LENGUAJE DE PROGRAMACION BASIC II

A N E X O

NOVIEMBRE, 1985

DECFI

CURSO: Lenguaje de Programación BASIC con aplicaciones (segunda parte)

COORDINADOR: M. en C. Carlos A. Ramos Larios

Calendario de los Subtemas

FECHA	EXPOSITOR	S U B T E M A S
Primer viernes 8 Nov. 85	CRL-HSS	1.1 Repaso de BASIC orientado a VAX 1.2 Acceso al lenguaje BASIC de la computadora VAX como intérprete.
Primer sábado 9 Nov. 85	SMZ-GRO	6.1 Uso del lenguaje BASIC en la computadora VAX 1.3 Principales comandos de DCL y del editor EDT 1.4 Instrucciones adicionales de BASIC en VAX.
Segundo viernes 15 Nov. 85	CRL-HSS	2.1 Introducción a las organizaciones de archivos. 2.2 Organización secuencial.
Segundo sábado 16 nov. 85	HAU-GRO	6.2 Programa para actualizar un archivo secuencial. 2.3 Organización relativa 2.4 Organización secuencial con índices (1a. parte).
Tercer viernes 22 nov. 85	SMZ-GRO	2.4 Organización secuencial con índices (2a. parte). 2.5 Organización por dispersión.
Tercer sábado 23 Nov. 85	HAU-VLC	6.3 Programas para actualizar un archivo secuencial con índices. 3.1 Archivos en IBM PC, Radio Shack y compatibles. 3.2 Archivos en APPLE y compatibles 4.1 Nóminas
Cuarto viernes 29 Nov. 85	HAU-CRL	3.3 Comentarios acerca del estado actual de la microcomputación en México 5.2 Análisis estructural de edificios.

FECHA	EXPOSITOR	S U B T E M A S
Cuarto sábado 30 Nov. 85	SMZ-CRL	6.4 Programa para actualizar un archivo en organización relativa. 6.5 Uso de programas de aplicación (Análisis estructural). 5.3 Precios Unitarios
Quinto viernes 6 Dic. 85	VLC-HSS	4.2 Almacén 4.3 Contabilidad 5.1 Ruta crítica
Quinto sábado 7 Dic. 85	VLC-HSS	6.5 Uso de programas de aplicación (ruta crítica). 4.4 Pronósticos.

2.1 Introducción a las organizaciones de archivos.

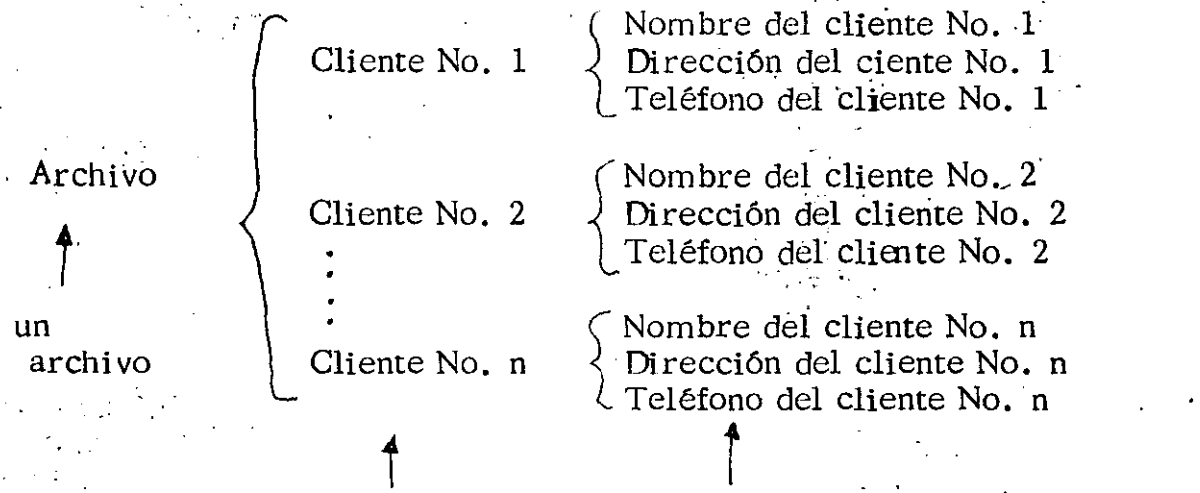
En lenguaje común:

Archivo: Colección de información.

En programación.

Archivo: Conjunto de registros
Registro: Conjunto de campos.
Campo: Dato elemental (conjunto de caracteres)

Ejemplo de los componentes de un archivo de clientes simple:



Ejemplo de un archivo con los componentes del ejemplo anterior:

Campos:	Nombre	Dirección	Teléfono
	Lic. Julio Hernández González	Av. Hidalgo # 6	537.21.92
	Srita. Ma. Eugenia López P.	Insurgentes 348-2	231.22.17
	.		
	.		
	.		
	.		
	Ing. José Luis Gómez	Calle 3 s/n	s/t

así puede quedar dentro de la computadora (disco, cinta, etc.)

Nomenclatura adicional para archivos, registros y campos:

- Longitud de un campo: número máximo de caracteres (posiciones) que ocupa. (se incluyen también los espacios).

  ejemplo: El campo de "teléfono" del primer registro del ejemplo anterior es de longitud 8.

- Conjunto de campos de longitud fija: Cuando todos los campos del mismo tipo de todos los registros tienen la misma longitud.

  ejemplo:

  Si los campos "teléfono de todos los registros del ejemplo anterior son de longitud 8, el conjunto de campos "teléfono"es de longitud fija en el archivo. Como en ocasiones es deseable definir campos de longitud fija, al último registro del ejemplo se le pueden añadir 5 espacios a los caracteres "s/t", para que también sea de longitud 8.

- Conjunto de Campos de Longitud Variable: Cuando al menos alguno de campos del tipo que se está considerando, de alguno de los registros es de longitud diferente a los demás.

  ejemplo:

  Si en el ejemplo anterior el campo de "dirección" del primer registro (Ave. Hidalgo No. 6) no se complementa con espacios, será de longitud diferente a la del segundo registro (Insurgentes 348-2), por lo que el conjunto de campos "dirección" será de longitud variable.

- Longitud de un registro: Suma de las longitudes de todos los campos que contiene.

- Archivo de Registros de longitud fija: Cuando todos los registros del archivo tienen la misma longitud.

- Archivo de registros de longitud variable: Cuando al menos alguno de los registros del archivo es de longitud diferente a los demás.

  ejemplo:

  La forma más común de obtener un archivo de registros de longitud variable es cuando los conjuntos de campos son de longitud variable, aunque también se puede lograr si algunos de los registros contienen campos adicionales a los que contienen los demás registros (por ejemplo, si el

segundo registro tuviera además un campo de "Razón Social").

Aunque poco común y sobre todo poco útil, (también hay que notar que la conjunción de campos adicionales (o faltantes) en registros puede dar lugar a un archivo de registros de longitud fija).

Principales formas de organizar los registros dentro de un archivo.

Atendiendo a la forma que se pueden acceder los registros dentro de un archivo, se definen las dos siguientes organizaciones elementales.

SECUENCIAL: Organización en la cual solo es posible acceder (o llegar) al registro inmediato siguiente al registro en turno.

ejemplo: Un archivo grabado en un cassette a un listado en una impresora.

RELATIVO (o Directo, o Random): Organización en la cual es posible acceder a cualquier registro si se proporciona únicamente su posición relativa.

ejemplo: Un vector DIM A(ZO) y la siguiente organización derivada:

INDEXADO (o por índices): Organización en la cual es posible acceder a cualquier registro si se proporciona el contenido de uno o más campos.

ejemplo: Se requiere el teléfono del cliente cuyo nombre es "Roberto García". Este campo se conoce con el nombre de llave de acceso al registro.

Existen otras organizaciones derivadas más complejas (bases de datos), sin embargo, como su nombre lo indica, todas ellas pueden derivarse (o construírse) a partir de las primeras dos, por lo que no se comentarán ahora.

Como se puede desprender de los conceptos y ejemplos dados anteriormente, las diferentes organizaciones de archivos en general son independientes del medio físico donde se soportan. Así, puede existir una organización secuencial en memoria, o en diskette, o en tarjetas, o en cassette, o una organización relativa en disco duro, o en memoria, etc.

La única excepción es que organizaciones relativas no pueden ser soportadas por dispositivos físicos secuenciales (stream) como son las cintas.

Esta organización (o sus derivados) requieren dispositivos físicos eminentemente directos (DASD = Direct Access Storage devicess), como son los discos, diskettes, memoria, etc.

Por razones fundamentalmente históricas, al estudio de las organizaciones de archivos cuando se soportan en memoria se le conoce con el nombre de "estructuras de datos", aunque con el tiempo esta división tendrá que desvanecerse y quizá - - "estructuras de datos" se utilice también para denotar organizaciones soportadas en los otros dispositivos físicos.

Importancia de los soportes magnéticos para los archivos (en comparación - con memoria).

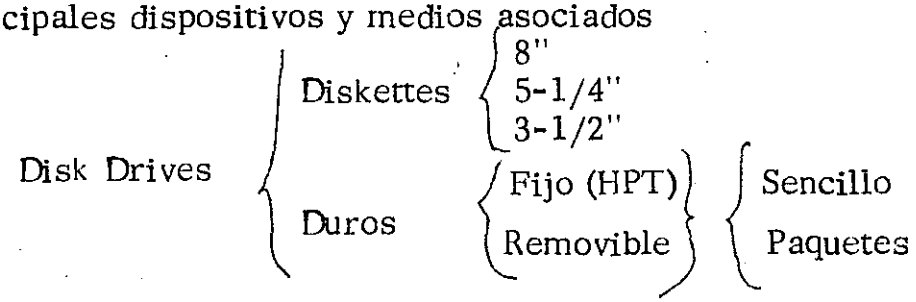
VENTAJAS:

- Permanencia de la informaci ón (memoria es volátil).
- Bajo costo
- Movilidad (removibles)
- Transportabilidad (varias marcas de computadoras).
- Alta densidad (6250 bpi en cintas de 9 canales)
- Confiabilidad (técnicas de paridad).

DESVENTAJAS:

- Tiempos de acceso (100 a 1000 veces más lentos)
- Algunos medios (y formateos) particulares para cada marca o modelo.
- Sensibles a destrucción (campos magnéticos, polvo, temperatura, contacto físico).

Principales dispositivos y medios asociados



- Tape Drives
  - Cassette de audio
  - Cinta 1/2"
  - Cartucho
    - especial
    - Video cassette
- Otros Magnéticos
  - Bandas magnéticas (tarjetas de crédito)
  - Caracteres Magnéticos.
- Otros no magnéticos
  - Tarjetas perforadas
  - Cintas de papel perforadas
  - Barras ópticas.
  - Caracteres ópticos.
  - Circuitos (PROMS, EIROMS)

Poco usadas actualmente

Ventajas adicionales del uso de archivos:

Expande la capacidad de memoria

Permite el uso de un mismo hardware con aplicaciones muy variadas y sobre todo, muestra la verdadera potencialidad de una computadora.

LENGUAJE DE PROGRAMACION BASIC - SEGUNDA PARTE

INTRODUCCION AL DCL
(DIGITAL COMMAND LENGUAJE)

ING. ALEJANDRO JIMENEZ GARCIA

NOVIEMBRE DE 1985

ACCESO AL SISTEMA VAX 11/780

INTRODUCCION:

El Lenguaje de Comandos de VAX, llamado DCL (Digital Command Language) es el primer lenguaje de alto nivel diseñado para comunicación interactiva y por BATCH con el mismo conjunto de instrucciones.

Esto representa un gran adelanto, dado que tradicionalmente se utiliza un lenguaje de comandos para comunicación interactiva y otro para comunicación por BATCH.

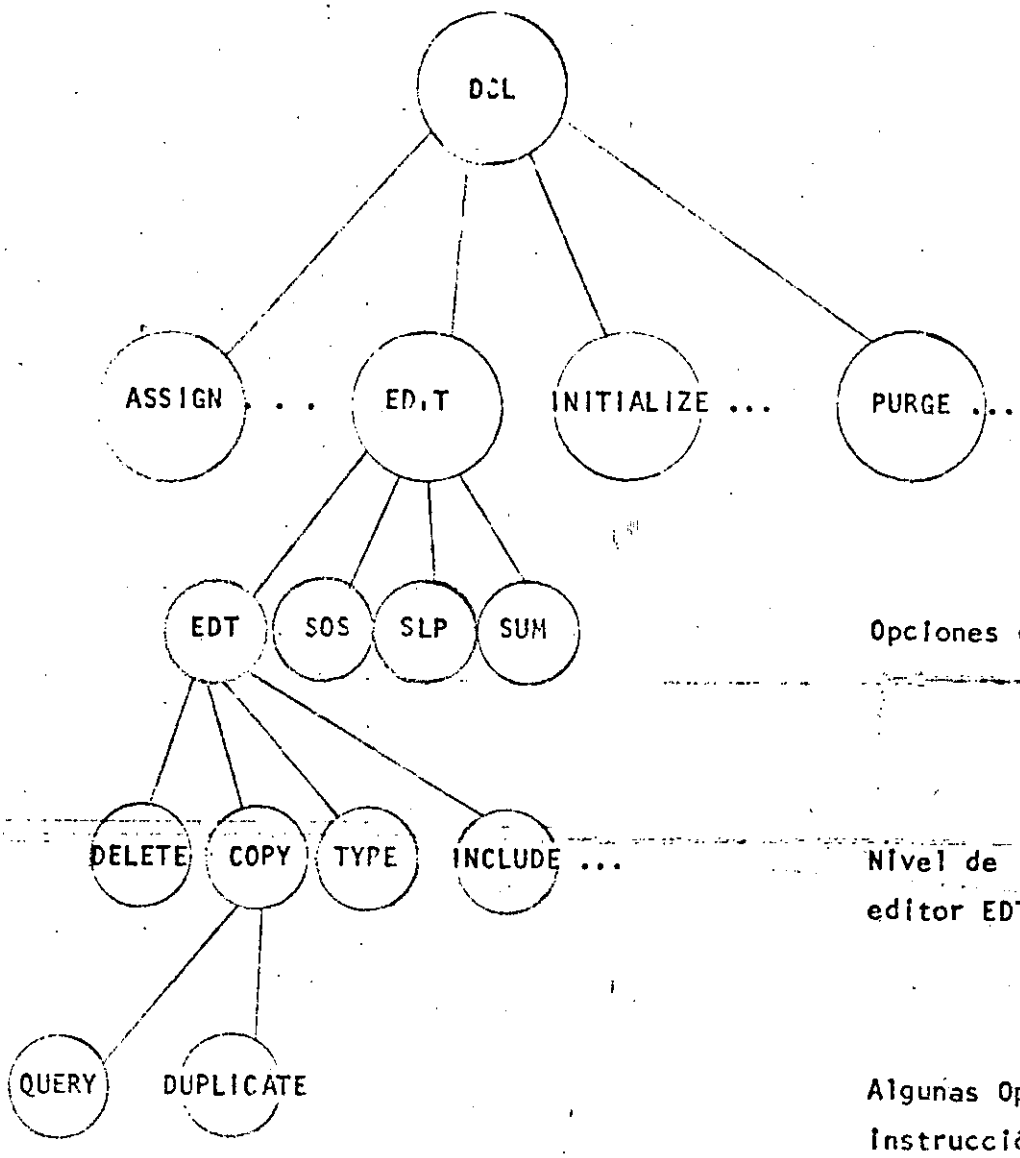
En el caso de IBM el lenguaje de Comandos JCL (Job Control Language) resulta sumamente difícil de programar, entender y aprender.

En Burroughs existen WFL (Work Flow Language) y CANDE (Command and Edit) para procesamiento por BATCH y procesamiento interactivo respectivamente, y si bien son sencillos de aprender y poderosos en su funcionamiento (en especial WFL), son dos lenguajes diferentes.

En el caso de DCL un solo lenguaje es usado para ambos casos, lo cual facilita su aprendizaje, esto, aunado a su poderío nos permite comunicación amplia y extensa con el Sistema VAX-11/780.

El lenguaje DCL proveé al usuario VAX con un extenso conjunto de Instrucciones para:

Desarrollo interactivo de programas
Ejecución y Control de programas por Batch
Manipulación de Dispositivos de Entrada y Salida
Manejo de Archivos de Información.



Nivel de Comandos de DCL

Opciones del comando EDIT.

Nivel de Instrucciones del editor EDT.

Algunas Opciones de la Instrucción COPY.

FIGURA 1.

A continuación se muestran los comandos básicos utilizados para poder tener acceso al sistema, así como los comandos necesarios para la edición de programas.

Este instructivo pretende ser un texto introductorio al sistema VAX-11/780, razón por la cual, y para facilitar ciertas descripciones es posible que algunos comandos sean presentados de una manera poco rigurosa y sin todas las opciones del mismo. Es por ello que no debe ser tomado como un curso formal sino como un manual de acceso rápido al Sistema.

ACCESO AL SISTEMA:

El lenguaje de Comandos DCL tiene una construcción de tipo arborescente, siendo el primer nivel el correspondiente a los diferentes comandos de DCL y en los subsiguientes niveles se encuentran las diferentes opciones de cada comando. En la figura 1 se muestra un ejemplo de esto.

CONVENCIONES USADAS EN ESTE INSTRUCTIVO:

1. Todas las instrucciones proporcionadas por el usuario están subrayadas.

2. El símbolo (D) asociado a un comando indica la opción que por default tomará el Sistema.

3. Puntos suspensivos hacia abajo indican que no se imprime todo el texto que el sistema desplegaría.

4. Puntos suspensivos horizontales indican que se requiere información adicional.

5. Algún dato encerrado en paréntesis cuadrados indica que el dato es opcional.

FORMATO DE LOS COMANDOS.

Los comandos son palabras del idioma inglés que describen la acción que se desea ejecutar.

Estos comandos pueden, opcionalmente, contener calificadores u opciones, que modifican la acción del comando.

Ejemplo:

```
$ PRINT  EJEMPLO.DAT
```

Este comando manda a la cola de impresión un listado del archivo EJEMPLO.DAT

```
$ PRINT/COPIES = 2 EJEMPLO. DAT
```

La acción de este calificador es mandar 2 copias del archivo especificado a la cola de impresión.

Todos los calificadores van separados por el símobolo ' / '

Ejemplo:

```
$ PRINT/COPIES=20/AFTER=23:  EJEMPLO. DAT
Comando     Calificadores     Archivo por imprimir
```

Manda 20 copias a la cola de impresión, pero estas no se imprimirán hasta después de las 23 horas.

Cuando alguno de los parámetros del comando no es especificado, el sistema lo pide, de este modo, nos ayuda a completar nuestro cometido llevándonos de la mano.

Ejemplo:

```
$ PRINT

$ File: EJEMPLO. DAT
```

Parámetro del comando PRINT pedido por el Sistema.

La forma general de un comando de DCL es

```
$ Comando /calificador 1/calificador 2 ... parámetros
  calificador 3/calificador 4 ...
```

Calificador 3 y calificador 4 son calificadores de los parámetros del comando, mientras que calificador 1 y calificador 2 son calificadores del comando.

LA TERMINAL VT 100

La terminal es el medio a través del cual nos comunicamos con la computadora, es por ello que conviene conocer algunas de sus características. A pesar de que parece una máquina de escribir común y corriente, tiene 3 diferencias esenciales:

1) Interpretación de Caracteres Mayúsculos y Minúsculos
2) El Buffer de Almacenamiento
3) Teclas de función especial.

1) Cuando se teclean comandos en minúsculas, el intérprete de DCL se encarga de cambiarlas a mayúsculas y mandar así el comando.

2) Después de mandar un comando, y mientras el intérprete de DCL lo ejecuta, el teclado no se bloquea, esto es, el usuario puede seguir tecleando aunque esto no aparezca en la terminal, ya que contiene un buffer de almacenamiento para este fin. Cuando el comando anterior ha terminado de ejecutarse, la terminal desplegará completo el nuevo comando que hemos estado tecleando hasta ese momento. En caso de haber mandado varios comandos, el sistema primero los ejecuta y luego los despliega.

3) Teclas de función especial.

En la tabla 1 se muestra un resumen de las funciones que realizan algunas de las teclas de la terminal.

Para poder observar las características físicas de una terminal, es necesario dar el comando:

$ SHOW TERMINAL

Table 1-1
Terminal Function Keys

Key	Function
RETURN	Carriage return; transmits the current line to the system for processing. (On some terminals, the RETURN key is labeled CR.) Before a terminal session, initiates login sequence.
Control keys	Define functions to be performed when the CTRL key and another key are pressed simultaneously. All CTRL/x key sequences are echoed on the terminal as ^x.
CTRL/C and CTRL/Y	During command entry, cancels command processing. Before a terminal session, initiates login sequence. Interrupts command or program execution and returns control to the command interpreter.
CTRL/I	Duplicates the function of the TAB key.
CTRL/K	Advances the current line to the next vertical tab stop.
CTRL/L	Form feed.
CTRL/O	Alternately suppresses and continues display of output to the terminal.
CTRL/Q	Restarts terminal output that was suspended by CTRL/S.
CTRL/R	Retypes the current input line and leaves the cursor positioned at the end of the line.
CTRL/S	Suspends terminal output until CTRL/Q is pressed.
CTRL/U	Discards the current input line.
CTRL/X	Discards the current line and deletes data in the type-ahead buffer.
CTRL/Y	(See CTRL/C.)
CTRL/Z	Signals end-of-file for data entered from the terminal.

Para poder modificar las características de una terminal es necesario dar el cómando:

$ SET TERMINAL/OPCION

Más adelante se verán en detalle éstos dos comandos.

COMO ENTRAR AL SISTEMA:

Para lograr la atención del sistema, basta con oprimir la tecla RETURN, o bien, simultáneamente CTRL/Y.

El sistema responde pidiendo el USERNAME.

Una vez que este último ha sido tecleado, el sistema pide el PASSWORD.

Se realiza el proceso de verificación de la clave o cuenta de usuario y si ésta es válida, el sistema responde:

WELCOME TO VAX/VMS VERSION 3.0

.
.
.

$

Siendo el símbolo dólar ($) el que nos indica que estamos en el nivel de comandos de DCL, con lo cual podemos empezar a comunicarnos con el sistema.

Ejemplo:

(RET)

USERNAME: OSITO (RET)

PASSWORD: PANDA NOTA: El password no se despliega al ser tecleado.

WELCOME TO VAX/VMS VERSION 3.0

$

COMO SALIR DEL SISTEMA:

Una vez terminado nuestro trabajo con la VAX, basta darel comando

LOGOUT /BRIEF (D)

/FULL

Con lo que terminará nuestra sesión de trabajo.

COMANDO HELP

La VAX contiene una extensa biblioteca de ayuda al usuario, la cual es desplegada con el comando HELP.

En caso de no recordar la sintaxis específica de un comando basta teclear:

$ HELP COMANDO

Y con ello el sistema nos indicará las características y requerimientos del mismo.

Ejemplo:

$ HELP

Despliega los comandos existentes en DCL

$ HELP PRINT

Despliega las funciones y requerimientos del comando PRINT

$ HELP PRINT/COPIES

Despliega las funciones del comando PRINT afectado por el calificador COPIES.

Debe hacerse notar, que el comando HELP es un comando de AYUDA y no de ENSEÑANZA por lo que el usuario no deberá esperar aprender DCL a través del comando HELP.

COMANDOS DE DCL

Se presenta a continuación un resumen de los comandos de DCL más utilizados, explicando su función y presentando las opciones más importantes de cada comando.

Para mejor referencia consultar el manual "COMMAND LANGUAGE USER'S GUIDE" número AA - DØ23B-TE DEC.

OPERADORES DE RELACION

Aritméticos	.EQ.	igual	.EQS.	Strings
	.GE.	mayor o igual	.GES.	
	.GT.	mayor	.GTS.	
	.LE.	menor o igual	.LES.	
	.LT.	menor	.LTS.	
	.NE.	diferente	.NES.	

.OR.	Booleanos
.AND.	
.NOT.	

COMANDO @

Ejecuta un procedimiento de comandos previamente almacenado en disco.

Formato: $@archivo

Archivo es el nombre del conjunto de comandos que se desea ejecutar. Este archivo se asume de tipo .COM por default.

Ejemplo:

$@ COMANDOS.COM

OPCIONES:

/OUTPUT = archivo2 :La salida generada por la ejecución del procedimiento de comandos es mandada al archivo 2 especificado.

Ejemplo:

$ @COMANDOS.COM/OUTPUT = SALIDA

COMANDO ASSIGN:

Hace una asignación de un nombre lógico con un dispositivo físico, o un archivo u otro nombre lógico.

Formato:

$ ASSIGN nombre equivalente nombre lógico

Nombre equivalente: especifica el nombre del dispositivo o archivo al que se le va a asignar un nombre lógico.

Nombre lógico: especifica el nombre lógico (1 a 63 caracteres) que se va a asociar con el dispositivo.

Ejemplo:

$ ASSIGN TTG3: SYS$PRINT

$ PRINT ARCHIVO

El archivo del sistema SYS$PRINT (Que es el archivo donde se generan las impresiones) se asigna al dispositivo TTG3: Al mandar la instrucción PRINT, la impresión saldrá en el dispositivo TTG3: el cual puede ser una impresora remota.

COMANDO BASIC

Se invoca al compilador BASIC

Formato:

$ BASIC archivo

Archivo es el programa fuente que será compilado.

Asume que el archivo es de tipo .BAS

Ejemplo 1:

$ BASIC PROG.BAS -

Compila el programa fuente PROG.BAS y genera el archivo objeto PROG.OBJ

Ejemplo 2:

$ BASIC

VAX-11 Basic

Ready

.
.
.

Si se teclea BASIC sin especificar el nombre de archivo se inicia una comunicación interactiva con la computadora, funcionando BASIC como intérprete.

OPCIONES:

Las opciones se muestran en el correspondiente manual de BASIC.

COMANDO COBOL

Invoca al compilador Cobol.

Formato:

$ COBOL archivo

Archivo es el programa fuente que será compilado.

Asume que el archivo es de tipo .COB

Ejemplo:

$ COBOL PROG.COB

Compila el programa fuente PROG.COB y genera el archivo objeto PROG.OBJ

OPCIONES:

Las opciones se muestran en el correspondiente manual de COBOL.

COMANDO DEASSIGN

Cancela las asignaciones de nombres lógicos.

Formato:

$ DEASSIGN nombre lógico

Ejemplo:

$ ASSIGN DBA1: COPY

$ DEASSIGN COPY

El comando ASSIGN asigna el nombre lógico COPY al dispositivo DBA1: El comando DEASSIGN suprime esta asignación.

COMANDO DELETE

Borra uno o más archivos almacenados en disco.

Formato:

$ DELETE lista de archivos

Ejemplo:

$ DELETE arch1.PAS;2, arch.BAS;*

Borra la 2a. versión del archivo arch1.PAS y además todas las versiones de arch.BAS

Que haría DELETE *.*;*

¡Cuidado! Borra todo.

OPCIONES.

/ CONFIRM Esta opción hace que el sistema pregunte antes de borrar un archivo.

Ejemplo:

$ DELETE/CONFIRM *.*;*

COMANDO DELETE/ENTRY

Sirve para descontinuar un JOB que esté en alguna cola de BATCH ó impresión.

Formato:

$ DELETE/ENTRY = Lista de números de JOB's nombre de cola

Número de JOB: Es el número asignado al JOB en la cola.

Número de cola: Cola de donde se van a descontinuar los Job's.

Ejemplo:

$ PRINT/HOLD ARCH. TXT

Job 110 entered on queue SYS$PRINT

.
.
.

$ DELETE/ENTRY = 110 SYS$PRINT

La instrucción PRINT coloca en la cola SYS$PRINT el archivo ARCH.TXT con el número 110 y lo mantiene en ella (Opción Hold) sin imprimirse hasta que se le indique posteriormente.

La instrucción DELETE/ENTRY borra de la cola SYS$PRINT el JOB 110 el cual ya no se imprimirá.

COMANDO DIRECTORY

Despliega una lista de los archivos almacenados en la clave del usuario.

Formato:

$ DIRECTORY [Lista de archivos]

Ejemplos

$ DIRECTORY

Despliega todos los archivos con todas sus versiones

$ DIRECTORY *.PAS

Despliega todos los archivos de tipo .PAS

COMANDO EDIT

Invoca al Editor de Pantalla.

Formato

EDIT archivo

Archivo es el nombre del archivo que deseamos editar.

Ejemplos: $ EDIT ARCHIVO.FOR

OPCIONES

/SOS (D)	Edita con el editor SOS
/EDT	Edita con el editor EDT
/SUM	Edita con el editor SUM
/SLP	Edita con el editor SLP

Si no se especifica la opción toma el Default SOS.

En la segunda parte de este escrito se hace una descripción del Editor EDT

COMANDO FORTRAN

Este comando invoca al Compilador FORTRAN

Formato

$ FORTRAN ARCHIVO

Archivo es el nombre del archivo que se desea compilar. Se asume que este archivo es de tipo .FOR

Ejemplos

$ FORTRAN PROG.FOR

COMANDO MAIL.

Este comando es usado para mandar mensajes de un usuario a otro.

FORMATO: $ MAIL

Para una mejor descripción del comando, consultar el manual VAX-11 Utilities Reference Manual.

COMANDO PASCAL.

Este comando invoca al compilador PASCAL.

FORMATO: $ PASCAL Archivo.

Archivo es el nombre del archivo que se desea compilar, se asume que este archivo es de tipo .PAS

Ejemplo:

$ PASCAL PROG.PAS

COMANDO PRINT.

Este comando sirve para meter en cola, uno o más archivos para impresión.

FORMATO: $ PRINT ARCH1, ARCH2

Si no se especifica el tipo de archivo el comando asume un archivo de tipo .LIS

Ejemplo:

$ PRINT ARCHIVO. PAS , ARCH1.COB

OPCIONES MAS USADAS.

/AFTER = hora.

Con esta opción el listado se imprimirá después de la hora especificada.

/QUEUE = nombre de cola.

Imprime el archivo especificado en la cola de impresión especificada; cada impresora o terminal impresora tiene asociado un nombre de cola.

Ejemplo:

$ PRINT ARCH/QUEUE = ANEXO:/AFTER = 20

Imprime el archivo ARCH.LIS en el dispositivo llamado ANEXO: después de las 20 horas.

COMANDO PURGE

Borra todas las versiones existentes de un archivo, excepto la más reciente (La de número de versión mayor).

FORMATO: $ PURGE [archivos]

Ejemplo:

$ PURGE

Borra todas las versiones de todos los programas excepto la más reciente.

$ PURGE *.COM

borra todas las versiones de los archivos de tipo .COM excepto la más reciente.

COMANDO RENAME.

Sirve para cambiar de nombre, tipo, versión etc. un archivo o directorio.

FORMATO: $ RENAME ARCH1, ARCH2

Ejemplo:

$ RENAME

$ FROM: Arch. PAS

$ TO: Archivo .COB

Cambia el nombre del archivo Arch.PAS a Archivo. COB

COMAND REQUEST.

Este comando es usado para mandar un mensaje al operador del sistema.

FORMATO: $ REQUEST "mensaje"

Ejemplos:

$ REQUEST "HOLA SR. OPERADOR"

COMANDO RUN.

Sirve para ejecutar un programa de tipo .EXE (es decir ya compilado y ligado).

FORMATO: $ RUN ARCHIVO

Ejemplo: $ RUN corre. exe

$ RUN PROG

Asume que el tipo de Archivo es .EXE

COMANDO STOP.

Sirve para detener la ejecución de un proceso que fue interrumpido con el comando CTRL/Y.

FORMATO: $ STOP Nombre del Proceso

Ejemplo: $ STOP PROGRAMA

Detiene la ejecución del proceso programa.exe.

COMANDO STOP/ABORT.

Aborta un job que se está imprimiendo.

FORMATO: $ STOP/ABORT Identificación de impresora

Ejemplo: $ STOP/ABORT LPA1:

Descontinúa la impresión que está saliendo por la impresora LPA1:

COMANDO STOP/ENTRY.

Descontinúa de la cola de Batch el proceso que esté corriendo.

FORMATO: $ STOP/ENTRY = Número de job nombre de cola

Ejemplo: $ STOP/ENTRY = 230 SYS$BATCH

Descontinúa de la cola SYS$BATCH al job número 230.

COMANDO SUBMIT.

Es usado para meter en cola uno o más procedimientos de comandos.

FORMATO: $ SUBMIT archivo

Ejemplo: $ SUBMIT PROGRAMA.COM

Mete el archivo de comandos PROGRAMA.COM a la cola SYS$BATCH que es la cola de BATCH que existe por default.

COMANDO TYPE.

Despliega el listado de uno o más archivos en la pantalla.

FORMATO: $ TYPE archivo

Ejemplo: $ TYPE MIARCHIVO.PAS

Listará el programa MIARCHIVO.PAS en la terminal (archivo SYS$OUTPUT).

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

LENGUAJE DE PROGRAMACION BASIC II

INTRODUCCION AL EDITOR EDT

NOVIEMBRE DE 1985

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

RESUMEN DE COMANDOS

EDITOR EDT EN LINEA

EDIT/EDT	Invoca al EDITOR y nos da el PROMPT (*) (Listo).
* HELP	Ayuda; explica como usar el HELP
* HELP < COMANDO >	Explica brevemente el COMANDO
* TYPE	Lista la línea actual.
* TYPE < # SECUENCIA	Lista la línea de esa secuencia.
* TYPE< LINF> <LSUP >	Lista de la secuencia inferior, hasta la secuencia superior. (LINF, LSUP).
* TYPE WHOLE	Lista del principio al [EOB]
* TYPE #SEC 1, # SEC 2, ...	Lista las SEC 1, 2 ...
* <# SECUENCIA>	Lista la línea con ese número de secuencia.
* INSERT	Mete texto, antes del cursor (a), pueden ser varias líneas o una sola. Termina con CTRL/Z.
* INSERT END	Se va meter el texto antes del [EOB] o sea, al final del texto que se tenía.

* REPLACE <# SECUENCIA>

Va a borrar esa línea (La que corresponde al # SECUENCIA) y va a permitir un INSERT.
(Dicho INSERT termina con CTRL/Z) y pueden ser varias líneas. La inserción se hará en la secuencia siguiente de la anterior borrada.
Ejemplo: REPLACE 3, se insertará en 2.1. 2.2 si la anterior hubiese sido dos.

* INSERT <# SECUENCIA>

Va a insertar texto antes de esa secuencia.
El texto termina con CTRL/Z.

* EXIT

Se sale del EDITOR/EDT y se salva el BUFFER. Me da además su tamaño.

* SUBSTITUTE !<TEXTO VIEJO>! <TEXTO NUEVO>!

El delimitador puede ser cualquier caracter especial menos $ ó &

Se sustituirá el texto viejo por el nuevo de la línea actual
Ejemplo:

```
* TYPE 2
   2     BAX  11/780

* SUBSTITUTE /B/V/
   2     VAX  11/780
```

* DELETE	Borra la línea actual. Ejemplo: * TYPE 2 2 VAX 11/780 * DELETE BORRA LA LINEA 2
* DELETE<# SECUENCIA>	Borra la línea numerada con esa secuencia.
* DELETE <LINF> THRU <LSUP>	Borra desde la secuencia inferior (LINF) hasta la secuencia superior (LSUP).
* DELETE BEFORE	Borra todas las líneas que están antes de la línea actual.
O P C I O N E S.	
* DELETE/QUERY	Me pregunta si va a borrar esa línea.
* RESEQUENCE	Renumera a las líneas, haciendo que todas las partes decimales (Ej. 2.1), queden como enteras. La resecuencia de uno es uno.
* WRITE <NOMBRE.TIPO>	Va a escribir el ARCHIVO MAIN (WORKFILE) en el archivo NOMBRE·TIPO, o sea, hace una copia del archivo MAIN a el archivo NOMBRE·TIPO

* INCLUDE <NOMBRE•TIPO> =<NAME> Va agregar el archivo (NOMBRE TIPO) al espacio de edición. Lo agrega en un BUFFER llamado (NAME) para este caso.

Ejemplo:

```
* INCLUDE  PATO•DAT = PATO
* INCLUDE  DONALD•DAT = DONALD
```

* SHOW BUFFER Nos va a mostrar los nombres de los BUFFERS existentes. El señalado con (=) será el archivo corriente. Al cual le afectarán los comandos.

Ejemplo:

```
* SHOW BUFFER

    = DONALD   10 LINES
      PATO     10 LINES
      MAIN     14 LINES
      PASTE     0 LINE
```

= <NAME> Cambia el buffer actual. Siendo ahora el actual buffer (NAME).

* COPY <# SEC 1> THRU <# SEC 2> TO<# SEC 3> Este comando copia el texto de un lugar a otro. (El texto no se borra del lugar del que fue copiado).

OPCION DEL COPY

* COPY FORMA DESEADA /QUERY El QUERY lo que hace es preguntarme si quiero que se realice la operación o no.

RESPUESTAS QUE PUEDE DAR:

Y (YES) - SI deseo que la línea mostrada sea copiada al destino.

N (NO) - SI no deseo que la línea mostrada sea copiada al destino.

A (ALL) - Copia el resto de las líneas al destino. Sin preguntar.

Q (QUIT) - Deja de copiar.

DESCRIPCION DE TECLAS:

DELETE	Borra el último caracter tecleado.
CTRL/S	Interrumpe el scrolling.
CTRL/Q	Reestablece el scrolling.
CTRL/Z	Es el caracter de control, el cual sirve para terminar un texto que esta siendo insertado.
CTRL/U	Borra una línea, desde el margen izquierdo hasta donde se encuentra el cursor.

NOTA: SCROLLING es el proceso de listar, desde la primer línea hasta el final.

CREACION DE ARCHIVOS.

- Para crear un archivo, invocamos al EDITOR, de la siguiente manera:

EDIT/EDT	Se invoca al EDITOR.
$ - file : < NOMBRE> . < TIPO>	Nos pregunta el nombre del archivo, si el archivo no existe (que es nuestro caso, ya que lo queremos crear), contesta.
Input File does not exit	
EOB	[EOB] (END OF BUFFER) Nos señala el fin del BUFFER. En este momento se ha creado un archivo.
INSERT	
* INSERT	Sirve inicialmente para llenar el archivo. Ya que a continuación se puede escribir un texto (el texto se sangra automáticamente).

Ejemplo:

* INSERT

Texto

-
-
-

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

LENGUAJE DE PROGRAMACION BASIC II

BASIC V A X - 11

Elaboradas por:

EDUARDO HERNANDEZ MEJIA
VICTOR MANUEL LEYVA ALATRISTE
JOSE ONTIVEROS JUNCO
LAURA SANDOVAL MONTAÑO

Recopiladas y editadas por:

JOSE ONTIVEROS JUNCO

NOVIEMBRE, DE 1985

BASIC VAX - 11

INTRODUCCION.

El BASIC, en la computadora VAX puede usarse de dos formas:
- como compilador normal
- como sistema interactivo (el medio ambiente es BASIC).

Cuando usemos el compilador BASIC, a nivel DCL,debemos:
- Crear un archivo, con algún editor, que contenga el programa fuente.
- Generar un módulo objeto del archivo fuente, con el comando de compilación $BASIC nombre_de_archivo.
- Generar una imagen ejecutable con el comando de DCL $LINK nombre
- Correr la imagen ejecutable con el comando de DCL $RUN nombre

Si existiera algún error en nuestro programa debemos corregir el programa con el editor y volver a seguir el proceso previamente descrito.

Cuando estemos en modo interactivo, el BASIC actuará similar al intérprete de las microcomputadoras, con la única diferencia que compila. Dentro de este modo podemos crear nuestro archivo (programa) tro de BASIC y ejecutarlo; en caso de existir error podemos editar línea o líneas, con el editor del BASIC y volverlo a ejecutar. Para ejecutar un programa podemos dar el comando RUN o RUNNH.

Existen en BASIC una serie de Calificadores, que se usan tanto a nivel DCL como a nivel interactivo, que modifican la acción específica de algunos comandos. La forma general será:
Instrucción/calificador(es).

Calificadores del comando BASIC a nivel DCL

/CHECK . checa, a tiempo de compilación, si existe un overflow aritmetico o un índice inválido.

/CHECK=[NO]BOUNDS
[NO]OVERFLOW
ALL
NONE

/CROSS genera una lista de referencia de las variables (CROSS REFERENCE).

y algunos más.

Un programa en BASIC consta de un conjunto de líneas numeradas que contienen instrucciones para el compilador BASIC. Una línea r tiene:
a Un número de línea
b) Una instrucción o mas(opcionales)
c) Un comentario (opcional)

s[illegible]que pueda exceder de 256 caracteres.

ELEMENTOS DE BASIC

números de linea.

Debe ser un número entero entre 1 y 32767 inclusive, y se utiliza para:
a) Indicar el orden de ejecución(secuencia de instrucciones)
b) Proporciona el control para saltos (GO TO)
C) Ayuda en el rastreo y carga de programas.

Cada número de linea debe ser único.

Para escribir comentarios, indique el inicio de éste con un signo de ! ; para poner varias instrucciones en una linea hay que separarlas con un simbolo de \ ; y un & seguido de la tecla return indica que la linea se continuará en otro renglón.

Palabras Reservadas (KEYWORDS)

Se usan para definir datos, ejecutar operaciones, invocar funciones,etc, y se pueden agrupar en :
a) Ejecutables (Print, Go To, Read, If, For, etc.)
b) No Ejecutables, (Data, Rem, Def, etc.)

s 'KEYWORDS' no pueden utilizarse como nombre de variable dentro d programa, ni tampoco pueden abreviarse. Algunas de estas 'KEYWORDS' estan formadas por más de una palabra; y los casos en lo referente a su escritura son los siguientes:
a) Con espacio opcional
GO TO, GO SUB, ON ERROR
b) Con espacio obligatorio
, MAT INPUT, MAT PRINT,INPUT LINE
c) Sin espacio
FNEND, SUBEND.

Comentarios

Existe en el lenguaje BASIC una 'KEYWORD' para indicar que lo que sigue, en la linea, es un comentario,y esta es REM. Puede estar en el inicio de una linea 10 REM EJEMPLO
o ser la última instrucción de una línea que contenga mas de una instruccion

```
20 A = 5
  \ B = 10
  \ REM COMENTARIO
```

Si terminado el comentario, pusiéramos una instrucción, esta no será reconocida como tal por el compilador. Otra forma de poner un comentario es con el simbolo !. No se deben escribir comentarios en las instrucciones DATA debido a que seran tratados como datos adicionales.

Conjunto de Caracteres

El BASIC usa todos los caracteres ASCII
letras de la A a la Z
dígitos del 0 al 9
caracteres especiales.

Tipos de Datos

SINGLE números de punto flotante almacenados en 32 bits
DOUBLE números de punto flotante almacenados en 64 bits
WORD números enteros de 16 bits (2 bytes)
LONG números enteros de 32 bits
STRING cadenas de caracteres, cada caracter se almacena en 1 byte.

Constantes

Las constantes en BASIC pueden ser de tres tipos:

a)Reales, número con ó sin punto decimal, en cualquiera de sus dos notaciones,
normal: -5.38 6.25 7
cientifica: .8415E-6 5E+3

b)Enteros número sin punto decimal seguido del simbolo %
29% -128%
El rango aceptable para un valor entero de de +- 32767% si se esta trabajando con el calificador WORD, o de +- 2147485647% con /LONG.

c)Strings es un conjunto de caracters encerrado entre " " o ' ' ejemplo: "HOLA TODOS",'CECAFI'

Variables

Las variables en BASIC pueden ser:

a)Reales, Nombre que se inicia en letra y puede tener 29 caracteres opcionales excepto guiones.
A, FAC_DE_ING, HOLA.HI

b)Enteras, Nombre que se inicia en letra, seguido de 28 caracteres opcionales y que termina en un %.
B%, ENTERO%
Trunca los valores reales que se les asignen.

c)Strings, Nombre que se inicia en letra, seguido de 28 caracteres opcionales y que termina en un $.
CADENA$, ALGO$.
Puede guardar hasta 65535 caracteres.

Expresiones

Las definiremos como operandos(constantes, variables, funciones o elementos de arreglos) unidos por operadores.
Los operadores pueden ser:

a)aritméticos
b)de strings
c)de relación
d)lógicos

a) Los operadores aritmeticos son:

+	adición	A+B
-	substracción	C% - 3%
*	multiplicación	2*ALGO
/	división	ALFA/BETA
^	potencia	CINCO ^ CUBO

Es posible combinar estas expresiones simples con otras, mediante los operadores, para hacer expresiones mas elaboradas. Al hacer esto es necesario seguir ciertas reglas para que la computadora pueda realizar exactamente las operaciones en la secuencia que le permita obtener el resultado que nosotros deseamos. Las reglas son:

i) la expresión se calculara de izquierda a derecha
ii) se valuarán en primer lugar todas las potencias
i en segundo lugar se valuarán las multiplicaciones o divisiones
ii...) en última instancia se evaluarán las sumas y restas...

En la expresión 3X2 - 5A/B + 9/Z, se calculara X2,
se multiplicará por 3,
se calcula 5A,
se divide este resultado entre B,
se calcula 9/Z,
y se procede a realizar la resta de 3X2 - 5A/B, y a esto se le suma 9/Z.

Si se desea modificar el orden de ejecución de las operaciones, podemos auxiliarnos de los parantesis ( ) cuyo contenido se calculara antes de las potencias; dentro de los parentesis se seguiran las reglas
establecidas para la evaluación de expresiones. Ejemplo: sea la expresion
algebraica

$$\frac{X - Y}{X + Y}$$

la expresion equivalente en BASIC sera: (X-Y)/(X+Y). Si en una expresion aritmetica existen varios grupos de parentesis, estos se iran resolviendo de adentro (interiores) hacia afuera (exteriores).

b. operadores de strings

Los strings en BASIC se pueden concatenar con el operador +

c) operadores de relación

Sirven para relacionar expresiones aritméticas o strings y obtener un resultado lógico (valores true o false), y son:

=	A = B	igual
<	A < B	menor
>	A > B	mayor
<=	A <= B	menor o igual
>=	A >= B	mayor o igual
<>	A <> B	diferente
==	A == B	aproximadamente igual, con los 4 bits mas significativos en /SINGLE o 12 en /DOUBLE
	A$==B$	será cierto si tienen posiciones y longitud iguales, sin blancos.

d) operadores lógicos

Sirven para unir resultados lógicos y son:

AND	producto lógico
OR	suma lógica
NOT	negación o complemento
XOR	OR exclusivo
EQV	equivalencia lógica (A EQUIV B es verdadera si A y B son iguales.)
IMP	implicación (A IMP B es falso si A es verdadero y B es falso).

Cabe hacer notar que en BASIC 0=false y -1=true. Las tablas de verdad de los operadores antes mencionados serían:

A	B	NOT A	OR	AND	EQV	XOR	IMP
0	0	-1	0	0	-1	0	-1
0	-1	-1	-1	0	0	-1	-1
-1	0	0	-1	0	0	-1	0
-1	-1	0	-1	-1	-1	0	-1

B A S I C V A X - 1 1

CARACTERISTICAS.

El compilador BASIC puede utilizarse de dos formas:

1) Como un compilador normal, o

2) Como un sistema interactivo.

El conjunto de caracteres usado es el empleado en el código ASCII.

Maneja cinco tipos de datos:

Punto Flotante:

1. Precisión sencilla (SINGLE): 32 bits.

2. Doble precisión (LONG): 64 bits.

Enteros:

3. Precisión sencilla (WORD): 16 bits.

4. Doble presición (LONG) 32 bits.

5. Alfanuméricos (STRINGS), pueden contener hasta 65,535 caracteres (bytes de 8 bits).

NOMBRES DE VARIABLES.

Pueden contener hasta 30 caracteres, el primero será una letra y los restantes pueden ser letras, dígitos, subguiones y/o puntos.

Ejemplo:

Tipo Real: SUELDO_DE_EMPLEADO
GRADOS_CENTIGRADOS

Tipo Entero: (Se finalizan con el caracter "%")
NUM_DE_ORDEN% TIPO.DE.REVISTA%

Tipo String: (Se finalizan con el caracter "$")
NOMBRE_ALUMNO$ NUM_DE_MATERIA$

ARREGLOS.

Pueden tener hasta dos dimensiones.

Se pueden definir como Reales, Enteros o Alfanuméricos.

La longitud depende del modo del arreglo:

- WORD de 0 a 32,767 elementos, y
- LONG de 0 a 2,147,483,647 elementos.

OPERADORES LOGICOS.

NOT Negación o Complemento.

AND Producto lógico.

OR Suma lógica.

XOR OR exclusivo.

EQV Equivalencia lógica (A EQV B es verdadera cuando A y B son iguales.

IMP Implicación (A IMP B es falso si A es verdadero y B es falso).

INSTRUCCIONES.

Los tipos de datos a los que se hace referencia en la sintaxis de las instrucciones, pueden ser:

1) WORD - Especifica un entero de 16 bits.

2) LONG - Especifica un entero de 32 bits.

3) INTEGER - Dafault al tipo de dato entero especificado al tiempo de compilación.

4) REAL - Default al tipo de dato punto flotante especificado al tiempo de compilación.

5) STRING - Especifica una cadena de caracteres.

CALL

Transfiere el control a un subprograma, opcionalmente se le pueden pasar parámetros.

El subprograma es compilado en forma independiente y puede ser escrito en cualquier otro lenguaje de VAX-11.

Los parámetros se pasan por:

- Dirección BY REF (default, excepto arreglos).
- Valor BY VALUE.
- Dirección de un descriptor BY DESC (para arreglos).

Sintaxis:

```
                  [  BY REF   ]                [  BY REF   ]                [  BY REF   ]
CALL nom-sub [ { BY DESC  } ]  (expr [ { BY DESC  } ] , [expr [ { BY DESC  } ] ...] )
                  [  BY VALUE ]                [  BY VALUE ]                [  BY VALUE ]
```

Ejemplo:

```
100 CALL SUB_PROGRAMA(R BY REF, D$()BY DESC)
```

CHANGE

Tiene la función de convertir una expresión alfanumérica a sus correspondientes valores decimales del código ASCII, o viceversa.

Sintaxis:

```
CHANGE { expr-alfa       } TO { arreglo-numérico }
       { arreglo-numérico }    { var-alfa        }
```

N O T A : El arreglo numérico debe ser unidimensional.

Ejemplo:

```
100 DIMENSION ARREGLO_NUMERICO(3)
200 !CONVIERTE DE EXP.ALFANUMERICA A NUMERICA
300 CHANGE "ABC" TO ARREGLO_NUMERICO
400 !CONVIERTE DE NUMEROS A CARACTERES
500 CHANGE ARREGLO_NUMERICO TO VAR_ALFA$
```

COMMON

Define una área de almacenamiento de datos que puede ser compartida por varios módulos de programa.

Sintaxis:

```
                                        var-num
                                        var-str [=exp-ent]
COMMON [(nom-area)]  [tipo-dato]  { arr-num(cte-num [,cte-num] )          }...
                                  { arr-str(cte-num [,cte-num] ) [=cte-num]
                                  { FILL-item
```

Ejemplo:

```
100 COMMON (NOM_DEL_AREA) VAR_NUM%,ALFA$=5%
```

EXTERNAL

Permite accesar rutinas del sistema operativo VAX/VMS y funciones que no estén definidas en el programa.

Sintaxis:

EXTERNAL tipo-dato [{ FUNCTION / CONSTANT }] nom-ext1 [,nom-ext2...]

Ejemplo:

```
100 EXTERNAL INTEGER FUNCTION SYS$CREMBX
200 SYS_STATUS%=SYS$CREMBX(,CANAL%,,,,,"LINK01")
```

N O T A : Si el programa accesa rutinas del sistema operativo el programa deberá ser compilado con el calificador /LONG

FNEND

Indica el fin físico y lógico de la definición de una función.

Ejemplo:

```
10 DEF FN.SENO(X)
20     FN.SENO=SIN(X)
30 FNEND
```

FNEXIT

Causa la salida de una función.

Ejemplo:

```
10 DEF FN.TANGENTE(REAL X)
20      IF X = 90 THEN FNEXIT
30      TANGENTE=TAN(X * PI)
40 FNEND
```

FOR

Esta instrucción nos permite construir ciclos iterativos (máximo 12 niveles de anidamiento).

1) FOR var-num=exp-num1 TO exp-num2 [STEP exp_num3]

2) FOR var-num=exp-num1 [STEP exp-num3] { WHILE / UNTIL } exp-log

3) instuc FOR var-num=exp-num1 TO exp-num2 [STEP exp-num3]

Ejemplo:

1)
```
10 FOR I%=0% WHILE I% < 10%
30      SUMA%=SUMA% + I%
90 NEXT I%
```

2)
```
10 SUMA%=SUMA%+1% FORI% = 1% TO 10%
```

FUNCTION

Indica el inicio de un subprograma como una función.

Sintaxis:

FUNCTION tipo-dato nom-sub ([tipo-dato] var [...])

N O T A : Acepta hasta 32 parámetros, se compila separadamente, finaliza con la instrucción FUNCTIONEND

Ejemplo:

```
100 FUNCTION INTEGER TRUNCA_DECIMAL (REAL X)
200         TRUNCA_DECIMAL=FIX(X)
300 FUNCTIONEND
```

FUNCTIONEND

Marca el fin de un subprograma tipo función y sólo podrá ser usada como tal; y regresa el control al programa que invocó la función.

FUNCTIONEXIT

Regresa el control de un subprograma función al programa que invocó a dicho subprograma.

Ejemplo:

```
100 FUNTION RAIZ_CUADRADA (REAL A)
110     IF A < 0 THEN FUNCTIONEXIT
120     RAIZ_CUADRADA=SQR(A)
130 FUNCTIONEND
```

MAP

Declara campos de datos en el registro y asocia a ellos con las variables del programa.

Sintaxis:

```
                                 var-num
                                 var-str [=exp-entera]
MAP (nom-map) [tipo-dato] { arr-num(cte-num [,cte-num] )                } [...]
                                 arr-str(cte-num [,cte-num] ) [=cte-num]
                                 FILL-item
```

Ejemplo:

```
15 MAP (REGISTRO) FILL%(4), ARREGLO(10,10), NOMBRE$=32
```

MAT

Permite crear implícitamente arreglos, asignar valores a elementos de arreglos, o redimensionar un arreglo. Además permite ejecutar operaciones, tales como multiplicación, suma, resta, transposición, inversión y obtención del determinante.

Sintaxis:

1)

MAT nom-arr = { CON / IDN / NUL$ / ZER } [(subs-exp1 [,subs-exp2])]

2)

MAT nom-arr1 = nom-arr2 { + / - / * } nom-arr3

3)

MAT nom-arr4 = (exp-num) * nom-arr5

4)

MAT nom-arr6 = { INV / TRN } (nom-arr7)

Ejemplos:

1) MAT MATRIZ_IDENTIDAD = IDN

2) MAT MATRIZ_PRODUCTO_A.B = A * B

3) MAT MULTIPLICA_ESCALAR = (3.1416) * ARREGLO

4) MAT INVERSA = INV A DETERMINANTE = DET

MAT INPUT

Asigna valores a los elementos de un arreglo desde una terminal o un archivo con formato-terminal.

Sintaxis:

```
MAT INPUT [ #exp-canal,] nom-arr [ (subs-exp1 [,subs-exp2] )] [,...]
```

Ejemplo:

```
1000 MAT INPUT DATOS%
```

MAT LINPUT

Recibe cadenas de datos desde una terminal o archivo con formato-terminal y se las asigna a los elementos del arreglo.

Sintaxis:

```
MAT LINPUT [#exp-canal,] nom-arr-str [(subs-exp1 [,subs-exp2] )]
```

Ejemplo:

```
1100 MAT LINPUT NOMBRE_EJEMPL$(10%)
```

MAT PRINT

Imprime el contenido de un arreglo en la terminal, o bien, los valores de cada elemento del arreglo en un registro de un archivo.

Sintaxis:

MAT PRINT [#exp-canal,] nom-arr [(subs-exp1 [,subs-exp2])] {;/,} [nom-arr...]

Ejemplo:

2535 MAT PRINT NOMBRE_EMPL$(10%)

MAT READ

Asigna valores a un arreglo desde una instrucción DATA.

Sintaxis:

MAT READ nom-arr [(subs-exp1 [,subs-exp2])] [,nom-arr...]

Ejemplo:

150 MAT READ MATRIZ%

ON ERROR GO BACK

Transfiere el control de un subprograma o DEF, al programa de dónde fue invocado, cuando ocurre un error.

Sintaxis:

{ON ERROR / ONERROR} GO BACK

Ejemplo:

10 ON ERROR GO BACK

RANDOMIZE

Nos permite variar la secuencia de números generados por la función RND, en cada corrida.

Síntaxis:

```
RANDOMIZE
```

Ejemplo:

```
30 RANDOMIZE
```

SLEEP

Suspende la ejecución del programa por un número de segundos determinado.

Síntaxis:

```
SLEEP exp-entera
```

Ejemplo:

```
10  I%=30
20  PRINT 'SE SUSPENDE LA EJECUCION POR';I%;"SEGS."
30  SLEEP I%
```

SUB

Indica el inicio de un subprograma Basic y especifica sus parámetros por número y tipo de dato.

El subprograma es compilado separadamente y permite hasta 32 parámetros.

Sintaxis:

```
SUB nom-sub ( [ tipo-dato ] { var
                            arreglo [ BY REF ] } [ ,... ] )
```

Ejemplo:

```
100  SUB SUB_PROGRAMA (A,B%,C$)
       .
       .
       .
500  SUBEND
```

SUBEND

Marca el fin de un subprograma Basic y regresa el control al programa que invocó al subprograma.

SUBEXIT

Permite regresar el control desde un subprograma al programa que hizo la llamada.

UNLESS

Sirve para ejecutar una instrucción solamente si una expresión de relación o lógica resulta falsa.

Síntaxis:

instrucción UNLESS { exp-rel / exp-log }

Ejemplo:

```
150 A=B UNLESS A>B
```

UNTIL

Nos sirve para realizar un ciclo iterativo o bien para modificar una instrucción, dependiendo de una condición dada.

Síntaxis:

1) UNTIL { exp-rel / exp-log }

```
[
|
|
[
NEXT
```

2) instr UNTIL { exp-rel / exp-log }

Ejemplo:

1)
```
10 UNTIL A=10
20        A=RND*10
30        PRINT A
40 NEXT
```

2)
```
100 LO.QUE.SEA=LO_OTRO + 1% UNTIL 1% = 15
```

WAIT

Especifica el número de segundos que el programa espera datos de la terminal.

Sintaxis:

```
WAIT exp-entera
```

Ejemplo:

```
300 WAIT 15%
350 INPUT "TIENES 15 SEG.PARA TECLEAR TUNOMBRE";A$
```

WHILE

Nos sirve para formar ciclos iterativos o modificar una instrucción mientras se cumpla una condición.

Sintaxis:

1) WHILE { exp-log / exp-rel }

[ ]

NEXT

2) instr WHILE { exp-log / exp-rel }

Ejemplos:

1) 100 WHILE I%< =100%

110 I% =I%+1

120 NEXT

2) 200 I%=I%+1 WHILE I%<=100%

FUNCIONES DEL LENGUAJE BASIC VAX 11/780

NOTA: Las funciones que no son comunes a las del lenguaje Basic de otras computadoras estan indicadas con el caracter "@"

ABS Proporciona el valor absoluto de una expresión numérica.

SINTAXIS: var-num = ABS(exp-num)

EJEMPLO:
```
100 VALOR_ABSOLUTO = ABS(-25.38)
110 !VALOR_ABSOLUTO = 25.38
```

ASCII Nos da el valor decimal del código ASCII (entre 0 y 255), correspondiente al primer caracter de una expresión caracter.

SINTAXIS: var-ent = ASCII(exp-car)

EJEMPLO: VALOR_ASC = ASCII("A") !VALOR_ASC=65

ATN Da el ángulo, en radianes, de una tangente especificada.

SINTAXIS: var-num = ATN(exp-num)

EJEMPLO: 120 ANG_RAD = ATN(TANGENTE)

CCPOS Da el caracter actual o posición del cursor sobre un canal (archivo) especificado.

SINTAXIS: var-num = CCPOS(canal)

EJEMPLO: 130 PANTALLA_CHAR = CCPOS(0%)

CHR$ Regresa el caracter correspondiente a un valor ASCII especificado.

SINTAXIS: var-str = CHR$(exp-ent)

EJEMPLO: 150 A$ = CHR$(65) ! A$="A"

@ COMP% Compara dos cadenas numéricas y da como resultado:

1	si	str-num1 > str-num2
0	si	str-num1 = str-num2
-1	si	str-num1 < str-num2

SINTAXIS: var-ent = COMP%(str-num1,str-num2)

EJEMPLO:
```
100 A% = COMP%('-12.5','-2.5')
120 ! A% = -1
```

COS Porporciona el coseno de un ángulo especificado en radianes.

SINTAXIS: var-num = COS(exp-num)

EJEMPLO: 320 COSENO_45G = COS(PI/4)

@ DATE$ Proporciona la fecha de la forma: dd-mmm-yy.

SINTAXIS: var-str = DATE$(0%)

EJEMPLO: 110 PRINT DATE$(0%)

@ DET Regresa el valor del determinante de la última matriz invertida con la instrucción MAT INV.

SINTAXIS: var-num = DET

EJEMPLO: 180 DETERMINANTE = DET

@ DIF$ Da como resultado una cadena de caracteres, cuyo contenido es la diferencia entre dos cadenas de caracteres numéricos

SINTAXIS: var-str = DIF$(exp-str1,exp-str2)

EJEMPLO: 500 RESTA_CHAR$ = DIF$('128','38')
600! RESTA_CHAR$ = '90'

ERL Nos da el número de la línea en la que ocurrió el último error.

SINTAXIS: var-ent = ERL

EJEMPLO: 300 PRINT "ERROR EN LA LINEA >> ";ERL

@ ERN$ Regresa el nombre del programa, subprograma, función(subprograma) o función definida con la instrucción DEF, donde ocurrió el último error.

SINTAXIS: var-str = ERN$

EJEMPLO: 250 PRINT ' ERROR EN EL MODULO >> ';ERN$

ERR Proporciona el código del último error ocurrido.

SINTAXIS: var-ent = ERR

EJEMPLO: 260 CODIGO_ERROR% = ERR

@ ERT$ Da una breve explicación referente a el código de error especificado.

SINTAXIS: var-str = ERT$(exp-ent)

EJEMPLO: 270 EXPLICA.ERROR$ = ERT$(CODIGO_ERROR%)

EXP Proporciona el valor de la función exponencial "e" elevada a una potencia especificada.

SINTAXIS: var-num = EXP(exp-num)

EJEMPLO:
```
400 FUNC_EXPONENCIAL = EXP(4)
```

FIX Se utiliza para obtener la parte entera de un número real.

SINTAXIS: var-num = FIX(exp-num)

EJEMPLO:
```
NUM.ENTERO% = FIX(NUM.REAL)
```

@ FORMAT$ Convierte un valor numérico a una cadena representada en un formato especificado (Las reglas para especificar el formato son las mismas que se usan para imprimir números con la instrucción PRINT USING).

SINTAXIS: var-str = FORMAT$(exp-num,exp-str)

EJEMPLO:
```
110 FORM_1$ = FORMAT$(5/32,"##.##")
  \ !FORM_1$=' 0.16'
```

INT Regresa un número real igual al más grande de todos los números enteros menores que o igual a un número especificado.

SINTAXIS: var-num= INT(exp-num)

EJEMPLO:
```
500 I = INT(3.3)   ! I=3
600 J = INT(-3.3)  ! J=-4
```

LEFT$ Extrae un subcadena de una cadena, avanzando de izquierda a derecha, sin alterar la cadena.

SINTAXIS: var-str = LEFT$(exp-str,exp-ent)

EJEMPLO:
```
300 SUB_STR$ = LEFT$('VICTOR',3%)
  \ !SUB_STR$='VIC'
```

LEN Regresa el número de caracteres que contiene una cadena.

SINTAXIS: var-ent = LEN(exp-str)

EJEMPLO:
```
200 LONG_CADENA = LEN('VICTOR MANUEL')
    !LONG_CADENA = 13
```

@ LOC Regresa una palabra entera de cuatro bytes (longword), en la cual especifica la dirección virtual de una variable o símbolo.

SINTAXIS: var-ent = LOC(símbolo)

EJEMPLO:
```
100 DECLARE INTEGER SIMBOLO
200 DIR_VIR% = LOC(SIMBOLO)
```

LOG Regresa el logaritmo natural de un número especificado.

SINTAXIS: var-num = LOG(exp-num)

EJEMPLO:
```
LOG_NAT = LOG(EXP(1))  !LOG_NAT=1
```

LOG10 Regresa el logaritmo decimal de un número especificado.

SINTAXIS: var-num = LOG10(exp-num)

EJEMPLO: LOG_DEC = LOG10(300)

@ MAR Regresa el ancho del marsen de un canal (archivo) especificado.

SINTAXIS: var-ent = MAR(canal)

EJEMPLO: 300 REC_LEN = MAR(0%)

MID$ Extrae una subcadena de una cadena, sin alterar esta última.

SINTAXIS: var-str = MID$(exp-str,exp-ent1,exp-ent2)

EJEMPLO: LAST_NAME$ = MID$('VICTOR LEYVA',7%,5%)
!LAST_NAME$ = 'LEYVA'

@ NOECHO Deshabilita el eco sobre un canal especificado.

SINTAXIS: var-ent = NOECHO(canal)

EJEMPLO: 900 SIN_ECO = NOECHO(0%)

@ NUM Regresa el número del renslón del último elemento tranferido dentro de un arreslo con una instrucción MAT de entrada/salida.

SINTAXIS: var-ent = NUM

EJEMPLO: NUM_RENGLON% = NUM

@ NUM2 Regresa el número de la columna del último elemento transferido dentro de un arreslo por medio de una instrucción MAT de entrada/salida.

SINTAXIS: var-ent = NUM2

EJEMPLO: NUM_COLUMNA% = NUM2

@ NUM$ Convierte una expresión numérica en una cadena de caracteres numéricos.

SINTAXIS: var-str = NUM$(exp-num)

EJEMPLO:
```
100 NUM_STR$ = NUM$(3.15*5)
    !NUM_STR$=' 15.75'
200 A$ = NUM$(15800*389) !A$=' .61462E+07'
```

@ NUM1$ Cambia una expresión numérica a una cadena de caracteres numéricos sin incluir espacios al principio (en el caso de números positivos) y sin producir la notación E.

SINTAXIS: var-str = NUM1$(exp-num)

EJEMPLO:
```
300 NUM_STR1$ = NUM1$(3.15*5)
    !NUM_STR1$='15.75'
400 B$ = NUM1$(15800*389) !B$='6146200'
```

@ PLACE$ Cambia la precisión de una cadena número rica. BASIC redondea o trunca la cadena numérica, dependiendo del argumento numérico (ver Tabla A).

SINTAXIS: var-str = PLACE$(str-num,exp-ent)

EJEMPLO:
```
700 NUMERO$ = PLACE$('12345.67891',0%)
720 ! NUMERO$='12346'
```

@ POS Busca una subcadena dentro de una cadena de caracteres, y regresa la posición donde empieza la subcadena.

SINTAXIS: varent = POS(exp-str,exp-substr,exp-ent)

EJEMPLO:
```
100 INDICE = POS(REGALUM$,'1234567890',9%)
    !busca en REG_ALUM$ a partir del caracter
    !nueve en que posición inician los números.
```

@ PROD$ Realiza el producto entre dos cadenas de caracteres numéricos y el resultado lo da como una cadena de caracteres numéricos. La precisión del resultado depende del argumento entero (ver Tabla A).

SINTAXIS: var-str = PROD$(str-num1,str-num2,exp-ent)

EJEMPLO: 300 MULTIPLICA$ = PROD$('-1.3E-5','83',0%)

@ QUO Da como resultado una cadena numérica, la cual es el cociente de dos cadenas numéricas. La precisión del resultado depende del argumento entero (ver Tabla A).

SINTAXIS: var-str = QUO$(str-num1,str-num2,exp-ent)

EJEMPLO: 200 DIV$ = QUO$(NUMERADOR$,DENOMINADOR$,0%)

RIGTH$ Extrae una subcadena de la parte derecha de una cadena de caracteres, sin alterar el contenido de esta última.

SINTAXIS: var-str = RIGHT$(exp-str,exp-ent)

EJEMPLO: 600 SUB_STR$ = RIGHT$('1234567890',5%)

RND Se utiliza para generar números aleatorios. Esta función genera un número mayor que o igual a cero y menor que uno. Se recomienda que se utilize la instrucción RANDOMIZE, precediendo a esta función, ya que de otra forma siempre se generará la misma secuencia de números aleatorios.

SINTAXIS: var-num = RND

EJEMPLO:
```
100 RANDOMIZE  \  NUM.ALEA = RND
```

**@ SEG$** Extrae una subcadena de una cadena de caracteres, sin alterar esta última. Esta función incluye las funciones LEFT$, MID$, RIGHT$.

SINTAXIS: var-str = SEG$(exp-str,exp-ent1,exp-ent2)

EJEMPLO:
```
600 REG_ALUM$ = "12345678332FRANCISCO L. M."
700 CVE_CARR$ = SEG$(REGALUM$,9%,10%)
800 ! CVE_CARR$="32"
```

SGN Evalua una expresión numérica, y da como resultado:
```
 1 si la expresión es positiva
 0 si la expresión es igual a cero
-1 si la expresió es negativa
```

SINTAXIS: var-ent = SGN(exp-num)

EJEMPLO: SIGNO = SGN(A*B-C)

SIN Regresa el seno de un ángulo especificado en radianes.

SINTAXIS: var-num = SIN(exp-num)

EJEMPLO: 333 SENO_90.GRADOS = SIN(PI/2)

SPACE$ Proporciona una cadena, cuyo contenido es un número especificado de espacios.

SINTAXIS: var-str = SPACE$(exp-ent)

EJEMPLO: 444 TIPO_ERROR$ = SPACE$(20%)

SQR Da como resultado la raiz cuadrada de una número.

SINTAXIS: var-num = SQR(exp-num)

EJEMPLO: 555 SQUARE_ROOT = SQR(X1)

@ SUM$ Regresa una cadena de caracteres numéricos, cuyo contenido es el resultado de la suma de dos cadenas numéricas.

SINTAXIS: var-str = SUM$(exp-str1,exp-str2)

EJEMPLO: 600 SUMA_STR$ = SUM$('3',"-23")
! SUMA_STR$ = "-20"

@ SWAP% Transpone los dos bytes menos significativos de un número entero.

SINTAXIS: var-ent = SWAP%(exp-ent)

EJEMPLO: 500 S_25% = SWAP%(3%)

TAB Se usa en combinación con la instrucción PRINT, para mover el cursor de impresión a una columna especificada.

SINTAXIS: PRINT TAB(exp-ent)

EJEMPLO: 700 PRINT TAB(32%),NOMBRE$

TAN Regresa la tangente de un ángulo expecificado en radianes.

SINTAXIS: var-num = TAN(exp-num)

EJEMPLO: 550 TANGENTE_360.GRADOS = TAN(2*PI)

@ TIME Proporciona la medida del tiempo como un número real.

SINTAXIS: var-num = TIME(exp-num)

Si <exp-num> es:	La función TIME regresa:
0	Los segundos transcurridos desde la media noche.
1	El tiempo de CPU en décimas de segundos del Job actual.
2	El tiempo en minutos que ha estado en sesion el Job actual.
3	Regresa cero.
4	Regresa cero.

EJEMPLO: 200 PRINT 'SEGS. TRANSCURRIDOS ';TIME(0)

TIME$ Regresa una cadena de caracteres, la cual muestra la hora del día de la forma: HH:MM AM o HH:MM PM.

SINTAXIS: var-str = TIME$(exp-ent)

EJEMPLO: PRINT " Son las: ";TIME$(0%)

@ TRM$ Borra los blancos que se encuentran al final de una cadena.

SINTAXIS: var-str = TRM$(exp-str)

EJEMPLO: 780 A$ = TRM$(B$)

@ VAL Regresa el valor en punto flotante de una cadena numérica.

SINTAXIS: var-num = VAL(str-num)

EJEMPLO: 300 VALOR_REAL = VAL(NUMEROS$)

@ VAL% Regresa el valor entero de una cadena numérica.

SINTAXIS: var-ent = VAL%(str-ent)

EJEMPLO: 340 VALOR_ENTERO% = VAL%("1982")

@ XLATE Traduce una cadena de caracteres a otra haciendo referencia a una tabla de caracteres.

SINTAXIS: var-str = XLATE(exp-str1,exp-str2)

EJEMPLO: 500 OUT_STR$ = XLATE(IN_STR$,TABLA$)

NOTA: <exp-str2> es la tabla de caracteres y puede contener hasta 256 caracteres, numerados del 0 al 255.

TABLA A. Valores de truncación y redondeo.
(El número usado es 12345.67891)

<exp-ent>	E f e c t o	Valor regresado
-5	Redondea a 100,000s	0
-4	Redondea a 10,000s	10000
-3	Redondea a 1000s	12000
-2	Redondea a 100s	12300
-1	Redondea a 10s	12350
0	Redondea a unidades	12346
1	Redondea a décimos	12345.7
2	Redondea a centécimos	12345.68
3	Redondea a milésimos	12345.679
4	Redondea a diez-milésimos	12345.6789
5	Redondea a cien-milésimos	12345.67891
9995	Trunca a 100,000s	0
9996	Trunca a 10,000s	10000
9997	Trunca a 1000s	12000
9998	Trunca a 100s	12300
9999	Trunca a 10s	12340
10,000	Trunca a unidades	12345
10,001	Trunca a décimos	12345.6
10,002	Trunca a centésimos	12345.67
10,003	Trunca a milésimos	12345.678
10,004	Trunca a diez-milésimos	12345.6789
10,005	Trunca a cien-milésimos	12345.67891

MANEJO DE ARCHIVOS

REGISTROS

Existen dos tipos de registros, los de longitud fija y los de longitud variable.

Los registros de longitud fija utilizan menos recursos que los de longitud variable, pero usan en forma ineficiente el espacio en memoria; en cambio, los registros de longitud variable utilizan eficientemente el espacio en memoria pero necesitan mas recursos en su procesamiento.

ORGANIZACION DE ARCHIVOS

Los tres tipos de organización mas importantes son:

- -secuencial
- -relativo
- -indexado.

NOMBRES DE ARCHIVOS

El nombre completo de un archivo, en la computadora VAX, consta de Nodo::Dispositivo:[Directorio]Nombre.Tipo;Versión

donde:

Nodo es cuando se tienen redes de computadoras.

Dispositivo es el nombre del dispositivo físico que contiene al archivo.

Directorio es el nombre del directorio donde se localiza el archivo.

Nombre es el nombre del archivo.

Tipo es una extensión del nombre del archivo: y generalmente, para datos, es de tipo DAT.

Versión es el número de la versión del archivo.

NOMBRES LOGICOS EN NOMBRES DE ARCHIVOS

En VAX se pueden definir nombres lógicos, a nivel DCL, para las especificaciones de archivos o dispositivos físicos.

Ejemplos:

Definiendo un nombre lógico para una especificación de archivo:

```
$ DEFINE PAYR  DBA1:[SENDER]PAYROL.DAT
```

Definiendo un nombre lógico a un dispositivo físico:

```
$ DEFINE DISK DBA0:
```

Una vez definidos, los podemos referenciar en nuestro programa

```
10 OPEN "PAYR" FOR OUTPUT AS FILE #1%, &
            ORGANIZATION SEQUENTIAL
20 OPEN "DISK" FOR INPUT AS FILE #2%
```

DEFAULTNAME

Se usa en la instrucción OPEN y asume que un archivo que se utiliza tiene especificaciones correctas de archivo, aun en los casos en que se no proporcione el nombre completo del mismo.

Ejemplo:

```
10 LINPUT "Nombre de archivo"; ARCHIVO$
20 OPEN ARCHIVO$ FOR INPUT AS FILE #3%, &
   ORGANIZATION SEQUENTIAL, DEFALTNAME ".DAT"
```

DISPOSITIVOS COMO NOMBRE DE ARCHIVOS

Se debe asignar el dispositivo, desde DCL, antes de leer o escribir.

```
$ALLOCATE  CR1:
$BASIC
NEW CRTN
50  MAP (DNG) AZ=0%
100 OPEN "CR1:" FOR INPUT AS FILE #1%, ACCESS READ,MAP DNG
110 GET #1%
```

INSTRUCCION OPEN

```
OPEN especificacion_de_archivo [ < FOR INPUT >] AS FILE #cana]
                               [ < FOR OUTPUT>]

[,ORGANIZATION] < SEQUENTIAL >  [< FIXED   >]     ]
                < INDEXED    >  [<VARIABLE>]
                < RELATIVE   >


[,ACCESS  <READ    >    ]  [,ALLOW  <NONE   >  ]
          <WRITE   >                 <READ   >
          <MODIFY  >                 <WRITE  >
          <SCRATCH>                  <MODIFY>
          <APPEND  >

[,RECORDSIZE expresión_entera_1]
[,FILESIZE   expresión_entera_2]
[,TEMPORARY]       [,DEFAULTNAME expresión_strins]
[, MAP nombre_del_mapa]
```

Se usa FOR INPUT para archivos que existen y FOR OUTPUT para archivos que van a crearse. Si se omiten ambas el archivo puede o no existir.

ARCHIVOS SECUENCIALES

Los archivos secuenciales contienen archivos 'virtualmente' contiguos, guardados en el orden en que fueron creados. No se pueden compartir archivos secuenciales para escribir, pero si para leer.

-Operaciones en archivos secuenciales.

PUT.- Transfiere datos desde el registro que esta en el 'buffer'al archivo. Se usa para escribir registros por primera vez.
Se pueden cambiar registros con UPDATE) solo se pueden escribir nuevos registros al final del archivo.
Se puede ir al final del Archivo directamente indicando ACCESS APPEND en el OPEN.

```
1000 MAP(BUFF) CODIGO$=3,NOMBRE$=20
1010 OPEN "ARCH.DAT" FOR INPUT AS FILE #2%,&
          ORGANIZATION SEQUENTIAL, &
          ACCESS APPEND,&
          MAP BUFF
1020 FOR I%=1% TO 50%
1030 INPUT "DAME CODIGO";CODIGO$
1040 INPUT "DAME NOMBRE";NOMBRE$
1050 PUT #2%
```

Cuando se procesan registros de longitud variable se puede usar la clausula COUNT que especifica el número de 'bytes' escritos. Por ejemplo:

```
          110 PUT #3%,COUNT 60%
```

FIND.- Localiza registros pero no mueve la información al 'buffer'. Se usa FIND para checar la existencia de un registro y coloca el apuntador de registros en el registro buscado, despues de lo cual, se puede ejecutar una operación de UPDATE.

```
FIND #canal
```

GET .- Lee un registro desde el archivo y lo deja en el 'buffer'.

```
GET #canal
```

UPDATE.- Reemplaza el contenido del resistro apuntado; el resistro nuevo debe ser del mismo tamaño del que reemplazara. Los archivos tienen que estar en disco y se debe ejecutar un GET antes de un UPDATE. Ejemplo:

```
10    ON ERROR GO TO 999
20    MAP(AAA) NOMBRE$=60%,NUM$=8%
30    OPEN "ARCH.DAT" FOR INPUT AS FILE#7%, &
          SEQUENTIAL,MAP AAA
50    INPUT "NOMBRE BUSCADO";NOM$
70    GET #7% WHILE NOM$ <> NOMBRE$
90    INPUT "número";NUM$
100   UPDATE #7%
110   GO TO 50
999   RESUME 1000
1000  CLOSE #7%
1010  PRINT "ACTUALIZACION REALIZADA"
1020  END
```

SCRATCH.- Borra todos los resistros desde donde se encuentra el apuntador de resistro hasta el final del archivo. Para hacer esto, hay que especificar ACCESS SCRATCH en el OPEN.

En archivos secuenciales, si no se especifica el tamaño de resistro (RECORDSIZE), se le asigna 132.

ARCHIVOS RELATIVOS.

Se puede accesar un resistro secuencialmente o en forma directa (al azar); de acuerdo a su posición en el archivo. No se puede crear un archivo relativo sin haber definido una longitud máxima de resistro. Si se utilize MAP se declara implicitamente el tamaño del resistro; si no se utilize MAP hay que utilizar el RECORDSIZE.

Operaciones con archivos relativos.

PUT: a) PUT #canal [,COUNT expresión_entera]
se use para acceso secuencial.
b) PUT #canal,RECORD expresión_entera [,COUNT exp_entera]
se use para acceso directo

Cuando se emplean resistros de longitud variable, se debe especificar COUNT.

Ejemplo:

```
100 PUT #10%, RECORD 134%
```

escribe el nuevo resistro en la localidad especificada por el número RECORD(134). Si se existe, marca el error 153%.

FIND: a) FIND #canal
b) FIND #canal, RECORD expresión_entera

Ejemplo:

```
70 FIND #5%, RECORD 26%
```

Si no existe el registro 26, se marca el error 155; el FIND se usa cuando la siguiente operación es GET, DELETE o UPDATE.

GET: a) GET #canal
b) GET #canal, RECORD expresión_entera

El GET trae la información del registro apuntado, y modifica el valor del apuntador a expresión_entera + 1.

Ejemplo:

```
10 GET #2%, RECORD 10%
20 GET #2%
```

En la linea 10 se lee el registro 10 y en la linea 20 se lee el registro 11.

UPDATE.- Escribe un nuevo registro en la localidad indicada por el apuntador de registro. Se puede hacer un UPDATE solo despues de un GET por lo que no es necesario especificar la clausula RECORD.

Se debe usar COUNT para especificar el tamaño del nuevo registro si es diferente del último registro accesado del archivo; se puede especificar una longitud igual a la de la última operación de entrada con RECOUNT. Ejemplo:

```
50 GET #3%, RECORD 404%
60 INPUT "NEW DATA";NEW.DATA$
70 UPDATE #3%, COUNT RECOUNT
```

La linea 70 escribe un registro del tamaño de NEW.DATA$. Una operación con FIND no indica el tamaño del registro.

DELETE.- Borra el registro. Se debe usar el GET o FIND antes de un DELETE. Ejemplo:

```
40 FIND #1%,RECORD 67%
50 DELETE 1%
```

Localiza el registro 67 y lo borra. Debido a que en si no se borra la localidad, se puede hacer un PUT despues de borrar el registro

```
60 PUT #1%, RECORD 67%.
```

ARCHIVOS INDEXADOS

Los archivos indexados contienen registros guardados en orden ascendente de la primera llave de acceso. Pueden contener mas de un índice (hasta 254). No se puede crear un archivo indexado sin definir el registro del 'buffer' y sin especificar el campo de la llave (la posición que guarda en el registro).

Las llaves se asignan en el OPEN y van acompañadas de un MAP en donde se definen los campos del registro.

```
10 MAP(INV) RECNO$,INV.NUM%,PROJECT.NUM$=8% ,    &
            JOB.SUPERVISOR$=32%
20 OPEN "INDEX.DAT" FOR OUTPUT AS FILE # 8%,   &
                    ORGANIZATión INDEXED FIXED,   &
                    PRIMARY KEY RECNO$,   &
                    ALTERNATE KEY INV.NUM%,   &
                    ALTERNATE KEY PROJECT.NUM$,   &
                    ALTERNATE KEY JOB.SUPERVISOR$,   &
                    MAP INV
```

BASIC asume que no hay llaves repetidas; si existen en el archivo, hay que especificarlo en el OPEN. Ejemplo:

```
PRIMARY KEY RECNO$ DUPLICATES;
```

Se puede tambien cambiar los valores de las llaves (excepto la primera) especificando CHANGES con DUPLICATES

```
ALTERNATE KEY INV.NUM% DUPLICATES CHANGES'
```

Operaciones con archivos indexados(tienen las mismas funciones que en los otros tipos de archivos).

```
PUT.- PUT #canal [, COUNT expresión_entera]

FIND:a) FIND #canal
                    acceso secuencial

      b) FIND # canal, KEY #expresión_entera {EQ} {exp_string}
                                             {GE} {exp_entera}
                                             {GT}
```

acceso indexado, en donde:

```
- KEY #expresiónentera = es el número de la llave
                    0 = número de la primera llave
                    1 = número de la primera llave alterna
                    2 = número de la segunda llave alterna
                    y asi sucesivamente.
- EQ  busca el primer registro con una llave igual
                    al string o valor entero.
- GT  busca el primer registro con una llave mayor
                    al string o valor entero.
- GE  busca el primer registro con una llave igual
                    o mayor al string o valor entero.
```

Ejemplo:

```
150 FIND #3%, KEY 0% EQ "JORGE"
```

Si no lo encuentra marca el error ERR=155

GET: a) GET #canal
b) GET #canal, KEY #exp_entera (GE) (exp_string)
(EQ) (exp_entera)
(GT)

Ejemplo:

```
100 GET #4%, KEY #0% GT "LAURA"
```

UPDATE #canal

Cuando se permiten archivos con llaves duplicadas, el nuevo registro debe ser del mismo tamaño que el anterior. Cuando no se permiten llaves primarias duplicadas el nuevo registro:

Puede no tener la longitud máxima del registro.
Debe incluir al menos el campo de la primera llave.

DELETE #canal
Un FIND o un GET deben preceder al DELETE.

```
10 FIND #2%, KEY #0% EQ "521-56-83"
20 DELETE #2%
```

RESTORE #canal, KEY #expresión_entera

Guarda el archivo especificando el tipo de llave para su acceso. Ejemplo:

```
1000 RESTORE #3%, KEY #0%
```

OPERACIONES CON ARCHIVOS

NAME.- Nos permite modificar el nombre del archivo a accesar.
NAME exp_string-1 AS exp_string-2

donde:
exp_string-1 es el nombre anterior del archivo
exp_string-2 es el nombre nuevo del archivo.

CLOSE.- Nos permite cerrar los archivos para terminar el programa y protegerlos. BASIC cierra los archivos cuando encuentra un END o llega a la última línea del programa.

CLOSE [#]canal [,canal] ...

RESTORE.- Regresa el apuntador de registro al principio del archivo.

RESTORE #canal

KILL.- Si la protección lo permite, se puede borrar un archivo.

KILL nombre_de_archivo

FREE y UNLOCK.- permite que dos o mas usuarios utilicen el archivo al mismo tiempo, siempre y cuando la protección lo permita.

Con FREE #canal, se da acceso libre a todos los registros previamente no libres. Ejemplo:
10 FREE #8%

Con UNLOCK #canal, se da acceso solo al último registro accesado. Ejemplo:
10 UNLOCK #8%

ACCESS y ALLOW (en el OPEN)

Con estas instrucciones se permite compartir el acceso a archivos.

ALLOW NONE .- no permite el acceso a otros usuarios. Es el valor de 'default' si no se usa ACCESS READ
ALLOW READ .- permite la lectura del archivo. Es el valor de 'default' si se usa ACCESS READ.
ALLOW MODIFY.- permite acceso completo a otros usuarios.
ALLOW WRITE .- permite a otros escribir en el archivo, pero previene las operaciones de UPDATE y DELETE.

```
ACCESS READ .- Permite solo GET y FIND.
ACCESS WRITE.- Permite solo PUT.
ACCESS MODIFY- Permite GET, FIND, PUT, UPDATE y DELETE
               en archivos relativos e indexados (DEFAULT).
ACCESS SCRATCH Permite GET, FIND, PUT, UPDATE y SCRATCH
               solo en archivos secuenciales.
ACCESS APPEND- Permite el PUT al final de un archivo
               secuencial, si este se encuentra en disco.
```

S O R T E N B A S I C

43

Para sortear un archivo desde un programa en BASIC existen dos modos:

1. Sortear el archivo tal y como esta definido c/u de sus registros (FILE INTERFACE).

2. Sortear selectivamente los registros de un archivo (RECORD INTERFACE).

FILE INTERFACE.

Para este modo se utilizan las siguientes rutinas externas:

SOR$PASS_FILES

SOR$INIT_SORT

SOR$SORT_MERGE

SOR$END_SORT

Para efectuar el sorteo se deben seguir los siguientes pasos:

a) Declarar las rutinas antes mencionadas como funciones enteras externas.

```
10  EXTERNAL  INTEGER  FUNCTION  SOR$PASS_FILES

20  EXTERNAL  INTEGER  FUNCTION  SOR$INIT_MERGE
```

b) Efectuar el llamado de la rutina SOR$PASS_FILES

<variable entera> = SOR$PASS_FILES ('<nombre archivo de entrada>',
<nombre archivo de salida>'

Esta rutina pasa los archivos de entrada y salida al SORT; si existen multiples archivos de entrada, es decir si se desean sortear varios archivos del mismo tipo, se deberá efectuar la llamada tantas veces como archivos de entrada haya. Estos archivos no deben estar abiertos.

Si resulto con éxito la rutina, la variable entera contendrá valor verdadero (1%).

c) Efectuar la llamada a SOR$INIT_SORT

<variable entera> = SOR$INIT_SORT (<nombre de registro>)

Nombre de registro: Registro declarado en un MAP y contendrá la llave y las opciones para el sorteo.

Ejemplo:

```
MAP (LLAVE1) WORD NO_LLAVES, TIPO_LLAVE, ORDEN, POS_INIC,
                  LONG_LLAVE

NO_LLAVES = 1%    !Número de llaves

TIPO_LLAVE = 1%   ! 1 indica que es tipo caracter

ORDEN = 0%        ! 0% = ordena ascendentemente; 1% = descendentemente.

POS_INIC = 1%     ! posición inicial de la llave
```

```
LONG_LLAVE = 4%   ! longitud de la llave
```

Por cada llave hay que indicar su tipo, orden, posición inicial y longitud.

d) Efectuar la llamada a SOR$SORT_MERGE

```
<variable entera> = SOR$SORT_MERGE
```

Esta rutina efectúa el sorteo.

e) Efectuar la llamada a SOR$END_SORT

```
<variable entera> = SOR$END_SORT
```

Esta rutina finaliza el sort y libera las áreas de trabajo del SORT.

RECORD INTERFACE.

Para el uso de este modo se tienen las siguientes rutinas:

SOR$INIT_SORT
SOR$RELEASE_REC
SOR$SORT_MERGE
SOR$RETURN_REC
SOR$END_SORT

Con este MODO podemos sortear selectivamente los registros de un archivo. El proceso es el siguiente:

a) Declarar las rutinas antes mencionadas como funciones enteras externas.

Ejemplo:

```
10   EXTERNAL   INTEGER   FUNCTION   SOR$INIT_SORT
```

b) Efectuar el llamado de la rutina SOR$INIT_SORT

```
<variable entera> = SOR$INIT_SORT (<variable1>,<variable2>)
```

En donde:

variable 1 = Es el nombre del campo que esta definido en 'MAP' de llaves y que contiene el número de llaves.

variable 2 = Es la variable que contiene la longitud máxima del registro que se va a sortear.

variable entera = Contendrá el valor de verdadero si la rutina tuvo éxito.

c) Efectuar el llamado de la rutina SOR$RELEASE_REC.

```
<variable entera> = SOR$RELEASE_REC (<variable string>)
```

En donde:

variable string =<llave 1> + <llave 2> + <datos> + <llave 1> + <llave 2>

Obsérvese que las llaves van repetidas ya que cuando se recupere este registro no se pueden obtener los campos definidos como llaves en el 'MAP' de llaves.

Esta rutina se llamará tantas veces como registros a sortear hayan.

d) Efectuar el llamado a la rutina SOR$SORT_MERGE

<variable entera> = SOR$SORT_MERGE

Con esta rutina se sortean los registros accesados con SOR$RELEASE_REC.

e) Efectuar el llamado a la rutina SOR$RETURN_REC

<variable entera> = SOR$RETURN_REC (<variable string>,
<variable entera>)

En donde:

<variable string> = En esta variable se recupera el registro.

<variable entera> = En esta variable se obtendrá el tamaño del registro.

Con esta rutina se recupera registro por registro ya sorteados.

Se debe llamar tantas veces como registros se quieran recuperar.

f) Efectuar el llamado a la rutina SOR$END_SORT.

<variable entera> = SOR$END_SORT

Esta rutina limpia áreas de trabajo del SORT.

M E R G E E N B A S I C

El utility MERGE toma de dos a diez archivos similares de entrada sor teados y efectúa el MERGE con ellos de acuerdo a las llaves especificadas, y genera un archivo de salida.

Al igual que SORT, el MERGE utiliza rutinas externas que deberán proveer:

- El número de archivos a ser mezclados.
- Especificaciones de archivos de entrada y salida.
- Información acerca de las llaves.
- La rutina de entrada.

El MERGE puede trabajar en dos MODOS:

- FILE INTERFACE
- RECORD INTERFACE

FILE INTERFACE.

La mezcla se efectuará entre archivos.

Para procesar el MERGE en este MODO se deben seguir los siguientes pasos:

a) Declarar las rutinas que utiliza el MERGE como funciones enteras externas.

```
EXTERNAL INTEGER FUNCTION SOR$PASS_FILES

EXTERNAL INTEGER FUNCTION SOR$INIT_MERGE

EXTERNAL INTEGER FUNCTION SOR$DO_MERGE
```

b) Efectuar el llamado a SOR$PASS_FILES

```
<variable entera> = SOR$PASS_FILES ('<archivo de entrada>'
                                    [,<archivo de salida>'])
```

Esta rutina indica al MERGE los archivos que se mezclarán (archivos de entrada) y el archivo de salida que contendrá el resultado de la mezcla.

Se deberá llamar a esta rutina por cada archivo que se desee mezclar; solamente la primera vez se indicará el archivo de salida.

c) Efectuar la llamada a la rutina SOR$INIT_MERGE.

```
<variable entera> = SOR$INIT_MERGE (<orden>,<variable en-
                                     tera 2>)
```

En donde:

orden = Es una variable o una constante entera que indica el número de archivos a mezclar.

variable entera 2 = Es el nombre del campo que contiene el número de llaves y que se encuentra dentro del 'MAP' de llaves.

d) Efectuar la llamada a la rutina SOR$DO_MERGE.

<variable entera> = SOR$DO_MERGE

Esta rutina ejecuta la mezcla y limpia las áreas utilizadas.

RECORD INTERFACE.

a) Declarar las rutinas como funciones enteras externas.

EXTERNAL INTEGER FUNCTION SOR$INIT_MERGE

EXTERNAL INTEGER FUNCTION SOR$RETURN_REC

EXTERNAL INTEGER FUNCTION SOR$END_SORT

b) Abrir todos los archivos de entrada compartiendo el mismo 'MAP'.

c) Efectuar el llamado a la rutina SOR$INIT_MERGE.

<variable entera> = SOR$INIT_MERGE (<orden>,<variable entera2>, <Long>)

En donde:

orden = Es una variable o constante entera que contiene el número de archivos a mezclar.

variable entera 2 = Es el nombre del campo que contiene el número de llaves y que se encuentra dentro del 'MAP' de llaves.

Long = Es una variable o constante entera que define el tamaño máximo del registro.

d) Efectuar la llamada a la rutina SOR$RETURN_REC.

<variable entera> = SOR$RETURN_REC (<variable string>,
<tamaño de registro>)

Esta rutina es la misma que se utiliza en SORT y al igual, se debe llamar cada vez que se accese un registro del MERGE.

e) Efectuar la llamada al SOR$END_SORT.

<variable entera> = SOR$END_SORT

Esta rutina limpiará las áreas de trabajo del MERGE.

```
 TY EX01.BAS
1!
 !
 ! Este programa ilustra algunas de las funciones de E/S
 ! de  BASIC incluyendo el abrir un archivo, uso del MAP
 ! el acceso de un registro del archivo y las rutinas de
 ! deteccion de errores por parte del usuario.
 !  El programa realiza lo siguiente:
 ! 1. Pide el nombre de un archivo secuencial, variable.
 ! 2. Si el archivo no existe vuelve al paso 1.
 ! 3. Imprime el nombre del archivo. Cuando se encuentre
 !    el EOF por primera vez, da un RESTORE al archivo y
 !    lo vuelve a imprimir por segunda vez.
 ! 4. Cuando el EOF se detecta por segunda vez el programa
 !    debe terminar. Como se requiere una accion diferente
 !    cuando el segundo  EOF (fin de datos) se detecte, la
 !    rutina de manejo de errores debera  cambiar en forma
 !    dinamica.
 !
05       ON ERROR GO TO 80
10       MAP (BUFF) DATA$=80
20       LINPUT 'dame el nombre del archivo ';FILE.NAME$
30       OPEN FILE.NAME$ FOR INPUT AS FILE #1,    &
            SEQUENTIAL VARIABLE, MAP BUFF, ALLOW READ,  &
            ACCESS READ, RECORDTYPE ANY
40       GET #1
50       PRINT SEG$(DATA$,1,RECOUNT)
70       GO TO 40
80       IF ERR=11%
            THEN RESUME 90
            ELSE IF (ERR=5%) OR (ERR=2%)
                 THEN CLOSE #1
                      PRINT 'nombre de archivo incorrecto'
                      RESUME 20
            ELSE ON ERROR GO TO 0
90       RESTORE #1
100      ON ERROR GO TO 120
110      GO TO 40
120      IF ERR=11%
            THEN RESUME 130
            ELSE ON ERROR GO TO 0
130      END
 !
 ! NOTAS:
 !
 ! Linea 05. Atrapa todos los errores en la linea 80.
 !
 ! Linea 10. La declaracion MAP permite un tamano maximo de
 !  registro de 80 bytes.  El nombre del 'buffer' es  BUFF.
 !  Este nombre debe aparecer el la clausula  MAP del OPEN.
 !  El 'string'  DATA$  queda definido de longitud fija, ya
 !  que aparece en el mapa MAP.
 !
 !
 !
 !
 !
 !
```

Linea 30. Debido a la inclucion de ACCESS READ y ALLOW READ, otros usuarios pueden compartir el archivo para lectura. Mas detalles de las clausulas ALLOW y ACCESS se daran posteriormente. Debe notarse que la impresion compartida de archivos secuenciales es posible si el archivo tiene registros de longitud fija pero no se permite si tienen longitud variable.
Por la inclucion de RECORDTYPE ANY, los atributos del registro no necesitan ser conocidos.

Linea 40. GET #1 transfiere un registro del archivo a la variable DATA$. Si el registro es menor de 80 bytes DATA$ sera rellenado con caracteres nulos, a la derecha, y si el registro es mayor de 80 bytes la computadora marcara un error.

Linea 50. Imprime solamente el numero de bytes en el registro determinados por la variable del sistema RECOUNT.

Linea 80. Si ERR=11, se ha detectado el EOF por primera vez. RESUME 90 borrara la bandera de error y transferira el control a la linea 90.
Si ERR <> 11, hay que checar por ERR 5 o ERR 2 que indican respectivamente 'ILLEGAL FILE NAME'(nombre ilegal de archivo) o 'FILE OR ACCOUNT NOT FOUND'(archivo o cuenta no encontrada). Cualquiera de estos dos errores indica que el nombre que se dio de archivo es incorrecto. El comando CLOSE #1 se requiere debido a que el canal #1 permanece en un estado de 'OPEN' a pesar de que el archivo no fue encontrado. Si no se da CLOSE #1, un error ocurrira cuando se intente abrir el archivo con el nombre correcto.
RESUME 20 se requiere para finalizar la rutina de error. Si el error no es ninguno de los tres mencionados, la ultima clausula de ELSE en la linea 80 permitira que la computadora maneje el error. Como un error existe, si se ejecuta la instruccion 'ON ERROR GO TO 0' ocasionara un rastreo que indicara que error es y en donde ocurrio, y el programa terminara.

Linea 90. Ocasiona que el apuntador del archivo se posicione en el primer registro y asi el archivo puede ser reimpreso.

Linea 100. Establece un nuevo procedimiento de error. Todos los errores se atraparan en la linea 120.

Linea 120. Prueba por el EOF. Si ya se encontro, entonces da un RESUME a la linea 130 y termina; en caso contrario permite que el sistema maneje el error.

```
1!
 ! Este programa acepta datos de la terminal y los escribe en un
 ! archivo que se llama OUT.DAT. Si el archivo no existe, un nuevo
 ! archivo es creado; si si existe, los datos seran grabados al
 ! final del archivo. El archivo que se crea es secuencial, con
 ! registros de longitud variable. El largo del registro se define
 ! en el MAP.
 !
 !
10       MAP (BUFF) DATA$=40
20       OPEN 'EX02.DAT' AS FILE #1,                          &
            SEQUENTIAL VARIABLE, ACCESS APPEND, MAP BUFF
         PRINT 'dame los datos; termina la linea con la tecla RETURN'
40       IF DATA$<>'' THEN
            PUT #1
            GO TO 10
         ELSE CLOSE #1
         END
 !
 ! Linea 10 fija a la variable DATA$ en 40 bytes. Si a una variable
 ! 'string' que aparezca en un MAP, no se le especifica el tamano,
 ! le sera asignado el tamano por omision ( 'default' ) = 16.
 ! Por ejemplo, los 'strings' en MAP o COMMON son estaticos o de
 ! tamano fijo, todos los demas 'strings' son dinamicos.
 !
 ! Linea 20 abre el archivo con 'ACCESS APPEND'. Cuando el archivo
 ! se crea, este parametro no surte efecto. Sin embrago, en las
 ! corridas subsecuentes del programa, el apuntador del archivo se
 ! colocara en el fin de este (EOF) cuando el archivo sea abierto.
 ! Esto permitira que los nuevos datos sean grabados al final del
 ! archivo. Cabe hacer notar que fracasara cualquier intento de
 ! anadir datos a un archivo secuencial a menos que el apuntador
 ! de los registros este posicionado al final del mismo.
 ! NOTA: La clausula 'FOR OUTPUT' NO debe incluirse en el OPEN.
 ! Si esta presenta, un NUEVO ARCHIVO SERA CREADO cada vez que se
 ! corra el programa, en lugar de anadirla al ya existente.
 !
 ! Linea 30 permite la entrada de hasta 40 caracteres. Cualquier
 ! caracter adicional se perdera debido al tamano del 'string'
 ! definido en el MAP.
 !
 ! Linea 40 hace la prueba de la linea vacia ( por ejemplo, solo
 ! se oprimio la tecla RETURN).
 ! La instruccion PUT siempre grabara 40 caracteres, sin importar
 ! el numero de caracteres que se teclearon. Si se desea crear
 ! registros de, realmente, longitud variable, hay que utilizar
 ! la opcion COUNT en el PUT como se muestra:
 !
 !                PUT #1, COUNT RECOUNT -2
 !                                          (-2 DEBIDO AL RETURN)
 ! NOTA: Como RECOUNT tiene el numero actual de caracteres de
 ! dato, el PUT puede fallar si se teclean mas de 40 caracteres
 ! debido a que el tamano maximo del registro esta fijo en 40.

$
```

```
TY EX03.BAS

1!
 !
   Este programa crea un  archivo  secuencial, de longitud
 ! fija, y pregunta por el numero de  registro. Este numero
 ! se usa  para  accesar el  archivo en  forma  directa. El
 ! contenido  del archivo se imprime  secuencialmente hasta
 ! el EOF.
 ! La mayoria de las aplicaciones requieren  acceso directo
 ! en archivos que son manejados tipicamente  como archivos
 ! relativos (RELATIVE) o indexados(INDEXED) ; este ejemplo
 ! ilustra el hecho de que un archivo SECUENCIAL FIJO puede
 ! accesarse en forma secuencial o directa.
 ! Una vez accesado el registro, el archivo puede accesarse
 ! secuencialmente desde ese punto.
 !
       DECLARE WORD REC. NO
       ON ERROR GO TO 19000
       MAP (BUFF) STRING IN.DATA=40
       OPEN "EX03.DAT" FOR OUTPUT AS FILE #1,              &
          SEQUENTIAL FIXED, MAP BUFF
       PRINT 'Por favor teclee los datos.'
       WHILE I%=0
          LINPUT IN.DATA
          IF IN.DATA='' THEN I%=1
              ELSE PUT #1
       NEXT
50     INPUT "Teclee el numero del registro a leer",REC.NO
       GET #1, RECORD REC.NO
       WHILE -1%
           PRINT IN.DATA
           GET #1
       NEXT
19000  IF (ERR=11%) THEN RESUME 19010
                    ELSE ON ERROR GO TO 0
19010  END
 !
 ! NOTAS:
 !
 ! Linea 10. Ocasiona el acceso directo del registro con el GET.
 !
 ! Linea 10.4.Accesa el archivo en forma secuencial, con el GET.

$
```

```
TY EX04.BAS
1!
 !  Este programa pide un nombre de archivo y crea un nuevo
 ! archivo que, cuando se mande a imprimir en papel, sera
 ! sobreimpreso. Este nuevo archivo es creado con el atributo
 ! RECORDTYPE FORTRAN. Cada registro del archivo de dato es
 ! impreso dos veces en el archivo de salida; el primer
 ! registro tiene un caracter de control en blanco (espacio),
 ! y el segundo registro tiene un caracter de control +
 ! para que el registro sea impreso sin dar el salto de linea.
 !
 ! La sobreimpresion puede ser muy util en la impresion de
 ! reportes que contenga partes que se desean remarcar.
 !
       ON ERROR GO TO 19000
10     MAP (BUFF) DATA$=81
20     MAP (BUFF) STRING CTRL_CHAR=1
       INPUT 'Deme el nombre del archivo ';FILENAME$
       OPEN FILENAME$ FOR INPUT AS FILE #1,                &
          SEQUENTIAL VERIABLE, MAP BUFF
40     START%=POS(FILENAME$,'.',1)
50     OUTFILE$=SEG$(FILENAME$,1,START%-1) + '.LIS'
60     OPEN OUTFILE$ FOR OUTPUT AS FILE #2,                &
          SEQUENTIAL VARIABLE, MAP BUFF, RECORDTYPE FORTRAN
70     GET #1
80     DATA$=' ' + SEG$(DATA$,1,RECOUNT)
       PUT #2, COUNT RECOUNT + 1
100    CTRL_CHAR='+'
       PUT #2, COUNT RECOUNT + 1
       GO TO 70
19000  IF (ERR=11%) THEN RESUME 19010
                    ELSE ON ERROR GO TO 0
19010  END
 !
 ! NOTAS:
 !
 ! Linea 10. El tamano maximo de registro esperado es de 80
 ! bytes. El 'buffer' se ha definido de 81 bytes para poder
 ! almacenar el caracter de control adicional colocado al
 ! inicio del registro antes de grabarse en el archivo de
 ! salida.
 !
 ! Linea 20. El segundo MAP solo se usa para definir a la
 ! variable CTRL_CHAR como el primer byte en el buffer.
 !
 ! Linea 40. Localiza al (.) en el nombre del archivo.
 !
 ! Linea 50. Crea un nuevo archivo llamado XXX.LIS
 ! donde XXX es el nombre del archivo de entrada.
 !
 ! Linea 60. OPEN OUTFILE$ con el atributo RECORDTYPE FORTRAN.
 ! para que cuando el archivo OUTFILE$ sea impreso, el primer
 ! caracter funcione como el caracter de control.
 !
 !
 !
 !
```

! Linea 80. Coloca un caracter blanco al inicio de DATA$.
! Cuando la linea se imprima, este caracter de control
! ocasionara una situacion normal de saltar una linea,
! imprimir y mandar un 'carriage return'.

! Linea 100. La segunda instruccion de MAP coloca a la variable
! CTRL_CHAR como el primer caracter del 'buffer'. Dado que el
! primer caracter ya ha sido puesto como un espacio en blanco,
! podemos utilizar a la variable CTRL_CHAR para colocar un +
! en la columna uno. Esta es una manera conveniente de usar un
! MAP que no se esta utilizando en el OPEN. Para mayores datos
! sobre el uso de multiples MAPas consulte el manual de BASIC.

$

```
TY EX06.BAS
1!
 !  Este programa acepta un nombre y una direccion via terminal y
 ! lo escribe en un  registro que contiene la longitud del nombre
 ! y la longitud de la direccion en las dos primeras palabras del
 ! registro, seguidos de los  'strings' que contienen el nombre y
 ! la direccion. Despues de terminada la entrada, el archivo sera
 ! impreso.
 !
       DECLARE WORD ADD.LEN,NAME.LEN
20     OPEN "EX06.DAT" AS FILE #1, SEQUENTIAL VARIABLE,     &
          RECORDSIZE 80, ACCESS APPEND
       PRINT 'De los datos y termine oprimiendo RETURN '
40     LINPUT 'Nombre ';NAME$
       IF NAME$='' THEN 120
60     NAME.LEN=LEN(NAME$)
70     LINPUT 'Direccion 'ADD$
80     ADD.LEN=LEN(ADD$)
       MOVE TO #1, NAME.LEN,ADD.LEN,NAME$,ADD$
100    PUT #1, COUNT NAME.LEN+ADD.LEN+4%
       GO TO 40
120    RESTORE #1
130    GET #1
140    MOVE FROM #1, NAME.LEN,ADD.LEN,NAME$=NAME.LEN,ADD$=ADD.LEN
       PRINT NAME$;'   ';ADD$
       GO TO 130
       END
 !
 ! NOTAS:
 !
 ! Linea 20. No use 'FOR OUTPUT'  ya que esto ocasionara la creacion
 ! de nuevas versiones cada vez  que se corra  el programa, en lugar
 ! de ir anadiendo los registros al final del archivo.
 !
 ! Lineas 60 y 80. Salvan las longitudes de los 'strings' del nombre
 ! y la direccion para usarlos en la linea 100.
 !
 ! Linea 100. adiciona un 4 al campo COUNT  debido a que el registro
 ! incluye las variables  NAME.LEN y  ADD.LEN que son de una palabra
 ! (word= 2 bytes).
 !
 ! Linea 140. Mueve  los campos de  dos bytes a  NAME.LEN y  ADD.LEN
 ! para que puedan ser utilizados  como las longitudes de los campos
 ! NAME$ y ADD$

$
```

```
TY EX07.BAS
1!
 !  Este programa ilustra el uso de MAP con MOVE TO y MOVE FROM
 ! en el mismo programa.
 !  Es el mismo ejemplo que  EX05.BAS excepto que solo se graba
 ! un registro por dueno. El  registro contiene  el  nombre, la
 ! direccion, el numero de vehiculos y los datos de cada uno de
 ! los vehiculos. El  'buffer' esta  disenado para  permitir un
 ! maximo de 10 vehiculos.
 !
10       MAP (AUTO) STRING NAM=20,ADD=40,WORD CAR.COUNT
         MAP (AUTO) FILL$=210,STRING MODEL=6,PLATE=6
         ON ERROR GO TO 19000
         DECLARE WORD YEAR,VALUE
20       OPEN 'AUTO.DAT' FOR OUTPUT AS FILE #1,
            SEQUENTIAL VARIABLE, MAP AUTO
30       INPUT 'De nombre y direccion ' ;NAM ;ADD
         IF RECOUNT<=2 THEN GO TO 80
         ELSE CAR.COUNT=0%
40       INPUT 'Modelo, a&o,valor y placas',MODEL,     &
            YEAR,VALUE,PLATE
50       MOVE TO #1,FILL$=62%+CAR.COUNT*16%,MODEL, &
         YEAR,VALUE,PLATE
60       CAR.COUNT=CAR.COUNT+1%
         INPUT 'Mas vehiculos del mismo dueno?'; MORE$
         IF MORE$='SI' THEN GO TO 40
70       PUT #1,COUNT CAR.COUNT*16%+62%
         GO TO 30
80       RESTORE #1
         WHILE -1%
             GET #1%
             PRINT NAM;ADD
             FOR I%=0% TO CAR.COUNT-1%
                 MOVE FROM #1%,FILL$=62%+I%*16,MODEL=6,  &
                    YEAR,VALUE,PLATE=6
             PRINT MODEL,YEAR,VALUE,PLATE
             NEXT I%
         NEXT
19000    IF ERR=11 THEN RESUME 19010
                        ELSE ON ERROR GO TO 0
19010    END
 !
 ! NOTAS:
 !
 ! Linea 10. El primer MAP se usa para colocar NAM, ADD y CAR.COUNT
 ! en los primeros 62 bytes del registro de salida.
 ! El segundo MAP define el tamano del 'buffer' de 222 bytes, donde
 ! 60 son para  NAM y ADD, 2 para CAR.COUNT y 160  para la informa-
 ! cion de los 10 autos(16 bytes por cada auto).
 ! MODEL=6 y PLATE=6 se incluyen simplemente para fijar los tamanos
 ' a  6 bytes cada uno,  pero esto no  implica que sean estatiticos
   en el programa.(el colocarlos en el MAP causa que sean de tamano
 ! estatico). No deben estar al inicio del MAPa porque cada vez que
 ! un nuevo valor de MODEL y PLATE sea leido, el  'string' NAM sera
 ! destruido. (NAM son los primeros 20 bytes del 'buffer')
 !
 !
```

! Linea 20. Por incluir MAP AUTO en el OPEN,los campos de NAM,ADD
! y CAR.COUNT ocupan los primeros 62 bytes del registro de salida
! y no necesitan ser movidos (MOVE) al 'buffer'. Pero, aunque los
! valores para NAM y ADD son leidos, el valor de CAR.COUNT estara
! en su lugar en el 'buffer' MAP AUTO
!
! Linea 50. El MOVE siempre inicia al principio del 'buffer';
! FILL$=62 se requiere para saltarsobre los campos NAM, ADD y
! CAR.COUNT. Para el primer auto, FILL$=CAR.COUNT*16 sera 0, por
! lo que MODEL, YEAR, VALUE y PLATE seguiran a NAM, ADD y
! CAR.COUNT en el buffer. Para el siguiente vehiculo, el valor de
! CAR.COUNT*16 sera de 16, etc.
!
! Linea 60. Note que si CAR.COUNT excede 10, el mensaje de error
! del sistema 'MOVE OVERFLOWS BUFFER' sera impreso.
!
! Linea 70. Cuando todos los datos de un propietario hayan sido
! tecleados, el registro sera grabado (PUT) en el archivo
! El tamano del registro depende del numero de vehiculos de cada
! propietario
!
! Linea 80.6. FILL$ debe usarse de la misma manera que en MOVE TO
! para saltar sobre los campos NAM, ADD y CAR.COUNT, asi como sobre
! los campos de MODEL, YEAR, VALUE y PLATE de cada auto. Note que
! una longitud DEBE incluirse para MODEL y PLATE en el MOVE FROM
! (El tamano de 'default' de un string en un MOVE FROM es 16 bytes.)
!

$

```
TY EX08.BAS
 1!
 !
 !   Este programa crea un archivo virtual llamado  A.DAT
 !  que contiene registros de 16 bytes empacados  32  por
 !  512 bytes(un bloque). Con el fin de reducir el numero
 !  de accesos al disco, un  RECORDSIZE de  2048 se usara
 !  para escribir 4 bloques en el disco con un solo  PUT.
 !
05       DECLARE LONG RECNO, WORD I,J
10       OPEN 'A.DAT' FOR OUTPUT AS FILE #1, virtual,    &
             RECORDSIZE 2048
15       REC.NO$='RECORD NO.   '
20       RECNO=0
30       FOR J=1 TO 3
40           FOR I=1 TO 128
50               RECNO=RCNO+1
60               MOVE TO #1, FILL$=(I-1)*16,REC.NO$,RECNO
                 NEXT I
90           PUT #1
100      NEXT J
120      END
 !
 !  Linea 10.  RECORDSIZE  2048 significa que cada PUT de la
 !  linea 90 grabara un  registro de longitud  2048 bytes en
 !  el disco.
 !
 !  Linea 40. Como  128  registros logicos se han empacado en
 !  el 'buffer',  se requiere utilizar  FILL$= (I-1)*16  para
    colocar cada registro de 16 bytes en su posicion correcta
    dentro del 'buffer'.
 !
 !  Linea 90. El PUT se ejecuta 3 veces. Cada PUT hara que un
 !  registro  de  2048  bytes ( 4 bloques ) sea  escrito para
 !  hacer un total de 12 bloques. Cada registro de 2048 bytes
 !  en disco contiene 128 registros logicos de  16 bytes cada
 !  uno, lo que hace un total de 3*128=384 registros logicos.

$
```

```
TY EX09.BAS
1!
 !
 !  Este programa solicita un  numero de  registro  para
 ! buscarlo en el archivo virtual creado por el programa
 ! EX08.BAS.
 !
10       DECLARE WORD REC.NO, NEW.BLOCK, OLD.BLOCK
20       OPEN 'A.DAT' FOR INPUT AS FILE #1, VIRTUAL
30       INPUT 'Deme el numero de registro que desea ',REC.NO
35       IF REC.NO=0 THEN 120
40       NEW.BLOCK=(REC.NO/129)+1
45       IF NEW.BLOCK=OLD.BLOCK THEN 60
50       GET #1, RECORD NEW.BLOCK
60       REC.NO=REC.NO-(128%*(NEW.BLOCK-1))
80       MOVE FROM #1, FILL$=(REC.NO-1)*16,DATA$=12,RECORD%
90       PRINT 'Registro del archivo ',DATA$,RECORD%
95       OLD.BLOCK=NEW.BLOCK
100      GO TO 30
120      END
 !
 !  Linea 40. Calcula   NEW.BLOCK   para determinar en cual de
 !  los tres registros(2048 bytes) contiene el registro logico
 !  que fue solicitado.  Como el resultado debe ser entero, el
 !  valor sera truncado. Este resultado se usara para realizar
 !  el acceso directo del registro fisico mediante un  GET.
 !
 !  Linea 45. La primera vez que el programa pasa por esta linea
 !  el valor de  OLD.BLOCK no estara definido. En las siguientes
 !  ocasiones que pase por esta linea, solo se accesara un nuevo
 !  registro fisico si el registro logico pedido NO se encuentra
 !  dentro del ultimo registro fisico accesado (2048 bytes).
 !
 !  Linea 50. El acceso directo mediante el GET leera el primer,
 !  segundo o tercer registro fisico, de 2048 bytes, del disco.
 !
 !  Linea 60. Despues de hecho el GET, el 'buffer' contendra 128
 !  registros logicos. Se requiere recalcular el valor de REC.NO
 !  para que el FILL$ que se usa en el  MOVE  ocasione el salto
 !  correcto para alcanzar el registro logico solicitado.
 !
 !  Linea 80. este MOVE hara que el registro logico solicitado
 !  sea enviado a las varibles DATA$ y RECORD%.  El  FILL$  se
 !  necesita para saltar sobre los  registros no  deseados que
 !  estan el el 'buffer'.
 !
 !  Linea 95. Asigna  OLD.BLOCK=NEW.BLOCK para que no se efectue
 !  ningun acceso al disco,  si el registro logico solicitado se
 !  encuentra en el mismo registro fisico.
$
```

```
10 !                        FILESORT.BAS
   !
   !     ESTE PROGRAMA MUESTRA EL USO Y GRAN UTILIDAD DEL
   !     SORT PARA SORTEAR TODOS LOS REGISTROS DE UN ARCHIVO.
   !     es probable que EL ARCHIVO DE ENTRADA (INFILE.DAT)
   !     TENGA CUALQUIER ORGANIZACION (EXCEPTUANDO LA ORGANIZACION
   !     INDEXADA), Y CUALQUIER FORMATO de registro.
   !     EL ARCHIVO DE SALIDA (OUTFILE.DAT) tendra organizacion
   !     CON RESPECTO A LA SECUENCIA.
   !     EL FORMATO DEL REGISTRO Y LOS ATRIBUTOS DEL
   !     ARCHIVO SERAN LOS MISMOS AL DEL ARCHIVO DE
   !     ENTRADA.
   !
   EXTERNAL INTEGER FUNCTION SOR$INIT_SORT,SOR$PASS_FILES !SORT routines

30 EXTERNAL INTEGER FUNCTION SOR$SORT_MERGE,SOR$END_SORT
   !
   !     ESTABLECE EL BUFFER LLAVE PARA DESCRIBIR CUALES LLAVES SON
   !     USADAS PARA SORTEAR EL ARCHIVO. EN ESTE CASO, UNICAMENTE
   !     SE USA UNA LLAVE.
   !
50 MAP (KEY_MAP) WORD NO_KEYS,KEY_TYPE,KEY_ORDER,START_POS,  &
                 KEY_LENGTH
53 DECLARE WORD REC_LENGTH,SORT_TYPE,WORK_FILES, LONG FILE_SIZE
60 NO_KEYS=1%                      ! NUMERO DE LLAVES A SORTEAR.
70 KEY_TYPE=1%                     ! TIPO DE LLAVE = 1 = CARACTER
80 KEY_ORDER=0%                    ! ORDEN ASCENDENTE.
90 START_POS=1%                    ! COMIENZA POSICION DE LLAVE
95 KEY_LENGTH=20%
96 !
   !     DEFINE OTROS PARAMETROS LLAMADOS POR SOR$INIT_SORT.
   !
   REC_LENGTH = 38%      ! LONGITUD MAXIMA DE REGISTRO
97 WORK_FILES = 0%       ! NO ARCHIVO DE TRABAJO,
                         ! PARA SORTEAR EN MEMORIA.
98 SORT_TYPE  = 2%       ! SORT ETIQUETADO
99 FILE_SIZE  = 10%      ! FILE_SIZE DEBE SER LONG WORD.
100 !
    !  NOMBRES DE ENTRADA Y SALIDA DE ARCHIVO
    !  PASAN A Sort.
    !
    I% = SOR$PASS_FILES('INFILE.DAT','OUTFILE.DAT')
110 IF (I% AND 1%) = 0% THEN PRINT "ERROR IN SOR$PASS_FILES =";I%
120 !
    !  PASA LA INFORMACION LLAVE Y SE INICIALIZA EL SORT EN
    !  SECUENCIA.
    !
    I% = SOR$INIT_SORT(NO_KEYS,REC_LENGTH,FILE_SIZE,WORK_FILES,&
                       SORT_TYPE)
130 IF (I% AND 1%) = 0% THEN PRINT "ERROR IN SOR$INIT_SORT =";I%
140 !
    !  PROCESO DE SORTEO DEL ARCHIVO
    !
    I% = SOR$SORT_MERGE
150 IF (I% AND 1%) = 0% THEN PRINT "ERROR IN SOR$SORT_MERGE =";I%

160 !

    !     LIMPIEZA EN AREAS DE TRABAJO DEL SORT
    !
    I% = SOR$END_SORT
170 IF (I% AND 1%) = 0% THEN PRINT "ERROR IN SOR$END_SORT =";I%
180 PRINT "SORT COMPLETED SUCCESSFULLY."
190 END
```

TY SORT1.BAS

```
10 !                          RECSORT.BAS
   !
   ! ESTE PROGRAMA SORTEA SELECTIVAMENTE LOS REGISTROS EN UN
   ! ARCHIVO.
   ! CADA REGISTRO ES LEIDO, Y SI EL CAMPO STATUS DEL REGISTRO
   ! NO CONTIENE LOS CARACTERES FULL, EL REGISTRO ES DESCARTADO.
   ! UNICAMENTE PARTES DEL REGISTRO ORIGINAL SON PASADAS A LAS
   ! RUTINAS DEL SORT A SER SORTEADAS.
   ! DESPUES QUE TODOS LOS REGISTROS HAN SIDO ACCESADOS, Y
   ! SORTEADOS, CADA UNO DE ESTOS YA SORTEADO ES RECUPERADO E IMPRESO.
   !
   EXTERNAL INTEGER FUNCTION SOR$INIT_SORT,SOR$SORT_MERGE,STR$UPCASE
30 EXTERNAL INTEGER FUNCTION SOR$END_SORT,SOR$RELEASE_REC,SOR$RETURN_REC
55 EXTERNAL INTEGER CONSTANT SS$_ENDOFFILE ! DECLARA ESTADO DEL CODIGO
60 !
   !    ESTABLECE LLAVE BUFFER POR REGISTRO SORTEADO. EN ESTE CASO
   !    DOS LLAVES SON USADAS.
   !
   MAP (KEY_MAP) WORD KEY_INFO,KEY_TYPE,KEY_ORDER,START_POS,&
                 KEY_LENGTH,KEY_TYPE_TWO,KEY_ORDER_TWO,     &
                 START_POS_TWO,KEY_LENGTH_TWO
62 KEY_INFO = 2%                        ! NUMERO DE LLAVES
64 KEY_TYPE = 1%                        ! TIPO DE LLAVE = CARACTER = 1
66 KEY_ORDER = 0%                       ! ORDEN ASCENDENTE
68 START_POS = 1%                       ! COMIENZO DE POSICION = 1
70 KEY_LENGTH = 4%                      ! LONGITUD DE LLAVE = 4
72 KEY_TYPE_TWO  = 1%                   ! TIPO DE LLAVE = CARACTER = 1
74 KEY_ORDER_TWO = 1%                   ! ORDEN DESCENDENTE
76 START_POS_TWO = 5%                   ! COMIENZO DE POSICION = 5
78 KEY_LENGTH_TWO = 8%                  ! LONGITUD DE LLAVE = 8
80 !
   ! ESTABLECE MAP POR REGISTROS LEIDOS DESDE EL ARCHIVO INFILE.DAT,
   ! OPEN INFILE.DAT, Y ESTABLECE MAXIMA LONGITUD DE UN REGISTRO.
   !
   MAP (MYMAP) NAME$=20,DEPT$=4,STATUS$=4,SALARY$=8,LONGEVITY$=2
81 UPSTATUS$ = SPACE$(4%)               ! STATUS$ SERA RETENIDO EN LETRAS MAYUSCULAS
85 OPEN 'INFILE.DAT' AS FILE #1, SEQUENTIAL FIXED, MAP MYMAP
87 REC_SIZE% = 44%                      ! MAXIMA LONGITUD DE REGISTRO PASADA
                                        ! A SORT
90 !
   !  INFORMACION LLAVE PASADA Y SE INICIALIZA LA SECUENCIA DEL SORT
   !
   I% = SOR$INIT_SORT(KEY_INFO,REC_SIZE%)
95 IF (I% AND 1%) = 0% THEN PRINT "ERROR IN SOR$INIT_SORT =";I%
98 ON ERROR GOTO 400                    ! PREPARA PARA ERROR EN FIN DE
                                        ! ARCHIVO
100 !
    !  LEE UN REGISTRO DESDE EL ARCHIVO Y CHECA PARA ASEGURARSE
    !  QUE EL CAMPO STATUS CONTENGA EL VALOR FULL. DE ESTA MANERA
    !  SE IDEA O CONSTRUYE EL REGISTRO PARA PASAR AL SORT, PASA
    !  EL REGISTRO E ITERA (LOOP) HASTA QUE EL FIN DE ARCHIVO
    !  ES DESCUBIERTO.
    !
    !
    !
    !
    !
```

```
    !
    !
    !
    GET # 1                          ! LEE UN REGISTRO
    I% = STR$UPCASE(UPSTATUS$,STATUS$) ! CONVERTIR A LETRAS MAYUSCULAS
                                     ! PARA PRUEBA
    CALL SYS$EXIT(I% BY VALUE) IF (I% AND 1%) = 0%
110 IF UPSTATUS$ <> 'FULL' GOTO 100   ! LOOP SI NO DESEA REGISTRO
120 C$ = DEPT$ + SALARY$ + NAME$ + DEPT$ + SALARY$ ! CONSTRUYE SORT REC
130 !
    ! NOTESE QUE LOS CAMPOS DE DEPT Y SALARY TUVIERON QUE SER USADOS
    ! DOS VECES, YA QUE SUS CONTENIDOS SON REQUERIDOS CUANDO LOS
    ! REGISTROS SON RECUPERADOS, Y NO PUEDEN SER OBTENIDOS DESDE EL
    ! CAMPO LLAVE YA QUE ESTE NO ES REGRESADO.
    !
    I% = SOR$RELEASE_REC(C$)    ! ENVIA REGISTRO MODIFICADO A SORTEAR
150 IF (I% AND 1%) = 0% THEN PRINT 'ERROR IN SOR$RELEASE_REC =';I%
160 GOTO 100                ! LOOP HASTA DESCUBRIR EL FIN DE ARCHIVO
400 !
    !  CHECA PARA ASEGURARSE QUE EL ERROR OCURRIDO FUE UN ERROR
    !  DE FIN DE ARCHIVO (11) EN LINEA 100 (CUANDO LA OPERACION
    !  GET ES PROCESADA).
    !
    IF (ERR=11%) AND (ERL=100%) THEN RESUME 500 &
                ELSE PRINT 'AN ERROR HAS OCCURRED'
500 I% = SOR$SORT_MERGE  ! SORTEA LOS REGISTROS ACCESADOS
510 IF (I% AND 1%) = 0% THEN PRINT 'ERROR IN SOR$SORT_MERGE =';I%
515 D$ = SPACE$(44%)  ! RESERVA ESPACIO PARA EL REGISTRO REGRESADO
520 I% = SOR$RETURN_REC(D$,L%)     ! DESEA EL PROXIMO REGISTRO SORTEADO
530 IF I% = SS$_ENDOFFILE GOTO 600 ! OBSERVA SI YA LEYO EL ULTIMO REC
  0 IF (I% AND 1%) = 0% THEN PRINT 'ERROR IN SOR$RETURN_REC =';I%
 50 PRINT D$              ! DESPLIEGA REGISTRO AL USUARIO
560 GOTO 520              ! LOOP HASTA QUE TODOS LOS REGISTROS SON LEIDOS
600 I% = SOR$END_SORT ! LIMPIA AREAS DE TRABAJO DEL SORT
610 IF (I% AND 1%) = 0% THEN PRINT 'ERROR IN SOR$END_SORT =';I%
620 PRINT 'ALL DONE.'
700 END
```

$

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

LENGUAJE DE PROGRAMACION BASIC II

ESTRUCTURAS DE DATOS

M. EN C. RICARDO CIRIA MERCE
ING. ALEJANDRO JIMENEZ GARCIA

NOVIEMBRE DE 1985

L I S T A S L I N E A L E S

Frecuentemente en la programación de sistemas se presentan problemas en los cuales es necesario trabajar con tablas de información. La forma más simple de estas estructuras es la llamada LISTA - LINEAL, en la cual las relaciones inter-elementales esenciales son lineales.

Así, una lista lineal se define como un conjunto de nodos o elementos donde n 0, tal que

$$x(o), x(1), \ldots, x(n)$$

son esos elementos de la lista.

Se puede observar que siempre será posible determinar cual es el primer elemento de la lista (cabeza de lista), y cual sigue o cual antecede a un elemento dado.

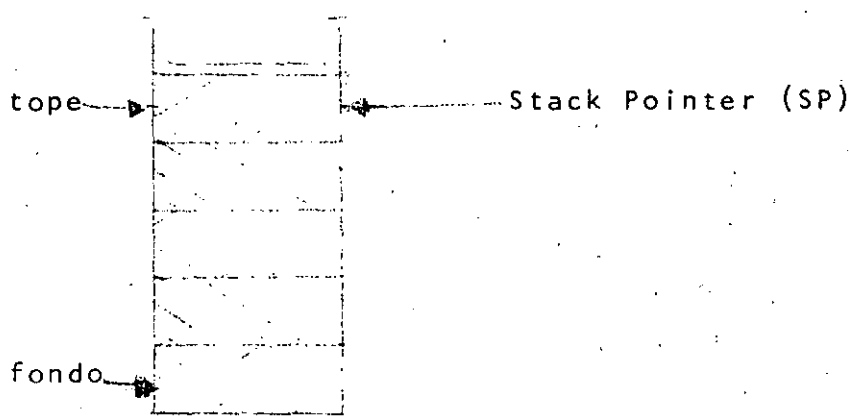
En una lista se pueden efectuar tres operaciones fundamentales:

- Acceso de un elemento,
- Inserción de un nuevo elemento a la lista, y
- Supresión de un elemento de la lista.

Dependiendo de las reglas con las que se realicen estas -- operaciones tendremos diferentes estructuras:

a) PILAS (en inglés STACK's).

Una pila es una lista lineal en la que el primer elemento que entra es el último que sale, o bien, el último en entrar es el primero en salir (Last Input First Output (LIFO)).

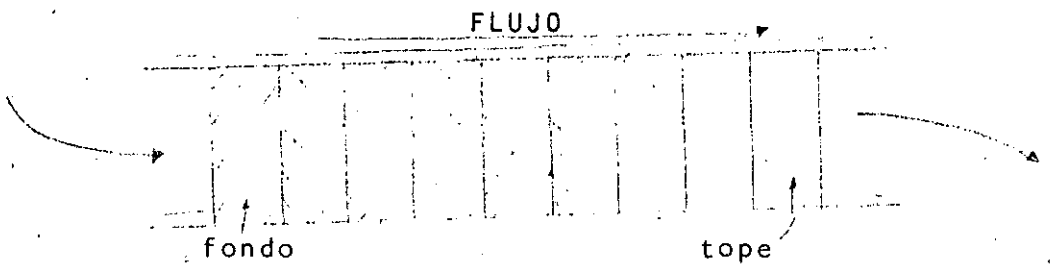


PILA

b) COLAS.

Las colas son listas lineales en las cuales el primer - - elemento en entrar es el primero en salir (First Input First Output (FIFO)).

Para accesar información es necesario ir al tope de la cola y para incluir información es necesario hacerlo en el fondo o principio de la misma.



c) COLAS DOBLES.

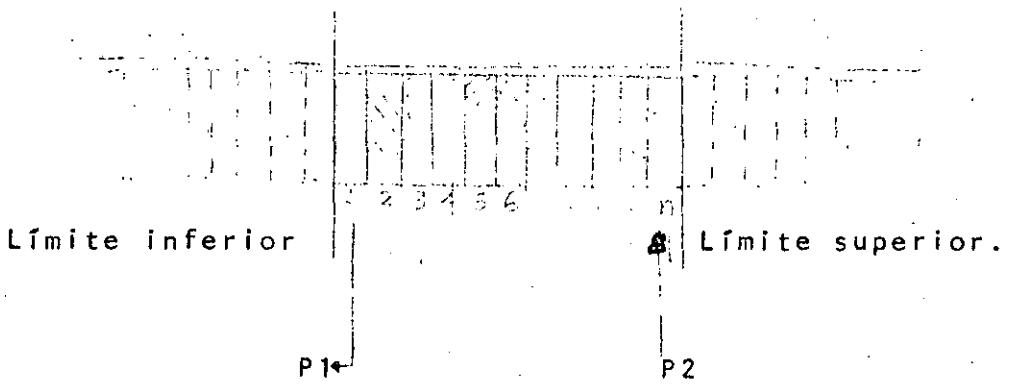
Son una combinación de las pilas y las colas.
La información puede entrar y salir por ambos lados.

IMPLEMENTACION:

a) IMPLEMENTACION SECUENCIAL:

En este tipo de implementación, la relación entre elementos de una estructura es secuencial físicamente, es decir, se sabe que un determinado elemento es el siguiente de una cierta lista con el simple hecho de que su representación interna se encuentre físicamente junto a la de su elemento anterior. Al hacer un acceso físico se hará un acceso lógico también. Es decir, en este tipo de im- - plementación es esencial la localización física de los elementos - - dentro de la memoria.

Por ejemplo, si tratáramos de representar las estructuras mencionadas anteriormente por medio de implementación secuencial, sería algo análogo a lo siguiente:



Representación secuencial de una lista lineal.

Nótese que los límites de las estructuras son fijos, es - decir, se sabe exactamente donde empieza una lista y dónde termina, y no existe manera de que una de ellas sobrepase sus límites, invadiendo otra. Nótese asimismo que los elementos de todas las estructuras van unidos uno con el otro físicamente, y esto es precisamente lo que nos da la pauta para saber cual elemento es parte de cual lista y para determinar cual es su posición lógica dentro de la misma lista.

Si se deseara insertar un elemento nuevo en una lista representada secuencialmente, se deberán seguir los siguientes pasos:

-Investigar de alguna manera si existe algún elemento disponible en la lista.

-En el caso de que la anterior pregunta resultara afirmativa, deberemos a continuación mencionar el lugar que deseamos que o - cupe el nuevo elemento dentro de la lista.

-Luego, recorrer una unidad de memoria la información de - todas las localidades cuya posición sea mayor o igual a la del elemento a insertar.

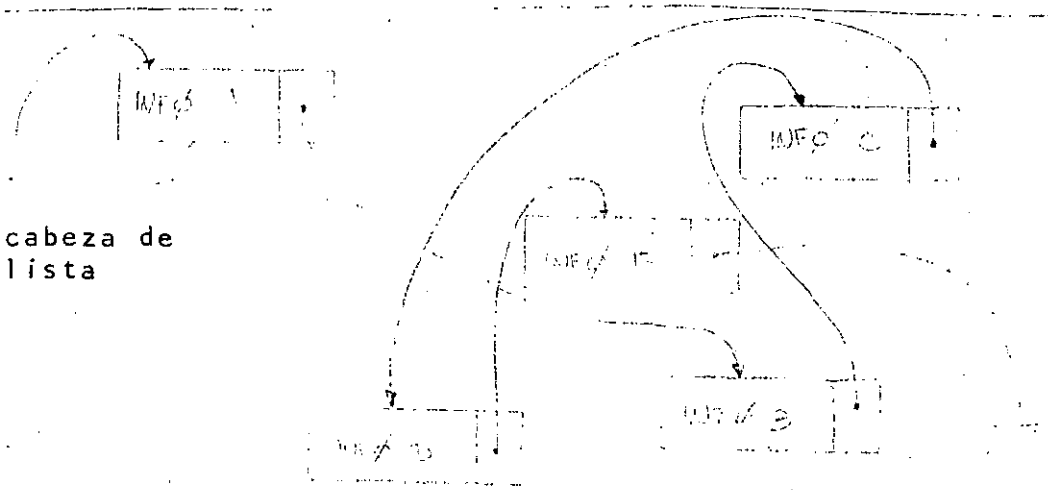
-Finalmente, vaciar la información correspondiente en el - lugar deseado.

b) IMPLEMENTACION LIGADA:

En este tipo de implementación, la relación entre los elementos no es por su continuidad física, sino que se determina la localización de un elemento de la lista con base en una liga, la cual es parte del mismo elemento, y no es más que la localidad de memoria en la cual se encuentra almacenado el siguiente elemento de la lista. Su relación posicional no importa.

Este tipo de implementación es muy versátil, ya que pueden crecer las estructuras muy fácilmente, cosa que en la representación secuencial no es posible.

La representación en memoria de una lista ligada sería como sigue:



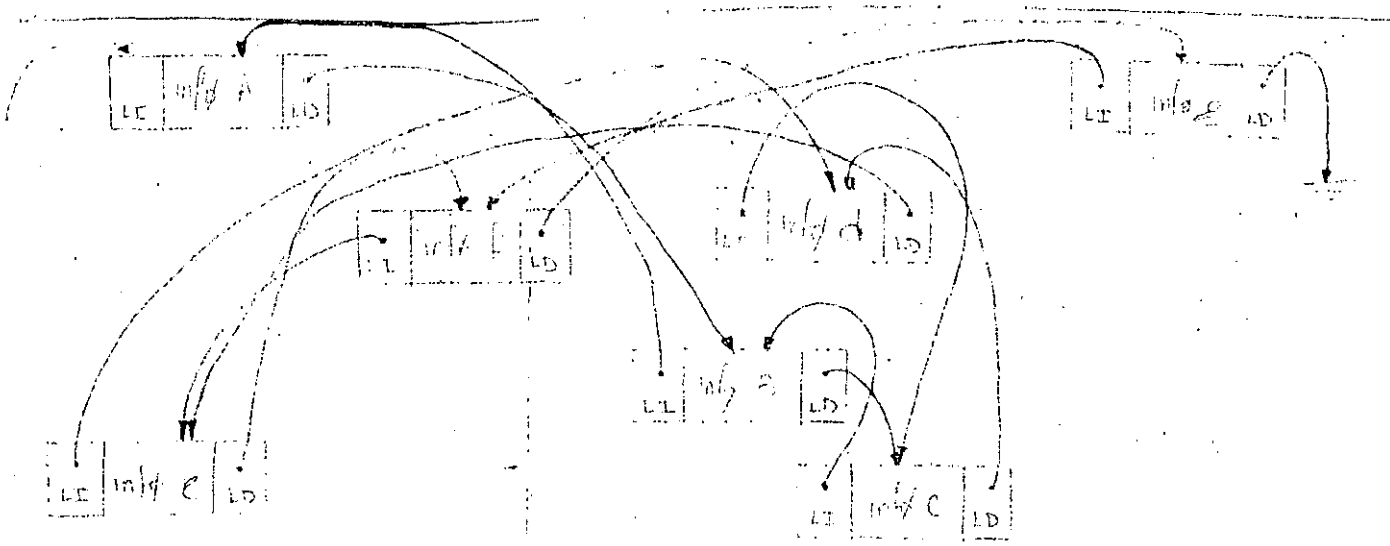
Si se quisiera insertar un nuevo elemento en una lista ligada, se tendría que hacer de la siguiente forma:

-Determinar de alguna manera (por ejemplo de una lista de elementos disponibles) la existencia de elementos disponibles en memoria.
-En caso afirmativo, guardar en la liga del elemento disponible la liga del elemento inmediatamente anterior a donde se desea hacer la inserción.
-A continuación, cambiar la liga del elemento anterior, guardando ahora el número del elemento disponible que usaremos para hacer la inserción.
-Finalmente, vaciar la información deseada en el elemento disponible.

c) IMPLEMENTACION DOBLEMENTE LIGADA:

En este tipo de implementación, la encadenación de elementos se realiza mediante dos ligas: una izquierda en la cual se guarda la localidad de memoria del elemento anterior de la lista, y un a derecha, en donde se almacena el número de localidad de memoria donde se encuentra guardado el siguiente elemento de la lista.

La representación en memoria de este tipo de implementación se presenta a continuación:



Si se quisiera insertar un elemento nuevo en una lista doblemente ligada se tendría que seguir el siguiente proceso:

-Investigar la existencia de una localidad de memoria -- disponible.

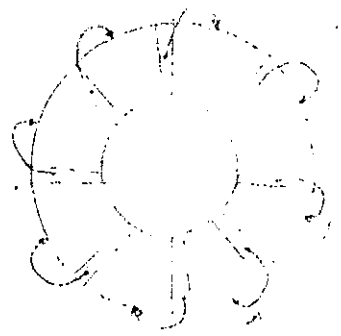
-Si existe esta localidad disponible, almacenar en su -- liga izquierda el número de la localidad de memoria del elemento- anterior a donde se desea hacer la inserción, y la liga derecha a donde apunta la liga derecha de este elemento.

-A continuación, apuntar la liga derecha del elemento anterior al elemento insertado, y la liga izquierda del siguiente elemento al mismo.

-Finalmente, vaciar la información deseada en el elemento ya insertado.

LISTAS CIRCULARES:

Las listas circulares son de utilidad en el aprovechamiento de espacio. Para listas ligadas es fácil hacer la conversión a un a lista circular simplemente haciendo que la liga del último elemento, en lugar de apuntar al vacío, apunte al primer elemento de la lista. En esta lista ya no tiene caso de hablar del primer o último elemento.



En el caso de que quisiéramos insertar un nuevo ele--- mento a la lista circular sería muy sencillo:

-Una vez conseguido de alguna manera un elemento dis- - ponible, se apunta su liga al elemento siguiente a donde se de-- see hacer la inserción. Luego, se apunta la liga del elemento anterior al elemento insertado, y finalmente se vaciaría la información deseada en el elemento correspondiente.

G R A F I C A S .

Un tipo muy usado de estructuras no-lineales es el formado por las GRAFICAS.

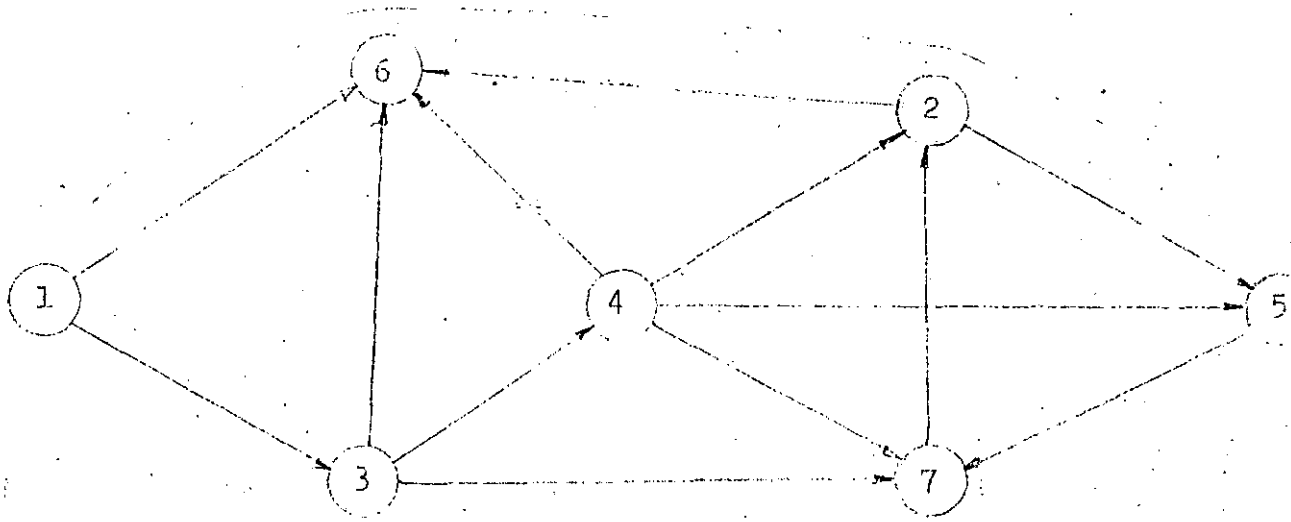
Una gráfica es un par ordenado G = V, R , donde V es un conjunto de vértices y R una colección de pares de elementos de V llamados ramas.

Si los elementos de una rama son un par ordenado, se dice que se tiene un a DIGRAFICA o GRAFICA DIRIGIDA.

Al número de ramas que llegan a un vértice se le llama GRADO o VALENCIA del vértice. En una digráfica se pueden distinguir las ramas que llegan, originando el INGRADO o VALENCIA POSITIVA, y los que salen, generando el EXGRADO o VALENCIA NEGATIVA.

Se acostumbra dibujar una gráfica representando los vértices por puntos y las ramas por líneas o flechas, si es digráfica.

Por ejemplo, sea:
G = 1,2,3,4,5,6,7 , (1,6),(1,3),(2,5),(2,6),(3,6),(3,7),(3,4), (4,2),(4,5),(4,6),(4,7),(5,7),(7,2) , su dibujo queda en la figura siguiente:



Un concepto importante en gráficas es el de camino: se dice que en un a digráfica hay un camino W(x,y) si se pueden encontrar las ramas (x,a_1), (a_1,a_2), ... (a_n,y). Es decir, es el conjunto de ramas para ir del nodo x al nodo y. En general si la gráfica es dirigida, $W(x,y) \neq W(y,x)$.

8

Una gráfica es conexa si todos sus nodos tienen al menos un a rama que los une con el resto de la gráfica; es decir, su grado o valencia es mayor o igual que la unidad.

Una gráfica es finita si se conoce el tamaño de R y el de V.

Las estructuras no-lineables más importantes en la computación son los ARBOLES.

DEFINICION RECURSIVA:

Si se tiene un conjunto T de un o o más nodos, se le llamará árbol si:

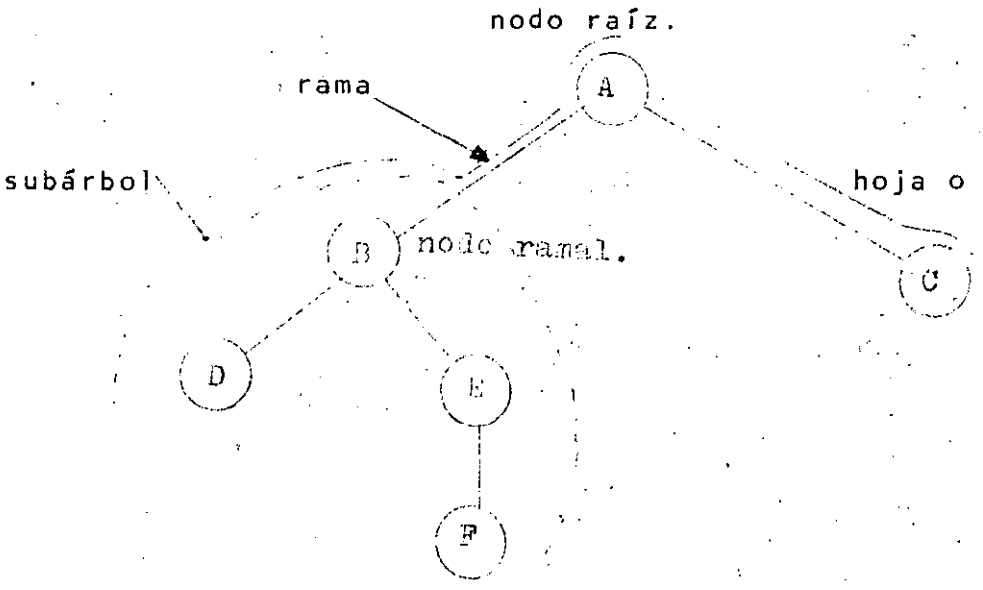
-Hay un nodo especial llamado "raíz" del árbol; y
-Los demás nodos están en una partición de conjuntos-ajenos To, ..., Tm (con m 0), tales que cada uno de ellos es a su vez un árbol. Los árboles To, ..., Tm se llaman subárboles de la raíz.

De esta definición se pueden desprender las siguientes propiedades:

-Cada nodo de un árbol es la raíz de un subárbol.
-Se define como GRADO de un nodo el número de sus subárboles.

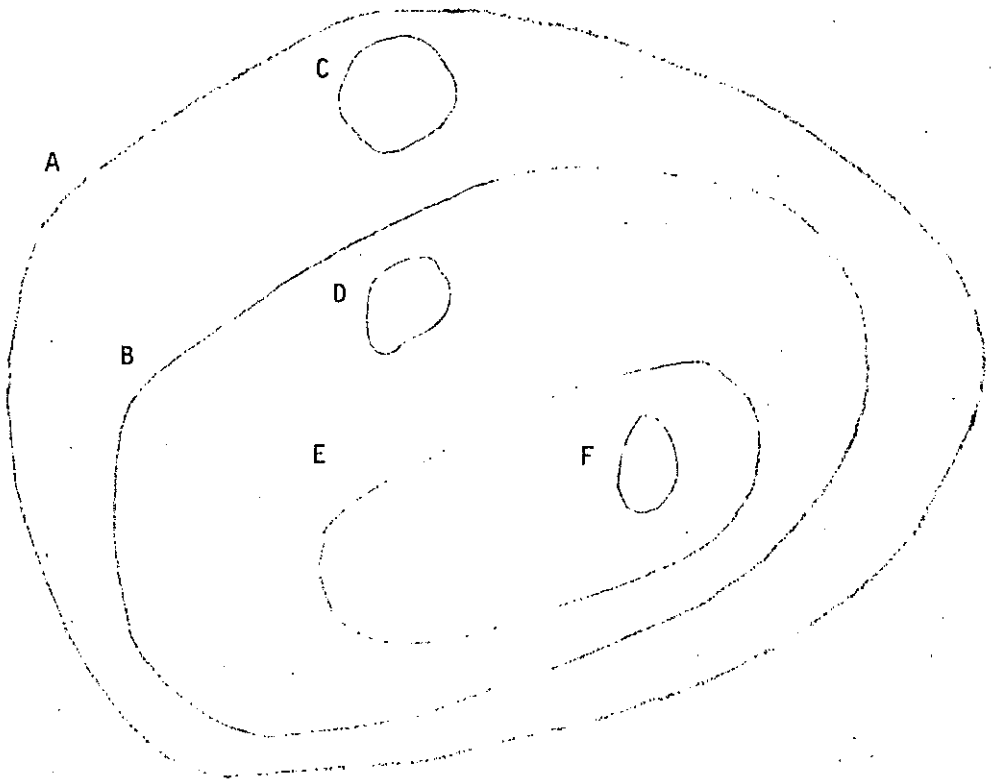
-Una hoja o nodo terminal es aquel cuyo grado es cero.
-Un nodo que no es terminal se llama nodo ramal.
-El nivel de un nodo se define diciendo que la raíz tiene nivel cero y que el nivel de cualquier nodo es igual al nivel de su antecesor más uno.

Por convención se dibujarán los árboles con la raíz en la parte superior, creciendo hacia abajo:



Existen otras formas de representar los árboles, algunas de las cuales se muestran a continuación :

Como conjuntos anidados:



Con paréntesis anidados:

(A (C) (B (D) (E (F))))

Por niveles:

NIVEL:	0	1	2	3
	A			
		B		
			E	
				F
		D		
	C			

Es necesario adoptar una terminología para referirse a las relaciones que guardan los nodos entre sí:

Se dice que la raíz es el PADRE de los subárboles, y que estos son los HIJOS de aquel y entre sí son HERMANOS. También se habla en la literatura de SUCESORES Y PREDECESORES.

Propiedades que surgen de la terminología usada:

-Existe un nodo único (la raíz) sin predecesores.

-Todo nodo, excepto la raíz tiene exactamente un predecesor inmediato.

-Existe un camino único entre la raíz y cualquier nodo.

-El nivel de un nodo es siempre uno más que el nivel de su padre y es igual al nivel de sus hermanos.

DEFINICION NO-RECURSIVA:

Un árbol es una gráfica G= V, R conexa, finita tal que el tamaño de V es igual al tamaño de R más uno, es decir, - el número de vértices es igual al número de ramas más uno.

RAMA DESCONECTANTE es aquella que al ser removida de la gráfica hace no-conexa a la gráfica.

De todo lo anterior se desprenden los siguientes teoremas:

TEOREMA 1: Una gráfica será un árbol si y sólo si todas sus ramas son desconectantes (Esto implica que para que una gráfica sea un árbol deberá ser ACICLICA).

TEOREMA 2: Una gráfica será un árbol si y sólo si en tre dos nodos cualesquiera (x,y) existe un camino único W(x,y).

TEOREMA 3: Una gráfica será un árbol si y sólo si cualquier rama adicional genera un ciclo (loop).

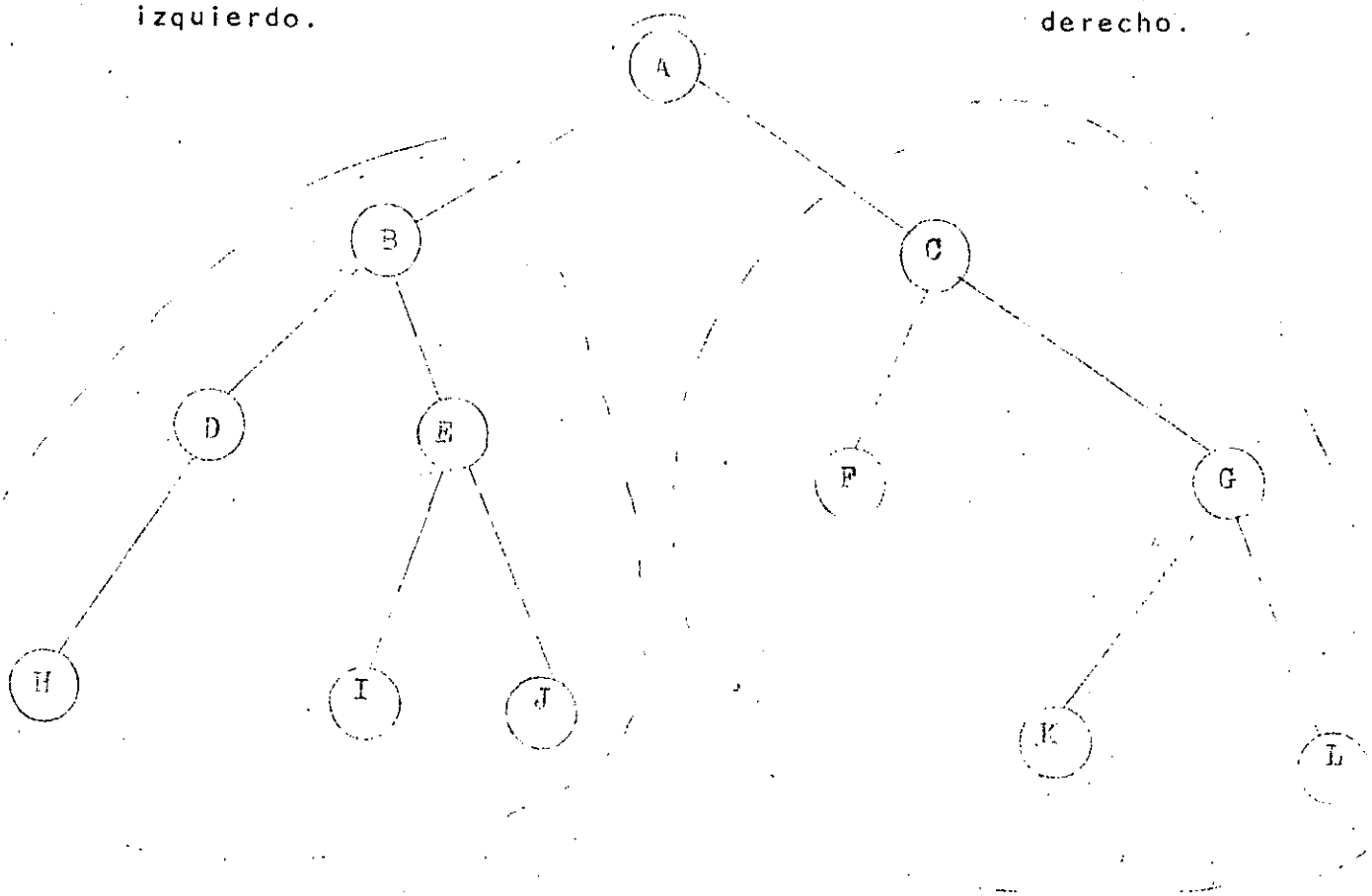
Un árbol ORDENADO es aquel en donde la posición relativa de sus subárboles hace sentido.

Un BOSQUE es un conjunto disjunto de árboles.

De particular interés en computación son los ARBOLES-BINARIOS. Un árbol binario es aquel que tiene a lo más grado 2 en cada uno de sus nodos. De esta propiedad se deduce que en este tipo de árboles, (cabe aclarar que el árbol binario no es precisamente un tipo de árboles), se pueden distinguir dos subárboles en cada nodo: el subárbol derecho y el subárbol izquierdo.

subárbol izquierdo.

subárbol derecho.

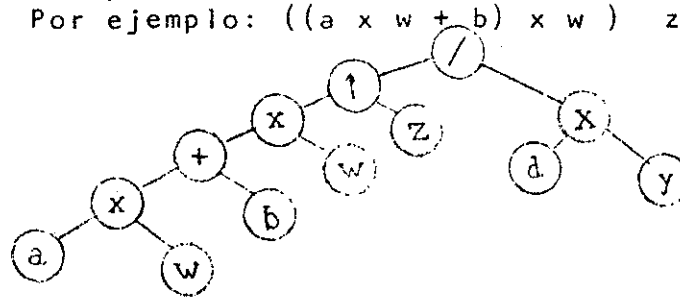


Existen muchas aplicaciones de los árboles binarios, tales como en la solución de arboles de decisión, en la representación de expresiones aritméticas, etc. Veamos un ejemplo:

Cuando se tiene una expresión aritmética, se tienen operandos y operadores (+,-,x,/,).

En el árbol de la expresión, los operandos son los nodos terminales y los operadores son los no-terminales. Su posición en el árbol dependerá de la prioridad con que se deba efectuar cada operación en la evaluación de la expresión.

Por ejemplo: ((a x w + b) x w) z / (d x y)



La representación interna de esta expresión sería, por ejemplo: (- significa, en una liga, nada).

localidad de memoria	liga izquierda	información	liga derecha
1	2	/	3
2	4	↑	5
3	6	x	7
4	8	x	9
5	-	z	-
6	-	d	-
7	-	y	-
8	10	+	11
9	-	w	-
10	12	x	13
11	-	b	-
12	-	a	-
13	-	w	-

Este tipo de representación de expresiones es muy útil en el diseño de compiladores.

Otra manera de usar este tipo de arboles es en la conversión de una expresión aritmética con paréntesis para indicar las prioridades de las operaciones a expresiones en notación polaca.

RECORRIDO DE ARBOLES:

Uno de los principales problemas al trabajar con arboles es el de recorrerlos, es decir, visitar cada uno de los nodos para obtener información.

Hay distintas formas de recorrer un árbol:

-PREORDEN:

- Se visita la raíz
- Se recorre el subárbol izquierdo.
- Se recorre el subárbol derecho.

-ENORDEN:

- Se recorre el subárbol izquierdo.

-Se visita la raíz.

-Se recorre el subárbol derecho.

-POSTORDEN:

-Se recorre el subárbol izquierdo.

-Se recorre el subárbol derecho.

-Se visita la raíz.

Los nombres de preorden, postorden y enorden de estas formas de recorrido se refieren al tiempo en el que se visita la raíz.

A continuación se presentan los algoritmos para recorrer un árbol por medio de las tres maneras mencionadas:

Notación:

- T - apuntador a la raíz del árbol.
- P - apuntador auxiliar para recorrerlo.
- STACK - pila auxiliar.
- Por "visitar nodo P" se entiende que se efectúan las operaciones necesarias. Por ejemplo, imprimir la información del nodo.
- La inserción y supresión de un elemento de la pila se denotará respectivamente por : STACK P y P STACK.

a) Algoritmo para recorrer un árbol binario de PREORDEN:

1 p T
2 si P ≠ - entonces ir a instrucción 4.
3 visitar nodo P
STACK P
P LI(P) e ir a instrucción 2.
4 Si STACK está vacio ir a instrucción 5.
En caso contrario: P STACK
P LD(P)
Ir a instrucción 2.
5 FIN

b) Algoritmo para recorrer un árbol binario en ENORDEN:

1 P T
2 Si P ≠ - entonces STACK P

P LI(P) e ir a instrucción 2.

En caso contrario, si STACK está vacio ir a instrucción 5.

3 P STACK

4 Visitar el nodo P

P LD(P)

ir a instrucción 2.

5 FIN

c) Algoritmo para recorrer un árbol binario en POSTORDEN:

Para este algoritmo se usarán dos pilas STACK, en la forma usual y R donde se meterá un uno si se recorre el subárbol izquierdo y un dos si el derecho, para luego visitar la raíz.

1 P T

2 Si P = - entonces ir a instrucción 8.

3 Si LI(P) ≠ - entonces STACK P
R 1
P LI(P) e ir a instrucción 2.

4 Si LD(P) ≠ - entonces STACK P
R 2
P LI(P) e ir a instrucción 2.

5 Visitar el nodo P

6 Si el STACK está vacio ir a instrucción 8.
En caso contrario: P STACK
K R

7 Si K≠1 entonces ir a instrucción 4.
En caso contrario ir a instrucción 5.

8 FIN.

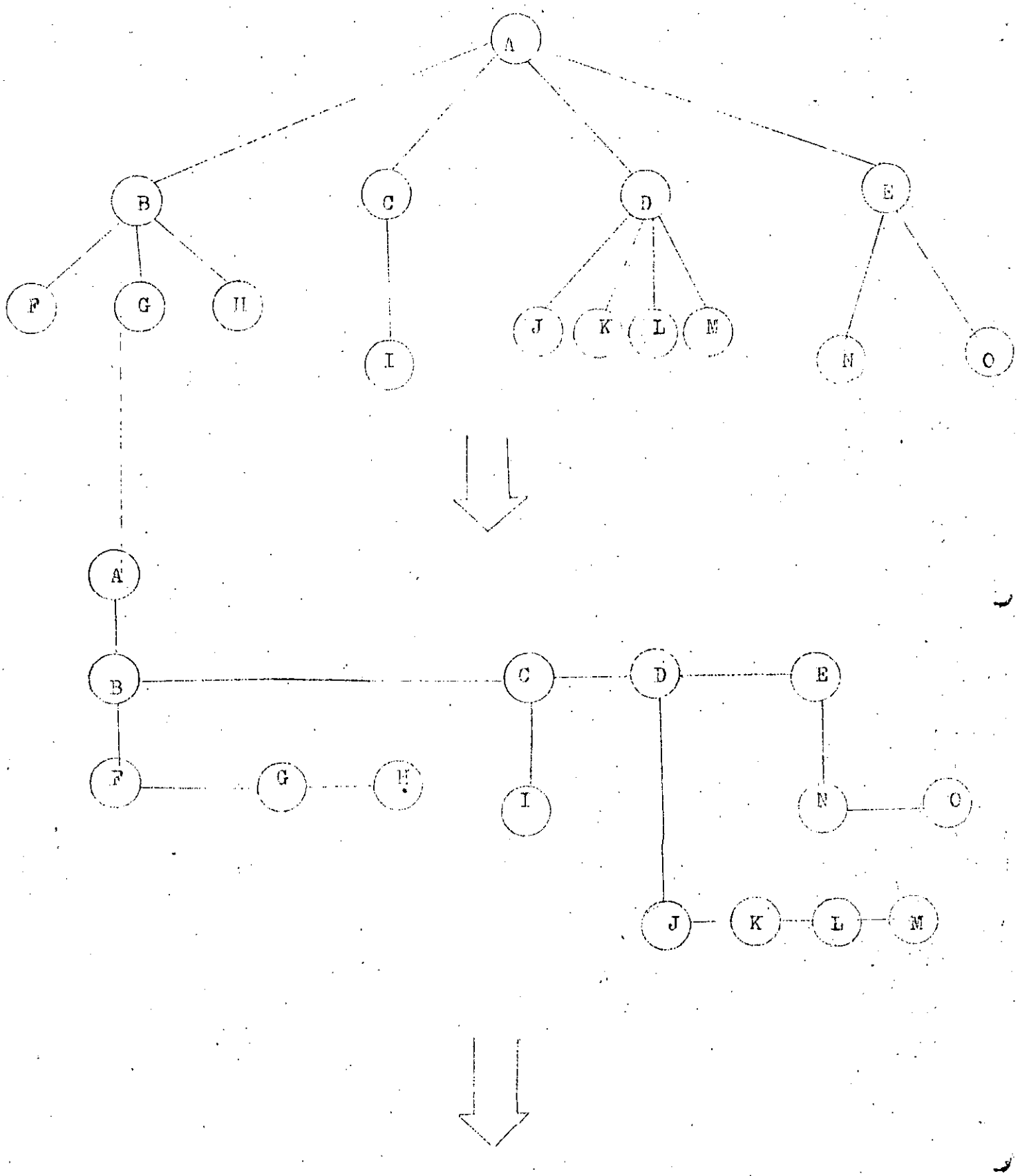
TRANSFORMACION DE CUALQUIER ARBOL A BINARIO:

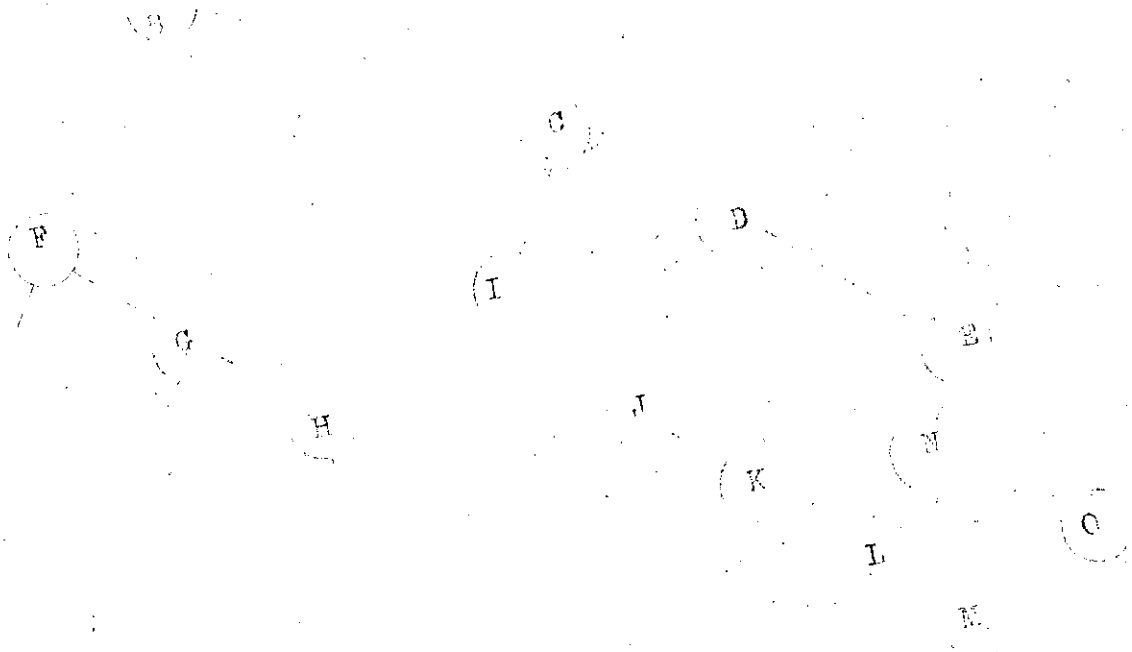
Como se dijo antes, los árboles binarios son una forma natural de representar estructuras arborecentes. A continuación se verá como representar cualquier árbol como uno binario.

La liga izquierda apuntará en primer sucesor de cada nodo, y la liga derecha a los hermanos de dicho nodo.

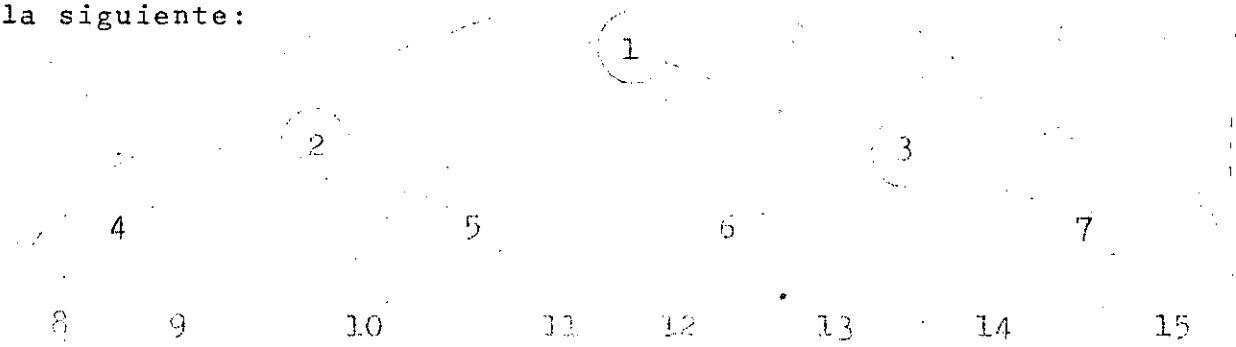
Esto es, la liga izquierda de cada nodo apunta a su primer hijo, y con la derecha a sus hermanos.

Por ejemplo, el árbol no binario representado en la primera figura de la siguiente página queda trans formado en uno binario como se muestra en la siguiente figura.


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
A'
B
C
D
E
F
G
H
I
N
O
J
K
L
M



Una manera de numerar los nodos de un árbol binario muy útil es la siguiente:



Con esta notación es muy fácil saber quién es el padre y quienes los hijos de un determinado nodo:

El padre del nodo K es el entero inmediato inferior de K/2;

Los hijos del nodo K son 2K y 2K+1.

Sin embargo, con esta representación puede haber mucho desperdicio de memoria, pues si no existe un subárbol, se dejará con un símbolo de nulidad la localidad de memoria correspondiente.

Ahora, generalizando las expresiones anteriores para un árbol t'ario, se tendrá:

El padre del nodo K es el entero inmediato superior a -(k-1)/t Los hijos del nodo k serán:

$t(k-1)+2,\ t(k-1)+3,\ \ldots,\ tk + 1.$

LENGUAJE DE PROGRAMACION BASIC II

SISTEMA ADMINISTRATIVO COMPUTARIZADO

NOVIEMBRE DE 1985

I N D I C E

I.-) PRESENTACION

II.-) INTRODUCCION

III.-) PROGRAMAS

IV.-) RUTA DE TRABAJO

2

I.- PRESENTACION

SISTEMA ADMINISTRATIVO POR COMPUTADORA

Este sistema, fue elaborado en el Centro de Cálculo de la FACULTAD DE INGENIERIA, de la UNAM.

OBJETIVO

El objetivo principal del sistema, es la aplicación de las técnicas más avanzadas de planeación y control de la producción, para elevar al máximo la capacidad productiva de una Empresa. Además de realizar de una forma más eficiente.

- La Contabilidad
- La Nómina
- El Control de Inventarios
- El Control de los Deudores y Acreedores
- Control de Trabajadores
- La Facturación
- Etc.

y lo mejor de todo, es que está diseñado para ser utilizado en una Microcomputadora sumamente económica y versátil pues

1.- Tiene el tamaño ideal (aproximadamente como una máquina de escribir.
2.- Fácil de manejar (puede ser utilizado por una secretaria).
3.- Eficiente servicio de mantenimiento (Bajo y muy económico).

3

es una microcomputadora:

APPLE II PLUS

de 64 kbytes de memoria

con 2 unidades de Discos de 5'1/4

IMPRESORA BIDIRECCIONAL que tiene una

velocidad de impresión aproximada de-

125 caracteres por segundo.

II.- INTRODUCCION

Ninguna organización progresista debe ignorar los beneficios del " proceso electrónico de datos" (EDP) pero

¿ Cuál es la manera lógica de empezar ?

¿ Y continuar ?

La experiencia ha demostrado que la aplicación de las técnicas - E.D.P. al Departamento Administrativo produce beneficios tangibles e inmediatos para la mayoria de las organizaciones que las emplean.

La razón no es difícil de encontrar. El Departamento Administrativo es el verdadero corazón de una organización. Dirige todas las- transacciones de ventas y pagos, los cuales proporcionan las estadísticas e información que necesita la Dirección para llegar a una decisión.

Una contabilidad eficáz es esencial para una organización eficiente.

Sin embargo, esto es más fácil decirlo que realizarlo debido a que gran parte del trabajo es fundamentalmente rutinario, tedioso y - quita demasiado tiempo, esto significa que la Dirección pierde -- tiempo esperando información vital acerca del estado de sus Deudores y Acreedores.

5

Basicamente las diferencias que introduce una Microcomputadora son. Eliminación de la rutina repetitivas y tediosas, eliminación de -- errores mediante la comprobación y recomprobación, y que los calculos se realizan electronicamente en una fracción de segundo.

Una Microcomputadora puede de está manera, desarrollar el trabajo,- más rapidamente y con poca posibilidad de error.

Fácil de empezar.

Otra gran ventaja de empezar a trabajar con una Microcomputadora es el hecho de que no interrumpe sus sistemas y procedimientos proba - dos. También, son muy flexibles con respecto al manejo de la docu - mentación lo cuál quiere decir que el formato de sus Fácturas, Es - tados de Cuenta, etc., no necesitan cambiar, además no producen -- ruidos, y se conectan como cualquier otro aparato Eléctrico.

Fácil de ampliar.

Sin embargo el mayor beneficio de las Microcomputadoras viene des - pués de que se han capturados y verificados los datos básicos pu -- diendose entonces a ser procesados, produciendo una información opor tuna acerca del Estado de Acciones. Analisis de Ventas. Deudores. - Acreedores, etc.:

Esta información está siempre al dia y es exacta, debido a que está basada en datos correctos como pueden ser el último pedido ó pago.- Por lo tanto, las Microcomputadoras le permiten a Ud. introducir el "E.D.P." en el corazón de su organización, (EL DEPARTAMENTO ADMINIS

TRATIVO) y también le permite expander y extender los beneficios al- Cerebro de la Organización, (LA DIRECCION)

Por eso las Microcomputadoras le proporcionan la manera lógica de - empezar el E.D.P. y continuarlo tal como se lo demostrará el resto - de este proyecto.

III.- PROGRAMAS

A.- FACTURACION

1) Facturación a trabajos del taller.

2) Facturación a Ventas de Mostrador.

A1 FACTURACION A TRABAJOS DE TALLER.

Este programa tiene como objetivo principal, la elaboración de una factura, sin embargo, no es lo único que realiza, pués aparte de ésto, efectúa actualizaciones a los archivos, tanto ó más importantes que la propia facturación. Es decir una vez elaborada una factura (dandole tan solo el tipo de factura y el número de las operaciones que va a facturar) el programa tomará de su archivo de clientes todos aquellos datos que sean necesarios para imprimir la factura (el tiempo aproximado para elaborar, totalmente una factura es de 100 segundos) por otra parte éste programa verificará que; cuando la factura sea a crédito, el importe de ésta, no exceda la cantidad máxima de crédito autorizada para ese cliente, pudiendo entonces suceder alguna de las 2 siguientes cosas (1F)

a) Si no pasa (ELSE)

El programa preguntará al operador si imprime ó nó la factura, si el operador responde "NO", el proceso termina. en caso afirmativo pasar al inciso "b"

8

b) Si si pasa (THEN)

Se imprimirá la factura y además, actualizará los siguientes datos:

b1- Importe de sus compras a crédito.

b2- Número total de compras.

b3- Importe total de todas sus compras.

b4- Fecha de la última compra.

b5- Sumaría una venta más, a ese tipo de factura, unicamente con fines estadisticos.

A2 FACTURACION A VENTAS DE MOSTRADOR

Este programa es similar al anterior sin embargo por las características propias de ésta operación, además de realizar la actualización de esos datos (menos el b5) actualizará directamente el inventario sirviendo entonces como "vales de salidas de Almacén" las propias facturas de mostrador.

NOTA 1: Debido a la capacidad de Almacénamiento de Datos de una Microcomputadora cuándo el número de Productos diferentes en el inventario es muy grande, será necesario descriminar algunos productos y dejar tán solo los más importantes (en el proceso normal), realizando entonces la actualización de los datos restantes al fin del Día.

NOTA 2: Recuerde que para que puedan trabajar estos programas, es necesario que ya hayan sido cargados y verificados los archivos de Cliente, Facturas tipo, así como los datos de to-

dos los productos de su Inventario, lo cuál puede tardarse dependiendo del tamaño de sus archivos desde "unas horas " hasta "varias semanas".

B.- INVENTARIOS

El objetivo principal de este programa es el de:

- Elaborar un reporte de todos los productos existentes en almacén así como sus costos.
- Elaborar un reporte que indique:
  Cuales productos están por debajo de su nivel mínimo para que posteriormente el Departamento de Compras realize el pedido.
- Permitir al operador accesar en forma directa e inmediata sus Archivos de Inventario.

NOTA 1: Las 2 únicas ventajas que tiene éste programa con respecto a los procedimientos normales es que:

a) automaticamente detecta aquellos productos que están abajo del punto de reorden eliminando en gran parte los faltantes en Almacén.

b) poder sacar los reportes de existencias, con la frecuencia que se requiera (diaria, semanal, quincenal, ó mensual).

C.- PRONOSTICOS:

Esté es desde el punto de vista productivo, el programa más importante de todo el sistema debido a que gracias a la información que proporcio na se tomarán "la mayoria" de las decisiones importantes de la Empresa tales como:

- Cuándo y con que condiciones de pago puede efectuar una inversión en maquinaria, terrenos, sistemas, viajes, etc.
- En cuánto a las decisiones relacionadas con el almacén:
  - Cuándo Comprar
  - De que tipo Comprar
  - Cuanto Comprar

Y en caso dado que estos procedimientos se hayan aplicado a datos contables como: Gastos de Fabricación del mes, Gastos de Mantenimiento - del mes, Utilidades del mes, etc., se podría conocer el futuro de la - Empresa (con probabilidades de "Acertar" que varian desde el 55% hasta el 95%) y poder mejorarlo notablemente aumentando con esto las utilida des finales, los procedimientos de Pronosticos que se utilizarán son:

- Estadísticas/Uno
- Estadísticas/Dos
- Estadísticas/Tres
- Promedio movil con Ajuste (2 Términos)
- Promedio movil con Ajuste (4 Términos)

- Promedio ponderado Exponencialmente (ALFA=0.3)
- Promedio ponderado exponencialmente Ajustado (ALFA=0.3)
- Promedio movil simple sin estacionalidad (4 Términos)
- Promedio movil simple con Estacionalidad (4 Términos)
- Promedio movil Ajustado sin estacionalidad (4 Términos)
- Promedio ponderado Exponencialmente simple sin estacionalidad - (ALFA=0.3)
- Promedio ponderado exponencialmente simple con estacionalidad - (ALFA=0.3)
- Promedio ponderado exponencialmente ajustado con estacionalidad (ALFA=0.3)
- Curva Exponencial sin Estacionalidad
- Curva Exponencial con Estacionalidad

NOTA 1: En la Empresa Lozano y Vazquez, S.A., estos procedimientos - aplicados exclusivamente a las ventas funcionan con 5% de - error (95% de Seguridad) lo cuál nos ayudó para la toma de - decisiones y se ha logrado que la producción en los 1os. 45- días de 1982 supere a la de los 1os. 45 días de 1981, con 36 motores, es decir generó un aumento del 22.5% en la produc - ción promedio diaria.

Este porcentaje "Tomelo muy en cuenta", para decidir si utiliza ó no éste sistema ya que ésto porcentaje aplicado a sus niveles de producción, le indicará el tiempo de recuperación, de su inversión.

Sin tomar en cuenta los beneficios adicionales que no se tomáron en cuenta en éste ejemplo ilustrativo.

D.- NOMINA

El objetivo principal de este programa es la elaboración tanto de los sobres, como de las listas de raya, calculando automaticamente todas aquellas deducciones y percepciones a las que incurre el asalariado.

Sin embargo ésto no es lo único que realiza, pues aparte de esto, se efectuan actualizaciones a los archivos de trabajadores y empleados, tan importantes como la propia nómina, por ejemplo:

- Actualiza los dias de vacaciones.
- Actualiza las estadísticas de asistencia.
- Actualiza los cobros pendientes.
- Actualiza la producción promedio.
- Actualiza el ahorro acumulado.
- Actualiza los impuestos retenidos.
- Etc.

NOTA 1: Nuevamente recuerde que gracias a las estadisticas y al conocimiento inmediato de los datos actualizados de todo su personal, podrá tomar decisiones sumamente importantes tales como:

- Ajustar en base a su eficiencia promedio semanal, el salario a un trabajador.

13

(subirlo ó bajarlo)

- Saber que elementos son valiosos en su Empresa, y cuidarlos.
- Poder planear la producción conjuntamente con el departamento de compras "ASEGURANDO" en base a las estadísticas de asistencias y eficiencia de sus trabajadores que se alcanzarán los objetivos.
- Poder planear en base al inciso anterior el Mantenimiento Preventivo de su maquinaria, recuerde que el mantenimiento preventivo en ocaciones representa el 1% del costo del mantenimiento correctivo.

¿ Alguna vez penso usted que todo eso sólo se puede hacer auxiliandose de las Estadísticas generadas por la Nómina?

NOTA 2: Recuerde que para que éste programa funcione se tiene primeramente que cargar toda la información de sus trabajadores, así como las tablas de impuestos actualizadas.

E.- PROGRAMA DE CREDITO Y COBRANZA

El objetivo principalmente de éste programa es tener anticipadamente, los reportes tanto de cobros como de pagos reduciendo con ésto las pérdidas de dinero por:

a) Retraso en los Cobros.

b) Descuentos por pronto Pago.

Si toma usted en cuenta la tasa de interes bancaria actual, así como el importe de sus movimientos a crédito, podrá medir la importancia de éste programa en su Empresa.

F.- CONTABILIDAD

El objetivo de éste programa es la elaboración de todos los libros contables que usted utilice, así como la elaboración simultanea de las pólizas, éste programa está diseñado para trabajar únicamente con "Cuentas y subcuentas"

G.- PROGRAMA EXTRAS

Tomando en cuenta que todos estos programas actuan como partes de un sistema y no son independientes de los demás (que es el caso de los paquetes comerciales) se proporcionan además una serie de programas tales como:

- Programa creación y actualización del archivo de trabajadores y empleados.
- Programa creación y actualización del archivo de clientes.
- Programa creación y actualización del archivo de proveedores.
- Programa creación y actualización del archivo de inventarios
- Programa que imprime las etiquetas de los sobres de correspondencia.
- Programa que escribe las cartas de felicitaciónes por honomásticos.
- Un disco lleno de juegos (aprox. 32 juegos).

IV.- RUTA DE TRABAJO

Difícilmente se puede establecer una ruta de trabajo para instalar todo un sistema Administrativo, pues el tiempo de creación de los archivos de trabajo depende como ya se mencionó del tamaño de los mismos y de lo organizada ó desorganizada que esté la información para poder darsela a la computadora. Por otra parte dependiendo de los volúmenes de trabajo en c/operación, "será ó no", necesario adquirir una o dos computadoras más, para poder instalar totalmente el sistema, y esto depende de los recursos económicos con que se cuente (pudiendose entonces retrasar la instalación hasta varios meses).

Sin embargo sí podemos recomendar el orden en que deben de irse instalando los programas:

1.- Facturación

2.- Pronosticos

3.- Inventarios

4.- Nómina

5.- Crédito y Cobranza

6.- Contabilidad

I N D I C E

1. INTRODUCCION

2. ESTADISTICAS Y BASE DE DATOS DEL ESTUDIO

3. METODOS DE PRONOSTICOS UTILIZADOS

4. PRONOSTICO DE VENTAS 1983

* Gráfica 1983

* Gráfica Comparativa 1980/1983

5. CONCLUSIONES Y SUGERENCIAS

* Gráfica de Eficiencia

-*-

17

17

1. INTRODUCCION *

18

Introducción:

A continuación se presenta, el estudio de Prónosticos, realizado en Lozano y Vázquez S.A. cuyo objetivo principal es conocer los pronósticos de ventas de la empresa, y con ello poder elaborar los programas anuales de mantenimiento preventivo, controlar los inventarios, - controlar las vacaciones del personal, y obviamente para poder evaluar la reacción que produce, en nuestras- ventas, la toma de una nueva decisión.

-*-

19

2. ESTADISTICAS Y BASE DE DATOS DEL ESTUDIO *

20

Para poder elaborar un estudio de pronóstico; es necesario tener las estadisticas de ventas por lo menos de los últimos 3 años, y para no tener que hacer conversiones de dinero de valor pasado a valor presente, ó, valor futuro, se decidió elaborar este estudio en base al número de unidades (en éste caso motores) vendidas por mes, por lo que se elaboró primeramente un programa en la computadora, que pedía unicamente la fecha (DIA/MES/AÑO) y la marca (VW-1500, VW-1600, DODGE 225, DODGE 360, etc).

En las tablas 1 y 2, se muestran los resultados finales que imprime éste programa.

TABLA # 1 21

ESTADISTICAS GENERALES
1981.

MES/DIA	D	L	M	M	J	V	S	*	TOTAL
ENE	0	15	17	25	12	14	11	*	94
FEB	0	13	20	16	12	23	8	*	92
MAR	0	22	24	17	22	13	11	*	109
ABR	0	16	20	18	17	16	12	*	99
MAY	0	14	18	15	22	18	14	*	101
JUN	1	20	12	13	25	22	10	*	103
JUL	0	14	16	17	13	16	12	*	88
AGO	0	10	10	15	23	17	14	*	89
SEP	0	19	12	14	10	21	16	*	92
OCT	0	19	20	17	26	20	10	*	112
NOV	2	16	17	18	17	24	10	*	104
DIC	0	27	28	27	16	13	9	*	120
TOTAL	3	205	214	212	215	217	137	*	1203

1203 MOTORES RECTIFICADOS
EN EL AÑO DE 1981.

TABLA # 2
ESTADISTICAS GENERALES 22
1982.

MES/DIA	D	L	M	M	J	V	S		TOTAL
ENE	1	22	22	14	24	30	13	*	126
FEB	0	24	18	20	20	15	9	*	106
MAR	0	22	29	22	24	16	11	*	124
ABR	2	17	13	18	20	19	11	*	100
MAY	0	9	16	16	21	15	11	*	88
JUN	1	17	16	23	15	13	13	*	98
JUL	0	15	14	29	21	17	16	*	112
AGO	0	15	18	20	22	27	21	*	123
SEP	0	16	21	21	20	26	12	*	116
OCT	0	12	16	17	14	21	16	*	96
NOV	0	19	24	23	25	10	11	*	112
DIC	1	23	33	34	20	15	12	*	138
TOTAL	5	211	240	257	246	224	156	*	1339

1339 MOTORES RECTIFICADOS
EN EL AÑO DE 1982.

23

Adicionalmente a éstas tablas, se imprimieron las tablas individuales por marca, con las cuales se puede estudiar técnicamente la estacionalidad de cada motor (si existiera), y la tendencia lineal para 1983 de c/u de ellos.

En las tablas 3 y 4 se muestra el resumen de éstas (en éste estudio analizaron 30 motores por año con tablas similares a la 1 y 2 pero por simplicidad solo se muestra el resumen final).

Por otra parte, en la práctica, no fue posible el recabar información similar de 1980, pero tomando en cuenta los estados financieros de ese año, y los precios de venta promedio de un motor se "estimó" el número total de motores vendidos por mes.

TABLA # 3
ANALISIS ANUAL 1981.

M O T O R	MAQUINA	# DE MOTOR	%	OBSERVACIONES	LUGAR MARCA	LUGAR INDIV
Chevrolet	230	11	0.9%			
Chevrolet	235	8	0.66%			
Chevrolet	250	48	4.0%	137→11.39%	4	4
Chevrolet	292	57	4.7%			
Chevrolet	350	13	1 %			
Datsun	1300	12	0.9%			
Datsun	1500	86	7.1%		6	5
Datsun	1600	13	1 %	111→ 9.22%		
Dodge	170	30	2.4%			
Dodge	225	156	13.21%			2
Dodge	318	75	6.2%	275→22.85%	1	1
Dodge	360	14	1.1%			
Ford	260	12	0.9%			
Ford	289	62	5.1%			
Ford	302	47	3.2%		3	
Ford	335	29	2.4%	205→17.0 %		
Ford	351	55	4.5%			
Opel	153	21	1.7%		8	
Opel	1700	4	0.3%	25→ 2 %		
Perkins		5	0.3%	5→ 0.4%	8	
Rambler	199	12	0.9%			
Rambler	232	15	1.2%			
Rambler	252	21	1.7%		5	
Rambler	258	63	5.2%			
Rambler	282	8	0.6%	119→ 9.9%		
Renault	1300	61	5 %	61→ 5 %	7	
V.W.	1200	18	1.5%			
V.W.	1500	88	7 %		2	
V.W.	1600	127	10 %	233→19.36%		3
OTROS		32	2.6%	32→ 2.66%	8	

TABLA # 4
ANALISIS A' ', 1982.

MOTOR	MAQUINA	# DE MOTOR	%	OBSERVACIONES	LUGAR FOR MAR	LUGAR INDIV
Chevrolet	230	21	1.56%			
Chevrolet	235	8	0.6%			
Chevrolet	250	33	2.46%		4	
Chevrolet	292	33	2.46%	181→13.51%		
Chevrolet	350	86	6.42%			4
Datsun	1300	10	0.75%			
Datsun	1500	56	4.18%	129→9.63%	6	8
Datsun	1600	63	4.7%			6
Dodge	225	227	16.95%			1
Dodge	318	60	4.48%	314→23.45%	1	7
Dodge	360	27	2.0%			
Ford	260	21	1.59%			
Ford	289	30	2.24%			
Ford	292	10	0.74%		3	
Ford	302	78	5.82%	215→16.05%		6
Ford	335	30	2.24%			
Ford	351	46	3.43%			9
Rambler	199	7	0.5%			
Rambler	232	15	1.12%			
Rambler	252	14	1.04%			
Rambler	258	89	6.64%	150→11.20%	5	3
Rambler	282	25	1.85%			
Renault	1300	63	4.70%	63→4.70%	7	6
V.W.	1500	84	6.27%		2	5
V.W.	1600	165	12.32%	249→18.59%		2
OTROS	-	38	2.83%	38→2.83%	8	

25

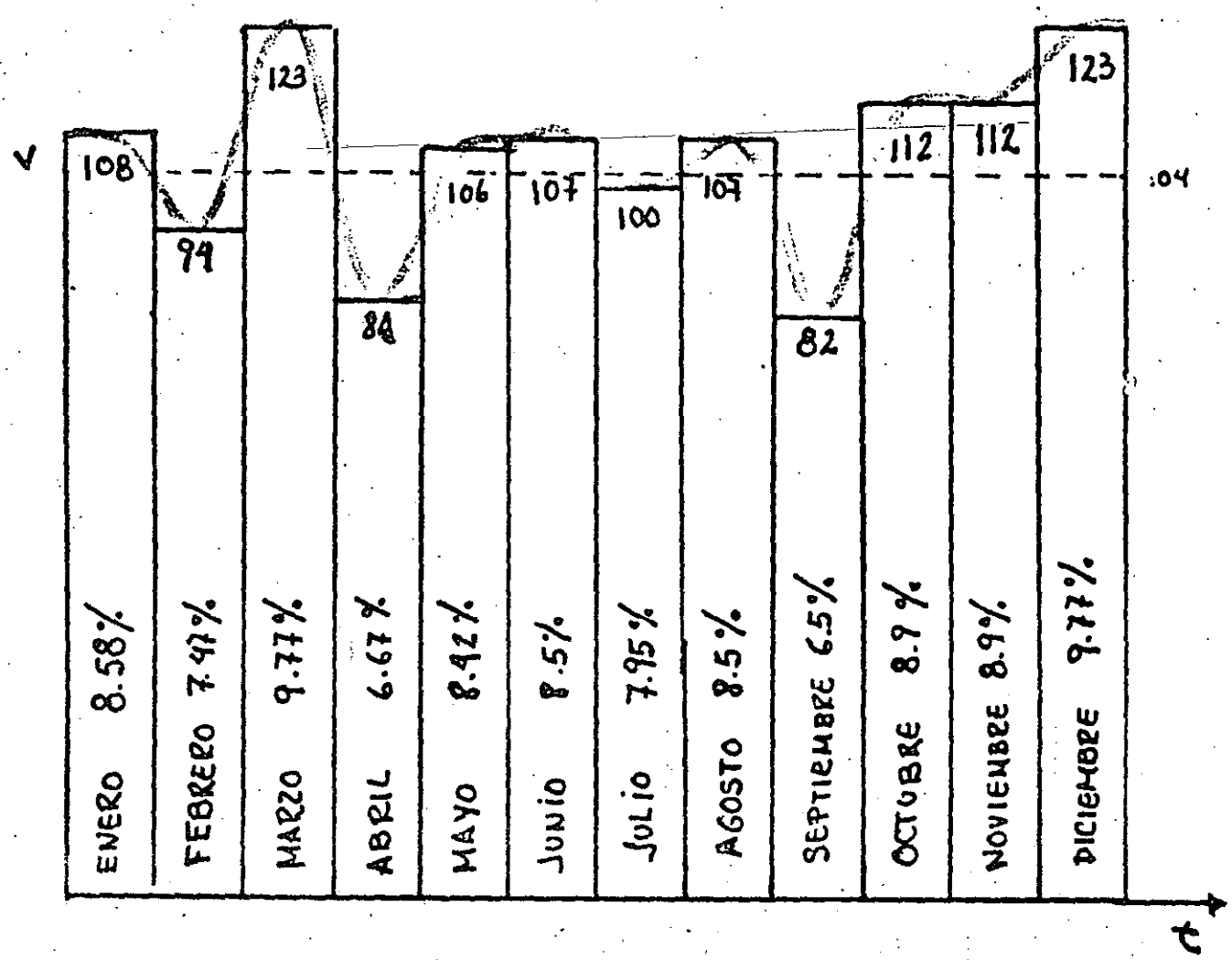
26

Las gráficas 1,2,3, muestran el comportamiento de las ventas en 1980, 1981,1982, respectivamente, y en la - gráfica 4; se muestran, éstas tres curvas juntas para poder analizar un poco su comportamiento.

La línea punteada, es la gráfica resultante de calcular el promedio aritmético mensual de las ventas de - los 3 años.

Una vez que se contó con toda ésta información, se - procedio a la elaboración de los programas para cal - cular los pronósticos de ventas para 1983, utilizando los siguientes metodos:

VENTAS 1980
(1258 motores)



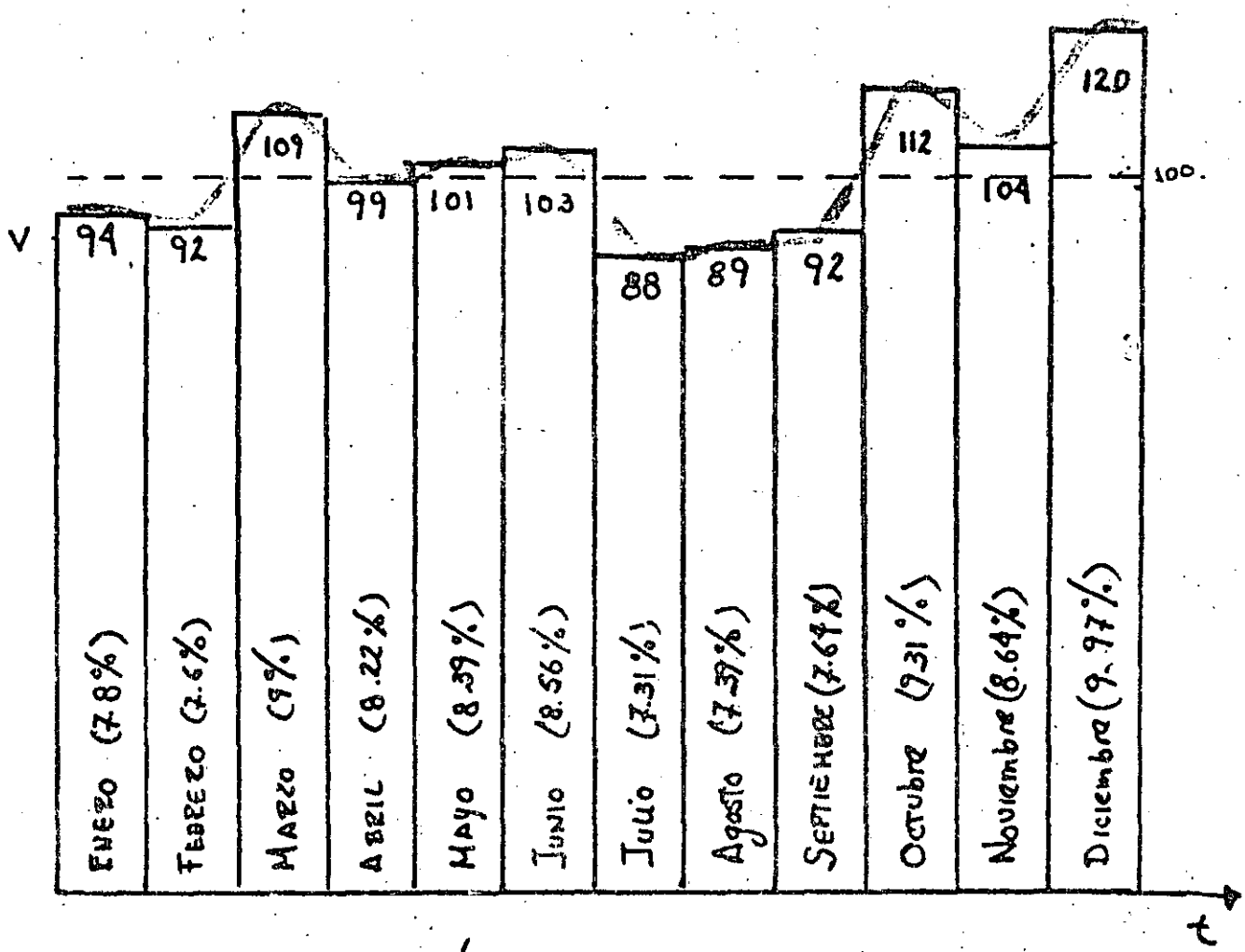
Promedio = 109 motores/mes
= 4.12 motores/día

Q #1

28

VENTAS 1981

(1203 motores)

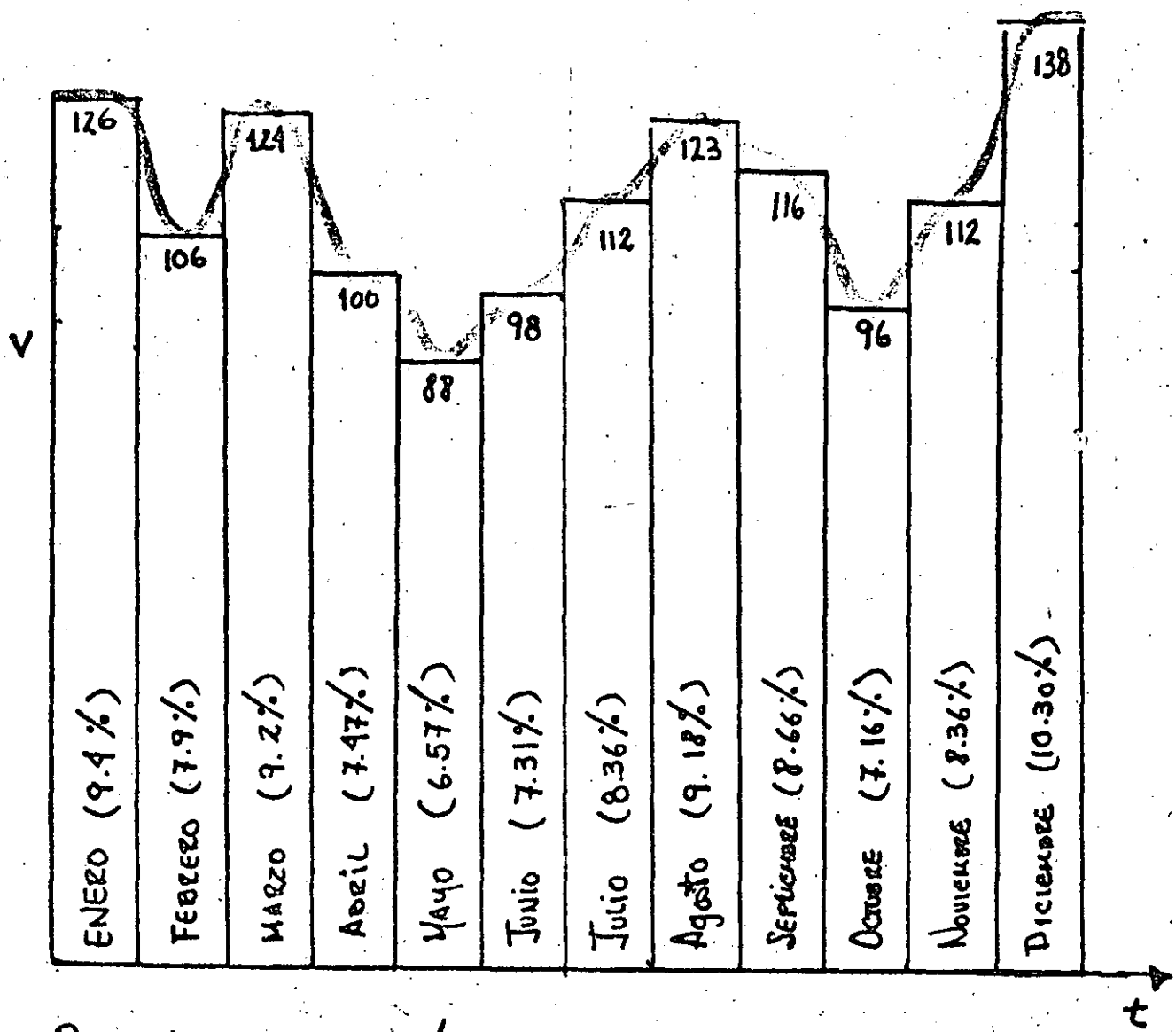


Promedios = 100 motores/mes

= 3.94 motores/día.

g # 2

VENTAS 1982
(1339 motores)

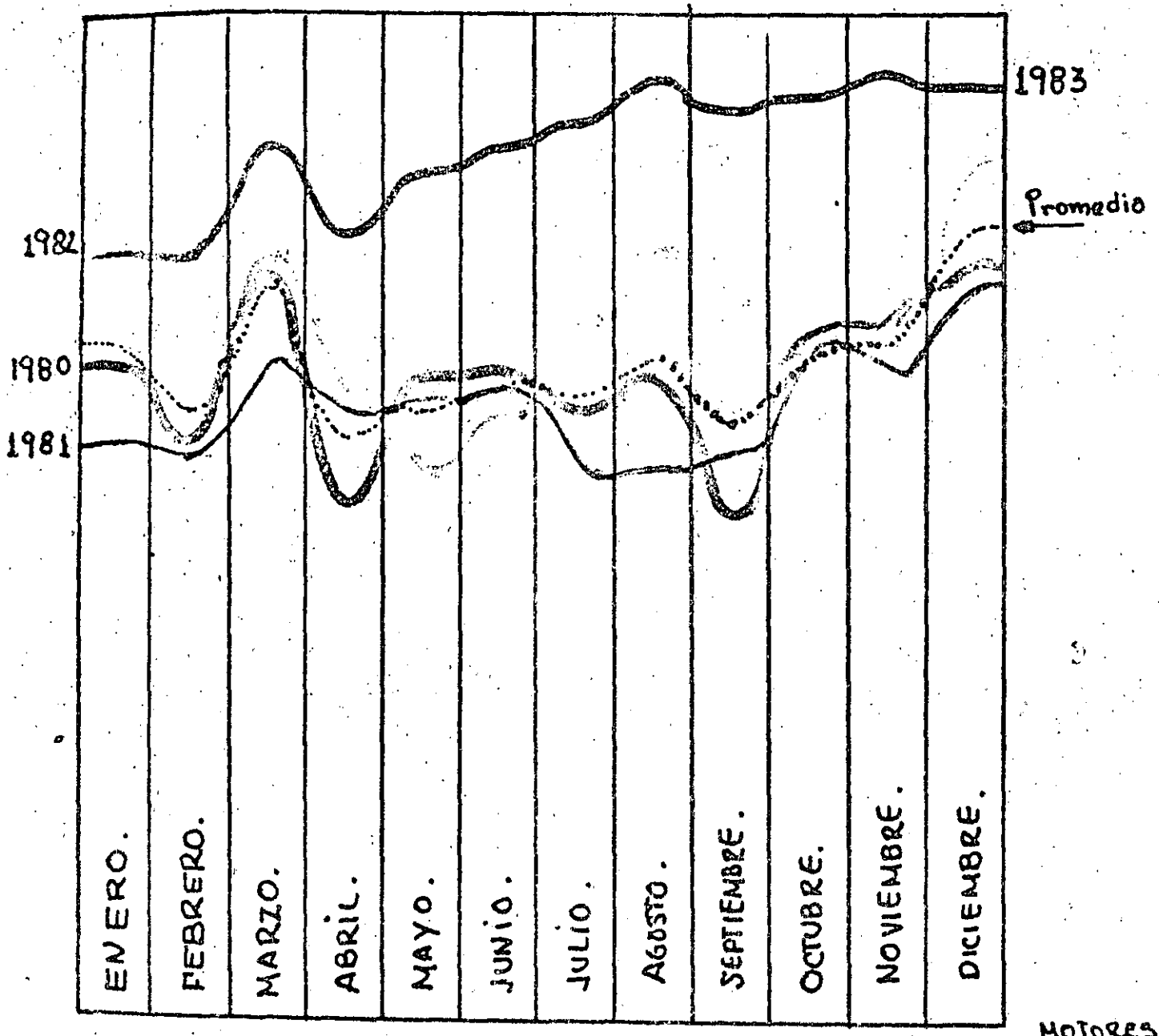


Promedio = 114 motores/mes.

= 4.39 motores/día.

Analisis Global
— 80/81/82 —

30



													Motores
1980	108	94	123	84	106	107	100	107	82	112	112	123	1258
1981	94	92	109	99	101	103	88	89	92	112	104	120	1203
1982	126	106	124	100	88	98	112	123	116	96	112	138	1339
Medio →	109	97	118	94	98	102	100	106	961	106	109	127	1266

31

3. METODOS DE PRONOSTICOS UTILIZADOS *

31

32

32

Metodos de Pronósticos utilizados:

a) Promedio movil con ajuste 2 términos.
b) Promedio movil con ajuste 4 términos.
c) Promedio ponderado exponencialmente Alfa=0.3
d) Promedio ponderado exponencialmente ajusta Alfa=0.3
e) Promedio movil simple sin estacionalidad de 4 términos.
f) Promedio movil simple con estacionalidad de 4 términos.
g) Promedio movil ajustado sin estacionalidad de 4 términos.
h) Promedio ponderado exponcialmente ajustado con estacionalidad Alfa=0.3
i) Promedio ponderado exponencialmente simple sin estacionalidad Alfa=0.3
j) Promedio ponderado exponencialmente simple con estacionalidad Alfa=0.3
k) Curva exponencial con estacionalidad.
l) Curva exponencial sin estacionalidad.
m) Línea recta.

33

TABLA # 5

PROMEDIO MOVIL AJUSTADO DE 2 TERMINOS

M E S	VENTA	PRONOSTICO	% E
ENE 1	108	0	0
FEB 2	94	0	0
MAR 3	123	0	0
ABR 4	84	119.75	.495
MAY 5	106	96	.094
JUN 6	107	82.25	.231
JUL 7	100	123.75	.237
AGO 8	107	99	.074
SEP 9	82	103.5	.262
OCT 10	112	81	.267
NOV 11	112	100.75	.100
DIC 12	123	134.5	.093
ENE 13	94	125.75	.337
FEB 14	92	95	.032
MAR 15	109	69.75	.360
ABR 16	99	111.75	.128
MAY 17	101	109.25	.081
JUN 18	103	94	.087
JUL 19	88	105	.193
AGO 20	89	85.75	.036
SEP 21	92	78	.152
OCT 22	112	93.5	.165
NOV 23	104	119.25	.146
DIC 24	120	117	.025
ENE 25	126	118	.063
FEB 26	106	139.5	.316
MAR 27	124	105.5	.149
ABR 28	100	113.5	.135
MAY 29	88	107.5	.221
JUN 30	98	67	.316
JUL 31	112	91.5	.183
AGO 32	123	123	0
SEP 33	116	136.25	.174
OCT 34	96	122.5	.276
NOV 35	112	85.75	.234
DIC 36	138	101	.268

ERROR PROMEDIO .163

PROMEDIO MOVIL AJUSTADO DE 4 TERMINOS SIN ESTACIONALIDAD

M E S	VENTA	PRONOSTICO	% E
ENE 1	108	0	0
FEB 2	94	0	0
MAR 3	123	0	0
ABR 4	84	0	0
MAY 5	106	0	0
JUN 6	107	0	0
JUL 7	100	0	0
AGO 8	107	94.56	.116
SEP 9	82	108.75	.326
OCT 10	112	93.89	.161
NOV 11	112	99.20	.114
DIC 12	123	105.54	.141
ENE 13	94	115.27	.226
FEB 14	92	118.58	.289
MAR 15	109	103.16	.053
ABR 16	99	100.64	.016
MAY 17	101	88.29	.125
JUN 18	103	97.12	.570
JUL 19	88	105.39	.197
AGO 20	89	94.20	.058
SEP 21	92	88.89	.033
OCT 22	112	85.91	.232
NOV 23	104	95.14	.085
DIC 24	120	105.18	.123
ENE 25	126	120.95	.040
FEB 26	106	134.25	.266
MAR 27	124	122.43	.012
ABR 28	100	127.54	.275
MAY 29	88	111.29	.264
JUN 30	98	90.54	.076
JUL 31	112	90	.196
AGO 32	123	90.12	.267
SEP 33	116	109.10	.059
OCT 34	96	124.54	.297
NOV 35	112	119.45	.065
DIC 36	138	114.25	.172
ENE 37	119.97	119.97	0
FEB 38	120.86	120.86	0
MAR 39	132.87	132.87	0
ABR 40	140.04	140.04	0
MAY 41	136.01	136.01	0
JUN 42	140.06	140.06	0
JUL 43	146.80	146.80	0
AGO 44	150.75	150.75	0
SEP 45	151.65	151.65	0
OCT 46	155.88	155.88	0
NOV 47	160.59	160.59	0
DIC 48	163.96	163.96	0

TABLA # 7

PROMEDIO PONDERADO EXPONENCIALMENTE ALFA=0.3 35

35

M E S	VENTA	PRONOSTICO	% E
ENE 1	108	0	0
FEB 2	94	108	.148
MAR 3	123	103.8	.156
ABR 4	84	109.56	.304
MAY 5	106	101.89	.038
JUN 6	107	103.12	.042
JUL 7	100	104.28	.042
AGO 8	107	103.00	.037
SEP 9	82	104.20	.270
OCT 10	112	97.54	.129
NOV 11	112	101.87	.090
DIC 12	123	104.91	.147
ENE 13	94	110.34	.173
FEB 14	92	105.43	.146
MAR 15	109	101.40	.069
ABR 16	99	103.68	.047
MAY 17	101	102.27	.012
JUN 18	103	101.89	.010
JUL 19	88	102.22	.161
AGO 20	89	97.95	.100
SEP 21	92	95.27	.035
OCT 22	112	94.28	.158
NOV 23	104	99.60	.042
DIC 24	120	100.92	.158
ENE 25	126	106.64	.153
FEB 26	106	112.45	.060
MAR 27	124	110.51	.108
ABR 28	100	114°56	.145
MAY 29	88	110.19	.252
JUN 30	98	103.53	.056
JUL 31	112	101.87	.090
AGO 32	123	104.91	.147
SEP 33	116	110.33	.048
OCT 34	96	112.03	.167
NOV 35	112	107.22	.042
DIC 36	138	108.65	.212

ERROR PROMEDIO .111

TABLA # 8

PROMEDIO PONDERADO EXPONENCIALMENTE

ALFA=0.3 CON ESTACIONALIDAD.

36

M E S	VENTA	PRONOSTICO	% E
ENE 1	108	108.58	0
FEB 2	94	96.66	.028
MAR 3	123	117.85	.041
ABR 4	84	93.68	.115
MAY 5	106	97.66	,078
JUN 6	107	101.96	.047
JUL 7	100	99.31	6.842
AGO 8	107	105.60	.013
SEP 9	82	96.00	.170
OCT 10	112	105.93	.054
NOV 11	112	108.58	.030
DIC 12	123	126.13	.025
ENE 13	94	103.83	.104
FEB 14	92	92.44	4.794
MAR 15	109	112.70	.033
ABR 16	99	89.59	.095
MAY 17	101	93.39	.075
JUN 18	103	97.50	.533
JUL 19	88	94.97	.079
AGO 20	89	100.98	.134
SEO 21	92	91.80	2.088
OCT 22	112	101.30	.095
NOV 23	104	103.83	1.558
DIC 24	120	120.61	5.138
ENE 25	126	115.57	.082
FEB 26	106	102.89	.029
MAR 27	124	125.44	.011
ABR 28	100	99.72	2.797
MAY 29	88	103.94	.181
JUN 30	98	108.52	.107
JUL 31	112	105.71	.056
AGO 32	123	112.40	.086
SEP 33	116	102.18	.199
OCT 34	96	112.75	.174
NOV 35	112	115.57	.031
DIC 36	138	134.25	.027

ERROR PROMEDIO .061

PROMEDIO PONDERADO EXPONENCIAL AJUSTADO ALFA=0.3

37

AÑO/MES		VENTA	PRONOSTICO	% E
	ENE 1	108	108	0
	FEB 2	94	99.59	.059
	MAR 3	123	122.38	.086
	ABR 4	84	96.12	.145
	MAY 5	106	100.37	.053
80	JUN 6	107	103.52	.032
	JUL 7	100	101.17	.011
	AGO 8	107	104.12	.026
	SEP 9	82	90.82	.107
	OCT 10	112	101.51	.093
	NOV 11	112	107.69	.038
	DIC 12	123	117.71	.042
	ENE 13	94	105.69	.124
	FEB 14	92	97.55	.060
	MAR 15	109	103.26	.052
	ABR 16	99	100.32	6.700
	MAY 17	101	100.58	.0159
81	JUN 18	103	101.45	.014
	JUL 19	88	93.15	.058
	AGO 20	89	89.21	2.441
	SEP 21	92	89.07	.031
	OCT 22	112	101.26	.095
	NOV 23	104	103.40	5.727
	DIC 24	120	114.10	.049
	ENE 25	126	123.48	.019
	FEB 26	106	116.30	.097
	MAR 27	124	122.65	.010
	ABR 28	100	111.49	.114
	MAY 29	88	97.78	.111
82	JUN 30	98	96.18	.018
	JUL 31	112	103.96	.071
	AGO 32	123	115.10	.064
	SEP 33	116	117.07	9.256
	OCT 34	96	105.93	.103
	NOV 35	112	109.19	.025
	DIC 36	138	126.63	.082
	ENE 37		129.39	
	FEB 38		132.14	
	MAR 39		134.90	
	ABR 40		137.65	"Pronóstico"
	MAY 41		140.41	
83	JUN 42		143.16	1983
	JUL 43		145.92	**
	AGO 44		148.67	
	SEP 45		151.33	E = .1048
	OCT 46		154.19	
	NOV 47		156.94	
	DIC 48		159.70	

TABLA # 10
ESTADISTICAS ANUALES

38

	VENTA 1980	VTA. 1981	VTA. 1982	PRO/A* 1983	PRO/B 1983	PRO/C 1983
ENERO	108	94	126	129.39	120	130
FEBRERO	94	92	106	132.15	121	116
MARZO	123	109	124	134.90	133	141
ABRIL	84	99	100	137.65	140	112
MAYO	106	101	88	140.41	136	117
JUNIO	107	103	98	143.16	140	122
JULIO	100	88	112	145.92	147	119
AGOSTO	107	89	123	148.67	151	126
SEPTIEMBRE	82	92	116	151.43	152	115
OCTUBRE	112	112	96	154.19	155	127
NOVIEMBRE	112	104	112	156.94	160	130
DICIEMBRE	123	120	138	159.70	163	151

Prónostico A= Promedio ponderado exponencialmente ajustado x=0.3

Pronóstico B= Promedio movil ajustado de 4 términos s-estacionalidad

Pronóstico C= Promedio ponderado exponencialmente ajustado con estacionalidad x=0.3

39

4. PRONOSTICO DE VENTAS 1983 *

* Gráfica 1983

* Gráfica Comparativa 1980/1983

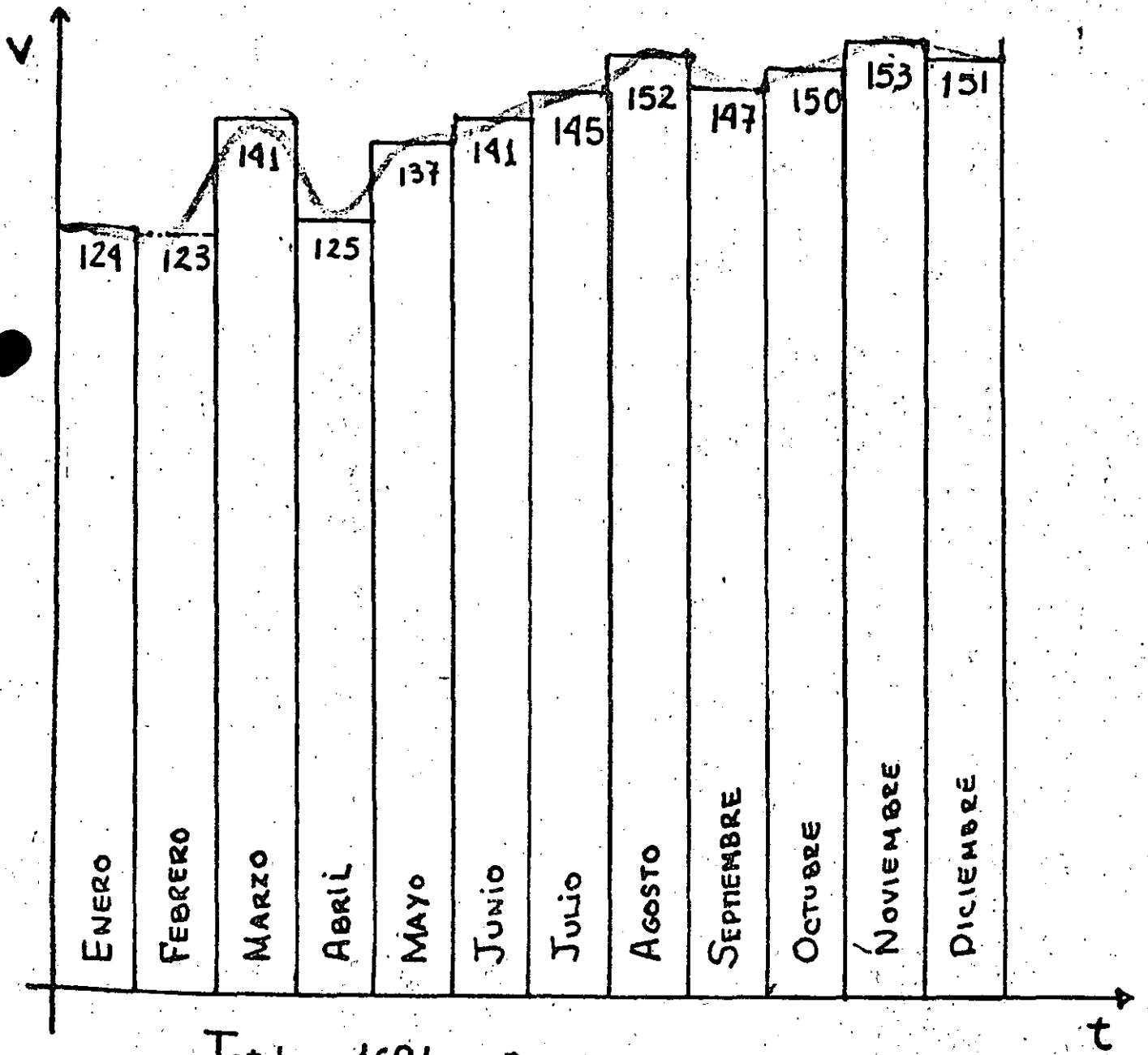
4/0

Para no complicar mucho el entendimiento de éste estudio, solo se presentan, la corrida de los métodos que tuvierón un error inferior al±20%.

Como se observa, el promedio ponderado exponencialmente ajustado con alfa=0.3 es el método que mejor se apega a las variaciones de muestras ventas.
Teniendo un error promedio de ± 4.5% a continuación se muestra la comparación de los valores dados con éste método, y las ventas reales de cada mes.

41

1983



Total → 1691 motores

Promedio → 141 motores / mes

5.54 motores / DIARIOS

42

Grafica comparativa
entre
la "curva de ventas"
y la
"curva de pronosticos"

al 82%

MES	VENTAS	PRONOSTICOS
ENE	108	108
FEB	91	99
MAR	123	114
ABR	85	96
MAY	106	110
JUN	101	103
JUL	100	101
AGO	103	104
SEPT	82	90
OCT	112	101
NOV	112	103
DIC	120	117
ENE	94	106
FEB	92	98
MAR	109	103
ABR	94	100
MAY	101	100
JUN	104	101
JUL	88	93
AGO	89	89
SEP	92	99
OCT	112	101
NOV	104	108
DIC	120	114
ENE	126	123
FEB	106	116
MAR	134	122
ABR	100	111
MAY	88	97
JUN	98	96
JUL	112	103
AGO	119	115
SEP	116	113
OCT	96	105
NOV	112	109
DIC	148	126
ENE	167	124
FEB	193	125
MAR		141
ABR		125
MAY		131
JUN		120
JUL		146
AGO		152
SEP		111
OCT		150
NOV		151
DIC		131

```
10 PRINT "READING DATA "; FOR I = 1 TO 5: PRINT ".": NEXT : PRINT
15 NN = 36
30 FOR I = 1 TO NN
35 READ ANIO(I)
40 NEXT
45 FOR J = 1 TO NN: X2 = X2 + J ^ 2: NEXT : FOR J = 1 TO NN: X = X + J: NEXT : FOR J =
   1 TO NN: LY = LY + 1 / LOG (10) * LOG (ANIO(J)): NEXT
46 FOR J = 1 TO NN: XLY = XLY + J * (1 / LOG (10) * LOG (ANIO(J))): NEXT : FOR J =
   1 TO NN: LX = LX + (1 / LOG (10) * LOG (J)): NEXT
47 AE = ((LY * X2) - (X * XLY)) / ((NN * X2) - (X ^ 2)): BE = ((NN * XLY) - (X * LY)) /
   ((NN * X2) - (X ^ 2)): AE = 10 ^ AE: BE = 10 ^ BE
48 FOR J = 1 TO NN: B(J) = AE * BE ^ J: NEXT
50 FOR I = 1 TO 36: ERR(I) = ANI(I) - B(I): NEXT : FOR I = 13 TO 24: B(I) = B(I) + ER
   R(I - 12): NEXT : FOR I = 1 TO 36: ERR(I) = ANIO(I) - B(I): NEXT : FOR I = 25 TO
   36: B(I) = B(I) + (ERR(I - 12) + ERR(I - 24)) / 2: NEXT
51 FOR I = 1 TO 36: PRINT ANIO(I), B(I), ABS (ANIO(I) - B(I)) / ANIO(I): NEXT
890 REM ********************************************************************
    *******
895 REM DATA REM109,119,110,122,130
900 DATA 850,750,650,550,570,620,670,760,930,1630,930,1000,1250,920,1300,1,2
    250,1450,1250,1390,1270,1520,1570,1760,2050,1150,1450,1750,1540,1340,1960,1010,
    1470,1700,1250,1830
1000 DATA 100,0.000010,66,69,107,287,197,272,23,35,18,337,833,3,2,15,92,244,128,2
    49,244,50,25,376,500,11,127,40,13,97,,5,87,101,471,903,818
```

```
10 DIM ANIO(50),PRON(37),ERR(37),EST(13),PEST(37),EAR(37)
20 CLS
30 REM ***********************************************************
   ***********************
40 PRINT "METODO PROMEDIO PONDERADO EXPONENCIALMENTE SIMPLE , ALFA=.3": PRINT ".": PRINT
   ".": PRINT "."
50 REM ***********************************************************
   ***********************
60 FOR I = 1 TO 36
70 READ ANIO(I)
80 NEXT
90 RESTORE
100 FOR I = 13 TO 24
110 ANIO(I) = (ANIO(I - 1) - ANIO(I - 2)) * .3 + ANIO(I - 2)
120 NEXT
130 FOR I = 13 TO 24:PR(I) = ANIO(I): NEXT
140 FOR I = 1 TO 36
150 READ ANIO(I)
160 NEXT
170 FOR I = 13 TO 24: PRINT ANIO(I),PR(I),(ANIO(I) - PR(I)) / ANIO(I): NEXT
900 REM DATA 850,750,650,520,590,620,670,760,930,1650,630,1000,1250,920,1300,10
    0,2250,1450,1250,1280,12
    10,1520,1570,1760,2050,1150,1450,1750,1940,1940,1950,15
    10,1470,1700,1250,183
1000 DATA 100,0.66,39,107,257,197,272,20,35,18,387,328,3,2,15,32,244,128,249,244,9
    0,25,374,500,11,127,40,13,97,75,87,101,471,363,818

]
]???
```

```
]LIST

 1  DIM ANIO(50),PREN(50),ERR(50),EST(40)
 2  HOME
 5  REM  ********************************************************************
      ********************************
10  PRINT "METODO PROMEDIO MOVIL SIMPLE DE 12 CON FACTORES ESTACIONALES"
11  REM  ********************************************************************
      ********************************
20  FOR I = 1 TO 36
30  READ ANIO(I)
40  NEXT
45  PRI = 13:FIN = 24
46  FOR I = 1 TO 12:SUM = SUM + ANIO(I): NEXT :::::: FOR I = 1 TO 12:EST(I) =ANIO(I
    ) / SUM: NEXT
50  FOR I = PRI TO FIN
60  ANIO(I) = (ANIO(I - 1) + ANIO(I - 2) + ANIO(I - 3) + ANIO(I - 4)) / 4
70  NEXT
80  FOR I = FIN + 1 TO PRI STEP  - 1:PREN(I) = ANIO(I - 1):ST = ST + PREN(I): NEXT
90  FOR I = PRI TO FIN:PREN(I) = ST * EST(I - 12): NEXT
95  RESTORE
98  FOR I = 1 TO 36
100  READ ANIO(I): NEXT
110  FOR I = PRI TO FIN
120 ERR(I) =  ABS ((ANIO(I) - PREN(I)) / ANIO(I))
130  NEXT
135 HOME
140  FOR I = PRI TO FIN
150  PRINT ANIO(I),PREN(I),ERR(I)
160  NEXT
899  REM  DATOS DEL LIBRO:
900  REM  DATA  350,750,650,530,590,620,670,760,930,1630,630,1000,1250,920,1300,102
     0,2250,1450,1250,1280,1270,1520,1570,1760,2050,1150,1450,1750,1840,1940,1950,13
     10,1470,1700,1250,1830
959  REM  DATOS MAESTROS:
1000  DATA  100.0,55,69.107,267,197,272,20,35,18,337,328,3,2,15,92,244,123,245,244,9
     0,25,376,500,11,127,40,13,97,75,87,101,471,308,818

]
```

```
LIST

1  DIM ANIO(50),PRE(50),ERR(50)
2  HOME
3  REM ************************************************************
   **********
10 PRINT "METODO PROMEDIO MOVIL SIMPLE AT.SIN ESTACIONALIDAD": FOR I = 1 TO 5: PRINT
   .": NEXT
11 REM <*********************************************************
   ***********
20 FOR I = 1 TO 36
30 READ ANIO(I)
40 NEXT
45 PRI = 5:FIN = 36
50 FOR I = PRI TO FIN
60 PRE(I) = (ANIO(I - 1) + ANIO(I - 2) + ANIO(I - 3) + ANIO(I - 4)) / 4
70 NEXT
80 REM FOR I = FIN + 1 TO PRI STEP -          1:PREN(I) = ANIO(I - 1): NEXT
85 RESTORE
90 FOR I = 1 TO 36
100 READ ANIO(I): NEXT
110 FOR I = PRI TO FIN
120 ERR(I) =  ABS ((ANIO(I) - PREN(I)) / ANIO(I))
130 NEXT
135 HOME
140 FOR I = PRI TO FIN
150 PRINT ANIO(I),PREN(I),ERR(I)
160 NEXT
899 REM DATOS DEL LIBRO:
900 REM DATA  850,750,650,520,590,620,670,760,930,1030,830,1090,1250,920,1300,102
    0,2250,1450,1250,1280,1270,1520,1570,1760,2030,1150,1450,1750,1840,1940,1950,13
    10,1470,1700,1250,1830
999 REM DATOS NUESTROS:
1000 DATA  100,0,66,69,107,267,197,272,20,35,18,237,828,3,2,15,92,244,123,249,244,9
    0,25,376,500,11,127,40,13,97,75,87,101,471,308,818

]
```

Promedio movil simple s/est 4 terminos

```
[illegible] DIM ANIO(50),PRENOSTICO(50),EST(15),PEST(37),ERR(37)
2 HOME
5 REM ********************************************************************
    ************************************
10 REM PRONOSTICO PROM. PONDERADO EXPONENCIALMENTE ALFA=.3
11 REM ********************************************************************
    ************************************
20 FOR I = 1 TO 37
30 READ ANIO(I)
40 NEXT
42 PRENOSTICO(1) = ANIO(1)
45 FOR I = 2 TO 37:PRENOSTICO(I) = PRENOSTICO(I - 1) + 0.3 * (ANIO(I) - PRENOSTICO(
    I - 1)): NEXT
46 FOR I = 37 TO 2 STEP - 1:PRENOSTICO(I) = PRENOSTICO(I - 1): NEXT :PRE(1) = 0
47 FOR I = 1 TO 37:ERR(I) = PRENOSTICO(I) - ANIO(I): NEXT :ERR(1) = 0
48 FOR I = 1 TO 36:SUMA = SUMA + ANIO(I): NEXT
49 FOR I = 1 TO 12:S1 = S1 + ANIO(I):S2 = S2 + ANIO(I + 12):S3 = S3 + ANIO(I + 24):
    NEXT
50 FOR I = 1 TO 12:EST(I) = (ANIO(I) + ANIO(I + 12) + ANIO(I + 24)) / SUMA: NEXT
55 FOR I = 1 TO 12:PEST(I) = EST(I) * S1:PEST(I + 12) = EST(I) * S2:PEST(I + 24) =
    EST(I) * S3: NEXT
70 FOR I = 1 TO 37: PRINT I;: HTAB (4): PRINT ANIO(I);: HTAB (10): PRINT PRENOSTICO
    (I);: HTAB (23): PRINT ERR(I);: HTAB (28): PRINT " ";PEST(I): NEXT
500 DATA 850,750,650,520,5[illegible]0,620,670,760,7[illegible]0,1[illegible]0,6[illegible]0,1000,1250,920,1[illegible]0,1020,225
    0,1450,1250,1260,1270,1520,1570,1760,2050,1150,1450,1750,1640,1940,195[illegible],1[illegible],14
    70,1700,125[illegible],1830
1000 DATA 100,[illegible],66,69,107,267,1[illegible]7,[illegible]72,21,35,18,327,328,3,2,15,92,244,128,24[illegible],244,9
    0,25,376,5[illegible],11,127,40,13,3[illegible],75,87,101,471,30[illegible],81[illegible]
```

Promedio ponderado exponencialmente $\alpha = 0.3$

```
' [illegible] ANIO(40),SIMPLE(40),DOBLE(40),PESTA(40),PM(40)
  [illegible]
' PRINT "[illegible]
  [illegible]
' PRINT "[illegible]
  [illegible]
  FOR I = 1 TO 36
  READ ANIO(I)
  NEXT
  FOR I = 1 TO 35:SIMPLE(I + 1) = (ANIO(I) + ANIO(I + 1)) / 2: NEXT
  SIMPLE(1) = ANIO(1)
  FOR I = 1 TO 35:DOBLE(I + 1) = (SIMPLE(I) + SIMPLE(I + 1)) / 2
  NEXT
  FOR I = 1 TO 36
  [illegible] = ANIO(I) - DOBLE(I)
  NEXT
  FOR I = 1 TO 36:PM(I) = (PESTA(I) * 2) + PESTA(I + 1) + P3(I)
  NEXT
  REM IMPRESION
  FOR I = 1 TO 36
  [illegible] I;: MID$ (A$, 4): PRINT ANIO(I);: MID$ (A$, 13): PRINT P3(I);: MID$ (A$, 20): PRINT P
  D(I);: MID$ (A$, 29): PRINT PM(I)
  NEXT
  DATA 55,,760,550,520,590,620,670,760,930,1630,630,1[illegible],720,1[illegible]
  ,1450,1250,1280,1270,1520,1570,17[illegible],2150,1150,1450,1[illegible],1840,1940,1[illegible],147
  0,17[illegible],1560,1530
  DATA 100,0,66,49,107,267,197,272,[illegible],35,18,347,829,3,2,15,93,344,122,241,244,[illegible]
  0,25,376,503,11,127,46,15,97,75,87,111,471,[illegible]
```

Promedio movil C/Ajuste
2 términos

```
100
! DIM ANIO(50),PSIMPLE(50),PDOBLE(50),RESTA(50),PM(50)
2 HOME
5 REM ************************************************************
    ***********************************
11 REM ************************************************************
    ***********************************
20 FOR I = 1 TO 36
30 READ ANIO(I)
40 NEXT
45 FOR I = 1 TO 33:PSIMPLE(I + 3) = (ANIO(I) + ANIO(I + 1) + ANIO(I + 2) + ANIO(I +
    3)) / 4: NEXT
60 FOR I = 1 TO 33
70 PDOBLE(I + 3) = (PSIMPLE(I) + PS(I + 1) + PS(I + 2) + PS(I + 3)) / 4
80 NEXT
90 FOR I = 1 TO 36
100 RESTA(I) = PSIMPLE(I) - PDOBLE(I)
110 NEXT
120 FOR I = 1 TO 34:PM(I) = (RESTA(I) * 2 / 3) + RESTA(I) + PS(I)
125 NEXT
130 REM ************************************************************
    ***********************************
140 FOR I = 36 TO 48
150 ANIO(I + 1) = PM(I)
160 PSIMPLE(I + 1) = (ANIO(I) + ANIO(I - 1) + ANIO(I - 2) + ANIO(I + 1)) / 4
165 PDOBLE(I + 1) = (PS(I) + PS(I - 1) + PS(I - 2) + PS(I + 1)) / 4
170 RESTA(I + 1) = PS(I + 1) - PS(I + 1)
175 PM(I + 1) = (RESTA(I + 1) * 2 / 3) + RESTA(I + 1) + PS(I + 1)
180 NEXT
400 REM ************************************************************
    ***********************************
500 REM  IMPRESION
510 FOR I = 1 TO 48
520 PRINT I;: HTAB (4): PRINT ANIO(I);: HTAB (13): PRINT PS(I);: HTAB (20): PRINT P
    D(I);: HTAB (29): PRINT PM(I)
521 IF I = 36 THEN  PRINT : PRINT : PRINT : PRINT
530 NEXT
900 DATA  650,750,650,520,590,620,670,760,930,1630,630,1000,1250,920,1300,1020,1050
    ,1450,1250,1280,1270,1520,1570,1760,2050,1150,1450,1750,1840,1940,1950,1310,147
    0,1700,1250,1860
1000 DATA  100,0,66,69,107,267,197,272,20,35,13,337,928,3,2,15,92,244,128,249,244,9
    0,25,376,500,11,127,40,13,97,75,97,101,471,308,819

]
]"
```

Promedio movil c/ajuste 4 termino

```
10 DIM ANIO(50),PS(50),PD(50),PA(50)
20 CLS
30 REM ************************************************************
   ************************************
40 REM    PROGRAMA PARA CALCULAR PROMEDIOS EXPONENCIALES AJUSTADOS
50 REM ************************************************************
   ************************************
60 FOR I = 1 TO 36
70 READ ANIO(I)
80 NEXT
90 PS(1) = ANIO(1):PD(1) = PS(1):PA(1) = ANIO(1)
100 FOR N = 2 TO 36:PS(N) = (ANIO(N) - PS(N - 1)) * .3 + PS(N - 1): NEXT
110 FOR N = 2 TO 36:PD(N) = (PS(N) - PD(N - 1)) * .3 + PD(N - 1): NEXT
120 FOR N = 2 TO 36:PA(N) = (PS(N) - PD(N)) * .429 + (PS(N) - PD(N)) + PS(N): NEXT
130 REM ************************************************************
   ************************************
140 FOR N = 37 TO 48:ANIO(N) = PA(N - 1)
150 PS(N) = (ANIO(N) - PS(N - 1)) * .3 + PS(N - 1)
160 PD(N) = (PS(N) - PD(N - 1)) * .3 + PD(N - 1)
170 PA(N) = (PS(N) - PD(N)) * .429 + (PS(N) - PD(N)) + PS(N)
180 NEXT
190 FOR I = 1 TO 48: PRINT I;: HTAB (6): PRINT ANIO(I);: HTAB (14): PRINT PS(I);: HTAB
    (28): PRINT PD(I);: HTAB (40): PRINT PA(I): NEXT
191 PRINT : HOME
200 FOR I = 1 TO 48: PRINT PA(I),: NEXT
210 REM ************************************************************
   ************
220 DATA  850,750,650,520,580,620,670,760,930,1630,620,1360,1250,920,1300,1060,225
    0,1450,1250,1580,1070,1520,1570,1760,2050,1150,1450,1750,1940,1840,1950,1310,1
    71,1750,1250,1991
230 DATA  100,0,50,60,107,267,197,273,20,35,18,307,828,0,2,15,92,244,126,249,244,5
    0,25,376,500,11,127,42,13,97,79,67,101,471,205,616
```

Promedio ponderado exponencialmente ajustado

alfa = 0.3

Promedio movil ajustado s/estacionalidad 4. terminos


```
1 '
2 '
3 DIM ANIO(36),PSIMPLE(50),PDOBLE(50),RESTA(50),PM(50)
4 '
5 REM  PARA PODER ESTE METODO PARA EL SIGUIENTE A TRES AÑOS, SOLO SE CAMBIAN LOS
     LIMITES DEL FOR DE LA IMPRESION Y DEL FOR DE LA LINEA 140 ASI COMO EL DE DATA
6 '
7 '
8 '
9 '
10 REM
11 '
12 FOR I = 1 TO 36
13 READ ANIO(I)
14 NEXT
15 FOR I = 1 TO 33:PSIMPLE(I + 2) = (ANIO(I) + ANIO(I + 1) + ANIO(I + 2) + ANIO(I +
     3)) / 4: NEXT
17 FOR I = 1 TO 33
18 PDOBLE(I + 3) = (PSIMPLE(I) + PSIMPLE(I + 1) + PSIMPLE(I + 2) + PSIMPLE(I + 3)) / 4
19 NEXT
20 FOR I = 1 TO 36
100 RESTA(I) = PSIMPLE(I) - PDOBLE(I)
110 NEXT
120 FOR I = 1 TO 36:PM(I) = (RESTA(I) * 2 / 3 + RESTA(I - 1) / 3 + PS(I)
125 NEXT
130 REM
135 '
136 '
137 TN = RESTA(34) * 2 / 3
140 FOR I = 34 TO 36
145 PM(I + 1) = PM(I) + TN
150 NEXT
400 REM
410 '
500 REM  IMPRESION
510 FOR I = 25 TO 36
520 PRINT ANIO(I),PM(I - 1), ABS ((ANIO(I) - PM(I - 1)) / ANIO(I))
521 IF I = 36 THEN  PRINT : PRINT : PRINT : PRINT
530 NEXT
900 DATA  950,750,150,500,570,623,370,760,920,1630,930,1000,1250,720,1310,1020,205
    0,1450,1250,1200,1270,1520,1570,1760,2050,1150,1450,1750,1640,1940,1950,1310,14
    70,1700,1250,1300
1000 DATA  100,0,66.67,167,267,137,273,20,35,10,337,333,3.3,15,92,244,133,243.244.9
    0.25.376,500,11,127,40,13,57,75,67,101,471,305,813
1010 FOR I = 13 TO 24
'
```

52

```
1  DIM ANIO(50),PR(50),EST(50),ES(150,50)
2  '
3  '
4  '
5  PRINT "SELECT DISTRIBUTION FOR OUTPUT DATA(ENTER DIGIT) : ALPA=1 CONSTANT
   LINEAL 3 PRINT 4 ESTOP "
6  REM ********************************************************************
   ********************************************
7  FOR I = 1 TO 36
8  READ ANIO(I)
9  NEXT
10 RESTORE
11 FOR I = 1 TO 12:SUM = SUM + ANIO(I): NEXT : FOR I = 1 TO 12:EST(I) = ANIO(I) / S
   UM: NEXT
12 FOR I = 13 TO 24
13 ANIO(I) = (ANIO(I - 1) - ANIO(I - 2)) * .2 + ANIO(I - 2)
14 NEXT
15 SUM = 0:FOR I = 13 TO 24:PR(I) = ANIO(I):SUM = SUM + PR(I): NEXT
16 FOR I = 13 TO 24:PR(I) = EST(I - 12) * SUM: NEXT
17 FOR I = 1 TO 36
18 READ ANIO(I)
19 NEXT
20 FOR I = 13 TO 24: PRINT ANIO(I),PR(I),(ANIO(I) - PR(I)) / ANIO(I): NEXT
100 REM, DATA    850,750,650,520,580,630,670,760,930,1630,680,1010,1250,721,1201,1
    120,1150,1450,1150,1200,1270,1521,1570,1730,2150,1150,1450,1750,1040,1240,1990,
    1310,1470,1710,1250,1910
110 DATA  100,0,46,61,137,237,177,273,26,35,10,337,319,3,3,15,92,244,128,249,244,3
    0,25,376,560,11,127,40,13,77,75,87,101,471,308,813

120
130
140
150
```

```
1  DIM ANIO(50),B(40):
2  CLS
3  REM ********************
     ********************
     ********************
     ********************
     ******
4  PRINT "METODO SUMA ERRORES
   AL CUADRADO" : FOR
   I = 1 TO 5: PRINT ".": NEXT

11 REM ********************
     ********************
     ********************
     ********************
     ******
15 NN = 36
20 FOR I = 1 TO NN
30 READ ANIO(I)
40 NEXT
45 FOR J = 1 TO NN: X2 = X2 + J *
   J: NEXT : FOR J = 1 TO NN: X =
   X + J: NEXT : FOR J = 1 TO N
   N: LY = LY + 1 / LOG (10) *
   LOG (ANIO(J)): NEXT
46 FOR J = 1 TO NN: XLY = XLY + J
   * (1 / LOG (10) * LOG (AN
   IO(J))): NEXT : FOR J = 1 TO
   NN: LX = LX + (1 / LOG (10) *
   LOG (ANIO(J))): NEXT
47 AE = ((X2 * LY) - (X * XLY)) /
   ((NN * X2) - (X * X)): BE = (
   (NN * XLY) - (X * LY)) / ((NN
   * X2) - (X * X)): AE = 10 ^
   AE: BE = 10 ^ BE
49 FOR J = 1 TO NN: B(J) = AE * B
   E ^ J: PRINT ANIO(J), B(J), ABS
   ((ANIO(J) - B(J)) / ANIO(J))
   : NEXT : END
999 REM ********************
     ********************
     ********************
     **
999 REM DATA REM 108,119,110,12
   2,130
999 REM DATA  850,750,650,520,
   550,620,670,760,930,1630,830
   ,1000,1250,920,1300,1020,225
   0,1450,1250,1280,1270,1520,1
   570,1760,2050,1150,1450,1750
   ,1540,1940,1950,1310,1470,17
   30,1950,1930
999 DATA  100,0.000010,66.69,1
   [illegible]137,272,30,35,13.33,
   [illegible]8.3,2.15,92,244,128,249,24
   4,50,25.37,500,11,197,40,13
   [illegible]5,8,16
```

```
10  DIM ANIO(50),PS(50),PD(50),PA(50),EST(13)
20  '
30  REM ***********************************************
    ***********************************
40  PRINT "METODO NUMERICO INTERFAZ EXPONENCIAL AJUSTADO ,ALFA=,3 OPERACION MIXTA"
    : PRINT ".": PRINT ".": PRINT "."
50  REM ***********************************************
    ***********************************
60  FOR I = 1 TO 36
70  READ ANIO(I)
80  '
90  PS(1) = ANIO(1):PD(1) = ANIO(1):PA(1) = ANIO(1)
95  FOR N = 2 TO 36:PS(N) = (ANIO(N) - PS(N - 1)) * .3 + PS(N - 1): NEXT
97  FOR N = 2 TO 36:PD(N) = (PS(N) - PD(N - 1)) * .3 + PD(N - 1): NEXT
98  FOR N = 2 TO 36:PA(N) = (PS(N) - PD(N)) * .429 + (PS(N) - PD(N)) + PS(N): NEXT
74  FOR I = 36 TO 1 STEP - 1:PA(I + 1) = PA(I): NEXT
75  REM ***********************************************
    ***********************************
80  FOR N = 13 TO 24:ANIO(N) = PA(N - 1)
81  PS(N) = (ANIO(N) - PS(N - 1)) * .3 + PS(N - 1)
82  PD(N) = (PS(N) - PD(N - 1)) * .3 + PD(N - 1)
83  PA(N) = (PS(N) - PD(N)) * .429 + (PS(N) - PD(N)) + PS(N)
85  NEXT
87  RESTORE : FOR I = 1 TO 36: READ ANIO(I): NEXT : FOR I = 1 TO 12:S1 = S1 + ANIO(I
    ): NEXT : FOR I = 1 TO 12:EST(I) = ANIO(I) / S1: NEXT : FOR I = 13 TO 24:ST = S
    T + PA(I): NEXT : FOR I = 13 TO 24::PA(I) = ST * EST(I - 12): NEXT
90  FOR I = 13 TO 24: PRINT ANIO(I),PA(I), AES ((ANIO(I) - PA(I)) / ANIO(I)): NEXT
95  END
100 FOR I = 1 TO 48: PRINT I;: HTAB (6): PRINT ANIO(I);: HTAB (14): PRINT PS(I);: HTAB
    (25): PRINT PD(I);: HTAB (40): PRINT PA(I): NEXT
101 PRINT : HOME
110 FOR I = 1 TO 48: PRINT PA(I),: NEXT
890 REM ***********************************************
    ***********************************
900 REM DATA   350,750,650,520,590,620,670,760,860,1030,830,1060,1250,720,1300,1120
    ,2250,1450,1250,1260,1270,1520,1570,1760,2050,1150,1450,1750,1040,1040,1250,121
    0,1470,1790,1250,1830
1000 DATA  100,0.66,69,107,267,197,272,20,35,19,337,828,3,2,15,92,244,126,249,244,8
    0,25,376,500,11,127,40,13,97,75,87,101,471,308,818

!
]?**
```

$\alpha = 0.3$

5. CONCLUSIONES Y SUGERENCIAS *

* Gráfica de Eficiencia

CONCLUSIONES Y RECOMENDACIONES:

Cabe señalar, que la idea de determinar el pronóstico de ventas para 1983, cumple su cometido, si éste logra representar fielmente su comportamiento futuro a fin de que se tenga un adecuado medio de apoyo para:

1. La planeación de abastecimientos que representen tan solo la cantidad necesaria, para que no se pare el área operativa. Evitando inversiones muertas, al darle mayor rotación a los inventarios; puesto que en relación a la situación económica general, no es conveniente invertir en inventarios que no tengan una pronta realización.

2. El efecto del comportamiento de las ventas de 1982* sobre los datos totales del estudio; impiden ver de modo realista el comportamiento de ventas para 1983 puesto que se esperan situaciones muy variables para este año pronosticado. Por lo que se recomienda, simplemente ajustarse a la tendencia arítmetica de los últimos 3 meses para conocer las ventas del siguiente mes.

3. Puesto que las ventas no se comportan con ningún tipo de estacionalidad, mensual ó por temporadas, se sugiere que el pronóstico obtenido sea utilizado para la programación de trabajos de mantenimiento preventivo así como para la programación de vacaciones del personal, a fin de que en meses donde se espera mayor trabajo, se cuenten con los elementos adecuados para un buen funcionamiento operativo.

* Variabilidad de ventas debido a la devaluación del peso .

4. Estos resultados, son la pauta para poder desarrollar nuevos mecanismos administrativos en cuanto a control o incremento de la eficiencia operativa actual, puesto que la producción actual; no corresponde necesariamente a la capacidad de producción instalada.

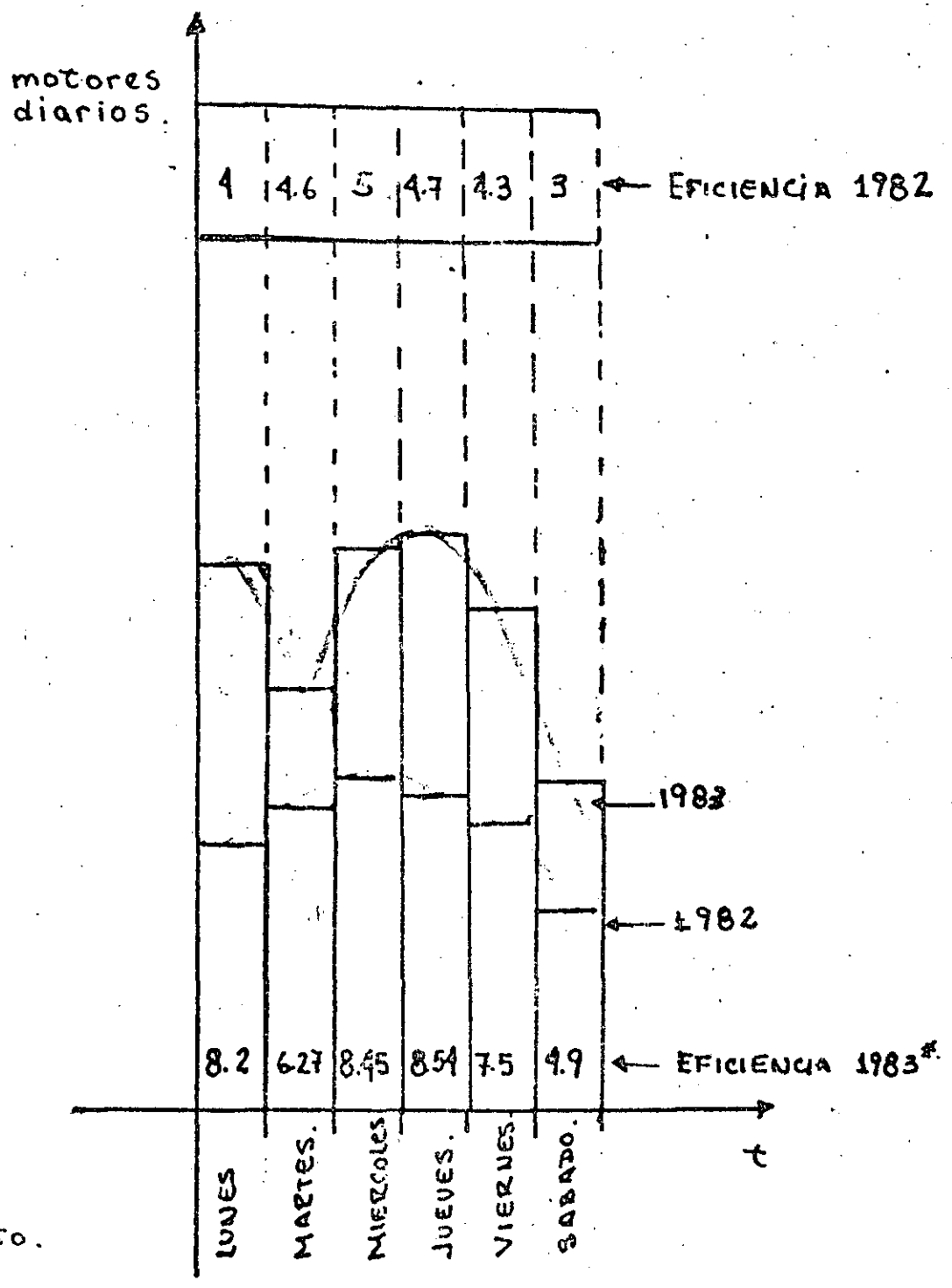
5. En las bajas en ventas pronosticadas (según lo muestran las gráficas) puede implementarse un plan intensivo - de capacitación, a fin de que cuando no exista el suficiente trabajo, los operadores o empleados puedan recibir cursos de capacitación ó adiestramiento, que según la empresa se marquen como necesarios.

6. El estudio permite ser ajustado más sofisticadamente- de tal forma que permita visualizar la campana de trabajo semanal, la que representa los niveles diarios - de eficiencia. La aplicación practica de esto; implica el tener los elementos necesarios para evaluar las conveniencias de otorgar un "puente" o la suspensión- de labores en un día de un determinado mes. Puesto - que se cuentan con los elementos necesarios (estadísticas) para poder evaluarlos y controlarlos. (ver gráfica anexa).

Por todo lo anterior, se solicita la autorización para la realización de los estudios a que dan origen el pronóstico de ventas realizado.

-*-

Campana Semanal de eficiencia



Y combustible.

LENGUAJE DE PROGRAMACION BASIC II

INTRODUCCION A CASOS DE ESTUDIO DE TIPO ESPECIAL

NOVIEMBRE DE 1985

INTRODUCCION.

Durante la utilización normal del editor CANDE de las computadoras BURROUGHS con que cuenta la U.N.A.M., muy a menudo el tiempo de respuesta de la computadora aumenta considerablemente debido, entre otros factores, a la gran carga de trabajo que tiene que soportar, por lo que las labores que deben desempeñar los usuarios se retrasan continuamente; el presente trabajo pretende dar una posible solución a este problema.

Las labores que efectúa cualquier usuario de la BURROUGHS a través de una terminal de CANDE se pueden dividir en dos partes: la primera consiste en ejecutar tareas, como sería una compilación, la ejecución de un programa, mandar a procesar un JOB, etc; y la segunda sería la edición de un archivo utilizando los múltiples comandos con que cuenta CANDE para ello.

Considerando la división de labores anterior, se observa que no es posible tratar de agilizar el tiempo de respuesta de la BURROUGHS disminuyendo el número de tareas en ejecución, ya que precisamente es ésa la labor que se desea que realice la computadora; pero reflexionando en la segunda parte se puede observar que, hasta cierto punto, es un uso inadecuado de recursos el estar utilizando un valioso tiempo de proceso de la BURROUGHS en entrar a una sesión de CANDE simplemente para editar un archivo, labor que puede efectuar eficientemente casi cualquier microcomputadora que existe en el mercado; tal vez no con la facilidad con que lo hace CANDE pero sí seguramente a un costo muchísimo menor y también, dadas las cargas de trabajo usuales en la BURROUGHS, con una mayor rapidez.

Si además de tomar en cuenta lo anterior se observa que muy a menudo un gran porcentaje del tiempo de una sesión de CANDE consiste en ejecución de comandos para la edición de un archivo, o bien de tiempo "muerto" durante el cual el usuario no teclea nada, entonces es evidente que una solución adecuada al problema antes mencionado sería efectuar las ediciones de archivos en una computadora separada de la BURROUGHS y, una vez editado el archivo de que se trate, transmitirlo a la BURROUGHS para su consiguiente proceso.

Esto agilizaría en gran medida la labor del usuario, ya que no se vería afectado por el tiempo de respuesta de la BURROUGHS durante la edición de sus archivos que es la labor que normalmente le retiene durante más tiempo en la terminal de la computadora, y tendría la ventaja adicional de que al no estar ese usuario utilizando a CANDE para sus ediciones, liberaría a la BURROUGHS de una parte de su carga de trabajo.

Para implementar adecuadamente una solución de este tipo, primero se tendría que efectuar un estudio relativo al modelo de computadora que se podría adquirir para este trabajo revisando que fuera factible interconectarla con la BURROUGHS para transmisión de datos y cuidando aspectos como precio, software disponible, etc. y, una vez adquirido el equipo, desarrollar el software necesario para que el enlace funcionara en forma adecuada.

Sin embargo, cuando se comenzó a desarrollar este trabajo ya estaba resuelta la primera parte de este problema, ya que la U.N.A.M. cuenta actualmente con un gran número de microcomputadoras Cromemco las cuales se pueden conectar a la BURROUGHS para transmisión y recepción de datos sin ningún problema, además de contar con un editor de archivos sumamente poderoso considerando que se trata de una microcomputadora.

La mínima capacidad de almacenamiento en disco con que cuenta el equipo Cromemco consiste en una unidad de discos flexibles de 5 1/4 pulgadas la cual puede almacenar un archivo de hasta 386K bytes en un solo disco flexible, lo cual implica que se podría transferir a la Cromemco un archivo de la BURROUGHS que ocupara 2195 segmentos de disco o más, ya que la microcomputadora Cromemco no requiere que se almacenen los espacios en blanco después del último caracter significativo de cada registro, como ocurre en la BURROUGHS, por lo que la mayoría de los archivos que normalmente se editan a través de CANDE se podrían editar en la Cromemco.

EL PROGRAMA DE COMUNICACION CANDE-CROMEMCO.

A fin de obtener una solución como la anteriormente descrita para su propio beneficio, el autor desarrolló el programa de comunicación para la computadora Cromemco antes mencionado cuidando que su utilización para la edición local de archivos de la BURROUGHS fuera lo más sencilla y poderosa posible; una breve descripción de la operación de este programa se presenta en seguida.

Una vez activo, el programa de comunicación simula el funcionamiento de una terminal, por lo que la computadora Cromemco opera en forma idéntica a la de cualquier terminal de CANDE, pero además el programa cuenta con varios comandos y opciones propios como son los siguientes:

Por medio de una opción del programa puede conseguirse que todo aquello que reciba la computadora Cromemco de la BURROUGHS sea almacenado en un archivo en el disco flexible de la Cromemco; esto tiene como principal objeto el poder transferir un archivo de la BURROUGHS a la Cromemco para su posterior edición en forma local en la computadora Cromemco; sin embargo, no es éste el único uso de esta opción.

Al poder almacenar en un archivo de la Cromemco todo lo que se recibe de la BURROUGHS es posible, por ejemplo, efectuar una compilación y almacenar los textos de los errores de la misma en el archivo de disco para analizarlos posteriormente sin tener que estar revisando la pantalla en ese momento e interfiriendo con la compilación; o bien, mandar a ejecución cualquier programa de utilería de la BURROUGHS y almacenar en el archivo todos los resultados que muestre en la pantalla para su posterior impresión.

La opción antes mencionada puede permitir que, junto con los caracteres que se reciben de la BURROUGHS, también se almacene en el archivo todo lo que transmite la Cromemco, con lo cual se puede obtener en un archivo en el disco de la Cromemco un ejemplo de una sesión de CANDE o de una ejecución de un programa mostrando las entradas y salidas del mismo, por lo que la elaboración de manuales de operación de programas de BURROUGHS podría hacerse en forma sumamente sencilla utilizando el editor de texto de la computadora Cromemco.

Por supuesto, el programa también cuenta con la posibilidad de transmitir un archivo del disco de la Cromemco a la BURROUGHS; el uso primario de esta función es el de transmitir a la BURROUGHS un archivo generado en la Cromemco, o bien, el de transmitir las modificaciones necesarias a un archivo de BURROUGHS para corregirle algunos errores, ya que se pueden desarrollar programas en la Cromemco para obtener las diferencias de un archivo antes y después de su edición a fin de transmitir a la BURROUGHS únicamente aquellas partes del archivo que sufrieron cambios durante su edición local en la computadora Cromemco.

Pero nuevamente hay otras posibilidades, como por ejemplo, utilizar los editores de texto de la Cromemco para generar archivos de texto en forma adecuada para su presentación (un ejemplo de ello es el presente artículo) y transmitirlos a la BURROUGHS para su impresión en forma masiva y rápida.

Además de los mencionados anteriormente, el programa de comunicación cuenta con otros comandos que facilitan su operación; es importante mencionar que todos los comandos y opciones propios del programa se accesan a través de una forma sencilla y son rápidos de aprender, además de que existe un comando de ayuda que despliega una descripción de todos los comandos existentes.

Para que una microcomputadora Cromemco pueda ser conectada a la BURROUGHS a fin de utilizar este programa, lo único necesario adicional en lo referente a hardware es una tarjeta Cromemco de TU-ART la cual proporcionará el puerto adicional de entrada y salida necesario para la transmisión y recepción de caracteres de la BURROUGHS. La mencionada tarjeta puede manejar velocidades de transmisión de 110, 150, 300, 1200, 2400, 4800 y 9600 baud, por lo que se podrá conectar a ella una línea de comunicación de la BURROUGHS que transmita a alguna de estas velocidades y que actualmente puede estar conectada a cualquier terminal común de CANDE.

El programa de comunicación CANDE-CROMEMCO está operando actualmente en una microcomputadora Cromemco System Zero conectada a la computadora BURROUGHS B6800 que se encuentra instalada en el edificio del IIMAS y ha demostrado ser eficiente y adecuado al uso para el que fué diseñado; el único problema que eventualmente se puede presentar durante la operación del programa consiste en la transmisión automática de archivos a la BURROUGHS cuando el tiempo de respuesta de la BURROUGHS es muy grande.

Como no existe ningún protocolo de comunicación en la línea de transmisión, la computadora Cromemco no puede darse cuenta si la BURROUGHS está recibiendo adecuadamente los caracteres que aquella le transmite y, si el tiempo de respuesta de la BURROUGHS es sumamente grande, podría no recibir todo lo que le mande la Cromemco; o bien, si se "cae el sistema" en la BURROUGHS la Cromemco tampoco podría darse cuenta de este hecho.

Sin embargo una solución a estos problemas es sencilla; puesto que el usuario está presente en el momento de la transmisión, revisando la pantalla de la Cromemco puede darse cuenta de que algo anda mal con la BURROUGHS y suspender el proceso para posteriormente revisar la información que recibió la BURROUGHS y continuar la transmisión a partir del punto indicado; o bien, puede utilizar la transmisión no automática de archivos, en la cual el usuario transmite una a una las líneas del archivo cada vez que él lo desea, por lo que se evitarían los problemas de transmitir caracteres a una gran velocidad cuando el tiempo de respuesta de la BURROUGHS es muy grande.

El autor de este programa, Sr. Antonio Pérez Ayala, desea informar a la Comunidad Universitaria que obsequiará copias de su programa para instalarse en el equipo Cromemco de cualquier dependencia de la U.N.A.M. que se lo solicite, junto con un manual de instrucciones para la operación del mismo; Antonio Pérez Ayala es estudiante de la carrera Ingeniería en Computación en la Facultad de Ingeniería de la U.N.A.M. y también colabora en esta Universidad en la Coordinación de la Administración Escolar, Subdirección de Diseño y Desarrollo de Nuevos Proyectos en el edificio del IIMAS, planta baja, cubículo "P"; teléfono 550-52-15 extensiones 4534, 5045 y 5046.

```
  1     PRINT\PRINT "    PROGRAMA DE LA RUTA CRITICA"
  2     PRINT\PRINT "  REALIZADO POR JORGE ONTIVEROS JUNCO"
  3     PRINT\PRINT "      COORDINACION DE PLANEACION"
  4     PRINT\PRINT "       FACULTAD DE INGENIERIA"
  5     PRINT\PRINT "         CENTRO DE CALCULO    "
  6     PRINT\PRINT "    COMPUTADORA DIGITAL VAX 11-780"
  7     PRINT\PRINT BEL
 10     OPEN "RESUL.LIS" FOR OUTPUT AS FILE #2, &
            ORGANIZATION SEQUENTIAL, RECORDSIZE 132
 11     PRINT#2,FF\FOR Z%=1 TO 20\PRINT#2\NEXT Z%
        PRINT#2, "    PROGRAMA DE LA RUTA CRITICA"
 12     PRINT#2, \PRINT#2,  "  REALIZADO POR JORGE ONTIVEROS JUNCO"
 13     PRINT#2, \PRINT#2,  "      COORDINACION DE PLANEACION"
 14     PRINT#2, \PRINT#2,  "       FACULTAD DE INGENIERIA"
 15     PRINT#2, \PRINT#2,  "         CENTRO DE CALCULO    "
 16     PRINT#2, \PRINT#2,  "    COMPUTADORA DIGITAL VAX 11-780"
 90 ON ERROR GO TO 14000
100     SLEEP 5%\REM PROGRAMA DE LA RUTA CRITICA
200     REM ADMITE CUANDO MAS 100 ACTIVIDADES CON MAXIMO 5 PRECEDENTES.
300     REM LAS ACTIVIDADES VAN NUMERADAS DE UNO A CIEN
400     REM REALIZO EL ING. JORGE ONTIVEROS JUNCO
405     !++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        !       MAPAS
        !++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
410     DECLARE INTEGER N,NI,NA,M,I,J,P, REAL COSTO,COSTO_ACUM
420     MAP(MAPA) NA$=40%,D$=5%,C$=10%, &
            A0$=3%,A1$=3%,A2$=3%,A3$=3%,A4$=3%,A5$=3%,FILL
425     MAP(MAPA) N$=0%
430     !++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ! LOS DATOS SE ACOMODAN DE LA SIGUIENTE MANERA:
        ! EN EL PRIMER REGISTRO VA EN NUMERO DE ACTIVIDADES DE LA RED
        !       DESDE LA PRIMERA COLUMNA Y DEBE VALER CUANDO MAS 100.
        !       JUSTIFICADO A LA DERECHA.
        !
        ! EN LOS SIGUIENTES  REGISTROS VAN  LAS ACTIVIDADES DE LA RED
        ! DE ACUERDO AL SIGUIENTE FORMATO:
        ! COLUMNAS  1 A 40 EN NOMBRE DE LA ACTIVIDAD
        ! COLUMNAS 41 A 45 LA DURACION, EN UNIDADES ENTERAS
        ! COLUMNAS 46 A 55 EL COSTO, EN VALOR REAL
        ! COLUMNAS 56 A 58 EL NUMERO DE ACTIVIDADES PRECEDENTES
        !       SI ESTE DATO ES MAYOR QUE CERO ENTONCES INDICAR LAS
        !       ACTIVIDADES PRECEDENTES
        ! COLUMNAS 59 A 61 PRIMERA ACTIVIDAD PRECEDENTE
        ! COLUMNAS 62 A 64 SEGUNDA
        ! COLUMNAS 65 A 67 TERCERA
        ! COLUMNAS 68 A 70 CUARTA
        ! COLUMNAS 71 A 73 QUINTA
        !++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
500     PRINT \ PRINT "RUTA CRITICA" \ PRINT
        PRINT #2,FF\PRINT#2, \PRINT#2,  "RUTA CRITICA"\PRINT #2,
503     PRINT BEL+BEL+BEL
505     LINPUT "DAME EL NOMBRE DEL ARCHIVO"; NOMBRE$
600     OPEN NOMBRE$  FOR INPUT AS FILE #1,ORGANIZATION SEQUENTIAL, &
            DEFAULTNAME ".DAT",MAP MAPA
800     GET #1%    ! N ES EL NUMERO DE ACTIVIDADES DE LA RED
                   N=VAL%(N$)
900     DIM INTEGER AU(100) !VECTOR AUXILIAR
1000    DIMENSION   A$(100) !NOMBRE DE LA ACTIVIDAD
1100    DIM INTEGER A(100,5)!MATRIZ DE ACTIVIDADES PRECEDENTES
1200    DIM REAL    C (100) !COSTO
1300    DIM INTEGER D (100) !DURACION
```

```
400     DIM INTEGER ES(100) !EARLY START  (INICIO PROXIMO)
500     DIM INTEGER EF(100) !EARLY FINISH (FIN    PROXIMO)
600     DIM INTEGER LS(100) !LATE  START  (INICIO TARDIO )
700     DIM INTEGER LA(100) !LATE  FINISH (FIN    TARDIO )
800     DIM INTEGER HL(100) !HOLGURA LIBRE
900     DIM INTEGER HO(100) !HOLGURA TOTAL
905     DIM REAL    COSTO_DIA(1000)
1000    REM                                                    6
1100    FOR I=1 TO N
1200            GET #1%
1210                    A$(I)=NA$
1220                    D(I)=VAL%(D$)
1230                    C(I)=VAL (C$)
1240                    A(I,0)=VAL%(A0$)
1250                    A(I,1)=VAL%(A1$)
1260                    A(I,2)=VAL%(A2$)
1270                    A(I,3)=VAL%(A3$)
1280                    A(I,4)=VAL%(A4$)
1290                    A(I,5)=VAL%(A5$)
1300    NEXT I
1350 SLEEP 5%
2400    PRINT     "DATOS LEIDOS","EL NUMERO DE ACTIVIDADES ES DE:";N
        PRINT#2,  "DATOS LEIDOS","EL NUMERO DE ACTIVIDADES ES DE:";N
2500    PRINT\PRINT#2,
2600    PRINT     "NODO NOMBRE                                    ";
        PRINT#2,  "NODO NOMBRE                                    ";
        PRINT#2,  "DURACION        COSTO NODOS PREDECESORES"
        PRINT     "DURACION        COSTO NODOS PREDECESORES"
2700    PRINT\PRINT#2,
2750    COSTO=0.0
2800    FOR I=1 TO N
2900            PRINT A$(I);
                PRINT USING " ###      $####,###.##   ##   ### ### ### ### ### ": &
                    D(I),C(I),A(I,0),A(I,1),A(I,2),A(I,3),A(I,4),A(I,5)
2910            PRINT#2,  A$(I);
                PRINT#2  USING " ###      $####,###.##   ##   ### ### ### ### ### ": &
                    D(I),C(I),A(I,0),A(I,1),A(I,2),A(I,3),A(I,4),A(I,5)
                COSTO=COSTO+C(I)
3000    NEXT I
3010    PRINT\PRINT USING "EL COSTO DEL PROYECTO ES DE $#######,###.##";COSTO\PRINT
3015    PRINT#2 \ PRINT#2 USING "EL COSTO DEL PROYECTO ES DE $#######,###.##": &
                COSTO \PRINT #2
3100    G=0         !BANDERA
3200    FOR I=1 TO N
3300            ES(I)=0
3400            EF(I)=D(I)
3500            HL(I)=9999
3600    NEXT I
3700    B=0         !BANDERA QUE INDICARA SI LA FASE UNO YA SE TERMINO
3800    FOR I=1 TO N
3900            IF A(I,0)=0 THEN 4400
4000                    FOR J=1 TO A(I,0)
4100                            P=A(I,J)
4200                            IF EF(P)<= ES(1) THEN 4300
4210                                    ES(I)=EF(P)
4220                                    B=1
4230                                    EF(I)=ES(I)+D(I)
4240                                    IF G<EF(I) THEN G=EF(I)
4300                    NEXT J
4400    NEXT I
4500    IF B=1 THEN 3700
```

```
      REM  FIN DE LA FASE UNO
4600  PRINT\PRINT BEL+BEL\PRINT
4700  PRINT "LA DURACION DEL PROYECTO ES DE ";G \PRINT\PRINT
4750 SLEEP 5%
4800  PRINT#2, "LA DURACION DEL PROYECTO ES DE ";G;"  DIAS"
      PRINT#2, \PRINT #2,
4900  FOR I=1 TO N
5000        LA(I)=G
5100        LS(I)=LA(I)-D(I)
5200  NEXT I
5300  B=0              ! INICIO DE LA FASE DOS
5400  FOR I=1 TO N
5500        IF A(I,0)=0 THEN 6400
5600              FOR J=1 TO A(I,0)
5700                    P=A(I,J)
5800                    IF LA(P)<=LS(I) THEN 6300
5900                          LA(P)=LS(I)
6000                          B=1
6200                          LS(P)=LA(P)-D(P)
6300              NEXT J
6400  NEXT I
6500  IF B=1 THEN 5300 ! FIN DEL CICLO DE LA FASE DOS
6600  PRINT "RUTA CRITICA"
6700  PRINT#2,FF\PRINT #2\PRINT #2, "RUTA CRITICA"
6800  PRINT "RESULTADOS" \ PRINT #2, "RESULTADOS"
6900  PRINT    " NODO  NOMBRE                                          ES   EF   LS   LF  HT   HL"
7000  PRINT#2," NODO  NOMBRE                                          ES   EF   LS   LF  HT   HL"
7100  FOR I=1 TO N
7200        HO(I)=LA(I)-EF(I)
      0     IF A(I,0)=0 THEN 7900
      0           FOR J=1 TO A(I,0)
7500                    P=A(I,J)
7600                    IF HL(I)> (ES(P)-ES(I)-D(I))THEN  &
                              HL(I)=ES(P)-ES(I)-D(I)
7700                    IF HL(I)<0 THEN HL(I)=0
7800              NEXT J
7900        IF ES(I)=LS(I) THEN HL(I)=0
8000  NEXT I
8100  FOR I=1 TO N
8200    PRINT USING "###     ",I;
        PRINT A$(I);
        PRINT USING " ###  ###  ###  ### ### ###" &
              ,ES(I);EF(I);LS(I);LA(I);HO(I);HL(I);
8300    PRINT#2  USING "###     ",I;
        PRINT#2,  A$(I);
        PRINT#2  USING " ###  ###  ###  ### ### ###" &
              ,ES(I);EF(I);LS(I);LA(I);HO(I);HL(I);
8400    IF ES(I)=LS(I) THEN PRINT " CRITICA" ELSE PRINT " "
8410    IF ES(I)=LS(I) THEN PRINT#2,  " CRITICA" ELSE PRINT#2,  " "
8450  NEXT I
8500  REM CLASIFICACION POR FECHA DE INICIO
8600  FOR I=1 TO N
8700        AU(I)=I
8800  NEXT I
8900  M=N-1! LIMITE VARIABLE DE LA CLASIFICACION
9000  B=0  ! BANDERA DE LA CLASIFICACION
9100  FOR J=1 TO M
9200        IF ES( AU(J)) <= ES( AU(J+1)) THEN 9700
9300              B=1
9400              T=AU(J)
      0           AU(J)=AU(J+1)
```

```
9600                    AU(J+1)=T
9700    NEXT J
9800    M=M-1
9900    IF B=1 THEN 9000
10000   PRINT\PRINT\PRINT    " LISTADO POR FECHA DE INICIO"
10100   PRINT#2,FF\PRINT#2,\  PRINT#2,\  PRINT#2," LISTADO POR FECHA DE INICIO"
10200   PRINT\PRINT\PRINT " NOMBRE DE LA ACTIVIDAD                    NUM   ES  EF  HL  HT    COSTO "
10300   PRINT#2, \PRINT#2, \PRINT#2,   " NOMBRE DE LA ACTIVIDAD                    NUM   ES  EF  HL  HT    COSTO "
10400   FOR I=1 TO N
10500           J=AU(I)
10600           PRINT  A$(J);
                PRINT USING " ###  ### ### ### ###  $###,###.## "; &
                AU(I),ES(J),EF(J),HO(J),HL(J);C(J)
10700           PRINT#2,   A$(J);
                PRINT#2  USING " ###  ### ### ### ###  $###,###.## "; &
                AU(I),ES(J),EF(J),HO(J),HL(J);C(J)
10800   NEXT I
10900   PRINT\PRINT BEL+BEL\PRINT  "DIAGRAMA DE BARRAS"
11000   PRINT#2,FF \ PRINT#2,\ PRINT#2,\  PRINT#2,"DIAGRAMA DE BARRAS","NUMERO DE ACTIVIDAD"
11100   NI=1  !LIMITE INFERIOR DEL NUMERO DE ACTIVIDADES
11200   NA=40 !LIMITE SUPERIOR INICIAL
11300   IF N<NA THEN NA=N
11400   PRINT "DIA";   \  PRINT #2, "DIA";
11500   FOR I=NI TO NA
11600           PRINT USING " ##";AU(I);   \ PRINT #2 USING " ##";AU(I);
11700   NEXT I
11800   PRINT " " \ PRINT #2, " "
11900   FOR L=0 TO G-1
12000           PRINT    USING "###";L+1;
12100           PRINT#2 USING "###";L+1;
12200           FOR I=NI TO NA
12300                   J=AU(I)
12400                   IF ES(J) <= L AND EF(J) > L THEN
                                PRINT    USING "  ";"X";
                                PRINT#2 USING "  ";"X";
                                COSTO_DIA(L+1)=COSTO_DIA(L+1)+ C(J)/D(J)
                        ELSE
                                IF EF(J) <= L AND EF(J)+HO(J) > L
                                THEN PRINT    USING "  ";"*";
                                     PRINT#2 USING "  ";"*";
                                ELSE PRINT    USING "  ";".";
                                     PRINT#2 USING "  ";".";

12500           NEXT I
12600           PRINT " "\PRINT#2, " "
12700   NEXT L
12800   PRINT \ PRINT \  PRINT\ PRINT#2, \PRINT#2, \PRINT #2,
12900   NI=NI+40
13000   NA=NA+40
13100   IF N>NA-20 THEN 11400
13200   PRINT\PRINT "ANALISIS DE COSTOS"\PRINT
13300   PRINT#2,FF\PRINT#2\PRINT #2, "ANALISIS DE COSTOS"\PRINT #2
13350   COSTO_ACUM=0
13360   PRINT\PRINT      "  DIA      COSTO     COSTO ACUMULADO  PORCIENTO"; &
          "  0                          50                              100"\PRINT
13370   PRINT#2\PRINT#2,"  DIA      COSTO     COSTO ACUMULADO  PORCIENTO"; &
          "  0                          50                              100"\PRINT#2
13400   FOR L=1 TO G
13500           COSTO_ACUM=COSTO_ACUM+COSTO_DIA(L)
13600           PRINT USING " ####   $###,### ##    $###,###,###.##   ##.### %"; &
                        L,COSTO_DIA(L),COSTO_ACUM,COSTO_ACUM/COSTO*100;
```

```
13610                   PRINT TAB(47+COSTO_ACUM/COSTO*50);"*"
13700             PRINT#2 USING " ####   $###,###.##    $#######,###.##   ##.### %";&
                          L,COSTO_DIA(L),COSTO_ACUM,COSTO_ACUM/COSTO*100;
1  .0             PRINT#2, TAB(47+COSTO_ACUM/COSTO*50);"*"
13800     NEXT L
13900     CLOSE #2%\GO TO 32767
14000     IF ERR=5% THEN PRINT "ESE ARCHIVO NO SE ENCONTRO"
                   RESUME 505
          ELSE PRINT "HAY UN ERROR NUM";ERR," EN LA LINEA";ERL
                   GO TO 32767
32767 END
```

10

PROGRAMA DE LA RUTA CRITICA

REALIZADO POR EL ING.JORGE ONTIVEROS JUNCO

FACULTAD DE INGENIERIA
CENTRO DE CALCULO
COORDINACION DE PLANEACION
COMPUTADORA DIGITAL VAX 11-780

OBJETIVO

Este programa obtiene la **RUTA CRITICA** de cuando más **100** actividades, teniendo cada una cuando más **5** actividades precedentes.

Las actividades las numera el programa del 1 al 100 conforme aparecen en el archivo de datos. Dentro del nombre de la actividad podemos indicarle el número de la misma pero sin olvidarnos que este número no lo tomará en cuenta el programa.

METODO

Se emplea un algoritmo diseñado por el Dr. Murray Lasso para redes pequeñas. Publicado en las Memorias de Congreso de Computación de la Univesidad La Salle, en 1982.

DATOS

Los datos se graban en un ARCHIVO que sigue el siguiente formato:

```
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
! LOS DATOS SE ACOMODAN DE LA SIGUIENTE MANERA:
! EN EL PRIMER REGISTRO VA EN NUMERO DE ACTIVIDADES DE LA RED
!   DESDE LA PRIMERA COLUMNA Y DEBE VALER CUANDO MAS 100.
!   JUSTIFICADO A LA DERECHA.
!
! EN LOS SIGUIENTES  REGISTROS VAN  LAS ACTIVIDADES DE LA RED
! DE ACUERDO AL SIGUIENTE FORMATO:
! COLUMNAS  1 A 40 EN NOMBRE DE LA ACTIVIDAD
! COLUMNAS 41 A 45 LA DURACION, EN UNIDADES ENTERAS
! COLUMNAS 46 A 55 EL COSTO, EN VALOR REAL
! COLUMNAS 56 A 58 EL NUMERO DE ACTIVIDADES PRECEDENTES
!       SI ESTE DATO ES MAYOR QUE CERO ENTONCES INDICAR LAS
!       ACTIVIDADES PRECEDENTES
! COLUMNAS 59 A 61 PRIMERA ACTIVIDAD PRECEDENTE
! COLUMNAS 62 A 64 SEGUNDA ACTIVIDAD PRECEDENTE
! COLUMNAS 65 A 67 TERCERA ACTIVIDAD PRECEDENTE
! COLUMNAS 68 A 70 CUARTA  ACTIVIDAD PRECEDENTE
! COLUMNAS 71 A 73 QUINTA  ACTIVIDAD PRECEDENTE.
!++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Al iniciar la corrida el programa preguntará por el nombre del archivo que contiene los datos, el cual debe proporcionarse indicando únicamente el NOMBRE, sin mencionar que es '.DAT'
Por ejemplo si se grabaron los datos en el archivo DATOS.DAT debemos indicar, como nombre del archivo, la palabra DATOS.

EJEMPLO DE DATOS

```
columnas
0         1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890123
 22
1  PISO DEL SOTANO                          2      100.  1 17  0  0  0  0
2  PLOMERIA GENERAL                         3      120.  1 17
3  ZAPATAS                                  4       90.  0
4  CIMENTACION                              2      200.  1  3
5  LOZAS DEL JARDIN                         5      100.  1  6
6  TERRACERIA                               2       50.  2 10 20
7  EQUIPO DE LA COCINA                      1      150.  1  8
8  PISOS TERMINADOS                         3      100.  1  9
9  YESO                                    10      100.  3  2 16 18
10 GOTEROS Y BAJADAS                        1       50.  1 19
11 MARCOS Y TECHOS                          4      150   1  4
12 LADRILLERIA                              6      200   1 11
13 INSTALACION ELECTICA                     1       50.  1 14
14 PINTURA                                  3       50.  2  7 22
15 TERMINAR PISOS                           2       50.  2 21 14
16 COLOCACION DE CONDUITS Y REGISTROS       2       50.  1 11
17 DRENAJE                                  1       75.  1  4
18 AIRE ACONDICIONADO                       4      200   2 11  1
19 TECHOS Y CANCELES                        2      150   1 12
20 BAJADAS PLUVIALES                        1       50.  1  4
21 CARPINTERIA                              1       50.  1  8
22 PLOMERIA TERMINADA                       2       50.  1  8
```

RESULTADOS

Como resultados obtenemos un listado de los datos, la duración y el costo total del proyecto, las fechas de inicio próximo y tardío, las fechas de fin próximo y tardío y las holguras libre y total de cada actividad, en el orden en que entraron.

Después se clasificarán por su fecha de inicio próximo y se imprimirán, se obtendrá la Ruta Crítica dibujada en diagramas de Gantt, y por último se obtendrá el análisis de costo por día de proyecto, el acumulado y la gráfica de esto último. EL PROGRAMA GENERA UN LISTADO DE RESULTADOS EN EL ARCHIVO 'RESUL.LIS' IDENTICO A LO QUE SALIO EN LA TERMINAL.

Para comentarios y aclaraciones favor de dirigirse al Ing. Jorge Ontiveros en el CECAFI.

DATOS

12

8.1

22

1	PISO DEL SOTANO	2	100.	1	17	0	0	0	0
2	PLOMERIA GENERAL	3	120.	1	17				
3	ZAPATAS	4	90.	0					
4	CIMENTACION	2	200.	1	3				
5	LOZAS DEL JARDIN	5	100.	1	6				
6	TERRACERIA	2	50.	2	10	20			
7	EQUIPO DE LA COCINA	1	150.	1	8				
8	PISOS TERMINADOS	3	100.	1	9				
9	YESO	10	100.	3	2	16	18		
10	GOTEROS Y BAJADAS	1	50.	1	19				
11	MARCOS Y TECHOS	4	150	1	4				
12	LADRILLERIA	6	200	1	11				
13	INSTALACION ELECTICA	1	50.	1	14				
14	PINTURA	3	50.	2	7	22			
15	TERMINAR PISOS	2	50.	2	21	14			
16	COLOCACION DE CONDUITS Y REGISTROS	2	50.	1	11				
17	DRENAJE	1	75.	1	4				
18	AIRE ACONDICIONADO	4	200	2	11	1			
19	TECHOS Y CANCELES	2	150	1	12				
20	BAJADAS PLUVIALES	1	50.	1	4				
21	CARPINTERIA	1	50.	1	8				
22	PLOMERIA TERMINADA	2	50.	1	8				

1.9.

13

13

PROGRAMA DE LA RUTA CRITICA

REALIZADO POR JORGE ONTIVEROS JUNCO

COORDINACION DE PLANEACION

FACULTAD DE INGENIERIA

CENTRO DE CALCULO

COMPUTADORA DIGITAL VAX 11-780

RUTA CRITICA

1.10

14

DATOS LEIDOS EL NUMERO DE ACTIVIDADES ES DE: 22

NODO	NOMBRE	DURACION		COSTO	NODOS PREDECESORES					
1	PISO DEL SOTANO	2	$	100.00	1	17	0	0	0	0
2	PLOMERIA GENERAL	3	$	120.00	1	17	0	0	0	0
3	ZAPATAS	4	$	90.00	0	0	0	0	0	0
4	CIMENTACION	2	$	200.00	1	3	0	0	0	0
5	LOZAS DEL JARDIN	5	$	100.00	1	6	0	0	0	0
6	TERRACERIA	2	$	50.00	2	10	20	0	0	0
7	EQUIPO DE LA COCINA	1	$	150.00	1	8	0	0	0	0
8	PISOS TERMINADOS	3	$	100.00	1	9	0	0	0	0
9	YESO	10	$	100.00	3	2	16	18	0	0
10	GOTEROS Y BAJADAS	1	$	50.00	1	19	0	0	0	0
11	MARCOS Y TECHOS	4	$	150.00	1	4	0	0	0	0
12	LADRILLERIA	6	$	200.00	1	11	0	0	0	0
13	INSTALACION ELECTICA	1	$	50.00	1	14	0	0	0	0
14	PINTURA	3	$	50.00	2	7	22	0	0	0
15	TERMINAR PISOS	2	$	50.00	2	21	14	0	0	0
16	COLOCACION DE CONDUITS Y REGISTROS	2	$	50.00	1	11	0	0	0	0
17	DRENAJE	1	$	75.00	1	4	0	0	0	0
18	AIRE ACONDICIONADO	4	$	200.00	2	11	1	0	0	0
19	TECHOS Y CANCELES	2	$	150.00	1	12	0	0	0	0
20	BAJADAS PLUVIALES	1	$	50.00	1	4	0	0	0	0
21	CARPINTERIA	1	$	50.00	1	8	0	0	0	0
22	PLOMERIA TERMINADA	2	$	50.00	1	8	0	0	0	0

EL COSTO DEL PROYECTO ES DE $ 2,185.00

LA DURACION DEL PROYECTO ES DE 34 DIAS

RUTA CRITICA

RESULTADOS

NODO	NOMBRE	ES	EF	LS	LF	HT	HL	
1	1 PISO DEL SOTANO	7	9	8	10	1	0	
2	2 PLOMERIA GENERAL	7	10	11	14	4	0	
3	3 ZAPATAS	0	4	0	4	0	0	CRITICA
4	4 CIMENTACION	4	6	4	6	0	0	CRITICA
5	5 LOZAS DEL JARDIN	21	26	29	34	8	0	
6	6 TERRACERIA	19	21	27	29	8	0	
7	7 EQUIPO DE LA COCINA	27	28	28	29	1	0	
8	8 PISOS TERMINADOS	24	27	24	27	0	0	CRITICA
9	9 YESO	14	24	14	24	0	0	CRITICA
10	10 GOTEROS Y BAJADAS	18	19	26	27	8	0	
11	11 MARCOS Y TECHOS	6	10	6	10	0	0	CRITICA
12	12 LADRILLERIA	10	16	18	24	8	0	
13	13 INSTALACION ELECTICA	32	33	33	34	1	0	
14	14 PINTURA	29	32	29	32	0	0	CRITICA
15	15 TERMINAR PISOS	32	34	32	34	0	0	CRITICA
16	16 COLOCACION DE CONDUITS Y REGISTROS	10	12	12	14	2	0	
17	17 DRENAJE	6	7	7	8	1	0	
18	18 AIRE ACONDICIONADO	10	14	10	14	0	0	CRITICA
19	19 TECHOS Y CANCELES	16	18	24	26	8	0	
20	20 BAJADAS PLUVIALES	6	7	26	27	20	0	
21	21 CARPINTERIA	27	28	31	32	4	0	
22	22 PLOMERIA TERMINADA	27	29	27	29	0	0	CRITICA

LISTADO POR FECHA DE INICIO

1.12

16

NOMBRE DE LA ACTIVIDAD	NUM	ES	EF	HL	HT		COSTO
3 ZAPATAS	3	0	4	0	0	$	90.00
4 CIMENTACION	4	4	6	0	0	$	200.00
11 MARCOS Y TECHOS	11	6	10	0	0	$	150.00
17 DRENAJE	17	6	7	1	0	$	75.00
20 BAJADAS PLUVIALES	20	6	7	20	0	$	50.00
1 PISO DEL SOTANO	1	7	9	1	0	$	100.00
2 PLOMERIA GENERAL	2	7	10	4	0	$	120.00
12 LADRILLERIA	12	10	16	8	0	$	200.00
16 COLOCACION DE CONDUITS Y REGISTROS	16	10	12	2	0	$	50.00
18 AIRE ACONDICIONADO	18	10	14	0	0	$	200.00
9 YESO	9	14	24	0	0	$	100.00
19 TECHOS Y CANCELES	19	16	18	8	0	$	150.00
10 GOTEROS Y BAJADAS	10	18	19	8	0	$	50.00
6 TERRACERIA	6	19	21	8	0	$	50.00
5 LOZAS DEL JARDIN	5	21	26	8	0	$	100.00
8 PISOS TERMINADOS	8	24	27	0	0	$	100.00
7 EQUIPO DE LA COCINA	7	27	28	1	0	$	150.00
21 CARPINTERIA	21	27	28	4	0	$	50.00
22 PLOMERIA TERMINADA	22	27	29	0	0	$	50.00
14 PINTURA	14	29	32	0	0	$	50.00
13 INSTALACION ELECTICA	13	32	33	1	0	$	50.00
15 TERMINAR PISOS	15	32	34	0	0	$	50.00

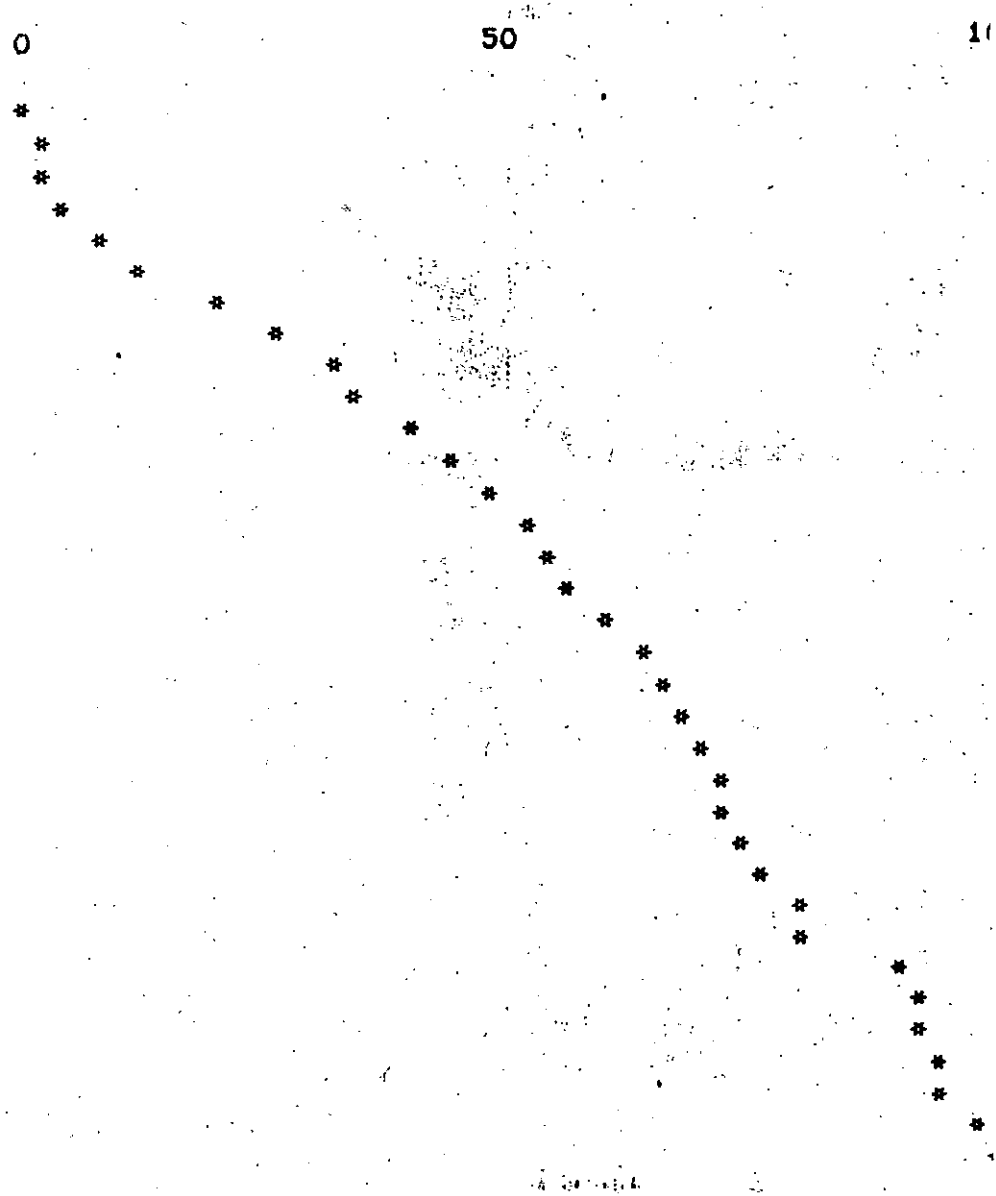
DIAGRAMA DE BARRAS — NUMERO DE ACTIVIDAD

17

DIA	3	4	11	17	20	1	2	12	16	18	9	19	10	6	5	8	7	21	22	14	13	15
1	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
6	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	.	.	X	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8	.	.	X	*	*	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	.	.	X	.	*	X	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
10	.	.	X	.	*	*	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11	.	.	.	.	*	.	*	X	X	X	.	.	.	.	.	.	.	.	.	.	.	.
12	.	.	.	.	*	.	*	X	X	X	.	.	.	.	.	.	.	.	.	.	.	.
13	.	.	.	.	*	.	*	X	*	X	.	.	.	.	.	.	.	.	.	.	.	.
14	.	.	.	.	*	.	*	X	*	X	.	.	.	.	.	.	.	.	.	.	.	.
15	.	.	.	.	*	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.
16	.	.	.	.	*	.	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.
17	.	.	.	.	*	.	.	*	.	.	X	X	.	.	.	.	.	.	.	.	.	.
18	.	.	.	.	*	.	.	*	.	.	X	X	.	.	.	.	.	.	.	.	.	.
19	.	.	.	.	*	.	.	*	.	.	X	*	X	.	.	.	.	.	.	.	.	.
20	.	.	.	.	*	.	.	*	.	.	X	*	*	X	.	.	.	.	.	.	.	.
21	.	.	.	.	*	.	.	*	.	.	X	*	*	X	.	.	.	.	.	.	.	.
22	.	.	.	.	*	.	.	*	.	.	X	*	*	*	X	.	.	.	.	.	.	.
23	.	.	.	.	*	.	.	*	.	.	X	*	*	*	X	.	.	.	.	.	.	.
24	.	.	.	.	*	.	.	*	.	.	X	*	*	*	X	.	.	.	.	.	.	.
25	.	.	.	.	*	.	.	.	.	.	.	*	*	*	X	X	.	.	.	.	.	.
26	.	.	.	.	*	.	.	.	.	.	.	*	*	*	X	X	.	.	.	.	.	.
27	.	.	.	.	*	.	.	.	.	.	.	.	*	*	*	X	.	.	.	.	.	.
28	.	.	.	.	.	.	.	.	.	.	.	.	.	*	*	.	X	X	X	.	.	.
29	.	.	.	.	.	.	.	.	.	.	.	.	.	*	*	.	*	*	X	.	.	.
30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*	.	.	*	.	X	.	.
31	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*	.	.	*	.	X	.	.
32	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*	.	.	*	.	X	.	.
33	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*	.	.	.	.	.	X	X
34	.	.	.	.	.	.	.	.	.	.	.	.	.	.	*	.	.	.	.	.	*	X

ANALISIS DE COSTOS

DIA	COSTO	COSTO ACUMULADO	PORCIENTO
1	$ 22.50	$ 22.50	1.030 %
2	$ 22.50	$ 45.00	2.059 %
3	$ 22.50	$ 67.50	3.089 %
4	$ 22.50	$ 90.00	4.119 %
5	$ 100.00	$ 190.00	8.696 %
6	$ 100.00	$ 290.00	13.272 %
7	$ 162.50	$ 452.50	20.709 %
8	$ 127.50	$ 580.00	26.545 %
9	$ 127.50	$ 707.50	32.380 %
10	$ 77.50	$ 785.00	35.927 %
11	$ 108.33	$ 893.33	40.885 %
12	$ 108.33	$ 1,001.67	45.843 %
13	$ 83.33	$ 1,085.00	49.657 %
14	$ 83.33	$ 1,168.33	53.471 %
15	$ 43.33	$ 1,211.67	55.454 %
16	$ 43.33	$ 1,255.00	57.437 %
17	$ 85.00	$ 1,340.00	61.327 %
18	$ 85.00	$ 1,425.00	65.217 %
19	$ 60.00	$ 1,485.00	67.963 %
20	$ 35.00	$ 1,520.00	69.565 %
21	$ 35.00	$ 1,555.00	71.167 %
22	$ 30.00	$ 1,585.00	72.540 %
23	$ 30.00	$ 1,615.00	73.913 %
24	$ 30.00	$ 1,645.00	75.286 %
25	$ 53.33	$ 1,698.33	77.727 %
26	$ 53.33	$ 1,751.67	80.168 %
27	$ 33.33	$ 1,785.00	81.693 %
28	$ 225.00	$ 2,010.00	91.991 %
29	$ 25.00	$ 2,035.00	93.135 %
30	$ 16.67	$ 2,051.67	93.898 %
31	$ 16.67	$ 2,068.33	94.661 %
32	$ 16.67	$ 2,085.00	95.423 %
33	$ 75.00	$ 2,160.00	98.856 %
34	$ 25.00	$ 2,185.00	% 100 %



OPTIMIZACION LINEAL CONTINUA

L P M O S S

OBJETIVO

REALIZAR LA OPTIMIZACION LINEAL CONTINUA DE UN MODELO MATEMATICO

METODO

---------- SIMPLEX ----------

ESTE PROGRAMA USA EL METODO DE LA GRAN M DEL METODO SIMPLEX PARA RESOLVER UN PROBLEMA DE OPTIMIZACION LINEAL, TAL COMO SE DESCRIBE EN EL LIBRO DE HILLIER AND LIEBERMAN, INTITULADO
'INTRODUCTION TO OPERATIONS RESEARCH'
EDITORIAL HOLDEN DAY.

DATOS

PRIMERA TARJETA: SE ESCRIBE EL TITULO DEL PROBLEMA (80 COLS.)

SEGUNDA TARJETA:

NUMERO DE VARIABLES	COLUMNAS 1 A 10
NUMERO DE RESTRICCIONES	COLUMNAS 11 A 20
TIPO DE OPTIMIZACION:	COLUMNAS 21 A 28
MAXIMIZA	
MINIMIZA	

TERCERA(S) TARJETA(S) (UNA POR CADA RESTRICCION)

NOMBRE DE LA RESTRICCION	COLUMNAS 1 A 10
TIPO DE RESTRICCION:	COLUMNA 13
< PARA INDICAR	<=
> PARA INDICAR	>=
= PARA INDICAR	=
VALOR DE LA RESTRICCION	COLUMNAS 21 A 30

CUARTA(S) TARJETA(S) (UNA POR CADA VARIABLE)

NOMBRE DE LA VARIABLE	COLUMNAS 1 A 10
VALOR EN LA FUNCION OBJETIVO	COLUMNAS 21 A 30

QUINTA(S) TARJETA(S) MATRIZ DEL SISTEMA, POR RENGLONES
CADA RENGLON DEBE INICIAR EN UNA NUEVA TARJETA
SE ESCRIBE UN DATO CADA 10 COLUMNAS.

NOTAS:

1) LOS NUMEROS SE DEBEN JUSTIFICAR A LA DERECHA.
2) LOS NOMBRES SE DEBEN JUSTIFICAR A LA IZQUIERDA.

EJEMPLO DE DATOS

```
COLUMNAS
C.        1         2         3         4
1234567890123456789012345678901234567890
```

EJEMPLO DE TAREA

```
         3          3MAXIMIZA
TIERRA       <          50.0
RHS/TRACT    <        1000.0
DINERO       <       70000.0
HA. MAIZ             3000.0
HA. SORGO            3800.0
HA. FRIJOL           4100.0
      1.0        1.0        1.0
     20.0       25.0       15.0
   1700.0     2075.0     2200.0
```

RESULTADOS

DATOS LEIDOS
TABLA INICIAL DEL METODO SIMPLEX
VARIABLES QUE ENTRAN Y SALEN DE LA BASE
TABLA TRANSFORMADA
MENSAJES CORRESPONDIENTES, SEGUN EL CASO:

BASE ACTUAL ES OPTIMA
SOLUCION NO ACOTADA
SOLUCION NO FACTIBLE

NOTAS

ES CONVENIENTE CREAR UN ARCHIVO DE DATOS Y ASIGNARLO A LA UNIDAD DE ENTRADA DEL PROGRAMA.

PARA ACLARACIONES Y COMENTARIOS FAVOR DE DIRIGIRSE CON EL ING. JORGE ONTIVEROS JUNCO, EN EL CECAFI

CENTRO DE CALCULO DE LA FACULTAD DE INGENIERIA

OPTIMIZACION LINEAL CONTINUA

LPMOSS

COMPUTADORA DIGITAL VAX 11/780

PLANEACION

LOS DATOS SON

TITULO :EJEMPLO DE TAREA
NUMERO DE RESTRICCIONES 3
NUMERO DE VARIABLES 3
TIPO DE OPTIMIZACION MAXIMIZA

RESTRIC. NUM.	NOMBRE	TIPO	RHS
1	TIERRA	<	50.00
2	'RS/TRACT	<	1000.00
3	DINERO	<	7000.00

VARIABLE NUM.	NOMBRE	COEFICIENTE EN LA FUNCION OBJETIVO
1	HA. MAIZ	30.00
2	HA. SORGO	38.00
3	HA. FRIJOL	41.00

MATRIZ DEL SISTEMA

1.00	1.00	1.00
20.00	25.00	15.00
170.00	207.50	220.00

FIN DE DATOS

-------- DATOS INICIALES -------- EJEMPLO DE TAREA

RENGLON	L. D.	BASE	HA. MAIZ	HA. SORGO	HA. FRIJOL	HOLGURA 1	HOLGURA 2	HOLGURA 3
FM	0.000		0.000	0.000	0.000	0.000	0.000	0.000
-C	0.000	Z	-30.000	-38.000	-41.000	0.000	0.000	0.000
TIERRA	50.000	HOLGURA 1	1.000	1.000	1.000	1.000	0.000	0.000
'RS/TRACT	1000.000	HOLGURA 2	20.000	25.000	15.000	0.000	1.000	0.000
DINERO	7000.000	HOLGURA 3	170.000	207.500	220.000	0.000	0.000	1.000

-------- TABLA INICIAL --------

RENGLON	L. D	BASE	HA MAIZ	HA. SORGO	HA. FRIJOL	HOLGURA 1	HOLGURA 2	HOLGURA 3
FM	0.000		0.000	0.000	0.000	0.000	0.000	0.000
-C	0.000	Z	-30.000	-38.000	-41.000	0.000	0.000	0.000
TIERRA	50.000	HOLGURA 1	1.000	1.000	1.000	1.000	0.000	0.000
'RS/TRACT	1000.000	HOLGURA 2	20.000	25.000	15.000	0.000	1.000	0.000
DINERO	7000.000	HOLGURA 3	170.000	207.500	220.000	0.000	0.000	1.000

HA. FRIJOL ENTRA A LA BASE. HOLGURA 3 SALE DE LA BASE

RENGLON	L. D.	BASE	HA. MAIZ	HA. SORGO	HA. FRIJOL	HOLGURA 1	HOLGURA 2	HOLGURA 3
FM	0.000		0.000	0.000	0.000	0.000	0.000	0.000
-C	1304.545	Z	1.682	0.670	0.000	0.000	0.000	0.186
TIERRA	18.182	HOLGURA 1	0.227	0.057	0.000	1.000	0.000	-0.005
'RS/TRACT	522.727	HOLGURA 2	8.409	10.852	0.000	0.000	1.000	-0.068
DINERO	31.818	HA. FRIJOL	0.773	0.943	1.000	0.000	0.000	0.005

******** LA BASE ACTUAL ES OPTIMA ********

EJEMPLO DE TAREA

```
0010 REM SAVE,9,'ANMAIN02.BAS'
0020 REM DIM N$20
0030 PRINT 'NOMBRE DE LA ESTRUCTURA?'
0040 INPUT N$
0050 PRINT 'NUMERO DE NAVES?';
0060 INPUT N1
0070 PRINT 'NUMERO DE CONDICIONES DE CARGA INDEPENDIENTES?'
0080 INPUT N2
0090 N3=4*N1
0100 N4=N1+1
0110 N5=4*N1+1
0120 DIM A(26,4)
0130 DIM V(26,4)
0140 DIM M(26,4)
0150 DIM B(26,4)
0160 DIM W(26,4)
0170 DIM N(26,4)
0180 DIM L(26)
0190 DIM G(26)
0200 DIM C(26)
0210 DIM I(26)
0220 DIM K(6,6)
0230 DIM T(3,3)
0240 DIM H(3,3)
0250 DIM S(63,6)
0260 DIM P(63,4)
0270 DIM X(3),Y(3),Z(3),Q(3,3),R(3,3),U(3,3),D(6)
0280 PRINT 'MODULO DE ELASTICIDAD?'
0290 INPUT E
0300 FOR X=1 TO N3+N4
0310 PRINT 'DATOS DE LA';
0320 IF X>N3 GOTO 350
0330 PRINT ' TRABE';X
0340 GOTO 360
0350 PRINT ' COLUMNA';X-N3
0360 PRINT 'LONGITUD?'
0370 INPUT L(X)
0380 PRINT 'INCLINACION?'
0390 INPUT G(X)
0400 G(X)=G(X)*3.14159/180
0410 PRINT 'AREA DE LA SECCION?'
0420 INPUT C(X)
0430 PRINT 'MOMENTO DE INERCIA?'
0440 INPUT I(X)
0450 NEXT X
0460 FOR X=1 TO N2
0470 PRINT 'DATOS DE LA CONDICION DE CARGA';X
0480 PRINT 'LA CARGA SE APLICA SOBRE:'
0490 PRINT '1.-NUDO'
0500 PRINT '2.-TRABE O COLUMNA O'
0510 PRINT '3.-FIN DE LA CONDICION'
0520 PRINT 'INDIQUE OPCION'
0530 INPUT Y
0540 ON Y GOTO 550,660,1110
0550 PRINT 'NUMERO DE NUDO?'
0560 INPUT N
0570 F1=0
0580 F2=0
0590 F3=0
0600 PRINT 'VALORES DE FX,FY,FZ?'
```

```
0620 P(3*N-2,X)=P(3*N-2,X)+F1
0630 P(3*N-1,X)=P(3*N-1,X)+F2
0640 P(3*N,X)=P(3*N,X)+F3
0650 GOTO 1100
 660 PRINT 'EL MIEMBRO CARGADO ES: '
 670 PRINT '1.-TRABE'
0680 PRINT '2-COLUMNA'
0690 PRINT 'INDIGUE OPCION'
0700 INPUT Y
0710 ON Y GOTO 720,750
0720 PRINT 'NUMERO DE LA TRABE?'
0730 Z=0
0740 GOTO 770
0750 PRINT 'NUMERO DE LA COLUMNA?'
0760 Z=N3
0770 INPUT N
0780 N=Z+N
0790 PRINT 'LA CARGA ES: '
0800 PRINT '1.-CONCENTRADA'
0810 PRINT '2.-DISTRIBUIDA'
0820 PRINT 'INDIGUE OPCION'
0830 INPUT Y
0840 ON Y GOTO 850,980
0850 F1=0
0860 F2=0
0870 PRINT 'VALORES DE FY,Fx?'
0880 INPUT F2,F1
0890 PRINT 'POSICION DONDE SE APLICA(N) LA(S) CARGA(S)?'
0900 INPUT P
 910 A(N,X)=A(N,X)-F1*P/L(N)
 920 V(N,X)=V(N,X)-F2*(L(N)-P)^2/(L(N)^3)*(3*P+L(N)-P)
0930 M(N,X)=M(N,X)-F2*P*(L(N)-P)^2/L(N)^2
0940 B(N,X)=B(N,X)-F1*(L(N)-P)/L(N)
0950 W(N,X)=W(N,X)-F2*P*P/(L(N)^3)*(3*(L(N)-P)-P)
0960 N(N,X)=N(N,X)+F2*P*P*(L(N)-P)/L(N)^2
0970 GOTO 1100
0980 W1=0
0990 W2=0
1000 PRINT 'VALORES DE WY,WX?'
1010 INPUT W2,W1
1020 PRINT 'POSICION DONDE TERMINA(N) LA(S) CARGA(S)?'
1030 INPUT P
1040 A(N,X)=A(N,X)-W1*P*P/2/L(N)
1050 V(N,X)=V(N,X)-W2*(P-P^2/L(N)+P^3/L(N)^2/2)
1060 M(N,X)=M(N,X)-W2/12*(3*P^3/L(N)-8*P*P-6*P*L(N))
1070 B(N,X)=B(N,X)-W1*P*(L(N)-P/2)/L(N)
1080 W(N,X)=W(N,X)-W2*(P*P/L(N)-P^3/L(N)^2/2)
1090 N(N,X)=N(N,X)+W2/12*(-3*P^3/L(N)+4*P*P)
1100 GOTO 470
1110 NEXT X
1120 FOR X=1 TO N3
1130 C=COS(G(X))
1140 S=SIN(G(X))
1150 FOR Y=1 TO N2
1160 P(3*X-2,Y)=P(3*X-2,Y)-(A(X,Y)*C-V(X,Y)*S)
1170 P(3*X-1,Y)=P(3*X-1,Y)-(A(X,Y)*S+V(X,Y)*C)
1180 P(3*X,Y)=P(3*X,Y)-M(X,Y)
1190 P(3*(X+1)-2,Y)=P(3*(X+1)-2,Y)-(B(X,Y)*C-W(X,Y)*S)
1200 P(3*(X+1)-1,Y)=P(3*(X+1)-1,Y)-(B(X,Y)*S+W(X,Y)*C)
1210 P(3*(X+1),Y)=P(3*(X+1),Y)-N(X,Y)
```

```
1230 NEXT X
1240 FOR X=N3+1 TO N3+N4
1250 C=COS(G(X))
1260 S=SIN(G(X))
1270 Z=(X-N3-1)*4+1
1280 FOR Y=1 TO N2
1290 P(3*Z-2,Y)=P(3*Z-2,Y)-(B(X,Y)*C-W(X,Y)*S)
1300 P(3*Z-1,Y)=P(3*Z-1,Y)-(B(X,Y)*S+W(X,Y)*C)
1310 P(3*Z,Y)=P(3*Z,Y)-N(X,Y)
1320 NEXT Y
1330 NEXT X
1340 FOR X=1 TO N3
1350 PRINT 'ENSAMBLADO TRABE';X
1360 GOSUB 3010
1370 X1=3*X-2
1380 Y1=1
1390 X2=1
1400 Y2=1
1410 GOSUB 3730
1420 X1=3*(X+1)-2
1430 Y1=1
1440 X2=4
1450 Y2=4
1460 GOSUB 3730
1470 X1=3*X-2
1480 Y1=4
1490 X2=1
1500 Y2=4
1510 GOSUB 3730
1520 NEXT X
1530 FOR X=N3+1 TO N3+N4
1540 PRINT 'ENSAMBLANDO COLUMNA';X-N3
1550 GOSUB 3010
1560 X1=3*((X-N3-1)*4+1)-2
1570 Y1=1
1580 X2=4
1590 Y2=4
1600 GOSUB 3730
1610 NEXT X
1620 FOR X=1 TO N5
1630 PRINT 'TRIANGULARIZANDO NUDO';X
1640 GOSUB 3790
1650 IF X=N5 GOTO 1940
1660 FOR I1=1 TO 3
1670 FOR J1=1 TO 3
1680 Q(I1,J1)=S(3*(X-1)+I1,3+J1)
1690 R(J1,I1)=0
1700 FOR K1=1 TO 3
1710 R(J1,I1)=R(J1,I1)+S(3*(X-1)+I1,K1)*S(3*(X-1)+K1,3+J1)
1720 NEXT K1
1730 NEXT J1
1740 NEXT I1
1750 FOR I1=1 TO 3
1760 FOR J1=1 TO 3
1770 S(3*(X-1)+I1,3+J1)=R(I1,J1)
1780 NEXT J1
1790 NEXT I1
1800 FOR I1=1 TO 3
1810 FOR J1=1 TO 3
1820 U(I1,J1)=0
```

```
1840 U(I1,J1)=U(I1,J1)+R(I1,K1)*G(K1,J1)
1850 NEXT K1
1860 NEXT J1
1870 NEXT I1
 380 FOR I1=1 TO 3
 390 FOR J1=1 TO 3
1900 S(3*X+I1,J1)=S(3*X+I1,J1)-U(I1,J1)
1910 NEXT J1
1920 NEXT I1
1930 NEXT X
1940 PRINT 'SUSTITUCION HACIA ADELANTE'
1950 FOR X=1 TO N2
1960 FOR Y=1 TO N5-1
1970 FOR I1=1 TO 3
1980 Z(I1)=0
1990 FOR J1=1 TO 3
2000 Z(I1)=Z(I1)+S(3*(Y-1)+I1,3+J1)*P(3*(Y-1)+J1,X)
2010 NEXT J1
2020 NEXT I1
2030 FOR I1=1 TO 3
2040 P(3*Y+I1,X)=P(3*Y+I1,X)-Z(I1)
2050 NEXT I1
2060 NEXT Y
2070 NEXT X
2080 PRINT 'SUSTITUCION HACIA ATRAS'
2090 FOR X=1 TO N2
2100 FOR I1=1 TO 3
2110 Z(I1)=0
2120 FOR J1=1 TO 3
2130 Z(I1)=Z(I1)+S(3*(N5-1)+I1,J1)*P(3*(N5-1)+J1,X)
 140 NEXT J1
2150 NEXT I1
2160 FOR I1=1 TO 3
2170 P(3*(N5-1)+I1,X)=Z(I1)
2180 NEXT I1
2190 FOR Y=N5-1 TO 1 STEP -1
2200 FOR I1=1 TO 3
2210 X(I1)=0
2220 FOR J1=1 TO 3
2230 X(I1)=X(I1)+S(3*(Y-1)+I1,J1)*P(3*(Y-1)+J1,X)
2240 NEXT J1
2250 NEXT I1
2260 FOR I1=1 TO 3
2270 Y(I1)=0
2280 FOR J1=1 TO 3
2290 Y(I1)=Y(I1)+S(3*(Y-1)+J1,3+I1)*P(3*Y+J1,X)
2300 NEXT J1
2310 NEXT I1
2320 FOR I1=1 TO 3
2330 P(3*(Y-1)+I1,X)=X(I1)-Y(I1)
2340 NEXT I1
2350 NEXT Y
2360 NEXT X
2370 FOR X=1 TO N3
2380 PRINT 'ELEMENTOS MECANICOS TRABE';X
2390 GOSUB 3010
2400 S=SIN(G(X))
2410 C=COS(G(X))
2420 FOR Y=1 TO N2
2430 FOR I1=1 TO 6
```

```
2450 FOR J1=1 TO 6
2460 D(I1)=D(I1)+K(I1,J1)*P(3*(X-1)+J1,Y)
2470 NEXT J1
2480 NEXT I1
2490 A(X,Y)=A(X,Y)+D(1)*C+D(2)*S
2500 V(X,Y)=V(X,Y)-D(1)*S+D(2)*C
2510 M(X,Y)=M(X,Y)+D(3)
2520 B(X,Y)=B(X,Y)+D(4)*C+D(5)*S
2530 W(X,Y)=W(X,Y)-D(4)*S+D(5)*C
2540 N(X,Y)=N(X,Y)+D(6)
2550 NEXT Y
2560 NEXT X
2570 FOR X=N3+1 TO N3+N4
2580 PRINT 'ELEMENTOS MECANICOS COLUMNA';X-N3
2590 GOSUB 3010
2600 C=COS(G(X))
2610 S=SIN(G(X))
2620 Z=(X-N3-1)*4+1
2630 FOR Y=1 TO N2
2640 FOR I1= 1 TO 6
2650 D(I1)=0
2660 FOR J1=1 TO 3
2670 D(I1)=D(I1)+K(I1,3+J1)*P(3*(Z-1)+J1,Y)
2680 NEXT J1
2690 NEXT I1
2700 A(X,Y)=A(X,Y)+D(1)*C+D(2)*S
2710 V(X,Y)=V(X,Y)-D(1)*S+D(2)*C
2720 M(X,Y)=M(X,Y)+D(3)
2730 B(X,Y)=B(X,Y)+D(4)*C+D(5)*S
2740 W(X,Y)=W(X,Y)-D(4)*S+D(5)*C
2750 N(X,Y)=N(X,Y)+D(6)
2760 NEXT Y
2770 NEXT X
2780 FOR X=1 TO N2
2784 PRINT 'CAMBIAR IMPRESORA A PRINCIPIO DE HOJA'
2790 PRINT 'ESTRUCTURA ',N$
2800 PRINT 'R E S U L T A D O S  C O N D I C I O N';X
2810 PRINT 'E L E M E N T O S  M E C A N I C O S'
2820 PRINT 'AXIAL',,'CORT',,'MOM'
2830 FOR Y=1 TO N3+N4
2840 IF Y>N3 GOTO 2870
2850 PRINT '              TRABE';Y
2860 GOTO 2890
2870 PRINT '                COLUMNA',Y-N3
2890 PRINT  A(Y,X),,V(Y,X),,M(Y,X)
2910 PRINT  B(Y,X),,W(Y,X),,N(Y,X)
2920 NEXT Y
2930 PRINT  'D E S P L A Z A M I E N T O S'
2940 PRINT  'NUDO','EN X',,'EN Y',,'GIRO'
2950 FOR Y=1 TO N5
2960 PRINT  Y,P(3*Y-2,X),,P(3*Y-1,X),,P(3*Y,X)
2970 NEXT Y
2980 NEXT X
2990 PRINT 'FIN DEL ANALISIS'
3000 GO TO 32767
3010 R(1,1)=E*C(X)/L(X)
3020 R(1,2)=0
3030 R(1,3)=0
3040 R(2,1)=0
3050 R(2,2)=12*E*I(X)/L(X)^3
```

```
3070 R(3,1)=0
3080 R(3,2)=R(2,3)
3090 R(3,3)=4*E*I(X)/L(X)
3100 T(1,1)=COS(G(X))
3110 T(1,2)=SIN(G(X))
3120 T(1,3)=0
3130 T(2,1)=-T(1,2)
3140 T(2,2)=T(1,1)
3150 T(2,3)=0
3160 T(3,1)=0
3170 T(3,2)=0
3180 T(3,3)=1
3190 FOR I1=1 TO 3
3200 FOR J1=1 TO 3
3210 Q(I1,J1)=0
3220 FOR K1=1 TO 3
3230 Q(I1,J1)=Q(I1,J1)+R(I1,K1)*T(K1,J1)
3240 NEXT K1
3250 NEXT J1
3260 NEXT I1
3270 T1=T(1,2)
3280 T(1,2)=T(2,1)
3290 T(2,1)=T1
3300 FOR I1=1 TO 3
3310 FOR J1=1 TO 3
3320 K(3+I1,3+J1)=0
3330 FOR K1=1 TO 3
3340 K(3+I1,3+J1)=K(3+I1,3+J1)+T(I1,K1)*Q(K1,J1)
3350 NEXT K1
3360 NEXT J1
3370 NEXT I1
3380 H(1,1)=1
3390 H(1,2)=0
3400 H(1,3)=0
3410 H(2,1)=0
3420 H(2,2)=1
3430 H(2,3)=0
3440 H(3,1)=-L(X)*SIN(G(X))
3450 H(3,2)=L(X)*COS(G(X))
3460 H(3,3)=1
3470 FOR I1=1 TO 3
3480 FOR J1=1 TO 3
3490 K(I1,3+J1)=0
3500 FOR K1=1 TO 3
3510 K(I1,3+J1)=K(I1,3+J1)-H(I1,K1)*K(3+K1,3+J1)
3520 NEXT K1
3530 NEXT J1
3540 NEXT I1
3550 FOR I1=1 TO 3
3560 FOR J1=1 TO 3
3570 K(3+J1,I1)=K(I1,3+J1)
3580 NEXT J1
3590 NEXT I1
3600 H(1,3)=H(3,1)
3610 H(3,1)=0
3620 H(2,3)=H(3,2)
3630 H(3,2)=0
3640 FOR I1=1 TO 3
3650 FOR J1=1 TO 3
3660 K(I1,J1)=0
```

```
3680 K(I1,J1)=K(I1,J1)-K(I1,3+K1)*H(K1,J1)
3690 NEXT K1
3700 NEXT J1
3710 NEXT I1
3720 RETURN
3730 FOR I1=0 TO 2
3740 FOR J1=0 TO 2
3750 S(X1+I1,Y1+J1)=S(X1+I1,Y1+J1)+K(X2+I1,Y2+J1)
3760 NEXT J1
3770 NEXT I1
3780 RETURN
3790 FOR I1=1 TO 3
3800 FOR J1=1 TO 3
3810 Q(I1,J1)=S(3*(X-1)+I1,J1)
3820 NEXT J1
3830 NEXT I1
3840 MAT U=INV(Q)
3850 FOR I1=1 TO 3
3860 FOR J1=1 TO 3
3870 S(3*(X-1)+I1,J1)=U(I1,J1)
3880 NEXT J1
3890 NEXT I1
3900 RETURN
32767 END
```

```
EJEMPLO BASIC PARTE 2
1
1
210000000.
1
30
0.001
0.00001
2
30
0.001
0.00001
2
-30
0.001
0.00001
1
-30
0.001
0.00001
5
90
0.001
0.00004
5
90
0.001
0.    004
1
3
0,-10,0
3
```

```
NOMBRE DE LA ESTRUCTURA?
NUMERO DE CONDICIONES DE CARGA INDEPENDIENTES?
MODULO DE ELASTICIDAD?
DATOS DE LA TRABE 1
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA TRABE 2
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA TRABE 3
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA TRABE 4
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA COLUMNA 1
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA COLUMNA 2
LONGITUD?
INCLINACION?
AREA DE LA SECCION?
MOMENTO DE INERCIA?
DATOS DE LA CONDICION DE CARGA 1
LA CARGA SE APLICA SOBRE;
1.-NUDO
2.-TRABE O COLUMNA O
3.-FIN DE LA CONDICION
INDIQUE OPCION
NUMERO DE NUDO?
VALORES DE FX,FY,FZ?
DATOS DE LA CONDICION DE CARGA 1
LA CARGA SE APLICA SOBRE;
1.-NUDO
2.-TRABE O COLUMNA O
3.-FIN DE LA CONDICION
INDIQUE OPCION
ENSAMBLADO TRABE 1
ENSAMBLADO TRABE 2
ENSAMBLADO TRABE 3
ENSAMBLADO TRABE 4
ENSAMBLANDO COLUMNA 1
ENSAMBLANDO COLUMNA 2
TRIANGULARIZANDO NUDO 1
TRIANGULARIZANDO NUDO 2
TRIANGULARIZANDO NUDO 3
TRIANGULARIZANDO NUDO 4
TRIANGULARIZANDO NUDO 5
SUSTITUCION HACIA ADELANTE
SUSTITUCION HACIA ATRAS
ELEMENTOS MECANICOS TRABE 1
```

ELEMENTOS MECANICOS TRABE 2
ELEMENTOS MECANICOS TRABE 3
ELEMENTOS MECANICOS TRABE 4
ELEMENTOS MECANICOS COLUMNA 1
ELEMENTOS MECANICOS COLUMNA 2
CAMBIAR IMPRESORA A PRINCIPIO DE HOJA
ESTRUCTURA EJEMPLO BASIC PARTE 2
R E S U L T A D O S C O N D I C I O N 1
E L E M E N T O S M E C A N I C O S

AXIAL	CORT	MOM
	TRABE 1	
4.12222	3.3935	4.97044
-4.12222	-3.3935	-1.57693
	TRABE 2	
4.12221	3.39349	1.57693
-4.12221	-3.39349	5.21003
	TRABE 3	
4.12235	-3.39354	-5.21003
-4.12235	3.39354	-1.57705
	TRABE 4	
4.12241	-3.39354	1.57708
-4.12241	3.39354	-4.97061
	COLUMNA 1	
4.99998	-1.87318	-4.39545
-4.99998	1.87318	-4.97044
	COLUMNA 2	
5.00003	1.87331	4.39594
-5.00003	-1.87331	4.97062

D E S P L A Z A M I E N T O S

NUDO	EN X	EN Y	GIRO
1	-.189507E-01	-.119045E-02	-.171128E-02
2	-.136945E-01	-.106071E-01	-.173003E-01
3	.202749E-05	-.351953E-01	.212342E-06
4	.136987E-01	-.106867E-01	.173001E-01
5	.189547E-01	-.119051E-02	.171035E-02

FIN DEL ANALISIS

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

(1)

S I S T E M A

C O N. C. R E. T. O. IBM S/23

ULTIMA FECHA DE ACCESO : 83/03/28

CLAVE DE ACCESO :

FECHA DE HOY : 83 / 03 / 00

A U T O R : S I S T E M A S C O M P U T A R I Z A D O S
E N I N G E N I E R I A
ING. CARLOS A. RAMOS LARIOS TEL.670-9826

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

(2)

OPCIONES DEL SISTEMA CON.C.RE.T.O S/23:

A) ALTAS,BAJAS,CAMBIOS,MUESTRA Y LISTADOS DE ARCHIVOS
B) ANALISIS DE COSTOS Y PRECIOS UNITARIOS
C) PRESUPUESTO DE LA OBRA
D) CONTROL DE AVANCES
E) EXPLOSION DE RECURSOS
F) RUTINAS DE MANTENIMIENTO

Z) SALIDA DEL SISTEMA

INDIQUE OPCION: Z

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

(3)

ALTAS,BAJAS,CAMBIOS,MUESTRA Y LISTADOS DE ARCHIVOS:

a) MATERIALES
b) CATEGORIAS
c) CUADRILLAS
d) MAQUINARIA
e) EQUIPOS AUXILIARES
f) ELABORACION DE MORTEROS
g) ELABORACION DE CONCRETOS
h) CONCEPTOS
i) SUBCONTRATOS
j) NOMBRES DE CAPITULOS
k) NOMBRES DE PARTIDAS
l) COMPONENTES

Z) REGRESO A MENU PRINCIPAL

INDIQUE OPCION:

ANALISIS DE COSTOS Y PRECIOS UNITARIOS:

a) MATERIALES
b) CATEGORIAS
c) CUADRILLAS
d) MAQUINARIA
e) EQUIPOS AUXILIARES
f) ELABORACION DE MORTEROS
g) ELABORACION DE CONCRETOS
h) CONCEPTOS

z) REGRESO A MENU PRINCIPAL

INDIQUE OPCION: *

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

PRESUPUESTO DE LA OBRA

a) ACTUALIZACION DATOS GENERALES DE LA OBRA
b) CALCULO PRESUPUESTO TOTAL
c) CALCULO PRESUPUESTO FALTANTE POR EJECUTAR

z) REGRESO A MENU PRINCIPAL

INDIQUE OPCION: z

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

CONTROL DE AVANCES

a) BORRA ULTIMO AVANCE
b) IMPRESION FORMA PRE-ESTIMACION
c) CAPTURA Y LISTADO DE AVANCES
d) CALCULO DE LA ESTIMACION
e) REPORTE DEL ESTADO DE AVANCES
f) ACUMULACION DEL AVANCE AL TOTAL

z) REGRESO A MENU PRINCIPAL

INDIQUE OPCION: z

EXPLOSION DE RECURSOS:

a) DEL TOTAL DE LA OBRA
b) DEL ULTIMO AVANCE
c) DEL LA OBRA FALTANTE POR EJECUTAR

z) REGRESO A MENU PRINCIPAL

INDIQUE OPCION: z

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

RUTINAS DE MANTENIMIENTO

a) INICIA DISKETTE DE ARCHIVOS
b) REORGANIZA ARCHIVO DE INSUMOS

z) REGRESO A MENU PRINCIPAL

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

S I S T E M A

C O N. C. R E. I. O. IBM S/23

SALIDA NORMAL

DEL SISTEMA

FECHA DE HOY : 83/03/29

A U T O R : S I S T E M A S C O M P U T A R I Z A D O S
E N I N G E N I E R I A
ING. CARLOS A. RAMOS LARIOS TEL.670-9826

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

4.4

34

A R C H I V O D E M A T E R I A L E S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 1

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E M A T E R I A L E S

CLAVE 1-M001
NOMBRE AGUA
UNIDAD M3
COSTO DE COMPRA 0000135.00
o/o POR FLETE 00.000

ULTIMO CAMBIO COSTO: 82/09/13 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0000135.00 POR M3
EL MATERIAL YA FUE DADO DE ALTA.

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E M A T E R I A L E S

CLAVE 1-M001
NOMBRE AGUA
UNIDAD M3
COSTO DE COMPRA 0000135.00
o/o POR FLETE 00.000

ULTIMO CAMBIO COSTO: 82/09/13 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0000135.00 POR M3
SE DA DE BAJA S/N? n

A R C H I V O D E M A T E R I A L E S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 3

CLAVE DEL MATERIAL 1-M999

EL MATERIAL NO HA SIDO DADO DE ALTA. intro

ARQ. GUILLERMO SANCHEZ GARCIA.
OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

A R C H I V O D E M A T E R I A L E S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 7

LISTADO DE LA CLAVE 1-M001 A LA CLAVE 1-M010

ARQ. GUILLERMO SANCHEZ GARCIA.
OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

A R C H I V O D E M A T E R I A L E S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 8

LISTADO DE LA LETRA A A LA LETRA

36

A R C H I V O D E C A T E G O R I A S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU-PRINCIPAL

INDIQUE OPCION: 4

CLAVE DE LA CATEGORIA: 6-01

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E C A T E G O R I A S

CLAVE	6-01
NOMBRE	PEON
SALARIO BASE DIARIO	0365.00
DIAS PARA PRIMA VACACIONAL	06
DIAS PARA GRATIFICACION	15
DIAS TRABAJADOS POR AÑO	304.00
o/o PARA FACTOR DE ZONA	100.00
o/o PARA IMSS	04.6250
o/o PARA EDUCACION	01.00
o/o PARA GUARDERIAS	01.00
o/o PARA INFONAVIT	01.00

ULTIMO CAMBIO COSTO: 83/02/03 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0000498.61 POR JOR.

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E C U A D R I L L A S

CLAVE: 7-01
NOMBRE: ALBAÑIL DE SEGUNDA Y PEON

NUMERO DE INSUMOS: 02 (0529)

ULTIMO CAMBIO COSTO: 83/02/03 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0000564.86 POR JOR.

intro

I N S U M O S D E L A C U A D R I L L A 37 7-61

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
6-03	0000.1000	001.000	6-01	0001.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS (22)

D A T O S D E M A Q U I N A R I A

CLAVE 4-01
NOMBRE MAQUINA DE PRUEBA
MARCA 1 MODELO 2 CAPACIDAD 3
POTENCIA EN HP 004.0
VALOR DE ADQUISICION 000000005.00
% DE RESCATE 03
VIDA ECONOMICA EN AÑOS 07.0
HORAS DE USO POR AÑO 0008.0
P O R C E N T A J E S P A R A:
INTERESES:009.0 , SEGUROS:010.0 , ALMAC.:011.0 , MANT.:012.0
NUMERO DE INSUMOS: 04 (0257)

ULTIMO CAMBIO COSTO: 00/00/00 , ULTIMO CALCULO COSTO: 00/00/00
COSTO DIRECTO UNITARIO: 00000000.00 POR HORA

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS (23)

I N S U M O S D E L A M A Q U I N A 4-01

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
1-01	0001.0000	001.000	1-02	0002.0000	001.000
6-03	0003.0000	001.000	7-04	0004.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

D A T O S D E E Q U I P O S

CLAVE 5-EA01
NOMBRE PICO
DE USOS PARA DEPRECIACION 001

NUMERO DE INSUMOS: 02 (0287)

ULTIMO CAMBIO COSTO: 00/00/00 , ULTIMO CALCULO COSTO: 00/00/00
COSTO DIRECTO UNITARIO: 0000000.00 POR USO

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

(15)

I N S U M O S D E L E Q U I P O 5-EA01

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
1-01	0001.0000	001.000	4-3	0000.1000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

(2)

D A T O S D E M O R T E R O S

CLAVE 2-10
NOMBRE MORTERO,CEMENTO,ARENA,1:1:6
UNIDAD M3

NUMERO DE INSUMOS: 04 (0221)

ULTIMO CAMBIO COSTO: 83/03/15 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0002889.46 POR M3

intro

I N S U M O S D E L M O R T E R O 39 12-10

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
1-M015	0000.1280	001.000	1-M012	0000.2620	001.020
1-M009	0001.1300	001.020	1-M001	0000.2790	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E C O N C R E T O S

CLAVE 3-1CON.S
NOMBRE CONCRETO DE F'C 150 PROP.1:2:4
UNIDAD M3.

NUMERO DE INSUMOS: 04 (0308)

ULTIMO CAMBIO COSTO: 83/03/15 , ULTIMO CALCULO COSTO: 83/03/16
COSTO DIRECTO UNITARIO: 0003089.34 POR M3

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

I N S U M O S D E L C O N C R E T O 3-1CON.S

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
1-M012	0000.3380	001.020	1-M009	0000.4500	001.020
1-M028	0000.7000	001.000	1-M001	0000.2200	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

40

D A T O S D E C O N C E P T O S

CLAVE : 8-A026
NOMBRE : LAMBRIN DE AZULEJO NORMAL DE 11X11 PEGADO CON PASTA DE PEGA AZULEJO
UNIDAD : M2
RENDIMIENTO POR JOR.: 004.5000

NUMERO DE INSUMOS: 04 (0476)

ULTIMO CAMBIO COSTO: 83/03/15 , ULTIMO CALCULO COSTO: 83/03/23
COSTO DIRECTO UNITARIO: 0001206.73 POR M2

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

I N S U M O S D E L C O N C E P T O : 8-A026

CLAVE	CANTIDAD	FACTOR	CLAVE	CANTIDAD	FACTOR
1-M007	0001.0500	001.000	2-40	0000.0025	001.000
2-30	0000.0028	001.000	7-66	0000.2200	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000
	0000.0000	001.000		0000.0000	001.000

intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

D A T O S D E S U B C O N T R A T O S

CLAVE : 2-01
NOMBRE : EXCAVACION EN TERRENO TIPO II CON MAQUINA PARA DAR PISO DE DESPLANTE.
UNIDAD : LOTE

ULTIMO CAMBIO COSTO: 83/03/15
COSTO DIRECTO UNITARIO: 0421250.00 POR LOTE

intro

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

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 4

CLAVE DEL CAPITULO : 01
NOMBRE : ALBANILERIA
intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

A R C H I V O D E P A R T I D A S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CLAVE
6) LISTADO COMPLETO POR NOMBRE
7) LISTADO PARCIAL POR CLAVE
8) LISTADO PARCIAL POR NOMBRE
0) MENU PRINCIPAL

INDIQUE OPCION: 4

CLAVE DE LA PARTIDA : 01A
NOMBRE : ALBANILERIA
intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

A R C H I V O D E C O M P O N E N T E S

1) ALTA
2) BAJA
3) CAMBIO
4) MUESTRA
5) LISTADO COMPLETO POR CAPITULO-PARTIDA-COMPONENTE
6) LISTADO COMPLETO POR COMPONENTE
7) LISTADO PARCIAL POR CAPITULO-PARTIDA-COMPONENTE
8) LISTADO PARCIAL POR COMPONENETE
0) MENU PRINCIPAL

INDIQUE OPCION: 4

CAPITULO : 01 : PARTIDA : A : COMPONENTE : 8-A033

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

C O M P O N E N T E S

CAPITULO DEL COMPONENTE	"01"
PARTIDA DEL COMPONENTE	"A"
CLAVE DEL COMPONENTE	"G-A033 "
POS. REL. IMP.	00000
CANT. TOTAL EN LA OBRA	00000065.50
CANT. ACUM. EN AVANCES ANTERIORES	000000000.00
CANT. EN AVANCE ACTUAL	000000000.00
CANT. ACUM. ANTERIOR EN $	000000000.00

FECHA ACUMULACION DEL ULTIMO AVANCE A ESTE COMPONEN: 00/00/00

Intro

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

ANALISIS DE COSTOS DIRECTOS DE MATERIALES

1) ANALISIS COMPLETO CON DESGLOSE
2) ANALISIS COMPLETO CON RESUMEN
3) ANALISIS DE UN GRUPO CON DESGLOSE
4) ANALISIS DE UN GRUPO CON RESUMEN

0) MENU PRINCIPAL

INDIQUE OPCION: 0

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

DATOS GENERALES DE LA OBRA

NOMBRE: PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

P O R C E N T A J E S P A R A:

INDIRECTOS Y UTILIDAD:	28.0	%MANO INTERMEDIO Y HERRAM.:	00.0
INDIRECTOS:	30.0	MANO INTERMEDIO:	00.0
UTILIDAD:	15.0	HERRAMIENTA MENOR:	00.0

ULTIMO CAMBIO PRESU: 83/03/28 , ULTIMO CALCULO PRESU: 83/03/28
PRESUPUESTO ACTUAL: $50,207,077.06

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

O P C I O N E S D E L R E P O R T E:

1) INCLUYENDO TODOS LOS CAPITULOS Y PARTIDAS

2) INCLUYENDO SOLO UN GRUPO DE CAPITULOS Y PARTIDAS

INDIQUE OPCION: 0

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

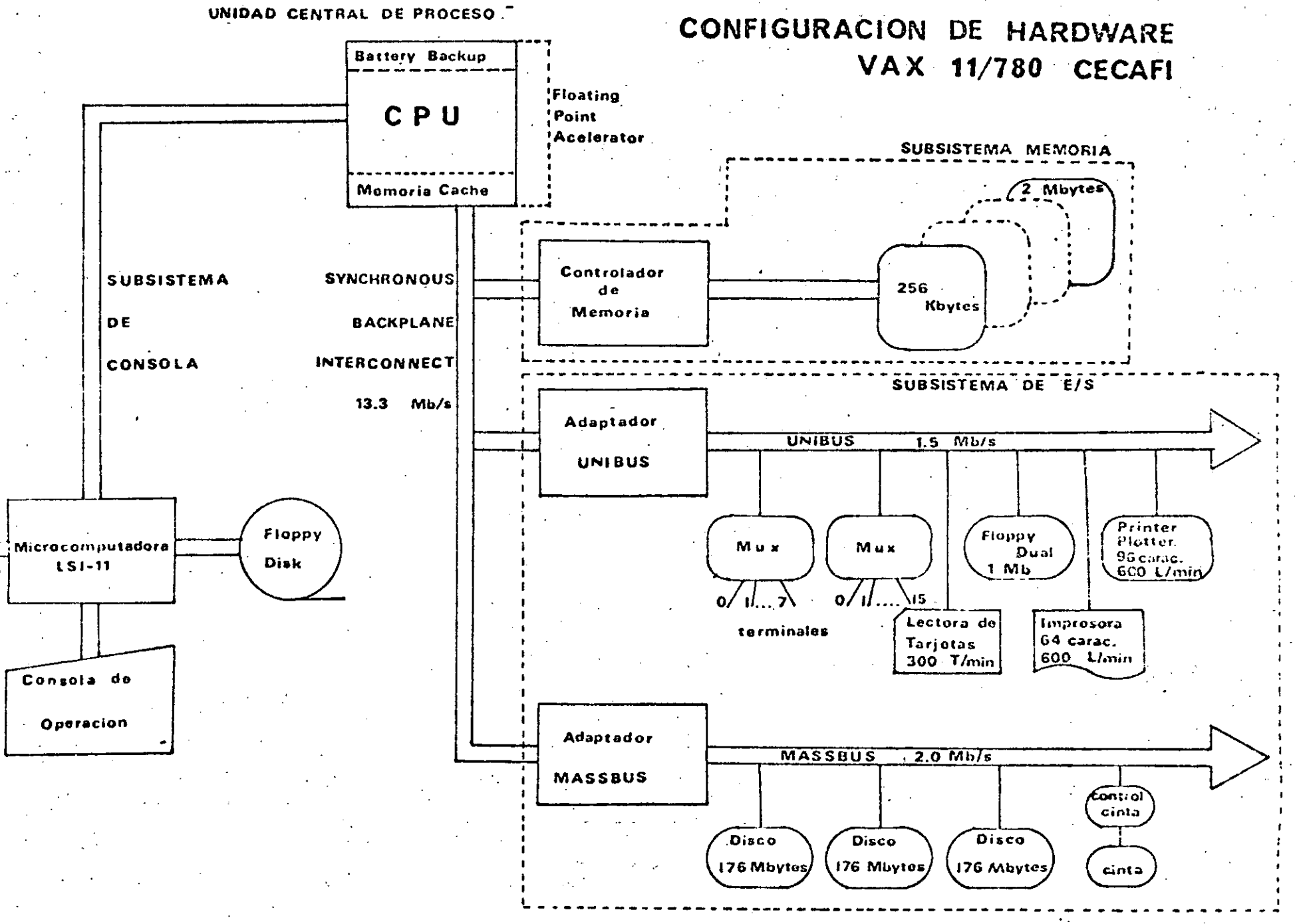
LENGUAJE DE PROGRAMACION BASIC II

A N E X O S

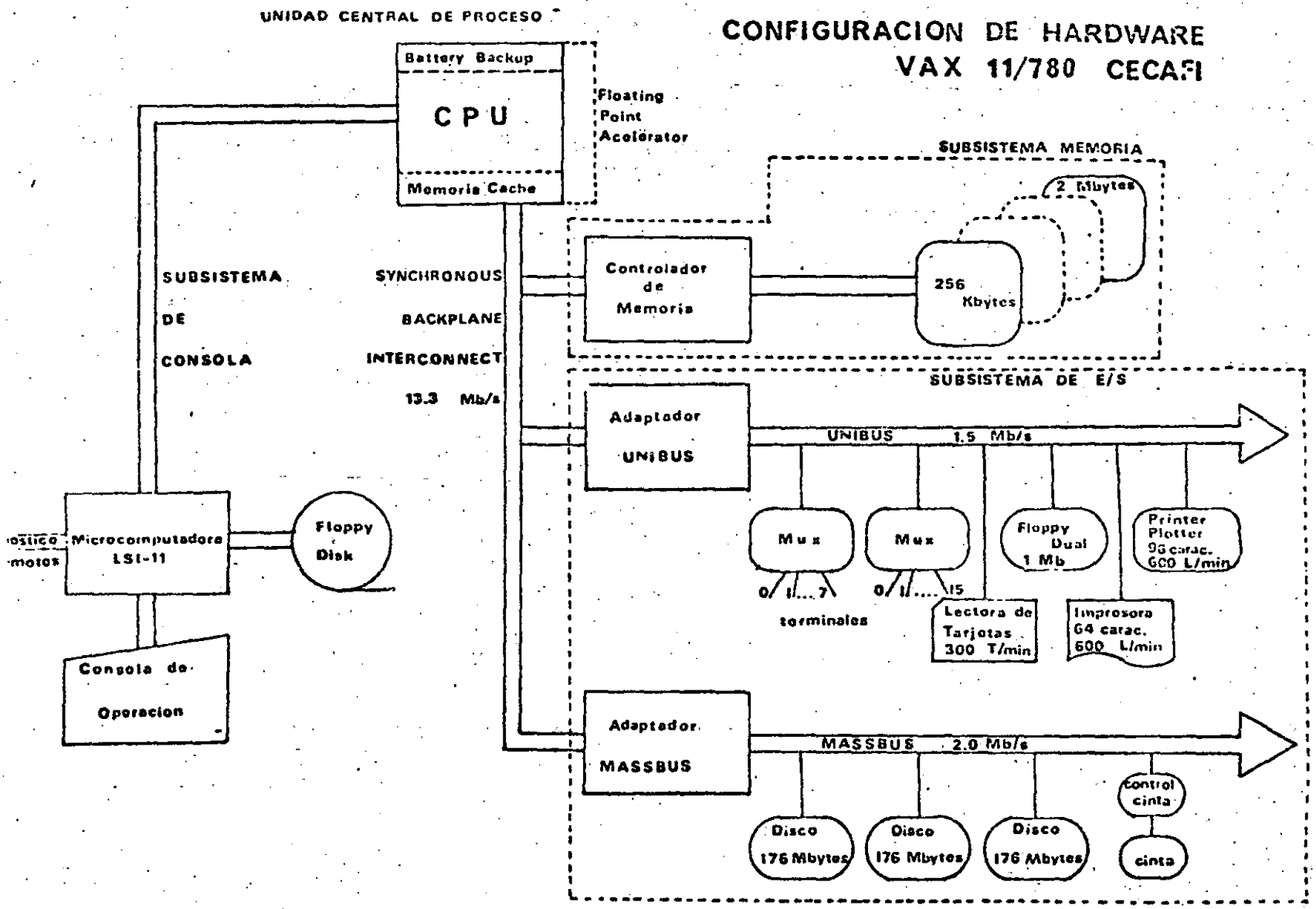
NOVIEMBRE DE 1985

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

CONFIGURACION DE HARDWARE
VAX 11/780 CECAFI



CONFIGURACION DE HARDWARE
VAX 11/780 CECAFI 1982



46

OBRA:
PROCESADORA DE AVES DE MORELOS 03/03/29
CALLE 21-ESTE CIVAC MORELOS
L I S T A D O D E M A T E R I A L E S
DE LA CLAVE:1-M001 , A LA CLAVE:1-M010

CLAVE	NOMBRE	COSTO	UNI.	ULTIMO CAMBIO	ULTIMO CALCULO	NOTAS
1-M001	AGUA	$ 135.00	M3	82/09/13	83/03/16	
1-M002	ACCESORIO DE BAÑO DE PORCELANA	$ 655.50	JGO.	83/03/15	83/03/16	
1-M003	ADOQUIN	$1.00	M2	82/06/14	83/03/16	
1-M004	ADOCRETO FIGURA DE CRUZ	$ 364.65	M2	82/09/11	83/03/16	
1-M005	ALAMBRE	$85.00	Kg	83/03/15	83/03/16	
1-M006	ALAMBRON	$75.00	Kg	83/03/15	83/03/16	
1-M007	AZULEJO LISO DE .11X.10	$ 748.00	M2	82/10/22	83/03/16	
1-M008	AZULEJO CON DIBUJO DE .11X.11	$1.00	M2	82/06/14	83/03/16	
1-M009	ARENA	$ 583.33	M3	83/02/03	83/03/16	
1-M010	BLOCK PESADO DE .20X.20X.40	$38.12	PZA.	83/03/15	83/03/16	

OBRA:
PROCESADORA DE AVES DE MORELOS 83/03/29
CALLE 21-ESTE CIVAC MORELOS
L I S T A D O D E M A T E R I A L E S
DE LA LETRA:A, A LA LETRA:A

CLAVE	NOMBRE	COSTO	UNI.	ULTIMO CAMBIO	ULTIMO CALCULO	NOTAS
1-M002	ACCESORIO DE BAÑO DE PORCELANA	$ 655.50	JGO.	83/03/15	83/03/16	
1-M136	ACEITE QUEMADO	$1.00	LT.	82/06/14	83/03/16	
1-M004	ADOCRETO FIGURA DE CRUZ	$ 364.65	M2	82/09/11	83/03/16	
1-M003	ADOQUIN	$1.00	M2	82/06/14	83/03/16	
1-M001	AGUA	$ 135.00	M3	82/09/13	83/03/16	
1-M005	ALAMBRE	$85.00	Kg	83/03/15	83/03/16	
1-M006	ALAMBRON	$75.00	Kg	83/03/15	83/03/16	
1-M169	ALCANTARILLA DE FIERRO .50X1.0	$3,500.00	MT.	83/03/15	83/03/16	
1-M145	ANILLOS	$1.00	PZA.	82/09/11	83/03/16	
1-M009	ARENA	$ 583.33	M3	83/02/03	83/03/16	
1-M008	AZULEJO CON DIBUJO DE .11X.11	$1.00	M2	82/06/14	83/03/16	
1-M007	AZULEJO LISO DE .11X.10	$ 748.00	M2	82/10/22	83/03/16	

OBRA:
PROCESADORA DE AVES DE MORELOS 83/03/29
CALLE 21-ESTE CIVAC MORELOS
L I S T A D O D E C O M P O N E N T E S
COMPLETO POR CAPITULO-PARTIDA-COMPONENTE

CAPITULO	PARTIDA	CLAVE COMPONENTE	POS. REL.	CANT. TOTAL EN LA OBRA	CANT. ACUM. AVANCES ANT.	CANT. AVAN. ACTUAL	CANT. ACUM. ANT. EN $	ULTIMO AVANCE
01	A	B-A001	00000	5050.00	0.00	0.00	0.00	00/00/00
01	A	B-A002	00000	2100.00	0.00	0.00	0.00	00/00/00
01	A	B-A003	00000	432.35	0.00	0.00	0.00	00/00/00
01	A	B-A004	00000	267.90	0.00	0.00	0.00	00/00/00
01	A	B-A005	00000	121.00	0.00	0.00	0.00	00/00/00
01	A	B-A007	00000	333.00	0.00	0.00	0.00	00/00/00
01	A	B-A017	00000	4270.00	0.00	0.00	0.00	00/00/00
01	A	B-A019	00000	65.50	0.00	0.00	0.00	00/00/00
01	A	B-A021	00000	1322.00	0.00	0.00	0.00	00/00/00
01	A	B-A022	00000	295.00	0.00	0.00	0.00	00/00/00
01	A	B-A023	00000	848.00	0.00	0.00	0.00	00/00/00
01	A	B-A026	00000	145.00	0.00	0.00	0.00	00/00/00
01	A	B-A027	00000	73.00	0.00	0.00	0.00	00/00/00

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
A N A L I S I S D E M A T E R I A L E S
COMPLETO POR CLAVE.

83/03/29

4.15

37

CLAVE	NOMBRE	COSTO DE COMPRA	o/o POR FLETE	IMPORTE FLETE	TOTAL
1-M001	AGUA	135.00	0.000	0.00	$ 135.00 P O R M3
1-M002	ACCESORIO DE BAÑO DE PORCELANA	655.50	0.000	0.00	$ 655.50 P O R JGO.
1-M003	ADOQUIN	1.00	0.000	0.00	$1.00 P O R M2
1-M004	ADOCRETO FIGURA DE CRUZ	364.65	0.000	0.00	$ 364.65 P O R M2
1-M005	ALAMBRE	85.00	0.000	0.00	$85.00 P O R KG
1-M006	ALAMBRON	75.00	0.000	0.00	$75.00 P O R Kg
1-M007	AZULEJO LISO DE .11X.10	748.00	0.000	0.00	$ 748.00 P O R M2
1-M008	AZULEJO CON DIBUJO DE .11X.11	1.00	0.000	0.00	$1.00 P O R M2
1-M009	ARENA	583.33	0.000	0.00	$ 583.33 P O R M3
1-M010	BLOCK PESADO DE .20X.20X.40	38.12	0.000	0.00	$38.12 P O R PZA.
1-M011	BLOCK LIGERO DE .20.X20.X40	43.47	0.000	0.00	$43.47 P O R PZA.
1-M012	CEMENTO GRIS NORMAL	4,000.00	0.000	0.00	$4,000.00 P O R TON

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
A N A L I S I S D E C A T E G O R I A S
DE LA CLAVE:6-03 , A LA CLAVE:6-03

83/03/29

40

CLAVE 6-03 NOMBRE: OFICIAL ALBAÑIL DE SEGUNDA

C A L C U L O

SALARIO BASE DIARIO =			485.00	
PERCEPCION ANUAL =	365.25	x	485.00	= 177,146.25
PRIMA VACACIONAL =	6.00	x	485.00	= 2,910.00
GRATIFICACION =	15.00	x	485.00	= 7,275.00
			TOTAL DEVENGADO =	187,331.25
I.M.S.S. =	0.0462500	x	187,331.25	= 8,664.07
EDUCACION =	0.0100000	x	187,331.25	= 1,873.31
GUARDERIAS =	0.0100000	x	177,146.25	= 1,771.46
INFONAVIT =	0.0100000	x	177,146.25	= 1,771.46
			SUMA ANUAL =	201,411.55

SALARIO REAL POR DIA LABORABLE=201,411.55 / 304.00 = $ 662.54

SALARIO REAL / SALARIO BASE = 1.3660

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
A N A L I S I S D E C U A D R I L L A S
DE LA CLAVE:7-G01 , A LA CLAVE:7-G01

83/03/29

42

CLAVE	NOMBRE / I N S U M O S	CANTIDAD UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
7-G1	ALBAÑIL DE SEGUNDA Y PEON						
6-01	PEON	1.0000 JOR.	1.000	490.61	490.61		
6-03	OFICIAL ALBAÑIL DE SEGUNDA	0.1000 JOR	1.000	662.54	66.25		
					SUMA :	$ 564.86	
				C O S T O D I R E C T O :		$ 564.86 P O R JOR	

CLAVE	NOMBRE INSUMOS	CANTIDAD	UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
3-2CON.S	CONCRETO DE F'C 200 PROP.1,2,2							
1-M001	AGUA	0.2050	M3	1.000	135.00	27.68		
1-M009	ARENA	0.4400	M3	1.020	583.33	261.80		
1-M012	CEMENTO GRIS NORMAL	0.3730	TON.	1.020	6000.00	2282.76		
1-M028	GRAVA TRITURADA DEL NO. 2 Y 3	0.6800	M3	1.000	1033.33	702.66		
						MATERIALES:	$3,274.90	
					COSTO DIRECTO:		$3,274.90	POR M3

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

83/03/29

ANALISIS DE CONCEPTOS
DE LA CLAVE:8-A074 , A LA CLAVE:8-A075

CLAVE	NOMBRE INSUMOS	CANTIDAD	UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
8-A074	TRABE DE NERVADURA DE 25X70 AR MADA CON 9 VAR. DE 5/8" EST. A 1/4" A CADA 25 CMS.							
1-M005	ALAMBRE	1.0000	Kg	1.000	85.00	85.00		
1-M006	ALAMBRON	1.9100	Kg	1.000	75.00	143.25		
1-M102	VARILLA DE 5/8 NORMAL	0.0140	TON.	1.000	55000.00	770.00		
1-M144	CIMBRA DE TRIPLAY MARINO	0.7000	M2.	1.000	350.00	245.00		
3-2CON.S	CONCRETO DE F'C 200 PROP.1,2,2	0.1750	M3	1.000	3274.90	573.11		
						MATERIALES:	$1,816.36	
7-02	ALBAÑIL DE SEGUNDA Y PEON	0.2000	JOR.	1.000	664.25	132.85		
7-02	ALBAÑIL DE SEGUNDA Y PEON	0.2000	JOR.	1.000	664.25	132.85		
7-03	OFICIAL CARPINTERO Y AY. CARP.	0.2000	JOR.	1.000	1420.70	284.14		
7-04	OFICIAL FIERRERO Y AY.FIERRERO	0.1650	JOR.	1.000	1461.68	241.18		
						MANO DE OBRA:	$ 791.02	
					COSTO DIRECTO:		$2,607.38	POR MT.
					INDIRECTOS Y UTILIDAD:		$ 730.07	.280 DEL COSTO D.

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS

83/03/29

PRESUPUESTO TOTAL DE LA OBRA
INCLUYENDO TODOS LOS COMPONENTES

COMPONENTE		CANTIDAD	UNI	PRECIO UNITARIO	IMPORTE	TOTALES CAPITULOS	NOTAS
01)ALBAÑILERIA							
A)ALBAÑILERIA							
8-A001	LIMPIEZA DE TERRENO	5,050.00	M2	17.01	85,900.50		
8-A002	TRAZO DE OBRA	2,100.00	M2	32.35	67,935.00		
8-A003	EXCAVACION EN TERRENO CON VOLEO A PICO Y PALA	432.35	M3	361.51	156,298.85		
8-A004	CIMENTACION DE MAMPOSTERIA COMUN DE PIEDRA BRAZA ASENTADA CON MORT.CEM.,CAL,ARENA 1:1:6	267.90	M3	2,765.21	740,799.76		
8-A005	BASES DE CASTILLOS DE .40X.40 ARMADO CON 4 VAR. DE 3/8 Y ESTRIBOS DE 1/4 A CADA 20 CMS.	121.00	ML	1,307.51	158,208.71		
8-A007	MURO DE TABIQUE ROJO RECOCIDO DE 14 CMS.DE ESPESOR ASENTADO CON MOR.CEM.CAL,ARENA 1:1:10	333.80	M2	662.37	221,099.11		
8-A017	APLANADO CERRADO DE MEZCLA CEMENTO, CAL,ARENA PROP. 1:1:8	4,270.00	M2	233.59	997,429.30		

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
83/03/29

A N A L I S I S D E M A Q U I N A R I A
DE LA CLAVE:4-01 , A LA CLAVE:4-01

4.16

CLAVE	NOMBRE I N S U M O S		CANTIDAD	UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
4-01	MAQUINA DE PRUEBA		MARCA: 1		MODELO: 2	CAPACIDAD: 3		POTENCIA EN HP: 0.00	
	VALOR DE ADQUISICION	VA=	5.00						
	VALOR DE RESCATE	VR=	0.30						
	TASA DE INTERES ANUAL	I=	9.00						
	o/o POR PRIMA DE SEGURO	S=	10.00						
	VIDA ECONOMICA EN ANOS	VE=	7.00						
	HORAS DE USO POR ANO	HA=	6.00						
	o/o POR ALMACENAJE	K=	0.00						
	o/o POR MANTENIMIENTO	R=	12.00						
	A) DEPRECIACION D=				(VA-VR)/(VE*HA)		0.08		
	B) INVERSION				(VA+VR)/(2*HA)*I/100		0.03		
	C) SEGUROS				(VA+VR)/(2*HA)*S/100		0.03		
	D) ALMACENAJE				D*K/100		0.00		
	E) MANTENIMIENTO				D*R/100		0.01		
						CARGOS FIJOS:		$0.15	
1-01			1.0000		1.000	0.00	0.00		insumo no existe
1-02			2.0000		1.000	0.00	0.00		insumo no existe
						CONSUMOS:		$0.00	
6-03	OFICIAL ALBANIL DE SEGUNDA		3.0000	JOR.	1.000	662.54	1987.62		
7-04			4.0000		1.000	0.00	0.00		insumo no existe
						OPERACION:		$1,987.62	
						MANDO INTERMEDIO:		$ 238.51	.120 DE LA OPER.
						C O S T O D I R E C T O:		$2,226.28	P O R HORA

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
83/03/29

A N A L I S I S D E E Q U I P O S
DE LA CLAVE:5-EA01 , A LA CLAVE:5-EA01

CLAVE	NOMBRE I N S U M O S	CANTIDAD	UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
5-EA01	PICO							
1-01		1.0000		1.000	0.00	0.00		insumo no existe
						MATERIALES:	$0.00	
4-3		0.1000		1.000	0.00	0.00		insumo no existe
						MAQUINARIA Y EQUIPO:	$0.00	
							$0.00	
						NUMERO DE USOS:	1	
						C O S T O D I R E C T O:	$0.00	P O R USO

ARQ. GUILLERMO SANCHEZ GARCIA.

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
83/03/29

A N A L I S I S D E M O R T E R O S
DE LA CLAVE:2-10 , A LA CLAVE:2-10

CLAVE	NOMBRE I N S U M O S	CANTIDAD	UNI.	FACTOR	COSTO	IMPORTE	TOTALES	NOTAS
2-10	MORTERO,CEMENTO,ARENA,1:1:6							
1-M001	AGUA	0.2790	M3	1.000	135.00	37.67		
1-M009	ARENA	1.1300	M3	1.020	583.33	672.35		
1-M012	CEMENTO GRIS NORMAL	0.3420	TON.	1.020	6000.00	1603.44		
1-M015	CAL HIDRA	0.1280	TON.	1.000	4500.00	576.00		
						MATERIALES:	$2,689.46	
						C O S T O D I R E C T O:	$2,089.46	P O R M3

CALLE 21-ESTE CIVAC MORELOS

PRESUPUESTO TOTAL DE LA OBRA FALTANTE POR EJECUTAR
INCLUYENDO TODOS LOS COMPONENTES

4.18

COMPONENTE	CANTIDAD	UNI	PRECIO UNITARIO	IMPORTE	TOTALES CAPITULOS	NOTAS
			SUMA 07.A :	$ 480,299.99		
				TOTAL 07 :	480,299.99	
08)INSTALACION HIDRAULICA						
A)INSTALACION HIDRAULICA						
9-19 INSTALACION HIDRAULICA	1.00	LOTE	1,454,572.42	1,454,572.42		
			SUMA 08.A :	$1,454,572.42		
				TOTAL 08 :	1,454,572.42	
09)INSTALACION ELECTRICA						
A)INSTALACION ELECTRICA						
9-20 INSTALACION ELECTRICA	1.00	LOTE	10,989,825.92	10,989,825.92		
			SUMA 09.A :	$10,989,825.92		
				TOTAL 09 :	10,989,825.92	
				GRAN TOTAL	$50,207,077.06	

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
EXPLOSION DE RECURSOS DEL TOTAL DE LA OBRA
INCLUYENDO TODOS LOS COMPONENTES

03/01/04

COMPONENTE		CANTIDAD	UNI	PRECIO UNITARIO	IMPORTE	TOTALES GRUPOS	NOTAS
MATERIALES							
1-M001	AGUA	419.05	M3	172.82	71,816.13		
1-M005	ALAMBRE	4,506.20	KG	108.80	490,274.56		
1-M006	ALAMBRON	3,251.33	KG	97.00	315,379.01		
1-M007	AZULEJO LISO DE .11X.10	164.30	M2	957.44	157,302.49		
1-M008	ARENA	1,201.17	M3	746.67	896,109.01		
1-M010	BLOCK PESADO DE .20X.20X.40	23,705.00	PZA.	48.77	1,156,002.52		
1-M011	BLOCK LIGERO DE .20.X20.X40	9,642.00	PZA.	55.64	480,851.68		
1-M012	CEMENTO GRIS NORMAL	526.74	TON.	7,600.00	4,002,342.40		
1-M014	CEMENTO BLANCO	4.09	TON.	20,608.00	84,237.66		
1-M015	GRAVILLA	51.59	TON.	5,760.00	297,168.83		
1-M020	GRAVA TRITURADA DEL NO. 2 Y 3	815.83	M3	1,322.66	1,083,030.03		
1-M061	PIEDRA BRAZA PARA CIMENTACION	497.93	M3	490.67	244,313.20		
1-M062	PIEDRA DE MINA	411.10	M3	746.67	307,014.40		
1-M070	TABIQUE ROJO REC.DE.05X.11X.24	37.95	MILL	3,120.00	194,280.50		
1-M073	TABIQUE DE PUEBLA	8.30	MILL	5,760.00	91,157.22		
1-M075	TUBO DE ALBAÑAL DE CONC.DE .15	147.40	PZA.	105.69	16,632.12		
1-M076	TUBO DE ALBAÑAL DE CONC.DE .20	146.30	PZA.	192.00	20,089.60		
1-M100	VARILLA DE 3/8 NORMAL	12.75	TON	70,400.00	897,585.29		
1-M101	VARILLA DE 1/2 NORMAL	17.56	TON.	70,400.00	1,236,377.12		
1-M102	VARILLA DE 5/8 NORMAL	1.72	TON.	70,400.00	121,164.03		
1-M103	VARILLA DE 3/4 NORMAL	1.99	TON.	70,400.00	140,427.44		
1-M141	CIMBRA DE MADERA	2,796.91	M2	192.00	537,026.48		
1-M143	CIMBRA DE TRIPLAY MARINO	212.94	M2.	440.00	93,693.12		
1-M145	CERO GRUESO	19.12	TON.	4,096.00	78,304.15		
1-M149	COLOR PARA CEMENTO	65.35	KG.	192.00	12,523.38		
1-M152	CERO FINO	6.41	TON.	4,096.00	26,250.97		
1-M155	TEPETATE	122.51	M3	300.00	37,200.54		
1-M156	TEZONTLE	137.92	M3	469.37	65,674.63		
1-M157	PISO DE CERAMICA STA.JULIA	570.54	M2	1,063.94	607,022.33		
1-M161	ZOCLO DE CERAMICA SANTA JULIA	1,015.00	PZA.	57.60	58,464.00		
1-M163	JUNTA DE DILAT. FEXPAN O SIM.	3,995.89	MT.	276.74	1,105,822.93		
1-M164	ELASTOFESI	977.39	KG.	32.40	31,667.27		
1-M169	ALCANTARILLA DE FIERRO .50X1.0	101.50	MT.	4,480.00	454,720.00		
1-M170	TUBO DE ALBAÑAL DE CONC.DE .30	82.50	PZA.	440.00	36,960.00		
1-M171	FESTEGRAL	53.30	KG.	76.80	4,093.44		
			TOTAL DE	MATERIALES		14,242,208.97	

MANO DE OBRA

OBRA:
PROCESADORA DE AVES DE MORELOS
CALLE 21-ESTE CIVAC MORELOS
EXPLOSION DE RECURSOS DEL TOTAL DE LA OBRA
INCLUYENDO TODOS LOS COMPONENTES

03/04/04

4.19

COMPONENTE		CANTIDAD	UNI	PRECIO UNITARIO	IMPORTE	TOTALES GRUPOS	NOTAS
9-01	EXCAVACION EN TERRENO TIPO II CON MAQUINA PARA DAR PISO DE DESPLANTE.	1.00	LOTE	795,200.00	795,200.00		
9-02	SACAR MATERIAL PRODUCTO DE EXCAVACION FUERA DE OBRA.	1.00	LOTE	784,896.00	784,896.00		
9-03	RELLENOS COMPACTADOS CON MAQUINA.	1.00	LOTE	463,523.84	463,523.84		
9-04	IMPERMEABILIZACION SOBRE LOSA	1.00	LOTE	451,543.04	451,543.04		
9-05	CONEXION A DRENAJE GENERAL	1.00	LOTE	29,440.00	29,440.00		
9-06	FALSO PLAFON DE TABLAROCA EN AREA CONEXOR.	80.00	LOTE	75,264.00	6,021,120.00		
9-07	TARQUEO CERRADORES DE PLUMAS Y CANGRE.	1.00	LOTE	97,280.00	97,280.00		
9-08	COLOCACION DE SOPORTES DE INSTALACIONES.	1.00	LOTE	73,920.00	73,920.00		
9-09	INSTALACION SANITARIA	1.00	LOTE	825,184.29	825,184.29		
9-10	EXCAVACION EN ROCA.	1.00	LOTE	1,317,367.94	1,317,367.94		
9-11	SACAR FUERA DE LA OBRA MATERIAL PRODUCTO DE EXCAVACION.	1.00	LOTE	299,801.60	299,801.60		
9-12	FORJAR CARCAMO DE SUCCION	1.00	LOTE	2,944.00	2,944.00		
9-13	FORJAR REGISTROS Y TAPA DE CISTERNA.	1.00	LOTE	4,409.32	4,409.32		
9-14	INSTALACIONES ESPECIALES	1.00	LOTE	264,749.99	264,749.99		
9-15	CARPINTERIA	1.00	LOTE	192,000.00	192,000.00		
9-16	VIDRIERIA	1.00	LOTE	349,331.99	349,331.99		
9-17	VIDRIERIA	1.00	LOTE	1,094,651.99	1,094,651.99		
9-18	VARIOS	1.00	LOTE	480,299.99	480,299.99		
9-19	INSTALACION HIDRAULICA	1.00	LOTE	1,454,572.42	1,454,572.42		
9-20	INSTALACION ELECTRICA	1.00	LOTE	10,989,825.92	10,989,825.92		
				TOTAL DE SUBCONTRATOS :		25,792,002.03	
				GRAN TOTAL		$50,207,009.28	

OBRA:
LIC. FERNANDO CASTELLANOS
C. RIO PANUCO COL. V. HERMOSA

82/12/09

LISTADO DE AVANCES
DE LA CLAVE:01A8-A001 , A LA CLAVE:01A8-A001

CAPITULO	PARTIDA	COMPONENTE	CANT. AVAN.
01	A	8-A001	20.00

OBRA:
LIC. FERNANDO CASTELLANOS
C. RIO PANUCO COL. V. HERMOSA
ESTIMACION TOTAL CORRESPONDIENTE AL ULTIMO AVANCE
INCLUYENDO TODOS LOS COMPONENTES

82/12/09

4.20

COMPONENTE	CANTIDAD	UNI	PRECIO UNITARIO	IMPORTE	TOTALES CAPITULOS	NOTAS
01)ALBAÑILERIA						
A)ALBAÑILERIA						
B-A001 LIMPIEZA DE TERRENO	20.00	M2	15.07	301.40		
SUMA 01.A				$ 301.40		
TOTAL 01					301.40	
GRAN TOTAL					$ 301.40	

OBRA:
LIC. FERNANDO CASTELLANOS
C. RIO PANUCO COL. V. HERMOSA

82/12/09

REPORTE TOTAL DEL ESTADO DE AVANCE DE LA OBRA
INCLUYENDO TODOS LOS COMPONENTES

	ACUMULADO ANTERIOR	EN EL ULTIMO AVANCE	ACUMULADO ACTUAL	PRESUPUESTO ACTUAL	FALTANTE POR EJECUTAR	NOTAS
01)ALBAÑILERIA						
A)ALBAÑILERIA						
B-A002 TRAZO DE OBRA						
EN UNIDADES(M2):	0.00	0.00	0.00	405.00	405.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$13,122.00	$13,122.00	
B-A003 EXCAVACION DE MATERIAL TEPETATOSO A PICO Y PALA						
EN UNIDADES(M3):	0.00	0.00	0.00	188.95	188.95	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$60,879.69	$60,879.69	
B-A004 CIMENTACION DE MAMPOSTERIA COMUN DE PIEDRA BRAZA ASENTADA CON MORT.CEM./CAL/ARENA 1:1:6						
EN UNIDADES(M3):	0.00	0.00	0.00	139.35	139.35	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$ 312,967.56	$ 312,967.56	
B-A005 BASES DE CASTILLOS DE .40X.40 ARMADO CON 4 VAR. DE 3/8 Y ESTRIBOS DE 1/4 A CADA 20 CMS.						
EN UNIDADES(ML):	0.00	0.00	0.00	47.00	47.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$45,053.73	$45,053.73	

OBRA:
LIC. FERNANDO CASTELLANOS
C. RIO PANUCO COL. V. HERMOSA

02/12/09

R E P O R T E T O T A L DEL ESTADO DE AVANCE DE LA OBRA
INCLUYENDO TODOS LOS COMPONENTES

4.21

	ACUMULADO ANTERIOR	EN EL ULTIMO AVANCE	ACUMULADO ACTUAL	PRESUPUESTO ACTUAL	FALTANTE POR EJECUTAR	NOTAS
EN UNIDADES(LOTE):	0.00	0.00	0.00	1.00	1.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$77,064.00	$77,064.00	
SUMA 07.A :	$0.00	$0.00	$0.00	$77,064.00	$77,064.00	
TOTAL 07 :	$0.00	$0.00	$0.00	$77,064.00	$77,064.00	
08)CERRAJERIA						
A)CERRAJERIA						
9-07 CERRAJERIA						
EN UNIDADES(LOTE):	0.00	0.00	0.00	1.00	1.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$28,960.00	$28,960.00	
SUMA 08.A :	$0.00	$0.00	$0.00	$28,960.00	$28,960.00	
TOTAL 08 :	$0.00	$0.00	$0.00	$28,960.00	$28,960.00	
09)YESERIA Y PINTURA						
A)YESERIA Y PINTURA						
9-08 YESERIA Y PINTURA						
EN UNIDADES(LOTE):	0.00	0.00	0.00	1.00	1.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$ 402,243.99	$ 402,243.99	
SUMA 09.A :	$0.00	$0.00	$0.00	$ 402,243.99	$ 402,243.99	
TOTAL 09 :	$0.00	$0.00	$0.00	$ 402,243.99	$ 402,243.99	
10)VARIOS						
A)VARIOS						
9-09 VARIOS						
EN UNIDADES(LOTE):	0.00	0.00	0.00	1.00	1.00	
EN $ (PRECIOS) :	$0.00	$0.00	$0.00	$ 114,080.00	$ 114,080.00	
SUMA 10.A :	$0.00	$0.00	$0.00	$ 114,080.00	$ 114,080.00	
TOTAL 10 :	$0.00	$0.00	$0.00	$ 114,080.00	$ 114,080.00	
GRAN TOTAL	$0.00	$ 301.40	$ 301.40	$6,402,802.87	$6,402,501.47	

M A R C O P L A N O
TARJETAS LEIDAS

11

** EJEMPLO BASIC PARTE II **

```
5 2 6 1 0
*PROP GEOM
1:4 0.001 0.00001
5:6 0.001 0.00004
#
*CONS ELAS
1:6 21000000 0.3
#
NO TERM
*INCIDENCIAS
1 1 2
2 2 3
3 3 4
4 4 5
5 6 1
6 7 5
*LON E INCLI
OPCION CERO
1 1 30
2 2 30
3 2 -30
4 1 -30
5:6 5 90
#
NO
NO
   **** UNICA    ****
1 TIPO
CLAVE 0
3 0 -10 0
#
```

MARCO PLANO
DATOS

** EJEMPLO BASIC PARTE II **

12

NUMERO DE NUDOS 5
NUMERO DE APOYOS 2
NUMERO DE BARRAS 6
NUMERO DE CONDICIONES DE CARGA INDEPENDIENTES. . 1
NUMERO DE CONDICIONES DE CARGA DEPENDIENTES. . . 0

PROPIEDADES DE LAS SECCIONES

BARRAS DE	A	AREA NORMAL	MOM. INERCIA CENTROIDAL	AREA EFECTIVA DE CORTANTE
1	4	0.0010	1.0000E-05	0.0008
5	6	0.0010	4.0000E-05	0.0008

CONSTANTES ELASTICAS

BARRAS DE	A	E	NU
1	6	21000000.000	0.300

INCIDENCIAS

BARRA	DE	A
1	1	2
2	2	3
3	3	4
4	4	5
5	6	1
6	7	5

COMPROBACION DE INCIDENCIAS

JUNTA	BARRA EN A	BARRA EN B
1	1	
1		5
2		1
2	2	
3		2
3	3	
4		3
4	4	
5		4
5		6
6	5	
7	6	

LONGITUDES E INCLINACIONES

BARRAS DE	A	LON	INCLI
1	1	1.000	30.00
2	2	2.000	30.00
3	3	2.000	-30.00
4	4	1.000	-30.00
5	6	5.000	90.00

CONDICIONES DE CARGA

1A.- **** UNICA ****

FUERZAS NODALES EN SISTEMA GLOBAL

NUDO	FX	FY	MZ
3	0.000	-10.000	0.000

ANCHO MAXIMO DE BANDA= 6

M A R C O P L A N O
RESULTADOS

** EJEMPLO BASIC PARTE II ** 13

**** UNICA ****

DESPLAZAMIENTOS DE LOS NUDOS

NUDO	DES-X	DES-Y	GIRO-Z
1	-0.0198	-0.0012	-0.0019
2	-0.0143	-0.0112	-0.0174
3	2.5732E-11	-0.0367	5.9987E-12
4	0.0143	-0.0112	0.0174
5	0.0198	-0.0012	0.0019

DESPLAZAMIENTOS DE LOS APOYOS

APOYO	DES-X	DES-Y	GIRO-Z
6	0.0000	0.0000	0.0000
7	0.0000	0.0000	0.0000

ELEMENTOS MECANICOS EN LAS BARRAS

BARRA	JUNTA	AXIAL	CORTANTE	MOMENTO
1	1	4.113	3.399	4.968
1	2	-4.113	-3.399	-1.570
2	2	4.113	3.399	1.570
2	3	-4.113	-3.399	5.228
3	3	4.113	-3.399	-5.228
3	4	-4.113	3.399	-1.570
4	4	4.113	-3.399	1.570
4	5	-4.113	3.399	-4.968
5	6	5.000	-1.863	-4.345
5	1	-5.000	1.863	-4.968
6	7	5.000	1.863	4.345
6	5	-5.000	-1.863	4.968

EQUILIBRIO DE LOS NUDOS

NUDO	F-X	F-Y	M-Z
1	0.0000	0.0000	0.0000
2	0.0000	-0.0000	0.0000
3	0.0000	-0.0000	0.0000
4	-0.0000	0.0000	0.0000
5	-0.0000	-0.0000	-0.0000

REACCIONES

APOYO	F-X	F-Y	M-Z
6	1.863	5.000	-4.345
7	-1.863	5.000	4.345