION:

ESTE MANUAL HA SIDO DESARROLLADO COMO UNA GUIA PARA EL USUARIO PRINCIPIANTE DE LA VAX/11-780 CON COMANDOS DCL TERMINALES TELEVIDEO. EN ESTA GUIA SE ENCONTRARAN LAS INSTRUCCIONES DE COMANDO MAS COMUNES ASI COMO SUS FORMATOS. MEDIANTE ESTA LO UNICO QUE SE PRETENDE ES DAR LA AYUDA NECESARIA PARA ENTRAR AL SISTEMA, ESCRIBIR, BORRAR, EDITAR, COMPILAR Y CORRER PROGRAMAS. TODO ESTO HECHO DE UNA FORMA SENCILLA Y CLARA QUE ESPERAMOS SEA DE UTILIDAD.

EDITH SILVA M.

JORGE ONTIVEROS

28 DE NOVIEMBRE DE 1983

INDICE

ENTRADA AL SISTEMA VAX11-780

1.1. QUE ES EL SISTEMA VAX 11/780 DEL CENTRO DE CALCULO.

La VAX-11/780 es una super minicomputadora de 32 bits por palabra de memoria, fabricada por Digital Equipment Corporation(DEC) , la cual forma parte de la serie de máquinas más populares utilizadas actualmente en las Universidades. La capacidad actual es de 2 megabytes y se tiene posibilidades de crecimiento muy considerables. La VAX se basa en las micros y minis PDP también desarrolladas por DEC, de las cuales existen dos en la Facultad. La VAX cuenta con una buena cantidad de software a disposicion de los alumnos como son los compiladores BASIC, FORTRAN, COBOL, PASCAL; así como paquetes desarrollados por el personal del Centro, y una gran cantidad de rutinas de comandos de gran utilidad y fácil acceso.

El nombre VAX viene de 'Virtual Address eXtension' que significa extensión de direccionamiento virtual y esto viene porque la VAX tiene más capacidad de direccionamiento virtual que sus antecesoras las PDP y tiene capacidad de memoria virtual lo que le permite procesar programas más grandes que el tamaño de su memoria, gracias al concepto de paginación.

Para entender mejor esta primera parte, es necesario explicar que es una computadora.

Una computadora es una máquina con gran velocidad para realizar operaciones, con una gran capacidad de almacenamiento de información pero que necesita que le digamos que secuencia de pasos hay que seguir.

Una computadora está compuesta por unidades de entrada, unidades de salida y la unidad central de procesamiento.

Las unidades de entrada pueden ser terminales, tarjetas, etc.

Las unidades de salida pueden ser impresoras, cintas, discos, terminales, etc.

La unidad central de proceso es el corazón de la

computadora y contiene la unidad aritmético-lógica, la unidad de control y la unidad de memoria. La unidad aritmético-lógica es la que realiza las operaciones aritméticas y booleanas, la unidad de control lleva el control de las operaciones (es la que gobierna) y la unidad de memoria es donde se almacenan los datos. La memoria puede ser memoria principal y memoria auxiliar (constituida por discos, cintas y/o diskettes).

La capacidad de las computadoras se mide en palabras o en bytes; una palabra es un conjunto de bytes (en Vax son dos bytes para WORD o cuatro bytes para LONGWORD. La VAX maneja registros de 32 bits (LONGWORD)), un Byte es un conjunto de bits (por lo general 8), y el bit es la unidad mínima de memoria, esto es, un lugar donde se puede almacenar un 0 ó un 1 (sistema binario de numeración).

Como la capacidad de las computadoras ha crecido mucho en los últimos años, se tienen factores que hacen manejables las cantidades de Bytes y son: Kilo=1024 (que es 2^10) entonces 1 Kb = 1024 bytes, Mega=1,048,144(que es 2^20) entonces 1 Mb = 1,048,144 bytes.

El sistema Operativo es el VMS (Virtual Memory System) que significa Sistema de Memoria Virtual el cual permite un manejo eficiente de la memoria física gracias al principio de la Memoria Virtual. El Sistema Operativo es el administrador de los recursos de la computadora.

Una página de memoria en Vax está constituida por 512 bytes que es a su vez un bloque en disco.

La definición de computadora que se proporciona es muy simple, para poder diferenciar computadoras, se subdividen las Unidades, por lo que la descripción de la configuración se realiza en forma más detallada.

La configuración actual del sistema es:

Subsistema de Consola:
- 1 microprocesador LSI-11
- 1 unidad de diskettes de 256 Kb
- 1 terminal LA-120

Unidad Central de Proceso con:
- 1 Respaldo de baterías para salvar a disco los procesos.
- 1 Acelerador de punto flotante.
- 1 Memoria caché.

Synchronos Backplane Interface
que es un canal que comunica a la memoria, al Massbus y al Unibus con el Procesador, con una capacidad de 13.3 Mb/seg.

Memoria principal:	1	controlador de memoria con 8 módulos de 256 Kb c/u = 2 Mb.
Massbus de 2 Mb/seg con:		
Almacenamiento Secundario	3	unidades de disco de 176 Mb cada una.
Almacenamiento en cinta	1	unidad de cinta magnética de 800/1600 bpi y 45 ips
Unibus de 1.5 Mb/seg con:		
Impresión	1	impresora de 600 lpm
	1	impresora graficadora
Lectura	1	lectora de tarjetas de 300 tarjetas por minuto
disco flexible	1	unidad dual de diskkette de 256 kb c/u
terminales	3	multiplexores con capacidad de 8 terminales cada uno.
	20	terminales de video VT 100
	4	terminales de papel LA 120 de 120 caracteres por seg. cada una.

Antes de entrar en sesión (conectarse con el sistema para utilizar los recursos de la computadora) es necesario describir el funcionamiento de algunas teclas importantes.

El teclado de los terminales se parece al de una máquina de escribir normal, pero tiene una serie de teclas adicionales como son:

RETURN que se utiliza para finalizar la información que le estamos dando a la computadora.
DELETE que sirve para borrar el caracter que acabamos de teclear.
CAPS LOCK que sirve para escribir todo con letras mayúsculas pero no para escribir los caracteres que están colocados en la parte superior de las teclas con dos caracteres.
SHIFT sirve para escribir los caracteres alternos de las teclas con dos caracteres(ejemplo 4 y $ están en la misma tecla).
CTRL sirve para dar instrucciones especiales para la computadora como seria CRTL U (oprimir simultaneamente las teclas CTRL y U) que nos permite desechar una linea completa.
NO SCROLL sirve para detener/continuar con el desplegado en la pantalla.

ESC que permite proporcionar una secuencia de ESCape para modificar las características de la pantalla o auxiliarnos en el manejo de la misma.
BACK SPACE genera un caracter (CTRL/H) que hace retroceder un caracter el cursor pero no borra el caracter encimado (aunque en la pantalla así lo parezca).
LINE FEED baja una línea en la pantalla (en modo local).

1.2. USERNAME.

Llegando a la sala de terminales lo primero que se tiene que hacer es encender el switch interruptor que se encuentra del lado izquierdo inferior posterior de la terminal. Tardará un breve lapso en aparecer un cursor y este tiempo que tarda la terminal es el que utiliza para calentarse , ya que la pantalla funciona como un simple cinescopio de televisión. Ya que apareció el cursor (un cuadro) hay que oprimir la tecla <return>, con la que se llama la atención del procesador y que pedirá el USERNAME asignado. Se deberá teclearlo y oprimir <return>

1.3. PASSWORD.

El PASSWORD es una clave secreta que va asociada con el USERNAME. En este momento el sistema está pidiendo esa clave , deberá escribirla pero tome en cuenta que no se desplegará en pantalla, sino que será interpretado por la máquina únicamente. Después de realizar esta operación oprimir <return> y se estará ya dentro del sistema. La operación general es:

password: No se despliega en pantalla!!!

1.4. MENSAJES DEL SISTEMA.

Lo primero que se verá al entrar al sistema será un mensaje que dirá Bienvenido al Sistema VAX/VMS V3.3 (Cecafi) posteriormente se podrán desplegar mensajes del sistema al usuario , tales como advertencias, mensajes varios , etc... Aparecerá el PROMPT (indicador) , el cual es un $ (que indica que estás en el DCL ó Digital Command Language).

Cuando esto sucede se dice que el usuario ha entrado en sesión lo que significa que se puede hacer uso de los recursos de la computadora.

1.5. DIFICULTADES DE ACCESO.

Los mensajes que podrá dar la máquina en caso de una dificultad de acceso son los siguientes:
a) Que aparezca un mensaje que diga USER AUTORIZATION FAILURE , esto es que se cometió un error al poner el USERNAME o el PASSWORD por lo que tendrá que iniciar nuevamente el proceso de entrada.
b) Si la computadora no responde puede ser que en ese momento el sistema esté fuera de servicio momentaneamente ('caido') por lo que deberá esperar el mensaje correspondiente al reinicio de operaciones.
c) Si se entra en sesión pero no permite ver el directorio, ni los archivos puede ser que el disco al que pertenece la clave no esté disponible, por lo que deberá terminar la sesión y preguntar a que hora estará disponible el disco.
En caso de cualquier problema con el equipo lo conveniente será consultar a los encargados en el mostrador de reservaciones o al asesor que se encuentre en ese momento en las terminales.

1.6. MANEJO DE PANTALLA

La terminal VT100 de la computadora tiene ciertas características que se pueden modificar para una sesión y otras más que se pueden manejar desde programa.

En la parte superior del teclado aparecen unos LEDs y una serie de palabras en inglés en letras pequeñas que son:

1) SET/CLEAR TAB que nos va a permitir borrar o fijar tabuladores de la pantalla.
2) CLEAR ALL TABS que borra todos los tabuladores.
3) LINE/LOCAL que nos permite pasar de modo LINEA a modo LOCAL y viceversa.
4) SETUP A/B nos permite pasar del SET UP A al SET UP B y viceversa.
5) TOGGLE 1/O nos va a permitir modificar las características de la pantalla.
6) TRANSMIT SPEED permitirá modificar la velocidad de transmisión de la terminal.
7) RECEIVE SPEED permitirá modificar la velocidad de recepción de la terminal.
8) 80/132 COLUMNS nos permite cambiar de 80 a 132 columnas y viceversa.
9) RESET ocasiona que todos los cambios que se hayan realizado se desechen y la terminal tome las características de Default.

Para poder hacer estos cambios es necesario oprimir la tecla de SET-UP, y se entra al SET UP A donde se puede ver la posición de los tabuladores.

Para pasar al SET UP B se oprime la tecla 5 donde se observa las velocidades de recepcion y transmision (que deben estar a 9600 en las terminales de Edificio Principal), y 4 conjuntos de 4 bits que determinan las características de la terminal. Para ver su significado se recomienda levantar con mucho cuidado el teclado y ver la parte inferior del mismo.

El significado de cada uno de estos es:

BYTE 1

SCROLL 0=RAPIDO 1=SUAVE
controla la velocidad de desplegado en la pantalla
AUTOREPEAT 0=APAGADO 1=ENCENDIDO
permite la repetición de caracteres al oprimir la tecla respectiva una sola vez.
SCREEN 0=FONDO NEGRO 1=FONDO BLANCO
Cambia el fondo de la pantalla
CURSOR 0=SUBGUION 1=CUADRO
Cambia el caracter del cursor

BYTE 2

MARGIN BELL 0=APAGADO 1=ENCENDIDO
Hace sonar la campana cuando se llega a la columna 72'
KEYCLICK 0=NO SUENA 1=SUENA
Permite que la campana suene cada vez que se oprime una tecla. (pero no igual de intenso que el MARGIN)
ANSI/VT52 0=VT52 1=ANSI
Nos permite simular una terminal VT 52
AUTO XON XOFF 0=APAGADO 1=ENCENDIDO
Señales de control para generar códigos sincronos

BYTE 3

0=# 1=# Libra Esterlina
Hace que la terminal despliegue un # o un caracter de Libra Esterlina(£)
WRAP AROUND 0=NO BAJA 1=SI BAJA
Cuando se llega a la columna 80 el cursor se baja a la siguiente linea.
NEW LINE 0=NO BAJA 1=SI BAJA
Cuando se oprime RETURN el cursor regresa a la columna uno y baja de renglón.
INTERLACE 0=APAGADO 1=ENCENDIDO
Se utiliza para gráficas de alta resolución.

BYTE 4

PARITY SENSE 0=NON 1=PAR
Determina que tipo de paridad se va a verificar
PARITY 0=APAGADA 1=ENCENDIDA
Indica si se verifica la paridad
BITS PER CHAR 0=7 (ASCII) 1=8 (EBCDIC)
Nos permite seleccionar el código de transmisión de la información. La VAX trabaja en ASCII.
POWER 0=60 HZ 1=50 HZ
Indica la frecuencia de la energía eléctrica

Para modificar alguna de las características es necesario avanzar con las flechas que están en el teclado hasta estar arriba del bit que queremos modificar y entonces modificarlo con la tecla 6.

Si se desea hacer pruebas del SET UP, se recomienda hacerlas cambiando primero a modo local (estando en SET UP A) y que al final de la sesión le des RESET (SET UP A con 0)

Existen además secuencias de teclas que nos permiten el manejo de la pantalla desde un programa. Para ver que hacen se recomienda trabajar en modo LOCAL antes de empezar a programarlas. Las más comunes son:

ESC[PnA	sube el cursor n líneas
ESC[PnB	baja el cursor n líneas
ESC[PnC	avanza el cursor n posiciones
ESC[PnD	retrocede el cursor n posiciones
ESC[Pl;PcH	mueve el cursor a la línea y columna indicadas por los dos parámetros.
ESC[Pl;Pcf	mueve el cursor a la posición indicada por los dos parámetros
ESC#3	agranda la mitad superior de las letras que se encuentren en esa línea
ESC#4	agranda la mitad inferior de las letras que se encuentren en esa línea
ESC#5	regresa la línea al tamaño normal
ESC#6	agranda a doble ancho las letras que se encuentren en esa línea
ESC[PsJ	borra caracteres en función de s : 0 = borra desde la posición actual hasta el final de la pantalla 1 = borra del inicio de la pantalla hasta la posición actual 2 = borra toda la pantalla pero el cursor no se mueve de su posición.

```
ESC[PsK      borra caracteres en la línea para  s  :
               0 = borra desde la posición hasta el final
               1 = borra del inicio de la línea hasta la
                   posición el cursor
               2 = borra toda la línea
ESC H        fija el tabulador donde está el cursor
ESC[Psg      borra los tabuladores, si  s = 0  borra el
             tabulador en la posición del cursor y si s=3
             borra todos los tabuladores.
ESC[Ps;...;Psm  permite modificar parte de la pantalla en
             función de  s :
               0 = atributos apagados
               1 = incrementa la intensidad
               4 = subraya
               5 = parpadea
               7 = con fondo blanco
ESC[Pt;Pb r  define la región del 'scroll' con t=línea
             superior y b=línea inferior.
```

2
PRIMERAS INSTRUCCIONES DE COMANDO

EL SISTEMA VAX 11-780 UTILIZA UN SIMBOLO PARA ESPECIFICAR SUS DISTINTOS MODOS DE OPERACION, ENTRE ESTOS SE TIENE:

COMANDOS DEL SISTEMA Caracterizada por un signo de pesos a la izquierda.

$

COMANDOS DEL EDITOR Caracterizada por un asterisco a la izquierda.

*

DENTRO DEL EDITOR Caracterizada por el mensaje de [EOB] en la parte inferior de la pantalla.

[EOB]

2.1. Dentro de las instrucciones básicas de operación del sistema VAX debemos considerar DIR y EDT. La instrucción DIR es la forma abreviada de DIRECTORY. DIR es una de las instruciones más frecuentes que se utilizará al estar operando el sistema. Este DIRECTORIO muestra el número de de programas y archivos que se tienen, sus nombres, características y versiones. Cada nombre de archivo se estructura de la siguiente manera.

DISPOSITIVO:[DIRECTORIO]NOMBRE-DE-ARCHIVO.EXTENSION;VERSION

El dispositivo es el nombre donde está físicamente guardado el archivo esto es disco, cinta, diskette, tarjetas; este nombre se forma con DB para disco, LP para impresora, MT para cinta magnética, DY para diskette, CR para la lectora, y va sesuido del nombre del controlador que consta de una letra (A,B,C, etc.) y el número que ocupa dentro del controlador (0,1,2,3, etc.) por lo que para la configuración actual de la VAX los nombres son:

DBA0: para el disco cero
DBA1: para el disco uno
DBA2: para el disco dos
LPA0: para la impresora
LPB0: para la impresora graficadora
CRA0: para la lectora de tarjetas
DYA0: para el dispositivo de diskettes
MTA0: para la cinta magnética
TTA0: a TTA7:, TTB0: a TTB7:, y TTC0: a TTC7: para las terminales

El nombre del directorio es el mismo que el del USERNAME pero puede ser también el de alsún subdirectorio (posteriormente se verá qué es y como se definen éstos)

El nombre del archivo lo asigna el usuario mediante un nombre que indique la finalidad de dicho archivo, este puede ser de 1 a 9 letras y números sin incluir caracteres especiales. La extensión(tipo) también se la asigna el usuario cuando se piensa escribir un programa en algún lenguaje en particular, ya sea BASIC, FORTRAN, PASCAL o DATOS. El usuario deberá asignar la extensión con las primeras tres letras del lenguaje que se esté utilizando esto es; suponsamos que queremos dar el nombre a un programa, a este se le va a llamar EJEMPLO y se escribirá en varios lenguajes:

```
BASIC       EJEMPLO.BAS
FORTRAN     EJEMPLO.FOR
COBOL       EJEMPLO.COB
PASCAL      EJEMPLO.PAS
```

Ejemplos de nombres de archivos:

```
DBA1:[CLASE]EJEMPLO.BAS;3
MTA0:[CINTA]TRABAJO.FOR;6
DBA2:[JUAN[10]]DATOS.PAS;4
```

CONSIDERACIONES:

Si no se proporciona el dispositivo, la computadora asume que se trata del dispositivo donde está asignada la clave; si no se proporciona el directorio se asume que es el de la clave; y si no se proporciona la versión, la computadora asume que queremos la última versión del archivo. Con todo lo anterior los nombres se pueden simplificar a:

```
EJEMPLO.BAS
DATOS.PAS
```

Para saber que archivos se tienen en el directorio o subdirectorio, existe el comando DIRECTORY que nos proporcionará la lista o el mensaje correspondiente.

Su forma abreviada es $ DIR <return>

Otras dos instrucciones útiles son CREATE DIR y SET DEF.

La primera permite crear subdirectorios dentro del espacio en disco. Esto es recomendable cuando la clave es compartida entre dos usuarios y/o se va a tener archivos con programas de diferentes asignaturas. El CREATE DIR se debe usar una sola vez.

Su formato general es:
$ CREATE/DIRECTORY especificación del directorio
donde especificación es [directorio.subdirectorio]
Ejemplo:
$ CREATE/DIR [CURSOS.EGK] <return>

Se crea el subdirectorio [.EGK] (iniciales de Ernesto Gutierrez Kuri) en el directorio [CURSOS]
$ CREATE/DIR [AMH200.BASIC]
Se crea el subdirectorio BASIC en la clave(directorio) [AMH200].

Para poder trasladarnos a éstos subdirectorios debemos emplear el comando SET DEFAULT.

$ SET DEFAULT [directorio.subdirectorio]
$ SET DEF [CURSOS.EGK] <return>
$ SET DEF [.BASIC] <return>

En el primer ejemplo del directorio [CURSOS] se va al subdirectorio [.EGK], y en el segundo ejemplo se le indica de donde se esté, se quiere ir al subdirectorio [.BASIC], en este caso si no se escribe el nombre del directorio, asume que es en el directorio en que se encuentra actualmente. Mientras no se este muy familiarizado con el uso del equipo es conveniente utilizar la forma completa.

Para regresar al directorio principal basta escribir:
$ SET DEF [CURSOS] si nuestra clave es CURSOS o
$ SET DEF [AMH200] si nuestra clave es AMH200, etc.

Para saber en que subdirectorio se encuentra el usuario, sobre todo cuando se tienen varios, será necesario utilizar la instrucción

$ SHOW DEFAULT que lo proporcionará.

Asociados al comando SHOW hay muchos parámetros, entre los cuales están:

$ SHOW TIME que proporciona la fecha y la hora.

$ SHOW QUOTA que dá el UIC, cuantos bloques se han utilizado, cuantos quedan disponibles y en que disco se está trabajando.

$ SHOW DAYTIME es equivalente a $ SHOW TIME.

2.2. COMANDO EDT

Se utiliza cuando se quiere crear algún programa o archivo, por lo que se deberásescribir el comendo EDT cuando se esté en modo comando del sistema, esto es que la maquina genere un signo de $ a la izquierda.
$ Se escribe EDT...

```
$ EDT <return>          <return> IMPLICA PRESIONAR LA TECLA
                                 RETURN ! ! !
```

Y aparece en el display... $-file:
En este momento se tienes la capacidad de dar el nombre y la extension al programa, lo cual se deberá hacer y presionar <return>.

En ese momento aparece el letrero:

```
Input file does no exist [EOB]
*
```

En este momento se deja de estar en DCL para estar en Editor; en el MODO EDITOR se puede crear y/o modificar archivos mediante los comandos propios del Editor que son:

CHANGE	CLEAR	COPY	DEFINE
DELETE	EXIT	FILL	
FIND	HELP	INCLUDE	INSERT
JOURNAL	KEYPAD	MOVE	
PRINT	QUIT	RANGE	REPLACE
RESEQUENCE	SET	SHOW	
SUBSTITUTE	TABS	TYPE	WRITE

Uno de los comandos más importantes es el HELP que ayuda a recordar el formato de los comendos, hay que entrar al Editor para ver su uso.

```
$ EDT AA. <return>

Input file does not exist
[EOB]

* HELP  <return>
HELP

You can set help on a topic by typing

     HELP topic subtopic subsubtopic...
```

```
A topic can have one of the following forms:

     1. An alphanumeric string
          (e.g. a command name, option, etc.)
     2. The match-all or wild card symbol (*)

     Examples:        HELP SUBSTITUTE NEXT
                      HELP CHANGE SUBCOMMAND
                      HELP CH

If a topic is abbreviated, the text for all topics
which match the abbreviation is displayed.

Additional information available:

CHANGE      CLEAR       COPY        DEFINE      DELETE
EXIT        FILL        FIND        HELP        INCLUDE
INSERT      JOURNAL     KEYPAD      MOVE        PRINT
QUIT        RANGE       REPLACE     RESEQUENCE  SET
SHOW        SUBSTITUTE  TABS        TYPE        WRITE


* EXIT

$ DBA2:[JORGEJAA.;1 No lines
```

Los comandos más utilizados son:

```
* INSERT
Su formato general es:
* INSERT [rango] [;  línea a ser insertada]
Donde rango puede ser:
 1) un número de línea
 2) la palabra BEGIN que indica el inicio del archivo
 3) la palabra END que indica el fin del archivo
 4) cualquier combinación de éstas separadas por dos puntos
    (:) o la palabra THRU.
 5) la palabra WHOLE que significa todo
 6) Rango + número
 7) Rango - número
 8) 'string' a buscar
 9) LAST
10) .  ( línea actual)
11) Número de línea   cuantas más
12) BEFORE
13) REST
```

Todo lo que está encerrado entre [] significa que es opcional, si no se especifica rango se asume la línea actual y si no se especifica línea a ser insertada se asume que es un conjunto de ellas que se darán a continuación.

Ejemplos:
* I 8 agrega líneas antes de la línea numerada con 8
* I 8; B = 2 * A
inserta la línea que aparece en el comando antes de la 8
Para salirse de INSERT se oprimen simultaneamente CRTL y Z
* INSERT
Permite introducir el archivo (programa), el cursor inicia en la columna 10 aproximadamente pero es la primera columna del archivo. Cada vez que damos <return> el cursor pasa a la siguiente línea. Para terminar hay que utilizar CTRL/Z.

* SUBSTITUTE
Permite corregir el archivo.
Su forma general es:
* SUBSTITUTE/string1/string2/[rango][BRIEF:n][/QUERY]
donde string1 es lo que está incorrecto, string2 es como se desea que quede, rango se explicó en el comando anterior. BRIEF:n permite indicar que no despliegue toda la línea sino unicamente 'n' caracteres de ésta y el calificador QUERY le indica a la computadora que se quiere confirmar antes de hacer cada substitución y las posibles respuestas son: Y(si), N(no), A(todas las restantes), y Q(que ya no efectúe ningún cambio más).

Ejemplos:
* S/herror/error/5
En este caso solo la línea 5 se modifica
* S/alumno/estudiante/WHOLE/QUERY
Se desea que efectúe las substituciones en todo el archivo pero que vaya preguntado antes de hacerlas.

* REPLACE
Permite substituir líneas existentes por las que se van a teclear a continuación; al terminar hay que dar CTRL/Z.
* REPLACE [rango][;línea a ser insertada]

* MOVE
Mueve líneas de texto de un lugar a otro.
* MOVE [rango1] TO [rango2] [/QUERY]
donde rango1 es el conjunto de líneas que se quieren mover y rango2 es el lugar donde se quieren que estén.

* COPY
Esta instrucción permite duplicar un conjunto de líneas en otro lugar del archivo.
* COPY [rango1] TO [rango2] [/QUERY] [/DUPLICATE:n]
Donde n es el número de veces que se desea duplicar el texto.

* RESEQUENCE
Cuando se desea que la numeración interna del archivo se normalice o modifique, se puede usar ésta instrucción.
* RESEQUENCE [rango] [/SEQUENCE] [:inicio [:incremento]]
* RES 2:5/SEQ :2:3
Se está pidiendo que el bloque de líneas que va del 2 al 5 se renumere iniciando en 2 y con incrementos de 3 en 3. Se hace notar que entre 2 y 5 pueden existir una gran cantidad de líneas.

* TYPE
Permite ver en la terminal el archivo, con la numeración interna proporcionada por el Editor.
* TYPE [rango] [/BRIEF:n] [/STAY]
donde el calificador /STAY ocasiona que el apuntador de línea no avance de su posición actual.
* TYPE 'string' busca la ocurrencia del string y nos despliega la línea; si se añade ALL desplegará todas las líneas que contengan el string

* FIND
Este comando permite ubicarse en una línea, esto es, el apuntador de líneas se mueve a ésta.
* FIND [rango]

* DELETE
Permite borrar líneas del archivo
* DELETE [rango]

* EXIT
Permite salirse del Editor salvando el archivo en el disco.

* QUIT
Se sale del Editor pero se desecha todo lo realizado en la sesión de Edición.

* INCLUDE
Con este comando se puede traer un archivo en el disco e incluirlo en el archivo que se está editando.
* INCLUDE nombre-de-archivo [rango]

* PRINT Manda al espacio en disco una copia del archivo que se está editando, pero incluye la numeración interna.
* PRINT nombre-de-archivo [rango]

* WRITE
Crea un archivo en nuestro espacio en disco, sirve para crear protecciones temporales cuando el sistema puede tener fallas continuas y nuestros archivos son muy grandes.
* WRITE nombre-de-archivo [rango] [/SEQ [:inicio[:inc.]]]

* CHANGE Este comando permite cambiarse al otro modo del Editor pero que consume más recursos de computadora.

Estaremos en modo INSERT.

*C <return>

Esto deja 'una hoja en blanco' donde se podrán escribir programas y archivos. Dejará en la pantalla un mensaje que dice [EOB] y en ese momento, el usuario estará dentro del EDITOR de caracteres, habiendose dejado el modo línea del COMANDO EDITOR. Este mensaje de [EOB] indica la dimensión del archivo en el que se está operando. Cuando se esté escribiendo un programa el usuario deberá tomar en cuenta los formatos de los diferentes lenguajes. Cuando se termine de escribir el programa se dará el comando de salida del modo CARACTER al modo LINEA, mediante el control CTRL-Z oprimiéndolas al mismo tiempo, con lo que se está ya en modo LINEA. Para poder salir a DCL(comando del sistema) se tienen dos opciones principales que son:

QUIT o EXIT esto es:

*QUIT <return> o
*EXIT <return>

La instrucción QUIT borra lo que se haya escrito, si es la primera vez que se llama por EDT al programa, en caso contrario deja la versión anterior, y el comando EXIT graba el archivo o programa en el area de trabajo por lo que se podrá contar con el en el directorio.

3
COMPILACION

3.1. INTRODUCCION A LA COMPILACION

La compilación es el procedimiento mediante el cual la computadora cambia los programas que se crearon mediante el EDITOR, a lenguaje máquina o binario, realizando la detección de errores durante este proceso. Este procedimiento, cuando se realiza en VAX consta de dos partes diferentes, la primera genera las extensiones OBJ u objeto que son la traducción de los programas realizada por el compilador, en esta parte es donde se identifican los errores de programación que son generalmente mala sintaxis de los verbos del programa o instrucciones inválidas. Suponiendo que el programa no tuvo errores, el compilador, el cual es el traductor, realiza la traduccion perfectamente; entonces se debe generar, mediante la instrucción LINK, la extensión EXE del programa; en este momento tu programa esta listo para correr. Lo que hace el LINK es ligar el módulo objeto con otros módulos objetos necesarios para la corrida, éstos módulos pueden ser creados por los usuarios o por el sistema.

Este procedimiento es:

```
-------------------------------------------------
: PROGRAMA ---> PROGRAMA OBJETO ---> PROGRAMA   :
: ESCRITO                            EJECUTABLE :
-------------------------------------------------
```

3.2. PASOS A SEGUIR SEGUN LENGUAJE DE PROGRAMACION:

PROGRAMA ESCRITO EN	PROGRAMA OBJETO	PROGRAMA EJECUTABLE
BASIC	$ BAS nombre-pros <ret>	$ LINK nombre-pros <ret>
FORTRAN	$ FOR nombre-pros <ret>	$ LINK nombre-pros <ret>
PASCAL	$ PAS nombre-pros <ret>	$ LINK nombre-pros <ret>
COBOL	$ COB nombre-pros <ret>	$ LINK nombre-pros <ret>
COBOL/ANSI	$ COB/ANSI n-pros <ret>	$ LINK nombre-pros <ret>

Después de realizar el paso de programa escrito a programa objeto pueden aparecer errores, en este momento se deben corregir para poder correr el programa.

4
CORRECCION Y EJECUCION DE LOS PROGRAMAS

4.1. EDITOR DE ERRORES:

Al efectuarse el proceso de compilación, es probable que salgan una serie de mensajes de error, esto durante la fase de conversión de programa ESCRITO a programa OBJETO. Para realizar la corrección de dichos errores, se debe utilizar la instrucción EDT para pasar de COMANDOS DEL SISTEMA a COMANDOS DE EDITOR y llegar a estar dentro del EDITOR. Todo esto viene en la parte dos de este manual. La forma en que se utilizará el comando EDT es dando el nombre de programa y la extensión del programa que tenga los errores, todo esto bajo el siguiente formato:

$ EDT nombre-del-programa.extension <return>

Ya una vez que el usuario se haya posicionado dentro del EDITOR por lo descrito en la parte tres de este manual, la correción del programa se puede efectuar mediante los comandos del Editor en modo línea. El sistema VAX 11-780 tiene una serie de funciones las cuales le ayudan al usuario a realizar una versatil corrección de los programas, llamadas FUNCIONES EDITOR en modo caracter. Es conveniente que el usuario primero se familiarice con los comandos de modo línea antes de querer utilizar el modo caracter. Las mas elementales son la tecla DELETE, la cual indica borrado de derecha a izquierda, y las cuatro flechas con las cuales se puede mover sobre lo escrito en las cuatro direcciones. Es importante recordar el no dejar pegadas las flechas ya que son repetitivas y la velocidad de transmisión del sistema es muy rápida; esto puede originar que se destruyan los archivos. Las FUNCIONES EDITOR se describirán en el siguiente cuadro y se refieren al KEYPAD(teclado auxiliar situado a la derecha del teclado:

FUNCION	CARACTERISTICAS
PF 1	Permite efectuar la operación alterna de cada tecla.
PF 2	Despliega todas las funciones del KEYPAD (HELP)

PF 3	Se posiciona en el siguiente STRING que se haya pedido que busque o dá el mensaje cuando no se encontro.
PF 1 PF 3	Aparece un mensaje que dice Search for.; se puede dar cualquier STRING que se deseé buscar y necesita darle la dirección de búsqueda.
PF 4	Borra línea por línea, estando situado al principio de ésta
PF 1 PF 4	Imprime la última línea que se haya borrado con PF 4 estando situado al lado izquierdo del display
,	Borra caracter por caracter de izquierda a derecha
PF1 ,	Imprime en el display el último caracter que se haya borrado con ,
-	Borra palabra por palabra de izquierda a derecha, esto es de blanco a blanco, los caracteres entre espacios
	Imprime en el display la última palabra que [illegible] con -
.	Avanza a la siguiente página, y depende de [illegible]ción actual encendida
PF 1 7	Permite ejecutar un comando del editor en modo línea, hay que orrimir <enter>
8	Avanza secciones de 16 renglones
PF 1 8	El texto seleccionado es formateado para llenar el número de columnas que se fije con SET WRAP (modo línea).
9	El párrafo seleccionado es guardado al final del BUFFER
PF 1 9	Substituye el texto seleccionado por el texto contenido en el BUFFER.
4	Indica que todos los movimiento deseados son hacia el final del archivo

PF 1 4	Se sitúa en la parte inferior del archivo
5	Indica que todos los movimientos deseados son hacia el inicio del archivo
PF 1 5	Se sitúa en la parte superior del archivo
6	Corta el párrafo que se haya seleccionado y lo guarda en un BUFFER.
PF1 6	Imprime el último párrafo que se haya selecionado y guardado en el BUFFER
1	avanza una palabra (en la dirección seleccionada
PF 1 1	Abre una línea donde esté el cursor para insertar texto.
2	Se posiciona al final de la línea donde se encuentre el cursor
PF 1 2	Borra todos los caracteres desde donde está el cursor hasta el fin de la línea
3	Avanza un caracter (en la dirección seleccionada)
PF1 3	Permite insertar un caracter ASCII
.	Inicia la selección de un párrafo que se quiera borrar o mover de lugar, abarcandolo con las flechas guia.
PF1 .	Cancela la selección del párrafo
ENTER	Permite que el comando de Editor modo línea indicado después de PF 1 7 sea ejecutado por la computadora
PF1 ENTER	substituye el STRING buscado por el texto en el BUFFER y busca la siguiente ocurrencia del STRING

Este diagrama en forma gráfica del KEYPAD es:

PF1	PF2	PF3	PF4
GOLD	HELP	FIND NEXT	DEL LINE
		find	und line
7	8	9	-
PAGE	SECTION	APPEND	DEL WORD
command	fill	replace	und word
4	5	6	,
ADVANCE	BACKUP	CUT	DEL CHAR
bottom	top	paste	und char
1	2	3	
WORD	EOL	CHAR	ENTER
chng case	del eol	specins	
	0	.	
LINE		SELECT	
open line		reset	subs

Al estar en Editor modo caracter pueden aparecer en el texto los mensajes del sistema. Para borrarlos hay que teclear CTRL/U.

4.2. VERSIONES .LIS/LIS:

Para ver los resultados de la compilación del programa, el ususario tiene una gran ventaja, las versiones .LIS. Estas versiones LIS se crean agregando /LIS en el proceso de compilacion, lo cual crea en el directorio los archivos NOMBRE-PROG.LIS las cuales se pueden ver mediante la instrucción TYPE.

Este cuadro es:

PROGRAMA ESCRITO	PROGRAMA OBJETO
BASIC	$ BAS/LIS nombre-pros
FORTRAN	$ FOR/LIS nombre-pros
PASCAL	$ PAS/LIS nombre-pros
COBOL	$ COB/LIS nombre-pros
COBOL/ANSI	$ COB/ANSI/LIS n-pros

COMANDO TYPE. Lo utiliza el ususario cuando está dentro de DCL.

Formato General:

$ TYPE nombre-del-programa.extensión;versión<return>

Esta instrucción despliega los archivos (programas, datos, resultados) en la pantalla sin ningún efecto para ellos; NO ES EL COMANDO TYPE DEL EDITOR.

Si el ususario pide un TYPE de un archivo .LIS generado en la compilación de su programa, aparecerá el listado del programa y los errores de sintaxis, si los hubo, intercalados en las líneas del listado. Cuando sean demasiados errores conviene imprimir el listado.

4.3. DANDO DATOS POR TERMINAL.

Cuando el usuario mande ejecutar su programa (imagen o módulo .EXE), deberá proporcionar los datos mediante la terminal y debe hacerlo siguiendo el formato especificado en el programa.

La forma de correr el programa es:

$ RUN nombre-del-programa <return>

Si el programa es en BASIC aparecerá una interrogación(?) pidiendo los datos, si es FORTRAN no aparecerá nada antes del cursor.

Si se desea interrumpir la ejecución del programa se puede hacer mediante las teclas CTRL-Y, CTRL-Z o CTRL-C dependiendo de la situación que se esté manifestando.

Para saber el tiempo de procesador que está consumiendo el programa el usuario deberá teclear CTRL-T.

4.4. CREACION DE ARCHIVOS DE DATOS

Cuando el programa requiere muchos datos es mejor crear archivos de datos (.DAT) mediante el editor para, posteriormente, asignarlos al canal de entrada de datos de tu programa. El formato para crearlo es:

$ EDT nombre-de-archivo.DAT <return>

Y crear el archivo que contenga los datos siguiendo los formatos indicados en el programa. Posteriormente se deberá asignar a SYS$INPUT mediante la instrucción ASSIGN que se verá posteriormente.

OTRAS INSTRUCIONES Y MODOS

5.1. OTRAS INSTRUCIONES:

Existe una gama muy amplia de comandos de sistema que se pueden usar en el sistema VAX 11-780 tales como SHOW, SET, etc..
Todos estos están resumidos en una biblioteca a la que se tiene acceso mediante el comado HELP, en el cual están los formatos de los comandos de DCL.
Algunas instrucciones están restringidas a la mayoría de los usuarios pero aun así hay gran versatilidad.

El formato de HELP es:

$ HELP <return>

A continuación aparece un desplegado con todos los comandos que existen en DCL y se puede pedir su sintaxis escribiendo el comando después del mensaje que dice Topic?.

La computadodora dará al usuario una pequeña explicación del comando y proporcionará información adicional que puede explicar. Si así se desea, el usuario deberá escribirla después de la palabra Subtopic?

La forma de salirse del HELP es mediante <return> sucesivos hasta que aparezca el signo de $.

Es conveniente que las primeras veces que se use la VAX se compenetre en el uso de este comando de DCL.

$ HELP <return>

Information available:

ACCOUNTING	ALLOCATE	ANALYZE	APPEND
ASSIGN	ATTACH	BACKUP	BASIC
BLISS	CANCEL	CC	CLOSE
COBOL	CONTINUE	CONVERT	COPY
CORAL	CREATE	DBO	DDL
DEALLOCATE	DEASSIGN	DEBUG	DECK
DEFINE	DELETE	DEPOSIT	DIFFERENCES
DIRECTORY	DISMOUNT	DUMP	EDIT
EOD	EOJ	Errors	

```
  EXAMINE       EXIT          FDL           FORTRAN
  GOTO          HELP          IF            INITIALIZE
  INQUIRE       JOB           Lexical
  LIBRARY       LINK          Login         LOGOUT
  MACRO         MAIL          MCR           MERGE
  MESSAGE       MONITOR       MOUNT         ON
  OPEN          PASCAL        PASSWORD      PATCH
  PHONE         PLI           PRINT         Procedure
  PURGE         Queues        READ          RENAME
  REPLY         REQUEST       RMS           RTL
  RUN           RUNOFF        SEARCH        SET
  SHOW          SORT          SPAWN
  Specify       START         STOP          SUBMIT
  Symbol_Assign               SYNCHRONIZE
  System        TECO          TYPE          UNLOCK
  WAIT          WRITE

Topic? dir <return>

DIRECTORY

   Provides a list of files or information about a
  file or group of files.

   Format:

        DIRECTORY   [file-spec[,...]]

  Additional information available:

  Parameters Qualifiers
  /BEFORE       /BRIEF        /COLUMN       /CREATED
  /DATE         /EXCLUDE      /EXPIRED      /FULL
  /HEADING      /MODIFIED     /OUTPUT       /OWNER
  /PRINTER      /PROTECTION                 /SINCE
  /SIZE         /TOTAL        /TRAILING     /VERSIONS


DIRECTORY Subtopic? /full <return>

DIRECTORY

  /FULL

   Lists full file attributes with each file.

   The /FULL qualifier overrides the default brief listing
format.


DIRECTORY Subtopic? <return>
```

Topic? <return>

$

5.2. BASIC INTERACTIVO

Una de las dos formas de trabajar en lenguaje BASIC en la VAX es mediante BASIC interactivo. Para lograr utilizar este BASIC es necesario invocarlo mediante el comando de DCL

$ BASIC <return>.

Este modo de trabajar en BASIC puede ser similar al utilizado en las microcomputadoras, con la diferencia de que éste BASIC compila el programa aunque no guarde el código objeto en el directorio. Esto es muy útil cuando se está probando o depurando un programa.

Formato:

$ BASIC <return>

A partir de este momento se está en el Ambiente BASIC y el sistema enviará un mensaje que dice:

VAX-11 BASIC V2.0

Ready

Después de esto, el usuario podrá dar un conjunto de instrucciones entre las que estan :

HELP de Basic
OLD
NEW

OLD <return>.

Permite traer al ambiente BASIC un archivo que exista en el directorio y que debe ser del tipo .BAS. Este archivo contiene un programa en lenguaje BASIC que se quiera correr.

NEW <return> Hace que BASIC borre el archivo existente en el ambiente y pida el nombre del nuevo programa BASIC que se va a crear. Este nombre debe tener entre uno y nueve caracteres

Formato:
NEW <return>

New file name-----Nombre_del_programa

Se pone el Nombre_del_programa y no hace falta ponerle la extension.

Cuando el programa es OLD la computadora envia el mensaje de Ready y cuando es NEW deja el cursor en la parte inferior del Display esperando que el usuario empiece a escribir el programa.

Cuando se termine de escribirlo podrá correrlo inmediatamente después mediante la instrucción RUN o RUNNH. Si existe algún tipo de error el BASIC lo indicará sin haber iniciado la ejecución.
Este error se podrá corregir con el comando EDIT del ambiente BASIC y tratar de ejecutar de nuevo el programa.

La instruccion LIST despliega el programa en la terminal.

Para grabar el programa en disco es necesario que se le de la instrucción SAVE para que se almacene en el directorio.

Para salir del ambiente BASIC se deberá dar la instrucción EXIT.

Ejemplo:

$ BASIC <return>

VAX-11 BASIC V2.0

Ready<LF>

help

HELP

The HELP command displays on-line information about BASIC statements, commands, directives, functions, conventions, and other topics. Type HELP to see a list of topics. Then enter a subtopic for more information. If you type a question mark in response to the prompt for a topic, BASIC displays the list of available topics.

```
Additional information available:

ARRAYS      CHARACTER   COMMANDS
COMMENTS    CONSTANTS   CONVENTIONS
DATA_TYPES  DIRECTIVES  ERRORS
 EXPRESSIONS             FUNCTIONS  HELP
IMMEDIATE   LABELS      LINE
MODIFIERS   QUALIFIERS  STATEMENTS  VARIABLES

Topic? stat

STATEMENTS


    Statements assign values, perform I/O, transfer
program control, and so forth. Program statements
are associated with a line number and stored for later
execution. A statement starting in the first column and
having no line number is executed as an immediate
mode statement. For additional information on
immediate mode statements, type "HELP IMMEDIATE".

    Statement modifiers are keywords that qualify or
restrict a statement. For help on modifiers, type "HELP
MODIFIERS".


  Additional information available:

  CALL          CHAIN         CHANGE        CLOSE
  COMMON        DATA          DECLARE
  DEF           DELETE        DIMENSION     END
  EXIT          EXTERNAL      FIND
  FNEND         FNEXIT        FOR           FREE
  FUNCTION      FUNCTIONEND
  FUNCTIONEXIT                GET           GOSUB
  GOTO          IF            INPUT
  ITERATE       KILL          LET           LINPUT
  LSET          MAP           MARGIN
  MAT           MOVE          NAME_AS       NEXT
  NOMARGIN      ON            OPEN
  OPTION        PRINT         PUT           RANDOMIZE
  READ          RECORD        REM
  REMAP         RESTORE       RESUME        RETURN
  RSET          SCRATCH       SELECT
  SLEEP         STOP          SUB           SUBEND
  SUBEXIT       UNLESS        UNLOCK
  UNTIL         UPDATE        WAIT          WHILE

STATEMENTS Subtopic? if

  IF
```

```
     The IF statement evaluates a conditional expression
and
     transfers program control, depending on the
resulting
     value.

     Format

     Conditional:

               {   statement...}}
               {THEN {lin-num   }}[     {lin-num      } .
IF cond-exp    {                 }[ELSE {statement..}] [END IF]
               {GOTO target     }

     Statement Modifier:

          statement IF cond-exp

     Examples

     100       IF A > 0
               THEN PRINT A
               ELSE PRINT "A IS NOT GREATER THAN 0"
               END IF

     200       PRINT A IF A > 0


STATEMENTS Subtopic?  <return>

Topic?  <return>


Ready <LF>

exit

$
```

IMPRESION DE ARCHIVOS

6.1. INSTRUCCION PRINT Y COPY

LA instrucción PRINT permite asignar a la impresora archivos y programa para su impresión. Este es una instrucción de COMANDOS DEL SISTEMA(DCL) por lo tanto se debe dar cuando este el signo de $.

El formato general es:

$ PRINT nombre_del_programa_o_archivo.extension;versión

Después el sistema genera un aviso de entrada de impresión:

Job nnn entered on SYS$PRINT

Donde nnn es el numero de listado de impresión que le asignó al programa.

Cuando el listado ha terminado de salir por al impresora, la computadora enviará el mensaje correspondiente.

Si se desea escribir varios archivos bajo la misma carátula de salida lo que se tiene que hacer es asignarlos despues de la instrucción PRINT separados por comas(,) o por mas(+).

El formato es:

$PRINT archivo_uno,archivo_dos+archivo_tres,...archivo_n <return>

Si no se escribe la versión, la computadora entiende que se quiere mandar a imprimir la última versión.

Está restringido por sistema y no tiene sentido mandar a imprimir archivos del tipo .OBJ y .EXE.

INSTRUCCION COPY

Esta instrucción pertenece a las instrucciones de DCL mediante la cual se puede asignar a un solo archivo un conjunto de ellos , o copiar archivos de un subdirectorio a otro.

El formato es:

$COPY <return>
$_from: que_archivo_vas_a_asignar <return>
$_to: a_que_archivo_lo_vas_a_asignar <return>

En algunos casos el sistema enviará un mensaje de incompatibilidad de archivos pero no tiene trascendencia.

En la parte de $to: del formato se debe asignar un nombre con la extensión que se desee, este nuevo archivo estará en el directorio personal mediante el nombre que se designó y contendrá todos los archivos que se determinaron en el $from: .

6.2. ASIGNACION DE INPUT Y OUTPUT:

Para poder entender perfectamente esta parte del manual es necesario hacer algunas aclaraciones:

Normalmente la salida de resultados de la corrida de un programa y la entrada de datos para poder efectuar la corrida de éste, está asignada a pantalla. Lo que se va a realizar en este caso es cambiar esa asignacion normal del sistema, de pantalla a una asignación de un archivo personal en disco.

El sistema de salida es el SYS$OUTPUT y el de entrada es el SYS$INPUT.

La aplicación práctica es el almacenar los resultados de un programa en un archivo SYS$OUTPUT y el tener los datos de un programa grabados en un archivo mediante la instrucción SYS$INPUT.

Las instrucciones de COMANDOS DE SISTEMA: ASSIGN SYS$OUTPUT y ASSIGN SYS$INPUT nos permiten crear archivos con los resultados de nuestros programas y asignar los datos contenidos en un archivo .DAT a un programa.

Toda instrucción ASSIGN debe llevar un respectivo DEASSIGN, esto es: debemos asignar de entrada SYS$INPUT o de salida SYS$OUTPUT y desasignar posteriormente con un DEASSIGN.

El formato general de SYS$OUTPUT:

```
$ASSIGN nombre_prog.RES SYS$OUTPUT <return>
$RUN nombre_del_programa <return>
$DEASS SYS$OUTPUT <return>
```

Si se desea se puede poner .LIS en lugar de .RES, es indiferente.

El archivo a imprimir es nombreprog.RES (o nombre.LIS si así se indicó) y aparecerá en el directorio y se puede mandar imprimir en el se encuentran los resultados.

Formato del SYS$INPUT:

```
$ASS nombre_de_archivo.DAT SYS$INPUT <return>
$RUN nombre_de_programa
$DEASS SYS$INPUT
```

El programa correrá con los datos contenidos en nombre_de_archivo.DAT.

Hay que aclarar que cuando se este en modo SYS$OUTPUT no aparece ningun resultado en pantalla sino que se enviarán a un archivo de salida.

7

MANTENIMIENTO DE LA BIBLIOTECA EN ORDEN

7.1. INSTRUCCIONES PURGE Y DELETE.

Para poder mantener la biblioteca en orden existen dos instrucciones de COMANDOS DEL SISTEMA muy importantes.

INSTRUCCION PURGE:

Esta instrucción depura el directorio dejando unicamente la última versión de cada programa que se tenga en el directorio.

Hay que tener mucho cuidado al utilizar esta instrucción ya que muchas veces la versión que sirve no es la última y si se ejecuta la instruccion PURGE el sistema eliminará la versión útil.

El formato es:

$ PURGE <return>

Aparecerá el signo de $.

INSTRUCCION DELETE:

Esta instrucción permite borrar un determinado archivo o familia de ellos.

El formato es:

$ DELETE nombre_del_archivo.extension;versión <return>

Puede aparecer el signo de $ o un mensaje que indique el hecho de tratar de borrar un archivo que no existe. O que se tiene privilegios para borrar el archivo (como son los subdirectorios)

El asterisco (*) tiene la particularidad de generalizar como sigue:

$ DEL *.NNN;*

Esto borra todas las versiones con esa extension, sin importar el nombre y versión.

$ DEL *.*;N

Borra el archivos cuya versión sea la indicada

$ DEL NNNN.*;*

Borra todas las versiones y extensiones del archivo indicado.

Las instrucciones para borrar pueden ser las mas fáciles de ejecutar, pero tambien son las mas peligrosas ya que se puede llegar a borrar archivos o programas que posteriormente harán falta.

El uso del (*) como generalizador (wild card) es válido en cualquier instrucción de DCL relacionada con archivos. Otro caracter es (%) que generaliza un solo caracter.

Una instrucción que nos permite cambiar el nombre a archivos ya existentes es el $ RENAME nombre-viejo nombre-viejo.

8
SALIENDO DEL SISTEMA

8.1. VERSIONES .JOU E INSTRUCCION RECOVER:

Durante una sesión de trabajo en el sistema VAX 11-780 puede llegar a suceder una interrupción de corriente eléctrica, o que ocurra otro tipo de falla. Esto no tiene gran trascendencia si se está en DCL pero si se encuentra dentro del Editor y no se puede volver normalmente al Editor o a DCL entonces toda la sesión de Editor podría haberse perdido, pero la computadora la guarda en un archivo de tipo .JOU.

El archivo o programa afectado por la interrupción quedará en el directorio con una extensión .JOU lo cual viene de la palabra JOURNAL, este tipo de archivos trae una especie de resumen de las operaciones efectuadas sobre dicho archivo en la sesión de Editor. Las versiones .JOU no se deben accesar de la misma forma que los archivos normales que se describen en el comando $ EDT. Lo que se tiene que hacer es utilizar el calificador RECOVER para recuperar la sesion interrumpida.

El formato general es:

$ EDT/RECOVER nombre_del_archivo_original <return>

Esto hará que la computadora reescenifique la sesión de Editor hasta el momento en que ocurrió la falla, esto es, la computadora hará de nuevo todas las operaciones que se realizaron en la sesión de Editor y lo dejará para que se continue.

8.2. INSTRUCCION LOGOFF:

Esta instrucción sirve para despedirse del sistema VAX 11-780, y se debe utilizar al finalizar la sesión de terminal.

Se puede utilizar el Comando LOGOFF, su abreviatura LOG o el símbolo LO.

Formatos:
$ LOGOFF <return>
$ LOG <return>
$ LO <return>
Aparecerá un mensaje de despedida del sistema y la conexion con VAX acaba de concluir.

9

PROCEDIMIENTOS DE COMANDOS

9.1 SIMBOLOS

Los símbolos son nombres de variables, de 1 a 9 caracteres, en las cuales podemos guardar números reales o enteros o strings (que pueden ser comandos de DCL).

Los símbolos pueden ser locales a un procedimiento o pueden ser globales a todos los procedimientos y comandos ejecutados en la sesión. Para almacenar datos en los símbolos se deben seguir las siguientes reglas:

Para valores numéricos locales:

SIMBOLO = valor

Para valores numéricos globales:

SIMBOLO == valor

Para strings locales:

SIMBOLO := string

Para string globales:

SIMBOLO :== string

Mediante estos símbolos se pueden redefinir las instrucciones para simplificar los procesos, pero sólo son válidos durante la sesión y únicamente en la clave del usuario. Ejemplo:

AUXILIO:==HELP

Cada vez que se escriba AUXILIO, la computadora ejecutará el comando HELP.

Si dentro del nombre del símbolo se inserta un asterisco, entonces todos los caracteres que le siguen son opcionales

Ejemplo
AUX*ILIO:==HELP
Con solo teclear AUX se ejecutará el comando HELP.

A nivel DCL se permite ejecutar operaciones con los símbolos. Estas operaciones son: suma, resta, producto y división para los valores numéricos y búsqueda, extracción y concatenación, y substitución de contenido en los strings.

9.2) PROCEDIMIENTOS DE COMANDOS

Un procedimiento de comandos es un archivo que contiene una secuencia de instrucciones de DCL. Este archivo se crea con el Editor y debe ser del tipo .COM. Cada instrucción debe estar precedida por el símbolo de $. Para continuar una instrucción en otra línea hay que escribir un guión al final de la línea y continuar en la siguiente sin poner el signo de $.

Para documentar el procedimiento de comandos se utilizará un signo de admiración (!) para indicar que el resto de la línea es un comentario.

Se deben utilizar los comandos completos, esto es, sin abreviaturas, y se recomienda el uso de la sangría en las líneas para mejorar la legibilidad y comprensión del procedimiento.

Para ejecutar un procedimiento de comandos es necesario preceder el nombre del archivo de comandos, del signo arroba (@) por ejemplo: @LOGIN.

Se puede ejecutarlo también mediante un SUBMIT, que ocasiona que el proceso entre a la cola de BATCH. También se puede ejecutarlo desde otro procedimiento de comandos.

Una manera sencilla de hacerlo, sobre todo si el nombre es muy largo (incluyendo dispositivo, directorio, subdirectorio, etc.) es utilizando un símbolo como sinónimo del procedimiento. Este símbolo puede definirse en el LOGIN.COM.

Para verificar la ejecución del procedimiento se pueden utilizar los comandos SET VERIFY al inicio y SET NOVERIFY al final para que todos los comandos sean desplegados en la terminal

Control de Entrada/Salida de los Procedimientos de Comandos Cuando se inicia una sesión, el sistema operativo (computadora) ejecuta un proceso en el cual se establecen las equivalencias iniciales de los siguientes nombres lógicos:

SYS$INPUT	Es el canal de entrada de datos.
SYS$OUTPUT	Es el canal para desplegar información.
SYS$ERROR	Es el canal para desplegar los mensajes de Error.
SYS$COMMAND	Es el canal para introducir comandos.
SYS$DISK	Es el canal donde se encuentran alojados los archivos.
SYS$LOGIN	Contiene el dispositivo y directorio iniciales al entrar en sesión.

Cuando se inicia una sesión, SYS$INPUT se encuentra asignado a la terminal en la cual se está trabajando. Cuando se ejecuta un procedimiento de comandos, se establece una nueva equivalencia para el nombre lógico SYS$INPUT, la cual será el propio archivo que contiene el procedimiento de comandos. Si se anidan procedimientos de comandos, es decir si es ejecutado un procedimiento dentro de otro procedimiento de comandos, SYS$INPUT es asignado al archivo que corresponde al procedimiento que se esté ejecutando en ese momento.

SYS$COMMAND es hecho equivalente al nivel de comandos en que se encuentre: si se ejecuta un procedimiento de comandos interactivamente, este nombre lógico es asignado a la terminal, mientras que si ejecutado desde batch es asignado a la lectora de tarjetas.

Otra forma de mandar ejecutar procedimientos de comandos es mediante el SUBMIT seguido del nombre del archivo. Esta ejecución entrará a la cola de BATCH.

UTILIZANDO SIMBOLOS EN PROCEDIMIENTOS DE COMANDOS.

Podemos definir un símbolo que contenga el nombre de un archivo y utilizarlo en los comandos encerrado entre apóstrofes (') que le indica a la computadora que efectué la substitución del símbolo por su valor.

El contenido de los símbolos puede tener hasta 255 caracteres y generalmente son nombres de archivos.

En un símbolo se pueden almacenar expresiones de tipo string o numéricas. Por ejemplo:

```
$ CODE = 4 + F$STRING('6') - A
```

donde A es un símbolo previamente definido.

9.3) OPERACIONES Y RELACIONES

Las expresiones pueden ser:

expresión modo	
valor entero	entero
valor string	string
función entera	entero
función string	string
símbolo entero	entero
símbolo string	string
+, - o .NOT. valor	entero
valor .AND. o .OR. valor	entero
string + o - string	string
entero + o - cualquier valor	entero
valor + o - entero	entero
valor * o / cualquier valor	entero
valor (comparación string) valor	entero
valor (comperación aritmética) valor	entero

En la tabla anterior cualquier 'valor entero' o 'valor' es un valor de caracteres string.

Existen algunas funciones para formar las expresiones y entre las más usadas están:

F$EXTRACT	extrae un substring
F$INTEGER	convierte a número un string
F$LENGTH	nos indica la longitud de un string
F$LOCATE	localiza un caracter en un string
F$LOGICAL	traduce un nombre lógico (definido previamente)
F$STRING	convierte un resultado entero a string.

Para asignar expresiones string a símbolos se usa el formato símbolo = expresión string.

Para asignar expresiones enteras a símbolos se emplea el siguiente formato: símbolo = expresión entera.

Los operadores en las expresiones son:

tipo operador	precedencia	operación
lógico .OR.	1	o
.AND.	2	y
.NOT.	3	complemento
comparac. .EQ.	4	= aritmético
aritmét. .GE.	4	>= "
.GT.	4	> "
.LE.	4	<= "
.LT.	4	< "
.NE.	4	<> "
comparac. .EQS.	4	= string
string .GES.	4	>= "
.GTS.	4	> "
.LES.	4	<= "
.LTS.	4	< "
.NES.	4	<> "
Operad. +	5	suma aritmética
arítm. -	5	diferencia "
+	7	signo positivo
-	7	signo negativo
*	6	producto aritmético
/	6	cociente "
Operad. +	5	concatenación string
string -	5	reducción string

Nota: la prioridad es 1 = más baja y 7 = la más alta.

Para utilizar los operadores lógicos se pueden usar valores enteros o sus correspondientes binarios si se precede de un % y una X. Ejemplo:

```
A = %X1000 .OR. %X0001
A =  8     .OR.    1
```

en ambos casos el resultado es 9.

Para controlar la ejecución de los comandos existen una serie de instrucciones que son:

```
IF expresión de comparación THEN comando
GO TO etiqueta
EXIT
STOP
```

Las etiquetas son nombres seguidos por dos puntos (:)

Para el manejo de los errores se utiliza el comando ON nivel de severidad THEN comando

Los errores pueden ser WARNING, ERROR, SEVERE_ERROR tal como se definen en el manual de DCL.

Un error simulado se obtiene con CONTROL_Y de la forma siguiente ON CONTROL_Y THEN GOTO FIN donde FIN es una etiqueta. Cuando se oprimen en forma simultánea CTRL y la Y entonces el procedimiento de comandos se dirige a la etiqueta FIN.

Para permitir el uso de CTRL/Y se debe escribir al inicio $SET CONTROL=Y y para desabilitarlo se escribe $SET NOCONTROL=Y pero debe tenerse mucho cuidado en su uso porque si el porceso se queda en un 'loop' no hay forma de detenerlo

Para escribir información se utiliza el $WRITE. El WRITE debe ir acompañado del nombre del dispositivo o archivo en el cual queremos imprimir la información. Si se desea desplegar algo en la pantalla se debe escribir $ WRITE SYS$OUTPUT expresión

Para leer información vía terminal se debe utilizar el comando $INQUIRE nombre del símbolo [promt]. El promt es el mensaje que la computadora imprimirá al realizar el INQUIRE.

Un ejemplo de esto puede ser:

$INQUIRE DATO Nombre de tu Programa.

Note que el promt va sin comillas.

Para profundizar más en esto, sería conveniente que tomaran un curso de DCL.

9.4) LOGIN

El LOGIN.COM es un archivo de comnados que el sistema ejecuta al entrar en sesión. En el se recomienda definir símbolos con los comandos más usados en nuestro diario trabajo, en la VAX.

Un ejemplo del uso de procedientos de comandos es:

```
$ SET NOVERIFY
$ ! COMPILA  LIGA  CORRE
$ WRITE SYS$OUTPUT "                    COMPILA  LIGA  COOORREEE
!!!"
$ WRITE SYS$OUTPUT ""
$ PARAM1:
$ IF P1 .NES. "" THEN GOTO ELSE1
$     INQUIRE P1 PROGRAMA
$     GOTO PARAM1
$ ELSE1:
$     PROGRAMA:=='P1'
$ ENDIF1:
$ PARAM2:
$ IF P2 .NES. "" THEN GOTO ELSE2
$     INQUIRE P2 LENGUAJE
$     GOTO PARAM2
$ ELSE2:
$     LENGUAJE:=='P2'
$ ENDIF2:
$ CADENA:='LENGUAJE'
$ DEPURA:=""
$ LOOP:
$     LONGITUD='F$LENGTH(CADENA)'
$     DESPLA='F$LOCATE("/",CADENA)'
$     IF DESPLA .EQ. LONGITUD THEN GOTO ENDLOOP
$         DIFER=LONGITUD-DESPLA
$         AUXILIO:='F$EXTRACT(DESPLA,DIFER,CADENA)'
$         CHARACTER:='F$EXTRACT(1,2,AUXILIO)'
$         IF CHARACTER .EQS. "DE" THEN GOTO ELSE4
$             DIFER=DIFER-1
$             CADENA:='F$EXTRACT(1,DIFER,AUXILIO)'
$             GOTO ENDIF4
$         ELSE4:
$             DEPURA:="/DEBUG"
$             GOTO ENDLOOP
$         ENDIF4:
$         GOTO LOOP
$ ENDLOOP:
$ !
$ ! COMPILACION
$ !
$ WRITE SYS$OUTPUT "Compilando"
$ 'LENGUAJE' 'PROGRAMA'
$ !
$ ! LIGADO
$ !
$ LONGITUD='F$LENGTH(PROGRAMA)'
$ DESPLA='F$LOCATE(".",PROGRAMA)'
$ IF DESPLA .EQ. LONGITUD THEN GOTO LIGA
$     PROGRAMA:='F$EXTRACT(0,DESPLA,PROGRAMA)'
```

```
$ LIGA:
$ WRITE SYS$OUTPUT "Ligando"
$ LINK'DEPURA' 'PROGRAMA'
$ !
$ ! EJECUTA
$ !
$ ASSIGN/USERMODE SYS$COMMAND SYS$INPUT
$ WRITE SYS$OUTPUT "Coooorriendo !!!!!!!!!!!!"
$ MAXFOR = 50
$ I = 1
$ FOR:
$ IF I .EQ. MAXFOR THEN GOTO ENDFOR
$       I = I+1
$       GOTO FOR
$ ENDFOR:
$ RUN 'PROGRAMA'
```

El siguiente programa define la posición de los tabuladores, limpia la pantalla y despliega el directorio.

```
$WRITE SYS$OUTPUT "<ESC>[2J"
$WRITE SYS$OUTPUT "<ESC>[;f"
$WRITE SYS$OUTPUT "<ESC>[3g"
$WRITE SYS$OUTPUT "<ESC>[1;7H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[7;10H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[10;13H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[13;17H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[17;21H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[21;25H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[25;29H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[29;33H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[33;37H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[37;41H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[41;45H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[45;49H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[49;53H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[53;57H<ESC>H"
$WRITE SYS$OUTPUT "<ESC>[?4;6;8h"
$LISTADO:==SET TERM/WIDTH=132
$PROGRAMA:==SET TERM/WIDTH=80
$Z:==DIRECTORY
$SUB:==SET DEF [CLAVE.SUBDIR]
$BAS:==SET DEF [CLAVE.BASIC]
$280:==SET TERM/WIDTH=80
$DIR
```

El caracter <ESC> resulta de oprimir 2 veces la tecla ESC, y se debe estar en Editor modo caracter.

AYUDAS

En el sistema existen una serie de ayudas para hacer el trabajo más sencillo. Estas ayudas son:

10.1. MAIL

El MAIL nos permite dejarle mensajes a otros usuarios para que cuando entre en sesión la computadora le avise que tiene un recado y lo lea; si el usuario está en sesión cuando mandes el mensaje, la computadora le indicará en ese momento que tiene un mensaje.

Para utilizar esta ayuda lo primero que debe hacerse es teclear MAIL <return>, y utilizar los comandos propios del MAIL como son: %SEND, %READ, %DELETE, %HELP, %EXIT, etc.

10.2. PHONE

La ayuda del PHONE es para entablar un diálogo con otros usuarios que estén en sesión en ese momento. Para invocarlo es necesario teclear PHONE <return> y utilizar los comandos del PHONE como son %DIAL, %ANSWER, %HANGUP, %DIRECTORY, etc.

10.3. DEBUGGER

El Debugger o depurador permite ver el contenido de ciertas variables durante una corrida y, si es necesario, modificar valores. Para poder utilizarlo es necesario compilar y ligar el programa con los calificadores /DEBUG y el dar el comando RUN el programa no inicia la corrida sino que la computadora le cede el control al Debugger (promt: DBG>). Los comandos básicos son: GO, SET BREAK, EVALUATE, CANCEL BREAK, etc.

Se recomienda el estudio de estas facilidades en el instructivo correspondiente (sobre todo el Debugger) o acudir al HELP de cada uno de ellos.

11
PROTECCIONES

Para proteger los archivos en una forma 'permanente' se puede valer de cintas magnéticas o de discos flexibles (diskettes). Estos tipos de almacenamientos van a permitir almacenar más información de la que se puede tener en nuestra área de disco.

11.1. CINTAS

Las cintas magnéticas son almacenamientos de tipo secuencial, las hay de 600, 800 y 1200 ft de longitud y se pueden grabar con una densidad de 800 ó 1600 bpi (bits por pulgada). El nombre que se le asigna al dispositivo inicia con las siglas MT, en la configuración actual, la cinta es la unidad MTA0:.

11.2. DISKETTES

Los diskettes o Floppy Disk es un almacenamiento de acceso directo, permite tener el directorio y subdirectorios como si se estuviera en el disco. El nombre del dispositivo es DYA0: ó DYA1: dado que hay dos unidades. En el Floppy (que es de 8 pulgadas) se puede grabar a densidad sencilla o a densidad doble lo que proporciona 256 Kb y 512 Kb respectivamente.

No todos los discos de 8 pulgadas sirven, por lo que antes de adquirirlos deben preguntar las marcas y modelos de éstos que le sirvan a la VAX.

11.3. COMANDOS

Para poder utilizar las cintas o los floppys es necesario inicializarlos, esto es darles una serie de atributos, que se le graban para que la computadora sepa, cada vez que los utilicemos, que características tienen. Estos atributos incluyen: un nombre, la densidad y la etiqueta.

Cada vez que se utilicen estos medios de almacenamiento es necesario 'adquirirlo' de la unidad para que ningún otro usuario pueda trabajar con la cinta o diskette.

Después de realizada la operación, es necesario indicarlo a la máquina que 'monte' el medio de almacenamiento, ésto es, que permita el acceso porque aunque físicamente esté

colocado, lógicamente aún no está disponible hasta haberlo 'montado'.

Si el dispositivo es un Floppy es necesario posicionarse en él con el $SET DEF DYA0:, y luego crear los subdirectorios para posteriormente irse a ellos.

Cuando la cinta hasta ahí ubicarnos en ella con el $SET DEF.

En este momento es cuando se puede realizar las operaciones de protección de disco al medio, mediante el $COPY, o hacer archivos del medio al disco, también con $COPY. Es necesario proporcionar los nombres completos (incluyendo dispositivos).

Después de esto hay que 'desmontar' y liberar el dispositivo.

Los Comandos de DCL necesarios son:

$INITIALIZE nombre-del-dispositivo[:] etiqueta-del-volumen

donde el calificador principal es /DENSITY=valor

donde valor debe ser 800 o 1600 para cintas o
SINGLE O DOUBLE para Floppy.

Esto es necesario hacerlo solo una vez, o cada vez que se desee borrar todo el contenido del medio de almacenamiento.

$MOUNT dispositivo [etiqueta] [nombre-lógico]

y sus calificadores principales son:
/FOREIGN
/DENSITY
/BLOCKSIZE
/RECORDSIZE
que se usan principalmente para cintas que vienen o van hacia otros equipos que no sean VAX.

$ALLOCATE dispositivo [nombre-lógico]

para lograr la privacia en el acceso.

$COPY archivo-origen archivo-destino

donde archivo-origen es el nombre completo (incluyendo el

dispositivo) del archivo que se desea copiar; es permitido utilizar los 'wild cards'. Y el archivo-destino es el nombre del archivo que se quiere que reciba el(los) archivo(s) que se copia(n). Cabe hacer notar que se puede omitir el dispositivo en el que nos encontramos.

Para saber que archivos se tienen, se puede pedir el Directorio con el comando $DIRECTORY.

$DISMOUNT dispositivo

Ocasiona que el medio de almacenamiento masivo sea liberado o desmontado.

$DEALLOCATE dispositivo

Sirve para liberar el dispositivo.

11.4. CINTAS DE OTROS EQUIPOS

Cuando se trata de cintas de otros equipos no hay que inicializarlos y debemos montarlas con el calificador /FOREIGN. Si la cinta no fué grabada en código ASCII será necesario correr algún programa que convierta de EBCDIC a ASCII, pero es necesario conocer las características con que fue grabada la cinta.

11.5. Ejemplos:

Para copiar archivos a la cinta

```
$ ALLOCATE MTA0:
$ INIT/DENSITY=1600 MTA0: CINTA   cinta nueva, omítase
                                  si la cinta ya estaba
                                  inicializada
$ MOUNT    MTA0:
$ COPY MIARCHIVO.*  MTA0:[]*.*    copie todos los archivos
                                  que se llamen MIARCHIVO
$ DISMOUNT MTA0:
$ DEALL    MTA0:
```

Para copiar más archivos a una cinta ya inicializada

```
$ ALLOCATE MTA0:
$ MOUNT    MTA0:
$ SET DEF MTA0:
$ DIR                             directorio de la cinta
$ COPY DBA1:[CURSO]ELARCHIVO.*    *.*
```

```
                                        copia todos los archivos
                                        que se llaman ELARCHIVO

$ DISMOUNT MTA0:
$ DEALL    MTA0:

Para bajar de cinta a disco

$ ALLOCATE MTA0:
$ MOUNT    MTA0:
$ SET DEF MTA0:
$ DIR                                   directorio de la cinta
$ SET DEF DBA1:[CURSO]
$ COPY MTA0:MIARCHIVO.*      *.*        copia a disco todos los
                                        archivos de la cinta
                                        que se llamen MIARCHIVO

$ DISMOUNT MTA0:
$ DEALL    MTA0:
```

LENGUAJE DE PROGRAMACION
COBOL

P R O G R A M A S
E J E M P L O S

AGOSTO, 1985

```
IDENTIFICATION DIVISION.
PROGRAM-ID.          DEMOSTRACION-7 IS INITIAL PROGRAM.
PROCEDURE DIVISION.
PRINCIPIO.
          DISPLAY " HOLA SOY EL PRIMER PROGRAMA COBOL "
          STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.          DEMOSTRACION-5.

ENVIRONMENT     DIVISION.

DATA            DIVISION.

PROCEDURE DIVISION.

PRINCIPIO.

        DISPLAY " HOLA SOY EL PRIMER PROGRAMA COBOL " WITH NO ADVANCING.
        STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-4.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

DATA DIVISION.


PROCEDURE DIVISION.

PRINCIPIO.

        DISPLAY " HOLA SOY EL PRIMER PROGRAMA COBOL " WITH NO ADVANCING.
        STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                    DEMOSTRACION-6.
AUTHOR.                        JORGE VALERIO.
                               ELABORADO PARA EL CURSO DE COBOL BASICO
                               IMPARTIDO EN EL CECAFI.
INSTALLATION.                  CECAFI.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
SOURCE-COMPUTER.               VAX11-780.
OBJECT-COMPUTER.               VAX11-780.

DATA           DIVISION.


PROCEDURE DIVISION.

PRINCIPIO.

        DISPLAY " HOLA SOY EL PRIMER PROGRAMA COBOL " WITH NO ADVANCING.
        STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-9.
AUTHOR.                     JORGE-VALERIO.
                            CU UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT     DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA            DIVISION.


PROCEDURE DIVISION.

PRINCIPIO.

        DISPLAY " HOLA SOY EL PRIMER PROGRAMA COBOL " WITH NO ADVANCING.
        STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-3.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

DATA           DIVISION.

WORKING-STORAGE SECTION.

77      NUMERO              PIC 9(06).
77      NOMBRE              PIC X(40).

PROCEDURE DIVISION.

PRINCIPIO.

    DISPLAY " HOLA DAME TU NOMBRE " WITH NO ADVANCING.
    ACCEPT NOMBRE.
    DISPLAY NOMBRE " DAME UN NUMERO " WITH NO ADVANCING.
    ACCEPT NUMERO.
    DISPLAY " TU NUMERO ES " NUMERO.
    STOP RUN.
```

```
PROGRAM-ID.                PIDE NOMBRE.
AUTHOR.                    JAVIER VILLEGAS
                           CENTRO NACIONAL DE INFORMACION.
INSTALLATION.              SPP & SPP/INIARA.
DATE-COMPILED.
DATE-WRITTEN.              SEPTIEMBRE 1983.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SOURCE-COMPUTER.           NCR 89/67
OBJECT-COMPUTER.           VAX 11/780

DATA DIVISION.

Working-Storage SECTION.

77    NUMERO               PIC 9(06).
77    NOMBRE               PIC X(40)9(02).

PROCEDURE DIVISION.

PRINCIPIO.

    DISPLAY " 'HOLA' DAME TU NOMBRE " WITH NO-ADVANCING.
    FROM CONSOLE ACCEPT NOMBRE.
    DISPLAY NOMBRE " DAME UN NUMERO" " WITH NO-ADVANCING.
    FROM CONSOLE ACCEPT NUMERO.
    DISPLAY " TU NUMERO ES " NUMERO.
    STOP RUN.
```

```
PROGRAM-ID.                    DEMOSTRACION-10.
AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM.
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01   REGISTRO-1.
     03 .
        05 SUB-1A PIC 9(18).
        05 SUB-1B PIC 9(18).
        05       PIC 9(18).
     03 SUB-2.
        05 SUB-2A PIC X(65).
        05 SUB-2B PIC A(65).
        05 SUB-2C PIC X(30).

PROCEDURE DIVISION.

PRINCIPIO.

        ACCEPT REGISTRO-1.
        DISPLAY " REGISTRO 1 " REGISTRO-1.
        STOP RUN.
```

```
 IDENTIFICATION DIVISION.

 PROGRAM-ID.                    DEMOSTRACION-9.
 AUTHOR.                        JORGE-VALERIO.
                                CD UNIVERSITARIA UNAM
                                MEXICO.
 INSTALLATION.                  CECAFI.
                                CENTRO DE CALCULO DE LA FACULTAD DE ING.
 DATE-WRITTEN.                  29-SEP-83.
 DATE-COMPILED.                 29-SEP-83.
 SECURITY.                      PRIVADA.
                                ESTE PROGRAMA ES PARA DEMOSTRACION.
                                PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                                MEDIO.

 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER.               VAX11-780.
 OBJECT-COMPUTER.               VAX11-780.
 SPECIAL-NAMES.                 CONSOLE IS CONSOLE.

 DATA          DIVISION.
 WORKING-STORAGE SECTION.
 01    REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC 9(02).
          05 SUB-1B PIC 9(02).
          05 SUB-1C PIC 9(02).
       03 SUB-2.
          05 SUB-2A PIC X(02).
          05 SUB-2B PIC X(02).
          05 SUB-2C PIC X(02).

 PROCEDURE DIVISION.

 PRINCIPIO.

          ACCEPT REGISTRO-1 FROM CONSOLE.
          DISPLAY " REGISTRO 1 " REGISTRO-1.
          STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EJ-MAS-PEQUENO.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  SUMANDO-1                          PIC 9(02).
77  SUMANDO-2                          PIC 9(02).
77  SUMA                               PIC 9(03).
PROCEDURE DIVISION.
1-INICIO.
    ACCEPT SUMANDO-1
    ACCEPT SUMANDO-2
    COMPUTE SUMA = SUMANDO-1 + SUMANDO-2
    DISPLAY SUMANDO-1 " + " SUMANDO-2 " = " SUMA.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                    DEMOSTRACION-9R.
AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1.
      03 GRUPO1.
         05 SUB-1A     PIC S9(02)V9(03) SIGN IS LEADING SEPARATE CHARACTER.

PROCEDURE DIVISION.

PRINCIPIO.

          ACCEPT REGISTRO-1.
          DISPLAY " REGISTRO 1 " REGISTRO-1.
          ADD 1 TO SUB-1A IN REGISTRO-1.
          DISPLAY " ADD 1 TO REGISTRO 1 " SUB-1A.
          STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SUMA.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 A PIC 9(02).
77 B PIC 9(02).
77 C PIC 9(02).
PROCEDURE DIVISION.
PRINCIPAL.
    DISPLAY "A (dos digitos)? " WITH NO ADVANCING
    ACCEPT A
    DISPLAY "B (dos digitos)? " WITH NO ADVANCING
    ACCEPT B
    ADD A B GIVING C.
    DISPLAY A "+" B "=" C
    STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MOVE-CORR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 REGISTRO-1.
   03 CAMPO-1                          PIC 9(02).
   03 CAMPO-2                          PIC X(04).
   03 CAMPO-3                          PIC 9(08).
   03 CAMPO-4                          PIC 9(03).
01 REGISTRO-2.
   03 CAMPO-1                          PIC 9(02) VALUE ZEROES.
   03 OTRO-CAMPO-1                     PIC X(15) VALUE ALL "X".
   03 OTRO-CAMPO-2                     PIC X(30) VALUE ALL "+".
   03 CAMPO-4                          PIC 9(03) VALUE 1.
   03 OTRO-CAMPO-3                     PIC 9(05) VALUE 91.
77 TEMPORAL-1                          PIC 9(02).
77 TEMPORAL-2                          PIC 9(03).
77 TEM1          PIC X(02).
77 TEM2          PIC 9(02).
PROCEDURE DIVISION.
INICIO.
    DISPLAY "DAME REGISTRO UNO"
    ACCEPT REGISTRO-1
    MOVE CAMPO-1 IN REGISTRO-1 TO TEMPORAL-1
    MOVE CAMPO-4 IN REGISTRO-1 TO TEMPORAL-2
    MOVE CORR REGISTRO-1 TO REGISTRO-2
    DISPLAY "REGISTRO DOS"
    DISPLAY REGISTRO-2
    ACCEPT TEM1
    DISPLAY "TEM1 = " TEM1 " TEM2 " TEM2
    MOVE TEM1 TO TEM2
    DISPLAY "DESPUES DEL MOVE"
    DISPLAY "TEM1 = " TEM1 " TEM2 " TEM2
    STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                    DEMOSTRACION-10.
AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC 9(02).
          05 SUB-1B PIC 9(02).
          05 SUB-1C PIC 9(02).
       03 SUB-2.
          05 SUB-2A PIC X(02).
          05 SUB-2B PIC X(02).
          05 SUB-2C PIC X(02).
01     REGISTRO-2.
       03 SUB-1.
          05 SUB-1A PIC 9(02).
          05 SUB-1B PIC 9(02).
          05 SUB-1C PIC 9(02).
       03 SUB-2.
          05 SUB-2A PIC X(02).
          05 SUB-2B PIC X(02).
          05 SUB-2C PIC X(02).

PROCEDURE DIVISION.

PRINCIPIO.

          ACCEPT REGISTRO-1.
          DISPLAY " REGISTRO 1 " REGISTRO-1.
          MOVE CORRESPONDING REGISTRO-1 TO REGISTRO-2.
          DISPLAY " REGISTRO 2 " REGISTRO-2.
          STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-11.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1.
      03 SUB-1.
         05 SUB-1A PIC 9(02).
         05 SUB-1B PIC 9(02).
         05 SUB-1C PIC 9(02).
      03 SUB-2.
         05 SUB-2A PIC X(02).
         05 SUB-2B PIC X(02).
         05 SUB-2C PIC X(02).
01    REGISTRO-2.
      03 SUB-1.
         05 SUB-1A PIC 9(02).
         05 SUB-1B PIC 9(02).
         05 SUB-1C PIC 9(02).
      03 SUB-2.
         05 SUB-2A PIC X(02).
         05 SUB-2B PIC X(02).
         05 SUB-2C PIC X(02).

PROCEDURE DIVISION.

PRINCIPIO.

         ACCEPT REGISTRO-1.
         DISPLAY ' REGISTRO 1 ' REGISTRO-1.
         MOVE SUB-1A IN REGISTRO-1 TO  SUB-1A IN REGISTRO-2.
         MOVE SUB-1B IN REGISTRO-1 TO  SUB-1B IN REGISTRO-2.
         MOVE SUB-1C IN REGISTRO-1 TO  SUB-1C IN REGISTRO-2.
         MOVE SUB-2A IN REGISTRO-1 TO  SUB-2A IN REGISTRO-2.
         MOVE SUB-2B IN REGISTRO-1 TO  SUB-2B IN REGISTRO-2.
         MOVE SUB-2C IN REGISTRO-1 TO  SUB-2C IN REGISTRO-2.
         DISPLAY ' REGISTRO 2 ' REGISTRO-2.
         STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-D12.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1.
      03 SUB-1.
         05 SUB-1A PIC 9(02).
         05 SUB-1B PIC 9(02).
         05 SUB-1C PIC 9(02).
      03 SUB-2.
         05 SUB-2A PIC X(02).
         05 SUB-2B PIC X(02).
         05 SUB-2C PIC X(02).
      03 SUB-3.
         05 SUB-3A.
            08 SUB-3A-SUB-3AA PIC A(02).
            08 SUB-3A-SUB-3AB PIC A(02).
         05 SUB-3B.
            08 SUB-3B-SUB-BA  PIC A(02).
            08 SUB-3B-SUB-BB  PIC A(02).

PROCEDURE DIVISION.

PRINCIPIO.

          MOVE 11 TO SUB-1 IN REGISTRO-1.
          MOVE 'AA' TO SUB-2C IN REGISTRO-1.
          MOVE 'XX' TO SUB-3B-SUB-BB IN REGISTRO-1.
          DISPLAY ' REGISTRO 1 ' REGISTRO-1.
          STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-13.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT     DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA            DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC 9(10).
          05 SUB-1B PIC 9(10).
          05 SUB-1C PIC 9(02)V9(02).

PROCEDURE DIVISION.

PRINCIPIO.

           MOVE 101 TO SUB-1A IN REGISTRO-1.
           MOVE 1234 TO SUB-1C IN REGISTRO-1.
*          REGISTRO 1 0000000101/0000000000/3400
           DISPLAY " REGISTRO-1 " REGISTRO-1.
           ACCEPT SUB-1C IN REGISTRO-1.
           DISPLAY "SUB-1C " SUB-1C IN REGISTRO-1.
*          SUB-1C IS /1234 -- 1234/ 12 -- 12 / 34 -- 34/
           MOVE SUB-1C TO SUB-1A.
           DISPLAY "SUB-1A " SUB-1A IN REGISTRO-1.
*          SUB-1C IS XXX.DD --- XXX
           STOP RUN.
```

```
PROGRAM-ID.                DEMOSTRACION-13.
AUTHOR.                    JORGE-VALERIO.
                           CD UNIVERSITARIA UNAM
                           MEXICO.
INSTALLATION.              CECAFI.
                           CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.              29-SEP-83.
DATE-COMPILED.             29-SEP-83.
SECURITY.                  PRIVADA.
                           ESTE PROGRAMA ES PARA DEMOSTRACION.
                           PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                           MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.           VAX11-780.
SOURCE-COMPUTER.           VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC VP(03)9(10).
          05 SUB-2A PIC 9(10).
01     REGISTRO-2.
       03 SUB-2B PIC X(04).
       03 SUB-2C PIC X(10).
77     CONT PIC 9(02) VALUE ZEROES.
PROCEDURE DIVISION.

PRINCIPIO.

       MOVE "100.10" TO SUB-2C.
       INSPECT SUB-2C TALLYING CONT FOR CHARACTERS BEFORE INITIAL ".".
       DISPLAY CONT SUB-2C(1:CONT).
       MOVE SUB-2C(1:CONT) TO SUB-2A.
       ADD 1 TO SUB-2A.
       DISPLAY SUB-2A.
       STOP RUN.
```

```
PROGRAM-ID.                    DEMOSTRACION-14.
AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE IIMS.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1.
      03 SUB-1.
         05 SUB-1A PIC 9(10).
         05 SUB-1B PIC 9(10).
         05 SUB-1C PIC 9(02)V9(02).
01    REGISTRO-2.
      03 SUB-1.
         05 SUB-1A PIC 9(10).
         05 SUB-1B PIC 9(10).
         05 SUB-1C PIC 9(02)V9(02).

PROCEDURE DIVISION.

PRINCIPIO.

         MOVE 101 TO SUB-1A IN REGISTRO-1.
         MOVE 1234 TO SUB-1C IN REGISTRO-1.
*        REGISTRO 1 0000000101/0000000000/3400
         DISPLAY " REGISTRO 1 " REGISTRO-1.
         ACCEPT SUB-1C IN REGISTRO-1.
         DISPLAY "SUB-1C REG-1 " SUB-1C IN REGISTRO-1.
*        SUB-1C REG-1 IS /1234 -- 1234/ 12 -- 12 / 34 --  34/
         MOVE SUB-1C IN REGISTRO-1  TO SUB-1A IN REGISTRO-2.
         DISPLAY "SUB-1A REG-2 " SUB-1A IN REGISTRO-2.
*        SUB-1A REG-2 IS XXX.DD --- XXX
         STOP RUN.
```

```
PROGRAM-ID.                    DEMOSTRACION-19.
AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC X(10).
          05 SUB-1B PIC X(05).
          05 SUB-1C PIC X(02).
01     REGISTRO-2.
       03 SUB-1.
          05 SUB-1A PIC X(10).
          05 SUB-1B PIC X(05).
          05 SUB-1C PIC X(02).
          05 FILLER PIC X(10).

PROCEDURE DIVISION.

PRINCIPIO.

          ACCEPT REGISTRO-1.
          DISPLAY "REGISTRO-1 " REGISTRO-1.
          DISPLAY "REG-1 SUB-1A " SUB-1A IN REGISTRO-1.
          DISPLAY "REG-1 SUB-1B " SUB-1B IN REGISTRO-1.
          DISPLAY "REG-1 SUB-1C " SUB-1C IN REGISTRO-1.
          MOVE SUB-1A IN REGISTRO-1 TO SUB-1B IN REGISTRO-2
                                       SUB-1C IN REGISTRO-2.
          DISPLAY "REGISTRO-2 " REGISTRO-2.
          DISPLAY "REG-2 SUB-1A " SUB-1A IN REGISTRO-2.
          DISPLAY "REG-2 SUB-1B " SUB-1B IN REGISTRO-2.
          DISPLAY "REG-2 SUB-1C " SUB-1C IN REGISTRO-2.
          STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-19.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1.
      03 SUB-1.
         05 SUB-1A PIC X(10).
         05 SUB-1B PIC X(05).
         05 SUB-1C PIC X(02).
01    REGISTRO-2.
      03 SUB-1.
         05 SUB-1A PIC X(10).
         05 SUB-1B PIC X(05).
         05 SUB-1C PIC X(02).
         05 FILLER PIC X(10).
77    CONT PIC 9(03) VALUE ZEROES.

PROCEDURE DIVISION.

PRINCIPIO.

          ACCEPT REGISTRO-1.
          DISPLAY "REGISTRO-1=" REGISTRO-1.
          DISPLAY "REG-1 SUB-1A " SUB-1A IN REGISTRO-1.
          DISPLAY "REG-1 SUB-1B " SUB-1B IN REGISTRO-1.
          DISPLAY "REG-1 SUB-1C " SUB-1C IN REGISTRO-1.
          INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J".
          DISPLAY "CONT All Jots " CONT.
          MOVE ZEROES TO CONT.
          INSPECT REGISTRO-1 TALLYING CONT FOR LEADING "J".
          DISPLAY "CONT Leading Jots " CONT.
          STOP RUN.
```

```
PROGRAM-ID.                  DEMOSTRACION-36.

***************************************************************************
*                                                                         *
*                      CARACTERES DE EDICION                              *
*                                                                         *
***************************************************************************

AUTHOR.                      JORGE-VALERIO.
                             CD UNIVERSITARIA UNAM
                             MEXICO.
INSTALLATION.                CECAFI.
                             CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                29-SEP-83.
DATE-COMPILED.               29-SEP-83.
SECURITY.                    PRIVADA.
                             ESTE PROGRAMA ES PARA DEMOSTRACION.
                             PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                             MEDIO.

***************************************************************************
*                                                                         *
ENVIRONMENT     DIVISION.
*                                                                         *
***************************************************************************

CONFIGURATION  SECTION.

OBJECT-COMPUTER.             VAX11-780.
SOURCE-COMPUTER.             VAX11-780.

***************************************************************************
*                                                                         *
DATA            DIVISION.
*                                                                         *
***************************************************************************

***************************************************************************
*                                                                         *
WORKING-STORAGE SECTION.
*                                                                         *
***************************************************************************

77 NUMERO-WS          PIC S9(10)V9(02)    SIGN IS LEADING SEPARATE CHARACTER.
77 NUMERO-SAL01-WS          PIC Z(10).Z(02).
77 NUMERO-SAL02-WS          PIC Z(10)VZ(02).
77 NUMERO-SAL03-WS          PIC $(10).$(02).
77 NUMERO-SAL04-WS          PIC $(01)*(09)V*(02).
77 NUMERO-SAL05-WS          PIC $(01)*(09).*(02).
77 NUMERO-SAL06-WS          PIC $(01)*(07).*(02).
77 NUMERO-SAL07-WS          PIC S9(10)V9(02).
77 NUMERO-SAL08-WS          PIC +9(10)V9(02).
77 NUMERO-SAL09-WS          PIC -9(10)V9(02).
77 NUMERO-SAL10-WS          PIC 9(10)V9(02)CR.
77 NUMERO-SAL11-WS          PIC 9(10)V9(02)DB.
77 NUMERO-SAL12-WS          PIC 9(10)P(08).
77 NUMERO-SAL13-WS          PIC 9(02)B(03)9(08)0(03)9(02).
77 NUMERO-SAL14-WS          PIC 9(01)/(03)9(03),9(03),9(03),0(03)9(02).
***************************************************************************
*                                                                         *
PROCEDURE DIVISION.
*                                                                         *
```

```
PRINCIPIO.
    DISPLAY " DAME UN NUMERO 9(10)V9(02) : " WITH NO ADVANCING .
    ACCEPT NUMERO-WS.
    MOVE NUMERO-WS TO NUMERO-SAL01-WS.
    DISPLAY " FMT: 01 " NUMERO-SAL01-WS.
    MOVE NUMERO-WS TO NUMERO-SAL02-WS.
    DISPLAY " FMT: 02 " NUMERO-SAL01-WS.
    MOVE NUMERO-WS TO NUMERO-SAL03-WS.
    DISPLAY " FMT: 03 " NUMERO-SAL03-WS.
    MOVE NUMERO-WS TO NUMERO-SAL04-WS.
    DISPLAY " FMT: 04 " NUMERO-SAL04-WS.
    MOVE NUMERO-WS TO NUMERO-SAL05-WS.
    DISPLAY " FMT: 05 " NUMERO-SAL05-WS.
    MOVE NUMERO-WS TO NUMERO-SAL06-WS.
    DISPLAY " FMT: 06 " NUMERO-SAL06-WS.
    MOVE NUMERO-WS TO NUMERO-SAL07-WS.
    DISPLAY " FMT: 07 " NUMERO-SAL07-WS.
    MOVE NUMERO-WS TO NUMERO-SAL08-WS.
    DISPLAY " FMT: 08 " NUMERO-SAL08-WS.
    MOVE NUMERO-WS TO NUMERO-SAL09-WS.
    DISPLAY " FMT: 09 " NUMERO-SAL09-WS.
    MOVE NUMERO-WS TO NUMERO-SAL10-WS.
    DISPLAY " FMT: 10 " NUMERO-SAL10-WS.
    MOVE NUMERO-WS TO NUMERO-SAL11-WS.
    DISPLAY " FMT: 11 " NUMERO-SAL11-WS.
    MOVE NUMERO-WS TO NUMERO-SAL12-WS.
    DISPLAY " FMT: 12 " NUMERO-SAL12-WS.
    MOVE NUMERO-WS TO NUMERO-SAL13-WS.
    DISPLAY " FMT: 13 " NUMERO-SAL13-WS.
    MOVE NUMERO-WS TO NUMERO-SAL14-WS.
    DISPLAY " FMT: 14 " NUMERO-SAL14-WS.
    STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-31.
*****************************************************************
*
*          IF THEN ELSE ENDIF
*                              ARITMETICO
*
*****************************************************************

AUTHOR.                     JORGE-VALERIO.
                            CO UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

*****************************************************************
*
   ENVIRONMENT    DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

*****************************************************************
*
   DATA          DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.

77 NUMERO                   PIC 9(10).

*****************************************************************
*
   PROCEDURE DIVISION.
*
*****************************************************************

PRINCIPIO.

*****************************************************************
*                       IF ARITMETICO
*****************************************************************

      MOVE 10 TO NUMERO.

      IF NUMERO IS NUMERIC THEN
         DISPLAY "<<NUMERO>> ES NUMERICO "
      ELSE
         DISPLAY "<<NUMERO>> ES NO NUMERICO "
      END-IF.

      IF NUMERO IS POSITIVE THEN
         DISPLAY "<<NUMERO>> ES POSITIVO"
```

```
           DISPLAY "<<NUMERO>> ES NO POSITIVO"
       END-IF.

       IF NUMERO IS ZERO THEN
          DISPLAY "<<NUMERO>> ES ZERO"
       ELSE
          DISPLAY "<<NUMERO>> ES NO ZERO"
       END-IF.

      *
      *    ESTO NO SE VALE
      *
      *    IF NUMERO IS NOT ALPHABETIC THEN
      *       DISPLAY "<<NUMERO>> ES NO NUMERICO "
      *    ELSE
      *       DISPLAY "<<NUMERO>> ES NUMERICO "
      *    END-IF.

       IF NUMERO IS EQUAL 10 THEN
      *    IS [ NOT ] =
           DISPLAY "EL NUMERO ES IGUAL A 10"
       ELSE
           DISPLAY "EL NUMERO NO ES IGUAL A 10"
       END-IF.

       IF NUMERO IS GREATER THAN 5 THEN
      *    IS [ NOT ] >
           DISPLAY "EL NUMERO ES MAYOR A 5"
       ELSE
          DISPLAY "EL NUMERO NO ES MAYOR A 5"
       END-IF.

       IF NUMERO IS LESS THAN 5 THEN
      *    IS [ NOT ] <
           DISPLAY "EL NUMERO ES MENOR A 5"
       ELSE
           DISPLAY "EL NUMERO NO ES MENOR A 5"
       END-IF.

       IF NUMERO IS LESS THAN 5 AND NUMERO IS GREATER 1 THEN
      *    IS [ NOT ] <
           DISPLAY "EL NUMERO ES MENOR A 5 Y MAYOR A 1"
       ELSE
          DISPLAY "EL NUMERO NO ES MENOR A 5 Y NO MAYOR A 1"
       END-IF.

       IF NUMERO IS LESS THAN 5 OR NUMERO IS GREATER 1 THEN
      *    IS [ NOT ] <
          DISPLAY "EL NUMERO ES MENOR A 5 O MAYOR A 1"
       ELSE
           DISPLAY "EL NUMERO NO ES MENOR A 5 O NO MAYOR A 1"
       END-IF.

      ******************************************************************
      *                    FIN DE PROGRAMA
      ******************************************************************

       STOP RUN.
```

```
PROGRAM-ID.             DEMOSTRACION-32.
*****************************************************************
*
*           IF THEN ELSE ENDIF
*                       ALFABETICO
*
*****************************************************************

AUTHOR.                 JORGE-VALERIO.
                        CD UNIVERSITARIA UNAM
                        MEXICO.
INSTALLATION.           CECAFI.
                        CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.           29-SEP-83.
DATE-COMPILED.          29-SEP-83.
SECURITY.               PRIVADA.
                        ESTE PROGRAMA ES PARA DEMOSTRACION.
                        PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                        MEDIO.

*****************************************************************
*
   ENVIRONMENT    DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.        VAX11-780.
SOURCE-COMPUTER.        VAX11-780.

*****************************************************************
*
   DATA           DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.

77 NOMBRE                PIC X(10).

*****************************************************************
*
   PROCEDURE DIVISION.
*
*****************************************************************

PRINCIPIO.

*****************************************************************
*                    IF ALFABETICO
*****************************************************************

      MOVE "AA" TO NOMBRE.

      IF NOMBRE IS NUMERIC THEN
         DISPLAY "<<NOMBRE>> ES NUMERICO "
      ELSE
         DISPLAY "<<NOMBRE>> ES NO NUMERICO "
      END-IF.

*
*     ESTO NO SE VALE
```

```
*       IF NOMBRE IS POSITIVE THEN
*          DISPLAY "<<NOMBRE>> ES POSITIVO"
*       ELSE
*          DISPLAY "<<NOMBRE>> ES NO POSITIVO"
*       END-IF.

*       IF NOMBRE IS ZERO THEN
*          DISPLAY "<<NOMBRE>> ES ZERO".
*       ELSE
*          DISPLAY "<<NOMBRE>> ES NO ZERO"
*       END-IF.

        IF NOMBRE IS ALPHABETIC THEN
           DISPLAY "<<NOMBRE>> ES ALFANUMERICO"
        ELSE
           DISPLAY "<<NOMBRE>> ES NO ALFANUMERICO"
        END-IF.

        MOVE "01" TO NOMBRE.

        IF NOMBRE IS ALPHABETIC THEN
           DISPLAY "<<NOMBRE>> ES ALFANUMERICO"
        ELSE
           DISPLAY "<<NOMBRE>> ES NO ALFANUMERICO"
        END-IF.

*       <<<<<<<<<<<<<<<<<<<< NO ALFANUMERICO >>>>>>>>>>>>>>>>>>>>

        IF NOMBRE IS EQUAL 10 THEN
*       IS [ NOT ] =
           DISPLAY "EL NOMBRE ES IGUAL A 10"
        ELSE
           DISPLAY "EL NOMBRE NO ES IGUAL A 10"
        END-IF.

        IF NOMBRE IS GREATER THAN 5 THEN
*       IS [ NOT ] >
           DISPLAY "EL NOMBRE ES MAYOR A 5"
        ELSE
           DISPLAY "EL NOMBRE NO ES MAYOR A 5"
        END-IF.

        IF NOMBRE IS LESS THAN 5 THEN
*       IS [ NOT ] <
           DISPLAY "EL NOMBRE ES MENOR A 5"
        ELSE
           DISPLAY "EL NOMBRE NO ES MENOR A 5"
        END-IF.

        IF NOMBRE IS LESS THAN 5 AND NOMBRE IS GREATER 1 THEN
*       IS [ NOT ] <
           DISPLAY "EL NOMBRE ES MENOR A 5 Y MAYOR A 1"
        ELSE
           DISPLAY "EL NOMBRE NO ES MENOR A 5 Y NO MAYOR A 1"
        END-IF.

        IF NOMBRE IS LESS THAN 5 OR NOMBRE IS GREATER 1 THEN
*       IS [ NOT ] <
           DISPLAY "EL NOMBRE ES MENOR A 5 O MAYOR A 1"
        ELSE
           DISPLAY "EL NOMBRE NO ES MENOR A 5 O NO MAYOR A 1"
        END-IF.
```

```
*      <<<<<<<<<<<<<<<<<<<< ALFANUMERICO >>>>>>>>>>>>>>>>>>>>

       MOVE "AA" TO NOMBRE.

       IF NOMBRE IS EQUAL "AA" THEN
       IS [ NOT ] =
          DISPLAY "EL NOMBRE ES IGUAL A AA"
       ELSE
          DISPLAY "EL NOMBRE NO ES IGUAL A AA"
       END-IF.

       IF NOMBRE IS GREATER THAN "A" THEN
*      IS [ NOT ] >
          DISPLAY "EL NOMBRE ES MAYOR A A"
       ELSE
          DISPLAY "EL NOMBRE NO ES MAYOR A A"
       END-IF.

       IF NOMBRE IS LESS THAN "AAA" THEN
*      IS [ NOT ] <
          DISPLAY "EL NOMBRE ES MENOR A 'AAA'"
       ELSE
          DISPLAY "EL NOMBRE NO ES MENOR A 'AAA'"
       END-IF.

       IF NOMBRE IS LESS THAN "AA" AND NOMBRE IS GREATER "A" THEN
*      IS [ NOT ] <
          DISPLAY "EL NOMBRE ES MENOR A 'AA' Y MAYOR A 'A'"
       ELSE
          DISPLAY "EL NOMBRE NO ES MENOR A 'AA' Y NO MAYOR A 'A'"
       END-IF.

       IF NOMBRE IS LESS THAN "AA" OR NOMBRE IS GREATER "A" THEN
*      IS [ NOT ] <
          DISPLAY "EL NOMBRE ES MENOR A 'AA' O MAYOR A 'A'"
       ELSE
          DISPLAY "EL NOMBRE NO ES MENOR A 'AA' O NO MAYOR A 'A'"
       END-IF.

*****************************************************************
*                      FIN DE PROGRAMA
*****************************************************************

       STOP RUN.
```

```
Identification division.
Program-id.     Solucion-de-la-ec-de-2do-grado
Author.         Adrian Alvarez Martinez.
Installation.   Cecafi.
Date-written.   Mayo 1985.
Date-compiled.  Agosto 1985.
Security.       Ninguna.

Environment division.
Configuration section.
Source-computer. Digital-Vax-11-780.
Object-computer. Digital-Vax-11-780.
Input-output section.
File-control.
     Select datos  assign to sys$input.
     Select salida assign to sys$output.

Data division.
File section.
Fd datos  data record is datos1.
01 datos1.
     03 A picture is s9(02)v9(02) sign is leading separate character.
     03 B picture is s9(02)v9(02) sign is leading separate character.
     03 C picture is s9(02)v9(02) sign is leading separate character.
Fd salida data record is linea.
01 Linea  picture is x(80).
Working-storage section.
77 Resultado picture is s9(02)v9(02) value zeroes.
77 Disc      picture is s9(02)v9(02).
77 X1        picture is s9(02)v9(02).
77 X2        picture is s9(02)v9(02).
01 Linea-aux.
     03    Filler picture is x(16) value 'el resultado es:'.
     03    Filler picture is x(02) value all ' '.
     03    Result picture is +9(02).9(02).
     03    Filler picture is x(55) value all ' '.

Procedure division.
Inicio.
     Open input datos.
     Open output salida.
     Display '.
     Display '      Solucion de la ecuacion cuadratica'.
     Display '      ******************************
     Display '.
     Display ' 3 datos en linea. 1: A, 2: B, 3: C,
     Display '                     s : signo   eedd: valor, sin punto decimal.'
     Display '                                 ee enteros
     Display '.                                dd decimales
     Display '.
     Display 'SeeddSeeddSeedd
     Read datos at end display 'No hubo datos.' stop run.
     If a is equal to zero and b is equal to zero and c is equal to 0 stop run.
     If a is equal to zero and b is equal to zero stop run.
     If a is equal to zero and c is equal to zero stop run.
     If b is equal to zero and c is equal to zero stop run.
     If a is equal zero then
        display 'Ecuacion de primer grado.'
        compute x1 = ((-1 * c)/b)
        move x1 to result
        write linea from linea-aux
     else
        compute disc = (b ** 2 - 4 * a * c)
```

```
      if disc is less than zero
         display "Raices complejas no puede resolverlo."
      else
         display "Raices reales."
         compute x1 = ((-b - disc ** 0.5)/(2*a))
         compute x2 = ((-b + disc ** 0.5)/(2*a))
         move x1 to result
         write linea from linea-aux
         move x2 to result
         write linea from linea-aux.
   Stop run.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MC-DONALDS-BURGUERS.
AUTHOR. YO.
INSTALLATION. AQUI.
DATE-WRITTEN. AYER.
DATE-COMPILED. HOY.
SECURITY. SEGURISIMO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ALIMENTOS.
    03 HAMBURGUESAS.
        05 CODIGO-HAMBURGUESAS PIC 9(03) VALUE 010.
        05 CON-QUESO.
            07 CODIGO-CON-QUESO PIC 9(03) VALUE 011.
            07 PRECIO-CON-QUESO PIC 9(03) VALUE 110.
            07 DESCRIPCION-CON-QUESO PIC A(31) VALUE "HAMBURGUESA CON QUESO"
        05 SIN-QUESO.
            07 CODIGO-SIN-QUESO PIC 9(03) VALUE 012.
            07 PRECIO-SIN-QUESO PIC 9(03) VALUE 100.
            07 DESCRIPCION-SIN-QUESO PIC A(31) VALUE "HAMBURGUESA SIN QUESO"
    03 PAPAS.
        05 CODIGO-PAPAS PIC 9(03) VALUE 020.
        05 PRECIO-PAPAS PIC 9(03) VALUE 035.
        05 DESCRIPCION-PAPAS PIC A(14) VALUE "ORDEN DE PAPAS".
    03 REFRESCOS.
        05 CODIGO-REFRESCOS PIC 9(03) VALUE 030.
        05 CHICO.
            07 CODIGO-CHICO PIC 9(03) VALUE 031.
            07 PRECIO-CHICO PIC 9(03) VALUE 050.
            07 DESCRIPCION-REFRESCO-CHICO PIC A(14) VALUE "REFRESCO CHICO".
        05 MEDIANO.
            07 CODIGO-MEDIANO PIC 9(03) VALUE 032.
            07 PRECIO-MEDIANO PIC 9(03) VALUE 060.
            07 DESCRIPCION-REFRESCO-MEDIANO PIC A(16) VALUE "REFRESCO MEDIAN
        05 GRANDE.
            07 CODIGO-GRANDE PIC 9(03) VALUE 033.
            07 PRECIO-GRANDE PIC 9(03) VALUE 070.
            07 DESCRIPCION-REFRESCO-GRANDE PIC A(15) VALUE "REFRESCO GRANDE"
01 SALIDA.
    03 TIPO-DE-ALIMENTO PIC A(21).
01 PIDE-LA-ORDEN.
    03 FILLER PIC X(19) VALUE "CUAL ES SU ORDEN? ".
01 PIDE-PEDIDO.
    03 FILLER PIC X(16) VALUE "ALGO MAS (S/N)? ".
77 SUMA PIC 9(04) VALUE ZERO.
77 PEDIDO-ALIMENTOS PIC X(01) VALUE "S".
77 CLAVE-ALIMENTO PIC 9(03).
77 CLAVE-QUESO PIC 9(03).
77 CLAVE-REFRESCO PIC 9(03).
PROCEDURE DIVISION.
PROGRAMA-PRINCIPAL.
    PERFORM PROCESO WITH TEST BEFORE UNTIL PEDIDO-ALIMENTOS EQUAL "N".
    DISPLAY "TOTAL: " SUMA.
    STOP RUN.
PROCESO.
    DISPLAY " ".
    DISPLAY "    ******************************".
    DISPLAY "    * HAMBURGESAS MC. DONALD'S *".
    DISPLAY "    ******************************".
```

```
    DISPLAY " "
    DISPLAY " CODIGO      PRODUCTO "
    DISPLAY " ------      --------- "
    DISPLAY "   010       HAMBURGESAS"
    DISPLAY "   020       PAPAS"
    DISPLAY "   030       REFRESCOS"
    DISPLAY " "
    DISPLAY "                    ___"
    DISPLAY PIDE-LA-ORDEN WITH NO ADVANCING.
    ACCEPT CLAVE-ALIMENTO.
    IF CLAVE-ALIMENTO EQUAL CODIGO-HAMBURGUESAS THEN
        DISPLAY " "
        DISPLAY " CODIGO      PRODUCTO "
        DISPLAY " ------      --------- "
        DISPLAY "   011       HAMBURGESAS CON QUESO "
        DISPLAY "   012       HAMBURGESAS SIN QUESO "
        DISPLAY " "
        DISPLAY "                    ___"
        DISPLAY "TIPO DE HAMBURGUESA " WITH NO ADVANCING
        ACCEPT CLAVE-QUESO
        IF CLAVE-QUESO EQUAL CODIGO-CON-QUESO THEN
            ADD PRECIO-CON-QUESO TO SUMA
            DISPLAY DESCRIPCION-CON-QUESO
        ELSE
            IF CLAVE-QUESO EQUAL CODIGO-SIN-QUESO THEN
                ADD PRECIO-SIN-QUESO TO SUMA
                DISPLAY DESCRIPCION-SIN-QUESO.
*           NOELSE
*           ENDIF
*       ENDIF
*   NOELSE
*   ENDIF
    IF CLAVE-ALIMENTO EQUAL CODIGO-PAPAS THEN
        ADD PRECIO-PAPAS TO SUMA
        DISPLAY DESCRIPCION-PAPAS.
*   NOELSE
*   ENDIF
    IF CLAVE-ALIMENTO EQUAL CODIGO-REFRESCOS THEN
        DISPLAY " "
        DISPLAY " CODIGO      PRODUCTO "
        DISPLAY " ------      --------- "
        DISPLAY "   031       REFRESCO CHICO "
        DISPLAY "   032       REFRESCO MEDIANO "
        DISPLAY "   033       REFRESCO GRANDE  "
        DISPLAY " "
        DISPLAY "                   ___"
        DISPLAY "TIPO DE REFRESCO " WITH NO ADVANCING.
        ACCEPT CLAVE-REFRESCO
        IF CLAVE-REFRESCO EQUAL CODIGO-CHICO THEN
            ADD PRECIO-CHICO TO SUMA
            DISPLAY DESCRIPCION-REFRESCO-CHICO
        ELSE
            IF CLAVE-REFRESCO EQUAL CODIGO-MEDIANO THEN
                ADD PRECIO-MEDIANO TO SUMA
                DISPLAY DESCRIPCION-REFRESCO-MEDIANO
            ELSE
                IF CLAVE-REFRESCO EQUAL CODIGO-GRANDE THEN
                    ADD PRECIO-GRANDE TO SUMA
                    DISPLAY DESCRIPCION-REFRESCO-GRANDE.
*               NOELSE
*               ENDIF
*           ENDIF
*   ENDIF
```

```
      NOELSE
      ENDIF
      PERFORM PEDIDO.
PEDIDO.
      MOVE 'A' TO PEDIDO-ALIMENTOS.
      PERFORM PANTALLA-PIDE-PEDIDO UNTIL PEDIDO-ALIMENTOS EQUAL 'S'
                                      OR PEDIDO-ALIMENTOS EQUAL 'N'.
PANTALLA-PIDE-PEDIDO.
      DISPLAY PIDE-PEDIDO WITH NO ADVANCING.
      ACCEPT PEDIDO-ALIMENTOS.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SOLUCION-DELA-EC-DE-2DO-GRADO.
AUTHOR. GRUPO13.
INSTALLATION. CECAFI.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. DIGITAL-VAX.
OBJECT-COMPUTER. DIGITAL-VAX.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATOS ASSIGN TO SYS$INPUT.
    SELECT RENGLON ASSIGN TO SYS$OUTPUT.
DATA DIVISION.
FILE SECTION.

FD DATOS
          DATA RECORD IS DATOS1.
01 DATOS1.
    03 A PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
    03 B PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
    03 C PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
FD RENGLON
          DATA RECORD IS LINEA.
01 LINEA  PIC X(80).
WORKING-STORAGE SECTION.
01 LINEA-AUX.
    03   FILLER PIC X(16) VALUE "EL RESULTADO ES:".
    03   FILLER PIC X(02) VALUE ALL " ".
    03   RESULT PIC +9(02).9(02).
    03   FILLER PIC X(55) VALUE ALL " ".
77 RESULTADO PIC S9(02)V9(02) VALUE ZEROES.
77 DISC PIC S9(02)V9(02).
77 X1 PIC S9(02)V9(02).
77 X2 PIC S9(02)V9(02).
PROCEDURE DIVISION.
INICIO.
    OPEN INPUT DATOS
         OUTPUT RENGLON
    display " "
    display "     Solucion de la ecuacion cuadratica. "
    display " "
    display " 3 datos en linea. 1: A,  2: B,  3: C,"
    display "                    s : signo     eeddi valor, sin punto decimal."
    display "                                  ee enteros "
    display "                                  dd decimales "
    display " "
    display "seeddseeddseedd"
    READ DATOS AT END DISPLAY "NO HUBO DATOS" STOP RUN.

    if a = 0 and b = 0 and c = 0 stop run.
    if a = 0 and b = 0 stop run.
    if a = 0 and c = 0 stop run.

    IF A = 0 THEN
       DISPLAY "ECUACION DE PRIMER GRADO"
       MOVE ZEROES TO X2
       COMPUTE X1 = ((-1 * C)/B)
       MOVE X1 TO RESULT
       WRITE LINEA FROM LINEA-AUX
```

```
ELSE
    COMPUTE DISC = (B ** 2 - 4 * A * C)
    IF DISC < 0 THEN
        DISPLAY "RAICES COMPLEJAS NO PUEDO RESOLVERLO"
        STOP RUN
    ELSE
        display "Raices reales"
        COMPUTE X1 = ((-B - DISC ** 0.5)/(2*A))
        COMPUTE X2 = ((-B + DISC ** 0.5)/(2*A))
        MOVE X1 TO RESULT
        WRITE LINEA FROM LINEA-AUX
        MOVE X2 TO RESULT
        WRITE LINEA FROM LINEA-AUX.
STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SOLUCION-DELA-EC-DE-2DO-GRADO.
AUTHOR. GRUPO13.
INSTALLATION. CECAFI.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. DIGITAL-VAX.
OBJECT-COMPUTER. DIGITAL-VAX.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATOS ASSIGN TO SYS$INPUT.
    SELECT RENGLON ASSIGN TO SYS$OUTPUT.
DATA DIVISION.
FILE SECTION.

FD DATOS
          DATA RECORD IS DATOS1.
01 DATOS1.
     03 A PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
     03 B PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
     03 C PIC S9(02)V9(02)
          SIGN IS LEADING SEPARATE.
FD RENGLON
     2    DATA RECORD IS LINEA.
01 LINEA  PIC X(80).
WORKING-STORAGE SECTION.
01 LINEA-AUX.
     03  FILLER PIC X(16) VALUE 'EL RESULTADO ES:'.
     03  FILLER PIC X(02) VALUE ALL " ".
     03  RESULT PIC +9(02).9(02).
     03   FILLER PIC X(55) VALUE ALL " ".
77  RESULTADO PIC S9(02)V9(02) VALUE ZEROES.
77 DISC PIC S9(02)V9(02).
77 X1 PIC S9(02)V9(02).
77 X2 PIC S9(02)V9(02).
PROCEDURE DIVISION.
INICIO.
    OPEN INPUT DATOS
         OUTPUT RENGLON
    display " "
    display "     Solucion de la ecuacion cuadratica. "
    display " "
    display " 3 datos en linea, 1: A,  2: B,  3: C,"
    display "                    s : signo    seddd valor sin punto decimal "
    display "                                 ee enteros "
    display "                                 dd decimales "
    display " "
    display 'seeddseeddseedd'
    READ DATOS AT END DISPLAY "NO HUBO DATOS" STOP RUN.

    if a = 0 and b = 0 and c = 0 stop run.
    if a = 0 and b = 0 stop run.
    if a = 0 and c = 0 stop run.

    IF A = 0 THEN
       DISPLAY 'ECUACION DE PRIMER GRADO'
       MOVE ZEROES TO X2
       COMPUTE X1 = ((-1 * C)/B)
       MOVE X1 TO RESULT
       WRITE LINEA FROM LINEA-AUX
```

```
    COMPUTE DISC = (B ** 2 - 4 * A * C)
    IF DISC < 0 THEN
        DISPLAY "RAICES COMPLEJAS NO PUEDO RESOLVERLO"
        STOP RUN
    ELSE
        display "Raices reales"
        COMPUTE X1 = ((-B - DISC ** 0.5)/(2*A))
        COMPUTE X2 = ((-B + DISC ** 0.5)/(2*A))
        MOVE X1 TO RESULT
        WRITE LINEA FROM LINEA-AUX
        MOVE X2 TO RESULT
        WRITE LINEA FROM LINEA-AUX.
STOP RUN.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. FORMATO-LIBRE.
 AUTHOR. YO
 INSTALLATION. VAX

 ENVIRONMENT DIVISION.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 NUM PIC X(09).
 77 SIGNOS PIC 9(02) VALUE ZEROES.
 77 NEG PIC 9(02) VALUE ZEROES.
 77 PUNTOS PIC 9(02) VALUE ZEROES.
 77 POS PIC 9(02) VALUE ZEROES.
 77 CONT PIC 9(02) VALUE ZEROES.
 77 CONT1 PIC 9(02) VALUE ZEROES.
 77 IMPRESION  PIC ----------.9(09).
 01  CAMPO.
     03  A PIC X(09).
     03  AA PIC S9(09)V9(09) VALUE ZEROES.
     03  AAA REDEFINES AA.
         05  INTEG PIC S9(09).
         05 ENTERO  REDEFINES INTEG PIC X(09) JUST RIGHT .
         05  DECS PIC 9(09).
         05  DECIMOS REDEFINES DECS PIC X(09).
     03  BBB PIC S9(09)V9(09).
 PROCEDURE DIVISION.
 INICIO.
     DISPLAY " Inserte un numero en cualquier formato."
     ACCEPT NUM
         INSPECT NUM TALLYING PUNTOS FOR ALL "."
         IF (PUNTOS = 1) OR (PUNTOS = 0) THEN
             INSPECT NUM TALLYING NEG FOR ALL "-",POS FOR ALL "+"
             ADD NEG POS GIVING SIGNOS
             IF (SIGNOS = 0) OR (SIGNOS = 1) THEN
                 INSPECT NUM REPLACING ALL "+" BY "0" ALL "-"
                       BY "0"  ALL " " BY "0"
                 MOVE ZEROES TO CONT
                 MOVE ZEROES TO CONT1
                     INSPECT NUM TALLYING CONT FOR CHARACTERS BEFORE
                 INITIAL "."
                 IF CONT < 09 THEN
                     MOVE NUM(1:CONT) TO ENTERO
                     ADD 2 TO CONT
                     SUBTRACT CONT FROM 09 GIVING CONT1
                     MOVE NUM(CONT:CONT1) TO DECIMOS
                     MOVE AAA TO BBB
                 ELSE
                     MOVE NUM TO BBB
                 END-IF
                 INSPECT BBB REPLACING ALL " " BY "0".
                 IF NEG = 1 THEN
                     MULTIPLY -1 BY BBB
                 NOELSE
                 END-IF
                 MOVE BBB TO IMPRESION
                 DISPLAY " Resultado : ", IMPRESION
                 STOP RUN
             ELSE
                 DISPLAY "ERROR: MAS DE UN SIGNO INCLUIDO"
             END-IF
         ELSE
```

```
        DISPLAY "ERROR: MAS DE UN PUNTO ENCONTRADO"
    END-IF.
```

```
PROGRAM-ID.                 DEMOSTRACION-13.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
77    ALFANUMERICOS         PIC      X(10).
77    CONT-1                PIC      9(02)      VALUE      ZEROES.
77    CONT-2                PIC      9(02)      VALUE      ZEROES.
77    ENTEROS               PIC      9(05).
77    DECIMALES             PIC      9(05).
77    ALFA-DECIMALES        PIC      X(05).
PROCEDURE DIVISION.

PRINCIPIO.

*
*   ALFANUMERICOS(DESDE-DONDE:POR-CUANTOS)
*
    MOVE "100.10" TO ALFANUMERICOS.
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ".".
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS BEFORE INITIAL " ".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS.
    ADD 2 TO CONT-1.
    SUBTRACT CONT-1 FROM CONT-2 GIVING CONT-2.
    ADD 1 TO CONT-2
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES.
    DISPLAY " ENTERO ---> " ENTEROS.
    DISPLAY " DECIMALES > " DECIMALES.
    STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-13.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
77    ALFANUMERICOS         PIC      X(10).
77    CONT-1                PIC      9(02)      VALUE      ZEROES.
77    CONT-2                PIC      9(02)      VALUE      ZEROES.
77    ENTEROS               PIC      9(05).
77    DECIMALES             PIC      9(05).
77    ENTEROS-2             PIC      9(05).
77    DECIMALES-2           PIC      9(05).
77    ALFA-DECIMALES        PIC      X(05).
PROCEDURE DIVISION.

PRINCIPIO.

*
*   ALFANUMERICOS(DESDE-DONDE:POR-CUANTOS)
*
    MOVE "100.10" TO ALFANUMERICOS.
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ".".
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS BEFORE INITIAL " ".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS.
    ADD 2 TO CONT-1.
    SUBTRACT CONT-1 FROM CONT-2 GIVING CONT-2.
    ADD 1 TO CONT-2.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES.
    DISPLAY " ENTERO --> " ENTEROS.
    DISPLAY " DECIMALES > " DECIMALES.

* SEGUNDO NUMERO

    MOVE ZEROES TO CONT-1.
    MOVE ZEROES TO CONT-2.
    MOVE "100.10" TO ALFANUMERICOS.
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ".".
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS BEFORE INITIAL " ".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS-2.
```

```
      SUBTRACT CONT-1 FROM CONT-2 GIVING CONT-2.
      ADD 1 TO CONT-2.
      MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
      INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
      MOVE ALFA-DECIMALES TO DECIMALES-2.
      DISPLAY " ENTERO ---> " ENTEROS-2.
      DISPLAY " DECIMALES > " DECIMALES-2.

X-SUMA.

      ADD ENTEROS TO ENTEROS-2.
      ADD DECIMALES TO DECIMALES-2.
      DISPLAY ENTEROS-2 "," DECIMALES-2.

      STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-13.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
77    ALFANUMERICOS         PIC      X(10).
77    CONT-1                PIC      9(02)      VALUE      ZEROES.
77    CONT-2                PIC      9(02)      VALUE      ZEROES.
77    ENTEROS               PIC      9(05).
77    DECIMALES             PIC      9(05).
77    ENTEROS-2             PIC      9(05).
77    DECIMALES-2           PIC      9(05).
77    ALFA-DECIMALES        PIC      X(05).

PROCEDURE DIVISION.

PRINCIPIO.

*
*    ALFANUMERICOS(DESDE-DONDE:POR-CUANTOS)
*
*     MOVE "100.10" TO ALFANUMERICOS.

     DISPLAY "DAME UN NUMERO  : " WITH NO ADVANCING.
     ACCEPT ALFANUMERICOS.
     INSPECT ALFANUMERICOS TALLYING
             CONT-1 FOR CHARACTERS BEFORE INITIAL ".".
     INSPECT ALFANUMERICOS TALLYING
             CONT-2 FOR CHARACTERS BEFORE INITIAL " ".
     MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS.
     ADD 2 TO CONT-1.
     SUBTRACT CONT-1 FROM CONT-2 GIVING CONT-2.
     ADD 1 TO CONT-2.
     MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
     INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
     MOVE ALFA-DECIMALES TO DECIMALES.

* SEGUNDO NUMERO

     MOVE ZEROES TO CONT-1.
     MOVE ZEROES TO CONT-2.
*     MOVE "100.10" TO ALFANUMERICOS.

     DISPLAY "DAME OTRO NUMERO  : " WITH NO ADVANCING.
     ACCEPT ALFANUMERICOS.
```

```
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ".".
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS BEFORE INITIAL " ".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS-2.
    ADD 2 TO CONT-1.
    SUBTRACT CONT-1 FROM CONT-2 GIVING CONT-2.
    ADD 1 TO CONT-2.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES-2.

XSUMA

    ADD ENTEROS TO ENTEROS-2.
    ADD DECIMALES TO DECIMALES-2.
    DISPLAY " SU SUMA ES : " ENTEROS-2 "," DECIMALES-2.

    STOP RUN.
```

```
PROGRAM-ID.                 DEMOSTRACION-19.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1.
       03 SUB-1.
          05 SUB-1A PIC X(10).
          05 SUB-1B PIC X(05).
          05 SUB-1C PIC X(02).
77     CONT PIC 9(03) VALUE ZEROES.

PROCEDURE DIVISION.

PRINCIPIO.

           DISPLAY "Cont REGISTRO-1          ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           DISPLAY "REGISTRO-1    : " REGISTRO-1.
           DISPLAY "REG-1 SUB-1A  : " SUB-1A IN REGISTRO-1.
           DISPLAY "REG-1 SUB-1B  : " SUB-1B IN REGISTRO-1.
           DISPLAY "REG-1 SUB-1C  : " SUB-1C IN REGISTRO-1.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J".
           DISPLAY "CONT All Jota      : " CONT.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING "J".
           DISPLAY "CONT Leading Jota : " CONT.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR CHARACTERS.
           DISPLAY "CONT Characters    : " CONT.
* AFTER
           DISPLAY "Cont after REGISTRO-1  ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J" AFTER INITIAL "@".
           DISPLAY "CONT All Jota after @      : " CONT.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING "J" AFTER INITIAL "@".
           DISPLAY "CONT Leading Jota after @ : " CONT.
* BEFORE
           DISPLAY "Cont before REGISTRO-1 ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J" BEFORE INITIAL "@".
           DISPLAY "CONT All Jota before @     : " CONT.
```

```
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING "J" BEFORE INITIAL
           DISPLAY "CONT Leadins Jota before @ : " CONT.
      * REPLACING
           DISPLAY "Replacing all REGISTRO-1 ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING CHARACTERS BY "A".
           DISPLAY "Replacing characters by A  : " REGISTRO-1.
           DISPLAY "Replacing REGISTRO-1       ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING ALL "J" BY "N".
           DISPLAY "Replacing All      J by N   : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING LEADING "N" BY "J".
           DISPLAY "Replacing Leading N by J    : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING FIRST "A" BY "B".
           DISPLAY "Replacing first    A by B   : " REGISTRO-1.
      * REPLACING BEFORE
           DISPLAY "Replacing before REGISTRO-1 ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING ALL "J" BY "N" BEFORE "@".
           DISPLAY "Replacing All      J by N before @ : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING LEADING "N" BY "J" BEFORE "@".
           DISPLAY "Replacing Leading N by J before @ : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING FIRST "A" BY "B" BEFORE "@".
           DISPLAY "Replacing first    A by B before @ : " REGISTRO-1.
      * REPLACING AFTER
           DISPLAY "Replacing after REGISTRO-1  ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING ALL "J" BY "N" AFTER "@".
           DISPLAY "Replacing All      J by N after @  : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING LEADING "N" BY "J" AFTER "@".
           DISPLAY "Replacing Leading N by J after @  : " REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING FIRST "A" BY "B" AFTER "@".
           DISPLAY "Replacing first    A by B after @  : " REGISTRO-1.
      * CONVERTING
           DISPLAY "Converting REGISTRO-1       ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 CONVERTING "JORGE" TO "JOSE ".
           DISPLAY "Converting JORGE TO JOSE    : " REGISTRO-1.
           STOP RUN.
```

```
PROGRAM-ID.                DEMOSTRACION-21.
AUTHOR.                    JORGE-VALERIO.
                           CD UNIVERSITARIA UNAM
                           MEXICO.
INSTALLATION.              CECAFI.
                           CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.              29-SEP-83.
DATE-COMPILED.             29-SEP-83.
SECURITY.                  PUBLICA.
                           ESTE PROGRAMA ES PARA DEMOSTRACION.
                           PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                           MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.           VAX11-780.
SOURCE-COMPUTER.           VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01     REGISTRO-1 PIC X(80).
77     CONT PIC 9(03) VALUE ZEROES.

PROCEDURE DIVISION.

PRINCIPIO.

           PERFORM INICIO.
           PERFORM INSPECT-TOTAL.
           PERFORM INSPECT-AFTER.
           PERFORM INSPECT-BEFORE.
           PERFORM REPLACING-TOTAL.
           PERFORM REPLACING-AFTER.
           PERFORM REPLACING-BEFORE.
           PERFORM CONVERTING-TOTAL.
           STOP RUN.
INICIO.

           PERFORM LIMPIA 24 TIMES.
           PERFORM LETRERO.
           PERFORM LIMPIA 05 TIMES.

LIMPIA.

           DISPLAY " ".

LETRERO.

           DISPLAY "CECAFI> Demostracion de la instruccion INSPECT ".

INSPECT-TOTAL.

           DISPLAY "Introduce un nombre en el cual se contaran todas las J.".
           DISPLAY "INPUT> Cont all J nombre ---> " WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J".
           DISPLAY "OUTPUT> CONT All J : " CONT.
           MOVE ZEROES TO CONT.
           PERFORM LIMPIA 2 TIMES.
```

```
PROGRAM-ID.             DEMOSTRACION-27.
*****************************************************************
*
*       SUMA (APROXIMADAMENTE) DOS NUMEROS REALES
*       INRODUCIDOS POR ACCEPT ALFANUMERICAMENTE
*       Y USANDO EL INSPECT, MOVE, Y ADD
*
*****************************************************************

AUTHOR.                 JORGE-VALERIO.
                        CD UNIVERSITARIA UNAM
                        MEXICO.
INSTALLATION.           CECAFI.
                        CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.           29-SEP-83.
DATE-COMPILED.          29-SEP-83.
SECURITY.               PRIVADA.
                        ESTE PROGRAMA ES PARA DEMOSTRACION.
                        PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                        MEDIO.

*****************************************************************
*
   ENVIRONMENT   DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.        VAX11-780.
SOURCE-COMPUTER.        VAX11-780.

*****************************************************************
*
   DATA          DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.
77   ALFANUMERICOS      PIC     X(10).
77   CONT-1             PIC     9(02)     VALUE     ZEROES.
77   CONT-2             PIC     9(02)     VALUE     ZEROES.
77   ENTEROS            PIC     9(05).
77   DECIMALES          PIC     9(05).
77   ENTEROS-2          PIC     9(05).
77   DECIMALES-2        PIC     9(05).
77   ALFA-DECIMALES     PIC     X(05).

*****************************************************************
*
   PROCEDURE DIVISION.
*
*****************************************************************

PRINCIPIO.

*****************************************************************
*                       PRIMER NUMERO
*****************************************************************

    DISPLAY "DAME UN NUMERO  : " WITH NO ADVANCING.
    ACCEPT ALFANUMERICOS.
    INSPECT ALFANUMERICOS REPLACING ALL " " BY "0".
```

```
           CONT-1 FOR CHARACTERS BEFORE INITIAL "."
    INSPECT ALFANUMERICOS TALLYING
           CONT-2 FOR CHARACTERS AFTER  INITIAL ".".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS.
    ADD 2 TO CONT-1.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES.

*****************************************************************************
*                        SEGUNDO NUMERO                                     *
*****************************************************************************

    MOVE ZEROES TO CONT-1.
    MOVE ZEROES TO CONT-2.
    DISPLAY "DAME OTRO  NUMERO  : " WITH NO ADVANCING.
    ACCEPT ALFANUMERICOS.
    INSPECT ALFANUMERICOS REPLACING ALL " " BY "0".
    INSPECT ALFANUMERICOS TALLYING
           CONT-1 FOR CHARACTERS BEFORE INITIAL "."
    INSPECT ALFANUMERICOS TALLYING
           CONT-2 FOR CHARACTERS AFTER  INITIAL ".".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS-2.
    ADD 2 TO CONT-1.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES-2.

*****************************************************************************
*                              SUMA                                         *
*****************************************************************************

    ADD ENTEROS TO ENTEROS-2.
    ADD DECIMALES TO DECIMALES-2.

*****************************************************************************
*                     MUESTRA LA SUMA                                       *
*****************************************************************************

    DISPLAY " SU SUMA ES : " ENTEROS-2 "," DECIMALES-2.

    STOP RUN.
```

```
PROGRAM-ID.                DEMOSTRACION-30.
*****************************************************************
*                                                               *
*       SUMA (APROXIMADAMENTE) DOS NUMEROS REALES               *
*       INTRODUCIDOS POR ACCEPT ALFANUMERICAMENTE               *
*       Y USANDO EL INSPECT, MOVE, Y ADD                        *
*                                                               *
*****************************************************************

AUTHOR.                    JORGE-VALERIO.
                           CD UNIVERSITARIA UNAM
                           MEXICO.
INSTALLATION.              CECAFI.
                           CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.              29-SEP-83.
DATE-COMPILED.             29-SEP-83.
SECURITY.                  PRIVADA.
                           ESTE PROGRAMA ES PARA DEMOSTRACION.
                           PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                           MEDIO.

*****************************************************************
*
   ENVIRONMENT    DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.           VAX11-780.
SOURCE-COMPUTER.           VAX11-780.

*****************************************************************
*
   DATA           DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.
77    LINEA                PIC    X(132).
77    ALFANUMERICOS        PIC    X(10).
77    CONT-1               PIC    9(02)      VALUE      ZEROES.
77    CONT-2               PIC    9(02)      VALUE      ZEROES.
77    ENTEROS              PIC    9(05).
77    DECIMALES            PIC    9(05).
77    ENTEROS-2            PIC    9(05).
77    DECIMALES-2          PIC    9(05).
77    ALFA-DECIMALES       PIC    X(05).

*****************************************************************
*
   PROCEDURE DIVISION.
*
*****************************************************************

PRINCIPIO.

*****************************************************************
*                       PRIMER NUMERO
*****************************************************************

    DISPLAY 'DAME UN NUMERO : ' WITH NO ADVANCING.
    ACCEPT LINEA.
```

```
    ADD 1 TO CONT-1.
    MOVE LINEA(CONT-1:10) TO ALFANUMERICOS.
    INSPECT ALFANUMERICOS REPLACING ALL " " BY "0".
    MOVE ZEROES TO CONT-1.
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ","
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS AFTER  INITIAL ",".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS.
    ADD 2 TO CONT-1.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES.

*****************************************************************************
*                        SEGUNDO NUMERO
*****************************************************************************

    MOVE ZEROES TO CONT-1.
    MOVE ZEROES TO CONT-2.
    DISPLAY "DAME OTRO  NUMERO  : " WITH NO ADVANCING.
    ACCEPT LINEA.
    INSPECT LINEA TALLYING CONT-1 FOR LEADING " ".
    ADD 1 TO CONT-1.
    MOVE LINEA(CONT-1:10) TO ALFANUMERICOS.
    INSPECT ALFANUMERICOS REPLACING ALL " " BY "0".
    MOVE ZEROES TO CONT-1.
    INSPECT ALFANUMERICOS TALLYING
            CONT-1 FOR CHARACTERS BEFORE INITIAL ","
    INSPECT ALFANUMERICOS TALLYING
            CONT-2 FOR CHARACTERS AFTER  INITIAL ",".
    MOVE ALFANUMERICOS(1:CONT-1) TO ENTEROS-2.
    ADD 2 TO CONT-1.
    MOVE ALFANUMERICOS(CONT-1:CONT-2) TO ALFA-DECIMALES.
    INSPECT ALFA-DECIMALES REPLACING ALL " " BY "0".
    MOVE ALFA-DECIMALES TO DECIMALES-2.

*****************************************************************************
*                           SUMA
*****************************************************************************

    ADD ENTEROS TO ENTEROS-2.
    ADD DECIMALES TO DECIMALES-2.

*****************************************************************************
*                      MUESTRA LA SUMA
*****************************************************************************

    DISPLAY " SU SUMA ES : " ENTEROS-2 "," DECIMALES-2.

    STOP RUN.
```

```
PROGRAM-ID.                DEMOSTRACION-34.
*****************************************************************
*                                                               *
*              PROGRAMA QUE REDUCE NOMBRES                      *
*                                                               *
*****************************************************************

AUTHOR.                    JORGE VALERIO.
                           CD UNIVERSITARIA UNAM
                           MEXICO.
INSTALLATION.              CECAFI.
                           CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.              29-SEP-83.
DATE-COMPILED.             29-SEP-83.
SECURITY.                  PRIVADA.
                           ESTE PROGRAMA ES PARA DEMOSTRACION.
                           PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                           MEDIO.

*****************************************************************
*                                                               *
   ENVIRONMENT     DIVISION.
*                                                               *
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.           VAX11-780.
SOURCE-COMPUTER.           VAX11-780.

*****************************************************************
*                                                               *
   DATA            DIVISION.
*                                                               *
*****************************************************************

WORKING-STORAGE SECTION.
77    LINEA-WS                    PIC    X(132)   VALUE   SPACES.
77    CONT1-WS                    PIC    9(03)    VALUE   ZEROES.
77    CONT2-WS                    PIC    9(03)    VALUE   ZEROES.
77    CONT3-WS                    PIC    9(03)    VALUE   ZEROES.
77    CONT4-WS                    PIC    9(03)    VALUE   ZEROES.
77    PIDE-CONT-WS                PIC    X(01)    VALUE     "S".
01    NOMBRE-WS.
      02 INICIAL-NOMBRE-WS        PIC    X(01)    VALUE   SPACES.
      02 APELLIDO-NOMBRE-WS       PIC    X(20)    VALUE   SPACES.

*****************************************************************
*                                                               *
   PROCEDURE DIVISION.
*                                                               *
*****************************************************************

EJECUTA.

    PERFORM PIDE-NOMBRE
            THRU PIDE-CONTINUACION
            TEST BEFORE UNTIL PIDE-CONT-WS EQUAL "N"
*   END-PERFORM
    PERFORM FIN-DE-PROGRAMA.

*****************************************************************
*                    PIDE NOMBRE COMPLETO                       *
```

```
*              LOS NOMBRES Y LOS APELLIDOS SE SEPARAN POR COMAS
*****************************************************************

PIDE-NOMBRE.

    DISPLAY " "
    DISPLAY " "
    DISPLAY "                          REDUCCION DE NOMBRES "
    DISPLAY " "
    DISPLAY " "
    DISPLAY "EL NOMBRE TENDRA LAS SIGUIENTES CARACTERISTICAS : "
    DISPLAY " "
    DISPLAY "<<<<< SEPARADOS NOMBRES Y APELLIDOS POR COMAS "
            "Y TERMINADO CON PUNTO  >>>>>"
    DISPLAY " "
    DISPLAY " "
    DISPLAY "DAME EL NOMBRE : " WITH NO ADVANCING.
    ACCEPT LINEA-WS.

*****************************************************************
*                       CHECA CONDICIONES                       *
*****************************************************************

CHECA-CONDICIONES.

    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ",".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTAN COMAS DE SEPARACION >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ".".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTA PUNTO                    >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    IF CONT1-WS NOT EQUAL 1 THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< HAY MAS DE UN PUNTO            >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ".".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS) TALLYING CONT3-WS FOR LEADING " ".
    IF CONT3-WS NOT EQUAL CONT2-WS THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< EL PUNTO NO ESTA AL FINAL      >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
```

```
*                    QUITA BLANCOS
*****************************************************************

QUITA-BLANCOS.

    PERFORM QUITA-HASTA-EL-PUNTO
        WITH TEST BEFORE UNTIL PIDE-CONT-WS IS EQUAL TO ".".
*   END-PERFORM

*****************************************************************
*              QUITA BLANCOS HASTA EL PUNTO
*****************************************************************

QUITA-HASTA-EL-PUNTO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE " ".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 1 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS)
        TALLYING CONT3-WS FOR LEADING " ".
    ADD CONT1-WS TO CONT3-WS.
    MOVE CONT2-WS TO CONT4-WS.
    SUBTRACT CONT3-WS FROM CONT2-WS.
    MOVE LINEA-WS(CONT3-WS:CONT2-WS) TO LINEA-WS(CONT1-WS:CONT4-WS).
    ADD -1 TO CONT1-WS.
    MOVE LINEA-WS(CONT1-WS:1) TO PIDE-CONT-WS.

*****************************************************************
*                        INICIAL                                *
*****************************************************************

PONE-INICIAL.

    MOVE LINEA-WS(1:1) TO INICIAL-NOMBRE-WS.

*****************************************************************
*                     PRIMER APELLIDO                           *
*****************************************************************

PONE-APELLIDO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ",".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS BEFORE " ".
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT3-WS.
    ADD 1 TO CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT3-WS)
        TALLYING CONT3-WS FOR CHARACTERS BEFORE " ".
    IF CONT3-WS IS ZERO THEN
        MOVE LINEA-WS(CONT1-WS:CONT2-WS) TO APELLIDO-NOMBRE-WS
    ELSE
        MOVE LINEA-WS(CONT1-WS:CONT3-WS) TO APELLIDO-NOMBRE-WS
    END-IF.
```

```
*****************************************************************
*                  DESPLIEGA LA INFORMACION DEL REGISTRO
*****************************************************************

DESPLIEGA-INFORMACION.

    DISPLAY " ".
    DISPLAY "            CONTENIDO DEL REGISTRO DE NOMBRE      ".
    DISPLAY " ".
    DISPLAY " ".
    DISPLAY " INICIAL  : " INICIAL-NOMBRE-WS.
    DISPLAY " APELLIDO : " APELLIDO-NOMBRE-WS.
    DISPLAY " ".
    DISPLAY " ".

*****************************************************************
*                  PIDE CONTINUACION
*****************************************************************

PIDE-CONTINUACION.

    PERFORM PIDE-SI-O-NO
            WITH TEST BEFORE UNTIL
            NOT    (PIDE-CONT-WS IS NOT EQUAL "S"
                AND PIDE-CONT-WS IS NOT EQUAL "N").
*   END-PERFORM

*****************************************************************
*                  PIDE-SI-O-NO
*****************************************************************

PIDE-SI-O-NO.

    MOVE SPACES TO PIDE-CONT-WS
    DISPLAY " ".
    DISPLAY " QUIERES CONTINUAR EL PROGRAMA (SI/NO) : " WITH NO ADVANCING.
    ACCEPT PIDE-CONT-WS.

*****************************************************************
*                  FIN DEL PROGRAMA
*****************************************************************

FIN-DE-PROGRAMA.

    STOP RUN.
```

```
*****************************************************************
*          **********   DIVISION DE IDENTIFICACION   **********
*
    IDENTIFICATION DIVISION.
PROGRAM-ID.              PROGRAMA-DE-PRUEBA-1.
AUTHOR.                  JORGE Y ARTURO.
*                          ***** * ******
INSTALLATION.            CECAFI.
DATE-WRITTEN.            8-SEPTIEMBRE-1982.
DATE-COMPILED.
SECURITY.                PRIVADA.
*
*          EL PROGRAMA ESCRIBE LA FECHA QUE SE LA DAMOS POR PANTALLA
*          REALIZA UNA OPERACION NUMERICA Y ESCRIBE ALGUNOS LETREROS.
*          FUE HECHO COMO PRUEBA PARA EL CURSO DE COBOL.
/
*
*          **********   DIVISION DEL MEDIO AMBIENTE   **********
*
ENVIRONMENT      DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.         DIGITAL-VAX.
OBJECT-COMPUTER.         DIGITAL-VAX.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
*
*          **********   DECLARACION DE ARCHIVOS I/O   **********
*
SELECT DATOS-LEC ASSIGN TO SYS$INPUT.
SELECT IMPRIME ASSIGN TO SYS$OUTPUT.
/
*
*          **********   DIVISION DE DATOS   **********
*
*
DATA DIVISION.
FILE SECTION.
FD DATOS-LEC
        DATA RECORDS ARE
                        DATOS-FECHA
                        DATOS-OPERA.
*          *** DESCRIPCION DEL REGISTRO PARA LOS DATOS DE LA FECHA ***
*
01      DATOS-FECHA.
        03  DIA                          PIC 9(02).
        03  MES                          PIC 9(02).
        03  ANIO                         PIC 9(02).
        03  FILLER                       PIC X(74).
*
* *** DESCRIPCION DEL REGISTRO PARA  LOS DATOS DE LA OPERACION ***
*
01      DATOS-OPERA.
        03  DATO-1                       PIC 9(05)V9(02).
        03  DATO-2                       PIC 9(05)V9(02).
        03  FILLER                       PIC X(64).
*
*          **********   ARCHIVO DE RESULTADOS   **********
*
FD      IMPRIME.
01      LINEA-DE-RESULTADOS               PIC X(132).
/
*
*          **********  REGISTROS INTERNOS AUXILIARES **********
```

```
*
WORKING-STORAGE SECTION.
77      SUMA                                PIC 9(09)V9(02).
77  HAY-DATOS                          PIC X(02) VALUE "SI".
    88  HAY-DATOS-OK                       VALUE "SI".
*
*          ***************  ENCABEZADOS  DE  IMPRESION  ***************
*
01      ENCA-UNO.
    03  FILLER                         PIC X(08) VALUE " FECHA: ".
        03  DIA-SAL                        PIC 9(02).
    03  FILLER                         PIC X(01) VALUE "/".
        03  MES-SAL                        PIC 9(02).
    03  FILLER                         PIC X(01) VALUE "/".
        03  ANIO-SAL                       PIC 9(02).
        03  FILLER                         PIC X(116) VALUE SPACES.
01      PRUEBA.
        03  PRUEBA-FECHA                   PIC 99/99/99.
    03  FILLER                         PIC X(04) VALUE "HOLA".
        03  FILLER                         PIC X(120) VALUE SPACES.
*
01      ENCA-DOS.
        03  FILLER                         PIC X(30) VALUE SPACES.
    03  FILLER                         PIC X(73) VALUE "U N I V E R
-       " S I D A D  N A C I O N A L
-       "  A U T O N O M A ,  D E  M E
-       " X I C O".
        03  FILLER                         PIC X(29) VALUE SPACES.
*
01      ENCA-TRES.
        03  FILLER                         PIC X(24) VALUE SPACES.
        03  FILLER                         PIC X(84) VALUE "C E N T R O
-       "  D E  C A L C U L O  D E
-       "L A  F A C U L T A D  D E
-       "I N G E N I E R I A".
        03  FILLER                         PIC X(24) VALUE SPACES.
*
01      RESULTADOS.
        03  FILLER                         PIC X(24) VALUE SPACES.
    03  FILLER                         PIC X(13) VALUE "LA SUMA DE
-       " ".
        03  DAT01-SAL                      PIC Z(09).9(02).
    03  FILLER                         PIC X(08) VALUE "  Y DE  ".
        03  DAT02-SAL                      PIC Z(09).9(02).
    03  FILLER                         PIC X(15) VALUE "  ES IGUAL
-       " A  ".
        03  SUMA-SAL                       PIC Z(09).9(02).
        03  FILLER                         PIC X(37) VALUE SPACES.
/
*
*          ************  DIVISION  DE  PROCEDIMIENTOS  ************
*
PROCEDURE DIVISION.
*
*          ***************  PROGRAMA  PRINCIPAL  ***************
*
COMIENZA.
        PERFORM ABRE-LOS-ARCHIVOS.
        READ DATOS-LEC AT END
        DISPLAY "SE ACABARON LOS DATOS"
        STOP RUN.
*       END-READ.
        MOVE DIA TO DIA-SAL
```

```
           MOVE MES TO MES-SAL
           MOVE ANIO TO ANIO-SAL
           READ DATOS-LEC AT END
           MOVE "NO" TO HAY-DATOS.
*           ENDREAD
           PERFORM REALIZA-CALCULOS UNTIL NOT HAY-DATOS-OK.
           PERFORM CIERRA-LOS-ARCHIVOS.
           STOP RUN.
*             ******* TERMINA EL PROGRAMA PRINCIPAL ********
/
*
*             ************** ZONA DE PROCEDIMIENTOS **************
*
ABRE-LOS-ARCHIVOS.
           OPEN
              INPUT DATOS-LEC
              OUTPUT IMPRIME.
IMP-ENCA.
           MOVE SPACES TO LINEA-DE-RESULTADOS
           WRITE LINEA-DE-RESULTADOS FROM ENCA-UNO
           WRITE LINEA-DE-RESULTADOS FROM ENCA-DOS AFTER 1 LINES.
           WRITE LINEA-DE-RESULTADOS FROM ENCA-TRES AFTER 2 LINES.
*ENDPER IMP-ENCA
REALIZA-CALCULOS.
           PERFORM IMP-ENCA
           COMPUTE SUMA = DATO-1 + DATO-2
           MOVE DATO-1 TO DATO1-SAL
           MOVE DATO-2 TO DATO2-SAL
           MOVE SUMA TO SUMA-SAL
           WRITE LINEA-DE-RESULTADOS FROM RESULTADOS AFTER 10 LINES
           READ DATOS-LEC AT END
           MOVE "NO" TO HAY-DATOS.
*           ENDREAD
CIERRA-LOS-ARCHIVOS.
           CLOSE
              DATOS-LEC
              IMPRIME.
*
*             ********** EL PROGRAMA PRUEBA HA TERMINADO **********
*
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     CHECA-SECUENCIA-TARJETAS.
AUTHOR.         JORGE VALERIO.
INSTALLATION.   CECAFI.
DATE-WRITTEN.   25-OCT-84.
DATE-COMPILED.
SECURITY.       PROGRAMA QUE CUENTA CUANTOS
                REGISTROS  SON INTRODUCIDOS
                Y CHECA SECUENCIA DE DATOS.

*               REVISADO 26 DE ENERO DE 1985.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NAME-FILE ASSIGN TO 'EVALERIO.COBOL.DATOS]CAPENEP.DAT'.

DATA DIVISION.
FILE SECTION.
FD NAME-FILE
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS RECORD-FILE
    LABEL RECORDS ARE STANDARD.
01 RECORD-FILE.
   03 FILLER          PIC X(12).
   03 NUMERO-TARJETA  PIC 9(01).
   03 FILLER          PIC X(67).
WORKING-STORAGE SECTION.
77 HAY-DATOS PIC X(01) VALUE 'S'.
77 CONTADOR  PIC 9(04) VALUE ZERO.
77 BASE      PIC 9(01) VALUE 1.

PROCEDURE DIVISION.
PROGRAMA-PRINCIPAL.
    OPEN INPUT NAME-FILE.
    PERFORM PROCESO UNTIL HAY-DATOS IS EQUAL 'N'.
    PERFORM SALIDA-FINAL.
    STOP RUN.

PROCESO.
    READ NAME-FILE AT END MOVE 'N' TO HAY-DATOS.
    IF HAY-DATOS IS EQUAL 'S' THEN
       PERFORM CHECA-SECUENCIA
       PERFORM INCREMENTA-SECUENCIA.
*   END-IF

CHECA-SECUENCIA.
    IF NUMERO-TARJETA NOT EQUAL BASE THEN
       DISPLAY 'FALTA LA TARJETA ' BASE ' EN LA SECUENCIA ' CONTADOR
       MOVE NUMERO-TARJETA  TO BASE.
*   END-IF

INCREMENTA-SECUENCIA.
    ADD 1 TO CONTADOR.
    ADD 1 TO BASE.
    IF BASE GREATER THAN 6 THEN
       MOVE 1 TO BASE.
*   END-IF

SALIDA-FINAL.
```

DISPLAY 'NUMERO DE TARJETAS LEIDAS ' CONTADOR.

```
*
*    **********    DIVISION DE IDENTIFICACION    **********
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID.          PROGRAMA-DE-PRUEBA-1.
 AUTHOR.              GOYO Y ARTURO.
                      **** * ******
 INSTALLATION.        CECAFI.
 DATE-WRITTEN.        9-NOVIEMBRE-1983.
 DATE-COMPILED.
 SECURITY.            PRIVADA.
*
*      EL PROGRAMA REALIZA UNA SUMA Y ESCRIBE SUS RESULTADOS.
*      FUE HECHO COMO PRUEBA PARA EL CURSO DE COBOL.
/
*
*      **********   DIVISION DEL MEDIO AMBIENTE   **********
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.     DIGITAL-VAX.
 OBJECT-COMPUTER.     DIGITAL-VAX.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*
*     **********  DECLARACION DE ARCHIVOS I/O   **********
*
 SELECT DATOS-LEC ASSIGN TO "DATENT.DAT".
 SELECT IMPRIME ASSIGN TO "SALIDA.DAT".
/
*
*     **********   DIVISION DE DATOS   **********
*
*
 DATA DIVISION.
 FILE SECTION.
 FD DATOS-LEC
     RECORD CONTAINS 80 CHARACTERS
     DATA RECORD IS DATOS-OPERA.
*
* *** DESCRIPCION DEL REGISTRO PARA LOS DATOS DE LA OPERACION ***
*
 01  DATOS-OPERA.
     03  DATO-1                          PIC 9(02).
     03  DATO-2                          PIC 9(02).
     03  FILLER                          PIC X(76).
*
*     **********  ARCHIVO DE RESULTADOS   **********
*
 FD  IMPRIME
      RECORD CONTAINS 132 CHARACTERS
      DATA RECORD IS LINEA-DE-RESULTADOS.
 01  LINEA-DE-RESULTADOS                 PIC X(132).
/
*
*     **********  REGISTROS INTERNOS AUXILIARES  **********
*
 WORKING-STORAGE SECTION.
 77  SUMA                                PIC 9(03)V9(02).
 77  HAY-DATOS                           PIC X(02) VALUE "SI".
*
*     **********  ENCABEZADOS DE IMPRESION  **********
*
```

```
 01  RESULTADOS.
     03  FILLER                          PIC X(24) VALUE SPACES.
     03  FILLER                          PIC X(12) VALUE 'LA SUMA DE
                                         ' '.
     03  DATO1-SAL                       PIC 9(02).9(02).
     03  FILLER                          PIC X(08) VALUE ' Y DE  '.
     03  DATO2-SAL                       PIC 9(02).9(02).
     03  FILLER                          PIC X(15) VALUE '  ES IGUAL
                                         ' A  '.
     03  SUMA-SAL                        PIC Z(03).9(02).
     03  FILLER                          PIC X(55) VALUE SPACES.
/
*
*       ************  DIVISION  DE  PROCEDIMIENTOS  ************
*
 PROCEDURE DIVISION.
*
*       ****************   PROGRAMA   PRINCIPAL   ****************
*
 COMIENZA.
     PERFORM ABRE-LOS-ARCHIVOS.
     READ DATOS-LEC AT END
         DISPLAY 'SE ACABARON LOS DATOS'
     STOP RUN
     END-READ
     PERFORM REALIZA-CALCULOS UNTIL HAY-DATOS = 'NO'
     PERFORM CIERRA-LOS-ARCHIVOS.
     STOP RUN.
*       *******  TERMINA  EL  PROGRAMA  PRINCIPAL  ********
/
*
*       **************  ZONA  DE  PROCEDIMIENTOS  **************
*
 ABRE-LOS-ARCHIVOS.
     OPEN
         INPUT DATOS-LEC
         OUTPUT IMPRIME.
 REALIZA-CALCULOS.
     MOVE ZEROES TO SUMA
     ADD DATO-1 DATO-2 TO SUMA
     DISPLAY 'SUMA = ' SUMA
     MOVE DATO-1 TO DATO1-SAL
     MOVE DATO-2 TO DATO2-SAL
     MOVE SUMA TO SUMA-SAL
     WRITE LINEA-DE-RESULTADOS FROM RESULTADOS AFTER 10 LINES
     READ DATOS-LEC AT END
         MOVE 'NO' TO HAY-DATOS
     END-READ.
 CIERRA-LOS-ARCHIVOS.
     CLOSE
         DATOS-LEC
         IMPRIME.
*
*       **********  EL  PROGRAMA  PRUEBA  HA  TERMINADO  **********
*
```

```
*
*     **********    DIVISION DE IDENTIFICACION    **********
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID.          PROGRAMA-DE-PRUEBA-1.
 AUTHOR.              LAURA Y ARTURO.
                      ***** * ******
 INSTALLATION.        CECAFI.
 DATE-WRITTEN.        7-SEPTIEMBRE-1983.
 DATE-COMPILED.
 SECURITY.            PRIVADA.
*
*       EL PROGRAMA REALIZA UNA SUMA Y ESCRIBE SUS RESULTADOS.
*       FUE HECHO COMO PRUEBA PARA EL CURSO DE COBOL.
/
*
*       **********    DIVISION DEL MEDIO AMBIENTE    **********
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.     DIGITAL-VAX.
 OBJECT-COMPUTER.     DIGITAL-VAX.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*
*     **********    DECLARACION DE ARCHIVOS I/O    **********
*
 SELECT DATOS-LEC ASSIGN TO 'SYS$INPUT'.
 SELECT IMPRIME ASSIGN TO 'SYS$OUTPUT'.
/
*
*     **********    DIVISION DE DATOS    **********
*
*
 DATA DIVISION.
 FILE SECTION.
 FD DATOS-LEC
     RECORD CONTAINS 80 CHARACTERS
     DATA RECORD IS DATOS-OPERA.
*
* *** DESCRIPCION DEL REGISTRO PARA LOS DATOS DE LA OPERACION ***
*
 01  DATOS-OPERA.
     03  DATO-1                          PIC S9(02)
                 SIGN IS LEADING SEPARATE CHARACTER.
     03  DATO-2                          PIC S9(02)
                 SIGN IS LEADING SEPARATE CHARACTER.
     03  FILLER                          PIC X(74).
*
*     **********    ARCHIVO DE RESULTADOS    **********
*
 FD  IMPRIME
     RECORD CONTAINS 132 CHARACTERS
     DATA RECORD IS LINEA-DE-RESULTADOS.
 01  LINEA-DE-RESULTADOS                 PIC X(132).
/
*
*     **********    REGISTROS INTERNOS AUXILIARES    **********
*
 WORKING-STORAGE SECTION.
 77  SUMA                                PIC -9(02).9(02).
 77  HAY-DATOS                           PIC X(02) VALUE 'SI'.
 77  TIEMPO                              PIC 9(08).
```

```
*
*      ***************  ENCABEZADOS  DE  IMPRESION  ************
*
 01  RESULTADOS.
     03  FILLER                          PIC X(24) VALUE SPACES.
     03  FILLER                          PIC X(12) VALUE "LA SUMA DE
-                                        " ".
     03  DATO1-SAL                       PIC -9(02).9(02).
     03  FILLER                          PIC X(08) VALUE " Y DE ".
     03  DATO2-SAL                       PIC -9(02).9(02).
     03  FILLER                          PIC X(15) VALUE " ES IGUAL
-                                        " A ".
     03  SUMA-SAL                        PIC -(03).9(02).
     03  FILLER                          PIC X(53) VALUE SPACES.
/
*
*      ************  DIVISION  DE  PROCEDIMIENTOS  ************
*
 PROCEDURE DIVISION.
*
*      ***************  PROGRAMA  PRINCIPAL  ***************
*
 COMIENZA.
     PERFORM ABRE-LOS-ARCHIVOS.
     READ DATOS-LEC AT END
          DISPLAY "SE ACABARON LOS DATOS"
     STOP RUN
     END-READ
     PERFORM REALIZA-CALCULOS UNTIL HAY-DATOS = "NO".
     PERFORM CIERRA-LOS-ARCHIVOS.
     STOP RUN.
*    *******  TERMINA EL  PROGRAMA  PRINCIPAL  ********
/
*
*      ***************  ZONA  DE  PROCEDIMIENTOS  ***************
*
 ABRE-LOS-ARCHIVOS.
     OPEN
          INPUT DATOS-LEC
          OUTPUT IMPRIME.
 REALIZA-CALCULOS.
     ACCEPT TIEMPO FROM TIME
     DISPLAY TIEMPO
     COMPUTE SUMA = DATO-1 + DATO-2
     DISPLAY "SUMA = " SUMA
     MOVE DATO-1 TO DATO1-SAL
     MOVE DATO-2 TO DATO2-SAL
     MOVE SUMA TO SUMA-SAL
     WRITE LINEA-DE-RESULTADOS FROM RESULTADOS AFTER 10 LINES
     READ DATOS-LEC AT END
          MOVE "NO" TO HAY-DATOS.
*    ENDREAD.
 CIERRA-LOS-ARCHIVOS.
     CLOSE
          DATOS-LEC
          IMPRIME.
*
*      **********  EL  PROGRAMA  PRUEBA  HA  TERMINADO  *********
*
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     ORGANIZACION-RELATIVA.
AUTHOR.         SERGIO Y ADRIAN.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ENTRADA ASSIGN TO "ENTRADA.DAT".
    SELECT RELATIVO ASSIGN TO "RELATIVO.DAT"
        ORGANIZATION IS RELATIVE.
DATA DIVISION.
FILE SECTION.
FD ENTRADA
    RECORD CONTAINS 40 CHARACTERS
    DATA RECORD IS REG-ENT.
01 REG-ENT.
    02 CARRERA-E                    PIC 9(02).
    03 NO-CUENTA-E                  PIC 9(08).
    03 NOMBRE-E                     PIC X(30).
FD RELATIVO
    RECORD CONTAINS 40 CHARACTERS
    DATA RECORD IS REG-REL
    ACCESS MODE IS RANDOM RELATIVE KEY IS LLAVE.
01 REG-REL.
    03 CARRERA-R                    PIC 9(02).
    03 NO-CUENTA-R                  PIC 9(08).
    03 NOMBRE-R                     PIC X(30).
*
WORKING-STORAGE SECTION.
77  LLAVE                          PIC 9(02) VALUE ZERO.
77  RESP                           PIC X(01).
*
PROCEDURE DIVISION.
INICIO.
    PERFORM ABRE-ARCHIVOS.
    READ ENTRADA AT END
        DISPLAY "ERROR, NO HAY DATOS"
        STOP RUN
    END-READ.
    PERFORM CREA-RELATIVO UNTIL CARRERA-E = 99
    DISPLAY "CON QUE REGISTRO QUIERES TRABAJAR?? (1-10),(99=FIN)"
    ACCEPT LLAVE
    DISPLAY "QUIERES MODIFICARLO?? (S/N)"
    ACCEPT RESP
    PERFORM TRABAJA-RELATIVO UNTIL LLAVE = 99
    PERFORM CIERRA-ARCHIVOS
    STOP RUN.
ABRE-ARCHIVOS.
    OPEN
        INPUT ENTRADA
        I-O   RELATIVO.
CREA-RELATIVO.
    ADD 1 TO LLAVE
    REWRITE REG-REL FROM REG-ENT INVALID KEY
        DISPLAY "ERROR EN LLAVE AL CREAR RELATIVO"
        DISPLAY "REG-REL = " REG-REL
    END-REWRITE.
    READ ENTRADA AT END
        DISPLAY "FIN DATOS DE ENTRADA, SE CREO EL RELATIVO"
    END-READ.
TRABAJA-RELATIVO.
    IF (RESP = "S") THEN
        PERFORM MODIFICA-REGISTRO
    ELSE
```

```
           PERFORM BORRA-REGISTRO
       END-IF
       DISPLAY "CON QUE REGISTRO QUIERES TRABAJAR?? (1-10),(99=FIN)"
       ACCEPT LLAVE
       DISPLAY "QUIERES MODIFICARLO?? (S/N)"
       ACCEPT RESP.
MODIFICA-REGISTRO.
           READ RELATIVO
               INVALID KEY
                   DISPLAY "ERROR AL LEER DEL RELATIVO"
           END-READ
           DISPLAY "REGISTRO A MODIFICAR :"
           DISPLAY REG-REL
           DISPLAY "DAME EL NUEVO REGISTRO :"
           ACCEPT REG-REL
           REWRITE REG-REL INVALID KEY
               DISPLAY "ERROR EN LLAVE AL CREAR RELATIVO".
               DISPLAY "REG-REL = " REG-REL
           END-REWRITE.
BORRA-REGISTRO.
       READ RELATIVO
           INVALID KEY
               DISPLAY "ERROR AL LEER DEL RELATIVO"
       END-READ
       DELETE RELATIVO RECORD
           INVALID KEY
               DISPLAY "ERROR AL BORRAR EL REGISTRO"
       END-DELETE.
CIERRA-ARCHIVOS.
       CLOSE ENTRADA
             RELATIVO.
```

```
*
*    **********    DIVISION  DE  IDENTIFICACION    **********
*
 IDENTIFICATION DIVISION.
 PROGRAM-ID.            PROGRAMA-DE-PRUEBA-1.
 AUTHOR.                LAURA Y ARTURO.
*                       ***** * ******.
 INSTALLATION.          CECAFI.
 DATE-WRITTEN.          7-SEPTIEMBRE-1983.
 DATE-COMPILED.
 SECURITY.              PRIVADA.
*
*       EL PROGRAMA REALIZA UNA SUMA Y ESCRIBE SUS RESULTADOS.
*       FUE HECHO COMO PRUEBA PARA EL CURSO DE COBOL.
/
*
*    **********  DIVISION DEL  MEDIO  AMBIENTE  **********
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER.       DIGITAL-VAX.
 OBJECT-COMPUTER.       DIGITAL-VAX.
 SPECIAL-NAMES.
     CONSOLE IS PANTALLA.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*
*    **********   DECLARACION DE ARCHIVOS I/O   **********
*
 SELECT DATOS-LEC ASSIGN TO 'SYS$INPUT'.
 SELECT IMPRIME ASSIGN TO 'SYS$OUTPUT'.
/
*
*    **********  DIVISION  DE  DATOS  **********
*
*
 DATA DIVISION.
 FILE SECTION.
 FD DATOS-LEC
     DATA RECORD IS
                    DATOS-OPERA.
*
* *** DESCRIPCION DEL REGISTRO PARA LOS DATOS DE LA OPERACION ***
*
 01  DATOS-OPERA.
     03  DATO-1                          PIC 9(02)V9(02).
     03  DATO-2                          PIC 9(02).9(02).
     03  FILLER                          PIC X(70).
*
*    **********   ARCHIVO  DE  RESULTADOS   **********
*
 FD  IMPRIME.
 01  LINEA-DE-RESULTADOS                 PIC X(132).
/
*
*    **********  REGISTROS  INTERNOS  AUXILIARES  **********
*
 WORKING-STORAGE SECTION.
 77  SUMA                                PIC 9(02).9(02).
 77  HAY-DATOS                           PIC X(02) VALUE 'SI'.
 77  TEMP                                PIC 9(02)V9(02).
 77  INDICE                              PIC 9(01) VALUE ZERO.
 77  HOLA                                PIC 9(02).
```

```
*
*      ***************  ENCABEZADOS  DE  IMPRESION  ***********
*
 01   RESULTADOS.
      03  FILLER                          PIC X(24) VALUE SPACES.
      03  FILLER                          PIC X(12) VALUE "LA SUMA DE
-                                         " ".
      03  DATO1-SAL                       PIC Z(09).9(02).
      03  FILLER                          PIC X(08) VALUE " Y DE ".
      03  DATO2-SAL                       PIC Z(09).9(02).
      03  FILLER                          PIC X(15) VALUE " ES IGUAL
-                                         " A  ".
      03  SUMA-SAL                        PIC Z(09).9(02).
      03  FILLER                          PIC X(37) VALUE SPACES.
 01   TABLA.
      03  VALORES OCCURS 6 TIMES ASCENDING KEY IS VAL.
          05 VAL                          PIC 9(01).
/
*
*      ************  DIVISION  DE  PROCEDIMIENTOS  ************
*
 PROCEDURE DIVISION.
*
*      ***************  PROGRAMA  PRINCIPAL  ***************
*
 COMIENZA.
      PERFORM ABRE-LOS-ARCHIVOS.
      PERFORM LEE UNTIL INDICE = 6
      MOVE ZERO TO INDICE
      PERFORM ESC UNTIL INDICE = 6
      READ DATOS-LEC AT END
          DISPLAY "SE ACABARON LOS DATOS"
      STOP RUN.
*     ENDREAD
      PERFORM REALIZA-CALCULOS UNTIL HAY-DATOS = "NO".
      PERFORM CIERRA-LOS-ARCHIVOS.
      STOP RUN.
*       *******  TERMINA  EL  PROGRAMA  PRINCIPAL  ********
/
*
*      ***************  ZONA  DE  PROCEDIMIENTOS  ***********
*
 ABRE-LOS-ARCHIVOS.
      OPEN
          INPUT DATOS-LEC
          OUTPUT IMPRIME.
 LEE.
      ADD 1 TO INDICE
      ACCEPT VAL(HOLA) FROM PANTALLA.
 ESC.
      ADD 1 TO INDICE
      DISPLAY "VAL  = ", VAL(INDICE).
 REALIZA-CALCULOS.
      DISPLAY "DATO-2 = ", DATO-2
      MOVE DATO-2 TO TEMP
      DISPLAY "TEMP = " TEMP
      COMPUTE SUMA = DATO-1 + TEMP
*                 ON SIZE ERROR DISPLAY "TE PASASTE".
*     IF DATO-1 = DATO-2 THEN
*         DISPLAY "SON IGUALES"
*     ELSE
*         DISPLAY "NO SON IGUALES".
*     END-IF.
```

```
      MOVE DATO-1 TO DATO1-SAL
      MOVE DATO-2 TO DATO2-SAL
      MOVE SUMA TO SUMA-SAL
      WRITE LINEA-DE-RESULTADOS FROM RESULTADOS AFTER 10 LINES
      READ DATOS-LEC AT END
          MOVE "NO" TO HAY-DATOS.
*     ENDREAD
 CIERRA-LOS-ARCHIVOS.
      CLOSE
          DATOS-LEC
          IMPRIME.
      PERFORM CIERRA-LOS-ARCHIVOS.
*
*       ********** EL PROGRAMA PRUEBA HA TERMINADO **********
*
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                 DEMOSTRACION-33.
*****************************************************************
*                                                               *
*                       PERFORM                                 *
*                                                               *
*****************************************************************

AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA:
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

*****************************************************************
*
  ENVIRONMENT    DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

*****************************************************************
*
  DATA           DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.

77 NUMERO                   PIC 9(10).
77 NUMERO-CONTROL           PIC 9(10).

*****************************************************************
*
  PROCEDURE DIVISION.
*
*****************************************************************

PRINCIPIO.

*****************************************************************
*                       PERFORM                                 *
*****************************************************************

        DISPLAY "EJECUCION PERFORM 1".
        PERFORM PARRAFO-1
                DISPLAY "TERMINE PERFORM UNO".
        END-PERFORM.

        DISPLAY "EJECUCION PERFORM 2".
        PERFORM PARRAFO-5 THRU PARRAFO-7
                DISPLAY "TERMINE PERFORM DOS".
*       END-PERFORM.
```

```
           DISPLAY "EJECUCION PERFORM 2".
           PERFORM PARRAFO-2 THRU PARRAFO-4 3 TIMES
                   DISPLAY "TERMINE PARRAFO TRES".
      *    END-PERFORM.

           DISPLAY "EJECUCION PERFORM 4".
           MOVE ZEROES TO NUMERO.
           MOVE ZEROES TO NUMERO-CONTROL.
           PERFORM PARRAFO-4 THRU PARRAFO-5
                   WITH TEST BEFORE UNTIL NUMERO = NUMERO-CONTROL
                   DISPLAY "TERMINE PERFORM 4"
      *    END-PERFORM.

           DISPLAY "EJECUCION PERFORM 5".
           MOVE ZEROES TO NUMERO.
           MOVE 4      TO NUMERO-CONTROL.
           PERFORM PARRAFO-4 THRU PARRAFO-5
                   WITH TEST AFTER UNTIL NUMERO = NUMERO-CONTROL
                   DISPLAY "TERMINE PERFORM 5"
      *    END-PERFORM.


           DISPLAY "EJECUCION PERFORM 6".
           MOVE ZEROES TO NUMERO.
           MOVE 4      TO NUMERO-CONTROL.
           PERFORM PARRAFO-5 THRU PARRAFO-7
                   WITH TEST BEFORE
                   VARYING NUMERO FROM 1 BY 1 UNTIL NUMERO = NUMERO-CONTROL
                   DISPLAY "TERMINE PERFORM 6"
      *    END-PERFORM.

           PERFORM FIN-DE-PROGRAMA.


       PARRAFO-1.
           DISPLAY "EJECUCION DEL PARRAFO 1".
           DISPLAY "EJECUCION DEL PERFORM 11".
           PERFORM PARRAFO-11
                   DISPLAY "TERMINE DE REALIZAR PERFORM 11".
      *    END-PERFORM.
           DISPLAY "EJECUCION DEL PERFORM 12".
           PERFORM PARRAFO-11 THROUGH PARRAFO-13
                   DISPLAY "TERMINE DE REALIZAR PERFORM 12".
      *     END-PERFORM.

       PARRAFO-11.
           DISPLAY "EJECUCION DEL PARRAFO 11".
       PARRAFO-12.
           DISPLAY "EJECUCION DEL PARRAFO 12".
       PARRAFO-13.
           DISPLAY "EJECUCION DEL PARRAFO 13".
       PARRAFO-2.
           DISPLAY "EJECUCION DEL PARRAFO 2".
       PARRAFO-3.
           DISPLAY "EJECUCION DEL PARRAFO 3".
       PARRAFO-4.
           DISPLAY "EJECUCION DEL PARRAFO 4".
           ADD 1 TO NUMERO.
       PARRAFO-5.
           DISPLAY "EJECUCION DEL PARRAFO 5".
       PARRAFO-6.
           DISPLAY "EJECUCION DEL PARRAFO 6".
       PARRAFO-7.
```

```
           DISPLAY "EJECUCION DEL PARRAFO 7".
PARRAFO-8.
           DISPLAY "EJECUCION DEL PARRAFO 8".
PARRAFO-9.
           DISPLAY "EJECUCION DEL PARRAFO 9".
PARRAFO-10.
           DISPLAY "EJECUCION DEL PARRAFO 10".

*****************************************************************
FIN-DE-PROGRAMA.
*****************************************************************
           STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                    NOMBRE.
*****************************************************************
*
*              PROGRAMA QUE REDUCE NOMBRES                      *
*
*****************************************************************

AUTHOR.                        JORGE-VALERIO.
                               CD UNIVERSITARIA UNAM
                               MEXICO.
INSTALLATION.                  CECAFI.
                               CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.                  29-SEP-83.
DATE-COMPILED.                 29-SEP-83.
SECURITY.                      PRIVADA.
                               ESTE PROGRAMA ES PARA DEMOSTRACION.
                               PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                               MEDIO.

*****************************************************************
*
   ENVIRONMENT     DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.               VAX11-780.
SOURCE-COMPUTER.               VAX11-780.

*****************************************************************
*
   DATA            DIVISION.
*
*****************************************************************

WORKING-STORAGE SECTION.
77    LINEA-WS                 PIC    X(132)    VALUE  SPACES.
77    CONT1-WS                 PIC    9(03)     VALUE  ZEROES.
77    CONT2-WS                 PIC    9(03)     VALUE  ZEROES.
77    CONT3-WS                 PIC    9(03)     VALUE  ZEROES.
77    CONT4-WS                 PIC    9(03)     VALUE  ZEROES.
77    PIDE-CONT-WS             PIC    X(01)     VALUE    "S".
01    NOMBRE-WS.
      02 INICIAL-NOMBRE-WS     PIC    X(01)     VALUE  SPACES.
      02 APELLIDO-NOMBRE-WS    PIC    X(20)     VALUE  SPACES.

*****************************************************************
*
   PROCEDURE DIVISION.
*
*****************************************************************

EJECUTA.
    PERFORM PIDE-NOMBRE
            THRU PIDE-CONTINUACION
            TEST BEFORE UNTIL PIDE-CONT-WS EQUAL "N"
*   END-PERFORM
    PERFORM FIN-DE-PROGRAMA.

*****************************************************************
*                  PIDE NOMBRE COMPLETO                         *
```

```
* CONVENCIONES                                                          *
*              LOS NOMBRES Y LOS APELLIDOS SE SEPARAN POR COMAS          *
*************************************************************************

PIDE-NOMBRE.

    DISPLAY " "
    DISPLAY " "
    DISPLAY "CECAFI> REDUCCION DE NOMBRES "
    DISPLAY "        - (APLICACION DE LA INSTRUCCION INSPECT Y SUBSTRINGS)"
    DISPLAY " "
    DISPLAY " "
    DISPLAY "EL NOMBRE TENDRA LAS SIGUIENTES CARACTERISTICAS : "
    DISPLAY " "
    DISPLAY "<<<<< SEPARADOS NOMBRES Y APELLIDOS POR COMAS "
            "Y TERMINADO CON PUNTO  >>>>>"
    DISPLAY " "
    DISPLAY " "
    DISPLAY "INPUT> DAME EL NOMBRE : " WITH NO ADVANCING.
    ACCEPT LINEA-WS.

*************************************************************************
*                        CHECA CONDICIONES                              *
*************************************************************************

CHECA-CONDICIONES.

    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ",".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTAN COMAS DE SEPARACION >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ".".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTA PUNTO                    >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    IF CONT1-WS NOT EQUAL 1 THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< HAY MAS DE UN PUNTO          >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ".".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS) TALLYING CONT3-WS FOR LEADING " ".
    IF CONT3-WS NOT EQUAL CONT2-WS THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< EL PUNTO NO ESTA AL FINAL >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
```

```
*****************************************************************
*                      QUITA BLANCOS                            *
*****************************************************************

QUITA-BLANCOS.

    PERFORM QUITA-HASTA-EL-PUNTO
            WITH TEST BEFORE UNTIL PIDE-CONT-WS IS EQUAL TO ".".
*   END-PERFORM.

*****************************************************************
*                 QUITA BLANCOS HASTA EL PUNTO                  *
*****************************************************************

QUITA-HASTA-EL-PUNTO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE " ".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 1 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS)
            TALLYING CONT3-WS FOR LEADING " ".
    ADD CONT1-WS TO CONT3-WS.
    MOVE CONT2-WS TO CONT4-WS.
    SUBTRACT CONT3-WS FROM CONT2-WS.
    MOVE LINEA-WS(CONT3-WS:CONT2-WS) TO LINEA-WS(CONT1-WS:CONT4-WS).
    ADD -1 TO CONT1-WS.
    MOVE LINEA-WS(CONT1-WS:1) TO PIDE-CONT-WS.

*****************************************************************
*                       INICIAL                                 *
*****************************************************************

PONE-INICIAL.

    MOVE LINEA-WS(1:1) TO INICIAL-NOMBRE-WS.

*****************************************************************
*                     PRIMER APELLIDO                           *
*****************************************************************

PONE-APELLIDO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ",".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS BEFORE " ".
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    ADD 1 TO CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS)
            TALLYING CONT3-WS FOR CHARACTERS BEFORE ",".
    IF CONT3-WS IS ZERO THEN
       MOVE LINEA-WS(CONT1-WS:CONT2-WS) TO APELLIDO-NOMBRE-WS
    ELSE
       MOVE LINEA-WS(CONT1-WS:CONT3-WS) TO APELLIDO-NOMBRE-WS
```

```
    END-IF.

*****************************************************************
*                    DESPLIEGA LA INFORMACION DEL REGISTRO
*****************************************************************

 DESPLIEGA-INFORMACION.

     DISPLAY " ".
     DISPLAY "OUTPUT> CONTENIDO DEL REGISTRO DE NOMBRE      ".
     DISPLAY " ".
     DISPLAY " ".
     DISPLAY " INICIAL   : " INICIAL-NOMBRE-WS.
     DISPLAY " APELLIDO  : " APELLIDO-NOMBRE-WS.
     DISPLAY " ".
     DISPLAY " ".


*****************************************************************
*                    PIDE CONTINUACION
*****************************************************************

 PIDE-CONTINUACION.

     PERFORM PIDE-SI-O-NO
             WITH TEST BEFORE UNTIL
             NOT    (PIDE-CONT-WS IS NOT EQUAL "S"
                 AND PIDE-CONT-WS IS NOT EQUAL "N").
*    END-PERFORM

*****************************************************************
                     PIDE-SI-O-NO
*****************************************************************

 PIDE-SI-O-NO.

     MOVE SPACES TO PIDE-CONT-WS.
     DISPLAY " ".
     DISPLAY " QUIERES CONTINUAR EL PROGRAMA (SI/NO) : " WITH NO ADVANCING.
     ACCEPT PIDE-CONT-WS.

*****************************************************************
*                    FIN DEL PROGRAMA
*****************************************************************

 FIN-DE-PROGRAMA.

     STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. FORMATO-LIBRE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NUM PIC X(09).
01 NUM-BIS OCCURS 9 TIMES PIC X(01).
77 SIGNOS PIC 9(02) VALUE ZEROES.
77 NEG PIC 9(02) VALUE ZEROES.
77 PUNTOS PIC 9(02) VALUE ZEROES.
77 POS PIC 9(02) VALUE ZEROES.
77 CUENTA-ENTEROS PIC 9(02) VALUE ZEROES.
77 CUENTA-DECIMALES PIC 9(02) VALUE ZEROES.
77 IMPRESION  PIC -(10).9(09).
77 I PIC 9(02) VALUE ZEROES.
01  CAMPO.
    03  A PIC X(09).
    03  AA PIC S9(09)V9(09) VALUE ZEROES.
    03  AAA REDEFINES AA.
        05  INTEG PIC S9(09).
        05  ENTERO  REDEFINES INTEG.
            06 ENTERO-BIS OCCURS 9 TIMES PIC X.
        05  DECS PIC 9(09).
        05  DECIMOS REDEFINES DECS PIC X(09).
    03  BBB  PIC S9(09)V9(09).
PROCEDURE DIVISION.
INICIO.
    ACCEPT NUM.
        INSPECT NUM TALLYING PUNTOS FOR ALL "."
        IF PUNTOS LESS 2 THEN
            INSPECT NUM TALLYING NEG FOR ALL "-",POS FOR ALL "+"
            ADD NEG POS GIVING SIGNOS
            IF SIGNOS LESS 2 THEN
                INSPECT NUM REPLACING ALL "+" BY "0" ALL "-"
                BY "0"  ALL " " BY "0"
                MOVE ZEROES TO CONT
                MOVE ZEROES TO CUNETA-DECIMALES
                    INSPECT NUM TALLYING CUENTA-ENTEROS FOR CHARACTERS BEFORE
                INITIAL "."
                IF CUENTA-ENTEROS < 09 THEN
                    PERFORM ASIGNA FROM I EQUAL 1 BY 1 UNTIL CUENTA-ENTEROS
                    ADD 2 TO CONT
                    SUBTRACT CUENTA-ENTEROS FROM 09 GIVING CONT1
                    MOVE NUM(CONT:CONT1) TO DECIMOS
                    MOVE AAA TO BBB
                ELSE
                    MOVE NUM TO BBB
                END-IF
                INSPECT BBB REPLACING ALL " " BY "0"
                IF NEG = 1 THEN
                    MULTIPLY -1 BY BBB
                NOELSE
                END-IF
                MOVE BBB TO IMPRESION
                DISPLAY " RES ", IMPRESION
                STOP RUN
            ELSE
                DISPLAY "ERROR: MAS DE UN SIGNO TECLEADO"
            END-IF
        ELSE
            DISPLAY "ERROR: MAS DE UN PUNTO TECLEADO"
        END-IF.
ASIGNA.
```

```
MOVE NUM-BIS(I) TO ENTERO-BIS(I)
MOVE
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EJEMPLO-SUMA.
AUTHOR. YO
INSTALLATION. VAX
DATE-WRITTEN. AYER
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 AA PIC X(05).
77 CONT PIC 9(02).
77 CONT1 PIC 9(02).
77  FF PIC Z(05).9(02).
01  CAMPO.
    03  A PIC X(05).
    03  AAA PIC 9(05)V9(05) VALUE ZEROES.
    03  AAA-1 REDEFINES AAA.
        05  INTEG PIC 9(05).
        05  ENTERO  REDEFINES INTEG PIC X(05)  JUST RIGHT.
        05  DECS PIC 9(05).
        05  DECIMOS REDEFINES DECS PIC X(05).
    03  BBB  PIC 9(05)V9(05).
PROCEDURE DIVISION.
INICIO.
        ACCEPT A
        INSPECT A REPLACING ALL "." BY "0"
        INSPECT A TALLYING CONT FOR CHARACTERS BEFORE INITIAL "."
        IF CONT < 5 THEN
        MOVE A(1:CONT) TO ENTERO
        ADD 2 TO CONT
        SUBTRACT CONT FROM 05 GIVING CONT1
        MOVE A(CONT:CONT1) TO DECIMOS
        MOVE AAA-1 TO BBB
        ELSE
            MOVE A TO BBB
        END-IF
        INSPECT BBB REPLACING ALL " " BY "0"
        ADD 1 TO BBB
        MOVE BBB TO FF
        DISPLAY " RES ", FF
        STOP RUN.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.                 DEMOSTRACION-21.
AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PUBLICA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION Y
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01    REGISTRO-1 PIC X(80).
77    CONT PIC 9(03) VALUE ZEROES.

PROCEDURE DIVISION.

PRINCIPIO.

          PERFORM INICIO.
          PERFORM INSPECT-TOTAL.
          PERFORM INSPECT-AFTER.
          PERFORM INSPECT-BEFORE.
          PERFORM REPLACING-TOTAL.
          PERFORM REPLACING-AFTER.
          PERFORM REPLACING-BEFORE.
          PERFORM CONVERTING-TOTAL.
          STOP RUN.

INICIO.

          PERFORM LIMPIA 24 TIMES.
          PERFORM LETRERO.
          PERFORM LIMPIA 05 TIMES.

LIMPIA.

          DISPLAY " ".

LETRERO.

          DISPLAY "CECAFI> Demostracion de la instruccion INSPECT ".

INSPECT-TOTAL.

          DISPLAY "Introduce un nombre en el cual se contaran todas las J."
          DISPLAY "INPUT> Cont'all J nombre ---> " WITH NO ADVANCING.
          ACCEPT REGISTRO-1.
          MOVE ZEROES TO CONT.
          INSPECT REGISTRO-1 TALLYING CONT FOR ALL "J".
          DISPLAY "OUTPUT> CONT All J : " CONT.
          MOVE ZEROES TO CONT.
          PERFORM LIMPIA 2 TIMES.
```

```
IDENTIFICATION DIVISION.

PROGRAM-ID.             DEMOSTRACION-21.
AUTHOR.                 JORGE-VALERIO,
                        CD UNIVERSITARIA UNAM
                        MEXICO.
INSTALLATION.           CECAFI.
                        CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.           29-SEP-83.
DATE-COMPILED.          29-SEP-83.
SECURITY.               PUBLICA.
                        ESTE PROGRAMA ES PARA DEMOSTRACION.
                        PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                        MEDIO.

ENVIRONMENT    DIVISION.

CONFIGURATION  SECTION.
OBJECT-COMPUTER.        VAX11-780.
SOURCE-COMPUTER.        VAX11-780.

DATA           DIVISION.
WORKING-STORAGE SECTION.
01   REGISTRO-1 PIC X(80).
77   CONT PIC 9(03) VALUE ZEROES.

PROCEDURE DIVISION.

PRINCIPIO.

        PERFORM INICIO.
        PERFORM INSPECT-TOTAL.
        PERFORM INSPECT-AFTER.
        PERFORM INSPECT-BEFORE.
        PERFORM REPLACING-TOTAL.
        PERFORM REPLACING-AFTER.
        PERFORM REPLACING-BEFORE.
        PERFORM CONVERTING-TOTAL.
        STOP RUN.
INICIO.

        PERFORM LIMPIA 24 TIMES.
        PERFORM LETRERO.
        PERFORM LIMPIA 05 TIMES.

LIMPIA.

        DISPLAY ' '.

LETRERO.

        DISPLAY '*CECAFI> Demostracion de la instruccion INSPECT .'.

INSPECT-TOTAL.

        DISPLAY "Introduce un nombre en el cual se contaran todas las J."
        DISPLAY "INPUT> Cant. all J nombre ---> " WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        MOVE ZEROES TO CONT.
        INSPECT REGISTRO-1 TALLYING CONT FOR ALL 'J'.
        DISPLAY 'OUTPUT> CONT ALL J : ' CONT.
        MOVE ZEROES TO CONT.
        PERFORM LIMPIA 2 TIMES.
```

```
           DISPLAY 'Introduce un nombre en el cual se contaran todas las J que sean lideres.'.
           DISPLAY 'INPUT> Cont leading J nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING 'J'.
           DISPLAY 'OUTPUT> CONT Leading J : ' CONT.
           MOVE ZEROES TO CONT.
           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se contaran todos los caracteres del nombre capturado.'.
           DISPLAY 'INPUT> Cont characters nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 TALLYING CONT FOR CHARACTERS.
           DISPLAY 'OUTPUT> CONT characters : ' CONT.

INSPECT-AFTER.

           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se contaran todas las J despues de una @.'.
           DISPLAY 'INPUT> Cont all J after @ nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL 'J' AFTER INITIAL '@'.
           DISPLAY 'OUTPUT> CONT all J after @ : ' CONT.
           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se contaran todas las J lideres despues de una @.'.
           DISPLAY 'INPUT> Cont leading J after @ nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING 'J' AFTER INITIAL '@'.
           DISPLAY 'OUTPUT> CONT leading J after @ : ' CONT.

INSPECT-BEFORE.

           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se contaran todas las J antes de una @.'.
           DISPLAY 'INPUT> Cont all J before @ nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR ALL 'J' BEFORE INITIAL '@'.
           DISPLAY 'OUTPUT> CONT All J before @ : ' CONT.
           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se contaran todas las J lideres antes de una @.'.
           DISPLAY 'INPUT> Cont leading J before @ nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           MOVE ZEROES TO CONT.
           INSPECT REGISTRO-1 TALLYING CONT FOR LEADING 'J' BEFORE INITIAL '@'.
           DISPLAY 'OUTPUT> CONT leading J before @ : ' CONT.

REPLACING-TOTAL.

           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se reemplazara por As.'.
           DISPLAY 'INPUT> Replacing characters nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING CHARACTERS BY 'A'.
           DISPLAY 'OUTPUT> Repalcing characters by As : ' REGISTRO-1.
           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se reemplazara toda J por N.'.
           DISPLAY 'INPUT> Replacing all J by N nombre ---> ' WITH NO ADVANCING.
           ACCEPT REGISTRO-1.
           INSPECT REGISTRO-1 REPLACING ALL 'J' BY 'N'.
           DISPLAY 'OUTPUT> Replacing All J by N : ' REGISTRO-1.
           PERFORM LIMPIA 2 TIMES.
           DISPLAY 'Introduce un nombre en el cual se reemplazara toda X por Y que sean lideres.'.
```

```
        DISPLAY 'INPUT> Replacing Leading X by Y Nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING LEADING 'X' BY 'Y'.
        DISPLAY 'OUTPUT> Replacing Leading X by Y : ' REGISTRO-1.
        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara la primera A por B.'.
        DISPLAY 'INPUT> Replacing first A by B nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING FIRST 'A' BY 'B'.
        DISPLAY 'OUTPUT> Replacing first A by B : ' REGISTRO-1.

REPLACING-BEFORE.

        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda J por N antes de una @.'.
        DISPLAY 'INPUT> Replacing all J by N before @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING ALL 'J' BY 'N' BEFORE '@'.
        DISPLAY 'OUTPUT> Replacing All J by N before @ : ' REGISTRO-1.
        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda X por Y antes de una @ que sean lide
        DISPLAY 'INPUT> Replacing leading X by Y before @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING LEADING 'X' BY 'Y' BEFORE '@'.
        DISPLAY 'OUTPUT> Replacing Leading X by Y before @ : ' REGISTRO-1.
        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda A por B antes de una @ o es la prim
        DISPLAY 'INPUT> Replacing first A by B before @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING FIRST 'A' BY 'B' BEFORE '@'.
        DISPLAY 'OUTPUT> Replacing first A by B before @ : ' REGISTRO-1.

REPLACING-AFTER.

        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda J por N despues de una @.'.
        DISPLAY 'INPUT> Replacing all J by N after @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING ALL 'J' BY 'N' AFTER '@'.
        DISPLAY 'OUTPUT> Replacing All J by N after @ : ' REGISTRO-1.
        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda X por Y despues de una @ que sean li
        DISPLAY 'INPUT> Replacing leading X by Y after @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING LEADING 'X' BY 'Y' AFTER '@'.
        DISPLAY 'OUTPUT> Replacing Leading X by Y after @ : ' REGISTRO-1.
        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se reemplazara toda A por B despues de una @ que sea la
        DISPLAY 'INPUT> Replacing first A by B after @ nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 REPLACING FIRST 'A' BY 'B' AFTER '@'.
        DISPLAY 'OUTPUT> Replacing first A by B after @ : ' REGISTRO-1.

CONVERTING-TOTAL.

        PERFORM LIMPIA 2 TIMES.
        DISPLAY 'Introduce un nombre en el cual se hace una conversion de J por A , L por B y M
        DISPLAY 'INPUT> Converting JLM to ABC nombre ---> ' WITH NO ADVANCING.
        ACCEPT REGISTRO-1.
        INSPECT REGISTRO-1 CONVERTING 'JLM' TO 'ABC'.
        DISPLAY 'OUTPUT> Converting JLM to ABC : ' REGISTRO-1.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                 DEMOSTRACION-35.

*****************************************************************
*                                                               *
*                        NOMINITA                               *
*                                                               *
*****************************************************************

AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

*****************************************************************
*
ENVIRONMENT    DIVISION.
*
*****************************************************************

CONFIGURATION  SECTION.

OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

    SELECT DATOS-ENTRADA ASSIGN TO "COB35.ENT".
    SELECT DATOS-SALIDA  ASSIGN TO "COB35.SAL".

*****************************************************************
*
DATA           DIVISION.
*
*****************************************************************

*****************************************************************
*
FILE SECTION.
*
*****************************************************************

*****************************************************************
*                                                               *
*                ARCHIVO   DE   DATOS-ENTRADA                   *
*                                                               *
*****************************************************************

FD  DATOS-ENTRADA

    RECORD CONTAINS 39 CHARACTERS
    DATA RECORD IS DATOS-FD.
```

```
    03    NOMBRE-DATOS-FD                    PIC X(30).
    03    SUELDO-DIARIO-DATOS-FD             PIC 9(04)V9(02).
    03    DIAS-TRABAJADOS-DATOS-FD           PIC 9(02).
    03    SEXO-DATOS-FD                      PIC X(01).

*****************************************************************
*                                                               *
*              ARCHIVO   DE   DATOS-SALIDA                      *
*                                                               *
*****************************************************************

FD  DATOS-SALIDA
    RECORD CONTAINS 132 CHARACTERS
    DATA RECORD IS LINEA-FD.

01  LINEA-FD.                             PIC X(132).

*****************************************************************
*
WORKING-STORAGE SECTION.
*
*****************************************************************

77   HAY-MAS-DATOS-WS                PIC X(02)          VALUE "SI".
77   SUELDO-MENSUAL-WS               PIC 9(04)V9(02).
77   SOBRE-SUELDO-WS                 PIC 9(06)V9(02).
77   NUMERO-LINEAS-WS                PIC 9(04)          VALUE 60.

01   TITULO-WS.
     03 FILLER                       PIC X(15)          VALUE SPACES.
     03 FILLER                       PIC X(06)          VALUE "NOMBRE".
     03 FILLER                       PIC X(59)          VALUE SPACES.
     03 FILLER                       PIC X(14)          VALUE "SUELDO MENSUAL".
     03 FILLER                       PIC X(68)          VALUE SPACES.

01   DETALLE-WS.
     03 FILLER                       PIC X(10)          VALUE SPACES.
     03 NOMBRE-DETALLE-WS            PIC X(30).
     03 FILLER                       PIC X(10)          VALUE SPACES.
     03 SUELDO-MENSUAL-DETALLE-WS    PIC $ZZZZZ9.99.
     03 FILLER                       PIC X(72)          VALUE SPACES.

*****************************************************************
*
PROCEDURE DIVISION.
*
*****************************************************************

INICIO.

    PERFORM ABRIR-ARCHIVOS.
    PERFORM PROCESO.
    PERFORM CERRAR-ARCHIVOS.
    PERFORM FIN-DE-PROGAMA.

*****************************************************************
*                                                               *
*              ABRIR   ARCHIVOS                                 *
*                                                               *
*****************************************************************

ABRIR-ARCHIVOS.
```

```
    OPEN    INPUT     DATOS-ENTRADA
            OUTPUT    DATOS-SALIDA.

*****************************************************************************

                        PROCESO

*
*****************************************************************************

PROCESO.

    PERFORM LECTURA
            WITH TEST BEFORE UNTIL NOT HAY-MAS-DATOS-WS IS EQUAL "SI".

*****************************************************************************
*
*                       LECTURA
*
*****************************************************************************

LECTURA.

    READ DATOS-ENTRADA
        AT END MOVE "NO" TO HAY-MAS-DATOS-WS.
    IF HAY-MAS-DATOS-WS EQUAL "SI" THEN
       PERFORM CALCULOS
       PERFORM ESCRITURA
    ELSE
       NEXT SENTENCE.
    ENDIF

*****************************************************************************
*
*                       CALCULOS
*
*****************************************************************************

CALCULOS.

    MULTIPLY        SUELDO-DIARIO-DATOS-FD
                BY  DIAS-TRABAJADOS-DATOS-FD
                GIVING  SUELDO-MENSUAL-WS.
    IF SEXO-DATOS-FD = "M" THEN
       PERFORM CINCO-PORCIENTO
    ELSE
       PERFORM DIEZ-PORCIENTO
    END-IF.

*****************************************************************************
*
*                       CINCO PORCIENTO
*
*****************************************************************************

CINCO-PORCIENTO.

    MULTIPLY        0.05
                BY  SUELDO-MENSUAL-WS
                GIVING  SOBRE-SUELDO-WS.
    ADD SOBRE-SUELDO-WS TO SUELDO-MENSUAL-WS.
```

```
*****************************************************************
*
*                      DIEZ PORCIENTO
*
*****************************************************************

DIEZ-PORCIENTO.

    MULTIPLY        0.10
          BY        SUELDO-MENSUAL-WS
       GIVING       SOBRE-SUELDO-WS.
    ADD SOBRE-SUELDO-WS TO SUELDO-MENSUAL-WS.

*****************************************************************
*                                                               *
*                        ESCRITURA                              *
*                                                               *
*****************************************************************

ESCRITURA.

    MOVE NOMBRE-DATOS-FD TO NOMBRE-DETALLE-WS.
    MOVE SUELDO-MENSUAL-WS TO SUELDO-MENSUAL-DETALLE-WS.
    IF NUMERO-LINEAS-WS > 57 THEN
       MOVE 4 TO NUMERO-LINEAS-WS
       WRITE LINEA-FD FROM TITULO-WS AFTER PAGE
       WRITE LINEA-FD FROM DETALLE-WS AFTER 2
    ELSE
       ADD 2 TO NUMERO-LINEAS-WS
       WRITE LINEA-FD FROM DETALLE-WS AFTER 2
    END-IF.

*****************************************************************
*                                                               *
*                     CERRAR-ARCHIVOS                           *
*                                                               *
*****************************************************************

CERRAR-ARCHIVOS.

    CLOSE   DATOS-ENTRADA
            DATOS-SALIDA.

*****************************************************************
*                                                               *
*                     FIN DE PROGRAMA                           *
*                                                               *
*****************************************************************

FIN-DE-PROGAMA.

    STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                 DEMOSTRACION-34.
*****************************************************************
*                                                               *
*          PROGRAMA QUE REDUCE NOMBRES                          *
*                                                               *
*****************************************************************

AUTHOR.                     JORGE-VALERIO.
                            CD UNIVERSITARIA UNAM
                            MEXICO.
INSTALLATION.               CECAFI.
                            CENTRO DE CALCULO DE LA FACULTAD DE ING.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
SECURITY.                   PRIVADA.
                            ESTE PROGRAMA ES PARA DEMOSTRACION.
                            PUEDE COPIARSE O REPRODUCIRSE POR CUALQUIER
                            MEDIO.

*****************************************************************
*                                                               *
   ENVIRONMENT    DIVISION.
*                                                               *
*****************************************************************

CONFIGURATION  SECTION.
OBJECT-COMPUTER.            VAX11-780.
SOURCE-COMPUTER.            VAX11-780.

*****************************************************************
*                                                               *
   DATA           DIVISION.
*                                                               *
*****************************************************************

WORKING-STORAGE SECTION.
77    LINEA-WS                    PIC    X(132)    VALUE    SPACES.
77    CONT1-WS                    PIC    9(03)     VALUE    ZEROES.
77    CONT2-WS                    PIC    9(03)     VALUE    ZEROES.
77    CONT3-WS                    PIC    9(03)     VALUE    ZEROES.
77    CONT4-WS                    PIC    9(03)     VALUE    ZEROES.
77    FIDE-CONT-WS                PIC    X(01)     VALUE      "S".
01    NOMBRE-WS.
      02 INICIAL-NOMBRE-WS        PIC    X(01)     VALUE    SPACES.
      02 APELLIDO-NOMBRE-WS       PIC    X(20)     VALUE    SPACES.

*****************************************************************
*                                                               *
   PROCEDURE DIVISION.
*                                                               *
*****************************************************************

EJECUTA.

    PERFORM PIDE-NOMBRE
            THRU PIDE-CONTINUACION
            TEST BEFORE UNTIL FIDE-CONT-WS EQUAL "N"
*   END-PERFORM
    PERFORM FIN-DE-PROGRAMA.

*****************************************************************
*                    PIDE NOMBRE COMPLETO                       *
```

```
*CONVENCION:                                                            *
*          LOS NOMBRES Y LOS APELLIDOS SE SEPARAN POR COMAS              *
*************************************************************************

PIDE-NOMBRE.

    DISPLAY " "
    DISPLAY " "
    DISPLAY "CECAFI> REDUCCION DE NOMBRES ".
    DISPLAY "        (APLICACION DE LA INSTRUCCION INSPECT Y SUBSTRINGS)"
    DISPLAY " "
    DISPLAY " "
    DISPLAY "EL NOMBRE TENDRA LAS SIGUIENTES CARACTERISTICAS : "
    DISPLAY " "
    DISPLAY "<<<<< SEPARADOS NOMBRES Y APELLIDOS POR COMAS "
            "Y TERMINADO CON PUNTO >>>>>"
    DISPLAY " "
    DISPLAY " "
    DISPLAY "INPUT> DAME EL NOMBRE : " WITH NO ADVANCING.
    ACCEPT LINEA-WS.

*************************************************************************
*                     CHECA CONDICIONES                                 *
*************************************************************************

CHECA-CONDICIONES.

    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ",".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTAN COMAS DE SEPARACION >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR ALL ".".
    IF CONT1-WS IS ZERO THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< FALTA PUNTO                    >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    IF CONT1-WS NOT EQUAL 1 THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< HAY MAS DE UN PUNTO            >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ".".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS) TALLYING CONT3-WS FOR LEADING " ".
    IF CONT3-WS NOT EQUAL CONT2-WS THEN
       DISPLAY " "
       DISPLAY "ERROR : <<< EL PUNTO NO ESTA AL FINAL      >>>"
       DISPLAY " "
       PERFORM EJECUTA
    END-IF.
```

```
*****************************************************************
*                      QUITA BLANCOS
*****************************************************************

QUITA-BLANCOS.

    PERFORM QUITA-HASTA-EL-PUNTO
            WITH TEST BEFORE UNTIL FIDE-CONT-WS IS EQUAL TO "."
*   END-PERFORM.

*****************************************************************
*                 QUITA BLANCOS HASTA EL PUNTO                  *
*****************************************************************

QUITA-HASTA-EL-PUNTO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE " ".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS.
    ADD 1 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS)
            TALLYING CONT3-WS FOR LEADING " ".
    ADD CONT1-WS TO CONT3-WS.
    MOVE CONT2-WS TO CONT4-WS.
    SUBTRACT CONT3-WS FROM CONT2-WS.
    MOVE LINEA-WS(CONT3-WS:CONT2-WS) TO LINEA-WS(CONT1-WS:CONT4-WS).
    ADD -1 TO CONT1-WS.
    MOVE LINEA-WS(CONT1-WS:1) TO FIDE-CONT-WS.

*****************************************************************
*                       INICIAL                                 *
*****************************************************************

FONE-INICIAL.

    MOVE LINEA-WS(1:1) TO INICIAL-NOMBRE-WS.

*****************************************************************
*                       PRIMER APELLIDO                         *
*****************************************************************

FONE-APELLIDO.

    MOVE ZEROES TO CONT1-WS.
    MOVE ZEROES TO CONT2-WS.
    MOVE ZEROES TO CONT3-WS.
    MOVE ZEROES TO CONT4-WS.
    INSPECT LINEA-WS TALLYING CONT1-WS FOR CHARACTERS BEFORE ",".
    INSPECT LINEA-WS TALLYING CONT2-WS FOR CHARACTERS BEFORE ".".
    ADD 2 TO CONT1-WS.
    SUBTRACT CONT1-WS FROM CONT2-WS.
    ADD 1 TO CONT2-WS.
    INSPECT LINEA-WS(CONT1-WS:CONT2-WS)
            TALLYING CONT3-WS FOR CHARACTERS BEFORE " ".
    IF CONT3-WS IS ZERO THEN
       MOVE LINEA-WS(CONT1-WS:CONT2-WS) TO APELLIDO-NOMBRE-WS
    ELSE
       MOVE LINEA-WS(CONT1-WS:CONT3-WS) TO APELLIDO-NOMBRE-WS.
```

```
    END-IF.

*****************************************************************
*                  DESPLIEGA LA INFORMACION DEL REGISTRO        *
*****************************************************************

DESPLIEGA-INFORMACION.

     DISPLAY " ".
     DISPLAY "OUTPUT> CONTENIDO DEL REGISTRO DE NOMBRE      ".
     DISPLAY " ".
     DISPLAY " ".
     DISPLAY " INICIAL  : " INICIAL-NOMBRE-WS.
     DISPLAY " APELLIDO : " APELLIDO-NOMBRE-WS.
     DISPLAY " ".
     DISPLAY " ".


*****************************************************************
*                     PIDE CONTINUACION                         *
*****************************************************************

PIDE-CONTINUACION.

    PERFORM PIDE-SI-O-NO
            WITH TEST BEFORE UNTIL
            NOT    (PIDE-CONT-WS IS NOT EQUAL "S"
                AND PIDE-CONT-WS IS NOT EQUAL "N").
*   END-PERFORM

*****************************************************************
*                     PIDE-SI-O-NO                              *
*****************************************************************

PIDE-SI-O-NO.

    MOVE SPACES TO PIDE-CONT-WS
    DISPLAY " ".
    DISPLAY "QUIERES CONTINUAR EL PROGRAMA (SI/NO) : " WITH NO ADVANCING.
    ACCEPT PIDE-CONT-WS.

*****************************************************************
*                     FIN DEL PROGRAMA                          *
*****************************************************************

FIN-DE-PROGRAMA.

    STOP RUN.
```

```
IDENTIFICATION DIVISION.
*************************************************************************
PROGRAM-ID.     TABLAS.
AUTHOR.         JORGE VALERIO.
 VSTALLATION.   CECAFI.
-ATE-WRITTEN.   25-OCT-84.
DATE-COMPILED.
SECURITY.

*               REVISADO 26 DE ENERO DE 1985.


ENVIRONMENT DIVISION.
*************************************************************************

CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

    SELECT DEPENDENCIA ASSIGN 'DBA1:[VALERIO.COBOL.DATOS]TABDEPEN.DAT'.
    SELECT CLAVES      ASSIGN 'DBA1:[VALERIO.COBOL.DATOS]CLAVES.DAT'.
    SELECT REPORTE     ASSIGN 'DBA1:[VALERIO.COBOL.REPORTES]REPCLAVES.LIS'.

DATA DIVISION.
*************************************************************************

 ILE SECTION.

FD   DEPENDENCIA
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-DEPENDENCIA.

01 RECORD-DEPENDENCIA.
   03 CLAVE-REC-DEP  PIC X(03).
   03 FILLER         PIC X(02).
   03 NOMBRE-REC-DEP PIC X(50).
   03 FILLER         PIC X(25).

FD   CLAVES
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-CLAVE.

01 RECORD-CLAVE.
   03 DEP-REC-CVE    PIC X(03).
   03 CARR-REC-CVE   PIC 9(02).
   03 FILLER         PIC X(75).


FD   REPORTE
     RECORD CONTAINS 132 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-REPORTE.

01   RECORD-REPORTE PIC X(132).
```

```
77 HAY-DATOS-TAB     PIC X(01) VALUE "S".
   88 NO-HAY-DATOS-TAB VALUE "N".
77 HAY-DATOS-CLAVE  PIC X(01) VALUE "S".
   88 NO-HAY-DATOS-CLAVE VALUE "N".

01 CARRERAS.
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 CIVIL          PIC X(60) VALUE "21   INGENIERO CIVIL
   03 MINAS          PIC X(60) VALUE "22   INGENIERO DE MINAS Y METALURGISTA
   03 GEOLOGO        PIC X(60) VALUE "23   INGENIERO GEOLOGO
   03 PETROLERO      PIC X(60) VALUE "24   INGENIERO PETROLERO
   03 TOPOGRAFO      PIC X(60) VALUE "25   INGENIERO TOPOGRAFO Y GEODESTA
   03 ELECTRICISTA   PIC X(60) VALUE "26   INGENIERO MECANICO ELECTRICISTA
   03 MECANICA       PIC X(60) VALUE "27   INGENIERO MEC.ELEC.(AREA MECANICA)
   03 INDUSTRIAL     PIC X(60) VALUE "28   INGENIERO MEC.ELEC.(AREA INDUSTRIAL)
   03 ELECTRONICO    PIC X(60) VALUE "29   INGENIERO MEC.ELEC.(AREA SIST.ELEC.Y EL
   03 FILLER         PIC X(60) VALUE "******************** ERROR EN CARRERA *****
   03 GEOFISICO      PIC X(60) VALUE "31   INGENIERO GEOFISICO
   03 COMPUTACION    PIC X(60) VALUE "32   INGENIERO EN COMPUTACION

01 CARRERAS-DEF REDEFINES CARRERAS.
   03 TABLA-CARRERAS OCCURS 32 TIMES PIC X(60).

01 DEPENDENCIAS-DEF.
   03 TABLA-DEPENDENCIAS OCCURS 300 TIMES
                         ASCENDING KEY IS CLAVE-TAB-DEP
                         INDEXED BY INDICE-DEPENDENCIA.
      05 CLAVE-TAB-DEP  PIC X(03).
      05 FILLER         PIC X(02).
      05 NOMBRE-TAB-DEP PIC X(50).
      05 FILLER         PIC X(25).

01 SALIDA.
   03 FILLER    PIC X(11) VALUE "CARRERA    ".
   03 CARR-SAL  PIC X(55) VALUE SPACES.
   03 FILLER    PIC X(13) VALUE "DEPENDENCIA  ".
   03 DEPEN-SAL PIC X(50) VALUE SPACES.
   03 FILLER    PIC X(03) VALUE SPACES.

PROCEDURE DIVISION.
******************************************************************************
```

ELECTR

```
PROGRAMA-PRINCIPAL.

    PERFORM ABRE-ARCHIVOS.
    PERFORM LEE-TABLA-DEP UNTIL NO-HAY-DATOS-TAB.
    PERFORM LEE-CLAVES    UNTIL NO-HAY-DATOS-CLAVE.
    PERFORM CIERRA-ARCHIVOS.
    STOP RUN.

ABRE-ARCHIVOS.

    OPEN INPUT  DEPENDENCIA.
    OPEN INPUT  CLAVES.
    OPEN OUTPUT REPORTE.
    SET INDICE-DEPENDENCIA TO 1.

LEE-TABLA-DEP.

    READ DEPENDENCIA AT END MOVE 'N' TO HAY-DATOS-TAB.
    IF NOT NO-HAY-DATOS-TAB THEN
       MOVE CLAVE-REC-DEP TO CLAVE-TAB-DEP(INDICE-DEPENDENCIA)
       MOVE NOMBRE-REC-DEP TO NOMBRE-TAB-DEP(INDICE-DEPENDENCIA)
       SET INDICE-DEPENDENCIA UP BY 1.
    ENDIF

LEE-CLAVES.

    READ CLAVES AT END MOVE 'N' TO HAY-DATOS-CLAVE.
    IF NOT NO-HAY-DATOS-CLAVE THEN
       PERFORM CHECA-CLAVES
       WRITE RECORD-REPORTE FROM SALIDA.
    ENDIF

CHECA-CLAVES.

    IF ( CARR-REC-CVE IS LESS THAN  21 )
       OR ( CARR-REC-CVE IS GREATER  32 )
       OR ( CARR-REC-CVE IS EQUAL  30 ) THEN
       MOVE SPACES TO CARR-SAL
       MOVE 'ERROR EN CARRERA ******* ' TO CARR-SAL
       MOVE CARR-REC-CVE TO CARR-SAL(28:3)
    ELSE
       MOVE TABLA-CARRERAS(CARR-REC-CVE) TO CARR-SAL.
    SET INDICE-DEPENDENCIA TO 1.
    SEARCH ALL TABLA-DEPENDENCIAS
           AT END PERFORM ERROR-DEPENDENCIA
           WHEN CLAVE-TAB-DEP(INDICE-DEPENDENCIA) IS EQUAL DEP-REC-CVE
           MOVE NOMBRE-TAB-DEP(INDICE-DEPENDENCIA) TO DEPEN-SAL.

ERROR-DEPENDENCIA.

    MOVE SPACES TO DEPEN-SAL.
    MOVE 'ERROR EN DEPENDENCIA' TO DEPEN-SAL.

CIERRA-ARCHIVOS.

    CLOSE DEPENDENCIA.
    CLOSE CLAVES.
    CLOSE REPORTE.
```

```
*****************************************************************************
 PROGRAM-ID. TARJETAS.
*****************************************************************************
*
* ESTE PROGRAMA CHECA LA SECUENCIA DE TARJETAS DE 4 EN 4
*
* FORMA UN NUEVO ARCHIVO SOLO CON LAS TARJETAS 1 Y 3.
*
*****************************************************************************

 ENVIRONMENT DIVISION.
*****************************************************************************

 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT INPUT-FILE ASSIGN TO "DATOS.DAT".
     SELECT OUTPUT-FILE ASSIGN TO "NUEVO.DAT".

 DATA DIVISION.
*****************************************************************************

 FILE SECTION.
 FD INPUT-FILE
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS RECORD-FILE
    LABEL RECORDS ARE STANDARD.
 01 RECORD-FILE.
    03 RECORD-PACK    PIC X(79).
    03 NUMERO-TARJETA PIC 9(01).
 FD OUTPUT-FILE
     RECORD CONTAINS 80 CHARACTERS
     DATA RECORD IS OUTPUT-RECORD
     LABEL RECORDS ARE STANDARD.
 01 OUTPUT-RECORD.
    03 FILLER         PIC X(80).

 WORKING-STORAGE SECTION.
 77 HAY-DATOS PIC X(01) VALUE "S".
 77 CONTADOR  PIC 9(04) VALUE 0.
 77 BASE      PIC 9(01) VALUE 1.

 PROCEDURE DIVISION.
*****************************************************************************

 PROGRAMA-PRINCIPAL.
     PERFORM ABRE-ARCHIVOS.
     PERFORM PROCESO UNTIL HAY-DATOS IS EQUAL "N".
     PERFORM SALIDA-FINAL.
     STOP RUN.

 ABRE-ARCHIVOS.
     OPEN INPUT INPUT-FILE.
     OPEN OUTPUT OUTPUT-FILE.

 PROCESO.
     READ INPUT-FILE AT END MOVE "N" TO HAY-DATOS.
     IF HAY-DATOS IS EQUAL "S" THEN
        PERFORM CHECA-SECUENCIA
        PERFORM INCREMENTA-SECUENCIA
        PERFORM GRABA-TARJETA.
*    END-IF
```

```
CHECA-SECUENCIA.
    ADD 1 TO CONTADOR.
    IF NUMERO-TARJETA NOT EQUAL BASE THEN
       DISPLAY 'FALTA LA TARJETA ' BASE ' EN LA SECUENCIA ' CONTADOR
       MOVE NUMERO-TARJETA  TO BASE.
*   END-IF.

INCREMENTA-SECUENCIA.
    ADD 1 TO BASE.
    IF BASE GREATER THAN 4 THEN
       MOVE 1 TO BASE.
*   END-IF.

GRABA-TARJETA.
    IF NUMERO-TARJETA EQUAL 1 OR NUMERO-TARJETA EQUAL 3 THEN
       MOVE RECORD-FILE TO OUTPUT-RECORD
       WRITE OUTPUT-RECORD.
*   ENDIF

SALIDA-FINAL.
    DISPLAY 'NUMERO DE TARJETAS LEIDAS ' CONTADOR.
```

```
*****************************************************************
PROGRAM-ID. REFORMATEA.

ENVIRONMENT DIVISION.
*****************************************************************

INPUT-OUTPUT SECTION.

FILE-CONTROL.
SELECT ENTRADA ASSIGN "DBA1:[SDE000.COBOL.FILES]RFC.DAT".
SELECT SALIDA  ASSIGN "DBA1:[SDE000.COBOL.FILES]RFCFORM.DAT".

DATA DIVISION.
*****************************************************************

FILE SECTION.
FD   ENTRADA
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-ENTRADA.

01 RECORD-ENTRADA.
     03 NOMBRE     PIC X(40).
     03 RFC        PIC X(10).
     03 IMSS.
        05 NUMERO PIC 9(04).
        05 DFO    PIC X(01).
     03 NUMEROST.
        05 HT-N      PIC 9(02).
        05 HT-E      PIC 9(02).
        05 PT-N      PIC 9(04).
        05 PT-E      PIC 9(04).
     03 FILLER    PIC X(13).

FD   SALIDA
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-SALIDA.

01   RECORD-SALIDA PIC X(80).

WORKING-STORAGE SECTION.

77 HAY-DATOS    PIC X(01) VALUE 'S'.

01 BASE-SALIDA.
     03 NOMBRE     PIC X(40).
     03 RFC        PIC X(10).
     03 IMSS.
        05 NUMERO PIC 9(04).
        05 DFO    PIC X(01).
     03 NUMEROST.
        05 HT-N      PIC 9(02).
        05 HT-E      PIC 9(02).
        05 PT-N      PIC 9(04).
        05 PT-E      PIC 9(04).
     03 FILLER    PIC X(13) VALUE SPACES.

PROCEDURE DIVISION.
*****************************************************************

PROGRAMA-PRINCIPAL.
```

```
      PERFORM ABRE-ARCHIVOS.
      PERFORM PROCESO-REFORMATEO UNTIL HAY-DATOS EQUAL "N".
      PERFORM CIERRA-ARCHIVOS.
      STOP RUN.

ABRE-ARCHIVOS.

      OPEN INPUT ENTRADA.
      OPEN OUTPUT SALIDA.

PROCESO-REFORMATEO.

      READ ENTRADA AT END MOVE "N" TO HAY-DATOS.
      IF HAY-DATOS EQUAL "S" THEN
         PERFORM EVALUA.
*     ENDIF

EVALUA.

      MOVE CORRESPONDING RECORD-ENTRADA TO BASE-SALIDA.

      IF HT-N IN RECORD-ENTRADA IS LESS THAN HT-E IN RECORD-ENTRADA THEN
         MOVE HT-N IN RECORD-ENTRADA TO HT-E IN BASE-SALIDA
         MOVE HT-E IN RECORD-ENTRADA TO HT-N IN BASE-SALIDA.
*     ENDIF

      IF PT-N IN RECORD-ENTRADA IS GREATER THAN PT-E IN RECORD-ENTRADA THEN
         MOVE PT-N IN RECORD-ENTRADA TO PT-E IN BASE-SALIDA
         MOVE PT-E IN RECORD-ENTRADA TO PT-N IN BASE-SALIDA.
*     ENDIF

      IF PT-N IN BASE-SALIDA IS EQUAL ZEROES THEN
         MOVE 30 TO PT-N IN BASE-SALIDA.
*     ENDIF

      IF PT-E IN BASE-SALIDA IS EQUAL ZEROES THEN
         MOVE 90 TO PT-E IN BASE-SALIDA.
*     ENDIF

      WRITE RECORD-SALIDA FROM BASE-SALIDA.


CIERRA-ARCHIVOS.

      CLOSE ENTRADA.
      CLOSE SALIDA.
```

```
*****************************************************************
IDENTIFICATION DIVISION.
*****************************************************************

PROGRAM-ID.                 TRANSFERENCIA-DE-NUMEROS.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               2-JULIO-1983.
DATE-COMPILED.
*
*     PROGRAMA QUE TRANFIERE NUMEROS DE UN ARCHIVO DE ENTRADA A UN
*     ARCHIVO DE SALIDA FORMATEADO E INDICA EL NUMERO DE REGISTROS
*     LEIDOS.

*****************************************************************
ENVIRONMENT   DIVISION.
*****************************************************************

CONFIGURATION  SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

INPUT-OUTPUT   SECTION.
FILE-CONTROL.
       SELECT ARCHIVO-ENTRADA ASSIGN TO "DBA1:.ENT".
       SELECT ARCHIVO-SALIDA  ASSIGN TO "DBA1:.SAL".

*****************************************************************
DATA          DIVISION.
*****************************************************************

FILE    SECTION.
FD      ARCHIVO-ENTRADA
        VALUE  OF ID   IS "DVALERIO.COBOL.DATOSINUMEROS".
01      REGISTRO-DE-ENTRADA.
        03 PARTE-UNO-ENTRADA       PIC 9(6).
        03 PARTE-DOS-ENTRADA       PIC 9(6).
        02 FILLER                  PIC X(68).
FD      ARCHIVO-SALIDA
        VALUE  OF ID   IS "DVALERIO.COBOL.REPORTESINUMEROS".
01      REGISTRO-DE-SALIDA.
        03 LIMP-UNO                PIC X(10).
        03 PARTE-UNO-SALIDA        PIC 9(6).
        03 LIMP-DOS                PIC X(10).
        03 PARTE-DOS-SALIDA        PIC 9(6).
        03 LIMP-TRES               PIC X(46).

WORKING-STORAGE SECTION.
77      REGISTROS-LEIDOS          PIC 9(3)  VALUE 000.
77      HAY-MAS-DATOS             PIC X(2)  VALUE "SI".

*****************************************************************
PROCEDURE DIVISION.
*****************************************************************

PRINCIPIO.
        PERFORM ABRE-ARCHIVOS.
        PERFORM REALIZA-CALCULOS.
        PERFORM CIERRA-ARCHIVOS.
        STOP RUN.

ABRE-ARCHIVOS.
        OPEN INPUT  ARCHIVO-ENTRADA
```

```
REALIZA-CALCULOS.
        PERFORM CONTINUA WITH TEST BEFORE
                         UNTIL NOT HAY-MAS-DATOS IS EQUAL 'SI'.
        PERFORM FIN-DE-PROCESO.

CONTINUA.
        READ ARCHIVO-ENTRADA AT END MOVE 'NO' TO HAY-MAS-DATOS.
        IF HAY-MAS-DATOS EQUAL 'SI' THEN
            ADD 1 TO REGISTROS-LEIDOS
            MOVE PARTE-UNO-ENTRADA  TO PARTE-DOS-SALIDA
            MOVE PARTE-DOS-ENTRADA  TO PARTE-UNO-SALIDA
            MOVE SPACES             TO LIMP-UNO
            MOVE SPACES             TO LIMP-DOS
            MOVE SPACES             TO LIMP-TRES
            WRITE REGISTRO-DE-SALIDA
        ELSE
            NEXT SENTENCE.
        END-IF.

FIN-DE-PROCESO.
        DISPLAY 'TOTAL DE REGISTROS LEIDOS ---> ' REGISTROS-LEIDOS.

CIERRA-ARCHIVOS.
        CLOSE ARCHIVO-ENTRADA ARCHIVO-SALIDA.
```

```
*****************************************************************
IDENTIFICATION DIVISION.
*****************************************************************

PROGRAM-ID.                 COB02.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               2-JULIO-1983.
DATE-COMPILED.

*           PROGRAMA QUE EMITE UN REPORTE DE CALIFICACIONES.
*           PROGRAMA DE TAREA DEL CURSO DE COBOL JULIO 83.
*
*           ESTE PROGRAMA USA LOS ARCHIVOS EN DISCO
*
*          CALIFALU.ENT:
*
*                 REG-1 NOMBRE DEL ALUMNO QUE HACE EL PROGRAMA
*                       Y SU NUMERO DE CUENTA
*
*                 REG-2 NOMBRE-DE-ALUMNO DE CLASE.
*                       CALIF UNO,DOS,Y DE EXAMEN
*
*          CALIFALU.SAL:
*
*                 IDENTIFICACION DEL CURSO GRUPO Y LUGAR
*                 NOMBRE Y NUMERO DE CUNTA DEL ALUMNO
*                 TABLA CON EL NOMBRE, CALIFICACIONES PARCIALES
*                 CALIFICACION EXAMEN Y PROMEDIO PONDERADO
*                 DEFINIDO POR LA FORMULA ((C1+C2)/2)*0.2 + CEX*0.8
*                 Y EL PROMEDIO GENERAL DEL CALCULO.
*

*****************************************************************
ENVIRONMENT   DIVISION.
*****************************************************************

CONFIGURATION  SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

INPUT-OUTPUT   SECTION.
FILE-CONTROL.
    SELECT ARCHIVO-ENTRADA ASSIGN TO 'DBA1:.ENT'.
    SELECT ARCHIVO-SALIDA  ASSIGN TO 'DBA1:.SAL'.

*****************************************************************
DATA           DIVISION.
*****************************************************************

FILE           SECTION.
FD    ARCHIVO-ENTRADA
      VALUE OF ID    IS "EVALERIO,COBOL,DATOS]CALIFALU"
      RECORD CONTAINS 80 CHARACTERS
      DATA RECORDS ARE NOMBRE-ALUMNO
                       REGISTRO-DE-ENTRADA.
01    NOMBRE-ALUMNO.
      03 NOMBRE-ALUMNO-CURSO     PIC X(40).
      03 NUMERO-DE-CUENTA        PIC 9(07).
      03 DIGITO-VERIFICADOR      PIC 9(01).
      03 FILLER                  PIC X(31).
01    REGISTRO-DE-ENTRADA.
      03 NOMBRE-ENTRADA          PIC X(32).
```

```
           03 CARRERA                  PIC X(03).
           03 CALIF-1                  PIC 9(02)V9(02).
           03 CALIF-2                  PIC 9(02)V9(02).
           03 CALIF-EXAMEN             PIC 9(02)V9(02).
           03 FILLER                   PIC X(34).
FD         ARCHIVO-SALIDA
           VALUE OF ID IS "EVALER10.COBOL.REPORTES.CALIFALU".
01         REGISTRO-DE-SALIDA          PIC X(132).

WORKING-STORAGE SECTION.
77         HAY-MAS-DATOS               PIC X(02)  VALUE  "SI".
77         SUMA-DE-PROMEDIO            PIC 9(06)  VALUE  ZEROES.
77         CPROMEDIO                   PIC 9(04)  VALUE  ZEROES.
77         PRIMERO                     PIC X(02)  VALUE  "SI".
77         SUMA-DE-ALUMNOS             PIC 9(04)  VALUE  ZEROES.

01         LETRERO-1.
           03 FILLER                    PIC X(58)  VALUE  SPACES.
           03 FILLER                    PIC X(17)  VALUE  "CENTRO DE CALCULO".
           03 FILLER                    PIC X(57)  VALUE  SPACES.

01         LETRERO-2.
           03 FILLER                    PIC X(59)  VALUE  SPACES.
           03 FILLER                    PIC X(14)  VALUE  "CURSO DE COBOL".
           03 FILLER                    PIC X(59)  VALUE  SPACES.

01         LETRERO-3.
           03 FILLER                    PIC X(62)  VALUE  SPACES.
           03 FILLER                    PIC X(09)  VALUE  "GRUPO 12".
           03 FILLER                    PIC X(62)  VALUE  SPACES.

           ALUMNO-SALIDA.
           03 FILLER                    PIC X(20)  VALUE  SPACES.
           03 FILLER                    PIC X(13)  VALUE  "ALUMNO      : ".
           03 NOMBRE-ALUMNO-CURSO       PIC X(40).
           03 FILLER                    PIC X(59)  VALUE  SPACES.

01         NUMERO-DE-CUENTA-SALIDA.
           03 FILLER                    PIC X(20)  VALUE  SPACES.
           03 FILLER                    PIC X(13)  VALUE  "NO. CUENTA: ".
           03 NUMERO-DE-CUENTA          PIC 9(07).
           03 FILLER                    PIC X(01)  VALUE  "-".
           03 DIGITO-VERIFICADOR        PIC 9(01).
           03 FILLER                    PIC X(90).

01         ENCABEZADO-1.
           03 FILLER                    PIC X(07)  VALUE  SPACES.
           03 FILLER                    PIC X(13)  VALUE  "CARRERA      ".
           03 FILLER                    PIC X(12)  VALUE  "N O M B R E.".
           03 FILLER                    PIC X(20)  VALUE  SPACES.
           03 FILLER                    PIC X(15)  VALUE  "CALIFICACION   ".
           03 FILLER                    PIC X(15)  VALUE  "CALIFICACION   ".
           03 FILLER                    PIC X(15)  VALUE  "CALIFICACION   ".
           03 FILLER                    PIC X(13)  VALUE  "  PROMEDIO   ".
           03 FILLER                    PIC X(20)  VALUE  SPACES.

01         ENCABEZADO-2.
           03 FILLER                    PIC X(52)  VALUE  SPACES.
           03 FILLER                    PIC X(15)  VALUE  "    UNO      ".
           03 FILLER                    PIC X(15)  VALUE  "    DOS      ".
           03 FILLER                    PIC X(15)  VALUE  "   EXAMEN    ".
           03 FILLER                    PIC X(35)  VALUE  SPACES.
```

```
01      ENCABEZADO-1.
        03 FILLER                    PIC X(132) VALUE ALL "=".

01      REGISTRO-SALIDA.
        03 FILLER                    PIC X(10)  VALUE SPACES.
        03 CARRERA                   PIC 9(02).
        03 FILLER                    PIC X(08)  VALUE SPACES.
        03 NOMBRE-ENTRADA            PIC X(32).
        03 FILLER                    PIC X(03)  VALUE SPACES.
        03 CALIF-1                   PIC Z(01)9(01).9(02).
        03 FILLER                    PIC X(10)  VALUE SPACES.
        03 CALIF-2                   PIC Z(01)9(01).9(02).
        03 FILLER                    PIC X(10)  VALUE SPACES.
        03 CALIF-EXAMEN              PIC Z(01)9(01).9(02).
        03 FILLER                    PIC X(10)  VALUE SPACES.
        03 PROMEDIO                  PIC Z(01)9(01).9(02).
        03 FILLER                    PIC X(27)  VALUE SPACES.

01      SALIDA-PROMEDIO-GENERAL.
        03 FILLER                    PIC X(50)  VALUE SPACES.
        03 FILLER                    PIC X(15)  VALUE "PROM. GRAL. : ".
        03 PROMEDIO-GENERAL          PIC Z(01)9(01).9(02).
        03 FILLER                    PIC X(62)  VALUE SPACES.

PROCEDURE DIVISION.

PRINCIPIO.

        PERFORM ABRE-ARCHIVOS.
        PERFORM REGISTROS-ALUMNOS WITH TEST BEFORE
                                  UNTIL NOT HAY-MAS-DATOS = "SI".
        PERFORM CIERRA-ARCHIVOS.
        STOP RUN.

ABRE-ARCHIVOS.
        OPEN    INPUT     ARCHIVO-ENTRADA.
        OPEN    OUTPUT    ARCHIVO-SALIDA.

REGISTROS-ALUMNOS.
        READ          ARCHIVO-ENTRADA
                AT END MOVE "NO" TO HAY-MAS-DATOS
        END-READ.
        IF    HAY-MAS-DATOS = "SI" THEN
              IF PRIMERO = "SI" THEN
                 MOVE "NO" TO PRIMERO
                  PERFORM ESCRIBE-LETREROS
                  PERFORM ESCRIBE-ALUMNO
                  PERFORM ESCRIBE-ENCABEZADO
              ELSE
                 PERFORM ESCRIBE-REGISTRO-ALUMNO
              END-IF
        ELSE
              DISPLAY " FIN DE DATOS DE ENTRADA"
              PERFORM ESCRIBE-TOTAL-PROMEDIO
        END-IF.

ESCRIBE-LETREROS.
        WRITE       REGISTRO-DE-SALIDA
              FROM  LETRERO-1
              AFTER 1 LINES.
*       END-WRITE.
        WRITE       REGISTRO-DE-SALIDA
              FROM  LETRERO-2
```

```
*           AFTER  1 LINES.
*           END-WRITE
            WRITE       REGISTRO-DE-SALIDA
                  FROM  LETRERO-3
                  AFTER 2 LINES.
            END-WRITE

ESCRIBE-ALUMNO.
            MOVE CORRESPONDING NOMBRE-ALUMNO TO ALUMNO-SALIDA.
            WRITE       REGISTRO-DE-SALIDA
                  FROM  ALUMNO-SALIDA
                  AFTER 2 LINES.
*           END-WRITE
            MOVE CORRESPONDING NOMBRE-ALUMNO TO NUMERO-DE-CUENTA-SALIDA.
            WRITE       REGISTRO-DE-SALIDA
                  FROM  NUMERO-DE-CUENTA-SALIDA
                  AFTER 1 LINES.
*           END-WRITE

ESCRIBE-ENCABEZADO.
            WRITE       REGISTRO-DE-SALIDA
                  FROM  ENCABEZADO-1
                  AFTER 2 LINES.
*           END-WRITE
            WRITE       REGISTRO-DE-SALIDA
                  FROM  ENCABEZADO-2
                  AFTER 1 LINES.
*           END-WRITE
            WRITE       REGISTRO-DE-SALIDA
                  FROM  ENCABEZADO-3
                  AFTER 1 LINES.
            END-WRITE
            WRITE       REGISTRO-DE-SALIDA
                  FROM  ENCABEZADO-3
                  AFTER 1 LINES.
*           END-WRITE

ESCRIBE-REGISTRO-ALUMNO.
            MOVE CORRESPONDING REGISTRO-DE-ENTRADA TO REGISTRO-SALIDA.
            COMPUTE CPROMEDIO = ((CALIF-1 IN REGISTRO-DE-ENTRADA
                                + CALIF-2 IN REGISTRO-DE-ENTRADA)/2)*0.2
                                + CALIF-EXAMEN IN REGISTRO-DE-ENTRADA*0.8.
            MOVE CPROMEDIO TO PROMEDIO IN REGISTRO-SALIDA.
            COMPUTE SUMA-DE-PROMEDIO = SUMA-DE-PROMEDIO + CPROMEDIO.
            COMPUTE SUMA-DE-ALUMNOS  = SUMA-DE-ALUMNOS + 1.
            WRITE       REGISTRO-DE-SALIDA
                  FROM  REGISTRO-SALIDA
                  AFTER 1 LINES.
*           END-WRITE

ESCRIBE-TOTAL-PROMEDIO.
            COMPUTE CPROMEDIO = SUMA-DE-PROMEDIO/SUMA-DE-ALUMNOS.
            MOVE CPROMEDIO TO PROMEDIO-GENERAL
                           IN SALIDA-PROMEDIO-GENERAL.
            WRITE       REGISTRO-DE-SALIDA
                  FROM  SALIDA-PROMEDIO-GENERAL
                  AFTER 3 LINES.
            END-WRITE

CIERRA-ARCHIVOS.

            CLOSE ARCHIVO-ENTRADA.
            CLOSE ARCHIVO-SALIDA.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.                    PANTALLA-MB.
AUTHOR.                        VALERIO JORGE.
INSTALLATION.                  CECAFI
DATE-WRITTEN.                  25-OCT-84.
DATE-COMPILED.
SECURITY.                      PROCESO QUE CAPTURA Y MUESTRA ALUMNOS
                               CON MB Y GUARDA LOS REGISTROS EN UN
                               ARCHIVO.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.               VAX-11.
OBJECT-COMPUTER.               VAX-11.
SPECIAL-NAMES.
    SYMBOLIC CHARACTERS ESCAPER PAREN-DER SEMICOLON CLEAR LUGAR ARE
                        28      92        60        75     103.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO ASSIGN "EVALERIO.COBOL.DATOS]ALUMCPAN.DAT".

DATA DIVISION.
FILE SECTION.
FD ARCHIVO
    RECORD CONTAINS 42 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORD IS ARCH-REG.
01 ARCH-REG.
    03 MATERIA-REGISTRO          PIC X(03).
    03 FILLER                    PIC X(01).
    03 NOMBRE-REGISTRO           PIC X(25).
    03 FILLER                    PIC X(01).
    03 NUMERO-REGISTRO           PIC X(08).
    03 FILLER                    PIC X(01).
    03 CALIFICACION-REGISTRO     PIC X(02).
    03 FILLER                    PIC X(01).
WORKING-STORAGE SECTION.
77 CONTADOR-MB          PIC 9(03) VALUE ZEROES.
77 HAY-DATOS            PIC X(01) VALUE 'S'.
01 REGISTRO.
    03 MATERIA-REGISTRO          PIC X(03).
    03 FILLER                    PIC X(01).
    03 NOMBRE-REGISTRO           PIC X(25).
    03 FILLER                    PIC X(01).
    03 NUMERO-REGISTRO           PIC X(08).
    03 FILLER                    PIC X(01).
    03 CALIFICACION-REGISTRO     PIC X(02).
    03 FILLER                    PIC X(01).
01 LISTA.
    03 FILLER           PIC X(01) VALUE ESCAPER.
    03 FILLER           PIC X(01) VALUE PAREN-DER.
    03 FILLER           PIC 9(02) VALUE 20.
    03 FILLER           PIC X(01) VALUE SEMICOLON.
    03 FILLER           PIC 9(02) VALUE 10.
    03 FILLER           PIC X(01) VALUE LUGAR.
    03 FILLER           PIC X(12) VALUE " El alumno ".
    03 NOMBRE-LISTA     PIC X(25).
    03 FILLER           PIC X(10) VALUE " tiene MB.".
    03 FILLER5          PIC X(33) VALUE SPACES.
01 LIMPIA-PANTALLA.
    03 FILLER           PIC X(01) VALUE ESCAPER.
    03 FILLER           PIC X(01) VALUE PAREN-DER.
    03 FILLER           PIC 9(02) VALUE 02.
```

```
    03 FILLER                PIC X(01)  VALUE CLEAR.
01  FON-CURSOR.
    03 FILLER                PIC X(01)  VALUE ESCAPE.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 RENGLON               PIC 9(02)  VALUE ZEROES.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 COLUMNA               PIC 9(02)  VALUE ZEROES.
    03 FILLER                PIC X(01)  VALUE LUGAR.
01  PANTALLA.
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 07.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 30.
    03 FILLER                PIC X(01)  VALUE LUGAR
    03 FILLER                PIC X(18)  VALUE "Captura de datos.".
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 10.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 15.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(13)  VALUE "Materia ____.".
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 12.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 15.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(34)  VALUE "Nombre ____________________.
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 14.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 15.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(17)  VALUE "Numero _____".
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 16.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 15.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(10)  VALUE "Calif __.".
01  CONT.
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 23.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 33.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(38)  VALUE
                             "Continuas introduciendo datos (S/N) ? ".
01  PANTALLA-FINAL.
    03 FILLER                PIC X(01)  VALUE ESCAPER.
    03 FILLER                PIC X(01)  VALUE PAREN-DER.
    03 FILLER                PIC 9(02)  VALUE 23.
    03 FILLER                PIC X(01)  VALUE SEMICOLON.
    03 FILLER                PIC 9(02)  VALUE 33.
    03 FILLER                PIC X(01)  VALUE LUGAR.
    03 FILLER                PIC X(27)  VALUE "Total de alumnos con MR : ".

PROCEDURE DIVISION.
```

```
    OPEN OUTPUT ARCHIVO.
    PERFORM PROCESO-LISTA
            WITH TEST BEFORE
            UNTIL HAY-DATOS EQUAL "N".
    PERFORM SALIDA-FINAL.
    STOP RUN.

PROCESO-LISTA.
    PERFORM CAPTURA.
    PERFORM EVALUA.
    PERFORM CONTINUA.

CAPTURA.
    DISPLAY LIMPIA-PANTALLA.
    DISPLAY PANTALLA.
    MOVE 10 TO RENGLON.
    MOVE 23 TO COLUMNA.
    DISPLAY PON-CURSOR WITH NO ADVANCING.
    ACCEPT MATERIA-REGISTRO IN REGISTRO.
    MOVE 12 TO RENGLON.
    MOVE 22 TO COLUMNA.
    DISPLAY PON-CURSOR WITH NO ADVANCING.
    ACCEPT NOMBRE-REGISTRO IN REGISTRO.
    MOVE 14 TO RENGLON.
    MOVE 22 TO COLUMNA.
    DISPLAY PON-CURSOR WITH NO ADVANCING.
    ACCEPT NUMERO-REGISTRO IN REGISTRO.
    MOVE 16 TO RENGLON.
    MOVE 21 TO COLUMNA.
    DISPLAY PON-CURSOR WITH NO ADVANCING.
    ACCEPT CALIFICACION-REGISTRO IN REGISTRO.
    WRITE ARCH-REG FROM REGISTRO.

EVALUA.
    IF CALIFICACION-REGISTRO IN REGISTRO EQUAL "MB" THEN
       MOVE NOMBRE-REGISTRO IN REGISTRO TO NOMBRE-LISTA
       DISPLAY LISTA
       ADD 1 TO CONTADOR-MB.
*   NOELSE.
*   ENDIF

CONTINUA.

    MOVE "N" TO HAY-DATOS.
    PERFORM PIDE
            UNTIL HAY-DATOS EQUAL "S" OR HAY-DATOS EQUAL "N".

PIDE.
    DISPLAY CONT WITH NO ADVANCING.
    ACCEPT HAY-DATOS.

SALIDA-FINAL.
    DISPLAY LIMPIA-PANTALLA.
    DISPLAY PANTALLA-FINAL WITH NO ADVANCING.
    DISPLAY CONTADOR-MB.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    CUENTA-REGISTROS.
AUTHOR.        JORGE VALERIO.
INSTALLATION.  CECAFI.
DATE-WRITTEN.  25-OCT-84.
ATE-COMPILED.
ECURITY.       PROGRAMA QUE CUENTA CUANTOS
               REGISTROS SON INTRODUCIDOS
               Y LOS GUARDA EN UN ARCHIVO.

*              REVISADO 26 DE ENERO DE 1985.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO ASSIGN "DVALERIO.COBOL.DATOS1CAPREG.DAT".

DATA DIVISION.
FILE SECTION.
FD ARCHIVO
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORD IS ARCH-REG.
01 ARCH-REG.
    05 MRC-ARC-TRJ-WS   PIC 9(03).
    05 DTA-ARC-TRJ-WS   PIC X(77).
ORKING-STORAGE SECTION.
 7 CNT-TRJ-WS          PIC 9(03) VALUE ZEROES.
01 RGT-TRJ-WS.
    05 MRC-RGT-TRJ-WS   PIC 9(03).
    05 DTA-RGT-TRJ-WS   PIC X(77).

PROCEDURE DIVISION.

PROGRAMA-PRINCIPAL.
    OPEN OUTPUT ARCHIVO.
    PERFORM LEE-TARJETA.
    PERFORM CUENTA-TARJETAS UNTIL MRC-RGT-TRJ-WS = 999.
*   ENDDO
    PERFORM SALIDA.
    CLOSE ARCHIVO.
    STOP RUN.

LEE-TARJETA.
    DISPLAY " INFORMACION (9999XXXXXX,...) CEND 9999 IN NUMERIC) -> " WITH NO A
    ACCEPT RGT-TRJ-WS.                                              ADVANCI

CUENTA-TARJETAS.
    ADD 1 TO CNT-TRJ-WS.
    MOVE RGT-TRJ-WS TO ARCH-REC.
    WRITE ARCH-REG.
    PERFORM LEE-TARJETA.

SALIDA.
    DISPLAY " EL NUMERO DE TARJETAS SON : " CNT-TRJ-WS.
```

```
*****************************************************************
IDENTIFICATION DIVISION.
*****************************************************************

PROGRAM-ID.                 DEMOSTRACION-2.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.

*****************************************************************
ENVIRONMENT    DIVISION.
*****************************************************************

CONFIGURATION  SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

SPECIAL-NAMES.
        CONSOLE IS PANTALLA
        SYMBOLIC CHARACTERS ESC  BJ  P2  P3 P4 BELL DOS
                            ARE 28   92  60 103 75  8   2.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT DATOS-ENTRADA ASSIGN TO SYS$INPUT.
        SELECT DATOS-ENT-NUM ASSIGN TO SYS$INPUT.

*****************************************************************
DATA    DIVISION.
*****************************************************************

FILE SECTION.
FD      DATOS-ENT-NUM
        DATA RECORD IS      NUMERO-ENT.
01      NUMERO-ENT.
        03 NUMERO            PIC 9(06).
        03 CHAR-NUMERO       REDEFINES
        NUMERO              PIC X(06).
FD      DATOS-ENTRADA
        DATA RECORD  IS     NOMBRE-ENT.
01      NOMBRE-ENT.
        03 NOMBRE            PIC X(15).

WORKING-STORAGE SECTION.
77      CONTINUACION        PIC X(02)    VALUE "SI".

PROCEDURE DIVISION.

PRINCIPIO.

        PERFORM ABRE-ARCHIVOS.
        PERFORM PIDE-NOMBRE.
        PERFORM MANEJO-DE-DATOS WITH TEST BEFORE
                                UNTIL CONTINUACION = "NO".
        PERFORM BORRA.
        STOP RUN.

ABRE-ARCHIVOS.
        OPEN   INPUT    DATOS-ENTRADA.
        OPEN   INPUT    DATOS-ENT-NUM.

BORRA.
```

```
           DISPLAY ESC P1 10 P2 01 P3          WITH NO ADVANCING.
           DISPLAY ESC P1 DOS P4               WITH NO ADVANCING.

PIDE-NOMBRE.
           PERFORM BORRA.
           DISPLAY " HOLA DAME TU NOMBRE " WITH NO ADVANCING.
           READ    DATOS-ENTRADA
                   AT END DISPLAY " YA NO HAY DATOS "
           END-READ.

MANEJO-DE-DATOS.
           PERFORM PIDE-NUMERO.
           DISPLAY NOMBRE " TU NUMERO ES " NUMERO.
           PERFORM PIDE-CONTINUACION.

PIDE-NUMERO.
           PERFORM BORRA.
           DISPLAY NOMBRE " DAME UN NUMERO " WITH NO ADVANCING.
           READ DATOS-ENT-NUM
                AT END DISPLAY "YA NO HAY DATOS"
                   STOP RUN
           END-READ.


PIDE-CONTINUACION.
           DISPLAY " ".
           DISPLAY "%COBO4 ? " NOMBRE
                   " CONTINUAS CON EL PROGRAMA <SI O NO> ? "
                      WITH NO ADVANCING.
           ACCEPT CONTINUACION FROM PANTALLA.
           IF CONTINUACION EQUAL "NO" THEN
              STOP RUN
           ELSE
              NEXT SENTENCE.
           END IF
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MIO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CONSOLE IS PANTALLA
    SYMBOLIC CHARACTERS
    TRES  ESCAPE CORCHETE PUNTOYCOMA JOTA ACHE PARENTESIS DOS GATO BE
     ARE   3   28   92       60       75   73    41       2    36  67.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I PIC 9(02).
77 J PIC 9(01) VALUE 1.
77 NOMBRE PIC X(10).
77 NUMERO PIC 9(06).
PROCEDURE DIVISION.
EMPIEZA.
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 ACHE WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE DOS JOTA WITH NO ADVANCING
    DISPLAY ESCAPE PARENTESIS 0
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 ACHE "I" WITH NO ADVANCING
    PERFORM RAYA VARYING I FROM 2 BY 1 UNTIL (I > 39)
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 40 ACHE "k" WITH NO ADVANCING
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 17 ACHE "COBOL"
            WITH NO ADVANCING
    DISPLAY ESCAPE GATO 3 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 00 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 40 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 17 ACHE "COBOL"
              WITH NO ADVANCING
    DISPLAY ESCAPE GATO 4 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 00 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 40 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 00 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 40 ACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 05 PUNTOYCOMA 00 ACHE "x" WITH NO ADVANCING
    ADD 4 TO J
    PERFORM RAYA VARYING I FROM 2 BY 1 UNTIL (I > 39)
    DISPLAY ESCAPE CORCHETE 05 PUNTOYCOMA 80 ACHE "J" WITH NO ADVANCING
    DISPLAY ESCAPE PARENTESIS BE
    DISPLAY ESCAPE CORCHETE 14 PUNTOYCOMA J ACHE WITH NO ADVANCING
    DISPLAY " HOLA DAME TU NOMBRE  : " WITH NO ADVANCING
    ACCEPT NOMBRE
    DISPLAY ESCAPE CORCHETE 16 PUNTOYCOMA 3 ACHE WITH NO ADVANCING
    DISPLAY NOMBRE " DAME UN NUMERO : " WITH NO ADVANCING
    ACCEPT NUMERO
    DISPLAY " "
    DISPLAY " "
    DISPLAY NOMBRE " TU NUMERO ES : " NUMERO
    DISPLAY ESCAPE CORCHETE 20 PUNTOYCOMA 1 ACHE WITH NO ADVANCING
    STOP RUN.
RAYA.
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE J PUNTOYCOMA I ACHE "a" WITH NO ADVANCING.
```

```
IDENTIFICATION DIVISION.
*************************************************************************
PROGRAM-ID. SORTEA.

ENVIRONMENT DIVISION.
*************************************************************************

INPUT-OUTPUT SECTION.

FILE-CONTROL.

    SELECT INPUT-FILE  ASSIGN 'DBA1:[SDE000.COBOL.FILES]INFILE.DAT'.
    SELECT OUTPUT-FILE ASSIGN 'DBA1:[SDE000.COBOL.FILES]OUTFILE.DAT'.
    SELECT SORT-FILE   ASSIGN 'DBA1:[SDE000.COBOL.FILES]SRTFILE.DAT'.

DATA DIVISION.
*************************************************************************

FILE SECTION.

FD INPUT-FILE
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS INPUT-RECORD.

01 INPUT-RECORD PIC X(80).

FD OUTPUT-FILE
    RECORD CONTAINS 80 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS OUTPUT-RECORD.

01 OUTPUT-RECORD PIC X(80).

SD SORT-FILE
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS RECORD-SORT.

01 RECORD-SORT.
    03 NOMBRE    PIC X(40).
    03 FILLER    PIC X(14).
    03 GPO-IMSS  PIC X(01).
    03 FILLER    PIC X(25).

PROCEDURE DIVISION.
*************************************************************************

REALIZA-SORTEO.

    SORT SORT-FILE ON ASCENDING KEY NOMBRE
                      DESCENDING KEY GPO-IMSS
                      USING INPUT-FILE
                      GIVING OUTPUT-FILE.
    STOP RUN.
```

```
*****************************************************************
IDENTIFICATION DIVISION.
*****************************************************************

PROGRAM-ID.                 COB03.
AUTHOR.                     JORGE-VALERIO.
INSTALLATION.               CECAFI.
DATE-WRITTEN.               29-SEP-83.
DATE-COMPILED.              29-SEP-83.
*
*    PROGRAMA SIMPLE DE DEMOSTRACION
*

*****************************************************************
ENVIRONMENT    DIVISION.
*****************************************************************

CONFIGURATION  SECTION.
SOURCE-COMPUTER.            VAX11-780.
OBJECT-COMPUTER.            VAX11-780.

*****************************************************************
DATA           DIVISION.
*****************************************************************

WORKING-STORAGE SECTION.
77     NUMERO               PIC 9(06).
77     NOMBRE               PIC X(40).

*****************************************************************
PROCEDURE DIVISION.
*****************************************************************

PRINCIPIO.
    DISPLAY " HOLA DAME TU NOMBRE " WITH NO ADVANCING.
    ACCEPT NOMBRE.
    DISPLAY NOMBRE " DAME UN NUMERO " WITH NO ADVANCING.
    ACCEPT NUMERO.
    DISPLAY " TU NUMERO ES " NUMERO.
    STOP RUN.
```

```
IDENTIFICATION DIVISION.
************************************************************************************
                                              PROGRAM - ID. DESCONOCIDO? ——>
ENVIRONMENT DIVISION.                         ESTO FUÉ POR UN TAB (tabulador)
************************************************************************************

INPUT-OUTPUT SECTION.

FILE-CONTROL.

    SELECT SECUENCIAL   ASSIGN TO DISK.
    SELECT INDEXADO     ASSIGN TO DISK
           ORGANIZATION IS INDEXED
           ACCESS MODE IS DYNAMIC
           RECORD KEY IS R-I-CLAVE.
    SELECT REPORTE      ASSIGN TO DISK.

DATA DIVISION.
************************************************************************************

FILE SECTION.

FD   SECUENCIAL
     RECORD CONTAINS 80 CHARACTERS
     VALUE OF ID "DEPENSEC.DAT"
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-SECUENCIAL.

01 R-SECUENCIAL.
   03 R-S-CLAVE        PIC X(03).
   03 FILLER           PIC X(02).
   03 R-S-NOMBRE       PIC X(50).
   03 FILLER           PIC X(25).

FD   INDEXADO
     RECORD CONTAINS 80 CHARACTERS
     VALUE OF ID "DEPENIND.DAT"
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-INDEXADO.

01 R-INDEXADO.
   03 R-I-CLAVE        PIC X(03).
   03 FILLER           PIC X(02).
   03 R-I-NOMBRE       PIC X(50).
   03 FILLER           PIC X(25).

FD   REPORTE
     RECORD CONTAINS 132 CHARACTERS
     VALUE OF ID "DEPENREP.DAT"
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS R-REPORT.

01   R-REPORT PIC X(132).

WORKING-STORAGE SECTION.

 77 HAY-DATOS-SEC     PIC X(01) VALUE "S".
    88 NO-HAY-DATOS-SEC VALUE "N".
 77 HAY-DATOS-CLAVE   PIC X(01) VALUE "S".
    88 NO-HAY-DATOS-CLAVE VALUE "N".
 77 ERROR-INDEXADO    PIC X(01) VALUE "N".
    88 ERROR-INDEXADO-OK VALUE "S".
```

```
01 DEPENDENCIAS-DEP.
   03 TABLA-DEPENDENCIAS OCCURS 300 TIMES
                         ASCENDING KEY IS CLAVE-TAB-DEP
                         INDEXED BY INDICE-DEPENDENCIA.
      05 CLAVE-TAB-DEP   PIC X(03).
      05 FILLER          PIC X(02).
      05 NOMBRE-TAB-DEP  PIC X(50).
      05 FILLER          PIC X(25).

01 C-REPORT.
   03 FILLER     PIC X(10) VALUE 'CECAFI    '.
   03 FILLER     PIC X(20) VALUE SPACES.
   03 FILLER     PIC X(30) VALUE 'REPORTE DE MOVIMIENTOS      '.
   03 FILLER     PIC X(40) VALUE SPACES.
   03 C-R-DD     PIC Z(02)/.
   03 C-R-MM     PIC Z(02)/.
   03 C-R-YY     PIC 9(02).
   03 FILLER     PIC X(06) VALUE SPACES.
   03 FILLER     PIC X(10) VALUE 'PAGINA = '.
   03 C-R-PAGE   PIC Z(03).
   03 FILLER     PIC X(10) VALUE SPACES.

01 C-PAGE        PIC 9(10) VALUE ZEROES.

01 C-LINE        PIC 9(02) VALUE 60.
   88 C-LINE-OK VALUE 60.

01 C-DATE.
   03 C-YY    PIC 9(02).
   03 C-MM    PIC 9(02).
   03 C-DD    PIC 9(02).

01 S-NORMAL.
   03 FILLER     PIC X(10) VALUE 'OK>       '.
   03 FILLER     PIC X(10) VALUE '    CLAVE '.
   03 S-N-CLAVE  PIC X(03) VALUE SPACES.
   03 FILLER     PIC X(07) VALUE SPACES.
   03 FILLER     PIC X(13) VALUE 'DEPENDENCIA  '.
   03 S-N-NOMBRE PIC X(50) VALUE SPACES.
   03 FILLER     PIC X(37) VALUE SPACES.

01 S-ERROR.
   03 FILLER     PIC X(10) VALUE 'ERROR>    '.
   03 FILLER     PIC X(10) VALUE '    CLAVE '.
   03 S-E-CLAVE  PIC X(03) VALUE SPACES.
   03 FILLER     PIC X(07) VALUE SPACES.
   03 FILLER     PIC X(13) VALUE 'DEPENDENCIA  '.
   03 S-E-NOMBRE PIC X(50) VALUE SPACES.
   03 FILLER     PIC X(10) VALUE SPACES.
   03 FILLER     PIC X(20) VALUE 'LLAVE DUPLICADA     '.
   03 FILLER     PIC X(17) VALUE SPACES.

PROCEDURE DIVISION.
*****************************************************************************

PROGRAMA-PRINCIPAL.

    PERFORM INIT-DATE.
    PERFORM ABRE-ARCHIVOS.
    PERFORM LEE-SECUENCIAL UNTIL NO-HAY-DATOS-SEC.
    PERFORM CIERRA-ARCHIVOS.
    STOP RUN.
```

```
INIT-DATE.

    ACCEPT C-DATE FROM DATE.
    MOVE C-DD TO C-R-DD.
    MOVE C-MM TO C-R-MM.
    MOVE C-YY TO C-R-YY.

ABRE-ARCHIVOS.

    OPEN INPUT SECUENCIAL.
    OPEN OUTPUT INDEXADO.
    OPEN OUTPUT REPORTE.
    SET INDICE-DEPENDENCIA TO 1.

LEE-SECUENCIAL.

    READ SECUENCIAL AT END MOVE "N" TO HAY-DATOS-SEC.
    IF NOT NO-HAY-DATOS-SEC THEN
      PERFORM ASSIGNA-TABLA
      PERFORM GRABA-INDEXADO.
*   ENDIF

ASSIGNA-TABLA.

      MOVE R-S-CLAVE  TO CLAVE-TAB-DEP(INDICE-DEPENDENCIA)
      MOVE R-S-NOMBRE TO NOMBRE-TAB-DEP(INDICE-DEPENDENCIA)
      SET INDICE-DEPENDENCIA UP BY 1.

GRABA-INDEXADO.

      MOVE R-S-CLAVE  TO R-I-CLAVE.
      MOVE R-S-NOMBRE TO R-I-NOMBRE.
      WRITE R-INDEXADO INVALID KEY MOVE "S" TO ERROR-INDEXADO.
      PERFORM ANS-ERROR-INDEXADO.

ANS-ERROR-INDEXADO.

      IF ERROR-INDEXADO-OK THEN
         MOVE "N" TO ERROR-INDEXADO
         PERFORM ANS-HEAD
         MOVE R-S-CLAVE  TO S-E-CLAVE
         MOVE R-S-NOMBRE TO S-E-NOMBRE
         WRITE R-REPORT FROM S-ERROR
         ADD 1 TO C-LINE
      ELSE
         PERFORM ANS-HEAD
         MOVE R-S-CLAVE  TO S-N-CLAVE
         MOVE R-S-NOMBRE TO S-N-NOMBRE
         WRITE R-REPORT FROM S-NORMAL
         ADD 1 TO C-LINE.
*     ENDIF

ANS-HEAD.

    IF C-LINE-OK THEN
      MOVE SPACES TO R-REPORT
      WRITE R-REPORT AFTER ADVANCING PAGE
      ADD 1 TO C-PAGE
      MOVE C-PAGE TO C-R-PAGE
      WRITE R-REPORT FROM C-REPORT AFTER ADVANCING 2 LINES
      MOVE SPACES TO R-REPORT
      WRITE R-REPORT AFTER ADVANCING 2 LINES
```

```
           MOVE 4 TO C-LINE.
      *    ENDIF.

       CIERRA-ARCHIVOS.

           CLOSE SECUENCIAL.
           CLOSE INDEXADO.
           CLOSE REPORTE.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     CUENTA-Y-MUESTRA-MB.
AUTHOR.         JORGE VALERIO.
INSTALLATION.   CECAFI.
DATE-WRITTEN.   25-OCT-84.
DATE-COMPILED.
SECURITY.       PROGRAMA QUE CUENTA CUANTOS
                ALUMNOS TIENEN MB. LOS
                REGISTROS SON INTRODUCIDOS
                EN UN ARCHIVO.

*               REVISADO 24 DE ENERO DE 1985.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO ASSIGN 'DVALERIO.COBOL.DATOS]CAPALUMB.DAT'.

DATA DIVISION.
FILE SECTION.
FD ARCHIVO
    RECORD CONTAINS 42 CHARACTERS
    LABEL RECORD IS STANDARD
    DATA RECORD IS ARCH-REG.
01 ARCH-REG.
    03 MATERIA              PIC X(03).
    03 FILLER               PIC X(01).
    03 NOMBRE               PIC X(25).
    03 FILLER               PIC X(01).
    03 NUMERO-DE-CUENTA     PIC X(08).
    03 FILLER               PIC X(01).
    03 CALIFICACION         PIC X(02).
    03 FILLER               PIC X(01).
WORKING-STORAGE SECTION.
77 CONTADOR-MB              PIC 9(03) VALUE ZEROES.
77 HAY-DATOS                PIC X(01) VALUE 'S'.
01 REGISTRO.
    03 MATERIA              PIC X(03).
    03 FILLER               PIC X(01).
    03 NOMBRE               PIC X(25).
    03 FILLER               PIC X(01).
    03 NUMERO-DE-CUENTA     PIC X(08).
    03 FILLER               PIC X(01).
    03 CALIFICACION         PIC X(02).
    03 FILLER               PIC X(01).
01 LISTA.
    03 FILLER               PIC X(12)  VALUE 'El alumno  '.
    03 NOMBRE-LISTA         PIC X(25).
    03 FILLER               PIC X(10)  VALUE ' tiene MB.'.
    03 FILLER               PIC X(33)  VALUE SPACES.
01 TITULO.
    03 FILLER               PIC X(80) VALUE '<<<<<<<<<Captura de datos>>>>>>>>>'.
01 RENGLON-1.                                                    Calificac
    03 FILLER               PIC X(80)  VALUE 'Mat. Nombre                    Numero
01 RENGLON-2.
    03 FILLER               PIC X(80)  VALUE '---------------------------------------

PROCEDURE DIVISION.
PROGRAMA-PRINCIPAL.
```

```
    OPEN OUTPUT ARCHIVO.
    PERFORM PROCESO-LISTA
            WITH TEST BEFORE
            UNTIL HAY-DATOS EQUAL "N".
    PERFORM SALIDA-FINAL.
    CLOSE ARCHIVO.
    STOP RUN.

PROCESO-LISTA.
    PERFORM CAPTURA.
    PERFORM EVALUA.
    PERFORM CONTINUA.

CAPTURA.
    DISPLAY TITULO.
    DISPLAY RENGLON-1.
    DISPLAY RENGLON-2.
    ACCEPT  REGISTRO.
    WRITE ARCH-REG FROM REGISTRO.

EVALUA.
    IF CALIFICACION IN REGISTRO IS EQUAL "MB" THEN
       MOVE NOMBRE IN REGISTRO TO NOMBRE-LISTA
       DISPLAY LISTA
       ADD 1 TO CONTADOR-MB.
*   NOELSE
*   ENDIF

CONTINUA.

    MOVE "H" TO HAY-DATOS.
    PERFORM PIDE
            UNTIL HAY-DATOS IS EQUAL "S" OR HAY-DATOS IS EQUAL "N".

PIDE.
    DISPLAY "Continuas introduciendo datos (S/N) ? " WITH NO ADVANCING.
    ACCEPT HAY-DATOS.

SALIDA-FINAL.
    DISPLAY "Total de alumnos con MB : " CONTADOR-MB.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRUEBA.
AUTHOR. GRUP013.
INSTALLATION. VAX 11-780.
DATE-WRITTEN. 29 OCTUBRE 1984.
DATE-COMPILED. 29 OCTUBRE 1984.
SECURITY. NINGUNA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATOS ASSIGN "NOMBRE.DAT"
    ORGANIZATION IS SEQUENTIAL.
    SELECT NOMYRFC ASSIGN "NOMYRFC.DAT"
    ORGANIZATION IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD DATOS
    RECORD CONTAINS 36 CHARACTERS
    DATA RECORD IS REGISTRO.
01 REGISTRO.
    03 NOMBRE PIC X(30).
    03 DIA PIC 99.
    03 MES PIC 99.
    03 ANIO PIC 99.
FD NOMYRFC
    RECORD CONTAINS 40 CHARACTERS
    DATA RECORD IS SALIDA.
01 SALIDA.
    03 NOMBRE.
       05 ACOMODADO OCCURS 30 TIMES PIC X(01).
    03 LETRAS PIC A(04).
    03 ANIO 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 ZERO.
77 PUNTO-Y-COMA-2 PIC 99 VALUE ZERO.
77 ENCONTRAR PIC X(02) VALUE "NO".
    88 YA-LA-ENCONTRE VALUE "SI".
77 I PIC 99 VALUE ZEROES.
77 J PIC 99 VALUE ZEROES.
77 LECTURA PIC X(02) VALUE "NO".
    88 FIN-DE-ARCHIVO 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.
PARRAFO-1.
PRINCIPAL.
    OPEN INPUT DATOS.
    OPEN OUTPUT NOMYRFC.
    PERFORM PROCESO UNTIL FIN-DE-ARCHIVO.
    CLOSE DATOS.
    CLOSE NOMYRFC.
    STOP RUN.
```

```
PROCESO.
    READ DATOS AT END MOVE "SI" TO LECTURA.
    MOVE ZEROES TO PUNTO-Y-COMA-1.
    MOVE ZEROES TO PUNTO-Y-COMA-2.
    MOVE ZEROES TO BLANCOS.
    MOVE ZEROES TO J.
    MOVE 1 TO I.
    MOVE "NO" TO ENCONTRAR
    IF NOT FIN-DE-ARCHIVO THEN
        PERFORM INSPECCIONA THRU IMPRIME
    ELSE
        NEXT SENTENCE.
*   ENDIF
INSPECCIONA.
    MOVE NOMBRE IN REGISTRO TO NOMBRES.
    MOVE INDICE(1) TO INDICE-LETRA(1).
    PERFORM ENCUENTRA-VOCAL UNTIL YA-LA-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 TO SALIDA.
    MOVE LETRAS-RFC TO LETRAS IN SALIDA.
    INSPECT NOMBRES TALLYING BLANCOS FOR CHARACTERS BEFORE INITIAL " ".
    ADD 2 TO BLANCOS
    IF BLANCOS >30 THEN
        MOVE 30 TO BLANCOS
    ELSE
        NEXT SENTENCE.
*   ENDIF
    PERFORM ACOMODA-NOMBRE VARYING I FROM PUNTO-Y-COMA-2 BY 1 UNTIL I = BLANC
    PERFORM ACOMODA-NOMBRE VARYING I FROM 1 BY 1 UNTIL I = PUNTO-Y-COMA-2.
    INSPECT NOMBRE IN SALIDA REPLACING ALL "%" BY " ".
IMPRIME.
    WRITE SALIDA.

ENCUENTRA-VOCAL.
    ADD 1 TO I.
    MOVE INDICE(I) TO VOCALES.
    IF VOCAL THEN
        MOVE "SI" TO ENCONTRAR
    ELSE
        NEXT SENTENCE.
*   ENDIF

ACOMODA-NOMBRE.
    ADD 1 TO J.
    MOVE INDICE(1) TO ACOMODADO(J).
```

```
Identification division.
Program-id.    Reporte-de-vinos.
Author.        Borrachos, S.A.
Installation.  Cecafi.
Security.      Programa que emite un reporte de la existencia de
               vinos en la bodesa de Don Gervancio determinando
               su precio y evaluando para hacer pedidos para
               completar existencia en bodesa.

Environment division.
Configuration section.
Source-computer. Vax-11-780.
Object-computer. Vax-11-780.
Input-output section.
File-control.
     Select vinos   assign to "vinos.dat".
     Select reporte assign to "reporte.lis".

Data division.
File section.
Fd   vinos
     label records are omitted
     record contains 63 characters
     data  record  is  vino.
01   vino.
     02 tipo-vino      picture is 9(01).
     02 calidad-vino   picture is 9(01).
     02 marca-vino     picture is X(20).
     02 proveedor-vino picture is x(30).
     02 costo-vino     picture is 9(05).
     02 bodesa-vino    picture is 9(06).
Fd   reporte
     label records are omitted
     record contains 132 characters
     data record   is  reporte-res.
01   reporte-res picture is x(132).
Working-storage section.
01   encabezado-1.
     02 filler picture is x(51) value spaces.
     02 filler picture is x(27) value all "*".
     02 filler picture is x(54) value spaces.
01   encabezado-2.
     02 filler picture is x(51) value spaces.
     02 filler picture is x(01) value "*".
     02 filler picture is x(25) value spaces.
     02 filler picture is x(01) value "*".
     02 filler picture is x(54) value spaces.
01   encabezado-3.
     02 filler picture is x(51) value spaces.
     02 filler picture is x(27) value "* Vinateria Don Gervancio *".
     02 filler picture is x(54) value spaces.
01   encabezado-4.
     02 filler picture is x(50) value spaces.
     02 filler picture is x(32) value "Reporte de existencias en bodesa".
     02 filler picture is x(50) value spaces.
01   encabezado-5.
     02 filler picture is x(50) value spaces.
     02 filler picture is x(32) value all "=".
     02 filler picture is x(50) value spaces.
01   encabezado-6.
     02 filler picture is x(03) value spaces.
     02 filler picture is x(04) value "tipo".
     02 filler picture is x(06) value spaces.
```

```
      02 filler picture is x(07) value 'calidad'.
      02 filler picture is x(12) value spaces.
      02 filler picture is x(05) value 'marca'.
      02 filler picture is x(23) value spaces.
      02 filler picture is x(09) value 'proveedor'.
      02 filler picture is x(16) value spaces.
      02 filler picture is x(05) value 'costo'.
      02 filler picture is x(04) value spaces.
      02 filler picture is x(08) value 'unidades'.
      02 filler picture is x(04) value spaces.
      02 filler picture is x(06) value 'precio'.
      02 filler picture is x(08) value spaces.
      02 filler picture is x(05) value 'notas'.
      02 filler picture is x(05) value spaces.
01    encabezado-7.
      02 filler picture is x(01) value spaces.
      02 filler picture is x(08) value all '-'.
      02 filler picture is x(05) value spaces.
      02 filler picture is x(08) value all '-'.
      02 filler picture is x(05) value spaces.
      02 filler picture is x(20) value all '-'.
      02 filler picture is x(05) value spaces.
      02 filler picture is x(30) value all '-'.
      02 filler picture is x(05) value spaces.
      02 filler picture is x(05) value all '-'.
      02 filler picture is x(04) value spaces.
      02 filler picture is x(08) value all '-'.
      02 filler picture is x(04) value spaces.
      02 filler picture is x(07) value all '-'.
      02 filler picture is x(04) value spaces.
      02 filler picture is x(12) value all '-'.
      02 filler picture is x(01) value spaces.
01    sal.
      02 filler          picture is x(01) value spaces.
      02 tipo-sal        picture is x(08).
      02 filler          picture is x(05) value spaces.
      02 calidad-sal     picture is x(08).
      02 filler          picture is x(05) value spaces.
      02 marca-sal       picture is x(20).
      02 filler          picture is x(05) value spaces.
      02 proveedor-sal   picture is x(30).
      02 filler          picture is x(05) value spaces.
      02 costo-sal       picture is z(05).
      02 filler          picture is x(05) value spaces.
      02 unidades-sal    picture is z(06).
      02 filler          picture is x(05) value spaces.
      02 precio-sal      picture is z(07).
      02 filler          picture is x(04) value spaces.
      02 nota-sal        picture is x(12).
01    fin-de-reporte.
      02 filler picture is x(50) value spaces.
      02 filler picture is x(32) value '******** Fin de reporte ********'.
      02 filler picture is x(50) value spaces.
01    EOF               picture is x(03) value 'no'.
01    existencias       picture is 9(09).

procedure division.
begin.
      Open input vinos.
      Open output reporte.
      Perform escribe-encabezado.
      Perform procesa-reporte until EOF is equal 'fin'.
      Perform escribe-fin.
```

```
      Close vinos.
      Close reporte.
      Stop run.

 Escribe-encabezado.
      Write reporte-res from encabezado-1 after  page.
      Write reporte-res from encabezado-2 before advancing 1 line.
      Write reporte-res from encabezado-3 before advancing 1 line.
      Write reporte-res from encabezado-2 before advancing 1 line.
      Write reporte-res from encabezado-1 before advancing 2 line.
      Write reporte-res from encabezado-4 before advancing 1 line.
      Write reporte-res from encabezado-5 before advancing 3 line.
      Write reporte-res from encabezado-6 before advancing 1 line.
      Write reporte-res from encabezado-7 before advancing 2 line.

 Escribe-fin.
      Write reporte-res from fin-de-reporte after advancing 2 line.

 Procesa-reporte.
      Perform lee-vino.
      If EOF not equal "fin" perform continua-reporte.

 Lee-vino.
      Read vinos record at end move "fin" to EOF.

 Continua-reporte.
      If tipo-vino equal 1 move "Tipico" to tipo-sal.
      If tipo-vino equal 2 move "Vino"   to tipo-sal.
      If tipo-vino equal 3 move "Brands" to tipo-sal.
      If tipo-vino equal 4 move "Otros"  to tipo-sal.
      If calidad-vino equal 1 move "Regular"  to calidad-sal.
      If calidad-vino equal 2 move "Bueno"    to calidad-sal.
      If calidad-vino equal 3 move "Exelente" to calidad-sal.
      Move marca-vino to marca-sal.
      Move proveedor-vino to proveedor-sal.
      Move costo-vino to costo-sal.
      Move bodega-vino to unidades-sal.
      Compute precio-sal rounded =  costo-vino
              + ( costo-vino * tipo-vino / 100 )
              + ( costo-vino * calidad-vino / 100 ) .
      Compute existencias rounded =  tipo-vino * calidad-vino * 1000.
      if bodega-vino is less than existencias move "Hacer pedido" to nota-sal
      Write reporte-res from sal before advancing 1 line.
```

```
 Identification division.
 Prosram-id.     Display-curso-file-allowins.
 Author.         Jorse Valerio.
 Installation.   Cecafi.
 Date-written.   Agosto 1985.
 Date-compiled.  Agosto 1985.
 Security.       Ninsuna.

 Environment division.
 Configuration section.
 Source-computer. Digital-Vax-11-780.
 Object-computer. Digital-Vax-11-780.
 Input-output section.
 File-control.
     Select formas assisn to "presunta.idx"
          organization is indexed
          access mode is sequential
          record key is llave-formas
          file status is formas-status.
     Select cuestionarios assisn to ""
          organization is indexed
          access mode is sequential
          record key is llave-cuestionarios
          file status is cuestionarios-status.

 Data division.
 File section.
 Fd  formas
     record contains 82 characters
     data record is resistro-formas.
 01  resistro-formas.
     05  llave-formas    pic x(03).
     05  nombre-archivo  pic x(13).
     05  presunta        pic 9(01).
         88 hace-presunta value 1.
     05  respuesta       pic 9(01).
     05  filler          pic x(64).
 Fd  cuestionarios
     record contains 82 characters
     value of id is nombre-cuestionarios
     data record is resistro-cuestionarios.
 01  resistro-cuestionarios.
     05 llave-cuestionarios       pic x(03).
     05 informacion-cuestionarios pic x(79).
 Workins-storase section.
 77  nombre-cuestionarios     pic x(40).
 77  fin-archivo-formas       pic x(02) value "no".
     88 fin-archivo-formas-ok value "si".
 77  fin-archivo              pic x(02) value "no".
     88 fin-archivo-ok value "sí".
 77  formas-status            pic x(02).
 77  cuestionarios-status     pic x(02).
 77  Contestacion             pic 9(01).
 77  lineas-impresas          pic 9(02) value zeroes.
     88 limite-lineas-impresas value 22.

 Procedure division.
 Inicio.
     perform ejercicios.
     Stop run.
 Ejercicios.
     Open input formas allowins readers.
     Perform read-formas.
```

```
    perform proceso-formas until fin-archivo-formas-ok.
read-formas.
    Read formas at end move "si" to fin-archivo-formas.
Proceso-formas.
    Move nombre-archivo to nombre-cuestionarios.
    Open input cuestionarios allowing readers.
    display "".
    Move zeroes to lineas-impresas.
    Perform lee-cuestionarios.
    Perform display-cuestionarios until fin-archivo-ok.
    Close cuestionarios.
    Move "no" to fin-archivo.
    If hace-pregunta
       perform pide-contestacion.
    perform pide-return.
    Perform read-formas.
Lee-cuestionarios.
    Move spaces to informacion-cuestionarios.
    Read cuestionarios next record at end move "si" to fin-archivo.
Display-cuestionarios.
    If limite-lineas-impresas
       Move zeroes to lineas-impresas
       Perform pide-return.
    Display informacion-cuestionarios.
    Add 1 to lineas-impresas.
    Perform lee-cuestionarios.
pide-contestacion.
    Perform lineas 1 times.
    Display "Cual es la pregunta correcta : " with no advancing.
    Accept contestacion.
    If contestacion not equal to respuesta
       Display "Respueta incorrecta. Opcion correcta : " respuesta
    else
       Display "Respuesta correcta".
    Perform lineas 2 times.
pide-return.
    Display "Return : " with no advancing.
    Accept contestacion.
    display "".
Lineas.
    Display " ".
```

```
Identification division.
Program-id.     Display-curso-file-allowins.
Author.         Jorge Valerio.
Installation.   Cecafi.
Date-written.   Agosto 1985.
Date-compiled.  Agosto 1985.
Security.       Ninguna.

Environment division.
Configuration section.
Source-computer. Digital-Vax-11-780.
Object-computer. Digital-Vax-11-780.
Input-output section.
File-control.
    Select formas assign to "Presunta.idx"
           organization is indexed
           access mode is sequential
           record key is llave-formas
           file status is formas-status.
    Select cuestionarios assign to ""
           organization is indexed
           access mode is sequential
           record key is llave-cuestionarios
           file status is cuestionarios-status.

Data division.
File section.
Fd  formas
    record contains 82 characters
    data record is registro-formas.
01  registro-formas.
    05  llave-formas    Pic x(03).
    05  nombre-archivo  Pic x(13).
    05  Presunta        Pic 9(01).
        88 hace-Presunta value 1.
    05  respuesta       Pic 9(01).
    05  filler          Pic x(64).
Fd  cuestionarios
    record contains 82 characters
    value of id is nombre-cuestionarios
    data record is registro-cuestionarios.
01  registro-cuestionarios.
    05 llave-cuestionarios       Pic x(03).
    05 informacion-cuestionarios Pic x(79).
Working-storage section.
77  nombre-cuestionarios    Pic x(40).
77  fin-archivo-formas      Pic x(02) value "no".
    88 fin-archivo-formas-ok value "si".
77  fin-archivo             Pic x(02) value "no".
    88 fin-archivo-ok value "si".
77  formas-status           Pic x(02).
77  cuestionarios-status    Pic x(02).
77  Contestacion            Pic 9(01).
77  Lineas-impresas         Pic 9(02) value zeroes.
    88 limite-lineas-impresas value 22.

Procedure division.
Inicio.
    Perform ejercicios.
    Stop run.
Ejercicios.
    Open input formas allowins readers.
    Perform read-formas.
```

```
    Perform proceso-formas until fin-archivo-formas-ok.
read-formas.
    Read formas at end move "si" to fin-archivo-formas.
Proceso-formas.
    Move nombre-archivo to nombre-cuestionarios.
    Open input cuestionarios allowing readers.
    display "".
    Move zeroes to lineas-impresas.
    Perform lee-cuestionarios.
    Perform display-cuestionarios until fin-archivo-ok.
    Close cuestionarios.
    Move "no" to fin-archivo.
    If hace-pregunta
       perform pide-contestacion.
    perform pide-return.
    Perform read-formas.
Lee-cuestionarios.
    Move spaces to informacion-cuestionarios.
    Read cuestionarios next record at end move "si" to fin-archivo.
Display-cuestionarios.
    If limite-lineas-impresas
       Move zeroes to lineas-impresas
       Perform pide-return.
    Display informacion-cuestionarios.
    Add 1 to lineas-impresas.
    Perform lee-cuestionarios.
pide-contestacion.
    Perform lineas 1 times.
    Display "Cual es la pregunta correcta : " with no advancing.
    Accept contestacion.
    If contestacion not equal to respuesta
       Display "Respueta incorrecta. Opcion correcta : " respuesta
    else
       Display "Respuesta correcta".
    Perform lineas 2 times.
pide-return.
    Display "Return : " with no advancing.
    Accept contestacion.
    display "".
Lineas.
    Display " ".
```

```
 Identification division.
 Program-id.     Indexa.
 Author.         Jorge Valerio.
 Installation.   Cecafi.
 Date-written.   Agosto 1985.
 Date-compiled.  Agosto 1985.
 Security.       Ninguna.

 Environment division.
 Configuration section.
 Source-computer. Digital-Vax-11-780.
 Object-computer. Digital-Vax-11-780.
 Input-output section.
 File-control.
     Select entrada assign to ""
          organization is sequential
          access mode is sequential
          file status is entrada-status.
     Select salida assign to ""
          organization is indexed
          access mode is random
          record key is llave-salida
          file status is salida-status.

 Data division.
 File section.
  Fd  entrada
      record contains 79 characters
      value of id is nombre-entrada
      data record is registro-entrada.
  01  registro-entrada.
      05 informacion-entrada pic x(79).
  Fd  salida
      record contains 82 characters
      value of id is nombre-salida
      data record is registro-salida.
  01  registro-salida.
      05 llave-salida       pic x(03).
      05 informacion-salida pic x(79).
 Working-storage section.
  77  numero-de-registro pic 9(03) value 001.
  77  nombre-entrada pic x(40).
  77  nombre-salida  pic x(40).
  77  fin-archivo    pic x(02) value "no".
      88 fin-archivo-ok value "si".
  77  entrada-status pic x(02).
  77  salida-status  pic x(02).

 Procedure division.
 Inicio.
     Accept nombre-entrada.
     Accept nombre-salida.
     Open input entrada.
     Open output salida allowing all.
     Perform lee-entrada.
     Perform escribe-salida until fin-archivo-ok.
     Close entrada.
     Close salida.
     Stop run.
 Lee-entrada.
     Move spaces to informacion-entrada.
     Read entrada at end move "si" to fin-archivo.
 Escribe-salida.
```

```
Move numero-de-registro to llave-salida.
Move spaces to informacion-salida.
Move informacion-entrada to informacion-salida.
Write registro-salida invalid key display 'Error de escritura'.
Add 1 to numero-de-registro.
Perform lee-entrada.
```

```
identification division.
*****************************************************************

program-id.  strins01.

*****************************************************************
*
* 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 strins para la transferencia de datos
*
*****************************************************************

environment division.
*****************************************************************

input-output section.
file-control.

    select name-master assign to 'namemst'.
    select print-file  assign to 'mstlst'.

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-ss            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(14) 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(09).
      05 filler       pic x(01) value '/'.
      05 h-dd         pic z(09).
      05 filler       pic x(01) value '/'.
      05 h-yy         pic z(09).
   03 filler          pic x(10) 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-set-terminal-input.

    display 'enter data in this format:'.
    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 n-nam-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-set-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-set-terminal-input.

999-eoj.

    close name-master.
    close print-file.
    display "end of program".
    stop run.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     PSEUDONOMINA.
 AUTHOR.         GRUPO13.
 INSTALLATION.   VAX-11/780.
 DATE-WRITTEN.   19-OCT-84.
 DATE-COMPILED.  19-OCT-84.
 SECURITY.       PRIVADO.
 *
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 *
 77 TOTAL-A-PAGAR    PIC 9(10)      VALUE ZEROES.
 77 PAGO-A           PIC 9(04)      VALUE 2000.
 77 PAGO-B           PIC 9(04)      VALUE 3000.
 77 PAGO-C           PIC 9(04)      VALUE 3500.
 77 CHECA-FILTRO     PIC X(01)      VALUE SPACES.
    88 ERROR-EN-DATO                VALUE "T".
 77 SALARIO          PIC 9(10)      VALUE ZEROES.
 01 LETRERO1.
    03 FILLER        PIC X(08)      VALUE SPACES.
    03 FILLER        PIC X(10)      VALUE "CLAVE IMSS".
    03 FILLER        PIC X(13)      VALUE SPACES.
    03 FILLER        PIC A(06)      VALUE "NOMBRE".
    03 FILLER        PIC X(23)      VALUE SPACES.
    03 FILLER        PIC A(07)      VALUE "SALARIO".
 01 LETRERO2.
    03 FILLER        PIC X(10)      VALUE SPACES.
    03 CLAVE-IMSS    PIC X(04).
    03 FILLER        PIC X(06)      VALUE SPACES.
    03 NOMBRE        PIC X(30).
    03 FILLER        PIC X(06)      VALUE SPACES.
    03 SALA          PIC $(10).00.

 01 LETRERO3.
    03 FILLER        PIC X(42)      VALUE SPACES.
    03 FILLER        PIC X(14)      VALUE 'TOTAL A PAGAR:'.
    03 PAGO-TOTAL    PIC $(07)9.99.

 01 REGISTRO.
    03 NOMBRE        PIC A(30)      VALUE ALL "*".
       88 CONTINUA                  VALUE SPACES.
    03 CLAVE-TRAB    PIC A(01)      VALUE SPACE.
       88 CLAVE-CORRECTA            VALUE "A" THRU "C".
       88 TIPO-A                    VALUE "A".
       88 TIPO-B                    VALUE "B".
       88 TIPO-C                    VALUE "C".
    03 CLAVE-IMSS    PIC X(04)      VALUE SPACES.
    03 HORAS-TRAB    PIC 9(02)      VALUE ZEROES.
    03 HORAS-EX      PIC 9(02)      VALUE ZEROES.
       88 MAS-DE-20                 VALUE 21 THRU 99.

 PROCEDURE DIVISION.

 PRINCIPAL.
     PERFORM LECTURA.
     PERFORM CALCULOS UNTIL CONTINUA.
     PERFORM IMPRESION.
     STOP RUN.
 *END-PRINCIPAL.


 LECTURA.
     DISPLAY "DATOS DEL TRABAJADOR:"
```

```
      DISPLAY "NOMBRE (30) CVE TRABAJADOR (1) CVE IMSS (4) HORAS"
              " NORMALES (2) HORAS EXTRA (2)"
      ACCEPT REGISTRO.
*END-LECTURA

CALCULOS.
      PERFORM FILTRA.
      IF ERROR-EN-DATO THEN
         DISPLAY "LA TARJETA DEL TRABAJADOR " NOMBRE IN REGISTRO " TIENE ERROR"
      ELSE
         PERFORM CALCULA-SALARIO
         PERFORM IMPRIME-RESULTADOS
         ADD SALARIO TO TOTAL-A-PAGAR.
*     ENDIF
      PERFORM LECTURA.
*END CALCULOS.


FILTRA.
      IF (NOT CLAVE-CORRECTA) OR (MAS-DE-20) THEN
         MOVE "T" TO CHECA-FILTRO
      ELSE
         MOVE "F" TO CHECA-FILTRO.
*     ENDIF
*END-FILTRA


CALCULA-SALARIO.
      IF TIPO-A THEN
         COMPUTE SALARIO = (PAGO-A*HORAS-TRAB)+(PAGO-A*HORAS-EX*1.5)
      ELSE
         IF TIPO-B THEN
            COMPUTE SALARIO = (PAGO-B*HORAS-TRAB)+(PAGO-B*HORAS-EX*1.5)
         ELSE
            COMPUTE SALARIO = (PAGO-C*HORAS-TRAB)+(PAGO-C*HORAS-EX*1.5).
*        ENDIF
*     ENDIF
*END CALCULA-SALARIO


IMPRIME-RESULTADOS.
      MOVE CORR REGISTRO TO LETRERO2.
      MOVE SALARIO TO SALA.
      DISPLAY LETRERO1.
      DISPLAY LETRERO2.
*END IMPRIME-RESULTADOS


IMPRESION.
      MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
      DISPLAY LETRERO3.
*END IMPRESION.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     PSEUDONOMINA.
AUTHOR.         GRUPO13.
INSTALLATION.   VAX-11/780.
DATE-WRITTEN.   19-OCT-84.
DATE-COMPILED.  19-OCT-84.
SECURITY.       PRIVADO.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
77 TOTAL-A-PAGAR     PIC 9(10)      VALUE ZEROES.
77 PAGO-A            PIC 9(04)      VALUE 2000.
77 PAGO-B            PIC 9(04)      VALUE 3000.
77 PAGO-C            PIC 9(04)      VALUE 3300.
77 CHECA-FILTRO      PIC X(01)      VALUE SPACES.
   88 ERROR-EN-DATO                 VALUE "1".
77 SALARIO           PIC 9(10)      VALUE ZEROES.
77 sigue             pic x(01)      value space.
01 LETRERO1.
   03 FILLER         PIC X(08)      VALUE SPACES.
   03 FILLER         PIC X(10)      VALUE "CLAVE IMSS".
   03 FILLER         PIC X(13)      VALUE SPACES.
   03 FILLER         PIC A(06)      VALUE "NOMBRE".
   03 FILLER         PIC X(25)      VALUE SPACES.
   03 FILLER         PIC A(07)      VALUE "SALARIO".
   03 FILLER         PIC X(04)      VALUE SPACES.
01 LETRERO11.
   03 FILLER         PIC X(08)      VALUE SPACES.
   03 FILLER         PIC X(10)      VALUE ALL "=".
   03 FILLER         PIC X(02)      VALUE SPACES.
   03 FILLER         PIC X(30)      VALUE ALL "=".
   03 FILLER         PIC X(10)      VALUE SPACES.
   03 FILLER         PIC X(10)      VALUE ALL "=".
   03 FILLER         PIC X(10)      VALUE SPACES.
01 LETRERO2.
   03 FILLER         PIC X(12)      VALUE SPACES.
   03 CLAVE-IMSS     PIC X(04).
   03 FILLER         PIC X(04)      VALUE SPACES.
   03 NOMBRE         PIC X(30).
   03 FILLER         PIC X(06)      VALUE SPACES.
   03 SALA           PIC $(10).00.
   03 FILLER         PIC X(68)      VALUE SPACES.
01 LETRERO3.
   03 FILLER         PIC X(42)      VALUE SPACES.
   03 FILLER         PIC X(14)      VALUE "TOTAL A PAGAR:".
   03 PAGO-TOTAL     PIC $(07)9.99.
01 REGISTRO.
   03 NOMBRE         PIC X(30)      VALUE ALL "*".
   03 CLAVE-TRAB     PIC X(01)      VALUE SPACE.
      88 CLAVE-CORRECTA              VALUE "A" THRU "C".
      88 TIPO-A                      VALUE "A".
      88 TIPO-B                      VALUE "B".
      88 TIPO-C                      VALUE "C".
   03 CLAVE-IMSS     PIC X(04)      VALUE SPACES.
   03 HORAS-TRAB     PIC 9(02)      VALUE ZEROES.
      88 NORMALES-OK                 VALUE 00 THRU 20.
   03 HORAS-EX       PIC 9(02)      VALUE ZEROES.
      88 EXTRAS-OK                   VALUE 00 THRU 20.

PROCEDURE DIVISION.
PRINCIPAL.
```

```
*PERFORM CAPTURA-UNTIL SIGUE = "N".
    PERFORM IMPRESION.
    STOP RUN.

CAPTURA.
    PERFORM LECTURA.
    PERFORM CALCULA-SALARIO
    PERFORM IMPRIME-RESULTADOS
    ADD SALARIO TO TOTAL-A-PAGAR.
    MOVE SPACE TO SIGUE.
    PERFORM siguiente UNTIL SIGUE = "S" OR SIGUE = "N".

LECTURA.
    MOVE SPACES TO REGISTRO.
    DISPLAY " ".
    DISPLAY "DATOS DEL TRABAJADOR".
    DISPLAY "====================".
    DISPLAY " ".
    DISPLAY "NOMBRE (30)".
    ACCEPT NOMBRE IN REGISTRO.
    PERFORM CLAVE-TRABAJADOR UNTIL CLAVE-CORRECTA.
    DISPLAY "CLAVE IMSS (4)".
    ACCEPT CLAVE-IMSS IN REGISTRO.
    PERFORM HORAS-NORMALES UNTIL NORMALES-OK.
    PERFORM HORAS-EXTRAS UNTIL EXTRAS-OK.

CLAVE-TRABAJADOR.
    DISPLAY "CLAVE TRABAJADOR (1)".
    ACCEPT CLAVE-TRAB IN REGISTRO.

HORAS-NORMALES.
    DISPLAY "HORAS TRABAJADAS (2)".
    ACCEPT HORAS-TRAB.

HORAS-EXTRAS.
    DISPLAY "HORAS EXTRA (2)".
    ACCEPT HORAS-EX.

siguiente.
    DISPLAY "Siguiente calculo (1)".
    accept sigue.

CALCULA-SALARIO.
    IF TIPO-A
      COMPUTE SALARIO = (PAGO-A*HORAS-TRAB)+(PAGO-A*HORAS-EX*1.5)
    ELSE
      IF TIPO-B
        COMPUTE SALARIO = (PAGO-B*HORAS-TRAB)+(PAGO-B*HORAS-EX*1.5)
      ELSE
        COMPUTE SALARIO = (PAGO-C*HORAS-TRAB)+(PAGO-C*HORAS-EX*1.5).

IMPRIME-RESULTADOS.
    MOVE CORR REGISTRO TO LETRERO2.
    MOVE SALARIO TO SALA.
    DISPLAY LETRERO1.
    DISPLAY LETRERO11.
    DISPLAY LETRERO2.

IMPRESION.
    MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
    DISPLAY LETRERO3.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     PSEUDONOMINA.
AUTHOR.         GRUPO13.
INSTALLATION.   VAX-11/780.
DATE-WRITTEN.   19-OCT-84.
DATE-COMPILED.  19-OCT-84.
SECURITY.       PRIVADO.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
77 TOTAL-A-PAGAR     PIC 9(10)       VALUE ZEROES.
77 PAGO-A            PIC 9(04)       VALUE 2000.
77 PAGO-B            PIC 9(04)       VALUE 3000.
77 PAGO-C            PIC 9(04)       VALUE 3500.
77 CHECA-FILTRO      PIC X(01)       VALUE SPACES.
   88 ERROR-EN-DATO                  VALUE "T".
77 SALARIO           PIC 9(10)       VALUE ZEROES.
77 sisue             Pic x(01)       value space.
01 LETRERO1.
   03 FILLER         PIC X(08)       VALUE SPACES.
   03 FILLER         PIC X(10)       VALUE "CLAVE IMSS".
   03 FILLER         PIC X(13)       VALUE SPACES.
   03 FILLER         PIC A(06)       VALUE "NOMBRE".
   03 FILLER         PIC X(25)       VALUE SPACES.
   03 FILLER         PIC A(07)       VALUE "SALARIO".
   03 FILLER         PIC X(04)       VALUE SPACES.
01 LETRERO11.
   03 FILLER         PIC X(08)       VALUE SPACES.
   03 FILLER         PIC X(10)       VALUE ALL "=".
   03 FILLER         PIC X(02)       VALUE SPACES.
   03 FILLER         PIC X(30)       VALUE ALL "=".
   03 FILLER         PIC X(10)       VALUE SPACES.
   03 FILLER         PIC X(10)       VALUE ALL "=".
   03 FILLER         PIC X(10)       VALUE SPACES.
01 LETRERO2.
   03 FILLER         PIC X(12)       VALUE SPACES.
   03 CLAVE-IMSS     PIC X(04).
   03 FILLER         PIC X(04)       VALUE SPACES.
   03 NOMBRE         PIC X(30).
   03 FILLER         PIC X(06)       VALUE SPACES.
   03 SALA           PIC $(10).00.
   03 FILLER         PIC X(48)       VALUE SPACES.
01 LETRERO3.
   03 FILLER         PIC X(42)       VALUE SPACES.
   03 FILLER         PIC X(14)       VALUE "TOTAL A PAGAR:".
   03 PAGO-TOTAL     PIC $(07)9.99.
01 REGISTRO.
   03 NOMBRE         PIC X(30)       VALUE ALL "*".
   03 CLAVE-TRAB     PIC X(01)       VALUE SPACE.
      88 CLAVE-CORRECTA              VALUE "A" THRU "C".
      88 TIPO-A                      VALUE "A".
      88 TIPO-B                      VALUE "B".
      88 TIPO-C                      VALUE "C".
   03 CLAVE-IMSS     PIC X(04)       VALUE SPACES.
   03 HORAS-TRAB     PIC 9(02)       VALUE ZEROES.
   03 HORAS-EX       PIC 9(02)       VALUE ZEROES.
      88 MAS-DE-20                   VALUE 21 THRU 99.

PROCEDURE DIVISION.
PRINCIPAL.
    PERFORM CALCULOS UNTIL sisue = "N".
```

```
    PERFORM IMPRESION.
    STOP RUN.

LECTURA.
    DISPLAY " ".
    DISPLAY "DATOS DEL TRABAJADOR".
    DISPLAY "====================".
    DISPLAY " ".
    DISPLAY "NOMBRE (30)".
    ACCEPT NOMBRE IN REGISTRO.
    DISPLAY "CLAVE TRABAJADOR (1)".
    ACCEPT CLAVE-TRAB IN REGISTRO.
    DISPLAY "CLAVE IMSS (4)".
    ACCEPT CLAVE-IMSS IN REGISTRO.
    DISPLAY "HORAS TRABAJADAS (2)".
    ACCEPT HORAS-TRAB.
    DISPLAY "HORAS EXTRA (2)".
    ACCEPT HORAS-EX.

CALCULOS.
    PERFORM LECTURA.
    PERFORM FILTRA.
    IF ERROR-EN-DATO
       DISPLAY "LA TARJETA DEL TRABAJADOR : " NOMBRE IN REGISTRO " TIENE ERROR
    ELSE
       PERFORM CALCULA-SALARIO
       PERFORM IMPRIME-RESULTADOS
       ADD SALARIO TO TOTAL-A-PAGAR.
    MOVE "T" TO SIGUE.
    PERFORM sisuiente UNTIL SIGUE = "S" OR SIGUE = "N".

sisuiente.
    DISPLAY "Siguiente calculo (1)".
    accept sisue.

FILTRA.
    IF (NOT CLAVE-CORRECTA) OR (MAS-DE-20)
       MOVE "T" TO CHECA-FILTRO
    ELSE
       MOVE "F" TO CHECA-FILTRO.

CALCULA-SALARIO.
    IF TIPO-A
       COMPUTE SALARIO = (PAGO-A*HORAS-TRAB)+(PAGO-A*HORAS-EX*1.5)
    ELSE
       IF TIPO-B
          COMPUTE SALARIO = (PAGO-B*HORAS-TRAB)+(PAGO-B*HORAS-EX*1.5)
       ELSE
          COMPUTE SALARIO = (PAGO-C*HORAS-TRAB)+(PAGO-C*HORAS-EX*1.5).

IMPRIME-RESULTADOS.
    MOVE CORR REGISTRO TO LETRERO2.
    MOVE SALARIO TO SALA.
    DISPLAY LETRERO1.
    DISPLAY LETRERO11.
    DISPLAY LETRERO2.

IMPRESION.
    MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
    DISPLAY LETRERO3.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     PSEUDONOMINA.
AUTHOR.         GRUPO13.
INSTALLATION.   VAX-11/780.
DATE-WRITTEN.   19-OCT-84.
DATE-COMPILED.  19-OCT-84.
SECURITY.       PRIVADO.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT NOMINA ASSIGN TO "NOMINA.DAT;1".

DATA DIVISION.
FILE SECTION.
FD NOMINA
    RECORD CONTAINS 512 CHARACTERS
    DATA RECORD IS NOMINA-REGISTRO.
01 NOMINA-REGISTRO.
    03 NOMBRE          PIC X(30).
    03 CLAVE-TRAB      PIC X(01).
    03 CLAVE-IMSS      PIC X(04).
    03 HORAS-TRAB      PIC 9(02).
    03 HORAS-EX        PIC 9(02).
    03 FILLER          PIC X(473).
WORKING-STORAGE SECTION.
77 TOTAL-A-PAGAR   PIC 9(10)      VALUE ZEROES.
77 PAGO-A          PIC 9(04)      VALUE 2000.
77 PAGO-B          PIC 9(04)      VALUE 3000.
77 PAGO-C          PIC 9(04)      VALUE 3500.
77 SALARIO         PIC 9(10)      VALUE ZEROES.
77 SIGUE           PIC X(01)      value SPACE.
77 FIN-DE-ARCHIVO  PIC X(02)      VALUE "NO".
    88 FIN-DE-ARCHIVO-OK          VALUE "SI".

01 LETRERO1.
    03 FILLER      PIC X(08)      VALUE SPACES.
    03 FILLER      PIC X(10)      VALUE "CLAVE IMSS".
    03 FILLER      PIC X(13)      VALUE SPACES.
    03 FILLER      PIC A(06)      VALUE "NOMBRE".
    03 FILLER      PIC X(25)      VALUE SPACES.
    03 FILLER      PIC A(07)      VALUE "SALARIO".
    03 FILLER      PIC X(04)      VALUE SPACES.
01 LETRERO1A.
    03 FILLER      PIC X(08)      VALUE SPACES.
    03 FILLER      PIC X(10)      VALUE ALL "=".
    03 FILLER      PIC X(02)      VALUE SPACES.
    03 FILLER      PIC X(30)      VALUE ALL " ".
    03 FILLER      PIC X(07)      VALUE SPACES.
    03 FILLER      PIC X(13)      VALUE ALL "=".
    03 FILLER      PIC X(10)      VALUE SPACES.
01 LETRERO2.
    03 FILLER          PIC X(12)      VALUE SPACES.
    03 CLAVE-IMSS      PIC X(04).
    03 FILLER          PIC X(04)      VALUE SPACES.
    03 NOMBRE          PIC X(30).
    03 FILLER          PIC X(07)      VALUE SPACES.
    03 SALARIO-SALIDA  PIC $(10).00.
    03 FILLER          PIC X(09)      VALUE SPACES.
01 LETRERO3.
```

```
                                          VALUE SPACES.
   03 FILLER           PIC X(14)         VALUE "TOTAL A PAGAR:".
   03 PAGO-TOTAL       PIC $(07)9.99.
01 REGISTRO.
   03 NOMBRE           PIC X(30)         VALUE ALL "*".
   03 CLAVE-TRAB       PIC X(01).        VALUE SPACE.
      88 CLAVE-CORRECTA                  VALUE "A" THRU "C".
      88 TIPO-A                          VALUE "A".
      88 TIPO-B                          VALUE "B".
      88 TIPO-C                          VALUE "C".
   03 CLAVE-IMSS       PIC X(04)         VALUE SPACES.
   03 HORAS-TRAB       PIC 9(02)         VALUE ZEROES.
      88 NORMALES-OK                     VALUE 00 THRU 40.
   03 HORAS-EX         PIC 9(02)         VALUE ZEROES.
      88 EXTRAS-OK                       VALUE 00 THRU 20.

PROCEDURE DIVISION.
PRINCIPAL.
    PERFORM CAPTURA.
    PERFORM PROCESA-NOMINA.
    STOP RUN.

CAPTURA.
    OPEN OUTPUT NOMINA ALLOWING ALL.
    PERFORM CAPTURA-INFORMACION UNTIL sigue = "N".
    CLOSE NOMINA.

CAPTURA-INFORMACION.
    PERFORM LECTURA.
    MOVE CORRESPONDING REGISTRO TO NOMINA-REGISTRO.
    WRITE NOMINA-REGISTRO.
    MOVE SPACE TO SIGUE.
    PERFORM SIGUIENTE UNTIL SIGUE = "S" OR SIGUE = "N".

LECTURA.
    MOVE SPACES TO REGISTRO.
    DISPLAY " ".
    DISPLAY "DATOS DEL TRABAJADOR".
    DISPLAY "====================".
    DISPLAY " ".
    DISPLAY "NOMBRE (30)".
    ACCEPT NOMBRE IN REGISTRO.
    PERFORM CLAVE-TRABAJADOR UNTIL CLAVE-CORRECTA.
    DISPLAY "CLAVE IMSS (4)".
    ACCEPT CLAVE-IMSS IN REGISTRO.
    PERFORM HORAS-NORMALES UNTIL NORMALES-OK.
    PERFORM HORAS-EXTRAS UNTIL EXTRAS-OK.

CLAVE-TRABAJADOR.
    DISPLAY "CLAVE TRABAJADOR (1) < A B C >".
    ACCEPT CLAVE-TRAB IN REGISTRO.

HORAS-NORMALES.
    DISPLAY "HORAS TRABAJADAS (2) < 00 .. 40 >".
    ACCEPT HORAS-TRAB IN REGISTRO.

HORAS-EXTRAS.
    DISPLAY "HORAS EXTRA (2) < 00 .. 20 >".
    ACCEPT HORAS-EX IN REGISTRO.

siguiente.
    DISPLAY " ".
    DISPLAY "Siguiente calculo (1)".
```

```
    accept sigue.

PROCESA-NOMINA.

    OPEN INPUT NOMINA.
    PERFORM ENCABEZADO.
    PERFORM LEE-NOMINA.
    PERFORM PROCESO-NOMINA UNTIL FIN-DE-ARCHIVO-OK.
    PERFORM IMPRESION-TOTAL.
    CLOSE NOMINA.

ENCABEZADO.
    DISPLAY " ".
    DISPLAY " ".
    DISPLAY " ".
    DISPLAY "NOMINITA Y ASOCIADOS, S.A."
    DISPLAY "CECAFI LAB."
    DISPLAY " ".
    DISPLAY " ".
    DISPLAY LETRERO1.
    DISPLAY LETRERO11.

LEE-NOMINA.
    READ NOMINA INTO REGISTRO AT END MOVE "SI" TO FIN-DE-ARCHIVO.

PROCESO-NOMINA.
    PERFORM CALCULA-SALARIO
    PERFORM IMPRIME-RESULTADOS
    ADD SALARIO TO TOTAL-A-PAGAR.
    PERFORM LEE-NOMINA.

CALCULA-SALARIO.
    IF TIPO-A
       COMPUTE SALARIO = ( PAGO-A * HORAS-TRAB IN NOMINA-REGISTRO ) +
                         ( PAGO-A * HORAS-EX IN NOMINA-REGISTRO * 1.5 )
    ELSE
       IF TIPO-B
          COMPUTE SALARIO = ( PAGO-B * HORAS-TRAB IN NOMINA-REGISTRO ) +
                            ( PAGO-B * HORAS-EX IN NOMINA-REGISTRO * 1.5 )
       ELSE
          COMPUTE SALARIO = ( PAGO-C * HORAS-TRAB IN NOMINA-REGISTRO ) +
                            ( PAGO-C * HORAS-EX IN NOMINA-REGISTRO * 1.5 ).

IMPRIME-RESULTADOS.
    MOVE CORRESPONDING NOMINA-REGISTRO TO LETRERO2.
    MOVE SALARIO TO SALARIO-SALIDA.
    DISPLAY LETRERO2.

IMPRESION-TOTAL.
    MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
    DISPLAY " ".
    DISPLAY LETRERO3.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     MORE.
AUTHOR.         Jorge Valerio.
INSTALLATION.   VAX-11/780.
DATE-WRITTEN.   19-OCT-84.
DATE-COMPILED.  19-OCT-84.
SECURITY.       PRIVADO.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11-780.
OBJECT-COMPUTER. VAX-11-780.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO ASSIGN TO "".

DATA DIVISION.
FILE SECTION.
FD ARCHIVO
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS REGISTRO
    VALUE OF ID IS NOMBRE.
01 REGISTRO pic x(80).
WORKING-STORAGE SECTION.
77 NOMBRE PIC X(10).
77 LINEAS PIC 9(02) VALUE 00.
    88 linea-maxima value 23.
77 marca-de-archivo pic x(02) value "no".
    88 fin-de-archivo value "si".

PROCEDURE DIVISION.
INICIO.
    DISPLAY "More ! Nombre de archivo : " with no advancing.
    accept nombre.
    OPEN INPUT archivo.
    perform lee-archivo.
    Perform despliega until fin-de-archivo.
    close archivo.
    stop run.

lee-archivo.
    move spaces to registro.
    read archivo at end move "si" to marca-de-archivo.

despliega.
    if linea-maxima
        perform blanco 50 times
        display "Teclee return : " with no advancing
        accept nombre
        display " "
        display registro
        move 1 to lineas
    else
        display registro
        add 1 to lineas.
    perform lee-archivo.

blanco.
    display " " with no advancing.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     PSEUDONOMINA.
AUTHOR.         GRUPO13.
INSTALLATION.   VAX-11/780.
DATE-WRITTEN.   19-OCT-84.
DATE-COMPILED.  19-OCT-84.
SECURITY.       PRIVADO.
*
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO-ENTRADA ASSIGN "DATOS.DAT".
    SELECT ARCHIVO-SALIDA ASSIGN "RESULT.LIS".
    SELECT ARCHIVO-ERRORES ASSIGN "ERRORES.LIS".
*
DATA DIVISION.
FILE SECTION.
FD ARCHIVO-ENTRADA
    DATA RECORD IS REGISTRO
    RECORD CONTAINS 39 CHARACTERS.
01 REGISTRO.
    03 NOMBRE          PIC A(30).
       88 CONTINUA                    VALUE SPACES.
    03 CLAVE-TRAB      PIC A(01).
       88 CLAVE-CORRECTA              VALUE "A" THRU "C".
       88 TIPO-A                      VALUE "A".
       88 TIPO-B                      VALUE "B".
       88 TIPO-C                      VALUE "C".
    03 CLAVE-IMSS      PIC X(04).
    03 HORAS-TRAB      PIC 9(02).
    03 HORAS-EX        PIC 9(02).
       88 MAS-DE-20                   VALUE 21 THRU 99.

FD ARCHIVO-SALIDA
    DATA RECORD IS REGISTRO-SALIDA
    RECORD CONTAINS 132 CHARACTERS.
01 REGISTRO-SALIDA PIC X(132).

FD ARCHIVO-ERRORES
    DATA RECORD IS REGISTRO-ERRORES
    RECORD CONTAINS 132 CHARACTERS.
01 REGISTRO-ERRORES PIC X(132).
WORKING-STORAGE SECTION.
*
77 TOTAL-A-PAGAR   PIC 9(10)         VALUE ZEROES.
77 PAGO-A          PIC 9(04)         VALUE 2000.
77 PAGO-B          PIC 9(04)         VALUE 3000.
77 PAGO-C          PIC 9(04)         VALUE 3500.
77 CHECA-FILTRO    PIC X(01)         VALUE SPACES.
   88 ERROR-EN-DATO                  VALUE "T".
77 SALARIO         PIC 9(10)         VALUE ZEROES.
77 ESCRIBI PIC X(02) VALUE "NO".
   88 YA-ESCRIBIO VALUE "YA".
01 LETRERO1.
   03 FILLER       PIC X(27)         VALUE SPACES.
   03 FILLER       PIC X(10)         VALUE "CLAVE IMSS".
   03 FILLER       PIC X(22)         VALUE SPACES.
   03 FILLER       PIC A(06)         VALUE "NOMBRE".
   03 FILLER       PIC X(22)         VALUE SPACES.
   03 FILLER       PIC A(33)         VALUE "SALARIO".
01 LETRERO2.
   03 FILLER       PIC X(30)         VALUE SPACES.
   03 CLAVE-IMSS   PIC X(04).
```

```
    03 FILLER          PIC X(14)        VALUE SPACES.
    03 NOMBRE          PIC A(30).
    03 FILLER          PIC X(06)        VALUE SPACES.
    03 SALA            PIC $(10).00.
    03 FILLER          PIC X(37)        VALUE SPACES.

01 LETRERO3.
    03 FILLER          PIC X(68)        VALUE SPACES.
    03 FILLER          PIC X(14)        VALUE 'TOTAL A PAGAR '.
    03 FILLER          PIC X(02)        VALUE SPACES.
    03 PAGO-TOTAL      PIC $(07)9.99.
    03 FILLER          PIC X(37)        VALUE SPACES.

01 LETRERO4.
    03 FILLER          PIC X(32)        VALUE SPACES.
    03 FILLER          PIC X(26)        VALUE 'LA TARJETA DEL TRABAJADOR '.
    03 NOMBRE          PIC A(30).
    03 FILLER          PIC X(12)        VALUE 'TIENE ERROR'.
    03 FILLER          PIC X(32)        VALUE SPACES.

PROCEDURE DIVISION.

PRINCIPAL.
    OPEN INPUT ARCHIVO-ENTRADA OUTPUT ARCHIVO-SALIDA ARCHIVO-ERRORES.
    PERFORM LECTURA.
    PERFORM CALCULOS UNTIL CONTINUA.
    PERFORM IMPRESION.
    CLOSE ARCHIVO-ENTRADA ARCHIVO-SALIDA ARCHIVO-ERRORES.
    STOP RUN.
*END-PRINCIPAL.

LECTURA.
    READ ARCHIVO-ENTRADA AT END MOVE SPACES TO NOMBRE IN REGISTRO.
*END-LECTURA.

CALCULOS.
    PERFORM FILTRA.
    IF ERROR-EN-DATO THEN
        MOVE NOMBRE IN REGISTRO TO NOMBRE IN LETRERO4
        WRITE REGISTRO-ERRORES FROM LETRERO4 AFTER 2 LINES
    ELSE
        PERFORM CALCULA-SALARIO
        PERFORM IMPRIME-RESULTADOS
        ADD SALARIO TO TOTAL-A-PAGAR.
*   ENDIF
    PERFORM LECTURA.
*END-CALCULOS.

FILTRA.
    IF (NOT CLAVE-CORRECTA) OR (MAS-DE-20) THEN
        MOVE 'I' TO CHECA-FILTRO
    ELSE
        MOVE 'F' TO CHECA-FILTRO.
*   ENDIF
*END-FILTRA.

CALCULA-SALARIO.
    IF TIPO-A THEN
        COMPUTE SALARIO = (PAGO-A*HORAS-TRAB)+(PAGO-A*HORAS-EX*1.5)
    ELSE
```

```
            IF TIPO-B THEN
                COMPUTE SALARIO = (PAGO-B*HORAS-TRAB)+(PAGO-B*HORAS-EXT*1.5)
            ELSE
                COMPUTE SALARIO = (PAGO-C*HORAS-TRAB)+(PAGO-C*HORAS-EXT*1.5).
*           ENDIF.
*       ENDIF.
*END CALCULA-SALARIO


IMPRIME-RESULTADOS.
        MOVE CORR REGISTRO TO LETRERO2.
        MOVE SALARIO TO SALA.
        IF NOT YA-ESCRIBIO THEN
            WRITE REGISTRO-SALIDA FROM LETRERO1
            MOVE "YA" TO ESCRIBI
        ELSE
            NEXT SENTENCE.
*       ENDIF.
        WRITE REGISTRO-SALIDA FROM LETRERO2 AFTER 2 LINES.
*END IMPRIME-RESULTADOS


IMPRESION.
        MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
        WRITE REGISTRO-SALIDA FROM LETRERO3 AFTER 2 LINES.
*END IMPRESION.
```

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     PSEUDONOMINA.
 AUTHOR.         GRUPO13.
 INSTALLATION.   VAX-11/780.
 DATE-WRITTEN.   19-OCT-84.
 DATE-COMPILED.  19-OCT-84.
 SECURITY.       PRIVADO.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
 77 TOTAL-A-PAGAR    PIC 9(10)      VALUE ZEROES.
 77 PAGO-A           PIC 9(04)      VALUE 2000.
 77 PAGO-B           PIC 9(04)      VALUE 3000.
 77 PAGO-C           PIC 9(04)      VALUE 3500.
 77 CHECA-FILTRO     PIC X(01)      VALUE SPACES.
    88 ERROR-EN-DATO                VALUE "I".
 77 SALARIO          PIC 9(10)      VALUE ZEROES.
 01 LETRERO1.
    03 FILLER        PIC X(08)      VALUE SPACES.
    03 FILLER        PIC X(10)      VALUE "CLAVE IMSS".
    03 FILLER        PIC X(13)      VALUE SPACES.
    03 FILLER        PIC A(06)      VALUE "NOMBRE".
    03 FILLER        PIC X(23)      VALUE SPACES.
    03 FILLER        PIC A(07)      VALUE "SALARIO".
    03 FILLER        PIC X(6)     VALUE SPACES.
 01 LETRERO2.
    03 FILLER        PIC X(10)      VALUE SPACES.
    03 CLAVE-IMSS    PIC X(04).
    03 FILLER        PIC X(06)      VALUE SPACES.
    03 NOMBRE        PIC X(30).
    03 FILLER        PIC X(06)      VALUE SPACES.
    03 SALA          PIC $(10).00.
    03 FILLER        PIC X(68)      VALUE SPACES.

 01 LETRERO3.
    03 FILLER        PIC X(42)      VALUE SPACES.
    03 FILLER        PIC X(14)      VALUE "TOTAL A PAGAR:".
    03 PAGO-TOTAL    PIC $(07)9.99.

 01 REGISTRO.
    03 NOMBRE        PIC X(30)      VALUE ALL "*".
       88 CONTINUA                  VALUE SPACES.
    03 CLAVE-TRAB    PIC X(01)      VALUE SPACE.
       88 CLAVE-CORRECTA            VALUE "A" THRU "C".
       88 TIPO-A                    VALUE "A".
       88 TIPO-B                    VALUE "B".
       88 TIPO-C                    VALUE "C".
    03 CLAVE-IMSS    PIC X(04)      VALUE SPACES.
    03 HORAS-TRAB    PIC 9(02)      VALUE ZEROES.
    03 HORAS-EX      PIC 9(02)      VALUE ZEROES.
       88 MAS-DE-20                 VALUE 21 THRU 99.

 PROCEDURE DIVISION.

 PRINCIPAL.
     PERFORM LECTURA.
     PERFORM CALCULOS UNTIL CONTINUA.
     PERFORM IMPRESION.
     STOP RUN.
*END-PRINCIPAL
```

```
LECTURA.
    DISPLAY "DATOS DEL TRABAJADOR:".
    DISPLAY "NOMBRE (30) CLAVE TRABAJADOR (1) CLAVE IMSS (4) HORAS"
-           " TRABAJADAS (2) HORAS EXTRA (2)"
    ACCEPT REGISTRO.
*END-LECTURA

CALCULOS.
    PERFORM FILTRA.
    IF ERROR-EN-DATO THEN
       DISPLAY "LA TARJETA DEL TRABAJADOR" NOMBRE IN REGISTRO "TIENE ERROR"
    ELSE
       PERFORM CALCULA-SALARIO
       PERFORM IMPRIME-RESULTADOS
       ADD SALARIO TO TOTAL-A-PAGAR.
*   ENDIF
    PERFORM LECTURA.
*END CALCULOS


FILTRA.
    IF (NOT CLAVE-CORRECTA) OR (MAS-DE-20) THEN
       MOVE "T" TO CHECA-FILTRO
    ELSE
       MOVE "F" TO CHECA-FILTRO.
*   ENDIF
*END-FILTRA


CALCULA-SALARIO.
    IF TIPO-A THEN
       COMPUTE SALARIO = (PAGO-A*HORAS-TRAB)+(PAGO-A*HORAS-EX*1.5)
    ELSE
       IF TIPO-B THEN
          COMPUTE SALARIO = (PAGO-B*HORAS-TRAB)+(PAGO-B*HORAS-EX*1.5)
       ELSE
          COMPUTE SALARIO = (PAGO-C*HORAS-TRAB)+(PAGO-C*HORAS-EX*1.5).
*      ENDIF
*   ENDIF
*END CALCULA-SALARIO


IMPRIME-RESULTADOS.
    MOVE CORR REGISTRO TO LETRERO2.
    MOVE SALARIO TO SALA.
    DISPLAY LETRERO1.
    DISPLAY LETRERO2.
*END IMPRIME-RESULTADOS


IMPRESION.
    MOVE TOTAL-A-PAGAR TO PAGO-TOTAL.
    DISPLAY LETRERO3.
*END IMPRESION.
```

```
identification division.
*************************************************************************

program-id.    strings01.

*************************************************************************
*
* 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 tramsferencia de datos
*
*************************************************************************

environment division.
*************************************************************************

input-output section.
file-control.

    select name-master assign to "EVALERIO.COBOL.DATOS(namemst.dat)",
    select print-file  assign to "EVALERIO.COBOL.REPORTES(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) max of 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-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.
```

```
 program-id. commind.
 environment division.
 input-output section.
 file-control.
     select entrada assign to disk.
     select salida  assign to disk.

 data division.
 ******************************************************************************

 file section.

 fd   entrada
      record contains 80 characters
      value of id "entrada.dat"
      label records are standard
      data record is r-entrada.

 01 r-entrada.
    03 dto-ent              PIC X(02).
    03 FILLER          PIC X(04).
    03 materia-ent          PIC X(18).
    03 clave-ent            PIC X(04).
    03 FILLER          PIC X(52).

 FD    salida
       RECORD CONTAINS 80 CHARACTERS
       VALUE OF ID "salida.DAT"
       LABEL RECORDS ARE STANDARD
       DATA RECORD IS R-salida.

 01 r-salida pic x(80).

 WORKING-STORAGE SECTION.
 77    hay-datos        pic x(01) value "s".
 01 r-base.
    03 dto-sal              PIC X(02).
    03 FILLER          PIC X(02) value spaces.
    03 clave-sal            pic X(04).
    03 materia-sal          PIC X(18).
    03 FILLER          PIC X(54) value spaces.

 PROCEDURE DIVISION.
 ******************************************************************************

 PROGRAMA-PRINCIPAL.

      open input entrada.
      open output salida.
      perform lee-graba until hay-datos equal "n".

 lee-graba.

      read entrada at end move "n" to hay-datos.
      if hay-datos equal "s" then
         move dto-ent to dto-sal
         move clave-ent to clave-sal
         move materia-ent to materia-sal
         write r-salida from r-base
      else
         next sentence.
 *    endif.
```

```
*****************************************************************************
PROGRAM-ID. SEPARA.

ENVIRONMENT DIVISION.
*****************************************************************************

INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT ENTRADA ASSIGN "DBA1:[SDE000.COBOL.FILES]MAESEO.DAT".
SELECT SALIDA  ASSIGN "DBA1:[SDE000.COBOL.FILES]RFC.DAT".

DATA DIVISION.
*****************************************************************************

FILE SECTION.
FD   ENTRADA
     RECORD CONTAINS 365 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-ENTRADA.

01 RECORD-ENTRADA.
     03 FOLIO          PIC X(06).
     03 NOMBRE         PIC X(40).
     03 RFC.
        05 RFC-LETTERS PIC X(04).
        05 RFC-NUM     PIC X(06).
     03 FILLER         PIC X(99).
     03 NUMEROS        PIC X(12).
     03 FILLER         PIC X(198).

FD   SALIDA
     RECORD CONTAINS 80 CHARACTERS
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS RECORD-SALIDA.

01   RECORD-SALIDA PIC X(80).

WORKING-STORAGE SECTION.

77 HAY-DATOS    PIC X(01) VALUE 'S'.

01 BASE-SALIDA.
     03 NOMBRE    PIC X(40).
     03 RFC       PIC X(10).
     03 IMSS.
        05 NUMERO PIC 9(04).
        05 GPO    PIC X(01).
     03 NUMEROST  PIC X(12).
     03 FILLER    PIC X(13) VALUE SPACES.

PROCEDURE DIVISION.
*****************************************************************************

PROGRAMA-PRINCIPAL.

     PERFORM ABRE-ARCHIVOS.
     PERFORM GRABA UNTIL HAY-DATOS EQUAL 'N'.
     PERFORM CIERRA-ARCHIVOS.
     STOP RUN.

ABRE-ARCHIVOS.
```

```
    OPEN OUTPUT SALIDA.

GRABA.

    READ ENTRADA AT END MOVE "N" TO HAY-DATOS.
    IF HAY-DATOS EQUAL "S" THEN
       PERFORM EVALUA.
*   ENDIF

EVALUA.

    IF REC IN RECORD-ENTRADA          NOT EQUAL "  "          AND
       REC-LETTERS IN RECORD-ENTRADA  NOT EQUAL "  "          AND
       REC-NUM IN RECORD-ENTRADA      NOT EQUAL "000000"     AND
       REC-NUM IN RECORD-ENTRADA      NOT EQUAL "  "         THEN
       MOVE CORRESPONDING RECORD-ENTRADA TO BASE-SALIDA
       INSPECT FOLIO IN RECORD-ENTRADA CONVERTING " " TO "0"
       PERFORM EVALUA-FOLIO-IMSS
       PERFORM EVALUA-NUMEROS
       INSPECT NOMBRE IN BASE-SALIDA CONVERTING ";" TO " "
       WRITE RECORD-SALIDA FROM BASE-SALIDA.
*   ENDIF

EVALUA-FOLIO-IMSS.

    IF FOLIO(1:1) EQUAL "0" OR FOLIO(1:1) EQUAL "3" OR FOLIO(1:1) EQUAL "4" TH
       MOVE "A" TO GPO.

    IF FOLIO(1:1) EQUAL "2" THEN
       MOVE "B" TO GPO.

    IF FOLIO(1:1) EQUAL "1" THEN
       MOVE "C" TO GPO.

    MOVE FOLIO(3:4) TO NUMERO.

EVALUA-NUMEROS.

    MOVE NUMEROS TO NUMEROST.

    INSPECT NUMEROST CONVERTING " " TO "0".

    IF NUMEROST(1:2) > "52" THEN
       MOVE "4" TO NUMEROST(1:1) .

    IF NUMEROST(3:2)  > "20" THEN
       MOVE "1" TO NUMEROST(3:1) .

    IF NUMEROST(5:4)  > "1500" THEN
       MOVE "0" TO NUMEROST(5:1) .

    IF NUMEROST(5:4)  < "0030" THEN
       MOVE "8" TO NUMEROST(7:1) .

    IF NUMEROST(9:4)  > "3000" THEN
       MOVE "2" TO NUMEROST(9:1) .

    IF NUMEROST(9:4)  < "0090" THEN
       MOVE "99" TO NUMEROST(11:2) .

CIERRA-ARCHIVOS.
```

```
CLOSE ENTRADA.
CLOSE SALIDA.
```

```
identification division.
program-id. etiquetas.
author.
installation. Cecafi.
security. Programa que emite etiquetas de un archivo de entrada a
          un archivo de salida.

environment division.
configuration section.
source-computer. Vax-11-780.
object-computer. Vax-11-780.
input-output section.
file-control.
    select cards-in assign to "[valerio.cobol.datos]nombres.dat".
    select mailing-labels assign to "[valerio.cobol.reportes]etiquetas.lis".

data division.
file section.
fd cards-in
    label records are omitted
    data record is card.
01 card.
    02 name picture is x(25).
    02 street picture is x(25).
    02 city picture is x(30).

fd mailing-labels
    label records are omitted
    data record is print-record.
01 print-record.
    02 print-line picture is x(132).

working-storage section.
01 end-of-data-indicator picture is x(03).

procedure division.
main-logic.
    open input cards-in output mailing-labels.
    move 'no' to end-of-data-indicator.
    perform read-a-card.
    perform read-print until end-of-data-indicator is equal to 'yes'.
    close cards-in mailing-labels.
    stop run.

read-a-card.
    read cards-in record at end move 'yes' to end-of-data-indicator.

read-print.
    move name to print-line.
    write print-record before advancing 1 line.
    move street to print-line.
    write print-record before advancing 1 line.
    move city to print-line.
    write print-record before advancing 4 line.
    perform read-a-card.
```

```
identification division.
program-id. salaries.
author.
installation. cecafi.
security. programa que calcula los salarios.

environment division.
configuration section.
source-computer. vax-11-780.
object-computer. vax-11-780.
input-output section.
file-control.
    select card-file assign to "[valerio.cobol.datos]employee.dat".
    select print-file assign to "[valerio.cobol.reportes]salario.lis".

data division.
file section.
fd  card-file
    label records are omitted
    data record is card-in.
01 card-in.
    02 name-in pic x(15).
    02 sex-code pic 9(01).
    02 salary-in pic 9(05)v9(02).
fd  print-file
    label records are omitted
    data record is print-line.
01  print-line.
    02 filler pic x(132).
working-storage section.
01  no-of-men pic 9(02) value zero.
01  no-of-women pic 9(02) value zero.
01  men-total-sal pic 9(07)v9(02) value zero.
01  women-total-sal pic 9(07)v9(02) value zero.
01  end-of-data-indicator pic x(03).
01  header-1.
    02 filler pic x(01) value space.
    02 filler pic x(41) value spaces.
    02 filler pic x(13) value "annual salary".
01  header-2.
    02 filler pic x(01) value space.
    02 filler pic x(16) value spaces.
    02 filler pic x(42) value "Employee name          men          women
01  data-line.
    02 filler pic x(01) value space.
    02 filler pic x(15) value spaces.
    02 name-out pic x(15).
    02 filler pic x(05) value spaces.
    02 men-sal-out pic z(03)9(04).9(02) blank when zero.
    02 filler pic x(05) value spaces.
    02 women-sal-out pic z(03)9(04).9(02) blank when zero.

procedure division.
main-logic.
    open input card-file output print-file.
    write print-line from header-1 after advancing page.
    write print-line from header-2 after advancing 2 lines.
    move spaces to print-line.
    write printline before advancing 2 lines.
    move "no" to end-of-data-indicator.
    perform read-card.
    perform process-and-print until end-of-data-indicator is equal "yes".
    perform print-summary.
```

```
    close-card-file print-file.
    stop run.
print-summary.
    move " t o t a l " to name-out.
    move men-total-sal to men-sal-out.
    move women-total-sal to women-sal-out.
    write print-line from data-line after advancing 3 lines.
    move " a v e r a g e" to name-out.
    divide no-of-men into men-total-sal giving men-sal-out.
    divide no-of-women into women-total-sal giving women-sal-out.
    write print-line from data-line after advancing 2 lines.
read-card.
    read card-file record at end move 'yes' to end-of-data-indicator.
process-and-print.
    move name-in to name-out.
    if sex-code = 1
        move salary-in to men-sal-out
        move zeros to women-sal-out
        add 1 to no-of-men
        add salary-in to men-total-sal
    else
        move salary-in to women-sal-out
        move zeros to men-sal-out
        add 1 to no-of-women
        add salary-in to women-total-sal.
    write print-line from data-line before advancing 1 line.
    perform read-card.
```

```
program-id. student.
author.
installation. cecafi.
security. programa que calcula la evaluacion de curso.

environment division.
configuration section.
source-computer. vax-11-780.
object-computer. vax-11-780.
input-output section.
file-control.
    select card-file assign to "[valerio.cobol.datos]student.dat".
    select print-file assign to "[valerio.cobol.reportes]student.lis".

data division.
file section.
fd  card-file
    record contains 80 characters
    label records are omitted
    data record is card-rec.
01  card-rec.
    02 stud-number pic 9(04).
    02 stud-name pic x(25).
    02 grade-1 pic 9(03).
    02 grade-2 pic 9(03).
    02 filler pic x(45).
fd  print-file
    record contains 133 characters
    label records are omitted
    data record is print-rec.
01  print-rec pic x(133).
working-storage section.
77  final-avg pic 9(03)v9(02).
77  tot-1 pic 9(06).
01  head-1.
    02 filler pic x(05) value spaces.
    02 item-1 pic x(07) value "student".
    02 filler pic x(05) value spaces.
    02 item-2 pic x(07) value "student".
    02 filler pic x(19) value spaces.
    02 item-3 pic x(05) value "final".
    02 filler pic x(86) value spaces.
01  head-2.
    02 filler pic x(05) value spaces.
    02 item-4 pic x(06) value "number".
    02 filler pic x(06) value spaces.
    02 item-5 pic x(04) value "name".
    02 filler pic x(22) value spaces.
    02 item-6 pic x(07) value "average".
    02 filler pic x(83) value spaces.
01  student-line.
    02 filler pic x(05) value spaces.
    02 stud-no-out pic 9(04).
    02 filler pic x(08) value spaces.
    02 stud-name-out pic x(25).
    02 filler pic x(01) value space.
    02 avg-out pic 9(03)v9(02).
    02 filler pic x(85) value spaces.

procedure division.
main-logic.
    open input card-file output print-file.
```

```
    move spaces to print-rec.
par-1.
    write print-rec from head-1 after advancing 1 line.
    write print-rec from head-2 after advancing 1 line.

par-2.
    read card-file at end go to end-par.
    add grade-1, grade-2 giving tot-1.
    divide tot-1 by 2 giving final-avg.
    move stud-number to stud-no-out.
    move stud-name to stud-name-out.
    move final-avg to avg-out.
    write print-rec from student-line after advancing 1 line.
    go to par-2.

end-par.
    close card-file print-file.
    stop run.
```

```
identification division.
program-id. traffic.
author.
installation. cecsfi.
security. programa de demostracion.

environment division.
configuration section.
source-computer. vax-11-780.
object-computer. vax-11-780.
input-output section.
file-control.
    select input-file assign to "[valerio.cobol.datos]traffic.dat".
    select output-file assign to "[valerio.cobol.reportes]traffic.lis".

data division.
file section.
fd  input-file
    label records are omitted
    data record is input-rec.
01  input-rec.
    02 daily-traffic pic 9(04).
    02 filler pic x(76).
fd  output-file
    label records are omitted
    data record is output-rec.
01  output-rec pic x(132).
working-storage section.
01  end-of-data pic x(03).
01  min pic 9(04).
01  max pic 9(04).
01  total pic 9(05).
01  day-counter pic 9(01).
01  daily-listing.
    02 filler pic x(31) value spaces.
    02 edit-traffic pic z,zz9.
01  page-header.
    02 filler pic x(27) value spaces.
    02 filler pic x(13) value "daily traffic".
01  week-header.
    02 filler pic x(30) value spaces.
    02 filler pic x(05) value "week ".
    02 week-counter pic 9(02).
01  stat-summary.
    02 filler pic x(04) value spaces.
    02 filler pic x(10) value "minimum = ".
    02 edit-min pic z,zz9.
    02 filler pic x(03) value spaces.
    02 filler pic x(10) value "maximum = ".
    02 edit-max pic z,zz9.
    02 filler pic x(03) value spaces.
    02 filler pic x(15) value "weekly total = ".
    02 edit-total pic zz,zz9.

procedure division.
main-logic.
    move "no" to end-of-data.
    move zero to week-counter.
    open input input-file output output-file.
    perform weekly-routine until week-counter is greater 10.
    close input-file output-file.
    stop run.
weekly-routine.
```

```
    perform read-data.
    if end-of-data is equal 'no'
        perform new-week
        perform read-print until day-counter is equal 7
        perform week-summary.
read-data.
    read input-file record at end move 'yes' to end-of-data
        move 7 to day-counter
        move 11 to week-counter.
new-week.
    add 1 to week-counter.
    move 1 to day-counter.
    move daily-traffic to min max total.
    write output-rec from page-header after advancing page.
    write output-rec from week-header after advancing 2 lines.
    move spaces to output-rec.
    write output-rec after advancing 1 line.
    move daily-traffic to edit-traffic.
    write output-rec from daily-listing after advancing 1 line.
read-print.
    perform read-data.
    if end-of-data is equal 'no'
        perform check-min-max
        add 1 to day-counter
        add daily-traffic to total
        move daily-traffic to edit-traffic
        write output-rec from daily-listing after advancing 1 line.
check-min-max.
    if daily-traffic is greater than max
        move daily-traffic to max.
    if daily-traffic is less than min
        move daily-traffic to min.
week-summary.
    move min to edit-min.
    move max to edit-max.
    move total to edit-total.
    write output-rec from stat-summary after advancing 2 lines.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EJEMPLO DE-CREACION-INDEXADO.
AUTHOR. LAURA Y SERGIO CARDOSO.
*
*
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
    SOURCE-COMPUTER. EN-VAX-11.
    OBJECT-COMPUTER. EN-VAX-11.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT SALIDA ASSIGN TO 'SALIDA.DAT'
           ORGANIZATION IS INDEXED.
    SELECT ENTRADA ASSIGN TO 'SYS$INPUT'.
*
*
DATA DIVISION.
FILE SECTION.
FD  SALIDA
    DATA RECORD IS REG-SAL
    ACCESS MODE DYNAMIC
    RECORD KEY IS NUMCTA-S.
01  REG-SAL.
    03 NUMCTA-S        PIC 9(08).
    03 NOMBRE-S        PIC X(32).
    03 PROMEDIO        PIC 99V99.
FD  ENTRADA
    DATA RECORD IS REG-ENT.
01  REG-ENT.
    03 NUMCTA-E        PIC 9(08).
    03 NOMBRE-E        PIC X(32).
    03 PROMEDIO        PIC 99V99.
*
WORKING-STORAGE SECTION.
*
*
PROCEDURE DIVISION.
PARRAFO-INICIO.
    OPEN INPUT ENTRADA.
    OPEN I-O SALIDA.
    READ ENTRADA AT END
        DISPLAY "YA NO HAY DATOS"
    END-READ.
    PERFORM CREA-INDEXADO UNTIL NUMCTA-E=99999999.
    DISPLAY "DAME EL NUMERO DE CUENTA A BUSCAR".
    ACCEPT NUMCTA-S.
    PERFORM BUSQUEDA UNTIL NUMCTA-S=99999999.
    DISPLAY "DAME NUMCTA PARA MODIFICAR NOMBRE"
    ACCEPT NUMCTA-S.
    PERFORM MODIFICA UNTIL NUMCTA-S=99999999.
    DISPLAY "DAME NUMCTA PARA BORRAR EL REGISTRO"
    ACCEPT NUMCTA-S.
    PERFORM BORRA UNTIL NUMCTA-S=99999999.
    STOP RUN.
CREA-INDEXADO.
    MOVE REG-ENT TO REG-SAL.
    WRITE REG-SAL INVALID KEY
        DISPLAY 'ERROR EN LLAVE' NUMCTA-E
    END-WRITE.
    READ ENTRADA AT END
        DISPLAY "FIN DE DATOS"
    END-READ.
BUSQUEDA.
```

```
    READ SALIDA KEY IS NUMCTA-S
        INVALID KEY DISPLAY "NO ENCONTRO NUMCTA"
    END-READ.
    DISPLAY "NUMCTA " NUMCTA-S "NOMBRE " NOMBRE-S.
    DISPLAY "DAME EL NUMERO DE CUENTA A BUSCAR".
    ACCEPT NUMCTA-S.
MODIFICA.
    START SALIDA KEY IS EQUAL NUMCTA-S INVALID KEY
        DISPLAY "NO HAY REGISTRO A MODIFICAR"
    END-START.
    DISPLAY "NOMBRE A MODIFICAR".
    ACCEPT NOMBRE-S.
    REWRITE REG-SAL INVALID KEY
        DISPLAY "SE SE HIZO EL REWRITE"
    END-REWRITE.
    DISPLAY "DAME NUMCTA PARA MODIFICAR NOMBRE"
    ACCEPT NUMCTA-S.
BORRA.
    READ SALIDA KEY IS NUMCTA-S INVALID KEY
        DISPLAY "NO HAY REGISTRO A BORRAR"
    END-READ.
    DELETE SALIDA INVALID KEY
        DISPLAY "NO SE EFECTUO EL DELETE"
    END-DELETE.
    DISPLAY "DAME NUMCTA PARA BORRAR EL REGISTRO"
    ACCEPT NUMCTA-S.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EJEMPLO-ARCHIVOS-INDEXADO.
AUTHOR.
*****************************************************************
*
*****************************************************************
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
    SOURCE-COMPUTER.   SOLO-PARA-DOCUMENTACION-VAX-11.
    OBJECT-COMPUTER.   SOLO-PARA-DOCUMENTACION-VAX-11.
SPECIAL-NAMES.
    CONSOLE IS PANTALLA
    SYMBOLIC CHARACTERS ESCAPER  PARM1 PARM2 PARM3 PARM4 CAMPANA DOS
                        ARE   26     92    60   101   75      8   2.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARCHIVO-DISCO ASSIGN TO "PRUEBA2.DAT;1"
        ORGANIZATION IS INDEXED.
*****************************************************************
*
*****************************************************************
DATA DIVISION.
FILE SECTION.
FD  ARCHIVO-DISCO
    RECORD CONTAINS 56 CHARACTERS
    VALUE OF ID IS ARCHIVO-DISCO-ID
    DATA RECORD IS REGISTRO-DISCO
    ACCESS MODE IS DYNAMIC
        RECORD KEY IS FICHA.
01  REGISTRO-DISCO.
    02  FICHA                          PIC 9(04).
    03  NOMBRE                         PIC X(27).
    03  SUELDO                         PIC $(03)V9(02).
    03  CALIFICACION                   PIC 9(02)V9(02).
*****************************************************************
*
*****************************************************************
WORKING-STORAGE SECTION.
77  ARCHIVO-DISCO-ID                   PIC X(20).
77  CONTINUA                           PIC X(01)  VALUE "S".
    88  SI-CONTINUA                               VALUE "S".
77  OPCION                             PIC X(01)  VALUE SPACES.
    88  HAY-ALTA                                  VALUE "A".
    88  HAY-BAJA                                  VALUE "B".
    88  HAY-MODIFICACION                          VALUE "M".
    88  NINGUNA                                   VALUE "N".
77  TECLA                              PIC X(01).
*****************************************************************
01  ESCAPER-CHARACTER-ID.  PIC X.
01  TERMINAL-ID.
    03  VT-ID                  PIC XX.
    03  FILLER                 PIC X(4).
01  LET-CNT      PIC 9(2)V99.
01  IN-WORD      PIC X(20).
01  DISP-COUNT   PIC 9(2).
01  HOME-UP.
    03  FILLER          PIC X  VALUE ESCAPER.
    03  FILLER          PIC X  VALUE PARM1.
    03  HOME-LINE       PIC 99 VALUE 00.
    03  FILLER          PIC X  VALUE PARM2.
    03  HOME-COL        PIC 99 VALUE 00.
    03  FILLER          PIC X  VALUE PARM3.
```

```
01  CLEAR-SCREEN.
    03  FILLER          PIC X  VALUE ESCAPER.
    03  FILLER          PIC X  VALUE PARM1.
    03  FILLER          PIC X  VALUE PARM4.

01  CURSOR-POSITIONER.
    03  FILLER          PIC X  VALUE ESCAPER.
    03  FILLER          PIC X  VALUE PARM1.
    03  LINE-POS        PIC 99 VALUE ZEROES.
    03  FILLER          PIC X  VALUE PARM2.
    03  COL-POS         PIC 99 VALUE ZEROES.
    03  FILLER          PIC X  VALUE PARM3.

01  FORMA.
    02 FORMA-1.
       03  FILLER            PIC X  VALUE ESCAPER.
       03  FILLER            PIC X  VALUE PARM1.
       03  LINE-4            PIC 99 VALUE 04.
       03  FILLER            PIC X  VALUE PARM2.
       03  COL-3             PIC 99 VALUE 09.
       03  FILLER            PIC X  VALUE PARM3.
       03  FILLER            PIC X(27) VALUE
           'FICHA : _____'.
    02 FORMA-2.
       03  FILLER            PIC X  VALUE ESCAPER.
       03  FILLER            PIC X  VALUE PARM1.
       03  LINE-4            PIC 99 VALUE 06.
       03  FILLER            PIC X  VALUE PARM2.
       03  COL-3             PIC 99 VALUE 09.
       03  FILLER            PIC X  VALUE PARM3.
       03  FILLER            PIC X(34) VALUE
           'NOMBRE : _____'.
01  X PIC X VALUE ZERO.
*
*****************************************************************
PROCEDURE DIVISION.
PARRAFO-INICIO.
*   DISPLAY CLEAR-SCREEN WITH NO ADVANCING

    DISPLAY ESCAPER PARM1 02 PARM2 00 PARM3
    DISPLAY ESCAPER PARM1 02 PARM4
    DISPLAY '          1)INICIAS CREACION ARCHIVO'
    DISPLAY '          2)YA LO CREASTE CON ANTERIORIDAD'
    DISPLAY CAMPANA CAMPANA CAMPANA CAMPANA
    STOP RUN
    ACCEPT X
    DISPLAY '          DAME EL NOMBRE DEL ARCHIVO '
    ACCEPT ARCHIVO-DISCO-ID FROM PANTALLA
    IF X = '1' THEN
       OPEN OUTPUT ARCHIVO-DISCO
    ELSE
       OPEN I-O ARCHIVO-DISCO.
    PERFORM LETREROS
    PERFORM FIN.
*****************************************************************
*
*****************************************************************
LETREROS.
    DISPLAY CLEAR-SCREEN WITH NO ADVANCING
    PERFORM LINEAS-BLANCAS
            10 TIMES
    IF SI-CONTINUA THEN
       PERFORM OPCIONES
```

```
*****************************************************************
OPCIONES.
    DISPLAY CLEAR-SCREEN WITH NO ADVANCING
    PERFORM LINEAS-BLANCAS
            5 TIMES
    DISPLAY '          TECLEA UNA OPCION DE: ' CAMPANA CAMPANA
    PERFORM LINEAS-BLANCAS
            2 TIMES
    DISPLAY '          A      PARA DAR UN REGISTRO DE ALTA;'
    DISPLAY '          M      PARA HACER UNA MODIFICACION;'
    DISPLAY '          B      PARA DAR DE BAJA ALGUN REGISTRO;'
    DISPLAY '          9      NINGUNA OPCION.' CAMPANA CAMPANA
    PERFORM LINEAS-BLANCAS
            3 TIMES
    DISPLAY '     OPCION: ' WITH NO ADVANCING
    ACCEPT OPCION FROM PANTALLA
    IF NOT NINGUNA THEN
        IF HAY-ALTA THEN
            PERFORM ALTA
        ELSE
            IF HAY-MODIFICACION
                PERFORM MODIFICACION
            ELSE
                DISPLAY '     *** ERROR *** AUN NO TENGO DISPONIBLE LA OPCION: ' OPCION
                DISPLAY CAMPANA CAMPANA CAMPANA
                PERFORM LINEAS-BLANCAS
                        4 TIMES
                DISPLAY '     >>>>>>><<<<<<< OPRIME LA TECLA (RETURN) ' WITH NO ADVANCING
                ACCEPT TECLA FROM PANTALLA
        END-IF.

*****************************************************************

*****************************************************************
LINEAS-BLANCAS.
    DISPLAY ' '.
*****************************************************************
*           ALTA
*****************************************************************
ALTA.
    DISPLAY CLEAR-SCREEN WITH NO ADVANCING
    PERFORM LINEAS-BLANCAS
            1 TIMES
    DISPLAY '                    **** PANTALLA DE ALTAS ****'
    PERFORM LINEAS-BLANCAS
            1 TIMES
    DISPLAY '     TECLEA LOS DATOS DE ACUERDO AL TAMANIO INDICADO POR LOS GUIONES'
    DISPLAY CAMPANA CAMPANA
    PERFORM LINEAS-BLANCAS
            2 TIMES
    DISPLAY '     ----'
    DISPLAY '          FICHA (4 DIGITOS): ' WITH NO ADVANCING
    ACCEPT FICHA FROM PANTALLA
    PERFORM LINEAS-BLANCAS
            2 TIMES
    DISPLAY '     ----------'
    DISPLAY '          NOMBRE (37 ALFANUMERICOS): ' WITH NO ADVANCING
```

```
    ACCEPT NOMBRE FROM PANTALLA
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *                    SUELDO ( 5 DIGITOS):  * WITH NO ADVANCING
    ACCEPT SUELDO FROM PANTALLA
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *          CALIFICACION (4 DIGITOS):  * WITH NO ADVANCING
    ACCEPT CALIFICACION FROM PANTALLA
    WRITE REGISTRO-DISCO
        INVALID KEY PERFORM REPORTA-ERROR-ALTA
    END-WRITE.
*****************************************************************
*****************************************************************
*          MODIFICACION
*****************************************************************
MODIFICACION.
    DISPLAY CLEAR-SCREEN WITH NO ADVANCING
    PERFORM LINEAS-BLANCAS
        1 TIMES
    DISPLAY *                  ,  **** PANTALLA DE MODIFICAC ****
    PERFORM LINEAS-BLANCAS
        1 TIMES
    DISPLAY *     TECLEA LOS DATOS DE ACUERDO AL TAMANIO INDICADO POR LOS GUIONES
    DISPLAY CAMPANA CAMPANA
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *                    FICHA (4 DIGITOS):  * WITH NO ADVANCING
    ACCEPT FICHA FROM PANTALLA
    READ ARCHIVO-DISCO INVALID KEY
        DISPLAY * NO EXISTE*
        GO TO CETREROS.
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *  *      NOMBRE (37 ALFANUMERICOS):  * WITH NO ADVANCING
    DISPLAY NOMBRE
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *                    SUELDO ( 5 DIGITOS):  * WITH NO ADVANCING
    DISPLAY SUELDO
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *      CALIFICACION (4 DIGITOS):  * WITH NO ADVANCING
    DISPLAY CALIFICACION
    DISPLAY *
    DISPLAY *           NOMBRE (37 ALFANUMERICOS):  * WITH NO ADVANCING
    ACCEPT NOMBRE FROM PANTALLA
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *                    SUELDO ( 5 DIGITOS):  * WITH NO ADVANCING
    ACCEPT SUELDO FROM PANTALLA
    PERFORM LINEAS-BLANCAS
        2 TIMES
    DISPLAY *
    DISPLAY *           CALIFICACION (4 DIGITOS):  * WITH NO ADVANCING
    ACCEPT CALIFICACION FROM PANTALLA
```

```
           REWRITE REGISTRO-DISCO
           INVALID KEY GO TO LETREROS.
************************************************************************
*
************************************************************************
 REPORTA-ERROR-ALTA.
     DISPLAY '    NO SE PUEDE ACCESAR ESTE REGISTRO PUES YA EXISTE LA FICHA: 'FICHA
     DISPLAY '    >>>>>>>>>>>>>>> OPRIME LA TECLA (RETURN)' WITH NO ADVANCING
     ACCEPT TECLA FROM PANTALLA.
************************************************************************
*
************************************************************************
 FIN.
     CLOSE ARCHIVO-DISCO
           WITH NO REWIND
     STOP RUN.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MIO.
AUTHOR. YO.
INSTALLATION. CECAFI.
DATE-WRITTEN. AYER.
DATE-COMPILED. HACE-RATITO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    SYMBOLIC CHARACTERS ESCAPE CORCHETE PUNTOYCOMA JOTA ACHE CAMPANA
                 ARE         28      92          60       75   73     8.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 NO-RENGLONES-A PIC 9(02) VALUE 99.
77 NO-RENGLONES-B PIC 9(02).
77 NO-COLUMNAS-A PIC 9(02) VALUE 50.
77 NO-COLUMNAS-B PIC 9(02).
77 I PIC 9(02).
77 J PIC 9(02).
77 K PIC 9(02).
77 IMPRESOS PIC -Z(03).
01 MATRIZ-A.
   03 RENGLON-A OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-A.
         05 COLUMNA-A OCCURS 10 TIMES PIC 9(02).
01 MATRIZ-B.
   03 RENGLON-B OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-B.
         05 COLUMNA-B OCCURS 10 TIMES PIC 9(02).
01 MATRIZ-C.
   03 RENGLON-C OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-A.
         05 COLUMNA-C OCCURS 10 TIMES PIC 9(03).
PROCEDURE DIVISION.
INICIA.
    MOVE ZEROES TO MATRIZ-A
    MOVE ZEROES TO MATRIZ-B
    MOVE ZEROES TO MATRIZ-C
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 ACHE WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 JOTA CAMPANA
    DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 01 ACHE WITH NO ADVANCING
    DISPLAY "NO DE RENGLONES DE A: (12) " WITH NO ADVANCING
    ACCEPT NO-RENGLONES-A
    IF NO-RENGLONES-A > 0 THEN
        DISPLAY "NO DE COLUMNAS DE A: (12) " WITH NO ADVANCING
        ACCEPT NO-COLUMNAS-A
        IF NO-COLUMNAS-A > 0 THEN
            DISPLAY " "
            PERFORM LEE-MATRIZ-A WITH TEST BEFORE
            VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
            AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-A)
            DISPLAY " "
            DISPLAY "  LA MATRIZ A  "
            DISPLAY " "
            PERFORM IMPRIME-A WITH TEST BEFORE
            VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
            PERFORM LINEAS-BLANCAS 2 TIMES
            DISPLAY "NO DE RENGLONES DE B: (12) " WITH NO ADVANCING
            ACCEPT NO-RENGLONES-B
            IF NO-RENGLONES-B > 0 THEN
                DISPLAY "NO DE COLUMNAS DE B: (12) " WITH NO ADVANCING
                ACCEPT NO-COLUMNAS-B
                IF NO-COLUMNAS-A = NO-RENGLONES-B THEN
                    DISPLAY " "
                    PERFORM LEE-MATRIZ-B WITH TEST BEFORE
                    VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-B)
```

```
                AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B).
                PERFORM LINEAS-BLANCAS 3 TIMES
                DISPLAY " "
                DISPLAY "  LA MATRIZ B  "
                DISPLAY " "
                PERFORM IMPRIME-B WITH TEST BEFORE
                VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-B)
                PERFORM LINEAS-BLANCAS 2 TIMES
                PERFORM MULTIPLICACION
                VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
                AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B)
                AFTER K FROM 1 BY 1 UNTIL (K > NO-RENGLONES-B).
                DISPLAY " "
                DISPLAY " EL PRODUCTO DE LAS MATRICES ES:"
                PERFORM IMPRESION WITH TEST BEFORE
                VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
            ELSE
                DISPLAY "LAS MATRICES NO SON CONFORMABLES"
            END-IF
        ELSE
            DISPLAY "MATRIZ INVALIDA"
        END-IF
    ELSE
        DISPLAY "MATRIZ INVALIDA"
    END-IF
    STOP RUN.
MULTIPLICACION.
    COMPUTE COLUMNA-C(I,J)=(COLUMNA-C(I,J) + COLUMNA-A(I,K)
    * COLUMNA-B(K,J)).
IMPRESION.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B).
PANTALLA.
    MOVE COLUMNA-C(I,J) TO IMPRESOS
    DISPLAY "  " IMPRESOS WITH NO ADVANCING.
LINEAS-BLANCAS.
    DISPLAY " ".
LEE-MATRIZ-A.
    DISPLAY "A(", I, ",", J, ")= (I2) " WITH NO ADVANCING
    ACCEPT COLUMNA-A(I,J).
LEE-MATRIZ-B.
    DISPLAY "B(", I, ",", J, ")= (I2) " WITH NO ADVANCING
    ACCEPT COLUMNA-B(I,J).
IMPRIME-A.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA2 WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-A).
IMPRIME-B.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA3 WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B).
PANTALLA2.
    DISPLAY "  ", COLUMNA-A(I,J) WITH NO ADVANCING.
PANTALLA3.
    DISPLAY "  ", COLUMNA-B(I,J) WITH NO ADVANCING.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. M10.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    CONSOLE IS PANTALLA
    SYMBOLIC CHARACTERS
    TRES  ESCAPE CORCHETE PUNTOYCOMA JOTA HACHE PARENTESIS DOS GATO BE
     ARE   3   28   92      60          75    73     41      52   36 67.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 I PIC 9(03).
77 J PIC 9(01) VALUE 1.
77 NOMBRE PIC X(10).
77 NUMERO PIC 9(05).
PROCEDURE DIVISION.
EMPIEZA.
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 HACHE WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE DOS JOTA WITH NO ADVANCING
    DISPLAY ESCAPE PARENTESIS 0
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 HACHE "l" WITH NO ADVANCING
    PERFORM RAYA VARYING I FROM 2 BY 1 UNTIL (I > 39)
    DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 40 HACHE "k" WITH NO ADVANCING
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 17 HACHE "COBOL"
                WITH NO ADVANCING
    DISPLAY ESCAPE GATO 3 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 00 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 02 PUNTOYCOMA 40 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 17 HACHE "COBOL"
                WITH NO ADVANCING
    DISPLAY ESCAPE GATO 4 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 00 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 03 PUNTOYCOMA 40 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 00 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 40 HACHE "x" WITH NO ADVANCING
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE 05 PUNTOYCOMA 00 HACHE "m" WITH NO ADVANCING
    ADD 4 TO J.
    PERFORM RAYA VARYING I FROM 2 BY 1 UNTIL (I > 39)
    DISPLAY ESCAPE CORCHETE 05 PUNTOYCOMA 40 HACHE "j" WITH NO ADVANCING
    DISPLAY ESCAPE PARENTESIS BE
    DISPLAY ESCAPE CORCHETE 14 PUNTOYCOMA 3 HACHE WITH NO ADVANCING
    DISPLAY "HOLA DAME TU NOMBRE  " WITH NO ADVANCING
    ACCEPT NOMBRE
    DISPLAY ESCAPE CORCHETE 16 PUNTOYCOMA 3 HACHE WITH NO ADVANCING
    DISPLAY NOMBRE "DAME UN NUMERO " WITH NO ADVANCING
    ACCEPT NUMERO
    DISPLAY ESCAPE CORCHETE 18 PUNTOYCOMA 3 HACHE WITH NO ADVANCING
    DISPLAY "TU NUMERO ES " NUMERO
    DISPLAY ESCAPE CORCHETE 23 PUNTOYCOMA 3 HACHE WITH NO ADVANCING
    PERFORM NADA VARYING I FROM 1 BY 10 UNTIL (I > 100)
    STOP RUN.
RAYA.
    DISPLAY ESCAPE GATO 6 WITH NO ADVANCING
    DISPLAY ESCAPE CORCHETE J PUNTOYCOMA I HACHE "q" WITH NO ADVANCING.
NADA.
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MIO.
AUTHOR. YO.
INSTALLATION. CECAFI.
DATE-WRITTEN. AYER.
DATE-COMPILED. HACE-RATITO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    SYMBOLIC CHARACTERS ESCAPE CORCHETE PUNTOYCOMA OTA ACHE CAMPANA
    ARE 28 93 40 75 73 8.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 NO-RENGLONES-A PIC 9(02) VALUE 99.
77 NO-RENGLONES-B PIC 9(02).
77 NO-COLUMNAS-A PIC 9(02) VALUE 50.
77 NO-COLUMNAS-B PIC 9(02).
77 I PIC 9(02).
77 J PIC 9(02).
77 K PIC 9(02).
77 IMPRESOS PIC -Z(03).
01 MATRIZ-A.
   03 RENGLON-A OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-A.
      05 COLUMNA-A OCCURS 10 TIMES PIC S9(02)
      SIGN IS LEADING.
01 MATRIZ-B.
   03 RENGLON-B OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-B.
      05 COLUMNA-B OCCURS 10 TIMES PIC S9(02)
      SIGN IS LEADING.
01 MATRIZ-C.
   03 RENGLON-C OCCURS 2 TO 10 DEPENDING ON NO-RENGLONES-A.
      05 COLUMNA-C OCCURS 10 TIMES PIC S9(03).
PROCEDURE DIVISION.
INICIA.
   MOVE ZEROES TO MATRIZ-A
   MOVE ZEROES TO MATRIZ-B
   MOVE ZEROES TO MATRIZ-C
   DISPLAY ESCAPE CORCHETE 00 PUNTOYCOMA 00 ACHE WITH NO ADVANCING
   DISPLAY ESCAPE CORCHETE 02 JOTA; CAMPANA
   DISPLAY ESCAPE CORCHETE 04 PUNTOYCOMA 01 ACHE WITH NO ADVANCING
   DISPLAY 'NO DE RENGLONES DE A: ' WITH NO ADVANCING
   ACCEPT NO-RENGLONES-A
   IF NO-RENGLONES-A > 0 THEN
      DISPLAY 'NO DE COLUMNAS DE A: ' WITH NO ADVANCING
      ACCEPT NO-COLUMNAS-A
      IF NO-COLUMNAS-A > 0 THEN
         PERFORM LEE-MATRIZ-A WITH TEST BEFORE
         VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
         AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-A)
         DISPLAY ' LA MATRIZ A: '
         PERFORM IMPRIME-A WITH TEST BEFORE
         VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
         PERFORM LINEAS-BLANCAS 2 TIMES
         DISPLAY 'NO DE RENGLONES DE B: ' WITH NO ADVANCING
         ACCEPT NO-RENGLONES-B
         IF NO-RENGLONES-B > 0 THEN
            DISPLAY 'NO DE COLUMNAS DE B: ' WITH NO ADVANCING
            ACCEPT NO-COLUMNAS-B
            IF NO-COLUMNAS-A = NO-RENGLONES-B THEN
               PERFORM LEE-MATRIZ-B WITH TEST BEFORE
               VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-B)
               AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B)
               PERFORM LINEAS-BLANCAS 2 TIMES
```

```
                    DISPLAY '  LA MATRIZ B  '.
                    PERFORM IMPRIME-B WITH TEST BEFORE
                    VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-B)
                    PERFORM LINEAS-BLANCAS 2 TIMES
                    PERFORM MULTIPLICACION
                    VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
                    AFTER J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B)
                    AFTER K FROM 1 BY 1 UNTIL (K > NO-RENGLONES-B)
                    DISPLAY '  LA MATRIZ C  '
                    PERFORM IMPRESION WITH TEST BEFORE
                    VARYING I FROM 1 BY 1 UNTIL (I > NO-RENGLONES-A)
                ELSE
                    DISPLAY 'LAS MATRICES NO SON CONFORMABLES'
                END-IF
            ELSE
                DISPLAY 'MATRIZ INVALIDA'
            END-IF
        ELSE
            DISPLAY 'MATRIZ INVALIDA'
        END-IF
    ELSE
        DISPLAY 'MATRIZ INVALIDA'
    END-IF
    STOP RUN.
MULTIPLICACION.
    COMPUTE COLUMNA-C(I,J)=(COLUMNA-C(I,J) + COLUMNA-A(I,K)
    * COLUMNA-B(K,J)).
IMPRESION.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B).
PANTALLA.
    MOVE COLUMNA-C(I,J) TO IMPRESOS
    DISPLAY '    ', IMPRESOS WITH NO ADVANCING.
LINEAS-BLANCAS.
    DISPLAY ' '.
LEE-MATRIZ-A.
    DISPLAY 'A(', I, ',', J, ')=' WITH NO ADVANCING
    ACCEPT COLUMNA-A(I,J).
LEE-MATRIZ-B.
    DISPLAY 'B(', I, ',', J, ')=' WITH NO ADVANCING
    ACCEPT COLUMNA-B(I,J).
IMPRIME-A.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA2 WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-A).
IMPRIME-B.
    PERFORM LINEAS-BLANCAS 2 TIMES
    PERFORM PANTALLA3 WITH TEST BEFORE
    VARYING J FROM 1 BY 1 UNTIL (J > NO-COLUMNAS-B).
PANTALLA2.
    DISPLAY '    ', COLUMNA-A(I,J) WITH NO ADVANCING.
PANTALLA3.
    DISPLAY '    ', COLUMNA-B(I,J) WITH NO ADVANCING.
```

DIRECTORIO DE ALUMNOS DEL CURSO "LENGUAJE DE PROGRAMACION COBOL" QUE SE IMPARTE EN ESTA DIVISION DEL 6 DE SEPTIEMBRE AL 5 DE OCTUBRE DEL PRESENTE AÑO.

1.- AGUILAR MONTOYA CARLOS MARTIN
S. C. T.
PROGRAMADOR
AVS. XOLA Y UNIVERSIDAD
COL. NARVARTE
DELEGACION BENITO JUAREZ
519-50-30

CAIRO No. 95
AZCAPOTZALCO
02080 MEXICO, D.F.
519-51-34

2.- BALTAZAR BARRERA MARTIN
I. N. F. O. N. A. V. I.T.
OPERADOR
XOAL Y EJE CENTRAL LAZARO CARDENAS

NORTE 172 No. 516-6
COL. PENSADOR MAR.
DELEGACION VENUSTIANO CARRANZA
15510 MEXICO, D.F.

3.- CARRASCO P. GONZALO J.
S. C.T. DIREC. GRAL. DESARROLLO TECNOLOGICO

4.- CORTES ROMERO JOAQUIN
CENTRO DE INFORMACION Y COMPUTACION
SISTEMAS
TACUBA No. 5
COL. CENTRO
DELEGACION CUAUHTEMOC
06000 MEXICO, D.F.
524-40-20

5.- GALICIA ROSALES J. ROBERTO
DIREC. GRAL. OBRAS MARITIMAS S. C.T.
JEFE DEPTO. TECNICO AREA GOLFO NORTE
INSURGENTES SUR No. 664-6o. PISO
COL. DEL VALLE
03100 MEXICO, D.F.
687-53-68

AV. CUAUHTEMOC No. 1103-302
DELEGACION BENITO JUAREZ
03100 MEXICO, D.F.

6.- GARCIA FLORES ENRIQUETA GUADALUPE
S. C.T.
SECRETARIA DIRECTOR PROGRAMACION PORTUARIA
PROVIDENCIA No. 807
COL. DEL VALLE
686-76-80

ESTAMPADO No. 251
COL. 20 DE NOVIEMBRE
DELEGACION VENUSTIANO CARRANZA
15300 MEXICO, D.F.
789-56-20

7.- GARDUÑO HERNANDEZ ENRIQUE
BANAMEX
ANALISTA DE PROCEDIMIENTO
ISABEL LA CATOLICA No. 165
COL. OBRERA
DELEGACION CUAUHTEMOC
761-70-38

CALLE 641-72
UNIDAD SAN JUAN DE ARAGON
DELEGACION GUSTAVO A. MADERO
09720 MEXICO, D.F.
794-43-57

8.- GOMEZ AMEZQUITA MIGUEL ANGEL
UNIVERSIDAD AUTONOMA METROPOLITANA
ANALISTA DE SISTEMAS
AV. SAN PABLO No. 180
REYNOSA TAMPS.
382-50-00

MANUEL GUTIERREZ NAJERA No. 263-202
COL. TRANSITO
DELEGACION CUAUHTEMOC
06820 MEXICO, D.F.
588-16-89

9.- GONZALEZ CARBAJAL ANA MARIA
S. C. T. DIREC. GRAL. DESARROLLO TEC.

10.- GONZALEZ LUIS
S. C.T.

11.- GUTIERREZ LOPEZ ANTONIO
BANCO DE MEXICO
TECNICO PROGRAMADOR
5 DE MAYO No. 2
COL. CENTRO
DELEGACION CUAUHTEMOC
06059 MEXICO, D.F.
512-81-55

12.- HERNANDEZ RUIZ MA. ALEJANDRA
DIREC. GRAL. OBRAS MARITIMAS
PROGRAMADOR
INSURGENTES SUR No. 664
COL. DEL VALLE
687-53-68

U. LINDAVISTA VALLEJO EDIF. 21-A-303
DELEGACION GUSTAVO A. MADERO
07720 MEXICO, D.F.
567-84-02

13.- LARA VELASCO JOSE ANTONIO
DIREC. GRAL. OBRAS MARITIMAS
AUXILIAR TECNICO
INSURGENTES SUR No. 664-6o. PISO
COL. DEL VALLE
DELEGACION BENITO JUAREZ
687-55-10

CUAUHTEMOCZIN No. 38-1
DELEGACION TLALPAN
691-33-22

14.- LLORENTE GIL LOURDES
DIREC. GRAL. OBRAS MARITIMAS
PROGRAMADOR
INSURGENTES SUR No. 664
COL. DEL VALLE
DELEGACION BENITO JUAREZ
523-80-94

DEPORTES M-140 L-10
PEDREGAL DE SAN NICOLAS TOTOLAPAN

15.- MARTINEZ HERNANDEZ ANTONIO
S. C. T.
INGENIERO AUXILIAR
INSURGENTES SUR No. 664
COL. DEL VALLE
DELEGACION BENITO JUAREZ
687-55-10

CALLE 17 No. 62
COL. JUAREZ PANTITLAN
56460 NEZAHUALCOYOTL
658-10-74

16.- MORENO GUTIERREZ ARTURO CARLOS
ICI DE MEXICO, S. A. DE C.V.
FORMULADOR DESARROLLO NUEVOS
PRODUCTOS Y PROGRAMACION
SAN JUAN IXHUATEPEC
754-44-77

CONVENTO DE LA MERCED No. 39
SANTA MAURA
54050 TLALNEPANTLA, EDO. DE MEXICO
398-58-25

17.- NAVARRO HERNANDEZ ARTURO
FAC. DE INGENIERIA
AYUDANTE DE PROFESOR "B"
CIUDAD UNIVERSITARIA
550-52-15 ext. 3732

RETORNO 27 No. 31
COL. AVANTE
DELEGACION COYOACAN
04460 MEXICO, D.F.
544-80-83

18.- PAREDES GARCIA MARIA DE LA LUZ
S. C.T.
AUXILIAR DE PROGRAMADOR
XOLA Y AV. UNIVERSIDAD
549-51-34

CALLE ORIENTE 235 No. 16
COL. AGRICOLA ORIENTAL
DELEGACION IXTACALCO
08500 MEXICO, D.F.
558-17-48

19.- REYES PIZANO ADOLFO
DIREC. GRAL. OBRAS MARITIMAS
JEFE OFNA. TOPOHIDROGRAFIA
AV. PROV. No. 807-6o. PISO
COL. DEL VALLE
823-66-24

AV. PROVIDENCIA No. 807-6o. PISO
561-03-64

20.- RIOS CASTILLO ANGEL
S. C. T.

21.- RODRIGUEZ LOZANO OCTAVIO
BANAMEX
ANALISTA DE SISTEMAS
FRAY. S. T. DE MIER No. 42-11o. PISO
COL. CENTRO
588-72-84

J. ANTONIO TORRES 806-204
COL. VIADUCTO PIEDAD
588-72-84

22.- ZARATE FIGUEROA MARCOS FCO.
FAC. INGENIERIA, U. N. .A M.
TECNICO ACADEMICO AUX. "B"
CD. UNIVERSITARIA
DELEGACION COYOACAN
550-00-41

AV. SAN JORGE MZ 817 LOTE 4
SANTA URSULA COAPA
04600 MEXICO, D.F.