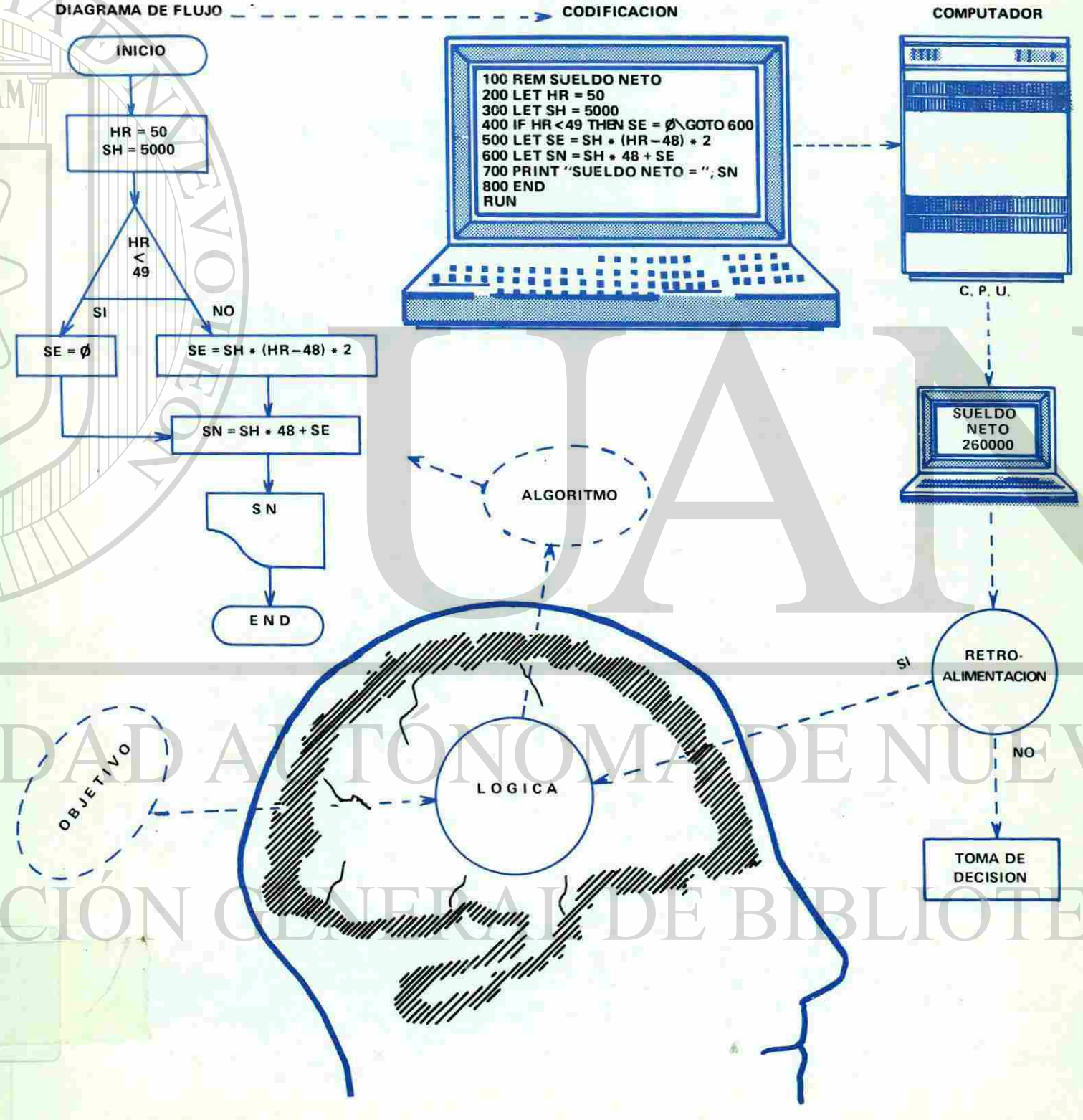
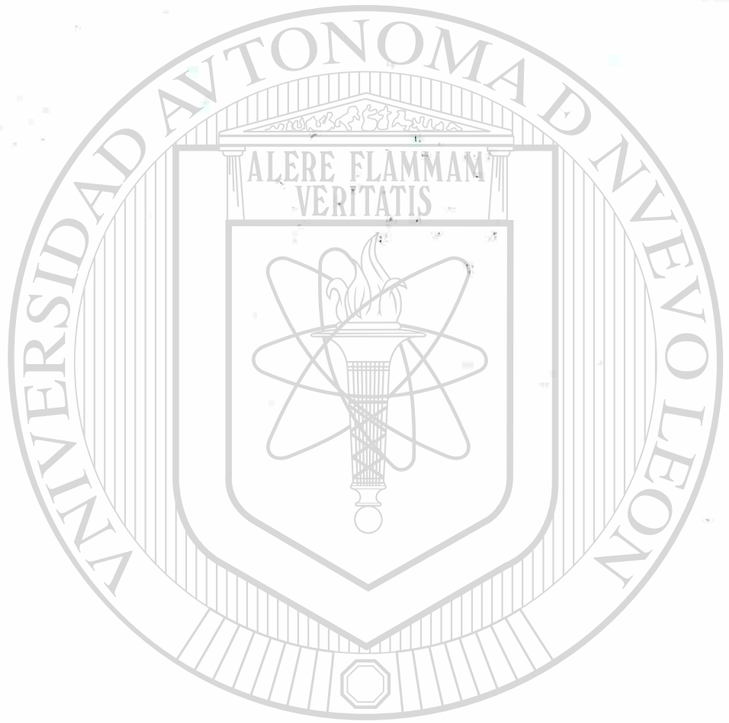
ING. HUGO SEGUNDO GONZALEZ GARCIA

PROGRAMACION BASIC





Nuestra institución se congratula hoy, al recibir esta aportación de uno de sus distinguidos catedráticos, el **Ing. Hugo Segundo González García.**

Este texto **PROGRAMACION BASIC**, *estamos seguros que será una ayuda eficaz, para nuestros estudiantes y para todo aquel que inicia en el estudio y manejo de los* **sistemas computacionales.**

Con toda obra humana es perfectible, pero el trabajo realizado por el maestro es invaluable. Sabemos de sus afanes y sus angustias por plasmar lo mejor de sus conocimientos y experiencia docente en este trabajo. A nombre de la **Facultad** *reconocemos su esfuerzo y lo felicitamos.*

Esta es una nueva publicación de nuestro **Centro de Investigaciones y Apoyo a la Docencia**, *que sigue adelante en la tarea de lograr que nuestros profesores trasciendan el aula y a través de textos como el presente, transmitan su mensaje a un mayor número de estudiantes.*

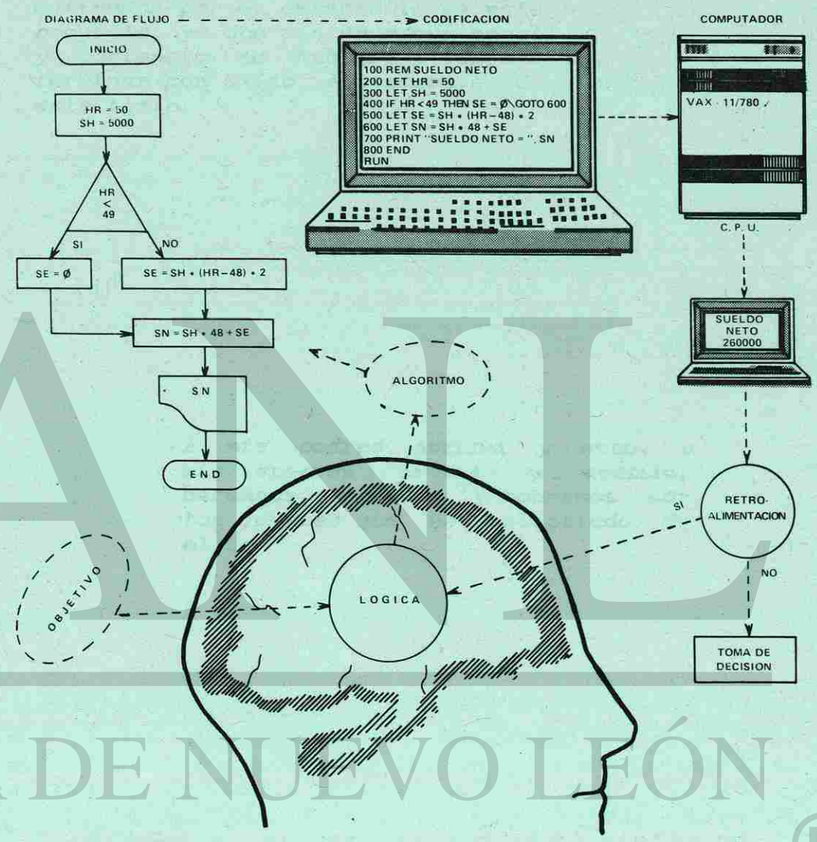
Damos pues, la bienvenida a este trabajo y esperamos la crítica constructiva que nos ayude a mejorarlo, para un óptico aprovechamiento académico.

EL DIRECTOR

C.P. GUMERSINDO CANTU HINOJOSA.

PROGRAMACION BASIC

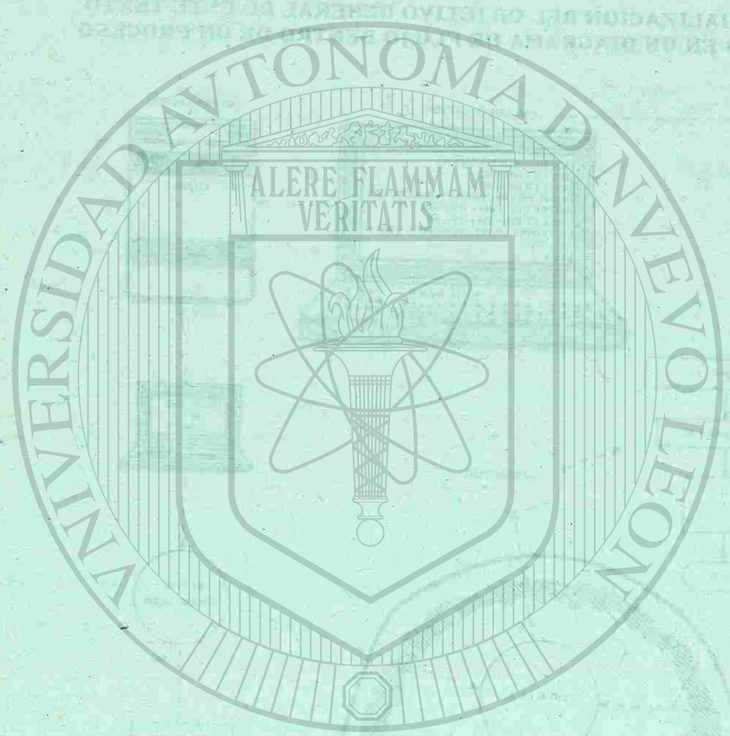
CONCEPTUALIZACION DEL OBJETIVO GENERAL DE ESTE TEXTO ESTRUCTURADO EN UN DIAGRAMA DE FLUJO DENTRO DE UN PROCESO



Nota del autor

El presente diagrama pretende que el alumno se conceptualice desde el inicio en el uso de este material, y observe la utilidad que se obtendrá en el manejo e interpretación del mismo dentro de su propio proceso de aprendizaje.

ING. HUGO SEGUNDO GONZALEZ GARCIA



A mi esposa **ALICIA** *y a mis hijos* **ELIZABETH** y **HUGO**, *que han soportado mis ausencias y días difíciles motivadas por el desarrollo de este proyecto, ya que sin su apoyo moral y paciencia no hubiera logrado terminar con éxito el escrito de este texto.*

A mis padres **ADELINA** *y* **HUGO**, *a mis suegros* **ANGELA** *y* **ROGELIO**, **hermanos, cuñados** *y* **sobrinos** *por los favores que he solicitado de ellos.*

A los **alumnos** *de FACPYA, porque me inspiraron y motivaron para la realización del texto.*

A **F.A.C.P.Y.A. U.A.N.L.**

RECONOCIMIENTOS

Tengo la oportunidad de agradecer a una persona que para mí, ha sido un guía en lo referente a la docencia, de la cual he aprendido mucho de sus experiencias, conocimientos y talento, puesto que dedica mucho de su tiempo a la capacitación de maestros de FACPYA, para el mejor desempeño de una cátedra, redundando ésto en una mayor eficiencia de los alumnos en su preparación curricular.

Gracias por sus consejos, motivación y asesoria didáctica para la presentación actual de este texto.

Con todo respeto para la **Lic. y Mtra. Belen Dalia Escamilla de la Cerda, Coordinadora de Apoyo a la Docencia.**

Un agradecimiento muy especial para **Juan E. Saldaña Pérez,** *por su motivación y trabajo incesante, que alimentó todo el texto al computador y aportó ideas valiosas para el diseño de la tipografía, fué una bonita experiencia la que compartimos con los muchos problemas a los que nos enfrentamos, pero al final logramos nuestros propósitos y objetivos.*

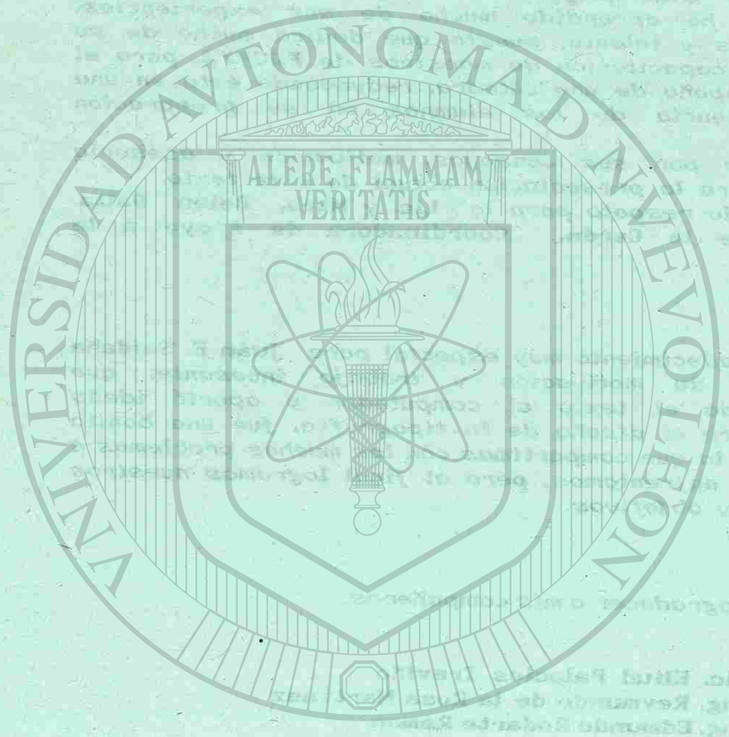
Deseo agradecer a mis compañeros.

Lic. Eliud Palacios Treviño
Ing. Reymundo de la Rosa Martínez
Ing. Edmundo Rodarte Ramón
Lic. José Ramón Rodríguez Garza
Lic. Patricia del Carmen Elguezabal López
Lic. Raúl S. Montoya Retta
Lic. Joel Mendoza Gómez

Por permitirme su tiempo para la revisión y asesoría en algunos puntos importantes de este manuscrito.

A mis amigos por su participación desinteresada

America García Sanchez
Katia Site Pérez Martínez
Sr. Jesús Licón Pérez
Jaime A. Pacheco Peña
Jaime E. Montemayor Elizondo
Fernando Ocañas Carreón



CONTENIDO

PROLOGO

Este texto se escribió con el objeto de que los estudiantes interesados en desarrollar sus habilidades en programación, les proporcione los elementos esenciales en el lenguaje que más se enseña y que su uso es el más extendido debido a que es fácil de aprender por su versatilidad, por la facilidad de aprendizaje de su sintaxis y de las técnicas de programación como es el **BASIC**.

El autor de esta obra, catedrático de la **Facultad de Contaduria Pública y Administración de la U.A.N.L.,** *preocupado por la variedad con que se presenta la programación* **BASIC** *en diversos textos y por su interés en lograr que el contenido de dicha obra pretenda ser eminentemente práctica, aunque sin abandonar las bases teóricas ha tenido la oportunidad de tomar en cuenta sus experiencias, así como las inquietudes de alumnos y maestros que le sugirieron que no existía una verdadera estandarización en la impartición de dicha materia en las aulas de estudio.*

El autor del texto, **Ing. Hugo Segundo González García** *egresado de la* **Facultad de Ingeniería Mecánica y Eléctrica de la U.A.N.L** *quien curso la carrera de* **Ingeniería en Electrónica y Comunicaciones** *durante el período 1979-1981, tiene una vasta experiencia en el campo educativo ya que su mayor anhelo ha sido la impartición de cátedra como lo ha estado desarrollando desde febrero de 1980 a la fecha en la* **Facultad de Contaduría Pública y Administración de la U.A.N.L.**

Solo me resta decir que el texto que tiene en sus manos ha sido elaborado muy cuidadosamente teniendo especial atención en su contenido y pretende cumplir con los objetivos propuestos para mejorar y elevar la calidad de la educación, precepto que todo catedrático debiera contemplar para el bien del estudiantado en general.

Enhorabuena por la acertada aportación del **Ing. Hugo Segundo González García,** *por el apoyo que brinda en esta materia a la cual aporta sus vastos conocimientos y experiencias y en lo cual considero a dicho catedrático como un experto en el lenguaje* **BASIC**.

LIC. ELIUD PALACIOS TREVIÑO

INTRODUCCION

El objetivo primordial del texto, es enseñar al lector la secuencia del procedimiento para resolver un problema utilizando tecnología computacional y sobre todo conocer y utilizar el lenguaje de programación **BASIC**.

Este texto, lo llevará poco a poco al entendimiento de cada tema, en los cuales se describirán en forma bastante sencilla los conceptos principales y la utilización del equipo, así mismo se describe lo más importante para manejar una terminal conectada a un computador.

Además se presenta un capítulo en el cual se muestra el procedimiento de como resolver un problema, realizando algoritmos y empleando los símbolos correspondientes para la elaboración del diagrama de flujo.

Se estudiarán los componentes de un programa como tipos de variables y operadores necesarios.

En lo referente a estatutos de **BASIC** *por ejemplo :* **REM, LET, GOTO, IF-THEN-ELSE, INPUT, PRINT, FOR-NEXT, DIM** *y otros, se le mostrará los formatos que necesita para la codificación, y la ejecución de estos para su mejor comprensión.*

En el lenguaje **BASIC**, *se pueden utilizar funciones como :* **SEN, COS, TAN, SGN, INT, LEFT, RIGHT, MID,** *y otros que se manejan en trabajos especiales.*

Para el estudio y mejor compresión de cada uno de los temas se complementa con ejercicios de aprendizaje que a su vez servirán para la práctica en el laboratorio.

Al final de cada capítulo, encontrará una **AOTOEVALUACION** *que le servirá como retroalimentación para que usted mismo se de cuenta de sus aciertos y errores que le permitan eficientar su aprendizaje en la aprehensión del conocimiento en esta materia.*

ING. HUGO SEGUNDO GONZALEZ GARCIA

CAPITULO 1

GENERALIDADES

OBJETIVO GENERAL

Introducir al estudiante en la historia del desarrollo del lenguaje de programación **BASIC.**

OBJETIVO ESPECIFICO

Cuando estamos practicando o desarrollando un sistema de información y éste es programado por algún lenguaje de programación (**BASIC, COBOL, FORTRAN,** *etc.), la persona que lo está utilizando, muchas veces desconoce realmente el significado y/o el área hacia donde está enfocado el lenguaje, así como de las personas que lo diseñaron.*

Por consiguiente, el alumno conocerá y comprenderá las ventajas y/o desventajas que el lenguaje de programación **BASIC** *puede proporcionarle, para que el uso de esta tecnología le sea realmente útil.*

1.1 QUE ES BASIC ?

Es un lenguaje de programación de alto nivel, que nos permite interactuar directamente con el computador, es decir, nos proporciona una respuesta inmediata. La programación **BASIC** *es muy fácil de aprender, puesto que la persona que lo estudia en pocas sesiones podrá programar.*

QUE SIGNIFICA BASIC ?

Beginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode.

Código simbólico de propósito general para la enseñanza a principiantes.

1.2 COMO SE INICIO BASIC ?

El lenguaje de programación BASIC, fué creado a fines de la década de los 60 por el DR. THOMAS KURTZ y el DR. JHON KEMENY del DARTMOUTH UNIVERSITY. Sus estudios los enfocarón al desarrollo de un lenguaje orientado al área educativa, con el fin de que los estudiantes o principiantes no tuvieran muchos problemas en programar una computadora. En la actualidad, **BASIC** *puede ser utilizado en distintas áreas como son : los negocios, la investigación, la ciencia, etc., y ésto se debe a la gran flexibilidad que poseen sus instrucciones.*

BASIC *puede trabajar en cualquier tamaño de computadora (microcomputadoras, minicomputadoras y macrocomputadoras) y solamente necesitamos hacer pequeños ajustes en alguna instrucción.*

1.3 VENTAJAS DEL LENGUAJE BASIC

1.- *Es fácil de aprender.*

2.- *La codificación es muy sencilla. Solamente como regla en cada instrucción, deberá estar asignada con un número de línea y en orden ascendente.*

3.- *BASIC en las áreas de matemáticas y científica, es muy valioso, puesto que cuenta con muchas funciones internas y una excelente capacidad para resolver problemas matemáticos.*

4.- *Usa funciones para el álgebra matricial.*

5.- *Permite manejar aplicaciones administrativas.*

6.- *No es necesario que la programación cumpla con el requisito de estar estructurada.*

1.4 DESVENTAJAS DEL LENGUAJE BASIC

1.- *No incluye en su juego de instrucciones la orden para clasificar (sortear).*

1.5 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Explique qué es* **BASIC** *?*

_____
_____
_____

2.- *Escriba el significado de* **BASIC** *?*

INGLES B_____A_____S_____I_____C_____

ESPAÑOL _____
_____

3.- *En qué década y por quién fué creado la programación* **BASIC** *?*

_____
_____
_____

4.- *Con qué fin fué creado el lenguaje* **BASIC** *y hacia donde está orientado ?*

_____
_____
_____

5.- *Escriba las áreas donde el lenguaje* **BASIC** *puede ser utilizado.*

_____
_____
_____

6.- *Escriba 4 ventajas que nos proporciona la programación* **BASIC.**

1 _____

2 _____

3 _____

4 _____

7.- *Cuál es la desventaja del lenguaje* **BASIC** *?*

_____
_____
_____

CAPITULO 2

ESTUDIO DE LA TERMINAL

OBJETIVO GENERAL

Capacitar al estudiante en el manejo y dominio de la terminal, además, introducirlo en el uso del **interpretador BASIC.**

OBJETIVO ESPECIFICO

Introducir al alumno en el conocimiento de la ubicación de las teclas más importantes de una terminal, también se le enseña los pasos que tendrá que realizar para poder iniciar o terminar de trabajar con el computador.

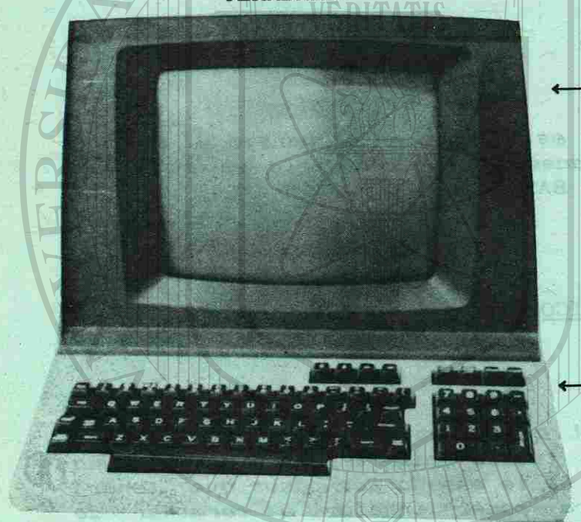
Uno de los temas más importantes, es además, el aprender el entendimiento de los comandos del **interpretador BASIC,** *pues con ellos podrá manejar con mucha facilidad los programas que llegue a realizar.*

2.1 QUE ES UNA TERMINAL ?

Es un dispositivo de ENTRADA/SALIDA (input/output) que está conectada al computador central para la transmisión y recepción de información.

La terminal tiene una pantalla como los cinescopios de una televisión, en donde vamos a observar la información que alimentamos o recibimos del computador, también cuenta con un teclado que se utiliza para dar entrada de información.

TERMINAL

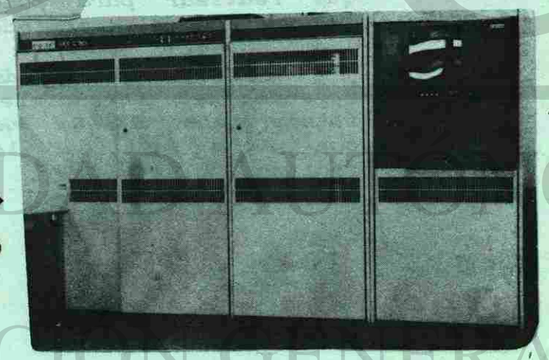


← **PANTALLA**

← **TECLADO**

↑

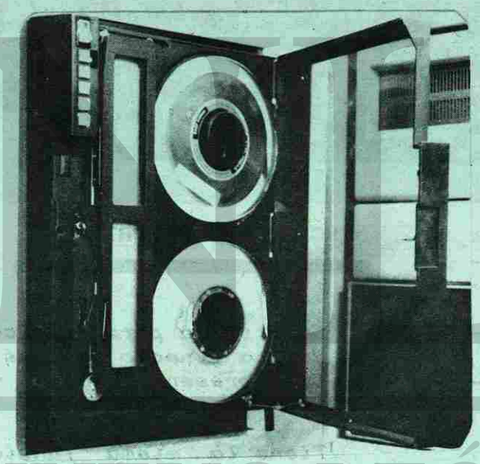
(INPUT-OUTPUT)



COMPUTADOR (C. P. U.)
UNIDAD CENTRAL DE PROCESAMIENTO

UNIDADES DE DISCOS MAGNETICOS

UNIDAD DE CINTA MAGNETICA



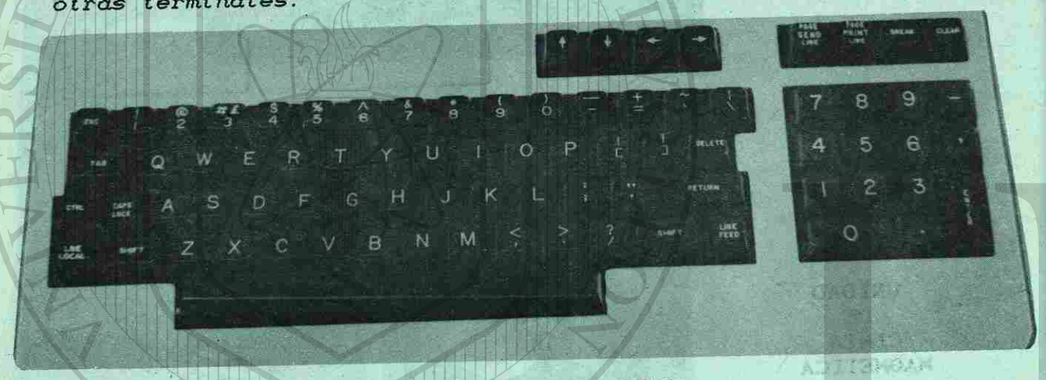
Los dátos de entrada van alimentarse por medio de una terminal y pasarán por el computador para ser procesados, los resultados obtenidos pueden observarse en la misma terminal o pueden grabarse directamente a una unidad externa como los discos o cintas, por lo regular van a disco y después se obtiene un respaldo de información a cinta si se desea.

La cinta magnética se utiliza como respaldo de la información que se encuentra almacenada en disco.

2.2 TECLADO DE LA TERMINAL

En este punto, vamos a ver las teclas más utilizadas para introducir un programa **BASIC** *al computador, algunas de ellas no se utilizan, pero no por ésto no son importantes, ésto se debe a que podemos alimentar al computador un programa de distintas maneras, por los diferentes editores con que cuenta el sistema operativo del computador.*

En la siguiente figura, se muestra el teclado de una terminal MIME-340, que puede ser semejante al utilizado por otras terminales.



TECLAS MAS IMPORTANTES

RETURN

Se utiliza para indicar la terminación de una línea o respuesta.
Se representará como **< CR >**.

ENTER

Tiene la misma función que la tecla **RETURN.** *Se representará como* **< CR >**.

DELETE

Sirve para borrar caracteres que se han escrito mal.

TAB

Es un tabulador de espacios, al oprimirlo el cursor salta 8 de ellos.

SHIFT

Es utilizado para escribir los caracteres superiores de las teclas. Se oprime la tecla **SHIFT** *y la tecla deseada simultáneamente.*

Ejemplo :

SHIFT y **#** / **3**

desplegará en pantalla el signo numérico (#).

CAPS LOCK

Es una tecla candado para escribir los caracteres en mayúsculas y minúsculas. Si se oprime la tecla se activa para mayúsculas, si no está activado será para minúsculas.

LINE LOCAL

LINE.- Significa que se encuentra en línea con el computador. Es decir, que podremos trabajar con la máquina.

LOCAL.- No se encuentra conectada al computador.

Si se encuentra oprimida la tecla estará en estado local, de lo contrario se encontrará en línea.

LINE FEED

Esta tecla sirve para dejar una línea en blanco.

CTRL

Es una tecla que al combinarse con otra genera una función especial.

LAS SIGUIENTES SON ALGUNAS DE SUS FUNCIONES :

Las teclas deben oprimirse simultáneamente.

CTRL **S** — *Sirve para detener cualquier ejecución o trabajo que esté realizando.*

CTRL **Q** — *Continúa con la ejecución o trabajo que estaba realizando antes de haberse detenido por el* **CTRL / S.**

CTRL **C** — *Interrumpe cualquier proceso que esté ejecutando.*

CTRL **R** — *Sirve para repetir la última línea que se esté escribiendo.*

CTRL **Y** — *Interrumpe cualquier proceso que se esté ejecutando, no es recomendable utilizarlo dentro del* **interpretador BASIC,** *puesto que se puede perder el programa que estamos realizando.*

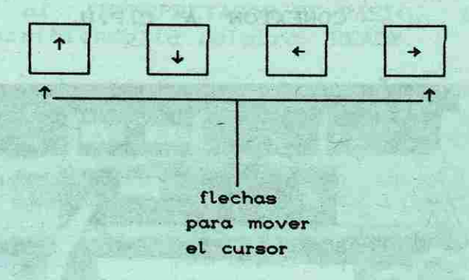
El **INTERPRETADOR BASIC,** *es para que el usuario esté interactuando directamente con el computador.*

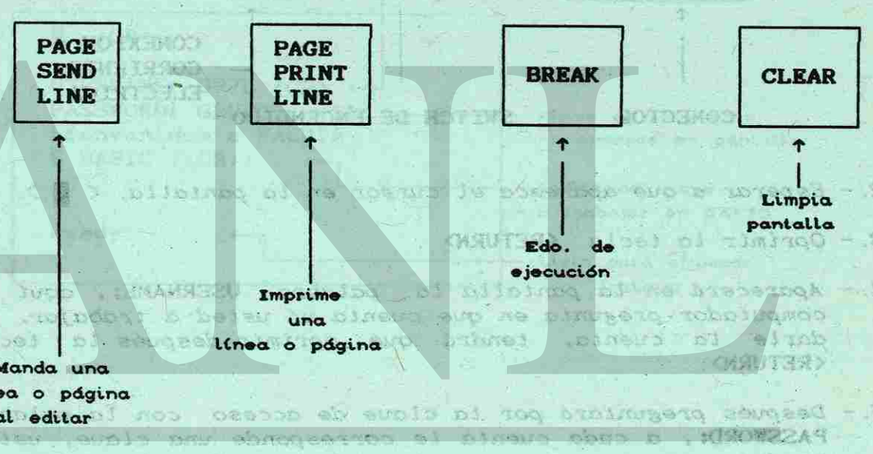
CTRL **I** — *Tiene la misma función que la tecla* **TAB.** *deja 8 espacios en blanco.*

CTRL **U** — *Sirve para borrar la última línea que se está escribiendo, no la borra de la pantalla.*

Las siguientes teclas no se utilizan cuando estemos alimentando un programa al computador, puesto que tienen otra función que puede perjudicar el programa que estamos realizando.

A continuación las presentamos.

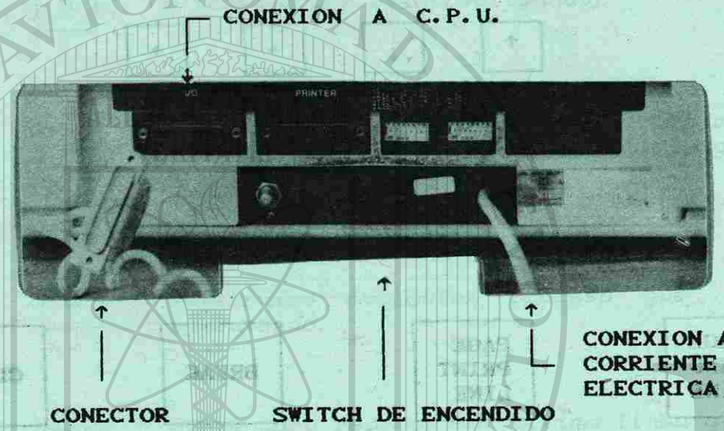




Si observan su terminal, en el lado derecho se encuentran unas teclas identificando los números, éstas sí las podrán utilizar, ya que es muy práctico para cuando se desea capturar información numérica.

2.3 PASOS PARA ENTRAR AL SISTEMA VAX-11/780

- *Encender switch. Se encuentra en la parte posterior de la terminal.*



2.- *Esperar a que aparezca el cursor en la pantalla.* (■).

3.- *Oprimir la tecla* **<RETURN>** .

4.- *Aparecerá en la pantalla la palabra* **USERNAME:**, *aquí el computador pregunta en que cuenta vá usted a trabajar. Al darle la cuenta, tendrá que oprimir después la tecla* **<RETURN>**.

5.- *Después preguntará por la clave de acceso con la palabra* **PASSWORD:**, *a cada cuenta le corresponde una clave, usted tendrá que conocer su clave de entrada, al teclear la clave oprima al final la tecla* **<RETURN>**.

Nota 1.- En caso de haber dado mal la cuenta o clave de acceso, tendrá que repetir los pasos desde el número 3.

Nota 2.- Al escribir el <PASSWORD>, éste no aparecerá en la pantalla.

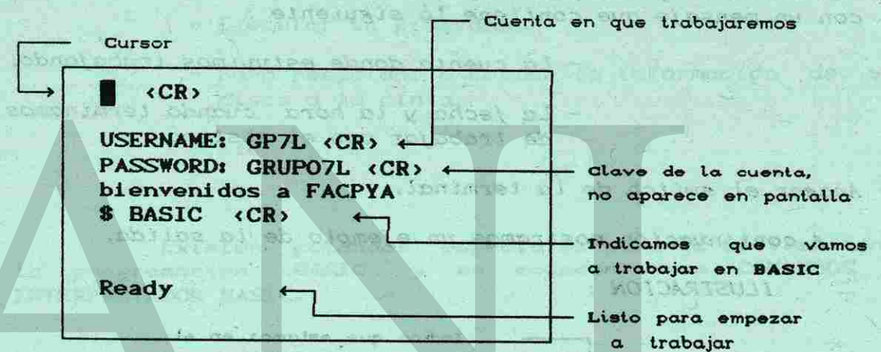
Nota 3.- Si escribió correctamente la cuenta y la clave, el computador le contestará con mensaje y le desplegará un signo de $.

Usted aquí, ya puede trabajar con los recursos del computador, puede pedir el directorio de programas que están almacenados, elaborar un programa en cualquier tipo de lenguaje, etc. Para trabajar con el **INTERPRETADOR BASIC,** *tendremos que hacer un sexto y último paso.*

6.- *Teniendo el signo de* **$**, *vamos a teclear la palabra* **BASIC** *o simplemente* **BAS** *y oprimimos* **<RETURN>**. *Así entramos directamente al* **INTERPRETADOR DE BASIC,** *la máquina le contestará escribiendo la palabra* **READY.**

A continuación mostramos un ejemplo de la entrada.

ILUSTRACION :



2.4 PASOS PARA SALIR DEL SISTEMA VAX-11/780

Cuando desea pedir al computador realizar cualquier operación, ella la ejecutará y al final le contestará con la palabra **READY** *(listo), indicándole que terminó de PROCESAR el trabajo. Por lo tanto, cuando quiera terminar de trabajar con el computador vamos a tener la palabra* **READY** *al final y tendremos que realizar los siguientes pasos :*

1.- *Escribir la palabra* **EXIT**, *que significa salida del interpretador* **BASIC** *y oprimir* **RETURN**. *Al dar este paso la máquina nos contestará con un signo de* **$**.

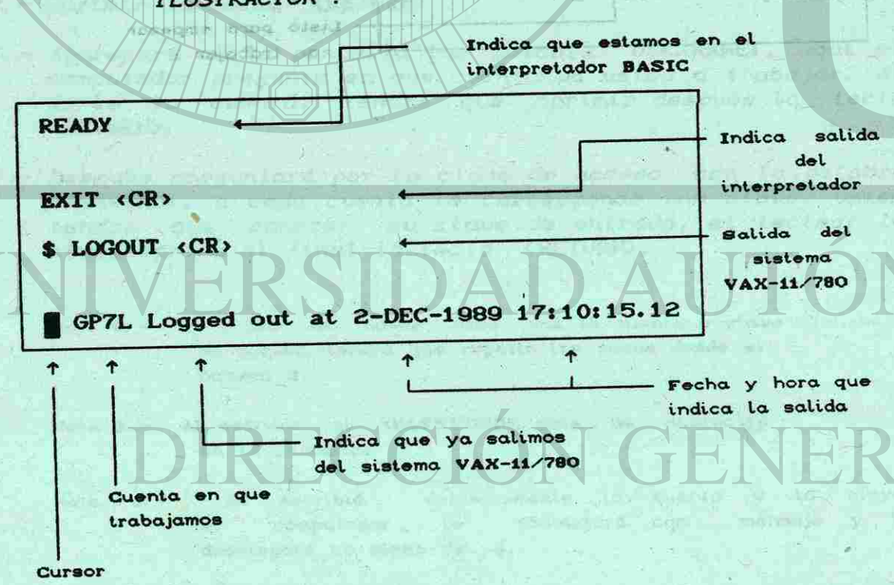
2.- *Teniendo el signo de* **$**, *vamos a escribir la palabra* **LOGOUT** *o* **LO** ; *con ésto le indico que quiero salir del sistema* **VAX-11/780**, *y después oprima* **<RETURN>**, *nos contestará con un mensaje que contiene lo siguiente :*

- *La cuenta donde estuvimos trabajando.*
- *La fecha y la hora cuando terminamos de trabajar con el sistema.*

3.- *Apagar el switch de la terminal.*

A continuación mostramos un ejemplo de la salida.

ILUSTRACION :



2.5 COMANDOS DEL INTERPRETADOR BASIC

Un comando es una instrucción de un lenguaje de control que sirve para realizar un trabajo específico de un computador.

Por ejemplo :

- *Cuando deseamos trabajar un compilador.*
- *Para poder crear un directorio o subdirectorio.*
- *Encadenar o enlazar un programa objeto.*
- *Ejecutar un programa.*
- *Para respaldar o copiar la información de un disco a la cinta.*
- *Etc.*

Existen comandos especiales para la utilización en la programación **BASIC** *y se conocen como* **COMANDOS DEL INTERPRETADOR BASIC.**

Siempre que utilicemos un comando **BASIC**, *la computadora nos contestará con la palabra* **READY**, *con ésto nos indica que el trabajo que solicitamos ya fué realizado.*

A continuación, estudiaremos los **COMANDOS** *que más se utilizarán para trabajar un programa.*

Recuerde, primero debemos estar dentro del **SISTEMA** *y realizar los pasos necesarios hasta que la computadora nos conteste* **READY**, *también se representará la tecla* **RETURN** *como* **<CR>**.

NEW .- *Comando que utilizaremos para asignarle nombre a nuestro programa.*

Existen dos formatos para la asignación del nombre :

Formato # 1

```
NEW 8T363465A  <CR>
Ready
```

Formato # 2

```
NEW                <CR>
New file name- - 8T363465A  <CR>
Ready
```

Donde :

8T363465A .- *Es el nombre del programa.*
- **8T** .- *Indica el grupo.*
- **363465** .- *Es la matrícula*
- **A** .- *Letra consecutiva*

Observación :

1.- El nombre del programa, puede contener hasta un máximo de 9 **CARACTERES.**

2.- En caso de no asignarle nombre al programa, el computador lo reconocerá con el nombre de **NONAME.**

3.- Si el nombre se **EXCEDE** de 9 caracteres, el computador le contestará con un mensaje de **ERROR.** El mensaje es el siguiente :

% BASIC-E-BADPROGNM, error in program name.

LIST .- *El comando* **LIST**, *sirve para listar una o varias instrucciones del programa.*

Formas de utilizar el comando **LIST** :

LIST ‹CR› .- *Lista* **todas** *las instrucciones del programa.*

LIST 100-150 ‹CR› .- *Lista las instrucciones de la* **100** *a la* **150.**

LIST 100,200,10 ‹CR›.- *Lista las instrucciones* **100, 200** *y la* **10.**

LIST 500 ‹CR› .- *Lista la instrucción* **500.**

En el comando **LIST**, *se puede omitir la letra "***T***", o sea que podemos escribir solamente* **LIS**, *por la facilidad que nos brinda el compilador de* **BASIC.**

Cada vez que utilicemos cualquiera de las formas anteriores, la máquina desplegará en la pantalla el nombre del programa, la fecha y la hora actual.

Ejemplo :

```
NEW 8T363465A  <CR>
Ready

10 LET A = 5                              <CR>
20 LET B = 6                              <CR>
30 LET C = A + B                          <CR>
40 PRINT "EL RESULTADO ES ";C             <CR>
50 END                                    <CR>

LIST 30   <CR>                    ←—— Lista la so
8T363465A      02-DIC-1984  17:42

30 LET C = A + B

Ready

LIST   <CR>                       ←—— Lista  todo

8T363465A      02-DIC-1984  17:45

10 LET A = 5
20 LET B = 6
30 LET C = A + B
40 PRINT "EL RESULTADO ES ";C
50 END

Ready
```

DELETE .- *Sirve para borrar una o varias instrucciones.*

Existen varias formas de utilizar el **DELETE** :

DELETE ‹CR›	.- *Borra* **todas** *las instrucciones.*
DELETE 100-500 ‹CR›	.- *Borra las instrucciones desde la* **100** *a la* **150.**
DELETE 100,200,10 ‹CR›	.- *Borra las instrucciones* **100, 200,** *y la* **10.**
DELETE 500 ‹CR›	.- *Borra la instrucción* **500.**

Se puede utilizar el comando **DELETE**, *escribiendo solamente las tres primeras letras (* **DEL** *), por la facilidad que nos brinda el compilador* **BASIC.**

Al utilizar cualquiera de las formas vistas anteriormente, el computador nos responderá con la palabra **READY.**

Ejemplo :

Supongamos que el programa contiene las siguientes instrucciones :

```
10 LET A = 5                              ‹CR›
20 LET B = 6                              ‹CR›
30 LET C = A + B                          ‹CR›
40 PRINT "EL RESULTADO ES ";C             ‹CR›
50 END                                    ‹CR›
```

Y deseo borrar las líneas **20** *y* **40** *utilizaría :*

```
DELETE 20,40 ‹CR›

Ready                    ←—— Indica que acabó el trabajo, o sea que borró las líneas 20 y 40.
```



```
LIST ‹CR›                ←—— Solicito el listado actualizado

10 LET A = 5
30 LET C = A + B
50 END

Ready
```

Otra manera de borrar una instrucción, es dándole el número de línea que desea borrar.

Ejemplo :

Las siguientes instrucciones fueron las que nos quedaron del ejemplo anterior.

```
10 LET A = 5
20 LET C = A + B
50 END
```

Si desea borrar la línea **30**, *solamente hacemos lo siguiente :*

```
30 <CR>                  ←—— Borra la línea # 30

                         ←—— Cursor, al borrar de esta manera,
                             no responderá con READY, porque
                             no utilizamos el comando DELETE

LIST                     ←—— Solicito el listado actualizado

10 LET   A = 5
50 END

Ready
```

EDIT .- *Comando que sirve para arreglar una instrucción, sin necesidad de volverlo a teclear.*

Formato :

EDIT *# línea/palabra anterior/palabra nueva/*

Donde :

EDIT	.-	*Es un comando* **BASIC.**
# DE LINEA	.-	*# De línea a modificar.*
PALABRA ANTERIOR	.-	*Palabra que se sustituye.*
PALABRA NUEVA	.-	*Palabra que agregaremos.*

Ejemplo :

Contamos con la siguiente instrucción :

```
100 PRINT "UNIVERSIDAD AUTONOMA DE NUEVO LEON"
```

Y si deseamos cambiar, la palabra UNIVERSIDAD por la palabra ESCUELA.

Sin necesidad de volver a teclear toda la instrucción, hacemos el siguiente paso :

```
EDIT 100/UNIVERSIDAD/ESCUELA/  <CR>
```

Al hacer lo anterior, la máquina imprimirá en la pantalla la línea con el cambio realizado.

```
100 PRINT "ESCUELA AUTONOMA DE NUEVO LEON"
```

Al trabajar con el comando **EDIT**, *usted buscará la palabra que desea sustituir.*

RUN .- *Comando que sirve para ejecutar el programa. Significa que va a procesar todas las instrucciones.*

Ejemplo :

Vamos a suponer, que las siguientes instrucciones, ya fueron alimentadas al computador.

```
NEW  8T363465A   <CR>

Ready

10 LET A = 5                              <CR>
20 LET B = 6                              <CR>
30 LET C = A + B                          <CR>
40 PRINT "EL RESULTADO ES ";C             <CR>
50 END                                    <CR>

RUN <CR>          <------  Ejecuta el programa

8T363465A     02-DIC-1988     17:59

EL RESULTADO ES 11

Ready
```

Observamos, que al ejecutar el programa la computadora presenta : el nombre del programa, la fecha y hora en que se ejecutó y por último el resultado del programa.

SAVE .- *Sirve para almacenar el programa en el disco. Obtiene una copia del programa que estamos realizando, almacenándolo en el disco en un directorio y lo ordena alfabéticamente.*

Ejemplo :

```
NEW 7L430824B <CR>

Ready

10 LET HR = 48                            <CR>
20 LET SDOHR = 1000                       <CR>
30  LET  SDONTO  =  SDOHR  *  HR          <CR>
40 PRINT "EL SUELDO NETO ES = "; SDONTO   <CR>
50 END                                    <CR>
```

```
RUN     <CR>          <---- Ejecutamos el programa

7L430824B       21-APR-89       12:50

EL SUELDO NETO ES = 48000     <---- Imprime el resultado

Ready

SAVE    <CR>          <---- Se almacena en el disco el programa 7L430824B

Ready                 <---- Terminó, ya está almacenado en el disco el programa
```

Si quisiera estar seguro de haberlo almacenado, solicitaremos que nos muestre los programas grabados en el directorio y efectuamos el siguiente paso :

```
$ DIRECTORY    <CR>   <---- Muestra todos los programas que están almacenados

DIRECTORY DRC0:[7L]

7L146875A.BAS;1   7L146875B.BAS;1   7L146875B.BAS;2
7L234567A.BAS;1   7L234567B.BAS;1   7L234567C.BAS;1
7L430824B.BAS;1   7L445565A.BAS;1   7L465432B.BAS;1
7L465432B.BAS;2   7L654789A.BAS;1   7L667543A.BAS;1
7L765894C.BAS;1

Total of 13 files.

Ready
```

Si observamos el directorio anterior, el programa **7L430824B.BAS** *se encuentra grabado.*

Vamos a explicar, que significa la información que nos presenta al solicitar el directorio de programas.

DIRECTORY DRC0:[7L]

- *Nos indica que estamos trabajando en el directorio* **[7L]**, *del disco* **DRC0**.

total of 13 files

- *Nos indica el número total de archivos almacenados. Pueden ser : archivo de datos, programas fuentes, objetos, ejecutables y temporales.*

Observación :

Al desear el directorio de programas, puede teclear solamente los primeros tres (3) caracteres (**DIR**).

Ejemplo :

```
$ DIR     <CR>        <---- Solicitamos el directorio

DIRECTORY DRC0:[7L]

7L146875A.BAS;1   7L146875B.BAS;1   7L146875B.BAS;2
7L234567A.BAS;1   7L234567B.BAS;1   7L234567C.BAS;1
7L430824B.BAS;1   7L445565A.BAS;1   7L465432B.BAS;1
7L465432B.BAS;2   7L654789A.BAS;1   7L667543A.BAS;1
7L765894C.BAS;1

Total of 13 files.

Ready
```

OLD .- *Con este comando, podemos llamar un programa que se encuentre almacenado en el directorio (disco), para poder trabajar con él.*

Formato :

OLD *Nombre*

Donde :

Nombre .- *Es el nombre del programa.*

El computador obtiene una copia del programa almacenado en disco y lo guarda en la memoria de la terminal.

Para qué deseamos llamar un programa que ya se encuentra almacenado ?

- *Muchas veces para actualizar el programa.*
- *Solamente consultar su programación.*
- *Terminar de alimentar las instrucciones.*
- *Etc.*

Ejemplo :

Deseo trabajar con el programa llamado **7L234567C**, *que se encuentra en el directorio de programas, para ello, debo realizar el siguiente paso :*

```
OLD  7L234567C  <CR>      <---- Llamo el programa

Ready                     <---- Indica que ya se encuentra en la memoria

LIST            <CR>      <---- Liste el programa

7L234567C      02-APR-1989    16:32

10 LET C1 = 100
20 LET C2 = 90
30 LET C3 = 80
40 LET SC = C1 + C2 + C3
50 PROM = SC / 3
60 PRINT "EL PROMEDIO ES = ";PROM
70 END

Ready
```


Observación :

En caso de que el programa que deseo llamar, no se encuentre en el directorio, la máquina contestará con el siguiente mensaje :

% BASIC-F-OPENIN, ERROR OPENING DRCO:[7L] NOMBRE.BAS AS INPUT -RMS-E-FNF, FILE NOT FOUND.

NOS INDICA QUE NO EXISTE EL PROGRAMA NOMBRE.BAS EN EL DIRECTORIO [7L] DEL DISCO DRCO:

Donde :

NOMBRE.BAS .- *Es el nombre del programa que se solicitó.*

RENAME .- *Sirve para cambiarle el nombre al programa.*

Formato :

RENAME NUEVO NOMBRE

Muchas veces, el nombre del programa no cumple con los requisitos de estandarización que se nos exige en sistemas o simplemente, al comenzar a trabajar se nos olvidó asignarle el nombre.

Ejemplo :

```
NEW 7L888888B  <CR>      <---- Se le asigna el nombre al programa

Ready

10 PRINT "HOLA"      <CR>
20 PRINT "U.A.N.L"   <CR>   <---- Instrucciones
30 PRINT "GRACIAS"   <CR>
40 END
```

```
LIST              <CR>   <---- Liste el programa

7L888888B     12-APR-1989     13:40

10 PRINT "HOLA"
20 PRINT "U.A.N.L"
30 PRINT "GRACIAS"
40 END

Ready                    <---- Termina de dar
                               el listado

RENAME  7L999999C        <---- Renombramos el
                               programa

Ready                    <---- Se realizó el
                               cambio

LIST                     <---- Listamos de
                               nuevo

7L999999C     12-APR-1989     12:41   <--+
                                          |
10 PRINT "HOLA"                      Ahora el
20 PRINT "U.A.N.L"                   programa es
30 PRINT "GRACIAS"                   7L999999C
40 END

Ready
```

REPLACE .- *Comando que sirve para reemplazar cambios o nuevas instrucciones que se agregaron al programa, generando una nueva versión en el directorio.*

Formato :

```
REPLACE      <CR>

Ready
```


UNSAVE .- *Sirve para borrar un programa que se encuentra almacenado en el directorio.*

Si existe un programa con varias versiones, borrará la versión más actualizada, (la versión que contenga el número más alto).

Formato :

```
UNSAVE       <CR>

Ready
```

EXIT .- *Este comando lo utilizamos para poder salir del* **interpretador BASIC.**

Formato :

```
EXIT         <CR>

$
```

Al contestar el computador con el signo de **$**, *nos indica que estamos fuera del* **interpretador BASIC.**

2.6 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Qué es una terminal ?.*

2.- *Explique la función de las siguientes teclas de la terminal.*

RETURN

DELETE

SHIFT

CAPS LOCK

CTRL **S**

CTRL **Q**

CTRL **C**

CTRL **U**

3.- *Qué nos indican las siguientes palabras ?.*

USERNAME : _____

PASSWORD : _____

4.- *Complete la siguiente ilustración de los pasos para entrar al sistema* **VAX-11/780.**

```
USERNAME :
PASSWORD :

bienvenidos a FACPYA
$

Ready
```

5.- *Complete la siguiente ilustración de los pasos para salir del sistema* **VAX-11/780.**

```
Ready


 GP7L Logged out at 2-DEC-1989 17:10:15:12
```

6.- *Qué es un* **COMANDO** *?.*

7.- *Explique los siguientes* **COMANDOS DE BASIC :**

NEW _____

LIST _____

DELETE _____

RUN _____

SAVE _____

OLD _____

RENAME _____

CAPITULO 3

DISEÑO DE UN PROGRAMA

OBJETIVO GENERAL

Enseñar y orientar al estudiante en la metodología para el desarrollo de un programa **BASIC.**

OBJETIVO ESPECIFICO

En este capítulo, se orienta al alumno en los pasos necesarios para iniciarse en el diseño de un programa, con los símbolos más importantes para el arreglo de un diagrama de flujo, así como la forma de construir las instrucciones con las que se alimentarán al computador.

7.- *Explique los siguientes* **COMANDOS DE BASIC :**

NEW _____

LIST _____

DELETE _____

RUN _____

SAVE _____

OLD _____

RENAME _____

CAPITULO 3

DISEÑO DE UN PROGRAMA

OBJETIVO GENERAL

Enseñar y orientar al estudiante en la metodología para el desarrollo de un programa **BASIC.**

OBJETIVO ESPECIFICO

En este capítulo, se orienta al alumno en los pasos necesarios para iniciarse en el diseño de un programa, con los símbolos más importantes para el arreglo de un diagrama de flujo, así como la forma de construir las instrucciones con las que se alimentarán al computador.

3.1 PASOS PARA EL DISEÑO DE UN PROGRAMA

Antes de mencionar los pasos, debemos conocer qué es un **programa ?**

Un **PROGRAMA** *es una serie ordenada de instrucciones en un lenguaje de alto nivel (***BASIC, COBOL, PASCAL,** *etc.).*

Existen varios tipos de lenguajes y cada uno está enfocado a distintas áreas de trabajo.

Los tipos de lenguajes de programación son : **BASIC, COBOL, FORTRAN, PASCAL, RPG, PL1, ADA,** *etc.*

Los pasos para el diseño de un programa, son básicamente cuatro :

1. - *Conocer el* **OBJETIVO** *del problema a resolver.*
2. - *Hacer un* **ALGORITMO.**
3. - *Diseñar un* **DIAGRAMA DE FLUJO.**
4. - **CODIFICAR** *el programa.*

3.2 PASO 1 .- OBJETIVO DEL PROBLEMA

Para poder diseñar un programa, debemos de saber como llegar al objetivo.

Un ejemplo sencillo es el siguiente :

> *Si me encuentro en mi carro, en el centro de la ciudad y mi objetivo es llegar a Saltillo, lógicamente tomaría la carretera a Saltillo ; pero en caso de tomar la carretera a Reynosa, tal vez llegue a Saltillo, pero con ciertas desventajas, tales como : cansancio, gasto de gasolina, mayor recorrido, etc. Si yo sé como llegar al objetivo, no tendré problemas para resolverlo.*

Esto mismo ocurre para programar una computadora, si sé como obtener una nómina, lo puedo programar; serían los mismos pasos manuales que las instrucciones a utilizar por el computador. Por lo tanto, debemos conocer el procedimiento para llegar al objetivo.

3.3 PASO 2.- CREACION DE UN ALGORITMO

3.3.1 *Un* **ALGORITMO** *es una serie de pasos lógicos que sirven para poder llegar a un objetivo. Es un procedimiento para resolver un problema.*

Los algoritmos pueden ser generales o detallados.

Ejemplo de un algoritmo general :

1. - *Inicio.*
2. - *Capturar información.*
3. - *Calcular deducciones.*
4. - *Calcular percepciones.*
5. - *Quién realizó horas extras ?.*
6. - *Calcular sueldo neto.*
7. - *Dar resultado del sueldo neto.*
8. - *Termina el algoritmo*

En el caso del ejemplo anterior, sabemos que en cada punto existen muchos pasos más, todos esos pasos faltantes van a conformar lo que es un algoritmo detallado.

Cuando llegue a diseñar un programa en la computadora, se recomienda hacer un algoritmo detallado, puesto que el computador no sabrá lo siguiente :

a) . - *Qué porcentaje de impuesto quitará ?.*
b) . - *Si el empleado trabajó horas extras ?.*
c) . - *Cuánto le asignará ?.*
d) . - *Qué datos va a preguntar ?.*
e) . - *Y muchas cuestionamientos más.*

Para poder diseñar un buen programa, siempre le daremos el detalle más mínimo que llegue a necesitar la computadora.

El punto que estamos tratando es el más importante para poder programar, ya que es pura **LOGICA** *y la práctica de éste, nos sirve para pensar en los pasos necesarios que emplearemos en la solución de nuestro problema que queremos computarizar.*

3.3.2 *A continuación se encuentran algunos ejemplos de algoritmos :*

Ejemplo 1.-

Algoritmo para obtener el promedio de un alumno.

1.- *Inicio del algoritmo.*
2.- *Cuál es el nombre del alumno ?.*
3.- *Qué calificaciones obtuvo en sus parciales ?.*
4.- *Sumar las calificaciones.*
5.- *Divida el resultado de la suma de calificaciones entre el número de calificaciones y nos dará el promedio del alumno.*
6.- *Presentar los resultados (Quién es el alumno y el promedio que obtuvo).*
7.- *Finaliza el algoritmo.*

Con el algoritmo anterior nos enseña lo siguiente :

QUE LOS PASOS LLEVAN ORDEN LOGICO

a).- *No puedo obtener el promedio si no tengo la suma.*

b).- *No puedo hacer la suma si no tengo las calificaciones.*

c).- *De dónde obtengo las calificaciones ?, preguntamos por ellas, de una lista !, de los examenes !, etc.*

Observe el ejemplo paso por paso y se dará cuenta, que lo haría en forma manual, con ésto le quiero demostrar que cuando se quiera diseñar un programa, son los mismos pasos manuales que los computarizados.

Lo importante es que entienda como llegar al objetivo que desea.

Ejemplo 2 .-

Algoritmo para obtener el sueldo neto de " N " empleados.

1.- *Inicio del algoritmo.*
2.- *Cuál es nombre del trabajador ?.*
3.- *Si hay trabajador, vé al paso 4, si no hay trabajador vé al paso 9.*
4.- *Cuántas horas trabajó ?.*
5.- *Cuanto le pagan por hora ?.*
6.- *Multiplicar las horas trabajadas por el sueldo hora y obtenemos el sueldo neto.*
7.- *Presentar los resultados (nombre del empleado y su sueldo neto).*
8.- *Ir por otro empleado, continuar con el paso 2.*
9.- *Termina el algoritmo.*

En este ejemplo, no estamos incluyendo deducciones, percepciones, total del sueldo neto, total de trabajadores, etc., pero se pueden agregar nadamás incluyendo los pasos en el lugar donde correspondan.

En el paso 3, se está tomando una decisión, si existe trabajador continuamos preguntando, en caso contrario, daremos por terminado nuestro problema.

Ejemplo 3 .-

Algoritmo para obtener los datos generales de " N " personas.

1.- *Inicio del algoritmo.*
2.- *Cuál es el nombre de la persona ?.*
3.- *Si hay persona vé al paso 4, si no al paso 10.*
4.- *Cuál es su dirección ?.*
5.- *Cuál es su teléfono ?.*
6.- *Qué edad tiene ?.*
7.- *En dónde trabaja ?.*
8.- *Dar los resultados (nombre, dirección, teléfono edad, lugar de trabajo).*
9.- *Ir por otra persona, continuar con el paso 2.*
10.- *Termina el algoritmo.*

Si observamos, en este algoritmo no estamos realizando ninguna operación aritmética, porque no existe la necesidad de hacerlo, solamente nos concretamos a preguntar.

Otra vez ocurre lo mismo que en ejemplos anteriores, el procedimiento manual sería el mismo para el computarizado.

Ejemplo 4.-

Algoritmo para obtener la comisión de cada vendedor que labora en la compañía " **XYZ** *" y obtener el total de la comisiones que tendrá que pagar dicha compañía. Son cinco (5) vendedores.*

Porcentaje de comisión :

Si la venta es menor a **$** *50,000.00, le asignaremos el 10% de comisión, pero si su venta fué igual o mayor a* **$** *50,000.00, le daremos el 20% de comisión.*

1. - *Inicio del algoritmo.*
2. - *Vendedor* 1. *viene el primer vendedor.*
3. - *Ya son más de 5 vendedores vé al paso 13, si no lo son vé al paso 4.*
4. - *Nombre del vendedor ?.*
5. - *Cuál fué su venta ?.*
6. - *La venta es menor a* **$** *50,000.00, vé al paso 7, si no, vé al paso 8.*
7. - *Comisión es igual a la venta por el porcentaje del 10%, Comisión = venta * .10 y despues vé al paso 9.*
8. - *Comisión es igual a la venta por el porcentaje de 20%, Comisión = venta * .20.*
9. - *Dar los resultados (nombre del vendedor, la venta que realizó y su comisión).*
10. - *Acumular la comisión del vendedor para el total de comisiones.*
    total de comisiones = total de comisiones + comisión.
11. - *Sumamos* 1 *al número de vendedores.*
12. - *Continuar con el paso 3.*
13. - *Dar el resultado del total de comisiones.*
14. - *Termina el algoritmo.*

En este ejemplo, vemos que trabajamos acumuladores y sumadores como lo demuestran los pasos 10 *y* 11 *respectivamente.*

Se toma una decisión en el paso 3, para saber si ya pasaron los 5 vendedores para calcularle su comisión dependiendo de su venta, en la cual también se toma la decisión en el paso 7.

3.4 PASO 3 .- DIAGRAMA DE FLUJO

3.4.1 *El* **DIAGRAMA DE FLUJO** *es una representación gráfica de todos los pasos (operaciones) a realizarse para poder llegar al objetivo.*

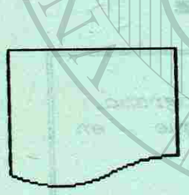
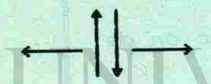
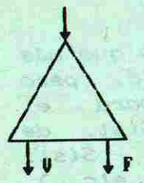
El diagrama de flujo es representado por símbolos que tienen un significado especial.

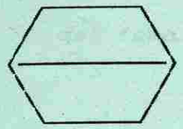
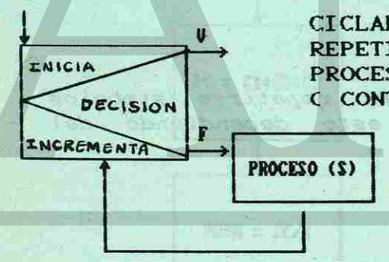
3.4.2 VENTAJAS *del diagrama de flujo :*

1. - *Visualizamos mejor la lógica del programa. Es más entendible en símbolos que en instrucciones.*
2. - *Detectamos más rápido los errores.*
3. - *Es muy utilizado para la documentación del programa y de un sistema de información.*
4. - *Nos ayuda a obtener nuestra codificación de instrucciones.*

3.4.3 SIMBOLOS MAS IMPORTANTES

A continuación, vamos a mostrar los símbolos que más se utilizan en el diseño de los diagramas de flujo, pero tambien hay que mencionar, que existen símbolos para el diseño de todo un sistema de información. (Sist. de contabilidad, Sist. de nómina, Sist. de inventarios, Sist. de tablas de amortización, Sist. de cuentas por pagar, etc.)

SIMBOLO	NOMBRE	SIGNIFICADO
	**INICIO o FIN**	*Para indicar el inicio o el fin del diagrama de flujo.*
	**PROCESO u OPERACION**	*Para especificar una operación aritmética o la asignación de un valor.*
	**LECTURA o ENTRADA**	*Espera la entrada de datos que va alimentar.*
	**IMPRESION**	*Nos indica los resultados que vamos a presentar.*
	**FLECHAS DE SEGUIMIENTO**	*Es el flujo del proceso.*
	**DECISION**	*Decide mediante una condición si es verdadera o falsa.*

SIMBOLO	NOMBRE	SIGNIFICADO
	**SUBRUTINA o SUBPROGRAMA**	*Especifica una serie de instrucciones que tendrá que procesar, dentro del mismo programa.*
	**CONECTOR**	*Para conectar diferentes símbolos del diagrama en una misma página.*
	**CONECTOR DE PAGINA**	*Para conectar diferentes símbolos del diagrama, que se encuentran en distintas páginas.*
	**CICLADOR o REPETIDOR DE PROCESOS (CONTADOR)**	*Sirve para repetir uno o varios procesos, dependiendo de una condición.*

Los símbolos anteriores son los utilizables para el diagrama de flujo, más adelante, estudiaremos que instrucciones **BASIC** *les corresponde a cada uno de ellos.*

3.4.4 Ejercicios Prácticos

En los siguientes ejemplos, vamos a mostrar la facilidad de construcción del diseño de un diagrama de flujo.

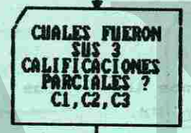
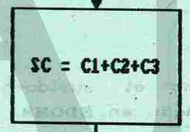
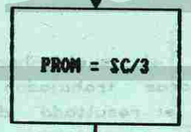
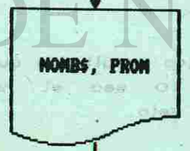
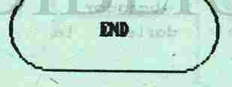
Teniendo como base la lógica y el seguimiento de nuestro problema, tomaremos de nuestro algoritmo, paso por paso y buscaremos el símbolo correspondiente de acuerdo al proceso que se vaya a realizar para el diseño del diagrama.

En un diagrama de flujo pueden repetirse símbolos cuantas veces sea necesario, todo esto dependiendo del programa que estemos realizando.

En el diagrama de flujo que a continuación se presenta, observarás que no se utilizarán todos los símbolos que anteriormente estudiamos, por ejemplo : el conector, decision, subrutina, ciclador, etc.

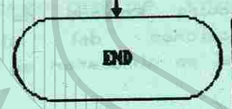
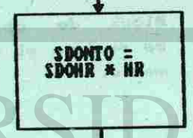
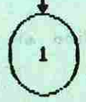
Ejemplo 1.-

Diagrama de flujo correspondiente al ejemplo # 1 del tema algoritmos. Obtener el promedio de un alumno.

Algoritmo	Diagrama de flujo	Observaciones
1		Block de inicio. Indica el principio del diagrama.
2		Block de lectura. Vamos a dar entrada al dato. En este caso, el nombre del alumno se almacena en **NOMB$**.
3		Block de lectura. Entrada de las calificaciones del alumno. Como son tres, se almacenan en **C1, C2, y C3.**
4		Block de operación. Se efectúa la suma de calificaciones.
5		Block de operación. Porque se realiza el cálculo del promedio.
6		Block de impresión. Presenta los resultados deseados. En este caso, el nombre y el promedio.
7		Indica el fin del diagrama.

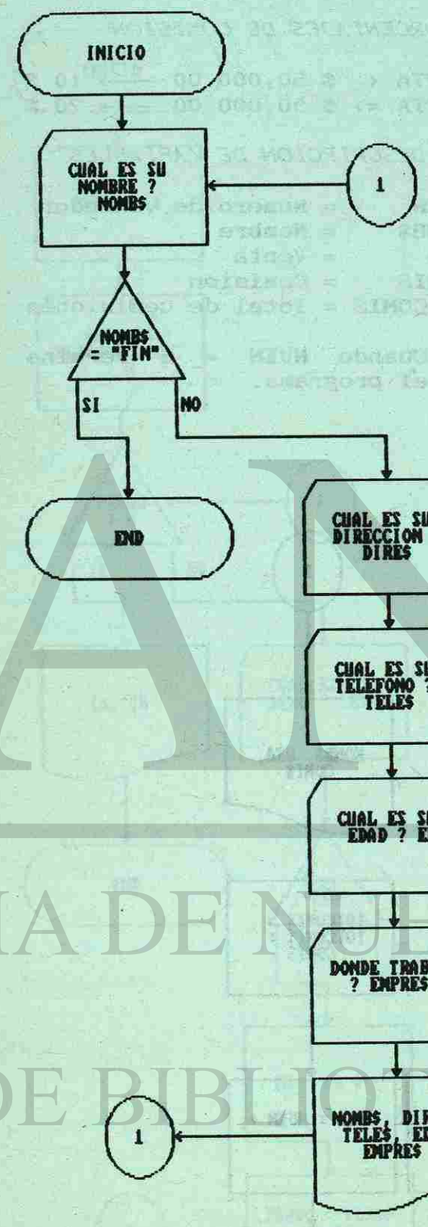
Ejemplo 2.-

Diagrama de flujo correspondiente al ejemplo # 2 del tema algorítmos. Obtener sueldo neto de cada empleado.

Algo-ritmo	Diagrama de flujo	Observaciones
1		
2		Almacena el nombre en NOMB$
3		Decide si el nombre es igual a FIN. Si fuera igual termina el proceso.
9		
		Almacenamos las horas que trabajó en HR.
4		
5		Almacenamos el sueldo-hora que le pagan en SDOHR.
6		Multiplica el sueldo-hora por las horas trabajadas y guardamos el resultado del sueldo neto en SDONTO.
7		Presenta los resultados que requiero. O sea el nombre y el sueldo neto.
8		Block conector, para que vaya a pedir los datos del siguiente trabajador o en su caso darle la palabra FIN.

Ejercicio 3.-

Diagrama de flujo correspondiente al ejemplo # 3 del tema algorítmos. Imprimir datos generales de " N " personas.



NOTA 1 :

En este ejercicio, sí utilizamos el símbolo de decisión y un conector por necesidad del programa.

NOTA 2 :

En este diagrama, como no realizamos operaciones aritméticas, solamente utilizamos blocks de lecturas para la aceptación de datos, impresión para resultados y decisión para la terminación.

Ejercicio 4.-

Diagrama de flujo correspondiente al ejemplo # 4 del tema algoritmos. Obtiene la comisión de cada vendedor y el total que tendrá que pagar la empresa **" XYZ "** *a cinco (5) vendedores.*

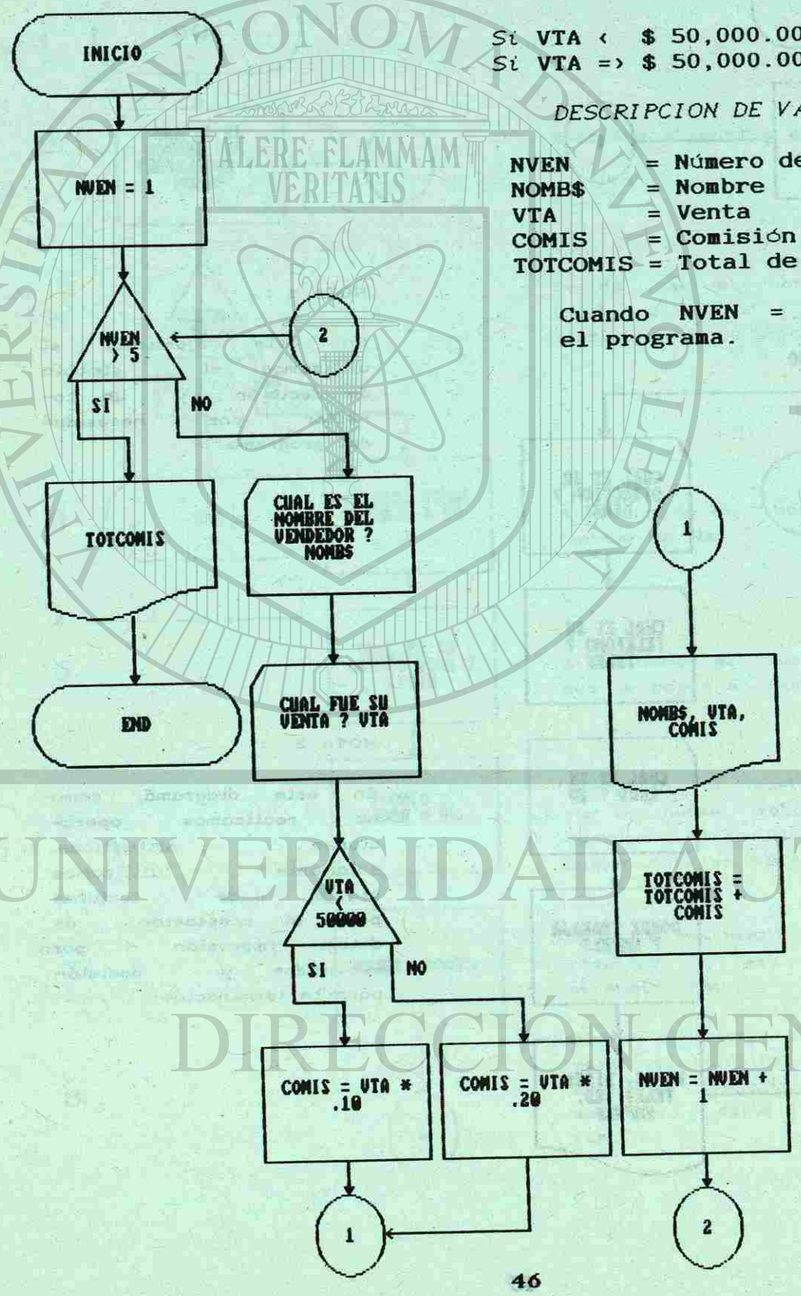
PORCENTAJES DE COMISION

Si **VTA ‹ $ 50,000.00** ⟶ **10 %**
Si **VTA => $ 50,000.00** ⟶ **20 %**

DESCRIPCION DE VARIABLES

NVEN = Número de vendedor
NOMB$ = Nombre
VTA = Venta
COMIS = Comisión
TOTCOMIS = Total de comisiones

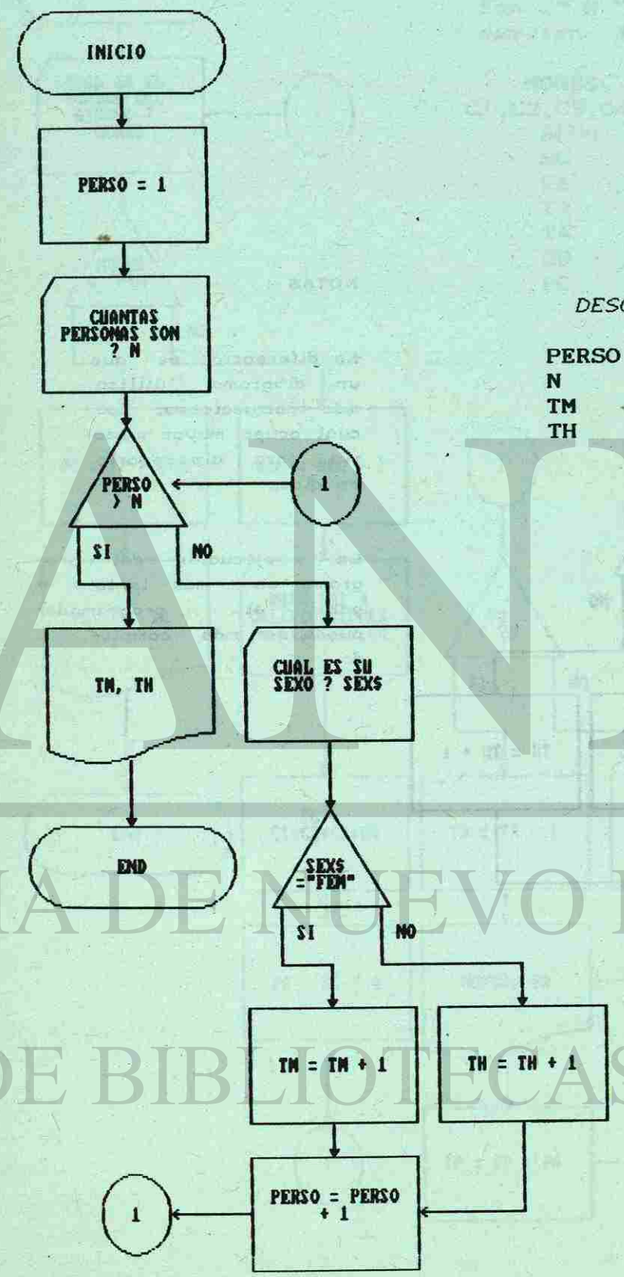
Cuando **NVEN = 6** termina el programa.



Ejercicio 5.-

Diseñar un diagrama de flujo, para saber el total de mujeres y el total de hombres de un grupo de personas.

Diagrama " A "

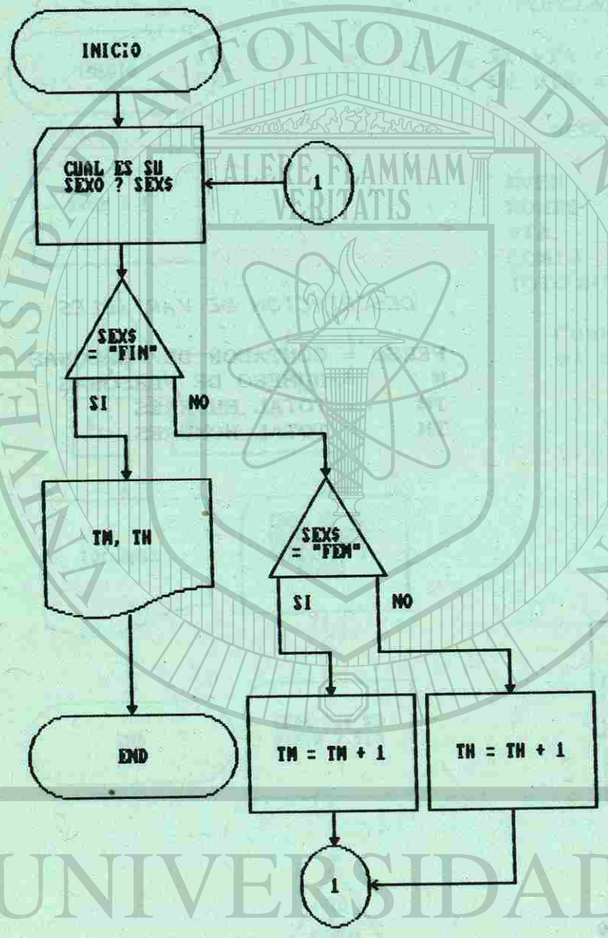


DESCRIPCION DE VARIABLES

PERSO = CONTADOR DE PERSONAS
N = NUMERO DE PERSONAS
TM = TOTAL MUJERES
TH = TOTAL HOMBRES

Un diagrama se puede diseñar de distintas maneras y todas ellas pueden llegar al mismo objetivo. En este caso vamos a resolver el problema anterior con un procedimiento distinto.

Diagrama " B "



NOTAS :

La diferencia es que un diagrama útiliza más instrucciones, lo cual ocupa mayor memoria para almacenarlo en disco.

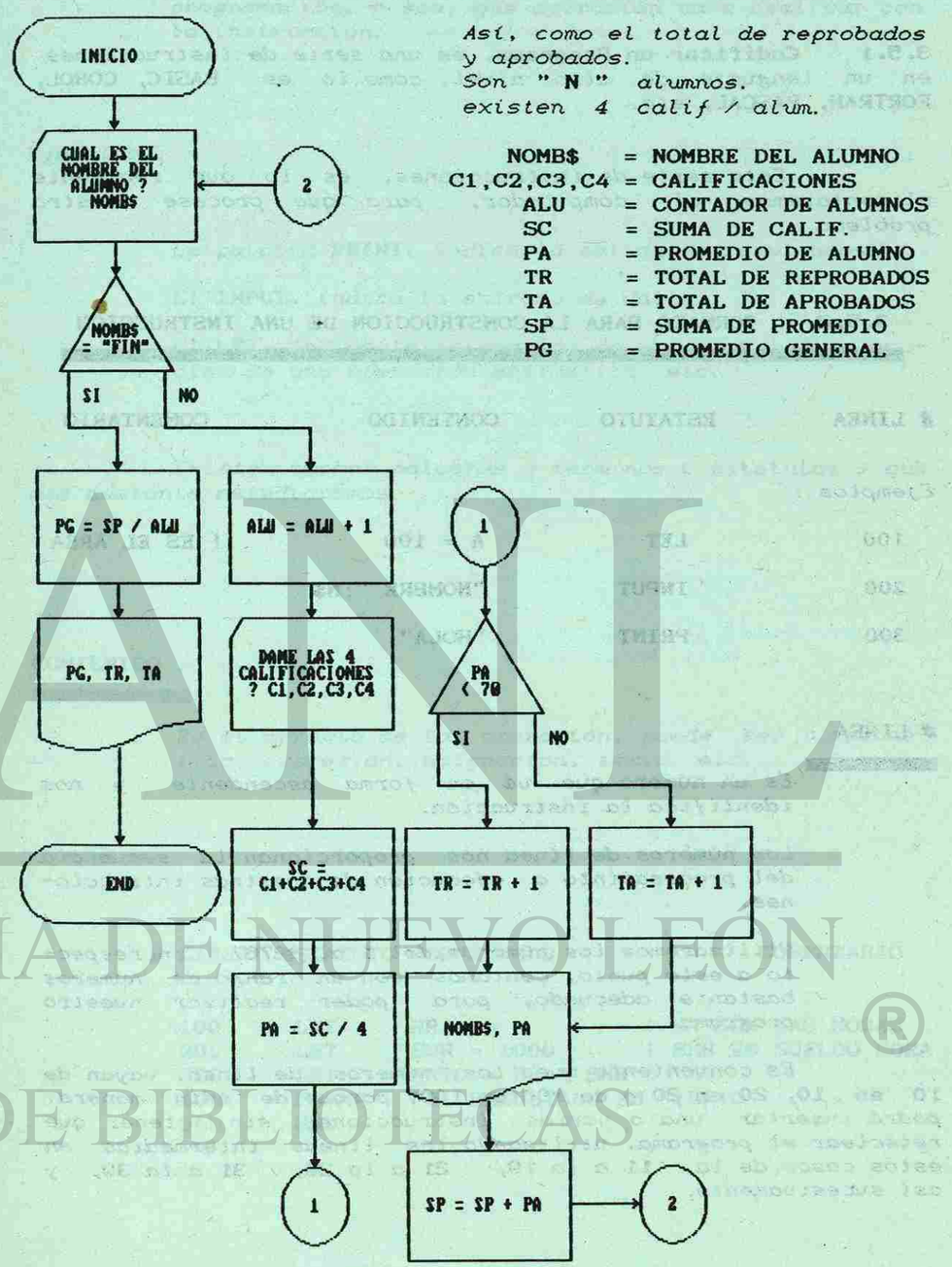
La ejecución del programa es más lento y para el programador puede ser más complido.

Ejemplo 6.-

Diseñar un diagrama de flujo para obtener el promedio de cada alumno y el promedio general del grupo.

Así, como el total de reprobados y aprobados.
Son " N " alumnos.
existen 4 calif / alum.

NOMB$	= NOMBRE DEL ALUMNO
C1,C2,C3,C4	= CALIFICACIONES
ALU	= CONTADOR DE ALUMNOS
SC	= SUMA DE CALIF.
PA	= PROMEDIO DE ALUMNO
TR	= TOTAL DE REPROBADOS
TA	= TOTAL DE APROBADOS
SP	= SUMA DE PROMEDIO
PG	= PROMEDIO GENERAL



3.5 PASO 4 .- CODIFICACION DE UN PROGRAMA

3.5.1 Codificar un Programa, *es una serie de instrucciones en un lenguaje de alto nivel, como lo es* **BASIC, COBOL, FORTRAN, PASCAL,** *etc.*

Esta serie de instrucciones, es lo que realmente alimentaremos al computador, para que procese nuestro problema.

3.5.2 FORMATO PARA LA CONSTRUCCION DE UNA INSTRUCCION

# LINEA	ESTATUTO	CONTENIDO	COMENTARIO

Ejemplos :

```
100        LET        A = 100          ! ES EL AREA
200        INPUT      "NOMBRE ";N$
300        PRINT      "HOLA"
```

 # LINEA :

Es un número que vá en forma ascendente y nos identifica la instrucción.

Los números de línea nos proporcionan la secuencia del procesaminto o ejecución de nuestras intrucciones.

Utilizaremos los números del **1** *al* **32767**. *Con respecto a este punto, contamos con un rango de números bastante adecuado, para poder realizar nuestro programa.*

Es conveniente que los números de línea, vayan de **10** *en* **10, 20** *en* **20** *o de* **100** *en* **100**, *porque de ésta manera, podrá insertar una o varias instrucciones, sin tener que reteclear el programa, utilizando las líneas intermedias en estos casos de la* **11** *a la* **19, 21** *a la* **29, 31** *a la* **39,** *y así sucesivamente.*

ESTATUTO :

Un estatuto es una palabra clave en un lenguaje de programación, o sea, que operación va a realizar con la instrucción.

Ejemplos :

La palabra **PRINT**, *indica la salida de resultados.*

El **INPUT**, *indica la entrada de datos.*

El **LET**, *indica la asignación de un valor o la ejecución de una operación aritmética, etc.*

Existen muchas palabras o términos (**estatutos** *) que más adelante estudiaremos.*

CONTENIDO :

Es el arreglo de la operación, puede ser : aritmética, expresión, asignación, texto, etc.

Ejemplos :

```
# LINEA  ESTATUTO  CONTENIDO          COMENTARIO

100      LET      HR = 5             ! HR SON LAS HORAS
200      LET      SHR = 1000         ! SHR ES SUELDO HORA
300      LET      SNTO = HR * SHR
400      PRINT    "EL RESULTADO ES ";SNTO
500      END
```

COMENTARIO :

En cada línea puede especificar un comentario si lo desea. Es útil, puesto que sabemos qué significa el valor que estamos trabajando.

Observe que en el ejemplo anterior, existen comentarios en las lineas **100 y 200.**

El signo de admiración (!), especifica que es un comentario.

Ya que conocemos como está formada una instrucción, vamos a desarrollar una codificación completa de un programa, aunque todavia desconocemos los estatutos de la programación **BASIC.**

3.5.3 *En el siguiente ejemplo codificaremos un diagrama que se realizó en temas pasados.*

Algoritmo	Diagrama de flujo	Codificación
1	INICIO	10 REM OBTIENE PROMEDIO
2	CUAL ES EL NOMBRE DEL ALUMNO ? NOMB$	20 INPUT "NOMBRE ALUMNO ";NOMB$
3	CUALES SON SUS 3 CALIFICACIONES PARCIALES ? C1,C2,C3	30 INPUT "CALIFICACIONES";C1,C2,C3
4	SC = C1+C2+C3	40 LET SC = C1 + C2 + C3
5	PROM = SC / 3	50 LET PROM = SC / 3
6	NOMB$, PROM	60 PRINT "ALUMNO (A) "; NOMB$ 70 PRINT "EL PROMEDIO ES "; PROM
7	END	80 END

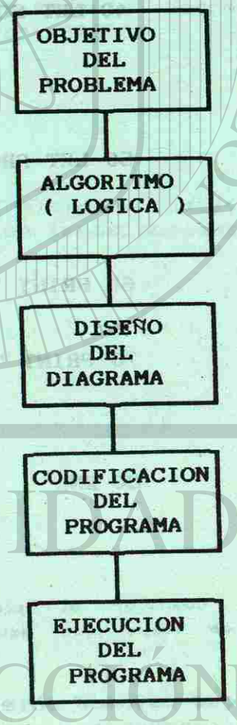
OBSERVACIONES :

Para poder codificar el programa, partimos de nuestro diagrama de flujo, que éste a su vez, se obtuvo del algorítmo.

Poco a poco que vayamos obteniendo experiencia en la programación, ya no tendremos necesidad de diseñar su diagrama, pero hay que recordar que nos ayuda mucho contar con él.

Concluida nuestra codificación, sólo nos queda alimentar las instrucciones al computador y ejecutar el programa.

Recuerden, el programa es la codificación; hasta aquí, hemos explicado los pasos para el diseño de un programa y lo podemos resumir de la siguiente manera :



3.6 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Qué es un* **PROGRAMA** *?*

2.- *Escriba 4 tipos de* **LENGUAJES DE PROGRAMACION.**

3.- *Cuáles son los pasos para el diseño de un programa ?*

1.- _____ 2.- _____

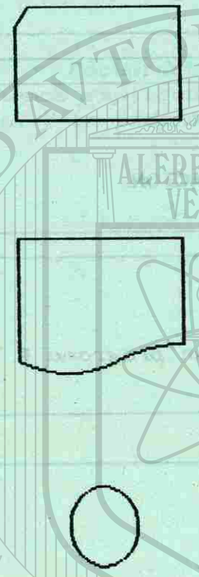
3.- _____ 4.- _____

4.- *Qué es un* **ALGORITMO** *?*

5.- *Defina lo que es un* **DIAGRAMA DE FLUJO.**

6.- *Escriba las ventajas que nos proporciona un* **DIAGRAMA DE FLUJO** *?*

7.- *Escriba el nombre y significado de los siguientes* **SIMBOLOS.**



8.- *Describa los pasos para el desarrollo del algoritmo del siguiente programa :*

En una caja que contiene 500 pelotas de diferentes colores, deseamos conocer cuantas de ellas son de color rojo, cuantas azules y el total de color negro.

9.- *Realice los pasos del algoritmo para la impresión de una tabla de multiplicar.*

10.- *Hacer el algoritmo para imprimir los números pares, a partir del 1 hasta el 250.*

11.- *Diseñar el algoritmo para obtener el total de años de un grupo de " N " personas.*

12.- *En la compañía TRANSFORMADORES S.A., desean conocer cuántos empleados tienen una antiguedad de 10, 20 y 30 años.*
Prepare el algoritmo para " N " empleados.

13.- *Diseñe los diagramas de flujo de los puntos 8, 9, 10, 11 y 12.*

14.- *Qué significa* **CODIFICAR** *?*

15.- *Escriba el formato para una instrucción.*

16.- *Cuál es la función de los números de línea y el máximo para utilizarlo en un programa* **BASIC** *?*

17.- *Defina lo que es un* **ESTATUTO.**

CAPITULO 4

COMPONENTES DE UN PROGRAMA

OBJETIVO GENERAL

Que el estudiante conceptualice los elementos básicos necesarios para el desarrollo del programa.

OBJETIVO ESPECIFICO

El alumno aprenderá las características de los tipos de variables donde podrá almacenar la información. Además conocerá la forma de como vá a realizar las expresiones tanto aritméticas, como lógicas y relativas, así como sus prioridades.

16.- *Cuál es la función de los números de línea y el máximo para utilizarlo en un programa* **BASIC** *?*

17.- *Defina lo que es un* **ESTATUTO.**

CAPITULO 4

COMPONENTES DE UN PROGRAMA

OBJETIVO GENERAL

Que el estudiante conceptualice los elementos básicos necesarios para el desarrollo del programa.

OBJETIVO ESPECIFICO

El alumno aprenderá las características de los tipos de variables donde podrá almacenar la información. Además conocerá la forma de como vá a realizar las expresiones tanto aritméticas, como lógicas y relativas, así como sus prioridades.

Los componentes de un programa **BASIC** son los siguientes :

1.- **CARACTERES.**

2.- **CONSTANTES.**

3.- **EXPRESIONES.**

4.- **VARIABLES.**

4.1 CARACTERES

El lenguaje BASIC utiliza el conjunto de **caracteres** del código americano estándar para el intercambio de información **ASCII**, (American Standar Code for Information Interchange).

El conjunto de caractéres ASCII, incluye :

a) .- Las letras de la **A** a la **Z**, tanto para mayúsculas como minúsculas.
(a,b,c,d...., A,B,C,D.....).

b) .- Los dígitos del **0** al **9**.

c) .- Todos los **Caracteres especiales**, tales como son : **%, -, +, *, @, $, :**, etc.

4.2 CONSTANTES

Una **constante** es un valor (número o letra) que nunca va a cambiar en el transcurso del programa, a menos que el programador lo requiera.

Las constantes pueden ser : reales, enteras o alfanuméricas.

Por ejemplo :

a) .- Reales ; 15.38, 1247.1, 1202.84, 2405.3, -94.7, -15.2

b) .- Enteras ; 2814, 3000, 5239, 12, 274

c) .- Alfanuméricas ; "UNIVERSIDAD"
"F.A.C.P.Y.A"
"JAVIER ELIZONDO"

4.3 EXPRESIONES

Las **expresiones** son operaciones aritméticas y lógicas.

Ejemplos :

```
X = Y + 2X + T

Y = A - B - C

Z = Y * Z / D

W = A + B - C * D

T = H * * 2
```

4.4 VARIABLES

Una **variable** es aquella que tiene capacidad para poder almacenar un dato.

También, se puede decir que una variable es caracter o un conjunto de caracteres (números, letras, o signos especiales).

4.4.1 *Los* **tipos de variables** *puden ser tres (3) :*

1) .- **REALES** (*Numéricas, ó Flotantes*).
2) .- **ENTERAS.**
3) .- **ALFANUMERICAS o STRING.**

1.- REALES

Una variable **real o flotante** *es aquella que puede almacenar un valor con decimal.*

CARACTERISTICAS DE VARIABLES REALES

a) .- *El* **Primer caracter** *del nombre debe ser alfabético (* **letra** *).*

b) .- *El* **nombre** *de la variable real, puede contener hasta* **30 Caracteres** *(letras, digitos y/o puntos).*

c) .- **No** *es permitido que el nombre lleve* **espacios en blanco** *intermedios.*

Ejemplos :

SUMA	**C87**	**HR**	**TOTPOL2**
PROM	**X**	**SDO.HR**	**INVTOT**
TOT.GRAL.NOM.	**COMIS**	**SDONTO**	**H3**
CALIF	**VTA**	**PREM**	**NOMINA**

2.- ENTERAS

Una variable **entera (%)** *es aquella que puede almacenar un valor entero.*

CARACTERISTICAS DE VARIABLES ENTERAS

a) .- *El* **primer caracter** *del nombre debe ser alfabético (* **letra** *).*

b) .- *El* **nombre** *de la variable entera puede contener hasta* **31 caracteres** *(letras, dígitos y/o puntos).*

c) .- **No** *es permitido que el nombre lleve* **espacios en blanco** *intermedios.*

d) .- *El* **último caracter** *del nombre debe ser un* **signo de porciento (%).**

Ejemplos :

NTO%	**TOT%**	**COMISION%**	**SUE%**
XYZ%	**POLIZA34%**	**HOJA%**	**COD.POS%**
VTA%	**HR.TRA.%**	**CON.LIN%**	**X%**
EDAD%	**ABCD%**	**SUM.CAL%**	**Z.1%**
C4%	**Y%**	**SDO.BRU%**	**PRE.EM.%**
PESO%	**KILOS%**	**METRO%**	**PERSEP%**
C10%	**ZYX%**	**SUMA%**	**T.VTA%**

3.- ALFANUMERICAS

Una variable **alfanumérica o string ($),** *es aquella que puede almacenar un valor con letras, números, letras-números y/o signos especiales.*

CARACTERISTICAS DE VARIABLES ALFANUMERICAS

a) .- *El* **primer caracter** *del nombre debe ser alfabético (letra).*

b) .- *El* **nombre** *de la variable alfanumérica puede contener hasta* **31 caracteres** *(letras, dígitos y/o puntos).*

c) .- **No** *es permitido que el nombre lleve* **espacios en blanco** *intermedios.*

d) .- *El* **último caracter** *del nombre debe ser el* **signo de dollar** *($).*

Ejemplos :

NOMB$	R.F.C.$	NAC.EMP$	HS2$
DIR.EMP$	FECHA$	DESCRIP$	MLCR$
TELE$	ABC.DE9$	NOM.ART8$	JMLL$
SEXO$	EDO.NAC$	A$	JGRL$
NOM.ALU$	Z34$	MAT.ALU$	EGH$
HSGG$	EGH$	HGH$	AHG$
JESP$	AU$	Y$	EPT$
TOTNOM$	POLIZA$	SDOBTO$	EDAD$

4.5 TIPOS DE OPERADORES

Existen tres tipos de operadores que son :

1.- *Operadores* **ARITMETICOS.**
2.- *Operadores* **RELACIONALES.**
3.- *Operadores* **LOGICOS.**

4.5.1 *Operadores* ARITMETICOS

En la siguiente tabla mostramos los diferentes operadores **aritméticos.**

TABLA DE OPERADORES ARITMETICOS		
OPERADOR	**EJEMPLO**	**SIGNIFICADO**
+	A + B	SUMAR A Y B
-	A - B	RESTAR EL VALOR DE A AL DE B
*	A * B	MULTIPLICAR A POR B
/	A / B	DIVIDIR A ENTRE B
^	A ^ B	ELEVAR A A LA POTENCIA B
**	A ** B	ELEVAR A A LA POTENCIA B

El computador ejecuta las operaciones aritméticas por prioridades como sigue :

1a. *Prioridad .- Elevar a la potencia.*

2a. *Prioridad .- Multiplicaciones y divisiones (*,/), en caso de que exista una operación, donde intervengan dos operadores con la misma prioridad, el computador ejecutará el primer operador que encuentre en un recorrido de izquierda a derecha.*

3a. *Prioridad .- Sumas y Restas (+ , -), se hace la misma observación que se hizo en la explicación de la 2a. prioridad.*

NOTA :

En caso de que una operación aritmética contenga paréntesis, el computador realizará todas las operaciones que se encuentren dentro de él, si son varios paréntesis, ejecutará primero el más interno, hasta el más externo.

Ejemplos de prioridad :

Ejemplo 1.

Si contamos con la siguiente operación aritmética :

R = A + B * C

Donde los valores son :

A = 10 **B = 100** **C = 2**

Qué valor obtendrá **R** *?*

Vamos a realizarlo paso por paso :

R = A + B * C ⟵ Operación aritmética que va a ejecutar el computador.

R = 10 + 100 * 2 ⟵ Sustituimos los valores y se multiplica 100 por 2 por ser mayor prioridad.

R = 10 + 200 ⟵ Por último sumamos 200 mas 10.

R = 210 ⟵ El resultado es 210.

En el ejemplo anterior, el resultado de R es el real, puesto que se ejecutó por sus prioridades.

Pero, que pasará si cambiamos las prioridades ?, primero sumamos y luego multiplicamos.

R = A + B * C ⟵ Operación aritmética.

R = 10 + 100 * 2 ⟵ Sustituimos los valores y sumamos 100 mas 10.

R = 110 * 2 ⟵ Ahora multiplicamos el 110 por 2.

R = 220 ⟵ El resultado es 220.

Notamos que el valor 220 de **R** *no es el real.*

Ejemplo 2.

Tenemos la siguiente operación aritmética :

R = Y / 2 + X * Y ** 2 - Z

Donde :

X = 2 **Y = 4** **Z = 10**

R = 4 / 2 + 2 * 4 ** 2 - 10 ⟵

Sustituimos valores, primero elevamos 4 a la potencia 2, por ser de mayor prioridad nos queda lo siguiente :

R = 4 / 2 + 2 * 16 - 10 ⟵

La multiplicación y división tienen la misma prioridad, por lo tanto, se ejecuta de izquierda a derecha, en este caso. primero divide 4 entre 2 y quedaría :

R = 2 + 2 * 16 - 10 ⟵ Multiplicamos 2 por 16 por ser de mayor prioridad que la suma y la resta.

R = 2 + 32 - 10 ⟵ La suma y resta tienen la misma prioridad, por lo tanto, se realiza de izquierda a derecha,haciendo primero la suma de 2 mas 32.

R = 34 - 10 ⟵ Por último restamos 34 menos 10.

R = 24 ⟵ El resultado final es 24.

Ejemplo 3.

Vamos a realizar este ejemplo, pero utilizando paréntesis.

R = T * Z + (Y + H * (A + B - C) / 2) * 10

Donde :

A = 10 B = 15 C = 5 H = 5 T = 2
Y = 10 Z = 5

Sustituimos los valores :

R = 2 * 5 + (10 + 5 * (10 + 15 - 5) / 2) * 10

Primero, el computador ejecuta el paréntesis más interno, realizando la suma de 10 + 15.

R = 2 * 5 + (10 + 5 * (25 - 5) / 2) * 10

Sigue ejecutando el paréntesis más interno, o sea resta 25 - 5.

R = 2 * 5 + (10 + 5 * 20 / 2) * 10

*Terminó de ejecutar el paréntesis interno, sigue con la multiplicación de 5 * 20 del paréntesis externo.*

R = 2 * 5 + (10 + 100 / 2) * 10

Continúa con la división de 100 entre 2.

R = 2 * 5 + (10 + 50) * 10

Realiza la suma que está dentro del paréntesis

R = 2 * 5 + 60 * 10

*Terminó de ejecutar los paréntesis, y realiza la operación de multiplicación de 2 * 5*

R = 10 + 60 * 10

*Ejecuta la multiplicación de 60 * 10*

R = 10 + 600

Por último, hace la suma de 10 + 600

R = 610

Por último quiero comentar lo siguiente :

Se preguntará, de que me sirve saber hacer las operaciones, si para eso cuento con la computadora ?

Bueno !, muchas veces, vamos a tener que crear las operaciones o fórmulas que tendrá que alimentar al computador, por cierta necesidad del programa que estemos realizando.

Imaginese !, que al obtener el sueldo neto del trabajador se lo pueda duplicar, por el mal acomodamiento de las variables.

Esta es la razón de la importancia de conocer el manejo de las prioridades en las operaciones aritméticas.

4.5.2 *Operadores* RELACIONALES

Los operadores relacionales comparan una expresión numérica o una expresión string. Y como conclusión nos dirá si es **falsa o verdadera**.

TABLA DE OPERADORES RELACIONALES		
OPERADOR	**EJEMPLO**	**SIGNIFICADO**
=	A = B	A IGUAL A B
<	A < B	A MENOR A B
>	A > B	A MAYOR A B
<= ó =<	A <= B	A MENOR IGUAL A B
	A =< B	A IGUAL MENOR A B
>= ó =>	A >= B	A MAYOR IGUAL A B
	A => B	A IGUAL MAYOR A B
<> ó ><	A <> B	A NO ES GUAL A B
	A >< B	
= =	A == B	A CONTIENE LO MISMO QUE B

Ejemplos :

a).- **IF A < B THEN** ⟶ Salida verdadera

↓
Salida falsa

b).- **IF A > B THEN**

c).- **IF A => B THEN**

4.5.3 *OPERADORES* LOGICOS

Los operadores lógicos son utilizados cuando trabajemos con más de una condición. Como conclusión me dirá si es fálsa o verdadera.

TABLA DE OPERADORES LOGICOS		
OPERADOR	EJEMPLO	SIGNIFICADO
NOT	NOT A%	EL OPUESTO LOGICO DE A
AND	A AND B	EL PRODUCTO DE A Y B
OR	A OR B	LA SUMA LOGICA DE A Y B
XOR	A XOR B	EL PRIVATIVO LOGICO DE A Y B
EQV	A EQV B	LA EQUIVALENCIA LOGICA DE A Y B
IMP	A IMP B	LA IMPLICACION LOGICA DE A Y B

Ejemplos :

a).- **IF A › B OR X = Z THEN** ⟶ Verdadera
↓
Falsa

b).- **IF A => B AND X = T OR Y ‹ Z THEN**

c).- **IF H=U OR G=O OR S›G OR G‹H THEN**

d).- **IF X = 20 OR Y = 4 THEN**

e).- **IF SEXO$ = "F" AND EDAD = 22 THEN**

AUTOEVALUACION DEL CONOCIMIENTO

1.- *Mencione los componentes de un* **PROGRAMA.**

2.- *Qué es una* **CONSTANTE** *?*

3.- *Explique qué es una* **EXPRESION.**

4.- *Qué es una* **VARIABLE** *?*

5.- *Explique los tipos de variables que puede trabajar* **BASIC.**

6.- *Mencione las caracteristicas de las* **VARIABLES.**

7.- *Distinga las siguientes variables que se presentan :*

R = Real o Flotante E = Entera A = Alfanumérica o String

CALIF ( )	**NOMB$ ( )**	**SUMA ( )**	**AU$ ( )**
SDONTO ( )	**PROM ( )**	**COMIS ( )**	**JESP$ ( )**
EDAD% ( )	**PESO% ( )**	**SUE% ( )**	**VTA ( )**
FECHA$ ( )	**H3 ( )**	**EGH$ ( )**	**C4% ( )**
HR ( )	**A$ ( )**	**NOMINA ( )**	**X% ( )**

8.- *Mencione los 3 tipos de operadores y especifique los operadores de cada uno de ellos.*

1.- _____

2.- _____

3.- _____

9.- *Obtenga el valor de " R " utilizando las prioridades de los operadores aritméticos.*

Donde :

A = 10	**B = 2**	**C = 5**	**D = 3**
E = 2	**X = 15**	**Y = 4**	**Z = 20**

Operaciones :

a).- **R = X + Y * D * Z / 2**

b).- **R = A * B - X / C + Y * * 2**

c).- **R = X + A * (Z - X - E * (E + B * * 3) / A)**

d).- **R = Z * Y * B / E + (A - C + E) - Y**

CAPITULO 5

ESTATUTOS PARA LA PROGRAMACION BASIC

OBJETIVO GENERAL

Describir en forma detallada los estatutos más frecuentes usados en la programación **BASIC.**

OBJETIVO ESPECIFICO

Existe una explicación concreta de cada uno de los estatutos **BASIC,** *con los cuales el alumno tendrá oportunidad de probar la función de estos en el laboratorio de sistemas; también observará la codificación de los estatutos para un mejor aprendizaje.*

NOTA :

Algunos estatutos se utilizan de varios maneras y el alumno podrá trabajarlos dependiendo de sus requerimientos o necesidades.

7.- *Distinga las siguientes variables que se presentan :*

R = Real o Flotante E = Entera A = Alfanumérica o String

CALIF ( )	**NOMB$ ( )**	**SUMA ( )**	**AU$ ( )**
SDONTO ( )	**PROM ( )**	**COMIS ( )**	**JESP$ ( )**
EDAD% ( )	**PESO% ( )**	**SUE% ( )**	**VTA ( )**
FECHA$ ( )	**H3 ( )**	**EGH$ ( )**	**C4% ( )**
HR ( )	**A$ ( )**	**NOMINA ( )**	**X% ( )**

8.- *Mencione los 3 tipos de operadores y especifique los operadores de cada uno de ellos.*

1.- _____

2.- _____

3.- _____

9.- *Obtenga el valor de " R " utilizando las prioridades de los operadores aritméticos.*

Donde :

A = 10	**B = 2**	**C = 5**	**D = 3**
E = 2	**X = 15**	**Y = 4**	**Z = 20**

Operaciones :

a).- **R = X + Y * D * Z / 2**

b).- **R = A * B - X / C + Y * * 2**

c).- **R = X + A * (Z - X - E * (E + B * * 3) / A)**

d).- **R = Z * Y * B / E + (A - C + E) - Y**

CAPITULO 5

ESTATUTOS PARA LA PROGRAMACION BASIC

OBJETIVO GENERAL

Describir en forma detallada los estatutos más frecuentes usados en la programación **BASIC.**

OBJETIVO ESPECIFICO

Existe una explicación concreta de cada uno de los estatutos **BASIC,** *con los cuales el alumno tendrá oportunidad de probar la función de estos en el laboratorio de sistemas; también observará la codificación de los estatutos para un mejor aprendizaje.*

NOTA :

Algunos estatutos se utilizan de varios maneras y el alumno podrá trabajarlos dependiendo de sus requerimientos o necesidades.

5.1 ESPECIFICACION DE COMENTARIOS

5.5.1 ESTATUTO REM o !

El estatuto **REM** o **!**, *nos sirve para poder hacer comentarios del programa que estamos realizando.*

Formatos :

# DE LINEA	**REM**	**COMENTARIO**
# DE LINEA	**!**	**COMENTARIO**

Los comentarios que llegamos a especificar dentro del programa, no son procesados, es decir, que el computador los va a ignorar, puesto que no tienen ninguna función de ejecución.

También, tenemos la flexibilidad de utilizar comentarios en cualquier parte del programa y agregar comentarios cuantas veces sea necesario.

En los siguientes ejemplos, se muestra la manera de como se debe de codificar los comentarios.

Ejemplo 1.

```
10  REM     PROGRAMADOR : ELIZABETH
20  REM     MATRICULA   : 841202
30  REM     GRUPO       : 2 A
40  REM     OBJETIVO    : OBTENCION DE LA NOMINA
50  REM     FECHA       : 2-DIC-84
```

Ejemplo 2.

```
10  !       PROGRAMADOR : ALICIA
20  !       MATRICULA   : 640524
30  !       GRUPO       : 5C
40  !       OBJETIVO    : OBTENCION DE LAS COMISIONES
50  !       FECHA       : 24-MAY-64
```


Ejemplo 3.

Tambien pueden utilizarse el **REM** *y el signo de exclamación (!) juntos en un mismo programa.*

```
10 REM     PROGRAMADOR : ELIZABETH
20 REM     MATRICULA   : 841202
30 !       GRUPO       : 2 A
40 !       OBJETIVO    : OBTENCION DE LAS COMISIONES
50 !       FECHA       : 2-DIC-84
```

Ejemplo 4.

```
10    REM
20    REM
30     !
40     !
50 TOTNOM = 0      ! TOTNOM ES EL TOTAL DE NOMINA
60   H = 0         ! H ES EL CONTADOR DE HORAS
```

5.2 REALIZACION DE OPERACIONES

5.2.1 ESTATUTO LET

El estatuto **LET** *nos permite asignarle el valor a una variable.*

Formato :

# DE LINEA	**LET**	**EXP**

Donde :

EXP.- *Es una asignación constante o una expresión aritmética.*

El estatuto **LET** *puede utilizarse de la siguientes formas :*

1.- *Representando una ecuación aritmética.*

```
10  LET  R = X * Y + Z ** 2
```

2.- *Incrementando un valor a una variable (hoja).*

```
10  LET  HOJA = HOJA + 1
```

3.- *Inicializando una variable con un valor.*

```
10  LET  H% = 20
20  LET  N$ = "ANGELICA"
30  LET  X = 5.89
```

Ejemplo 1.

```
10 REM    PROGRAMADOR : JULIO CESAR
20 REM    OBJETIVO    : SUMA DE DOS NUMEROS (10 Y 20)
30 LET      N1%         = 10
40 LET      N2%         = 20
50 LET      R% = N1% + N2%
60 PRINT  "EL RESULTADO ES ";R%
70 END

RUN                    <------  Al ejecutar el programa nos
                                imprimirá lo siguiente :

EL RESULTADO ES 30
```

Ejemplo 2.

```
10  REM  PROGRAMADOR : MIGUEL ANGEL
20  REM  OBJETIVO  : PROMEDIO DE UN ALUMNO QUE TIENE
30  REM              LAS CALIFICACIONES 70.5,89.5,80
40  LET  C1 = 70.5
50  LET  C2 = 89.5
60  LET  C3 = 80
70  LET  PROM = (C1 + C2 + C3)/3
80  PRINT  "EL PROMEDIO DEL ALUMNO ES ";PROM
90  END

RUN                    <------  Si ejecutamos el programa
                                el resultado sería :

EL PROMEDIO DEL ALUMNO ES 80
```


Ejemplo 3.

```
10 REM  PROGRAMADOR : LUPITA
20 REM  OBJETIVO    : OBTENCION DE SUELDO NETO
30 LET  NOM$ = "LUPITA HERNANDEZ"
40 LET  HR% = 10      ! HR SON HORAS TRABAJADAS
50 LET  SDOHR = 1550.10
60 LET  SNTO = HR% * SDOHR  ! SNTO ES EL SUELDO NETO
70 PRINT "EL NOMBRE DEL EMPLEADO (a) ";NOMB$
80 PRINT "SU SUELDO NETO ES DE ";SNTO
90 END

RUN                    <------  Ejecutamos e imprimirá :

EL NOMBRE DEL EMPLEADO (a) LUPITA HERNANDEZ
SU SUELDO NETO ES DE 15501
```

5.3 ENTRADA DE DATOS

Existen varias formas de alimentar al computador los datos con los cuales va a trabajar. Por ejemplo, podemos utilizar los estatutos **READ-DATA, RESTORE, INPUT** *e inclusive el estatuto* **LET** *que anteriormente estudiamos.*

5.3.1 ESTATUTO READ - DATA

La función del estatuto **READ,** *es de leer las variables a las cuales se le asignan los datos por medio del estatuto* **DATA.**

Formato del estatuto **READ** *:*

```
# DE LINEA  READ  VARIABLE(S).
```

Donde :

VARIABLE(S) *.- Es una o más variables numéricas, enteras y/o alfanuméricas (string). Todas las variables deben de ir precedidas por una coma.*

Ejemplos :

```
100  READ     A$, X, T, TOTAL, Y%, NOMB$
200  READ     DIRE$, TELE$, SUELDO
300  READ     TOTNOM
```

El estatuto **DATA,** *es el que contiene los datos que van a ser asignados a las variables que se especifiquen en el estatuto* **READ.**

Formato del estatuto **DATA**

# LINEA	DATA	CONSTANTE(S)

Donde :

CONSTANTE(S).- *Es una o más constantes numéricas, enteras y/o alfanuméricas (string). En éstas últimas su dato debe de ir entre comillas. Todas las constantes deben de ir precedidas por una coma.*

Ejemplos :

```
2000   DATA     50.5, 300, "ELIZABETH"
3000   DATA     "PETROLEOS MEXICANOS", 800, 572.6
4000   DATA     12.8, 600
```

OBSERVACIONES DEL ESTATUTO READ

1.- No es válido, si tiene mas variables que constantes en el **DATA.**
2.- No es válido, si no existe por lo menos un estatuto **DATA.**
3.- Pueden existir varios estatutos **READ,** para un mismo estatuto **DATA.**
4.- Pueden ir en cualquier lugar del programa, inclusive en la línea con multiples estatutos.

OBSERVACIONES DEL ESTATUTO DATA

1.- Pueden existir varios estatutos **DATA,** para un mismo estatuto **READ.**
2.- Puede existir un **DATA** sin **READ,** pero es ignorado por el estatuto **DATA** por el **BASIC.**
3.- El estatuto **DATA** debe de ir solo en una una línea.
4.- Puede ir en cualquier lugar del programa

Ejemplo 1.

```
100  READ     A, B%, C$
200  DATA     5.2, 100, "KATIA"
300  PRINT    A
400  PRINT    B%
500  PRINT    C$
600  END
```

```
RUN                          <------  Ejecutamos el programa

 5.2
 100                         <------  Resultados
KATIA
```

Ejemplo 2.

```
100  READ     NOMB$, SDOHR, HR%
200  DATA     "AMERICA", 5000.00, 20
300  LET      SDONTO = SDOHR * HR%
400  PRINT    "EL SUELDO NETO DE ";NOMB$;" ES ";SDONTO
500  END      END
```

```
RUN                          <------  Ejecutamos el programa

EL SUELDO NETO DE AMERICA ES 100000
```

OBSERVACIONES :

En los ejemplos anteriores, vemos que el listado de datos especificado por el estatuto **DATA,** están listados en el mismo orden del tipo de variable que les corresponden. También, podemos observar que en las variables alfanuméricas o string (c$, nomb$) sus datos se encuentran entre comillas.

5.3.2 ESTATUTO RESTORE

La función de estatuto **RESTORE**, *es la de restaurar los datos ya leídos en un* **DATA**. *O sea, nos permite poder leer más de una vez los mismos datos en el mismo programa. El apuntador de datos se posiciona al principio del primer estatuto* **DATA** *que se encuentre en el programa.*

Formato de estatuto **RESTORE :**

# DE LINEA	RESTORE

Ejemplo :

```
100  READ      A, B, C
200  PRINT     "LOS NUMEROS DE A,B,C  SON ";A,B,C
300  RESTORE
400  READ      X, Y, Z
500  PRINT     "LOS VALORES DE X,Y,Z  SON ";X,Y,Z
600  DATA      10,20,30,40,50,60,70,80,90
700  END

RUN                          <------ Ejecutamos el progama

LOS VALORES DE A,B,C  SON  10        20        30
LOS VALORES DE X,Y,Z  SON  10        20        30
```

PROCEDIMIENTO DE EJECUCION.

Primero el **READ** *lee las variables A, B, C, ya reconocidas éstas, el* **DATA** *asigna los valores 10,20,30, a las variables respectivamente (A = 10, B = 20, C = 30), por lo tanto el apuntador del* **DATA** *queda indicando el valor 40, luego el computador ejecuta la instrucción 200 imprimiendo los resultados de A,B,C que son 10,20,y 30, ahora ejecuta la instrucción 300 que es el* **RESTORE** *y automáticamente el apuntador del* **DATA** *pasa a colocarse en el valor 10 del* **DATA**, *después ejecuta la instrucción 400 y lee mediante un nuevo* **READ** *las variables X,Y,Z reconociendolas y el* **DATA** *se encarga de asignarle los valores 10, 20, 30 (x = 10, y = 20, z = 30), ejecuta línea 500 imprimiendo los resultados de x, y, z, (10,20,30), ignora la línea 600 y termina el programa.*

5.3.3 ESTATUTO INPUT

El estatuto **INPUT** *nos permite asignar datos en el transcurso del programa cuando se está ejecutando. Al procesar un estatuto* **INPUT**, *el computador escribirá un signo de interrogación (?), de esta manera nos espera para que asignemos los datos.*

Formato del estatuto **INPUT :**

# DE LINEA	INPUT	VARIABLE (S)
# DE LINEA	INPUT	"LETRERO ";VARIABLE (S)

Donde :

VARIABLE (S) .- *Es una o más variables numéricas, enteras y/o alfanuméricas (string). Todas las variables deben de ir precedidas por una coma.*

"LETRERO" .- *Es un mensaje del tipo de dato que vamos alimentar.*

Ejemplos :

```
100  INPUT    A%
200  INPUT    X, Y$, W
300  INPUT    "CUAL ES EL NOMBRE DEL CLIENTE ";NOMB$
400  INPUT    "DAME LAS HORAS TRABAJADAS ";HR%
500  INPUT    "CUAL ES SU EDAD Y SEXO "ED%,SEX$
```

NOTA :

Es preferible utilizar el estatuto **INPUT** especificando un letrero para el mayor entendimiento del dato que vamos alimentar.

A continuación, vamos a ver unos ejemplos utilizando el estatuto **INPUT**, *también cómo nos pide los datos y cómo los almacenaremos.*

Ejemplo **1.**

```
100  INPUT    A, B, C
200  LET      S = A + B + C
300  PRINT    "EL RESULTADO DE LA SUMA ES ";S
400  END

RUN                          <------  Ejecutamos el programa

? 100,500,18     <CR>

EL RESULTADO DE LA SUMA ES  618
```

PROCEDIMIENTO DE EJECUCION :

1.- *En la línea 100, se encuentra el estatuto* **INPUT** *y automáticamente nos preguntará por los datos de A, B, C, apareciendo en la pantalla el signo de interrogación (?).*

2.- *Le alimentamos los datos 100, 500, 18 y se los asignará respectivamente a las variables del estatuto* **INPUT** *(A = 100, B = 500, C = 18).*

3.- *Ejecuta la operación aritmética que se encuentra en la línea 200 y el resultado de ésta lo almacenará en la variable* **S**.

4.- *Al ejecutar la línea 300, imprimirá el resultado de la variable* **S**, *pero antes pondrá el letrero :* (**EL RESULTADO DE LA SUMA ES**).

5.- *Termina la ejecución del programa.*

Otra forma de alimentar los datos a este programa sería :

```
RUN                          <------  Volvemos a ejecutarlo

? 100    <CR>                <------  Nos pregunta por los datos de A,B,C,pero si
                                      nadamás le alimentamos el dato que se le
                                      asignará a la variable "A", el computador
                                      volverá a poner el signo de interrogación
                                      (?) preguntando por los datos de "B" y "C"

? 500    <CR>                <------  Le asignamos el valor 500 para "B" y volverá
                                      a poner el signo de interrogación para la
                                      variable "C".

? 18                         <------  Asignamos el valor 18 a "C".

EL RESULTADO DE LA SUMA ES  618
```

Ejemplo 2.

En este ejemplo trabajaremos el estatuto **INPUT** *con letrero, el computador primeramente imprimirá el letrero especificado y despues pondrá el signo de interrogación (?).*

```
100  INPUT   "NOMBRE DEL EMPLEADO ";NOMB$
200  INPUT   "CUANTAS HORAS TRABAJO ";HR
300  INPUT   "CUAL ES SU SUELDO HORA ";SDOHR
400  LET     SDONTO = HR * SDOHR
500  PRINT   "EL EMPLEADO ES "; NOMB$
520  PRINT   "SU SUELDO NETO ES ";SDONTO
600  END

RUN                          <------  Ejecutamos el programa

NOMBRE DEL EMPLEADO ? LUIS CARLOS  <CR>
CUANTAS HORAS TRABAJO ? 45
CUAL ES SU SUELDO HORA ? 15000

EL EMPLEADO ES LUIS CARLOS
SU SUELDO NETO ES  675000
```

Ejemplo 3.

```
100  PRINT  "CONSTRUCTORA DEL NORTE S.A."
200  PRINT  "     DATOS  GENERALES      "
300  PRINT
400  INPUT  "NOMBRE -------- ";NOMB$
500  INPUT  "DIRECCION ----- ";DIRE$
600  INPUT  "TELEFONO ------ ";TELE$
700  INPUT  "EDAD ---------- ";ED%
800  INPUT  "SEXO ---------- ";SEX$
900  PRINT
1000 PRINT  NOMB$, DIRE$, TELE$, ED%, SEX$
1010 END
```

```
RUN                          <------  Ejecutamos el programa

CONSTRUCTORA DEL NORTE S.A.
     DATOS  GENERALES

NOMBRE -------- ? ANTONIO
DIRECCION ----- ? MORELOS 1098 OTE
TELEFONO ------ ? 70-08-07
EDAD ---------- ? 28
SEXO ---------- ? MASCULINO

ANTONIO     MORELOS 1098 OTE     70-08-07     28     MASCULINO
```

5.4 SALIDA DE IMPRESION

5.4.1 ESTATUTO PRINT

El estatuto **PRINT**, *nos permite desplegar datos (resultados) o letreros por la terminal, en el transcurso de la ejecución del programa.*

DISTINTAS FORMAS DE UTILIZAR EL **PRINT** :

```
# DE LINEA   PRINT   VARIABLE(S)
# DE LINEA   PRINT   "LETRERO"
# DE LINEA   PRINT   "LETRERO"; VARIABLE(S)
# DE LINEA   PRINT   VARIABLE1; VARIABLE2;.....;ETC.
# DE LINEA   PRINT   VARIABLE1, VARIABLE2,.....,ETC.
# DE LINEA   PRINT   TAB(N); VARIABLE(S)
# DE LINEA   PRINT   TAB(N); "LETRERO"
# DE LINEA   PRINT   TAB(N); "LETRERO"; VARIABLE(S)
```


Donde :

VARIABLES(S). - *Es una o más variables numéricas, enteras y/o alfanuméricas (string). Todas las variables deben de ir precedidas por una coma o punto y coma.*

"LETRERO" . - *Es un mensaje del tipo de dato que vamos a imprimir o simplemente un texto de información.*

. - *La función del punto y coma es imprimir los resultados de las variables de salida, separándolas dos espacios entre ellas.*

. - *La función de la coma es imprimir los resultados de las variables de salida, separándolas por zonas entre ellas.*

La terminal contiene zonas de 10 columnas de ancho en cada línea, tiene características de su funcionamiento por lo cual, el programador puede cambiar el número de zona dependiendo de su necesidad.

TAB(N) . - *Tabulador de espacios, donde " N " es el número de la columna requerida por el programador.*

Ejemplos :

```
100  PRINT  SDOTOT
100  PRINT  NOMB$; HORAS; SDOHR; SDONTO
100  PRINT  NUMPOLIZA, FECHA$, CAR.CRE, VALOR
100  PRINT  "EMPACADORA DE CARNES S.A. DE C.V."
100  PRINT  "TOTAL DE NOMINA "; TOTNOMI
100  PRINT  X; T; H%; W$
100  PRINT  H$, Y%, Z
```

Ejemplos de práctica :

Ejemplo 1.

```
100  PRINT  "CAPTURA DE DATOS GENERALES"
200  INPUT  "CUAL ES SU NOMBRE "; NOMB$
300  INPUT  "DEME SU DIRECCION "; DIRE$
400  INPUT  "ESCRIBA SU TELEFONO "; TELE$
500  INPUT  "CUAL ES SU PUESTO "; PUESTO$
600  INPUT  "QUE SUELDO NORMAL TIENE "; SDO
700  PRINT \ PRINT
800  PRINT  "COMPAÑIA ELIZONDO S.A."
900  PRINT  "REPORTE DE DATOS GENERALES"
1000 PRINT \ PRINT
1010 PRINT  "NOMBRE --------- "; NOMB$
1020 PRINT  "DIRECCION ------ "; DIRE$
1030 PRINT  "TELEFONO ------- "; TELE$
1040 PRINT  "PUESTO --------- "; PUESTO$
1050 PRINT  "SU SUELDO ES --- "; SDO
1060 END
```

```
RUN                          <------ Ejecutamos el programa

CAPTURA DE DATOS GENERALES

CUAL ES SU NOMBRE ? NORMA LETICIA GARZA
DEME SU DIRECCION ? AVE. 19 # 909
ESCRIBA SU TELEFONO ? 45-15-82
CUAL ES SU PUESTO ? DEPTO. DE VENTAS
QUE SUELDO NORMAL TIENE ? 750000

COMPAÑIA ELIZONDO S.A.
REPORTE DE DATOS GENERALES

NOMBRE --------- NORMA LETICIA GARZA
DIRECCION ------ AVE. 19 # 909
TELEFONO ------- 45-15-82
PUESTO --------- DEPTO. DE VENTAS
SU SUELDO ES ---  750000
```


Ejemplo 2.

```
100  INPUT  "DEME LA VENTA 1 ---- "; VTA1
110  INPUT  "CUAL ES LA VENTA 2 - "; VTA2
120  INPUT  "CUANTO EN LA VENTA 3 "; VTA3
130  INPUT  "DEME LA VENTA 4 ---- "; VTA4
140  INPUT  "CUAL FUE LA VENTA 5  "; VTA5
200  EMPRE$ = "CONSTRUCTORA ELECTRONICA S.A."
300  TOTVTA = VTA1 + VTA2 + VTA3 + VTA4 + VTA5
400  PRINT \ PRINT          ! BRINCA DOS RENGLON
500  PRINT  EMPRE$
550  PRINT \ PRINT
600  PRINT  VTA1, VTA2, VTA3, VTA4, VTA5
650  PRINT
700  PRINT  VTA1,,,VTA2
750  PRINT
800  PRINT  "TOTAL DE VENTAS "; TOTVTA
900  END
```

```
RUN                          <------ Ejecutamos el programa

DEME LA VENTA 1 ---- ? 1500
CUAL ES LA VENTA 2 - ? 3875.25
CUANTO EN LA VENTA 3 ? 600
DEME LA VENTA 4 ---- ? 12000
CUAL FUE LA VENTA 5  ? 890

CONSTRUCTORA ELECTRONICA S.A.

 1500          3875.25          600          12000          890
 1500                                       3875.25

TOTAL DE VENTAS  18865.3
```

Ejemplo 3.

En este ejemplo, observamos la diferencia de utilizar el estatuto TAB.

```
100  PRINT "NO ESTOY UTILIZANDO EL ESTATUTO TAB"
200  PRINT                          ! BRINCA UN RENGLON
300  PRINT "SUPER GARCIA S.A. DE C.V."
400  PRINT "ABARROTES EN GENERAL"
500  PRINT "MAYOREO Y MENUDEO"
600  PRINT \ PRINT                  ! BRINCA DOS RENGLONES
700  PRINT TAB(23);"SI ESTOY UTILIZANDO EL TAB"
800  PRINT                          ! BRINCA UN RENGLON
900  PRINT TAB(28);"SUPER GARCIA S.A. DE C.V."
1000 PRINT TAB(30);"ABARROTES EN GENERAL"
1100 PRINT TAB(31);"MAYOREO Y MENUDEO"
1200 PRINT \ PRINT \ PRINT     ! BRINCA 3 RENGLONES
1300 PRINT TAB(29);"OBSERVE LA DIFERENCIA"
1400 PRINT TAB(27);"EXISTE MEJOR PRESENTACION"
1500 PRINT TAB(36);"GRACIAS"
1600 END

RUN                          <------ Ejecutamos el programa

NO ESTOY UTILIZANDO EL ESTATUTO TAB

SUPER GARCIA S.A. DE C.V.
ABARROTES EN GENERAL
MAYOREO Y MENUDEO


SI ESTOY UTILIZANDO EL TAB

SUPER GARCIA S.A. DE C.V.
ABARROTES EN GENERAL
MAYOREO Y MENUDEO

OBSERVE LA DIFERENCIA
EXISTE MEJOR PRESENTACION
GRACIAS
```


5.4.2 ESTATUTO PRINT USING

El estatuto **PRINT USING**, *nos permite desplegar los resultados y letreros en forma editada. El utilizar este estatuto, nos ayuda a presentar en forma más entendible los resultados al usuario.*

Formato de codificación :

```
# DE LINEA   PRINT USING  "CAMPO(S)"; VARIABLE(S)
# DE LINEA   PRINT USING  "CAMPO(S)"; DATO(S)
```

Donde :

USING .- *Using significa usando.*

CAMPO(S) .- *Son los formatos especificados para desplegar los resultados.*

VARIABLE(S) .- *Son los que contienen los resultados que se van a desplegar, pueden ser flotantes, enteros o alfanuméricos.*

Ejemplo 3.

En este ejemplo, observamos la diferencia de utilizar el estatuto TAB.

```
100  PRINT "NO ESTOY UTILIZANDO EL ESTATUTO TAB"
200  PRINT                          ! BRINCA UN RENGLON
300  PRINT "SUPER GARCIA S.A. DE C.V."
400  PRINT "ABARROTES EN GENERAL"
500  PRINT "MAYOREO Y MENUDEO"
600  PRINT \ PRINT                  ! BRINCA DOS RENGLONES
700  PRINT TAB(23);"SI ESTOY UTILIZANDO EL TAB"
800  PRINT                          ! BRINCA UN RENGLON
900  PRINT TAB(28);"SUPER GARCIA S.A. DE C.V."
1000 PRINT TAB(30);"ABARROTES EN GENERAL"
1100 PRINT TAB(31);"MAYOREO Y MENUDEO"
1200 PRINT \ PRINT \ PRINT     ! BRINCA 3 RENGLONES
1300 PRINT TAB(29);"OBSERVE LA DIFERENCIA"
1400 PRINT TAB(27);"EXISTE MEJOR PRESENTACION"
1500 PRINT TAB(36);"GRACIAS"
1600 END

RUN                          <------ Ejecutamos el programa

NO ESTOY UTILIZANDO EL ESTATUTO TAB

SUPER GARCIA S.A. DE C.V.
ABARROTES EN GENERAL
MAYOREO Y MENUDEO


SI ESTOY UTILIZANDO EL TAB

SUPER GARCIA S.A. DE C.V.
ABARROTES EN GENERAL
MAYOREO Y MENUDEO

OBSERVE LA DIFERENCIA
EXISTE MEJOR PRESENTACION
GRACIAS
```


5.4.2 ESTATUTO PRINT USING

El estatuto **PRINT USING**, *nos permite desplegar los resultados y letreros en forma editada. El utilizar este estatuto, nos ayuda a presentar en forma más entendible los resultados al usuario.*

Formato de codificación :

```
# DE LINEA   PRINT USING  "CAMPO(S)"; VARIABLE(S)
# DE LINEA   PRINT USING  "CAMPO(S)"; DATO(S)
```

Donde :

USING .- *Using significa usando.*

CAMPO(S) .- *Son los formatos especificados para desplegar los resultados.*

VARIABLE(S) .- *Son los que contienen los resultados que se van a desplegar, pueden ser flotantes, enteros o alfanuméricos.*

A CONTINUACION ESTUDIAREMOS LOS DISTINTOS FORMATOS DE CAMPO QUE PODEMOS UTILIZAR CON EL ESTATUTO **PRINT USING.**

CARACTER	SIGNIFICADO	EJEMPLO	DESCRIPCION
#	SIGNO NUMERICO	#####	SE UTILIZA PARA IMPRIMIR VALORES NUMERICOS.
\	BACK SLASH	\ \	UTILIZADO PARA VARIABLES ALFANUMERICAS.
.	PUNTO	###.##	PARA IMPRIMIR UN VALOR CON DECIMALES.
,	COMA	##,###.##	IMPRIME CANTIDADES SEPARADAS POR COMAS.
-	GUION	###,##-	IMPRIME CANTIDADES NEGATIVAS, SE PUEDE UTILIZAR DE VARIAS FORMAS.
$	SIGNO DE DOLLAR	$$#,###.##	IMPRIME CANTIDADES CON SIGNO DE DOLLAR.
*	ASTERISCO	**##.##	EL ASTERISCO SIGNIFICA PROTECCION.
'	APOSTROFE		PARA ESPECIFICAR VALORES JUSTIFICADOS.
'L	APOSTROFE LEFT	'LLLLL	JUSTIFICA EL VALOR DEL RESULTADO A LA IZQUIERDA DEL CAMPO ESPECIFICADO.
'R	APOSTROFE RIGHT	'RRRRR	JUSTIFICA EL VALOR DEL RESULTADO A LA DERECHA DEL CAMPO ESPECIFICADO.
'C	APOSTROFE CENTER	'CCCCC	JUSTIFICA EL VALOR DEL RESULTADO AL CENTRO DEL CAMPO ESPECIFICADO.

Ejemplos de **CAMPOS NUMERICOS (#)**

```
100  A% = 5271
110  B% = 300
120  PRINT USING "##### "; A%
130  PRINT USING "##### "; B%
140  END

RUN

 5271
  300
```

Si el formato del campo especificado es más grande que el valor que se imprimirá, el resultado quedará justificado a la derecha como lo muestra el ejemplo anterior.

OTRA FORMA DE PODER PRESENTAR SUS RESULTADOS

```
100  A% = 5271
110  B% =  300
120  PRINT USING  "#####    ##### "; A%,B%
130  END

RUN

 5271      300
```

OBSERVACION :

En el formato de impresión se tiene 4 espacios en blanco, sin embargo en el resultado aparecen 6, esto se debe a que el valor de B% es más pequeño que el campo de impresión, puesto que su valor fue justificado a la derecha.

```
100  SDOHOR = 32784
110  HRSTRA = 48
120  SDONTO = SDOHOR * HRSTRA
130  PRINT USING "######  ##  ######## "; &
     SDOHR,HRSTRA,SDONTO
140  END

RUN

 32784  48   1573630
```

Al campo se le puede agregar un mensaje, por ejemplo :

```
100  C1 = 100
200  C2 = 90
300  C3 = 80
400  PROME = (C1 + C2 + C3) / 3
500  PRINT USING "LAS CALIFICACIONES SON "+&
     "  ###  ###  ###"; C1,C2,C3
600  PRINT USING "EL PROMEDIO DEL ALUMNO FUE "+&
     "  ###"; PROME
700  END

RUN

LAS CALIFICACIONES SON   100   90   80
EL PROMEDIO DEL ALUMNO FUE    90
```

Ejemplos para utilizar el BACK SLASH (\ )

```
100  L$ = "EMPACADORA DEL NORTE"
110  PRINT USING "\                    \"; L$
120  END

RUN

EMPACADORA DEL NORTE
```

Al especificar un formato de campo alfanumérico, los BACK SLASH (\ ) *son ocupados por un caracter, en el ejemplo anterior, la primera y última letra del letrero son impresas en las posiciones de los* BACK SLASH (\ ) *y las demás en los espacios correspondientes.*

Observemos que pasa, si el formato de campo es más pequeño al valor que se va a imprimir en el siguiente ejemplo :

```
100  ENCA$ = "PETROLEOS MEXICANOS S.A."
110  PRINT USING "\          \"; ENCA$
120  END

RUN

PETROLEOS MEX
```


Observamos que el letrero fué truncado por el campo de impresión, puesto que al trabajar el **PRINT USING** lleva mayor prioridad sus formatos de campo.

Ejemplo, cuando el formato de campo es mayor a los valores.

```
100  NOMB1$ = "LAURA"
110  NOMB2$ = "CLAUDIA"
120  PRINT USING "\       \    \       \"; &
     NOMB1$,NOMB2$
130  END

RUN

LAURA        CLAUDIA
```

Como los formatos de campo son más grandes que los nombres de Laura y Claudia, éstos fueron justificados a la izquierda de los campos y se completa con blancos a la derecha.

```
100  VEND$ = "ANABEL TREVIÑO"
110  VENTA = 5000
120  PORCE = .20
130  COMIS = VENTA * PORCE
140  PRINT USING "\          \ VENDIO ####  "+ &
     "COMISION ####"; VEND$, VENTA, COMIS
150  END

RUN

ANABEL TREVINO  VENDIO 5000  COMISION 1000
```

Ejemplos para la utilización del PUNTO (.)

El punto se utiliza para poder imprimir cantidades numéricas con el punto decimal.

Ejemplo :

```
100  VENTA = 32650.78
200  COMIS = 7548.65
300  SDOHR = 15720.50
400  TOTNOM = 1197840.00
500  PRINT USING  "#######.##"; VENTA
600  PRINT USING  "#######.##"; COMIS
700  PRINT USING  "#######.##"; SDOHR
800  PRINT USING  "#######.##"; TOTNOM
900  END

RUN

  32650.80
   7548.65
  15720.50
1197840.00
```

Si observamos el ejemplo anterior, los resultados fueron justificados a la derecha.

Al utilizar el punto en el estatuto PRINT USING, *solamente se agrega en el formato de campo.*

```
100  CLAVE% = 248
110  ARTICU$ = "TORNILLOS DE 1/2 PULG."
120  CANT = 10
130  PRECIO = 500
140  PAGO = CANT * PRECIO
150  PRINT USING "\                    \  "+ &
     "#####.##"; ARTICU$, PAGO
160  END

RUN

TORNILLOS DE 1/2 PULG.    5000.00
```


Ejemplo para la utilización de la COMA (,)

Cuando deseamos que la cantidad que vamos a imprimir esté separada por comas (,) en el formato de impresión.

```
100  PRINT USING  "##,###"; 30000
200  PRINT USING  "#,###,###"; 1000000
300  PRINT USING  "##,###.##"; 7455.80
400  PRINT USING  "#,###.##"; 24.95
500  END

RUN

30,000
1,000,000
  7,455.80
     24.95
```

```
100  A = 30000
200  B = 1000000
300  C = 7455.80
400  D = 24.95
500  PRINT USING  "##,###,###.##"; A
600  PRINT USING  "##,###,###.##"; B
700  PRINT USING  "##,###,###.##"; C
800  PRINT USING  "##,###,###.##"; D
900  END

RUN

    30,000.00
 1,000,000.00
     7,455.80
        24.95
```

Ejemplo para la utilización del GUION (-)

Se utiliza el guión (-), cuando la cantidad que esperamos como resultado puede llegar a ser negativa.

```
100  PRINT USING  "#####.##-"; 35.92
110  PRINT USING  "#####.##-"; 578.24
120  PRINT USING  "#####.##-"; -312
130  PRINT USING  "#####.##-"; -97456.8
140  PRINT USING  "#####.##-"; -284.75

RUN

   35.92
  578.24
  312.00-
97456.80-
  284.75-
```

En el ejemplo anterior vamos a explicar cada una de las instrucciones :

1.-*En la línea 100, como el valor es más pequeño que el campo especificado, el valor es justificado a la derecha y no lleva el signo negativo por ser una cantidad positiva.*

2.-*En la línea 110, ocurre lo mismo que la línea anterior.*

3.-*La línea 120, justifica el valor a la derecha, como no tiene la cantidad de decimales, agrega los ceros (0) y por último como es negativa la cantidad imprime el signo en el resultado.*

4.-*En la instrucción 130, no justifica, puesto que el tamaño de la cantidad es igual a la del campo, imprime el signo por ser negativo el número, agrega el último decimal como cero (0).*

5.-*En la instrucción 140, justifica el valor a la derecha e imprime el signo por ser negativa la cantidad.*

Ejemplos para la utilizar el SIGNO DE DOLLAR ($)

Cuando deseamos que la cantidad la imprima con signo de dollar, simplemente se le agrega en el formato de campo.

```
100  PRINT USING  "$####.##"; 3462.20
200  PRINT USING  "$####.##"; 50.25
300  PRINT USING  "$####.##"; 200.00
400  END

RUN

$3462.20
$  50.25
$ 200.00
```

OBSERVACIONES DEL EJEMPLO :

Al utilizar solamente un signo de dollar ($) :

1.- Si el tamaño de la cantidad es igual al formato de campo, el signo de dollar quedaría pegado a la cantidad como en la línea 100.

2.- Si la cantidad es más pequeña que el formato de campo, el valor será justificado a la derecha, dejando espacios en blanco hasta el signo de dollar ($), como en la instrucción 110 y 120.

```
100  PRINT USING  "$$####.##"; 15467.26
110  PRINT USING  "$$####.##"; 8429.32
120  PRINT USING  "$$####.##"; 21.95
130  PRINT USING  "$$####.##"; 450.00
140  PRINT USING  "$$####.##"; 1.76
150  END

RUN

$15467.30
 $8429.32
   $21.95
  $450.00
    $1.76
```

OBSERVACION DEL EJEMPLO ANTERIOR :

*Al utilizar dos signos de dollar (**$$**)*

1.- Si la cantidad es más pequeña que el formato de campo, el signo de dollar quedará pegado a la cantidad, imprimiendo solamente un signo, como en los casos de las lineas 120, 130 y 140.

Ejemplo utilizando los siguientes signos : (. , - # $)

```
100  PRINT USING  "$$#,###.##";   35832.25
110  PRINT USING  "$$#,###.##";   154.78
120  PRINT USING  "$$#,###.##";   7962.64
130  PRINT USING  "$$#,###.##-";  -1361.37
140  PRINT USING  "$$#,###.##-";  -100
150  END

RUN

$35,832.30
  $154.78
$7,962.64
$1,361.37-
  $100.00-
```


Ejemplo para utilizar el **SIGNO ASTERISCO (*)**

*Al signo asterisco (*) se le considera como una protección al valor que se va a imprimir.*

```
100  PRINT USING  "**#####.##";  57824.96
200  PRINT USING  "**#####.##";  42658
300  PRINT USING  "**#####.##";  1794
400  PRINT USING  "**#####.##";  3811.4
500  PRINT USING  "**#####.##";  254
600  PRINT USING  "**#####.##";  168.2
700  PRINT USING  "**#####.##";  223.45
800  PRINT USING  "**#####.##";  9.7
900  PRINT USING  "**#####.##";  8
910  PRINT USING  "**#####.##-"; -200
920  PRINT USING  "**#####.##-"; -1.5
930  END

RUN

57825.96
42658.00
*1794.00
*3811.40
**254.00
**168.20
**223.45
****9.70
****8.00
**200.00-
****1.50-
```

OBSERVACIONES :

1.- Si la cantidad es más pequeña que el formato de campo, es completada por asteriscos, como en las líneas 300 a la 920.

2.- Si no existen decimales en la cantidad, se imprimen ceros (0).

3.- Si la cantidad es negativa, en la impresión aparecerá el signo, siempre y cuando se especifiquen en el formato de campo.

Ejemplo para utilizar el **SIGNO APOSTROFE (')**

El apóstrofe nos indica que los valores a imprimir pueden ser justificados a la izquierda, derecha o centro. El apóstrofe puede ser utilizado solamente para imprimir variables alfanuméricas o string.

El apóstrofe es ocupado por un caracter en la impresión como en los ejemplos siguientes :

```
100  PRINT USING  " ' "; "AMALIA"
200  END

RUN

 A


100  PRINT USING  " ' " ; "PEMEX S.A."
200  END

RUN

 P
```

Ejemplo para utilizar el formato de **JUSTIFICACION A LA IZQUIERDA ('L)**

```
100  PRINT USING "'LLLLLLLLL"; "EMPRESA ELECTRONICA"
200  PRINT USING "'LLLLLLLLL"; "HOLA"
300  PRINT USING "'LLLLLLLLL"; "JOHNNY"
400  END

RUN

EMPRESA EL
HOLA
JOHNNY
```

OBSERVACION :

Todos los resultados están justificados a la izquierda.

Ejemplo para utilizar el formato de **JUSTIFICACION A LA DERECHA ('R)**

```
100 PRINT USING "'RRRRRRRRR"; "ANA MARIA"
200 PRINT USING "'RRRRRRRRR"; "FERRETERIA DEL NORTE"
300 PRINT USING "'RRRRRRRRR"; "UNI"
400 END

RUN

 ANA MARIA
FERRETERIA
      UNI
```

OBSERVACION :

Los resultados están justificados a la derecha.

Ejemplo para utilizar el formato de **JUSTIFICACION AL CENTRO ('C)**

```
100  PRINT USING  "'CCCCCCCCC"; "U"
200  PRINT USING  "'CCCCCCCCC"; "UN"
300  PRINT USING  "'CCCCCCCCC"; "UNI"
500  END

RUN

   U
  UN
 UNI
```

Ejemplo para justificar :

```
100  PRINT USING "'LLLLLLLLLLLLLL"; "PEMEX S.A."
200  PRINT USING "'LLLLLLLLLLLLLL"; "COMPRAS"
300  PRINT USING "'LLLLLLLLLLLLLL"; "AL DIA"
400  PRINT \ PRINT
500  PRINT USING "'RRRRRRRRRRRRRR"; "PEMEX S.A."
600  PRINT USING "'RRRRRRRRRRRRRR"; "COMPRAS"
700  PRINT USING "'RRRRRRRRRRRRRR"; "AL DIA"
800  PRINT \ PRINT
900  PRINT USING "'CCCCCCCCCCCCCC"; "PEMEX S.A."
910  PRINT USING "'CCCCCCCCCCCCCC"; "COMPRAS"
920  PRINT USING "'CCCCCCCCCCCCCC"; "AL DIA"
930  END

RUN

PEMEX S.A.
COMPRAS
AL DIA

   PEMEX S.A.
      COMPRAS
       AL DIA

PEMEX S.A.
   COMPRAS
   AL DIA
```


5.5 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Explique el estatuto* **REM.**

2.- *Para qué sirve el estatuto* **LET.**

3.- *Qué contiene el estatuto* **DATA.**

4.- *Explique el estatuto* **INPUT.**

5.- *Mencione las diferentes formas de utilzar el* **PRINT** *y explíquelas.*

6.- *Para qué sirve utilizar el* **PRINT USING.**

7.- *Escriba 5 diferentes formatos de campo que se utilizarán con el estatuto* **PRINT USING.**

8.- *Cuál es la función del* **'L, 'R,** *y* **'C** *en el* **PRINT USING.**

CAPITULO 6

TRANSFERENCIA DE CONTROL

OBJETIVO GENERAL

Utilizar los diferentes estatutos de transferencia de control, así como el empleo de subrutinas dentro de un programa **BASIC.**

OBJETIVO ESPECIFICO

En éste capítulo, se aprenderá las transferencias que pueden utilizarse dentro del lenguaje **BASIC** *y las distintas formas de poder codificar estos estatutos dentro de un programa.*

Ademas utilizará el manejo, función y codificación de los estatutos para utilizar subrutinas locales y recicladores.

7.- *Escriba 5 diferentes formatos de campo que se utilizarán con el estatuto* **PRINT USING.**

8.- *Cuál es la función del* **'L, 'R,** *y* **'C** *en el* **PRINT USING.**

CAPITULO 6

TRANSFERENCIA DE CONTROL

OBJETIVO GENERAL

Utilizar los diferentes estatutos de transferencia de control, así como el empleo de subrutinas dentro de un programa **BASIC.**

OBJETIVO ESPECIFICO

En éste capítulo, se aprenderá las transferencias que pueden utilizarse dentro del lenguaje **BASIC** *y las distintas formas de poder codificar estos estatutos dentro de un programa.*

Ademas utilizará el manejo, función y codificación de los estatutos para utilizar subrutinas locales y recicladores.

6.1 TRANSFERENCIA DE CONTROL INCONDICIONAL GOTO

6.1.1 GOTO

El estatuto **GOTO**, *transfiere el control del programa a una línea deseada continuando con el proceso. La transferencia puede realizarse a un # de línea ascendente o descendente. Es recomendable que un programa no lleve muchos estatutos* **GOTO.**

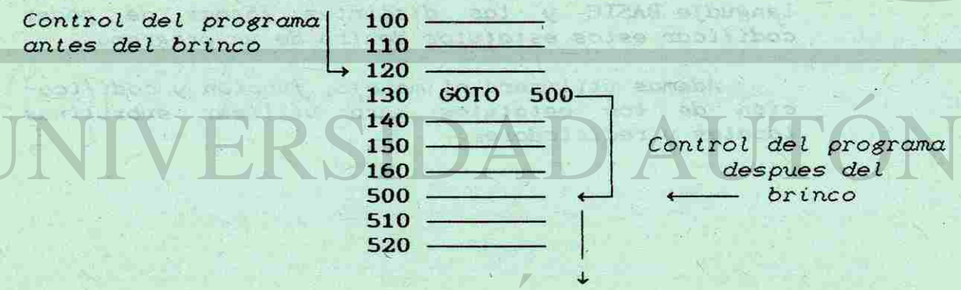
Formato :

GOTO # DE LINEA

Donde :

 # DE LINEA .- *Es la siguiente línea del programa que será ejecutada.*

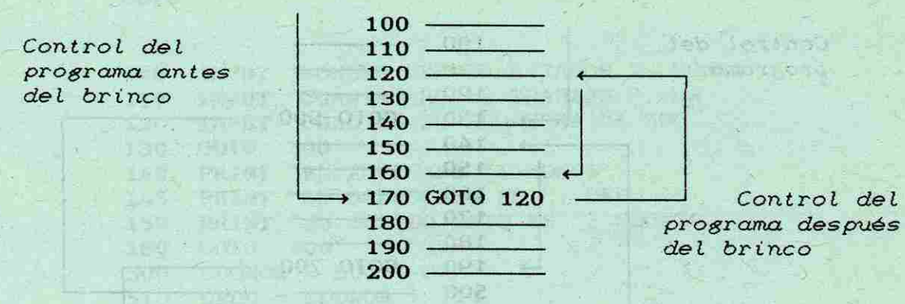
Ilustración 1.



PROCEDIMIENTO DE EJECUCION

Primero ejecuta las líneas 100, 110, 120 y 130.
Brinca y continúa con las líneas 500, 510 y 520.

Ilustración 2.



PROCEDIMIENTO DE EJECUCION

Primeramente ejecuta la líneas 100, 110, 120, 130, 140, 150, 160 y 170.

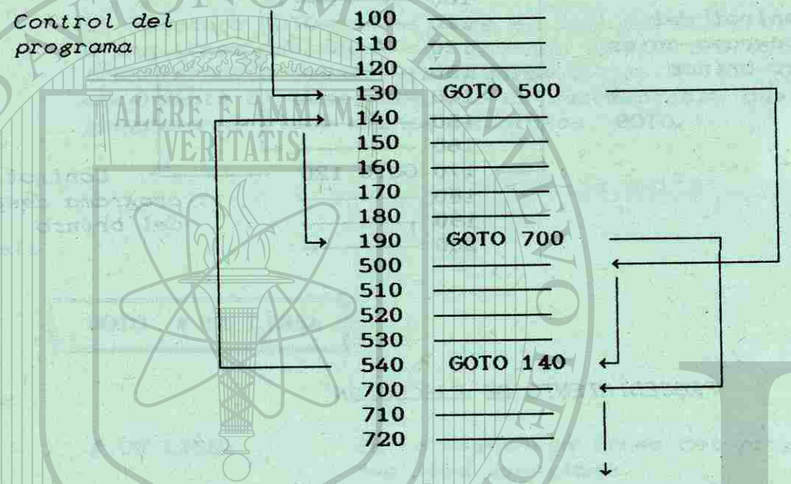
Brinca y continúa con las líneas 120, 130, 140, 150 y 160.

OBSERVACION :

El ejercicio anterior terminará ciclado, o sea que el computador estaría ejecutando las líneas desde la 120 a la 170.

Ilustración 3.

INSTRUCCIONES:



PROCEDIMIENTO DE EJECUCION.

1.- Ejecuta las líneas 100, 110, 120 y 130.

2.- Brinca continuando con la 500, 510, 520, 530 y la 540.

3.- Después ejecuta las líneas 140, 150, 160, 170, 180 y la 190. Por el brinco que se encuentra en la línea 540.

4.- Por último ejecutará las líneas 700, 710 y 720.

Ejemplo 1.

Vamos a obtener el sueldo neto de un trabajador quitándole una deducción del 10% de su sueldo normal

Este ejemplo es para observar el trabajo del estatuto **GOTO.**

```
100  INPUT "NOMBRE DEL TRABAJADOR "; NOMB$
110  INPUT "CUANTAS HORAS TRABAJO "; HR
120  INPUT "CUAL ES SU SDO-HORA "; SDO
130  GOTO  500
140  PRINT "EL EMPLEADO "; NOMB$
145  PRINT "SU DEDUCCION ES "; DEDU
150  PRINT "SU SUELDO NETO ES "; SDONTO
160  GOTO  900
500  SDONOR = SDO * HR
510  DEDU = SDONOR * .10
520  SDONTO = SDONOR - DEDU
530  GOTO  140
900  END

RUN

NOMBRE DEL TRABAJADOR ? CARLOS
CUANTAS HORAS TRABAJO ? 48
CUAL ES SU SDO-HORA ? 5000

EL EMPLEADO CARLOS
SU DEDUCCION ES  24000
SU SUELDO NETO ES  216000
```

6.2 TRANSFERENCIA DE CONTROL CONDICIONAL

Existen dos estatutos de transferencia condicional, el **ON GOTO** *y el* **IF THEN ELSE.**

La transferencia (brinco) se realiza mediante una condición.

A continuación vamos a describir el fundamento de estos estatutos.

6.2.1 ON GOTO

El estatuto **ON GOTO**, *transfiere (brinca) el control del programa a una línea determinada, dependiendo del valor que tenga la expresión, continuando con el proceso. La transferencia puede ser a un # de línea ascendente o descendente.*

Formato :

ON EXPRESION GOTO LIN(1),LIN(2),.....,LIN(N)

Donde :

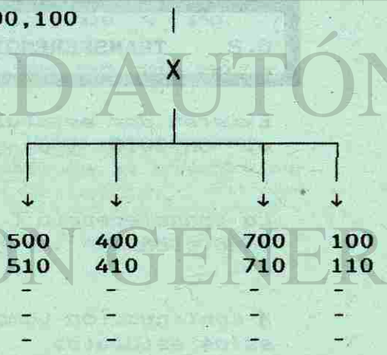
EXPRESION.- *Es la transferencia. Puede ser una variable flotante o entera, es recomendable que sea entera, para protegernos de la flotante en caso de que tomara un valor con decimal.*

LIN(1),
LIN(2),
LIN(3).- *Son los números de lineas a donde se puede hacer la transferencia y éstos deben de ir precedidos por una coma.*

Ilustración 1.

INSTRUCCIONES

```
100 ———
110 ———
120 ———
130 ———
140 ON X GOTO 500,400,700,100
400 ———
410 ———
500 ———
510 ———
520 ———
700 ———
710 ———
720 ———
730 ———
```



PROCEDIMIENTO DE EJECUCION

1.- Ejecuta las líneas 100,110,120,130 Y 140.

2.- Si X = 1, brinca a la **500** y continúa.

3.- Si X = 2, brinca a la **400** y continúa.

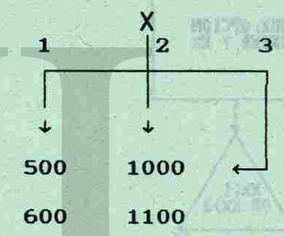
4.- Si X = 3, brinca a la **700** y continúa.

5.- Si X = 4, brinca a la **100** y continúa.

Ilustración 2.

INSTRUCCIONES

```
100 ———
200 ———
300 ON X GOTO 500, 1000, 1100
400 ———
500 ———
600 ———
1000 ———
1100 ———
1200 ———
```



PROCEDIMIENTO DE EJECUCION

1.- Ejecuta las líneas 100, 200 y 300.

2.- Si X = 1 brinca a la **500** y continúa.

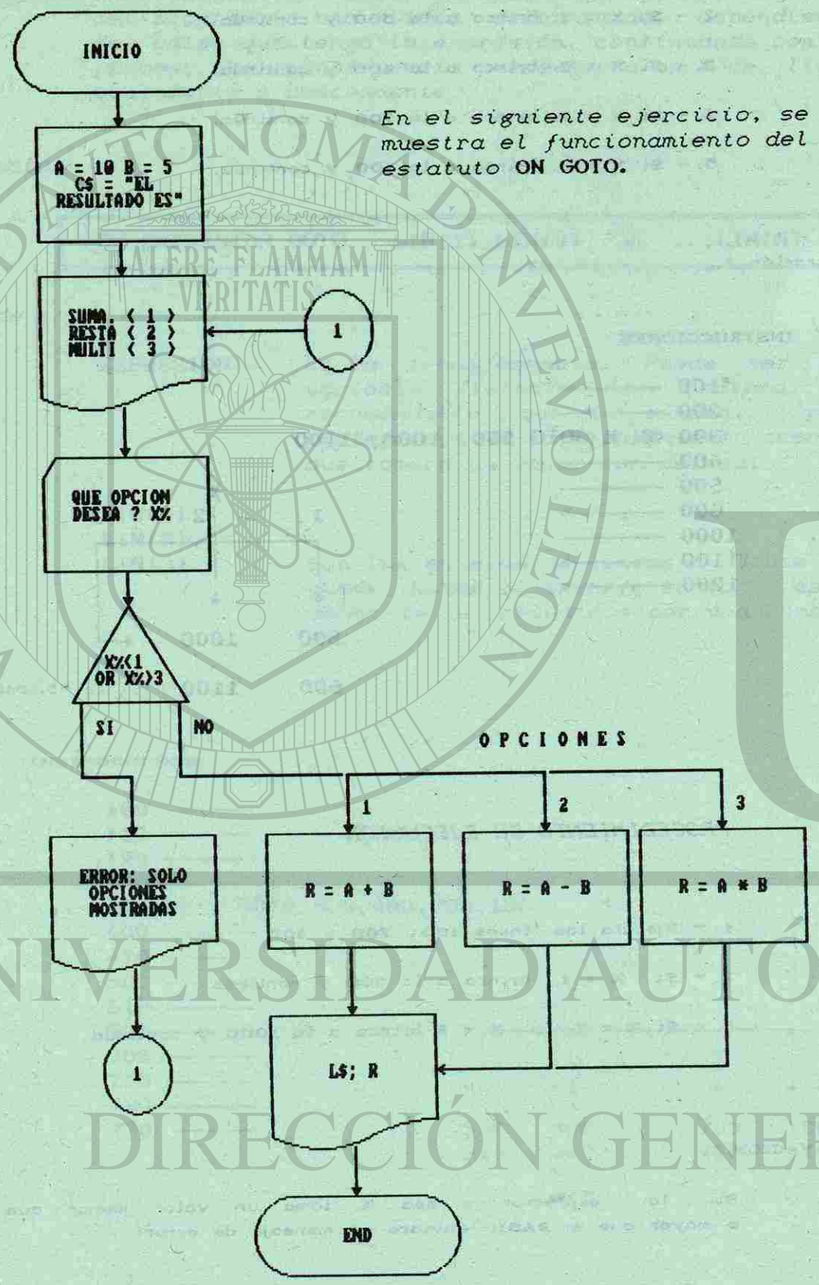
3.- Si X = 2 ó **X = 3** brinca a la **1000** y continúa.

OBSERVACION :

Si la expresión o sea X, toma un valor menor que 1 o mayor que 4, BASIC enviará un mensaje de error.

Ejemplo :

DIAGRAMA DE FLUJO



En el siguiente ejercicio, se muestra el funcionamiento del estatuto **ON GOTO.**

CODIFICACION

```
100  REM PROGRAMADOR : MARIA ELENA
110  LET A = 10
120  LET B = 5
130  LET L$ = "EL RESULTADO ES "
140  PRINT
150  PRINT TAB(26); "PRUEBA DEL ESTATUTO ON GOTO"
160  PRINT TAB(31); "LAS OPCIONES SON :"
170  PRINT \ PRINT
180  PRINT TAB(25); "SUMA DE A Y B ------------ <1>"
190  PRINT TAB(25); "RESTA DE A Y B ----------- <2>"
200  PRINT TAB(25); "MULTIPLICACION DE A Y B -- <3>"
210  PRINT TAB(31); "QUE OPCION DESEA"; \ INPUT X%
220  IF X%<1 OR X%>3 THEN PRINT "ERROR SOLO "+ &
     "OPCIONES MOSTRADAS" \ GOTO 140
230  ON X% GOTO 1000, 2000, 3000
1000 LET R = A + B \ GOTO 4000
2000 LET R = A - B \ GOTO 4000
3000 LET R = A * B
4000 PRINT L$; R
4500 END
```

6.2.2 IF THEN ELSE

Formato :

IF ‹CONDICION›	THEN ESTATUTO(S) THEN # DE LINEA GOTO # LINEA	ELSE ESTATUTO(S) ELSE # DE LINE A

Donde :

‹CONDICION› .- *Es la condición especificada, puede ser verdadera o falsa.*

 # DE LINEA .- *Es el número de línea, donde puede transferir el control del programa.*

ESTATUTO(S) .- *Puede ser uno o varios estatutos de BASIC, exceptuando el MAP, COMMON y el DIM.*

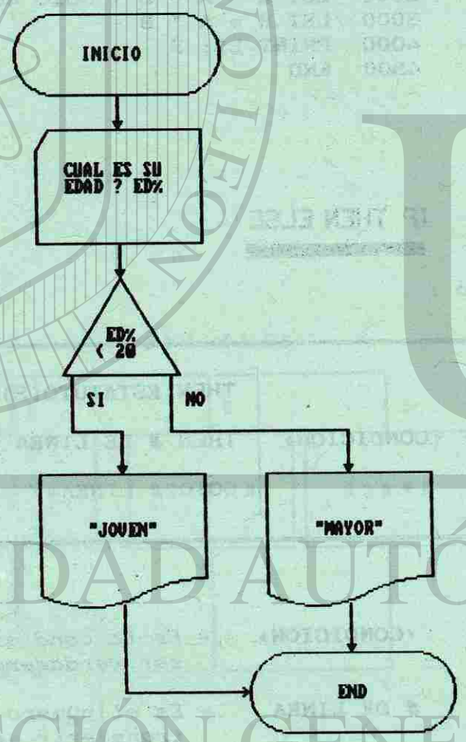
El estatuto **IF THEN ELSE**, *lo utilizamos cuando deseamos realizar operaciones dependiendo de cierta condición.*

Si la condición es verdadera, BASIC ejecuta la cláusula **THEN o GOTO**. *Pero si es falsa, BASIC ejecutaría la cláusula* **ELSE.**

Ejemplo :

Imprimir la palabra JOVEN, si la persona tiene una edad menor a 20 años y la palabra MAYOR, si es igual o mayor a 20 años.

DIAGRAMA DE FLUJO



CODIFICACION

```
100  INPUT "CUAL ES SU EDAD "; ED%
110  IF ED% < 20 THEN 130
120  PRINT "LA PERSONA ES MAYOR" \ GOTO 140
130  PRINT "LA PERSONA ES JOVEN"
140  END

RUN

CUAL ES SU EDAD ? 15
LA PERSONA ES JOVEN
```

PROCEDIMIENTO :

1.- Asignamos a la edad el valor 15

2.- En la línea 110, hace la comparación de 15 < 20, como es menor, brinca a la 130.

3.- Imprime el letrero "JOVEN".

4.- Termina.

Vamos a ejecutar de nuevo el programa :

```
RUN

CUAL ES SU EDAD ? 45
LA PERSONA ES MAYOR
```

PROCEDIMIENTO :

1.- Asignamos el valor 45 a la variable ED%.

2.- En la línea 110 compara si 45 < 20, como no cumple la condición, automáticamente ejecuta la siguiente línea.

3.- En la línea 120, imprime el letrero "mayor".

4.- Termina.

NOTA 1 :

En la línea 110 del programa, la instrucción puede quedar arreglada de esta otra forma :

```
110  IF ED% < 20 THEN 130 ELSE 120
```

Si ejecutamos el programa, obtendríamos los mismos resultados que el anterior.

NOTA 2 :

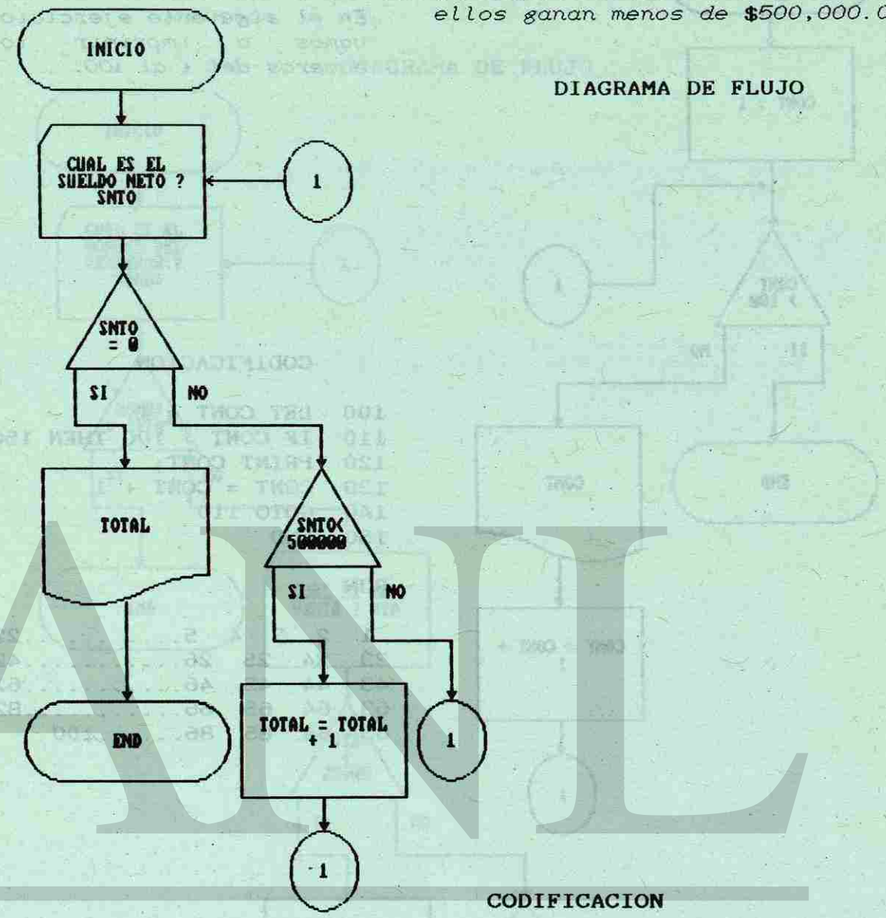
El programa puede trabajar con menos instrucciones, borrando las líneas 120 y 130, y cambiando la línea 110 como sigue :

```
100  INPUT "CUAL ES SU EDAD "; ED%
110  IF ED% < 20 THEN PRINT "LA PERSONA ES JOVEN"
     ELSE PRINT "LA PERSONA ES MAYOR"
140  END

RUN

CUAL ES SU EDAD ? 30
LA PERSONA ES MAYOR
```


Ejemplo :

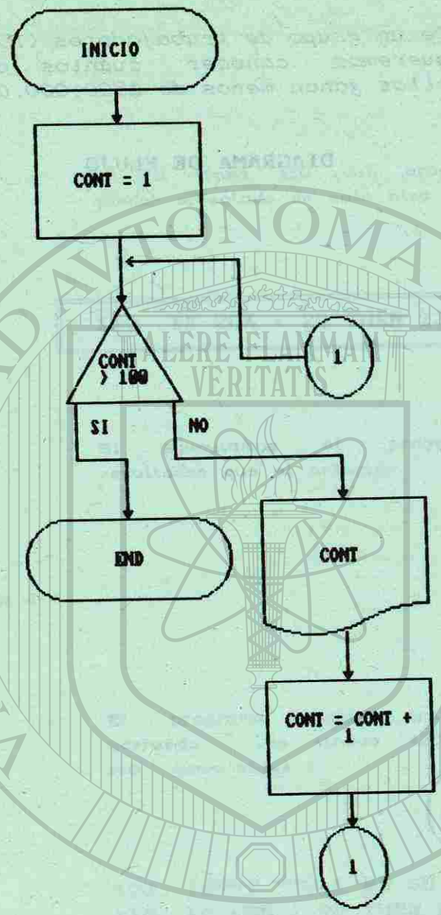
De un grupo de trabajadores (N), queremos conocer cuántos de ellos ganan menos de $500,000.00

DIAGRAMA DE FLUJO



CODIFICACION

```
100  REM PROGRAMADOR : HUGO JUNIOR
110  REM MATRICULA   : 020289
120  REM GRUPO       : 4T
130  REM OBJETIVO    : TOTAL DE TRABAJADORES QUE
135  REM               GANAN MENOS DE $500,000.00
140  REM FECHA       : 02-FEB-89
150  INPUT "CUAL ES SU SUELDO NETO "; SDONTO
160  IF SDONTO = 0 THEN 500
170  IF SDONTO < 500000 THEN TOTAL = TOTAL + 1
180  GOTO 150
500  PRINT "EL TOTAL DE PERSONAS FUERON "; TOTAL
510  END
```



Ejemplo :

En el siguiente ejercicio, vamos a imprimir los números del 1 al 100.

CODIFICACION

```
100  LET CONT = 1
110  IF CONT > 100 THEN 150
120  PRINT CONT;
130  CONT = CONT + 1
140  GOTO 110
150  END

RUN

 1  2  3  4  5.....22
23  24  25  26.....42
43  44  45  46.....62
63  64  65  66.....82
83  84  85  86.....100
```

Ejemplo :

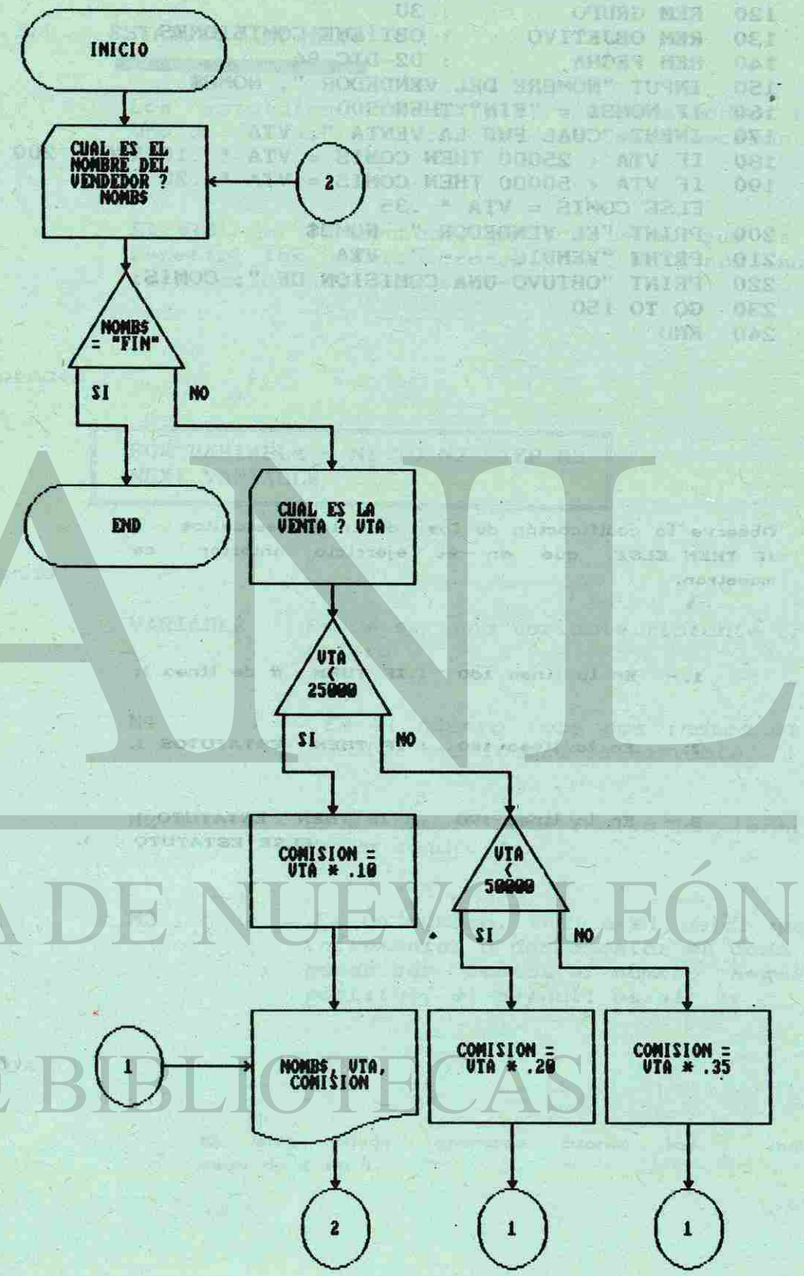
Diseño del diagrama de flujo y codificación del programa, para obtener la comisión de cada vendedor que labora en la compañía FERRETERA MEXICANA S.A.

Los porcentajes de comisión son los siguientes :

Si la venta es menor (<) a $25,000.00, se le dará el 10%, si la venta es igual-mayor (=>) a $ 25,000.00, pero menor (<) a $50,000.00, se le asignará el 20% y por último, si la venta es igual-mayor (=>) a $50,000.00, obtendrá el 35%.

El número de vendedores es "N", ("N" puede tomar cualquier número).
Imprimir : el nombre del vendedor, la venta y su comisión.

DIAGRAMA DE FLUJO



CODIFICACION

```
100  REM PROGRAMADORA : ELIZABETH
110  REM MATRICULA    : 021284
120  REM GRUPO        : 3U
130  REM OBJETIVO     : OBTIENE COMISIONES
140  REM FECHA        : 02-DIC-84
150  INPUT "NOMBRE DEL VENDEDOR "; NOMB$
160  IF NOMB$ = "FIN" THEN 500
170  INPUT "CUAL FUE LA VENTA "; VTA
180  IF VTA < 25000 THEN COMIS = VTA * .10 \GOTO 200
190  IF VTA < 50000 THEN COMIS = VTA * .20
     ELSE COMIS = VTA * .35
200  PRINT "EL VENDEDOR "; NOMB$
210  PRINT "VENDIO ---- "; VTA
220  PRINT "OBTUVO UNA COMISION DE "; COMIS
230  GO TO 150
240  END
```

Observe la codificación de los distintos estatutos **IF THEN ELSE**, que en el ejercicio anterior se muestran.

1.- En la línea 160 (**IF THEN** # de línea).

2.- En la línea 180 (**IF THEN ESTATUTOS**).

3.- En la línea 190 (**IF THEN ESTATUTO**)
(**ELSE ESTATUTO**).

6.3 INTERACCIONES

6.3.1 ESTATUTO FOR-NEXT

Los estatutos **FOR-NEXT**, *sirven para poder repetir una o varias instrucciones las veces necesarias.*

El **FOR**, *es el contador de las veces que se van a repetir las instrucciones y el* **NEXT** *nos indica la terminación de cada ciclo.*

Formato :

```
FOR VARIABLE = N1 TO N2 STEP N3
NEXT VARIABLE
```

Donde :

VARIABLE .- *Puede ser una variable flotante (real) o entera .*

N1 .- *Es el número que nos indica el valor inicial que tomará la variable.*

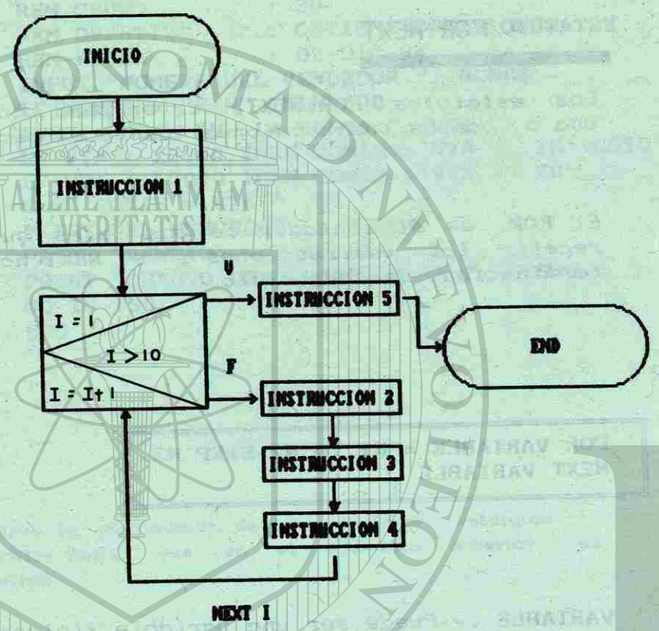
N2 .- *Es el número que indica la terminación de la condición.*

N3 .- *Es un número, indica el valor que va a incrementar o decrementar en cada ciclo, puede ser cualquier número negativo o positivo, el default es el +1*

NOTA :

El step puede ignorarse cuando los incrementos sean de 1 en 1.

Ejemplo ilustrativo :



CODIFICACION

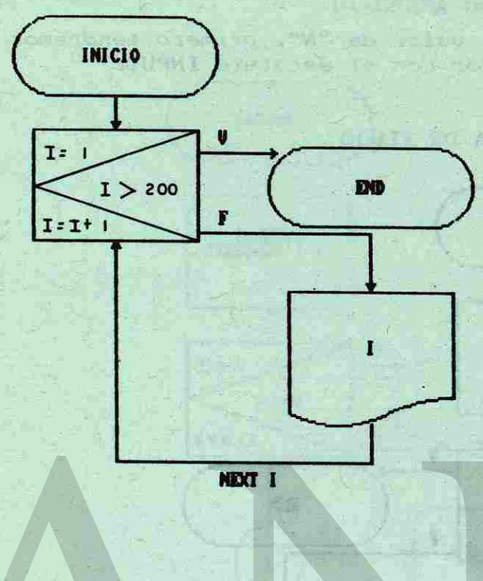
```
100  INSTRUCCION 1
110  FOR I = 1 TO 10 STEP 1
120  INSTRUCCION 2
130  INSTRUCCION 3
140  INSTRUCCION 4
150  NEXT I
160  INSTRUCCION 5
170  END
```

NOTA :

Las instrucciones 2, 3, 4 serán repetidas 10 veces.

Ejemplo :

Aplicando el estatuto **FOR-NEXT,** *imprima los números del 1 al 200*

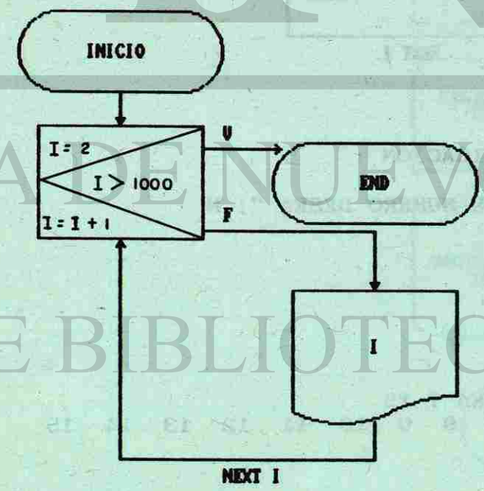


CODIFICACION

```
100  FOR I = 1 TO 200 STEP 1
110  PRINT I;
120  NEXT I
130  END

RUN

 1  2  3  4  5  6  7.....200
```

Ejemplo :

Imprimir los números pares hasta el 1000.



CODIFICACION

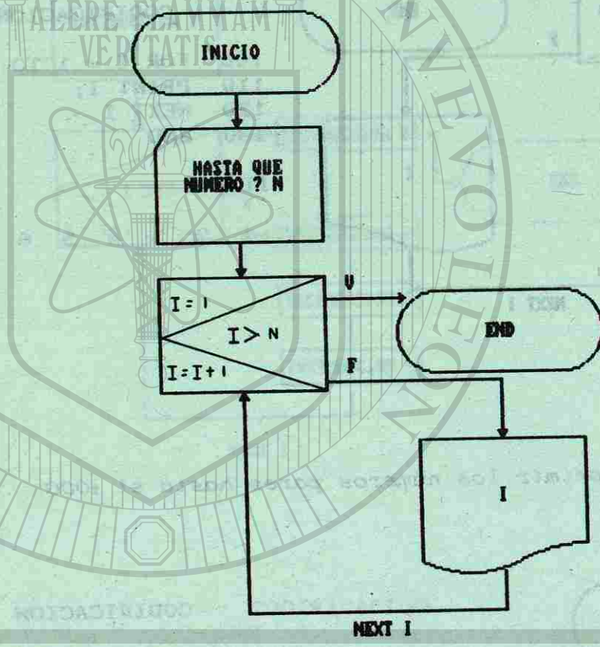
```
100  FOR I = 2 TO 1000 STEP 2
110  PRINT I;
120  NEXT I
130  END

RUN

 2  4  6  8  10  12.....1000
```

Ejemplo :

Imprimir los números del 1 hasta "N". ("N" puede ser cualquier número, eje.- 5, 100, 10, 5000, 500 etc.)

Como desconocemos el valor de "N", primero tendremos que preguntar su valor con el estatuto **INPUT.**

DIAGRAMA DE FLUJO



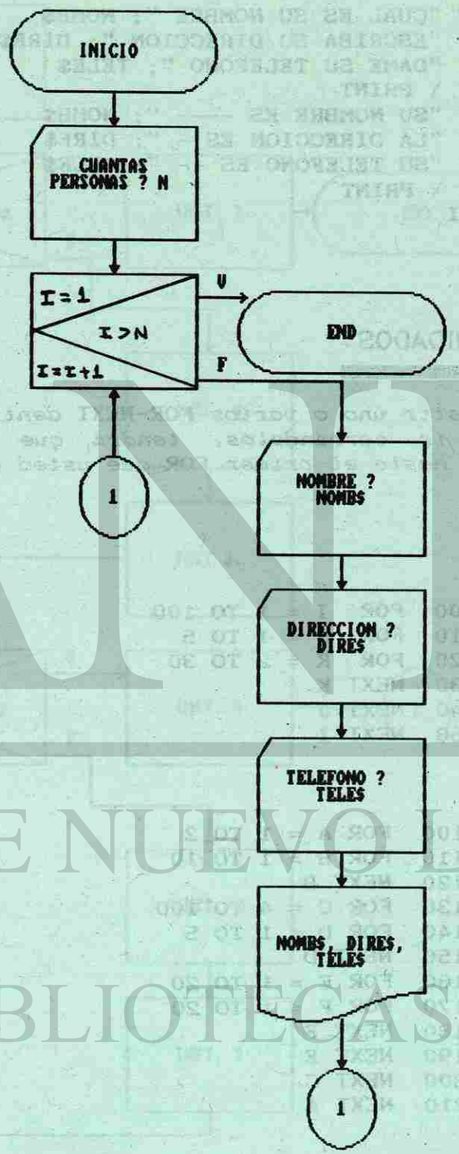
CODIFICACION :

```
100   INPUT "HASTA QUE NUMERO DESEA "; N
110   FOR I = 1 TO N
120   PRINT I;
130   NEXT I
140   END
RUN

HASTA QUE NUMERO DESEA ? 15
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
```


Ejemplo :

Imprimir el nombre, dirección y teléfono para "N" personas :

DIAGRAMA DE FLUJO



CODIFICACION

```
100  REM  PROGRAMADOR   : CLAUDIA
110  REM  MATRICULA     : 020289
120  REM  GRUPO         : 8T
130  REM  OBJETIVO      : IMPRIMIR DATOS PERSONALES
140  REM  FECHA         : 02-FEB-89
150  INPUT "CUANTAS PERSONAS SON "; N
160  FOR I = 1 TO N
170  INPUT "CUAL ES SU NOMBRE "; NOMB$
180  INPUT "ESCRIBA SU DIRECCION "; DIRE$
190  INPUT "DAME SU TELEFONO "; TELE$
200  PRINT \ PRINT
210  PRINT "SU NOMBRE ES ---- "; NOMB$
220  PRINT "LA DIRECCION ES - "; DIRE$
230  PRINT "SU TELEFONO ES -- "; TELE$
240  PRINT \ PRINT
250  NEXT I
260  END
```

6.3.2 FOR-NEXT ANIDADOS

Pueden existir uno o varios **FOR-NEXT** *dentro de otro, para poder ir cerrándolos, tendrá que cerrar el último* **FOR** *hasta el primer* **FOR** *que usted abrió.*

Ejemplo **legal :**

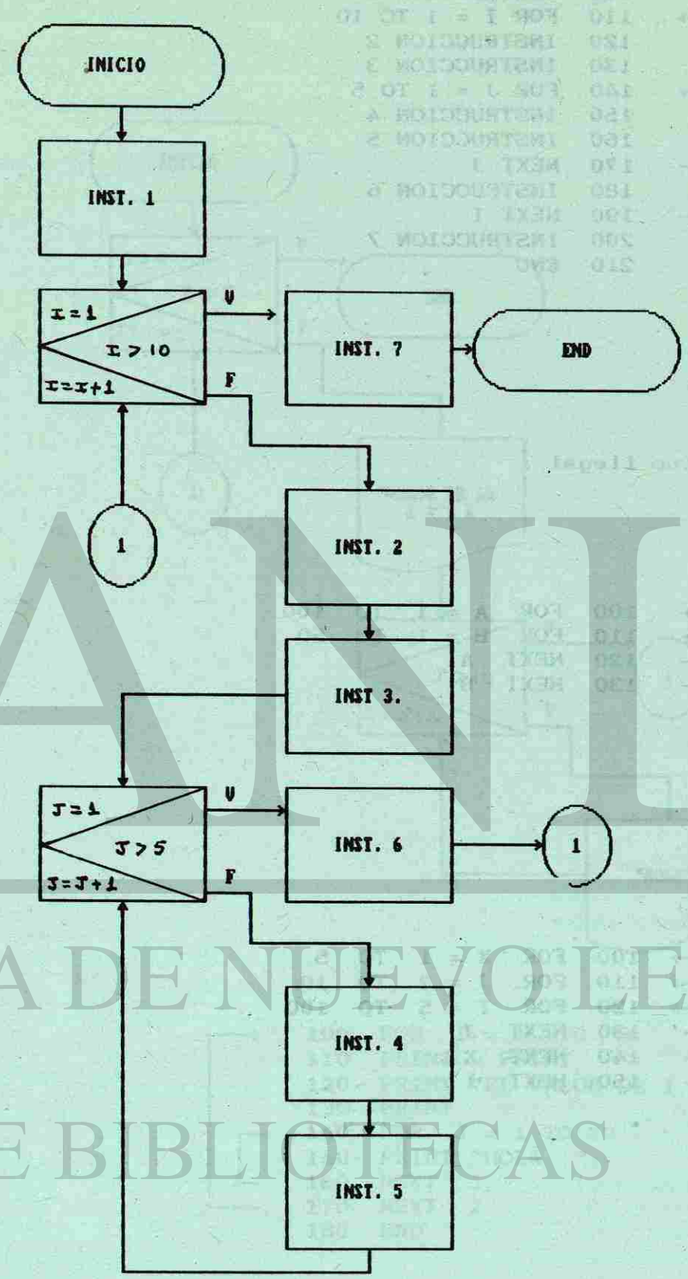
```
100  FOR  I = 1 TO 100
110  FOR  J = 1 TO 5
120  FOR  K = 2 TO 30
130  NEXT K
140  NEXT J
150  NEXT I
```

Ejemplo **legal :**

```
100  FOR A = 1 TO 2
110  FOR B = 1 TO 10
120  NEXT B
130  FOR C = 4 TO 100
140  FOR D = 1 TO 5
150  NEXT D
160  FOR E = 1 TO 20
170  FOR F = 1 TO 20
180  NEXT F
190  NEXT E
200  NEXT C
210  NEXT A
```


Ejemplo ilustrativo **legal.**

DIAGRAMA DE FLUJO



CODIFICACION

```
100  INSTRUCCION 1
110  FOR I = 1 TO 10
120  INSTRUCCION 2
130  INSTRUCCION 3
140  FOR J = 1 TO 5
150  INSTRUCCION 4
160  INSTRUCCION 5
170  NEXT J
180  INSTRUCCION 6
190  NEXT I
200  INSTRUCCION 7
210  END
```

Ejemplo ilustrativo **ilegal** :

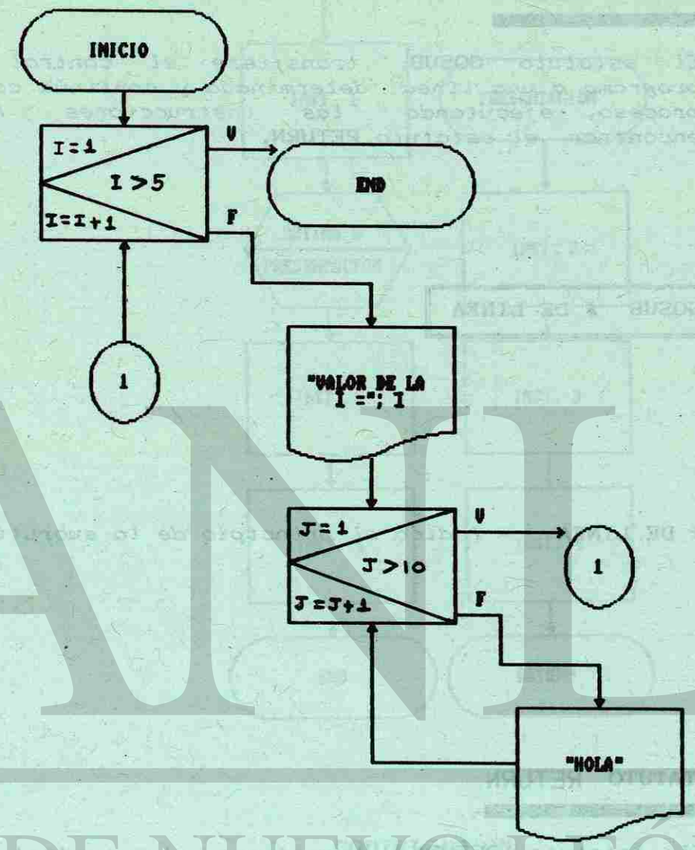
```
100  FOR  A = 1  TO  100
110  FOR  B = 1  TO  50
120  NEXT  A
130  NEXT  B
```

Ejemplo **ilegal** :

```
100  FOR  X = 1  TO  5
110  FOR  J = 2  TO  10
120  FOR  I = 5  TO  100
130  NEXT  J
140  NEXT  X
150  NEXT  I
```


Ejemplo de dos **FOR-NEXT ANIDADOS :**

DIAGRAMA DE FLUJO




```
100  FOR  I = 1  TO  5
110  PRINT \ PRINT
120  PRINT "EL VALOR DE I = "; I
130  PRINT
140  FOR  J = 1 TO 10
150  PRINT "HOLA  ";
160  NEXT  J
170  NEXT  I
180  END
```

6.4 SUBRUTINAS LOCALES

6.4.1 ESTATUTO GOSUB

El estatuto **GOSUB,** *transfiere el control del programa a una línea determinada y continúa con el proceso, ejecutando las instrucciones hasta encontrar el estatuto* **RETURN.**

Formato :

```
GOSUB  # DE LINEA
```

Donde :

 # DE LINEA .- *Indica el principio de la subrutina.*

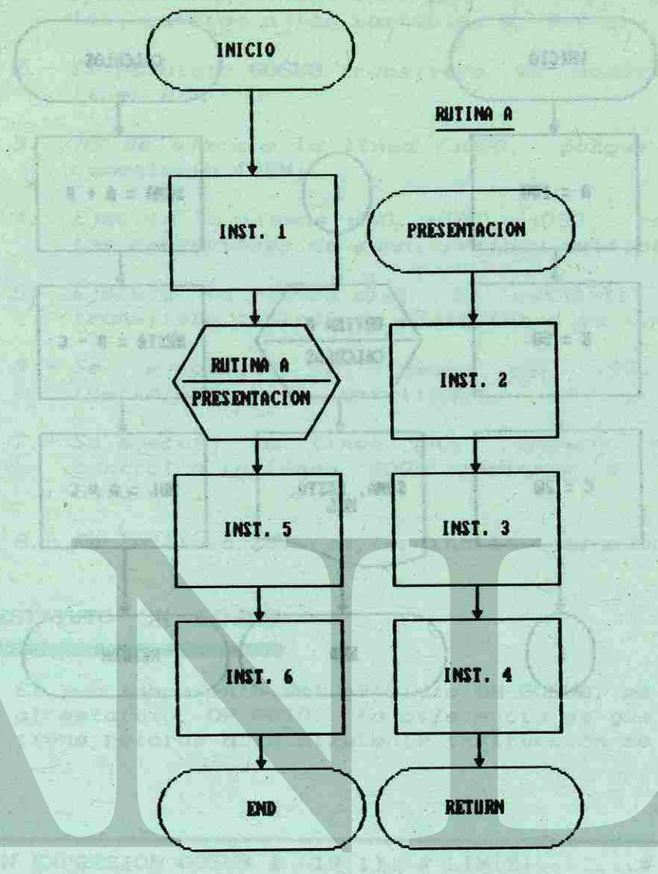
6.4.2 ESTATUTO RETURN

El estatuto **RETURN,** *transfiere el control del programa a la siguiente línea del* **GOSUB** *que fué ejecutado.*

Formato :

```
RETURN
```

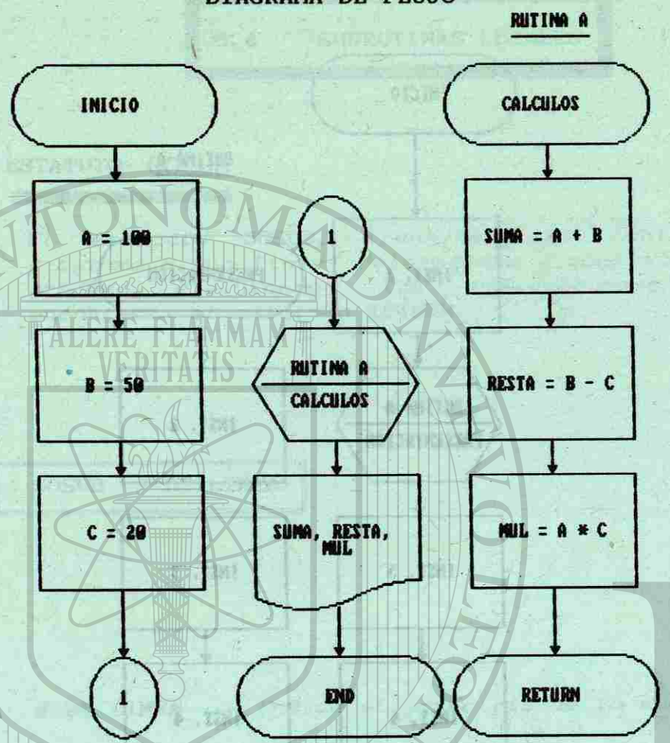

Ejemplo ilustrativo :



CODIFICACION

```
100   INSTRUCCION 1
110   GOSUB 1000
120   INSTRUCCION 5
130   INSTRUCCION 6
140   GOTO 2000
1000  INSTRUCCION 2
1010  INSTRUCCION 3
1020  INSTRUCCION 4
1030  RETURN
2000  END
```

DIAGRAMA DE FLUJO



CODIFICACION

```
100  A = 100
110  B = 50
120  C = 20
130  GOSUB 1000
140  PRINT "LA SUMA DE A + B ES "; SUMA
150  PRINT "LA RESTA DE B - C ES "; RESTA
160  PRINT "LA MULTIPLICACION DE A * C ES ";MUL
170  GOTO 2000
1000  REM RUTINA DE CALCULOS
1010  SUMA = A + B
1020  RESTA = B - C
1030  MUL = A * C
1040  RETURN
1050  END
```


PROCEDIMIENTO DE EJECUCION :

1.- *Ejecuta las líneas 100, 110, y 120, asignando los valores a las variables A, B y C.*

2.- *El estatuto* **GOSUB** *transfiere el control a la línea 1000*

3.- *No se ejecuta la línea 1000, porque es un comentario (***REM***).*

4.- *Ejecuta la lineas 1010, 1020 y 1030, realizando las operaciones de suma, resta y multiplicación.*

5.- *Ejecuta la línea 1040. El estatuto* **RETURN**, *transfiere control de ejecución a la línea 140.*

6.- *Se ejecutan la lineas 140, 150, y 160, imprimiendo los resultados.*

7.- *Se ejecuta la línea 170, transfiriendo el control a la línea 2000 mediante el estatuto* **GOTO**.

8.- *En la línea 2000 se termina la ejecución.*

6.4.3 ESTATUTO ON GOSUB

El funcionamiento del estatuto **ON GOSUB**, *es parecido al estatuto* **ON GOTO**, *la diferencia es que* **ON GOSUB** *tiene retorno a la siguiente instrucción de él.*

Formato :

```
ON EXPRESION GOSUB # LIN(1), # LIN(2).......# LIN(N)
```

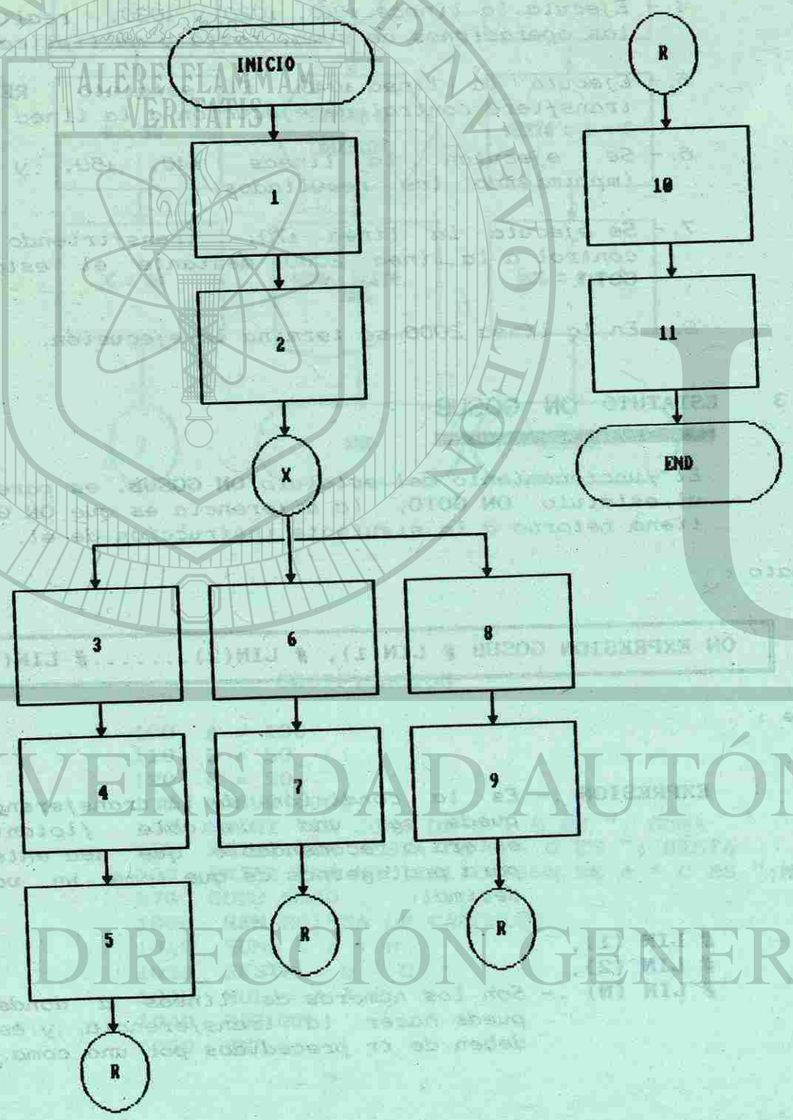
Donde :

EXPRESION .- *Es la condición de la transferencia, puede ser una variable flotante o entera o recomendable que sea entera, para protegernos de que tome un valor decimal.*

 # LIN (1), # LIN (2), # LIN (N) .- *Son los números de líneas a donde se puede hacer la transferencia y éstos deben de ir precedidos por una coma.*

El estatuto **ON GOSUB**, *transfiere el control del programa a una línea determinada, dependiendo del valor que tenga la expresión, continuando con el proceso hasta encontrar el estatuto* **RETURN**, *el cual devuelve el control a la siguiente línea del* **ON GOSUB**.
La transferencia puede ser ascendente o descendente.

DIAGRAMA DE FLUJO



Ejemplo ilustrativo :

```
100  ———————
110  ———————
120  ON X GOSUB 1000,2000,3000
130  ———————
140  ———————
150  GOTO 5000
1000 ———————
1010 ———————
1020 ———————
1030 RETURN
2000 ———————
2010 ———————
2020 RETURN
3000 ———————
3010 ———————
3020 RETURN
5000 END
```

6.5 SUSPENCION DE EJECUCION

Los estatutos **SLEEP** *y* **WAIT**, *sirven para poder suspender por determinado tiempo la ejecución del programa.*

El estatuto **SLEEP** *se utiliza mucho para mensajes y el estatuto* **WAIT** *cuando espera un dato de entrada o sea al utilizar el estatuto* **INPUT**.

6.5.1 ESTATUTO SLEEP

Formato :

```
SLEEP N
```

Donde :

N .- *Es el número entero, que representa el lapso de tiempo en segundos que se vá a suspender la ejecución.*

Ejemplo :

```
100  INPUT "ESTAN CORRECTOS SUS DATOS "; Y$
110  IF Y$ = "SI" THEN PRINT "SI ESTAN CORRECTOS" \&
     GOTO 600
120  IF Y$ = "NO" THEN PRINT "NO ESTAN CORRECTOS" \&
     GOTO 600
130  PRINT "ERROR, TECLEE SOLO  SI  O  NO"
140  SLEEP 3
150  GOTO 100
600  END

RUN
ESTAN CORRECTOS SUS DATOS ? X
ERROR, TECLEE SOLO  SI  O  NO
ESTAN CORRECTOS SUS DATOS ? SI
SI ESTAN CORRECTOS
```

OBSERVACION :

Asignamos el valor de **X** a la variable, los **IFs** de las líneas **110** y **120** no se cumplen, por lo tanto, ejecuta la línea **130** imprimiendo el mensaje de error, después espera **3** segundos y brinca a la línea **100**, volviendo a preguntar sí están correctos.

6.5.2 ESTATUTO WAIT

Formato :

WAIT N

Donde :

N .-*Es un número entero, que representa el lapso de tiempo en segundos que espera el computador para que asignemos el dato de entrada por la terminal.*

Ejemplo :

```
100  WAIT 10
110  INPUT "CUAL ES SU NOMBRE "; NOMB$
120  PRINT "SU NOMBRE ES "; NOMB$
130  END

RUN

CUAL ES SU NOMBRE ?
%BAS-F-KEYWAIEXH,  KEYBOARD WAIT EXHAUSTED

Ready
```

PROCEDIMIENTO DE EJECUCION :

BASIC, espera a que usted le asigne el dato de entrada (nombre) en la terminal, en este caso, dá un tiempo de 10 segundos para recibir el dato.

Si el tiempo de espera termina y no dió el dato, BASIC le manda un mensaje de error (% keyboard wait exausted).

6.6 INTERRUPCION DE EJECUCION

ESTATUTOS STOP Y END

6.6.1 ESTATUTO STOP

El estatuto **STOP**, *detiene la ejecución del programa, nos permite poder examinar en determinados puntos de nuestro programa errores de lógica.*

Formato :

STOP

Al ser ejecutado el estatuto **STOP,** *BASIC imprime lo siguiente :*

```
STOP AT LINE N
#
```

Donde:

N .- *Es el número de línea del estatuto* STOP.

.- *Es el prompt del estatuto* STOP *es la indicación de que la ejecución fué detenida.*

En respuesta al prompt (#), puedes contestar con las cláusulas siguientes :

CONT .- *Le indicas al computador que continúe con la ejecución del programa.*

EXIT .- *Le indicas que se salga del* STOP, *y lo mande al comando de BASIC o sea READY.*

Ejemplo :

Ejercicio para probar las cláusulas **CONT** *y* **EXIT** *al utilizar el estatuto* STOP.

```
100  A = 100
110  B = 50
120  C = A + B
130  STOP
140  PRINT "EL RESULTADO ES "; C
150  END
RUN

%BAS-I-STO, Stop
-BAS-I-FROLINMOD,  from line 130 in module JOH23
Ready

CONT    <CR>
EL RESULTADO ES  150
```


Ejecutaremos de nuevo para probar la cláusula **EXIT.**

```
RUN

%BAS-I-STO, Stop
%-BAS-I-FROLINMOD,  from line 130 in module JOH23
Ready

EXIT    <CR>
$
```

NOTA :

Al utilizar la cláusula EXIT se saldrá del INTERPRETADOR BASIC.

OBSERVACION:

El estatuto STOP , no cierra los archivos que se estén utilizando en el programa.

6.6.2 ESTATUTO END

El estatuto **END,** *nos indica la terminación de la ejecución del programa.*

Formato :

```
END
```

OBSERVACION :

El estatuto END, sí cierra todos los archivos que se estén utilizando en el programa.

6.7 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Cuál es la función de la transferencia de control incondicional* **GOTO.**

2.- *Mencione los dos estatutos de transferencia condicional.*

1. _____ 2. _____

3.- *Cuál es la función del estatuto* **ON GOTO.**

4.- *Escriba el formato del estatuto* **ON GOTO.**

5.- *Describa la transferencia de control condicional.*

6.- *Escriba el formato completo para utilizar el estatuto* **IF THEN ELSE.**

7.- *Mencione la función de los estatutos* **FOR-NEXT.**

8.- *Escriba el formato que utiliza el* **FOR-NEXT.**

9.- *Si usted abriera 4 estatutos* **FOR**, *como se muestra a continuación, cómo los cerraría ?.*

1.- FOR X		1.- NEXT
2.- FOR J	——— →	2.- NEXT
3.- FOR A	——— →	3.- NEXT
4.- FOR T		4.- NEXT

10.- *Cuál es la función del* **GOSUB ?.**

11.- *Cuáles son los estatutos de* **SUSPENCION DE EJECUCION** *y explíquelos.*

12.- *Para qué sirve el estatuto* **END** *?.*

CAPITULO 7

FUNCIONES

OBJETIVO GENERAL

Estudiar las diferentes funciones que se pueden utilizar en el lenguaje de programación **BASIC.**

OBJETIVO ESPECIFICO

Introduce al alumno en el manejo de las funciones más importantes de **BASIC,**

Por ejemplo :

Funciones Algebraicas **INT, FIX, ABS,** *etc.*

Función Exponencial **EXP**

Funciones Logarítmicas **LOG, LOG10**

Funciones String **LEN, LEFT, RIGHT, MID,** *etc.*

Que le podrán ayudar a obtener muchas facilidades en la búsqueda de problemas especiales.

11.- *Cuáles son los estatutos de* **SUSPENCION DE EJECUCION** *y explíquelos.*

12.- *Para qué sirve el estatuto* **END** *?.*

CAPITULO 7

FUNCIONES

OBJETIVO GENERAL

Estudiar las diferentes funciones que se pueden utilizar en el lenguaje de programación **BASIC.**

OBJETIVO ESPECIFICO

Introduce al alumno en el manejo de las funciones más importantes de **BASIC,**

Por ejemplo :

Funciones Algebraicas **INT, FIX, ABS,** *etc.*

Función Exponencial **EXP**

Funciones Logarítmicas **LOG, LOG10**

Funciones String **LEN, LEFT, RIGHT, MID,** *etc.*

Que le podrán ayudar a obtener muchas facilidades en la búsqueda de problemas especiales.

7.1 FUNCIONES ALGEBRAICAS

7.1.1 INT

La función **INT**, *nos regresa un número de valor real a un número de valor entero.*

Formato :

```
INT ( Expresión )
```

Ejemplo :

```
100  PRINT INT (34.59)
110  PRINT INT (1000.85)
120  PRINT INT (26.99)
130  PRINT INT (-259.89)
140  PRINT INT (-8.2)
150  PRINT INT (-150)
160  END

RUN

 34
 1000
 26
-260
-9
-150
```

7.1.2 FIX

La función **FIX**, *regresa un valor real a un valor entero.*

Formato :

```
FIX ( Expresión )
```


Ejemplo :

```
100  PRINT FIX (220.0)
110  PRINT FIX (10)
120  PRINT FIX (125.92)
130  PRINT FIX (.8)
140  PRINT FIX (-200)
150  PRINT FIX (-4.78)
160  END

RUN

 220
 10
 125
 0
-200
-4
```

7.1.3 ABS

La función **ABS**, *nos dá el valor absoluto de un número real.*

Formato :

```
ABS ( EXPRESION )
```

Ejemplo :

```
100  PRINT ABS (-150)
110  PRINT ABS (-10.9)
120  PRINT ABS (-42.25)
130  PRINT ABS (-2.445566)
140  PRINT ABS (-1)
150  PRINT ABS (100)
160  PRINT ABS (75.4)
170  END

RUN

 150
 10.9
 42.25
 2.44557
 1
 100
 75.4
```

7.1.4 SQR

La función **SQR**, *sirve para obtener la raíz cuadrada de un número*

Formato :

```
SQR ( EXPRESION )
```

Ejemplo :

```
100  PRINT SQR (25)
110  PRINT SQR (128)
120  PRINT SQR (48.25)
130  PRINT SQR (4.5)

RUN

 5
 11.3137
 6.94622
 2.12132
```

OBSERVACION :

No se puede obtener la raíz cuadrada de un número negativo.

7.1.5 SGN

La función **SGN**, *nos determina si la expresión es positiva, negativa o cero.*

Si la expresión es positiva, imprime un **+ 1**.

Si la expresión es negativa, imprime un **- 1**.

Y si la expresión es cero, imprimirá un **0** *(cero)*

Formato :

```
SGN ( EXPRESION )
```

Ejemplo :

```
100  PRINT "1000 = "; SGN (1000)
110  PRINT "-8.49 = "; SGN (-8.49)
120  PRINT "CERO 0 = "; SGN (0)
130  END

RUN

1000 =  1
-8.49 = -1
CERO 0 =  0
```

7.2 FUNCION EXPONENCIAL

7.2.1 EXP

La función **EXP**, *sirve para obtener el valor exponencial de un número.*

Formato :

```
EXP  ( EXPRESION )
```

Ejemplo :

```
100  PRINT EXP (1)
110  PRINT EXP (45)
120  PRINT EXP (2.5)
130  PRINT EXP (-5)
140  END

RUN

 2.71828
 .349343E+20
 12.1825
 .673795E-02
```

7.3.1 FUNCIONES LOGARITMICAS

7.3.1 LOG

La función **LOG**, *sirve para obtener el valor del logaritmo natural de un número.*

Formato :

```
LOG ( EXPRESION )
```

Ejemplo :

```
100  PRINT LOG (4)
110  PRINT LOG (1)
120  PRINT LOG (52)
130  END

RUN

 1.38629
 0
 3.95124
```

7.3.2 LOG10

La función **LOG10**, *obtiene el valor del logaritmo base 10 de un número.*

Formato :

```
LOG10  ( EXPRESION )
```


Ejemplo :

```
100  PRINT LOG10 (25)
110  PRINT LOG10 (1)
120  PRINT LOG10 (4)
130  END

RUN

 1.39794
 0
 .60206
```

7.4 FUNCIONES TRIGONOMETRICAS

La función **PI**, *es el valor de la veces que cabe un radio en una circunferencia.*

Formato :

7.4.1 PI

Ejemplo :

```
100  PRINT PI
110  PRINT "EL VALOR DE PI = "; PI
120  END

RUN

 3.14159
EL VALOR DE PI =  3.14159
```

7.4.1 SIN, COS, TAN y ATN

Las funciones **SIN, COS, TAN,** *nos dá el valor del seno, coseno y tangente de un ángulo expresado en radianes.*

La función del **ATN,** *calcula el arco del valor de la tangente*

Formato :

```
SIN ( EXPRESION )
COS ( EXPRESION )
TAN ( EXPRESION )
ATN ( EXPRESION )
```

Donde :

EXPRESION .- *Es un valor en radianes.*

Ejemplo :

```
100 PRINT SIN(0), SIN(45), SIN(90), SIN(180), SIN(360)
105 PRINT
110 PRINT COS(0), COS(45), COS(90), COS(180), COS(360)
115 PRINT
120 PRINT TAN(0), TAN(45), TAN(90), TAN(180), TAN(360)
125 PRINT
130 PRINT ATN(0), ATN(45), ATN(90), ATN(180), ATN(360)
140 END

RUN

 0      .850904      .893997      -.801153      .958916

 1      .525322      -.448 74     -.59846       -.283691

 0      1.61978      -1.9952      1.33869       -3.38014

 0      1.54858      1.55969      1.56524       1.56802
```


7.5 FUNCIONES STRING (CADENA DE CARACTERES)

La programación BASIC, maneja muchas funciones para manipular cadenas de caracteres.

Anunciamos unas de las más importantes con sus respectivos formatos :

7.5.1 LEN (STRING) .- Nos dá el número de caracteres del contenido del string.

7.5.2 INSTR(N,STRING1,STRING2).- Busca desde "N", la posición donde se encuentra el contenido del string2, dentro del string1 "N" es un número.

7.5.3 LEFT$(STRING,N) .- Obtiene la izquierda del contenido del string, las posiciones "N" inclusive, "N" es un número.

7.5.4 RIGHT$(STRING,N) .- Obtiene la derecha del contenido del string, las posiciones "N" inclusive, "N" es un número.

7.5.5 MID$(STRING,N,N1) .- Obtiene un rango de caracteres del contenido del string, desde la posición "N" inclusive, las posiciones "N1".

7.5.6 STRING$(N,N1) .- Repite el número de veces el caracter especificado por "N1". "N" nos indca el número de veces a repetir.

7.5.7 SPACE$(N) .- Sirve para intercalar espacios en blanco en una línea de impresión. "N" indica el número de espacios requeridos.

OBSERVACIONES :

1.- Al utilizar las funciones **LEN** y **INSTR**, el resultado debe de almacenarse en una variable entera.

2.- Al utilizar las funciones **LEFT$**, **RIGHT$**, **MID$**, **STRING$** y **SPACE$**, el resultado debe almacenarse en una variable alfanumérica.

Ejemplo :

```
100 A$ = "LICENCIADO EN INFORMATICA ADMINISTRATIVA"
110 B$ = "FORMATICA"
120 TAMAÑO% = LEN(A$)
130 POSIOCION% = INSTR(1,A$,B$)
140 IZQUIERDA$ = LEFT$(A$,10)
150 DERECHA$ = RIGHT$(A$,27)
160 RANGO$ = MID$(A$,15,11)
170 ASTERISCO$ = STRING$(15,42)
180 ESPACIOS$ = SPACE$(20)
190 PRINT "EL TAMAÑO DE A$ ES -------- "; TAMAÑO%
200 PRINT "POSICION DE FORMATICA ----- "; POSICION%
210 PRINT "LA IZQUIERDA DE A$ ES ----- "; IZQUIERDA$
220 PRINT "LA DERECHA DE A$ ES ------- "; DERECHA$
230 PRINT "EL RANGO DE A$ ES --------- "; RANGO$
240 PRINT "REPETICION DEL ASTERISCO -- "; ASTERISCO$
250 PRINT "ESPACIOS EN ------ "; ESPACIOS$; "BLANCO"
260 END
```

RUN

```
EL TAMAÑO DE A$ ES --------  40
POSICION DE FORMATICA -----  17
LA IZQUIEDA DE A$ ES ------ LICENCIADO
LA DERECHA DE A$ ---------- ADMINISTRATIVA
EL RANGO DE A$ ES --------- INFORMATICA
REPETICION DEL ASTERISCO -- **************
ESPACIOS EN ------                    BLANCO
```


7.6 FUNCIONES STRING NUMERICAS

Las funciones **STRING NUMERICAS**, *sirven para poder realizar operaciones aritméticas con variables alfanuméricas.*

7.6.1 SUM$

La función **SUM$**, *sirve para realizar sumas algebraicas.*

Formato :

SUM$ (STRING1, STRING2)

Descripción :

El valor del **STRING2** *es sumado al valor del* **STRING1.**

Ejemplo :

```
100  A$ = "500"
110  B$ = "100"
120  C$ = SUM$(A$,B$)
130  PRINT "EL RESULTADO DE LA SUMA ES "; C$
140  END
```

RUN

```
EL RESULTADO DE LA SUMA ES 600
```

Ejemplo :

```
100  A$ = "10.98"
110  B$ = "2.42"
120  C$ = SUM$(A$,B$)
130  PRINT "LA SUMA ES "; C$
140  END
```

RUN

```
LA SUMA ES 13.4
```

Ejemplo :

```
100  A$ = ".5432"
110  B$ = "6.389"
120  C$ = SUM$(A$,B$)
130  PRINT "LA SUMA ES "; C$
140  END

RUN

LA SUMA ES 6.9322
```

7.6.2 DIF$

La función **DIF$**, *nos ayuda a realizar diferencias.*

Formato :

DIF$ (STRING1, STRING2)

Descripción :

El valor del **STRING2** *es restado al valor del* **STRING1.**

Ejemplo :

```
100  X$ = "258"
110  Y$ = "134"
120  Z$ = DIF$(X$,Y$)
130  PRINT "LA DIFERENCIA DE X$ A Y$ ES "; Z$
140  END

RUN

LA DIFERENCIA DE X$ A Y$ ES 124
```

Ejemplo :

```
100  H$ = "469.78"
110  I$ = "214.25"
120  J$ = DIF$(H$,I$)
130  PRINT "LA DIFERENCIA ES "; J$
140  END

RUN

LA DIFERENCIA ES 255.53
```


Ejemplo :

```
100  A$ = "1.245"
110  B$ = ".98"
120  C$ = DIF$(A$,B$)
130  PRINT "LA DIFERENCIA ES "; C$
140  END

RUN

LA DIFERENCIA ES .265
```

7.6.3 PROD$

La función **PROD$**, *es utilizado para realizar productos.*

Formato :

PROD$ (STRING1, STRING2, X%)

Descripción :

El valor del **STRING1** *es multiplicado por el valor del* **STRING2. X%** *nos indica el número de decimales que deseamos utilizar.*

Ejemplo :

```
100  A$ = "50"
110  B$ = "20"
120  C$ = PROD$(A$,B$,2)
130  PRINT "EL RESULTADO DE LA MULTIPLICACION ES "+&
     ; C$
140  END

RUN

EL RESULTADO DE LA MULTIPLICACION ES 1000
```

Ejemplo :

```
100  A$ = "22.75"
110  B$ = "4.63"
120  C$ = PROD$(A$,B$,2)
130  D$ = PROD$(A$,B$,4)
140  PRINT "RESULTADO CON 2 DECIMALES ES "; C$
150  PRINT "RESULTADO CON 4 DECIMALES ES "; D$
160  END

RUN

RESULTADO CON 2 DECIMALES ES 105.33
RESULTADO CON 4 DECIMALES ES 105.3325
```

7.6.4 QUO$

La función **QUO$**, *nos ayuda a obtener divisiones.*

Formato :

```
QUO$ ( STRING1, STRING2, X% )
```

Descripción :

El valor de **STRING1** *es dividido por el valor del* **STRING2, X%** *nos indica el número de decimales que deseamos utilizar.*

Ejemplo :

```
100  A$ = "220"
110  B$ = "4"
120  C$ = QUO$(A$,B$,2)
130  PRINT "EL RESULTADO DE LA DIVISION ES "; C$
140  END

RUN

EL RESULTADO DE LA DIVISION ES 55
```


Ejemplo :

```
100  A$ = "176.92"
110  B$ = "12 "
120  C$ = QUO$(A$,B$,2)
130  D$ = QUO$(A$,B$,6)
140  PRINT "EL RESULTADO CON 2 DECIMALES ES "; C$
150  PRINT "EL RESULTADO CON 6 DECIMALES ES "; D$
160  END

RUN

RESULTADO CON 2 DECIMALES ES 14.74
RESULTADO CON 6 DECIMALES ES 14.743333
```

7.7 FUNCIONES DE FECHA Y TIEMPO

7.7.1 DATE$(0)

La función **DATE$(0)**, *sirve para poder desplegar la fecha actual, en la forma DD-MMM-AA.*

Formato :

```
DATE$(0)
```

Donde :

0 .- *Es la representación de la fecha actual*

Ejemplo :

```
100  PRINT "REPORTE DE LA NOMINA AL DIA "; DATE$(0)
110  PRINT TAB(26); "LISTADO GRAL. AL DIA ";DATE$(0)
120  PRINT TAB(15); DATE$(0)
130  PRINT DATE$(0)
140  END

RUN

REPORTE DE LA NOMINA AL DIA 02-FEB-89
                        LISTADO GRAL. AL DIA 02-FEB-89
                  02-FEB-89
02-FEB-89
```

NOTA :

Existen otras formas de poder imprimir fechas, las cuales pueden investigarlas en el lenguaje referece manual primera parte (cap. 5 pag. 30).

7.7.2 TIME$(O)

La función **TIME$(O)**, *sirve para poder desplegar la hora actual, en la forma HH:MM A.M ó HH:MM P.M.*

Formato :

```
TIME$(O)
```

Donde :

O .- *Es la representación de la hora actual.*

Ejemplo :

```
100  PRINT TIME$(O)
110  PRINT TAB(15); TIME$(O)
120  PRINT "EL REPORTE FUE IMPRESO A LAS "; TIME$(O)
130  PRINT TAB(25);"EL REPORTE FUE IMPRESO A LAS ";&
     TIME$(O)
140  END

RUN

13:50 PM
                13:50 PM
EL REPORTE FUE IMPRESO A LAS 13:50 PM
                        EL REPORTE FUE IMPRESO A LAS 13:50 PM
```

NOTA :

Existen otras formas de poder imprimir los tiempos, las cuales pueden estudiarlas en el libro lenguaje reference manual primera parte (cap 5 pag 30).

7.8 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Escriba los tipos de funciones que puede trabajar* **BASIC.**

2.- *Mencione la función algebraica* **INT.**

3.- *Para qué sirve la función* **ABS.**

4.- *Explique la función* **SGN.**

5.- *Escriba las funciones* **TRIGONOMETRICAS.**

6.- *Explique las siguientes funciones :*

LEN (STRING) _____

_____

LEFT$ (STRING, N) _____

_____

RIGHT$ (STRING,N,) _____

_____

MID$ (STRING,N,N1) _____

_____

SPACE$ (N) _____

_____

7.- *Explique las siguientes funciones* **STRING NUMERICAS.**

SUM$ _____

DIF$ _____

PROD$ _____

QUO$ _____

8.- *Explique la funciones* **DATE$(0)** *y* **TIME$(0).**

_____

_____

_____

_____

CAPITULO 8

ARREGLOS

OBJETIVO GENERAL

El estudiante conceptualizará, comprenderá y utilizará correctamente la definición y uso de arreglos o tablas de una y dos dimensiones en un programa **BASIC.**

OBJETIVO ESPECIFICO

El estudiante se iniciará en la utilización, manejo y codificación de programas con arreglos para realizar una estructuración más eficiente en el ordenamiento de la información (DATOS).

6.- *Explique las siguientes funciones :*

LEN (STRING) _____

_____

LEFT$ (STRING, N) _____

_____

RIGHT$ (STRING,N,) _____

_____

MID$ (STRING,N,N1) _____

_____

SPACE$ (N) _____

_____

7.- *Explique las siguientes funciones* **STRING NUMERICAS.**

SUM$ _____

DIF$ _____

PROD$ _____

QUO$ _____

8.- *Explique la funciones* **DATE$(0)** *y* **TIME$(0).**

_____

_____

_____

_____

CAPITULO 8

ARREGLOS

OBJETIVO GENERAL

El estudiante conceptualizará, comprenderá y utilizará correctamente la definición y uso de arreglos o tablas de una y dos dimensiones en un programa **BASIC.**

OBJETIVO ESPECIFICO

El estudiante se iniciará en la utilización, manejo y codificación de programas con arreglos para realizar una estructuración más eficiente en el ordenamiento de la información (DATOS).

8.1 ESTATUTO DIMENSION (DIM)

Al utilizar el estatuto **DIM [ENSION]**, *nos permite generar tablas o arreglos, para poder separar memoria, en la cual vamos almacenar información.*

Formato :

```
DIM  NOMB(TAMAÑO), NOMB(TAMAÑO),.......,NOMB(TAMAÑO)
```

Donde :

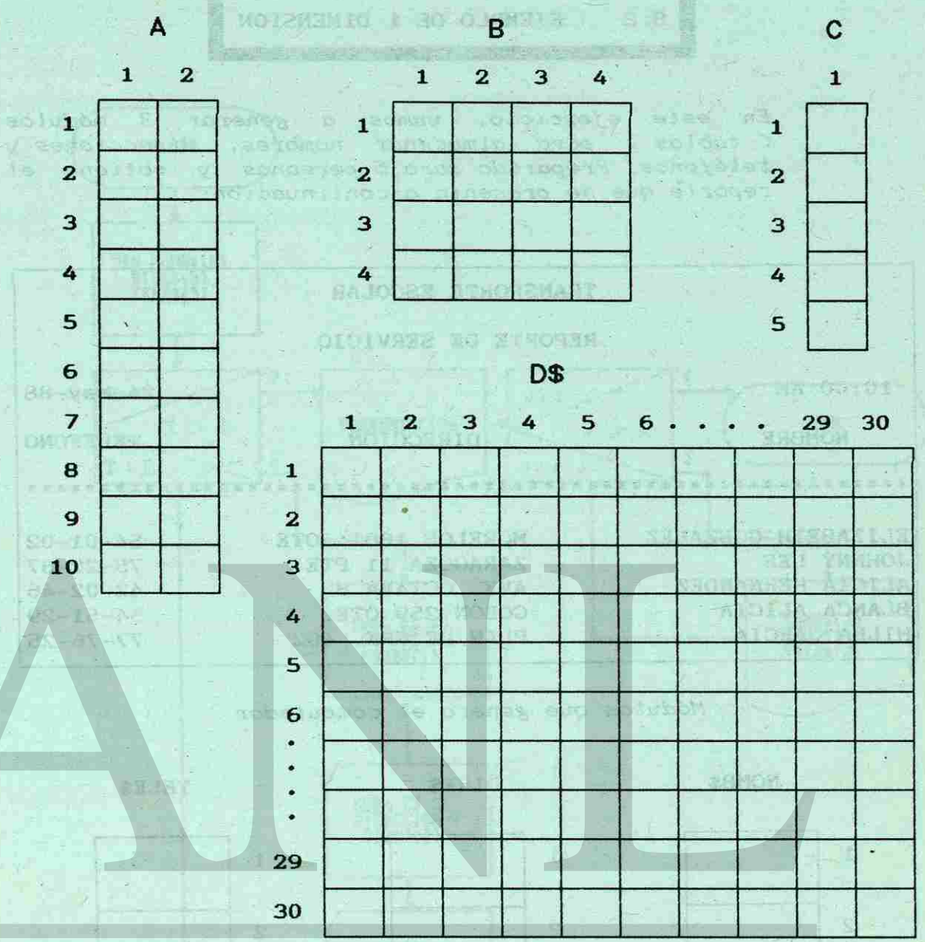
NOMB .- *Es el nombre del arreglo, puede ser cualquier tipo de variable.*

TAMAÑO .- *Puede ser uno o dos números enteros, nos indica el tamaño del arreglo. El máximo número puede ser 32767.*

Ejemplo :

```
DIM  A(10,2), B(4,4)

DIM  C2(5), D$(30,30)

DIMENSION  X$(5), Z(2,2)
```


Ilustración de lo anterior :

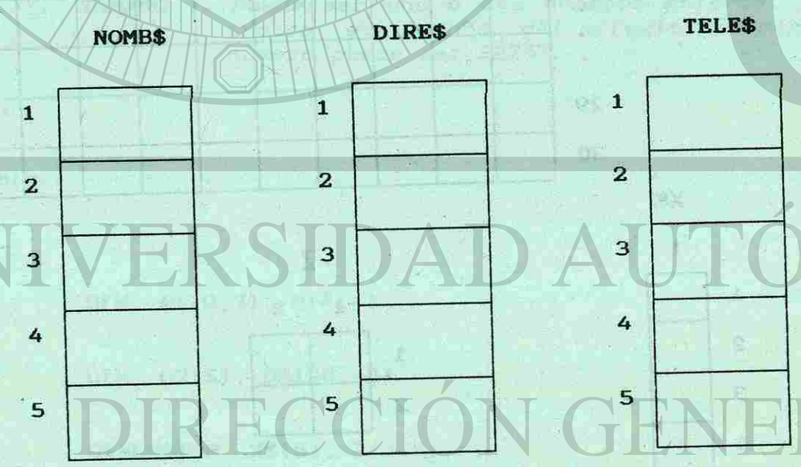


8.2 EJEMPLO DE 1 DIMENSION

En este ejercicio, vamos a generar 3 módulos (tablas) para almacenar nombres, direcciones y teléfonos. Preparado para 5 personas y obtiene el reporte que se presenta a continuación:

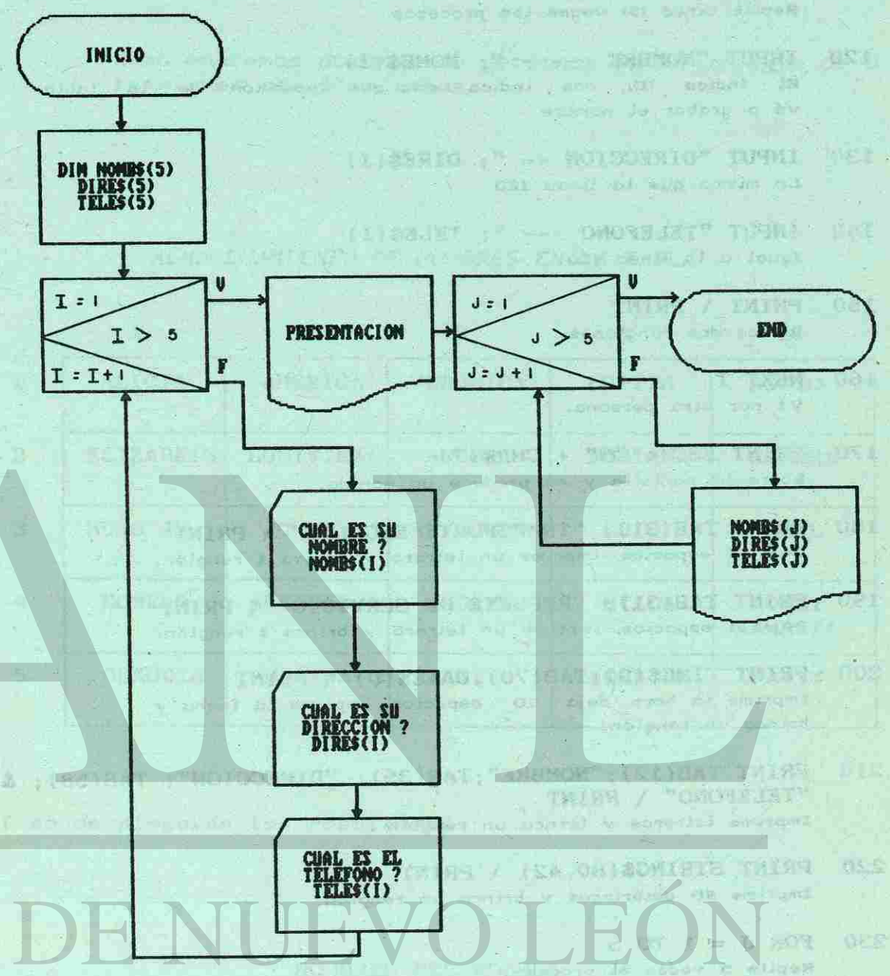
```
                    TRANSPORTE ESCOLAR

                    REPORTE DE SERVICIO

10:00 AM                                              24-May-88

     NOMBRE                    DIRECCION                TELEFONO

**********************************************************************

ELIZABETH GONZALEZ        MORELOS 1081  OTE.          54-01-02
JOHNNY LEE                ZARAGOZA 11 PTE.            75-22-87
ALICIA HERNANDEZ          AVE. OCTAVA 85              42-02-46
BLANCA ALICIA             COLON 259 OTE.              54-51-29
HILDA GARCIA              PLAN DE GPE. 402            77-76-25
```

Módulos que genera el computador



A continuación se presenta el diagrama de flujo del programa.

DIAGRAMA DE FLUJO



La codificación del programa se presenta enseguida.

```
100  DIM NOMB$(5), DIRE$(5), TELE$(5)
     Genera las tres (3) tablas.

110  FOR  I = 1  TO  5
     Repite cinco (5) veces los procesos

120  INPUT "NOMBRE ----- "; NOMB$(I)
     El índice (I), nos indica en que posición de la tabla se
     vá a grabar el nombre.

130  INPUT "DIRECCION -- "; DIRE$(I)
     Lo mismo que la línea 120

140  INPUT "TELEFONO --- "; TELE$(I)
     Igual a la línea 120.

150  PRINT \ PRINT
     Brinca dos renglones.

160  NEXT I
     Vá por otra persona.

170  PRINT ESC + "M" + CHR$(7)
     Borra la pantalla y se produce un sonido.

180  PRINT TAB(31); "TRANSPORTE ESCOLAR" \ PRINT
     Deja 31 espacios, imprime un letrero y brinca 1 renglón.

190  PRINT TAB(31); "REPORTE DE SERVICIO" \ PRINT
     Deja 31 espacios, imprime un letrero y brinca 1 renglón.

200  PRINT TIME$(0);TAB(70);DATE$(0) \ PRINT
     Imprime la hora, deja  70  espacios, imprime la fecha y
     brinca un renglón.

210  PRINT TAB(12);"NOMBRE";TAB(35); "DIRECCION"; TAB(58); &
     "TELEFONO" \ PRINT
     Imprime letreros y brinca un renglón

220  PRINT STRING$(80,42) \ PRINT
     Imprime 80 asteriscos y brinca un renglón.

230  FOR J = 1 TO 5
     Repite 5 veces el proceso.

240  PRINT TAB(10);NOMB$(J);TAB(31);DIRE$(J);TAB(58);TELE$(J)
     Imprime los valores de la tabla, mediante el índice (J).

250  NEXT J
     Vá por otro valor de la tabla.

260  END
     Termina el programa.
```


8.3 EJEMPLO DE 2 DIMENSIONES

Almacenaremos nombres de personas en un arreglo de 5 renglones por 5 columnas.

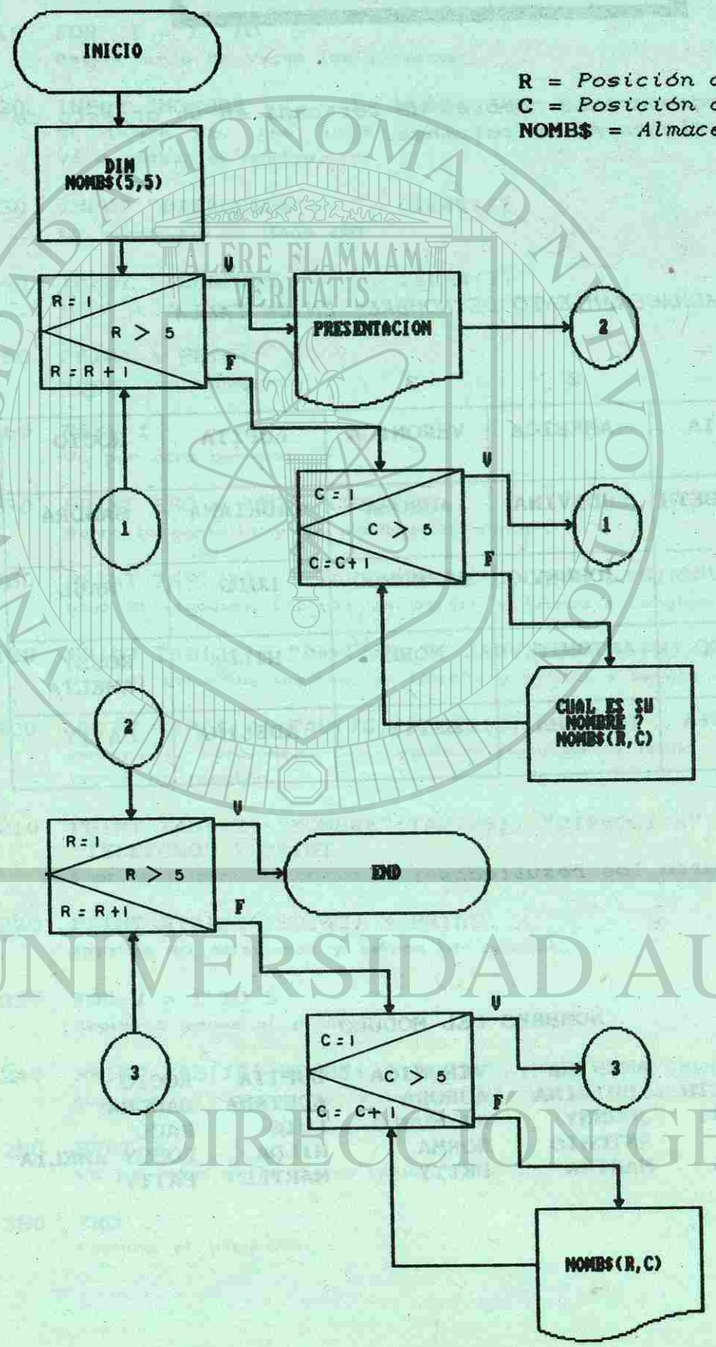
ALMACENAMIENTO DE NOMBRES EN LA TABLA

	1	2	3	4	5
1	ALICIA	AMERICA	VERONICA	LUPITA	ROCIO
2	ELIZABETH	LUDIVINA	AURORA	ADRIANA	SANDRA
3	HUGO JR.	JOHNNY	REYMUNDO	LUIS	RAUL
4	HOMERO	ANTONIO	NORMA	HILDA	ROSSY AMELIA
5	CLAUDIA	MARTHA	BETTY	ADELINA	PATTY

Así se desplegarán los resultados :

```
                    NOMBRES DEL MODULO

ALICIA      AMERICA     VERONICA    LUPITA      ROCIO
ELIZABETH   LUDIVINA    AURORA      ADRIANA     SANDRA
HUGO JR.    JOHNNY      REYMUNDO    LUIS        RAUL
HOMERO      ANTONIO     NORMA       HILDA       ROSSY AMELIA
CLAUDIA     MARTHA      BETTY       MARTEL      PATTY
```

DIAGRAMA DE FLUJO



R = *Posición del renglón*
C = *Posición de la columna*
NOMB$ = *Almacena el nombre*

CODIFICACION

```
010  REM PROGRAMADOR : ELIZABETH
020  REM MATRICULA   : 841202
030  REM GRUPO       : 2 A
040  REM OBJETIVO    : MANEJO DE 2 DIMENSIONES
050  REM FECHA       : 02-DIC-84
     Comentarios del programa.

100  DIM NOMB$(5,5)
     Crea un arreglo de 5 por 5.

110  FOR R = 1 TO 5
     Repite 5 veces el proceso, donde
     " R " es el control del renglón.

120  FOR C = 1 TO 5
     Repite 5 veces el proceso, donde
     " c " es el control de la columna.

130  INPUT "NOMBRE DE LA PERSONA  ";NOMB$(R,C)
     Almacenamos el nombre en NOMB$ en la
     posición de R,C.

140  NEXT C          ! Vá por otro nombre.

150  NEXT R          ! Vá por otro renglón.

160  PRINT TAB(31); "NOMBRES DEL MODULO"  \  PRINT
     Deja 31 espacios, imprime el encabezado y
     brinca un renglón.

170  FOR R = 1 TO 5
     Repite 5 veces el proceso, donde
     " R " es el control del renglón.

180  FOR C = 1 TO 5
     Repite 5 veces el proceso, donde
     " C " es el control de la columna.

190  PRINT NOMB$(R,C),
     Imprime el nombre de la posición de R,C
     del modulo.

200  NEXT C
     Vá por otro nombre del módulo

210  PRINT
     Brinca un renglón.

220  NEXT R
     Vá por otro renglón del módulo.

230  END             ! Termina la ejecución.
```

8.4 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Para qué nos sirve utilizar el estatuto* **DIMENSION.**

_____
_____
_____

2.- *Escriba el formato de codificación del* **DIMENSION.**

_____
_____
_____

3.- *Cuál sería el tamaño máximo de un* **ARREGLO.**

_____
_____
_____

4.- *Mencione 4 ejemplos donde se pueda utilizar* **ARREGLOS.**

_____
_____
_____

INDICE ANALITICO

8.4 AUTOEVALUACION DEL CONOCIMIENTO

1.- *Para qué nos sirve utilizar el estatuto* **DIMENSION.**

_____
_____
_____

2.- *Escriba el formato de codificación del* **DIMENSION.**

_____
_____
_____

3.- *Cuál sería el tamaño máximo de un* **ARREGLO.**

_____
_____
_____

4.- *Mencione 4 ejemplos donde se pueda utilizar* **ARREGLOS.**

_____
_____
_____

INDICE ANALITICO

GLOSARIO

ALGORITMO .-

Serie de pasos lógicos para llegar a un objetivo.

ASCCI .-

American Standar Code for Information Interchange.
Código americano estandar para el intercambio de información.

BASIC .-

Beginner's All-purpose Symbolic Instruction Code.
Código simbólico de propósitos general para la enseñanza a principiantes.

CARACTER .-

Es un número, letra o signo especial.

CODIFICAR .-

Serie de instrucciones de un lenguaje de alto nivel.

COMPUTADORA .-

Conjunto de dispositivos electrónicos y mecánicos que sirven para almacenar grandes volumenes de información, procesa y nos da resultado.

COMANDO .-

Instrucción de control para realizar un trabajo específico de un computador.

DIAGRAMA DE FLUJO .-

Representación gráfica de los pasos logicos para resolver un problema.

DIRECTORIO .-

Almacen de programas.

DISPOSITIVOS .-

Equipo externo que se encuentra conectado al computador, puede ser de entrada o salida.

ESTATUTO .-

Son las palabras reservadas por el lenguaje de alto nivel.

INPUT / OUTPUT .-

Nos indica si es entrada o salida.

INTERACTUAR .-

Es una comunicación en forma directa.

PROCEDIMIENTO .-

Es la secuencia de varias operaciones.

PROGRAMA .-

Serie de instrucciones de un lenguaje de alto nivel.

B I B L I O G R A F I A

INTRODUCCION AL PROCESAMIENTO DE DATOS
PARA LOS NEGOCIOS

ORILIA

Segunda edición

Mc. Graw Hill

PDP-11 BASIC-PLUS-2

LENGUAJE REFERENCE MANUAL