UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

APUNTE 206

PRACTICAS DEL

LABORATORIO DE SISTEMAS DE COMPUTO

ENRIQUE RIVERA MEDINA

DIVISION DE INGENIERIA MECANICA Y ELECTRICA
DEPARTAMENTO DE COMPUTACION

FI/DIME/87-009

I N D I C E .

P r e s e n t a c i ó n .

Las prácticas para el laboratorio de Sistemas de Cómputo están estructuradas de la siguiente manera :

La primera práctica es un repaso del lenguaje Basic, conceptos básicos de computación y del editor de Basic . El núcleo central de esta práctica se encuentra en el repaso de Basic , viniendo dentro de los apéndices el demás material que deberá cubrirse en esta práctica, o sea "Cómo usar la computadora Radio Shack", "Técnicas de programación" y "Editor" .

La segunda práctica corresponde al manejo de cadenas de caracteres en Basic , imprescindible para aplicaciones no numéricas . Dentro de la práctica vienen descritas las funciones y los procedimientos de Basic para realizar este manejo , ilustrados con ejemplos .

La tercera práctica consiste en la primera parte del tema "Archivos" , concerniente al uso de los archivos secuenciales , su diferencia con los archivos directos , cómo se crean y cómo se usan .

En la cuarta práctica se termina el tema de archivos viendo el uso y creación y carga de archivos directos . En el ejercicio correspondiente a esta práctica se comenzará a realizar el proyecto final del curso , es conveniente cubrir el apéndice relacionado con el mismo , "Nómina" .

La quinta práctica cubre el tema de ordenamientos , estudiando dos ordenamientos : el Burbuja y el Shell . Se describe su funcionamiento y algoritmo correspondiente .

En la sexta práctica se ven dos métodos de búsqueda : la búsqueda lineal y la búsqueda binaria .

En los ejercicios correspondientes a la quinta y sexta prácticas se continúa desarrollando el proyecto de nómina , de tal forma que al concluir la sexta práctica este deberá de estar bastante avanzado , restando sólo hacer el cálculo de la nómina y algunos reportes para finalizarlo .

Los apéndices de las prácticas son los siguientes :

"Cómo usar la computadora Radio Shack", que da las orientaciones básicas para hacer uso de esta computadora .

"Técnicas de programación", donde se dan orientaciones para programar .

"Uso del editor de Basic" .

"Guía rápida" , conteniendo de manera resumida las instrucciones de Basic , los comandos del editor y del TRSDOS .

"Errores" , donde se da la explicación de los errores que manda la computadora en Basic y se indica su posible causa .

"Basic en la computadora Altos 5086 " , donde se ven las principales diferencias entre el basic de la microcomputadora Radio Shack y el basic de la microcomputadora Altos .

"Nomina" , se describe un sistema de nómina en general y en particular se especifica el sistema de nómina que se desarrollará como proyecto final .

"Aplicaciones" , se describen algunas aplicaciones de la computación .

Finalmente viene una bibliografía donde se podrá consultar más ampliamente los temas desarrollados en las prácticas .

Enrique Rivera Medina .
Octubre 1984 .

[illegible]

Esta práctica tiene el propósito de ser un repaso de las principales instrucciones del lenguaje BASIC de programación,del que se deben de tener antecedentes por la materia de "Computadoras y Programación" . En caso de no tener [illegible] se recomienda la compra de cualquier libro [illegible] . Al final de las prácticas viene una bibliografía que podrá orientar al respecto . En la Facultad se imparte [illegible] de "Computadoras y Programación" que cubren [illegible] este propósito .

El [illegible] tiene como ventajas el ser bastante comprensible,"hablar a las personas",fácil y agradable de aprender .

En la PAC Model II se carga el BASIC desde el sistema operativo con solo teclear la palabra BASIC . Cuando se [illegible] archivos,se debe especificar su número en esta fase y en la instrucción correspondiente se verá con más detalle [illegible] estando en BASIC podemos regresar al nivel de sistema operativo tecleando la palabra SYSTEM .

Constantes y Variables .

El Basic [illegible] nos referimos a partir de este momento es el de la Radio Shack Model II .

El Basic puede usar varios tipos de datos:

Numéricos : Se representan cantidades y con los que podemos hacer las operaciones matemáticas comunes .

Cadenas : Son [illegible] con las que podemos hacer operaciones [illegible] especiales. (ver práctica de "Cadenas de caracteres") .

En los [illegible] numéricos el Basic nos proporciona tres [illegible] precisión simple y de doble precisión,cuyos [illegible] tipos ocupan,respectivamente,2,4 y 8 bytes para almacenar la cantidad ,teniendo por lo tanto distinto [illegible] de exactitud.

En la declaración de las variables se pone un signo para discriminar entre los [illegible] tipos : " % "," ! " y " # " para enteros,simple y doble,respectivamente . Las variables de tipo cadena [illegible] el caracter "$" .

Ej. ENT% , [illegible] , DOBLEP# , CADENA$.

Para los nombres de variables,el primer carácter debe de ser letra y los demás pueden ser letras o números . La computadora solo [illegible] los dos primeros .

Los operadores que se pueden usar en BASIC son de tres tipos : numéricos,lógicos y de cadenas de carácteres .
Los operadores numéricos son : + , - , * , / , \ (división entera) , ^ (exponenciación),MOD (residuo) .
EJ . 7 \ 2 nos da 3 y 7 MOD 3 nos da 1 .
Los operadores lógicos son <,>,<= o =<,>= o =>,<> o >< (operadores de relación) y AND, OR, XOR, EQV , IMP, NOT (operadores booleanos) .
Los operadores de cadenas se verán en la práctica correspondiente .

I n s t r u c c i o n e s B á s i c a s .

Las " instrucciones de comando " son :

-AUTO comienzo,incremento .
Numera líneas automáticamente desde el número "comienzo" con un incremento "incremento"

-DELETE comienzo-fin .
Borra las líneas comprendidas entre "comienzo" y "fin" .

-KILL "archivo" .
Borra el archivo "archivo" del disco .

-LIST comienzo-fin .
Despliega las líneas entre "comienzo" y "fin" .

-LLIST comienzo-fin .
Igual que list,pero hacia la impresora .

-LOAD "archivo" .
Carga el programa "archivo" en Basic .

-NAME "nombreviejo" TO "nombrenuevo" .
Renombra un archivo .

-NEW .
Borra el programa de memoria RAM .

-RENUM nuevalínea,comienzo,incremento .
Renumera el programa desde "comienzo",poniendo como primera línea "nuevalínea" con un incremento de "incremento"

-RUN .
Ejecuta el programa. Se le puede indicar que programa o bien a partir de que línea empiece a ejecutar .

-SAVE "archivo",A .
Salva el programa "archivo" en disco. Si se le pone la opción "A" se guarda en formato ASCII,si se omite se guarda en forma comprimida .

-SYSTEM "comando" .
Ejecuta el comando del sistema y regresa a BASIC . Si el "comando" se omite entonces se regresa al sistema únicamente .

Las [illegible] de Definición e Iniciación son :
[illegible] de Memoria .
[illegible] variables numéricas,nulas las cadenas;
Proveer [illegible] para cadenas de caracteres.

[illegible] .
[illegible] del Programa,estos datos pueden ser
numéricos [illegible] de caracteres,deben de estar
separados [illegible] los datos así guardados podrán ser
leídos [illegible] instrucción READ .

[illegible]
[illegible] variables cuya primer letra este
comprendida [illegible] letras será de tipo numérico de
doble [illegible]

[illegible]
[illegible] las variables enteras .

[illegible] de letras .
[illegible] para precisión sencilla .

[illegible] de letras .
[illegible] para variables tipo cadena .

[illegible] Función (argumento1,...) = fórmula .
Define [illegible] de nombre "nombre de función" con
argumento [illegible] .
10 [illegible] PESOS = 1000 * DOLARES
20 [illegible] dólares ";DOLARES
30 [illegible] en pesos...";FNPESOS(DOLARES)

[illegible] (lista de dimensiones),arreglo2(lista de
dimensiones)
[illegible] variables que serán arreglos
de elementos [illegible] DIM MATRIZ(10,14) .

[illegible]
[illegible] devuelve el espacio en memoria
para [illegible]

[illegible]
[illegible] puede usar también el
apostrofe [illegible] . ' REM Esto es un comentario .

[illegible]
[illegible] lea desde el primer DATA del
programa.

[illegible] de una declaración .
[illegible] separación .
Sirve [illegible] asignaciones directas,es optativa .

[illegible]
[illegible] asigna el valor leído a la variable
correspondiente.

[illegible] variable1 , variable2 .
[illegible] valores de las variables 1 y 2 .

Instrucciones de secuencia de programa .

-END .
Termina el programa (en el Modelo II es optativa.

-FOR / NEXT .

FOR variable = valor inicial TO valor final STEP incremento .

Esta instrucción establece una secuencia iterativa de tipo "HAZ MIENTRAS"(do while). Ej.

```
10 PRINT "Se harán 103 pasadas"
20 FOR INDICE% = 1 TO 103
30         PRINT "PASADA NUMERO  : ";INDICE%
40 NEXT INDICE% 'END FOR
```

-GOSUB línea .
Va a una subrutina especificada . Al terminar la realización de la subrutina el control del programa pasa a la siguiente línea del GOSUB. La subrutina se termina de ejecutar cuando se encuentra un RETURN .

-GOTO línea .
Transfiere el control del programa incondicionalmente a la línea "línea" .

-IF...THEN...ELSE .
Bifurcación condicional .

IF expresión booleana THEN instrucción o número de línea ELSE instrucción o número de línea .
ELSE ... es opcional . Si la condición es verdadera se realiza lo especificado por el THEN,si es falsa se realiza la parte del ELSE .

-ON ... GOTO .
Pregunta por un valor y salta a diferentes líneas del programa,según el caso .
ON valor de prueba GOTO línea1,línea2,...
El valor de prueba es una expresión numérica desde 0 hasta 255 .
Esta instrucción se ajusta a la figura del CASE . Si para un valor no se encuentra la línea correspondiente,se pasa a la siguiente instrucción del ON...GOTO .

-ON...GOSUB .
Similar a la anterior,pero con subrutinas.

-RETURN .
Retorno de la subrutina al programa donde fue llamada .

/ Salida .

XmBIC se encuentra esta
ión del programa y pide se le
...nos valores "var1","var2",etc .
un carácter "?" .

el cursor en la esquina

PRINT "HOLA";A,B

pero el despliegue es hacia la

siones de entrada/salida que
...entos directos , en la práctica
...plicadas . En cuanto a las
... tienen variantes que en este
...er el objetivo del mismo .

...que maneja el BASIC se localiza en

s .

...serie numérica de la forma
...los dos elementos de la
...se encuentran sumando los dos
...(K-1)+T(K-2) . Un programa
...meros n términos de la serie es

...minos de la serie de Fibonacci.

...de términos que deseas (N>1)";N%

...no. ";I%;" = ";SEG%

... ? [SI,NO] ";SIGUE$

...Chan .

```
No. 2 .
'Programa que resuelve la ecuación cuadrática .

10REM Este programa nos da las raices de la ecuación
20REM cuadratica Ax^2+Bx+C=0 , Indica  que  tipo  de
30REM raices encuentra.
40'     Inicia el programa principal .
50SIGUE$="SI"
60'WHILE SIGUE$ = "SI"
70 IF SIGUE$ = "SI" THEN 80 ELSE 250
80      INPUT "> Dame los coeficientes A,B,C "
90      IF A=0 THEN 100 ELSE 110
100               GOSUB 300                    ' ECUACION LINEAL
110               GOTO  220
120      ELSE
130               D = B^2 - 4*A*C
140               IF D > 0 THEN 140 ELSE 150
150                        GOSUB 350           ' RAICES REALES DISTINTAS
160                        GOTO  220
170               'ELSE
180                        IF D = 0 THEN 170 ELSE 180
190                                 GOSUB 400  ' RAICES REALES IGUALES
200                                 GOTO  220
210                        'ELSE
220                                 GOSUB 450  ' RAICES COMPLEJAS
230                        'ENDIF
240               'ENDIF
250     'ENDIF
260     INPUT "> Quieres continuar ? [ SI , NO ] ";SIGUE$
270     GOTO 70
280'FIN MAZ
290END   ' P R O G R A M A   P R I N C I P A L
300'
310     S u b r u t i n a s
320'
330REM  ECUACION LINEAL
340     RAIZ = - C / B
350     PRINT "> La ecuación es lineal ,su raiz es :";RAIZ
360RETURN        'ECUACION  LINEAL
370'
380REM  RAICES REALES DISTINTAS
390     R1 = - B / (2*A) + SQR (D) / (2*A)
400     R2 = - B / (2*A) - SQR (D) / (2*A)
410     PRINT "> La ecuación tiene raices reales distintas: ";R1,R2
420RETURN        'RAICES REALES DISTINTAS
430'
440REM  RAICES REALES IGUALES
450     RAIZ = - B / (2*A)
460     PRINT "> La ecuación tiene raices reales iguales: ";RAIZ
470RETURN        'RAICES REALES IGUALES
480'
490REM  RAICES COMPLEJAS CONJUGADAS
500     REAL = - B / (2*A)
510     IMAG = SQR ( -D ) / (2*A)
520     PRINT "> Las raices son complejas conjugadas : "
530     PRINT "> Parte real = ";REAL;" imaginaria = ";IMAG
540RETURN        'RAICES COMPLEJAS CONJUGADAS
550'
```

No. 3 .
Programa que crea una tabla con nombres y edades .

```
 10 REM Este programa hace uso de dos arreglos , NOMBRE$ y EDAD%
 20 REM Pide los datos del teclado y despues los escribe .
 30 REM El máximo de datos por tabla es de diez .
 40 CLEAR 300 ' Espacio a las cadenas (ver prácticas de cadenas)
 50 DIM NOMBRE$(10),EDAD%(10) ' Son 11 elementos,del cero al diez
 60 FOR I% = 1 TO 10
 70      INPUT "> Dame el nombre ";NOMBRE$(I%)
 80      INPUT "> Dame la edad   ";EDAD%(I%)
 90 NEXT I% 'ENDFOR
100 PRINT "> La tabla creada es la siguiente : "
110 PRINT
120 FOR I% = 1 TO 10
130      PRINT "*** Nombre : ";NOMBRE(I%);" *** Edad : ";EDAD%(I%)
140 NEXT I% 'ENDFOR
150 PRINT "> Es todo lo que el programa hace ..."
160 END
```

E J e r c i c i o s .

I. Al ejemplo anterior agregar una subrutina que pida un determinado nombre,lo busque en la tabla y de la edad correspondiente . Si no lo encuentra que lo indique y lo meta en la tabla,pidiendo su edad . Otra subrutina debe de pedir la edad,buscarla en la tabla y dar el nombre correspondiente.
Caso de no hallarla ,debe de meterla a la tabla y pedir el nombre correspondiente . Al finalizar el programa debe de indicar a la persona con la mayor edad y la menor edad .
En caso de que exista más de una persona con una determinada edad,debe de indicar todas estas personas . Se debe de entregar un listado del programa y uno de la ejecución,mostrando la introducción inicial de las tablas y los dos procesos realizados. Deben de ponerse al menos dos personas con la misma edad en las tablas de datos .

II. Haga un programa que muestre los números primos que se encuentran entre 1 y 300 . Un número primo es aquel que sólo se puede dividir exactamente entre uno y entre sí mismo.
EJ . 11,13,17,19 .

C A D E N A S D E C A R A C T E R E S .

I n t r o d u c c i ó n .

La comunicación escrita del ser humano se realiza básicamente a través de carácteres y no de datos numéricos,por eso es básico que una computadora tenga la habilidad de procesar datos alfanuméricos,compuestos de series o cadenas de carácteres (strings) .

Las computadoras se han asociado tradicionalmente con la solución de problemas numéricos,tales como raíces de ecuaciones,cálculos de tensiones y otras agradables aplicaciones que tienen como objetivo primordial obtener una respuesta numérica.

En la actualidad el procesamiento de problemas no numéricos es cosa corriente,problemas tales como traducción de lenguajes,edición de textos,reconocimiento de patrones y manipulación simbólica de ecuaciones matemáticas. En este tipo de problemas las respuestas numéricas ocurren pocas veces .

La entidad fundamental de una cadena de carácteres es el carácter . El conjunto de carácteres que puede manipular una computadora como la Radio Shack es el llamado alfabeto ASCII.
El códiso ASCII (American Standard Code for Information Interchange) es una manera de representar internamente en la computadora los carácteres .

El conjunto de carácteres ASCII se puede dividir en :

1)Letras minúsculas y MAYUSCULAS del alfabeto inglés.

2)Dígitos (0 ... 9) .

3)Carácteres especiales (+,-,/,=,!,@,;,etc) .

4)Carácteres de control tales como : STX=start of text,ACK=acknowledge,CR=carriage return,FS=file separator,etc

Otros conjuntos de carácteres pueden incluir símbolos matemáticos,letras griegas,carácteres de graficación , etc .

En la computadora un carácter es representado internamente como una serie de bits,en el caso del código ASCII,una serie de siete bits . Este conjunto de bits tiene una equivalencia numérica,por ejemplo,el carácter "A" se representa en binario por "1000001" que en decimal equivale a 65 . El carácter "B" se representa en binario por "1000010" que en decimal es 66 . Mediante esta equivalencia numérica se puede entonces tener relaciones de magnitud entre carácteres y podemos afirmar que "A" < "B" dado que 65 < 66 . Generalizando tenemos que :

"El milusos" = "El milusos" <> " El Milusos "

"ANACLETA" < "LUISA" < "MACLOVIA" < "ROSA" < "SEGISMUNDA"

Cadenas de carácteres en BASIC

.Declaración.

Las variables de tipo "cadena de carácteres" se declaran en BASIC agregando al nombre de la variable el carácter "$". EJ. cadena$.
Las constantes de este tipo se limitan entre comillas . EJ. NOMBRE$="BACO"

Podemos definir que las variables que se inicien con determinada letra o letras sean de tipo "cadena de carácteres" mediante la instrucción "DEFSTR". ej.
DEFSTR A-I
significa que las varibles cuya primer letra sea A,B,C,D,E,F,G,H o I serán de este tipo,entonces la variable DIRECCION es tipo "cadena" y la variable JUTLANDIA no lo es, dado que comienza con J .
Podemos declarar arreglos de tipo cadena de carácteres,p.ej. DIM RENGLON$(10) . En este caso,cada elemento RENGLON$(i) sera una cadena de carácteres.

La máquina almacena el contenido de las cadenas en una área de memoria principal especial,el tamaño de esta área no es fijo y lo podemos determinar mediante la instrucción CLEAR,esta instrucción reserva el número de bytes que indiquemos y los emplea en el almacenamiento de cadenas de carácteres . La instrución tiene el efecto de poner en cero todas las variables numéricas y hacer nulas todas las cadenas de carácteres .

El espacio inicial que da la máquina es de 100 bytes , el espacio que en un momento la máquina este destinando para almacenamiento de cadenas se puede obtener mediante la función FRE . El argumento de FRE en este caso debe de ser una cadena cualquiera,ej FRE(" ") nos dara 100,si no hemos colocado un clear que modifique el espacio inicial. Si el argumento de FRE es numérico nos dará la cantidad total de memoria disponible,ej. FRE(8) nos dará 31491 .

Ej. CLEAR 80 , inicia las variables a reser... de memoria para las cadenas. CLEAR , inicia las ... no modifica el espacio para cadenas .

Cuando excedemos el valor que reservamos la ... marca un error de "Out of String Space" . Si ... utilizar cadenas en un programa es conveniente ... CLEAR 0 para que sea más eficiente .

E n t r a d a / S a l i d a .

INPUT .

La instrución input permite introducir valores ... teclado . La cadena de carácteres que introduzcamos ... estar limitada por comillas en caso de que contenga co... puntos o blancos al principio.

Ej .

```
INPUT ">Dame la filosofía de tu vida "; SIESQUELA...
```

LINE INPUT .

Es similar a INPUT ,excepto que no des... interrosación cuando está esperando los datos,s... leerse una variable y no es necesario limitar la ca... comillas,los blancos al principio los toma como parte ... cadena. La forma de terminar la introducción de un... es mediante un ENTER .

Ej .

```
LINEINPUT "> Mejor dame tu deporte favorito "; DA...
```

INPUT$.

Permite introducir cadenas de un tamaño determina... pronto se termina de teclear el último carácter requer... ejecución del programa continúa . (No se necesita pres... ENTER.) Los carácteres tecleados no son desplegados ... pantalla .

Ej .

```
1 PRINT "> Dame el mayor secreto de tu vidaven 10 ca... .
2 SECRETO$ = INPUT$ ( 10 )
```

READ / DATA .

Read se usa en conjunción con Data .

Ej.

```
1 READ GUSTOS$,AFICIONES$,VIDAFUTURA
2 DATA " BACARDIACOS ","  LEVANTAMIENTO DE TARRO ", 0
```

INKEY$.

Esta función nos retorna un carácter si es presionado desde el teclado,en caso de que no sea tecleado retorna una cadena nula de carácteres(de longitud cero). El carácter tecleado no es desplegado a la pantalla . Para que tome por fuerza un carácter se le debe de colocar dentro de un ciclo iterativo . La cadena nula es representada por "" .

Ej.

```
1 A$ = INKEY$ : IF A$ = "" THEN 1 ELSE PRINT A$
```

PRINT .

Se utiliza se la misma forma que para variables numéricas.

EJ . PRINT A$

Operaciones de cadenas .

Se pueden hacer comparaciones entre variables o constantes de tipo cadena de carácteres . Los símbolos válidos para hacer operaciones son : = , > , < , <= , >=. Cuando se usen constantes,deben ponerse entre comillas.

La operación de concatenación consiste en unir varias cadenas de carácteres. Se realiza con el operador "+" .

Ej.

```
10 A$="HOLA"
20 B$="QUE ONDA"
30 C$=A$ + " , " + B$ + "?"
40 PRINT C$
```

La impresión resultante será : HOLA , QUE ONDA?

F u n c i o n e s d e c a d e n a s .

CHR$.

La función CHR$ nos proporciona el carácter especificado por el argumento,este carácter puede ser ASCII o de gráfica . El argumento puede variar en el ranso de 0 a 255 . los carácteres de graficación se hallan del 128 al 159 .

EJ. CHR$(65) nos dara el carácter "A" .

ASC .

Esta función es la inversa de la anterior,da el código correspondiente al carácter que demos como argumento,si es una cadena la que damos nos da el código del primer carácter.

EJ .

```
1 CLS
2 PRINT " CARACTERES DE GRAFICACION "
3 FOR I% = 128 TO 159
4PRINT "CODIGO :";I%;" IGUAL A ";ASC(CHR$(I%));" CARACTER ";CHR$(I%)
5 NEXT I%
```

LEN .

Esta función nos proporciona la lonsitud de la cadena especificada . EJ. LEN ("JUA! JUA!") nos proporciona el valor nueve.

LEFT$.

Nos proporciona la porción izquierda de una cadena. La sintaxis es la sisuiente:

LEFT$(cadena,numero)

Donde número es la cantidad de carácteres que deseamos sean tomados de la parte izquierda de la cadena .

RIGHT$.

Es la función similar a left$,con la diferencia de que actúa en la parte derecha de la cadena .

EJ.

```
1 INDICA$ = "PARTE IZQUIERDA ---- PARTE DERECHA"
2 IZ$     = LEFT$ ( INDICA$ , 15 )
3 DE$     = RIGHT$( INDICA$ , 13 )
```

Entonces IZ$ contendrá "PARTE IZQUIERDA" y DE$ "PARTE DERECHA".

MID$.

Esta instrucción nos trae una subcadena de la cadena especificada.

Sintaxis :

MID$ (cadena , posición , longitud)

La instrucción toma de la "cadena",a partir de la posición "posición" un número "longitud" de bytes .

EJ.

```
1 SALUDO$ = " BUENOS DIAS , TROMPAMERO "
2 PRINT MID$(SALUDO,9,4)
```

La impresión resultante es : DIAS .

Mediante la función MID$ se pueden implementar las funciones left$ y right$,por lo que se puede decir que :

```
LEFT$(A$,N)  = MID$ (A$,1,N)

RIGHT$(A$,N) = MID$ ( A$, LEN(A$)-N+1 , N)
```

Esta instrucción tiene una segunda forma de uso que es la siguiente :

```
MID$(cadena original,posición,longitud)=cadena reemplazante
```

Tiene el efecto de reemplazar en la "cadena original" a partir de la posición "posición" un número "longitud" de bytes tomados de la "cadena reemplazante".

EJ.

```
1 MENTIRA$ = " PERO QUE BIEN TE VES HELENITA ! "
2 MID$ ( MENTIRA$,11,4 ) = "FEO "
3 MID$ ( MENTIRA$,23,8 ) = "MONSTRUO"
4 PRINT MENTIRA$        ' O el otro yo del doctor Merengue
```

La impresión resultante seria :

PERO QUE FEO TE VES MONSTRUO !

INSTR .

La sintaxis de esta instrucción es la siguiente :

INSTR (posición , cadena principal , subcadena)

Esta función nos regresa un número que indica la posición dentro de la cadena principal donde la subcadena comienza. La "posición" indica desde que posición se comienza a buscar en la cadena principal . Si no se encuentra la subcadena ,la función regresa un cero.

EJ.

```
10 A$ = "HOLA BILL"
20 PRINT INSTR ( 1 , A$ , "BILL" )
```

La impresión seria : 6 . Si la posición se omite se toma como 1 .

SPACE$.

Esta función nos regresa una cadena de espacios blancos de longitud determinada .

Ej.

```
10 BLANCOS$ = SPACE$(20)
```

Tendremos veinte espacios en blanco en la cadena BLANCOS$.

STR$.

Esta función convierte un argumento numérico en cadena de carácteres.

Ej.

```
10 NOESNUMERO$ = STR$ ( 34.34 )
```

En NOESNUMERO$ se tendra la cadena " 34.34" con la que se podrán hacer concatenaciones,etc ;pero no sumas,multiplicaciones,etc.

VAL .

Esta función evalúa el string que le demos como argumento,es la inversa a la función STR$.

Ej.

```
10 PRINT VAL("100 MUGROSOS PESOS")   ' Imprime 100
20 PRINT VAL("1234E5")               ' Imprime 1.234E+08
30 B = VAL ( "3" + "5" + "*" + "2") ' Asigna a B el número 35
40 PRINT B
```

STRING$.

Esta función nos regresa una cadena de longitud y carácteres dados.

Ej.

```
10 ASTERISCOS$ = STRING$ (35,"*")
```

En la variable asteriscos$ tendremos treinta y cinco asteriscos .

DATE$.

Esta función toma la fecha de la máquina y la mete en una cadena de carácteres . Si nosotros,un buen día sábado 28 del primaveral mes de abril (ah! las rosas...) del año de 1979 hubieramos usado esta función DATE$,tendríamos un string con los siguientes carácteres : SATAPR281979118 45 . Nos estaría informando que ese feliz y dulce día fue el día 118 del año en el cuarto mes del mismo y que fue el quinto de la semana (tomando al lunes como cero). Ej. PRINT DATE$.

TIME$.

Esta función nos trae la hora del día .

Ej.

```
10 A$ = TIME$ : PRINT A$
```

Obtendríamos como impresión , por ejemplo 14.47.18 en formato hh.mm.ss .

P r o g r a m a E j e m p l o .

```
5  CLS
8  '
9  '
10 '     P r o g r a m a   E j e m p l o
20 '     R F C   Resistro Federal de Causantes
22 '     Este programa lee registros con el siguiente formato :
24 '     Nombre  Apellido Paterno  Apellido Materno  dd/mm/aaaa
25 '     Entre el nombre y ap.pat,ap.pat y ap.mat,ap.mat y dd/mm/aaaa
28 '     puede haber una cantidad indeterminada de espacios en blanco
30 '     Ej.   BRIAGOBERTO    MEMELAS       TLACHICOTON  23/01/1941
32 '     El programa lee el resistro y nos entresa el RFC:METB-410123
36 '
38 '     P R O G R A M A   P R I N C I P A L .
40 '
45 PRINT "> Resistro = Nombre  Ap.Pat  Ap.Mat  dd/mm/aaaa"
50 LINE INPUT "> Da resistro,para terminar programa teclea FIN : ";A$
55'Do While
60 IF A$ <> "FIN" THEN 65 ELSE 300
65       I = 1
70       GOSUB   1000     ' quitablancos
80       N$ = MID$ (A$,I,1)
90       GOSUB   1120     ' quitahalla
100      AP$ = MID$(A$,I,1)
102      I = I + 1
104      GOSUB   1160     ' hallavocal
106      AP$ = AP$ + MID$ (A$,I,1)
110      GOSUB   1120     ' quitahalla
120      AM$ = MID$ (A$,I,1)
130      GOSUB   1120     ' quitahalla
140      DD$ = MID$ (A$,I,2)
150      MM$ = MID$ (A$,I+3,2)
160      AA$ = MID$ (A$,I+8,2)
170      RFC$= AP$ + AM$ + N$ + "-" + AA$ + MM$ + DD$
180      PRINT "> El RFC de ";A$;" es ";RFC$
185      PRINT "> Resistro = Nombre  Ap.Pat  Ap.Mat  dd/mm/aaaa"
190      LINE INPUT "> Da resistro,para terminar teclea  FIN : ";A$
200      GOTO 60
300'END WHILE
305'
310 END '        P r o g r a m a   R F C .
320'
```

```
997'
998'     S u b r u t i n a s .
999'
1000REM SUBRUTINA QUITABLANCOS
1001' Esta rutina recorre el apuntador I mientras
1002' el carácter de la cadena A$ sea blanco .
1010     IF MID$(A$,I,1) = " " THEN 1020 ELSE 1040
1020              I = I + 1
1030              GOTO 1010
1040     'ENDWHILE
1050RETURN        'END QUITABLANCOS
1051'
1052'
1053'
1060REM SUBRUTINA HALLABLANCOS
1061' Esta subrutina recorre el apuntador I mientras
1062' el carácter de la cadena A$ sea distinto de blanco .
1070     IF MID$(A$,I,1) <> " " THEN 1080 ELSE 1100
1080              I = I + 1
1090              GOTO 1070
1100     'ENDWHILE
1110RETURN        'END HALLABLANCOS
1111'
1112'
1119'
1120REM SUBRUTINA HALLAQUITA
1121' Esta rutina recorre el apuntador I saltando los carácteres
1122' no blancos¥una vez hecho esto recorre los carácteres blancos
1123' hasta hallar el siguiente carácter distinto de blanco .
1130     GOSUB 1060        ' HALLABLANCOS
1140     GOSUB 1000        ' QUITABLANCOS
1150RETURN        'END HALLAQUITA
1151'
1152'
1153'
1160REM SUBRUTINA HALLAVOCAL
1170' Esta rutina recorre el apuntador I mientras
1180' el carácter de la cadena A$ sea distinto de vocal .
1190     VOCALES$ = "AEIOU"
1200     IF INSTR(1,VOCALES$,MID$(A$,I,1))=0 THEN 1210 ELSE 1230
1210              I = I + 1
1220              GOTO 1200
1230     'ENDWHILE
1240RETURN        'END HALLAVOCAL
1250'
1260'
1270'
```

E j e r c i c i o s .

I . ANALISIS DE FRASES .
El programa debe leer hasta 10 líneas de texto .
Se consideran palabras las cadenas de caracteres delimitadas por blancos,",",".",";","y",":"
El objetivo del programa es obtener una tabla de dos columnas,la primera conteniendo las palabras que aparecen en el texto y la segunda la frecuencia con que aparecen .
ej. "Nací sin saber por qué,viví sin saber por qué y muero sin saber por qué":(Gassendi)
tabla : palabra frecuencia
sin 3
saber 3
por 3
qué 3
y 1
nací 1
viví 1
muero 1
El algoritmo general es el siguiente :
1 . Se lee el texto .
2 . Repite mientras haya texto hasta el paso 5 .
3 . Busca siguiente palabra de texto .
4 . Si la palabra encontrada no está en la tabla insértala al final de la misma .
5 . Actualiza la columna de frecuencias .
6 . Imprime el texto y la tabla .

II . E S P I O N A J E .
La humanidad le ha concedido (mediante unos dolarcillos de por medio) el contrato de un programa codificador de texto que debe de realizar el siguiente :
Lee un texto y por cada carácter lo mapea según la siguiente tabla :
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.;:,!?M$b (b=espacio blanco)
KZLXHYNVUPTRSQ4I3UCVIWO.;:,!?M$JIHGFEDCBbA
ej. HOLA 7?$K
Ahora el texto es invertido.
ej. 9R7N K$?A

III . C O N A E S P I O N A J E .
Mediante el desgravamen deducible de impuestos,de unos cuantos rublos conseguimos de la KUR el contrato del programa decodificador del texto cifrado de la CIA .

Para ti :
Realizar el programa cifrador y descifrador que muestre :
a) El texto a encriptar .
b) El mensaje cifrado .
c) El mensaje descifrado .

IV . Q U I T A B L A N C O S .
Modifica la rutina de quitablancos mostrada en el programa ejemplo para que vigile que el contador 'i' no sobrepase a la longitud de la cadena leída y como sería en el caso de dar una cadena con espacios únicamente .

A R C H I V O S

Parte uno.

El BASIC Model II nos proveé de dos medios para el acceso a archivos:

-Secuencial:se empieza a leer o escribir desde el principio del archivo;subsecuentes lecturas o escrituras son hechas en las siguientes posiciones del archivo . En este modo el número de carácteres leídos o escritos puede variar.

-Directo:se lee o escribe en cualquier registro del archivo que se especifique (también se le llama acceso aleatorio). Aquí los datos leídos o escritos deben estar en bloques de longitud fija llamados registros .

Al llamar al BASIC desde el TRSDOS,se debe de seleccionar el máximo número de archivos que pueden estar abiertos simultáneamente de la siguiente manera :

```
BASIC -F:n
```

donde n es el número de archivos abiertos simultáneamente,máximo 15 .

Para hacer cualquier operación de entrada/salida en un archivo en disco se debe de realizar la operación de apertura del archivo primero, en ella se especifica el tipo de archivo que se tiene,el nombre que lleva en el disco y el buffer de la máquina al que se asigna . El buffer es un área en memoria principal,de tipo RAM,de 256 bytes.

```
OPEN modo,b,file,rl
```

donde :

```
modo="I"para entrada secuencial .
     "O"para salida secuencial .
     "D"o"R"para entrada/salida directa .
b    = número del buffer,de uno a quince .
file= nombre físico en el disco del archivo.
rl   = en acceso directo la longitud del registro,
       por omisión su valor es de 256 carácteres.
```

ej.

```
OPEN "I",2,"ARCHIVO/SEQ"
OPEN "D",1,"ARCHIVO/DIR",50
```

En el segundo ejemplo se ve que para acceso directo nosotros podemos especificar la longitud del registro,p.ej. 50 bytes .

ACCESO SECUENCIAL .

Es la forma más sencilla de guardar y recuperar información de un archivo . Cada que se abre el archivo para salida secuencial el contenido previo que haya tenido se pierde .

Los datos escritos secuencialmente usualmente incluyen delimitadores para indicar donde comienza y donde termina cada registro.

EJ.

Un archivo puede consistir de lineas de texto terminadas con caracteres <CR>,o bien de numeros separados por espacios, o bien de información alternada,numérica y alfanumérica.

El archivo es escrito en carácteres ASCII,un byte para cada carácter de datos. EJ.
12.34b lo almacena en 6 bytes,incluyendo el blanco b.

Los archivo secuenciales son escritos con una lonsitud de registro de uno. Esto es importante cuando uno cierra el archivo y lo abre para acceso directo,en este caso se debe de especificar una rl=1 (lonsitud de registro unitaria) .

ejemplo 1.

```
 5 ' EJEMPLO DE LA CREACION  DE UN ARCHIVO SECUENCIAL.
10 OPEN "O",1,"ESTAT/TXT"
20 INPUT "> NUMERO DE ALUMNOS";N%
30 FOR I%=1 TO N%
40       INPUT">DAME NOMBRE Y ESTATURA";ALUMNO$,EST
50       PRINT #1,ALUMNO$;",";EST
60 NEXT I%
70 CLOSE  1
80 END
```

En este ejemplo en la instrucción 10 se abre un archivo para acceso secuencial de salida,asignándolo al buffer uno y llamándolo dentro del disco como "ESTAT/TXT". Mediante la instrucción 50 escribimos el string alumno$ y el número est,delimitados por una ",".En la línea 70 cerramos el archivo asignado al buffer número 1,para poder volver a operar con el tendremos que volver a darle un OPEN . Si buscamos en el directorio del disco,después de ejecutar el anterior programa,veremos que "ESTAT/TXT" está presente,teniendo una lonsitud de registro unitaria .

ejemplo 2.

```
 5 ' EJEMPLO DE LA LECTURA DE UN ARCHIVO SECUENCIAL .
10 OPEN "I",1,"ESTAT/TXT"
15 CUENTA%=0
20 'DO WHILE NO HAYA FIN DE DATOS
30 IF NOT(EOF(1)) THEN 40 ELSE 70
40       INPUT#1,ALUMNO$,EST
50       PRINT "EL ALUMNO ";ALUMNO$;" MIDE ";EST
55       CUENTA%=CUENTA%+1
60       GOTO 30
70 'END DO WHILE
80 CLOSE 1
85 PRINT "SE LEYERON ";CUENTA%;" REGISTROS"
90 END
```

En este ejemplo el archivo que acabamos de crear anteriormente se abre para lectura secuencial. En la instrucción 30 empieza un ciclo iterativo do while que termina al ser verdadera la condición de EOF del archivo 1 .
La variable entera cuenta% se usa para contar el número de registros leídos .

La sintaxis de las instrucciones usadas es la siguiente:

```
PRINT#b, etc ---> escribe en el buffer b .
INPUT#b, etc ---> lee en el buffer b .
CLOSE#b      ---> cierra el archivo asignado al buffer b .
EOF(b)       ---> detector de fin de archivo (End-Of-File).

Otras instrucciones que se pueden usar son :
LINE INPUT#b,etc ---> similar al input,los strings se delimitan
                      por el <CR> o 256 bytes.
INPUT$(n,b)      ---> lee n carácteres del disco al buffer b.
```

Dado el problema de que al abrir un archivo secuencial para output el contenido previo se pierde,cuando queremos actualizar un archivo,modificar su contenido o bien agregarle registros se debe de seguir el siguiente algoritmo :

1 . Abrir el archivo para entrada secuencial .
2 . Abrir un nuevo archivo para salida secuencial .
    (este nuevo archivo tiene un nombre diferente y esta asignado a otro buffer de memoria).
3 . Dado el caso de que se quiera actualizar los registros, leer un bloque de datos del archivo original y actualizarlo.
4 . Escribir el bloque en el nuevo archivo.
5 . Repetir los pasos 3 y 4 hasta que todos los datos se hayan actualizado .
6 . Dado el caso de querer agregar nuevos registros,leer los registros y escribirlos en el segundo archivo .
7 . Cerrar ambos archivos .

E J e r c i c i o :

Hacer un programa que construya un archivo que contenga,en cada registro,nombre,dirección y teléfono de cada alumno;el programa debe también de emitir,de una manera presentable,un reporte de estos datos . Dado el caso de querer modificar algún registro,el programa deberá presentar una rutina que lea los registros y pregunte si se quiere modificar el registro y en caso afirmativo que campo o campos se desean modificar . También debe de permitir el introducir más registros en el archivo.

La práctica deberá de ser entregada mostrando el contenido del archivo antes de las modificaciones,después de éstas y después de haber agregado registros al archivo.

A R C H I V O S

P a r t e d o s .

ACCESO DIRECTO .

El acceso directo ofrece algunas ventajas con respecto al acceso secuencial. Por ejemplo,en vez de comenzar a operar en el archivo al comienzo podemos empezar en cualquier punto del archivo que especifiquemos ; para actualizar archivos o agregar registros no necesitamos seguir todo un proceso como el mostrado en la primera parte de la práctica ; es más eficiente el acceso directo,más rápido ; cuando se abre un archivo para acceso directo las operaciones de lectura y escritura se realizan usando el mismo buffer ; podemos estructurar mas claramente nuestros datos .

Esta estructuracion se lleva a cabo mediante la instruccion FIELD que organiza el buffer en campos . La sintaxis es la siguiente :

FIELD b,n1 AS c1,n2 AS c2, ...

donde : b = numero del buffer asignado al archivo
ni = número de bytes de que consiste el campo ci
ci = campo string que apunta hacia el área de memoria reservada para el buffer b .

Ej. FIELD 1, 30 AS NOMBRE$,40 AS DIRECCION$,7 AS TEL$
Estructura al buffer 1 en tres campos con una longitud total de 77 bytes . Cada uno de los campos es almacenado en el buffer,no en el área de strings.

La longitud maxima del registro es de 256 bytes .

La definicion de los campos puede ser "encimada",esto es,se puede estructurar el buffer de varias maneras diferentes .
Ej.
FIELD 1 , RFC$ AS 10 , ...
FIELD 1 , AP$ AS 2,AM$ AS 1,NOM$ AS 1,FECHANAC$ AS 6,...
Aquí el campo RFC$ se refiere a los diez carácteres de que consta el RFC y los campos AP$,AM$,NOM$,FECHANAC$ se refieren a las partes que lo constituyen .

Para tener acceso a los registros en el disco se usan las instrucciones GET y PUT ("lee y escribe"):

GET b,número de registro
Trae el registro especificado por el numero,por omisión de este trae el siguiente registro y lo deposita en el buffer b .
Ej. GET 1,25 GET 1

PUT b,número de registro
Mueve los datos del buffer b a el registro especificado,por omision el número de registro actual es usado.
Ej. PUT 1,25 PUT 1

Para introducir información dentro de los campos se usan las instrucciones LSET y RSET.

LSET campo=expresión string
Justifica la información de izquierda a derecha;si la expresión string no llena el campo son añadidos blancos en la derecha,si es demasiado larga ,los carácteres sobrantes a la derecha son ignorados .
EJ. LSET NOMBRE$="ENRIQUE" LSET PRECIO$=ARTICULO$(I)

RSET campo=expresión string
Justifica a la derecha la información;si la expresión no llena el campo pone blancos a la izquierda,si es demasiado larga los sobrantes a la derecha son ignorados .
EJ. RSET ARTICULO$="TORTILLA" RSET RFC$=AP$+AM$+NP$+FN$

Otras funciones de entrada/salida que se pueden usar en el manejo de archivos de acceso directo son :
MKD$ (número de doble precisión)
MKI$ (número entero)
MKS$ (número de simple precisión)
Estas funciones convierten información numérica a información de tipo string,según sea el caso del número,entero,de simple precisión o de doble precisión,el string que se retorna es de 2 , 4 u 8 bytes de longitud.
EJ.
```
10 OPEN "D",1,"EJEMPLO"
20 FIELD 1, 2 AS ENTERA$,4 AS SIMPLE$,8 AS DOBLE$
30 LSET ENTERA$ = MKI$(1000)
40 LSET SIMPLE$ = MKS$(1000.1)
50 LSET DOBLE$  = MKD$(1000.00001)
60 PUT 1
70 CLOSE 1
```

Las funciones inversas de estas tres últimas son :
CVD (cadena de carácteres)
CVI (cadena de carácteres)
CVS (cadena de carácteres)

Estas funciones restauran información de tipo string a tipo numérico de doble precisión,entera o de precisión sencilla . Según cada caso el string debe de tener longitudes de 8 , 2 y 4 bytes,si es menor ocurre un error .

EJ.
```
10 OPEN "D",1,"EJEMPLO"
20 FIELD 1,2 AS ENTERA$,4 AS SIMPLE$,8 AS DOBLE$
30 GET 1
40 PRINT "NUMEROS ";CVI(ENTERA$),CVS(SIMPLE$),CVD(DOBLE$)
50 CLOSE 1
```

Otras funciones son :
EOF (número de buffer) detecta el fin de archivo,regresa un cero cuando no lo halla y un -1 cuando lo encuentra . Su uso es igual que en los archivos secuenciales .

LOC (número de buffer) entresa el número del último resistro que ha sido procesado a partir de que se abrió el archivo, tambien es válido usarlo en archivos secuenciales .

```
EJ.
10 OPEN "D",1,"NOMBRE"
20 FIELD 1,15 AS NOM$
30 N$ = "YO MERO"
35 GET 1
40 IF N$ = NOM$ THEN PRINT "LOCALIZADO EN ";LOC(1) :CLOSE 1:END
50 GOTO 35
```

Este ejemplo manda un error de End of File si no localiza a N$ en el archivo,por la razón de que intenta leer cuando el archivo ya ha sido leído en su totalidad .

LOF (número de buffer) Esta función nos da el número del último registro del archivo,válida para archivos secuenciales también. Ver su uso en el programa ejemplo 1 .

ejemplo 1.

```
 1  CLS
 5  PRINT "> EJEMPLO DE ENTRADA Y SALIDA EN UN ARCHIVO DIRECTO"
10  OPEN"D",1,"GENTE/TXT",70
20  FIELD 1,30 AS NOM$,40 AS DIR$
30  I% = 1
40  INPUT">QUIERES INTRODUCIR UN REGISTRO ? [ S / N ] : ";RESP$
50  'DO WHILE HAYA DATOS
60  IF RESP$="S" THEN 70 ELSE 150
70     INPUT">DA NOMBRE :";N$
80     INPUT">DA DIRECCION:";D$
90     LSET NOM$=N$
100    RSET DIR$=D$
110    PUT  1,I%
120    I% = I% + 1
130    INPUT">QUIERES INTRODUCIR OTRO REGISTRO ? [ S / N ] : ";RESP$
140    GOTO 60
150 'END WHILE
160 PRINT "> SE ESCRIBIERON ";I%-1;" REGISTROS EN EL ARCHIVO"
180 INPUT">QUIERES LEER UN REGISTRO ? [ S / N ] : ";RESP$
190 'DO WHILE QUIERA LEER UN REGISTRO
200 IF RESP$="S" THEN 210 ELSE 390
210    MAL$="S"
220    'DO WHILE ESTE MAL
230    IF MAL$="S" THEN 240 ELSE 320
240             INPUT"> DA número DE REGISTRO A LEER ";R%
250             IF( R% <= LOF(1) ) AND (R% > 0)THEN 260 ELSE 280
260                     MAL$="N"
270                     GOTO 300
280             'ELSE
290                     PRINT "> NUMERO  DE REGISTRO EQUIVOCADO "
300             'ENDIF
310             GOTO 230
320    'ENDWHILE
330    GET 1,R%
340    PRINT "> REGISTRO  NUMERO : ";R%
350    PRINT "> NOMBRE           : ";NOM$
360    PRINT "> DIRECCION        : ";DIR$
370    INPUT">QUIERES LEER OTRO REGISTRO ? [ S / N ] :";RESP$
380    GOTO 200
390 'ENDWHILE
400 CLOSE 1
410 END ' PROGRAMA EJEMPLO
```

En este ejemplo el archivo se abre para acceso directo en la línea 10;se define el buffer en dos campos en la 20;a partir de la 50 hasta la 150 se halla un do while de escritura al archivo en disco;en las líneas 90 y 100 se asigna información a los campos del buffer y en la 110 se escribe el buffer;el do while de la 190 a la 390 lee el registro que se le indique;se debe de notar la manera de que se justifico la información en los campos,a la izquierda en el nombre y a la derecha en la dirección;en la línea 400 se cierra el archivo .

E j e r c i c i o :

Primera parte de "Nómina Semanal" .

Hacer un programa que tenga el siguiente menú de procedimientos :

a) Cargar en un archivo directo una tabla que contenga diez categorías de empleado y sus respectivos sueldos/hora .

b) Cargar o agregar en un archivo directo "Maestro" registros de la forma :

[clave del empleado][nombre][rfc][fech.ins][categoría][infonavit]

[fondo ahorro][cuota sindical][domicilio][teléfono]

donde:

[clave del empleado]	es numérica .
[rfc]	es de la forma PPMNAAMMDD .
[fecha ingreso]	es de la forma AAMMDD .
[categoría]	se refiere a una de las 10.
[infonavit]	monto acumulado del ídem .
[fondo ahorro]	monto acumulado de ahorro ; si es -1,el empleado optó por no tenerlo .
[cuota sindical]	vale "S"í o "N"o .

c) Modificación de los registros del archivo maestro,presuntando la clave del empleado cuyo registro se quiere modificar,se despliega este registro y se presuntan los cambios que se desean realizar .

d) Impresión de un reporte conteniendo toda la información contenida en el archivo maestro .

N o t a :

La introducción del RFC debe realizarse usando una subrutina que presunte la fecha de nacimiento y saque el RFC.
Esta rutina ya se vio en el curso .
Se debe de validar la fecha que se introduzca (p.ej. no se aceptan fechas del tipo 31 de febrero de 1965) .

O R D E N A M I E N T O S

Introduccion .

El ordenar es una de las operaciones más importantes que se practican sobre una estructura de datos y consiste en establecer un orden de precedencia entre los distintos elementos de la estructura,estableciendo esta precedencia de acuerdo a uno o mas campos seleccionados . A estos campos se les llama llaves .

Existe una gran cantidad de algoritmos para realizar esta operación. En esta práctica se verán únicamente dos algoritmos

Para seleccionar el algoritmo conveniente a nuestro caso se debe de tener en cuenta el tipo de memoria en la que se encuentran los datos(directa de alta,media velocidad,secuencial),la forma en que el sistema operativo maneja los archivos y la memoria,los tiempos de acceso de los dispositivos,la cantidad,tipo y distribución inicial de los datos,la eficiencia del método de ordenamiento(basada en el número de comparaciones e intercambios y el monto de memoria que requiere).

Algoritmo de la Burbuja .

El algoritmo de la burbuja es un método de ordenamiento de los llamados por intercambio,debido a que trasponen o intercambian sistemáticamente pares de datos que se encuentran fuera de orden hasta que dejen de existir los pares fuera de orden. Se le llama burbuja debido a la manera en que las llaves de menor "peso" van subiendo a la "superficie".

El algoritmo empieza comparando las llaves n y n-1 y las intercambia si n<n-1;despues compara las llaves n-1 y n-2,intercambiándolas si n-1 < n-2 . El proceso sigue hasta comparar las llaves 1 y 2 . Una vez hecho esto,la llave de menor valor ha alcanzado la "superficie".

Ahora se repite el procedimiento sobre las llaves situados en el rango de 2 y n.

Para el proceso se requieren,como máximo,n-1 pasadas. Cuando en una pasada no se realiza ningún intercambio el conjunto ya ha sido ordenado.

Algoritmo :

Considérese a las llaves almacenadas en el vector " a " y el total de datos igual a " n " .

```
INICIO ALGORITMO BURBUJA
 limite-inferior = 1
 hubo-intercambio= verdadero
 do while hubo-intercambio
        limite-inferior = limite-inferior + 1
        hubo-intercambio= falso
        for i = n hasta limite-inferior step -1
                if a(i) < a(i-1) then
                        hubo-intercambio = verdadero
                        auxiliar = a(i)
                        a(i) = a(i-1)
                        a(i-1) = aux
                endif
        endfor
 endwhile
FIN ALGORITMO BURBUJA
```

Ejemplo:

Considérese el conjunto de llaves (10,12,8,1,6,19,4)

Datos.	Pasadas: 1	2	3	4	5
10	1	1	1	1	1
12	10	4	4	4	4
8	12	10	6	6	6
1	8	12	10	8	8
6	4	8	12	10	10
19	6	6	8	12	12
4	19	19	19	19	19

El algoritmo de la burbuja es apropiado en datos semiordenados,el número de intercambios que se realizan depende de los datos,pero en promedio es mayor que n-1 y cercano a n^2/4. El espacio de memoria adicional usado es únicamente el de la variable auxiliar.

Algoritmo Shell .

El algoritmo Shell es un algoritmo de los llamados por inserción;estos algoritmos suponen al conjunto de datos ordenado,después,para una llave k que se deseé agregar al conjunto,se determina el lugar que debe de ocupar y se mueven los datos una posición para insertarlo en su posición correcta . En particular el algoritmo Shell clasifica un conjunto de llaves ordenando sucesivamente subconjuntos de llaves del conjunto a ordenar.

Los subconjuntos se definen tomando las llaves del conjunto mediante una secuencia de incrementos almacenados en un vector INC de dimension T . El elemento final INC(T) debe de ser siempre unitario .

Algoritmo:

Se considera a las llaves en el arreglo " a " y a los incrementos en"inc",siendo n el total de llaves y T el total de incrementos .

```
INICIO SHELL.
for cuenta = 1 to T
        delta = inc ( cuenta )
        for j = (delta + 1) to n
                i = j - delta
                k = a ( j )
                salida = falso
                do
                        if k < a( i ) then
                                a( i + delta ) = a ( i )
                                i = i - delta
                        else
                                salida = verdadero
                        endif
                until i <= 0 or salida = verdadero
                a ( i + delta ) = k
        endfor
endfor
FIN SHELL
```

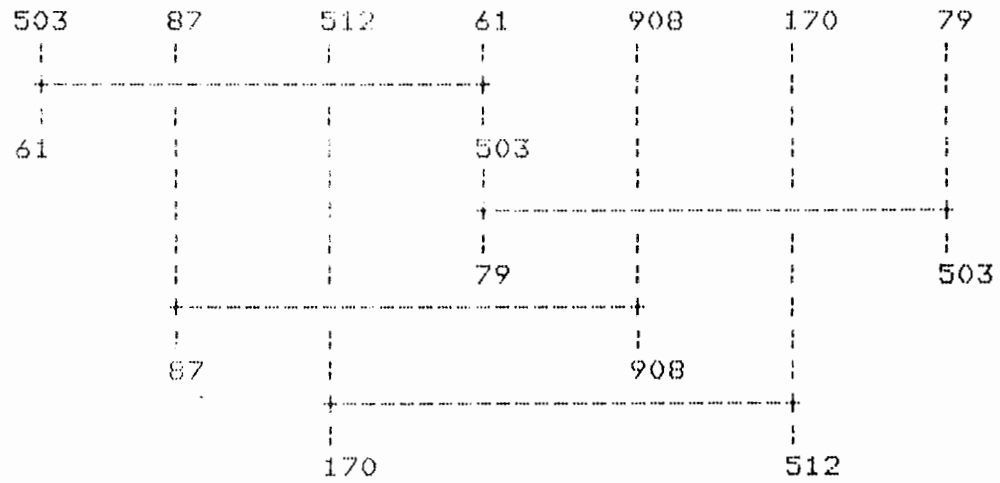
Ejemplo . Considerese el siguiente conjunto de llaves :

503,087,512,061,908,170,79

siendo el vector de incrementos :

inc(1)=3,inc(2)=2,inc(1)

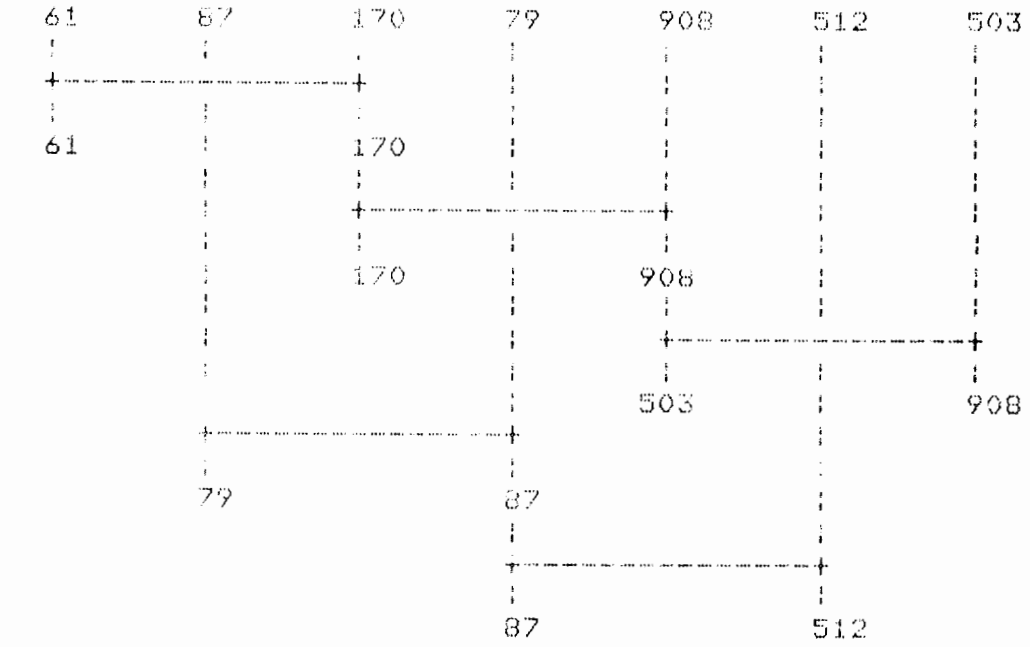
Entonces la primera fase del ordenamiento es (inc(1)=3):



Al finalizar la primera nuestros datos estan así :

61 87 170 79 908 512 503

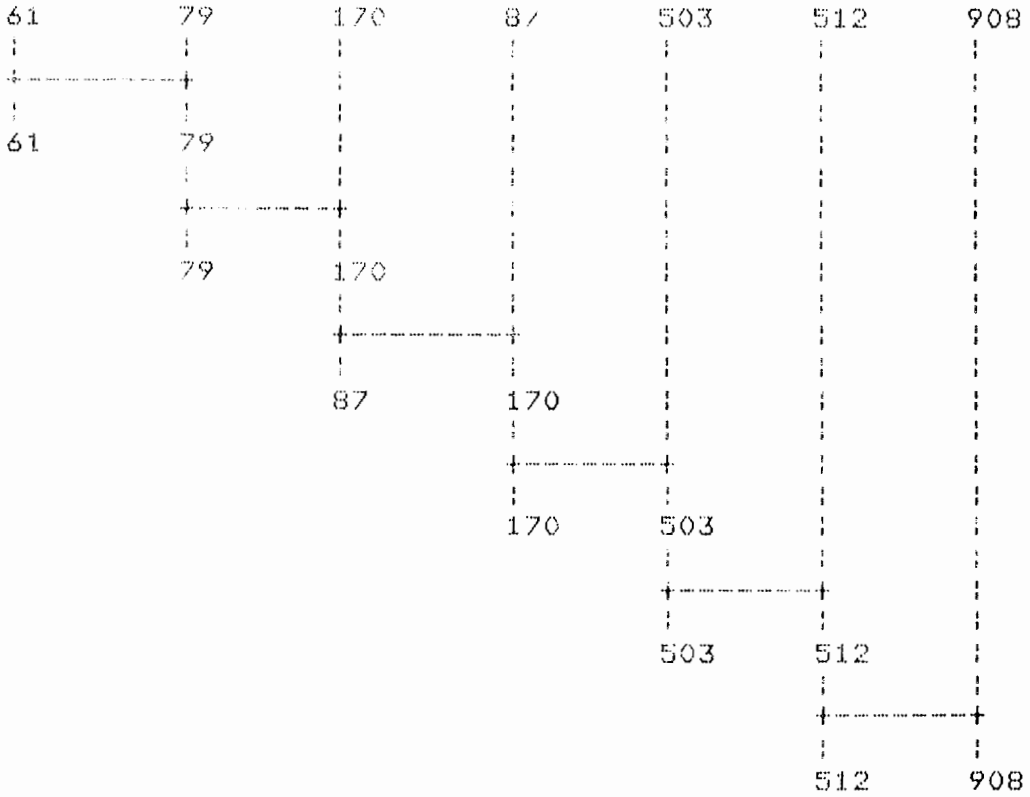
La segunda es (inc(2)=2):



Al finalizar la segunda los datos están así:

61 79 170 87 503 512 908

La tercera es (inc(1)=1):



Al finalizar la tercera los datos ya están ordenados :

61 79 87 170 503 512 908

La generación de los incrementos es muy importante,debido a que el número de comparaciones del algoritmo Shell esta en función de los incrementos que se usen.

Algunas maneras de generar estos valores son :

- Si el número de incrementos es de dos , inc(1) = 1.72 (n) ^ (1/3) e inc(2) = 1 , con estos incrementos el número de comparaciones es proporcional a n^(5/3)

- Para más incrementos la secuencia inc(k)= 2 ^ k - 1 para 1<=k<=log n ,con estos incrementos el número de comparaciones es proporcional a n^(3/2)

- Otra forma es (n = total de datos a ordenar):

```
INICIO
total = n
T = 0
do
        aux = total div 2
        T   = T + 1
        inc(T) = aux
        total = aux
until inc(T) = 1
FIN
```

E J e r c i c i o :

Al menú del programa realizado para la práctica de archivos de acceso directo agregarle una nueva rutina que pregunte por que llave se quiere ordenar el archivo maestro,pudiendo ser llave lo siguiente : clave del empleado,nombre,fecha de ingreso,categoria,infonavit,fondo de ahorro y fecha de nacimiento,obteniendo esta última a partir del RFC. La rutina debe de preguntar tambien en qué orden se desea el ordenamiento:ascendente o descendente.

La rutina debe de ser hecha usando el algoritmo SHELL.

La práctica debe de presentarse acompañada de un listado mostrando al archivo maestro en su forma original y después del ordenamiento. Los ordenamientos mostrados deberán de ser al menos tres,de llaves distintas .

B U S Q U E D A S

Introducción .

En los sistemas de procesamiento de información las operaciones más frecuentes sobre las estructuras de datos son las de agregar,borrar,modificar y consultar elementos .

Todas estas operaciones tienen una acción común: la de buscar el elemento sobre el que se desea realizar el proceso.

La operación de búsqueda se puede realizar de varias maneras,las cuales se agrupan en dos categorías de búsquedas,por comparación de llaves y por transformación de llaves .

Los algoritmos de búsqueda que veremos caen en la primera de estas categorías,la de búsquedas por comparación de llaves,llamada así por que se realiza una comparación directa de la llave buscada con las llaves de los elementos de la estructura de datos .

L I N E A L .

La búsqueda lineal es el método más sencillo y consiste en recorrer secuencialmente todos los elementos de la estructura.

Este método es aplicable en estructuras que estén ordenadas o desordenadas. Es de gran utilidad si la estructura es pequeña .

La eficiencia del algoritmo puede medirse en base a las comparaciones necesarias para hallar el elemento buscado. Si son n elementos,el promedio será de n/2 comparaciones y en el peor caso se requerirán n comparaciones .

Algoritmo .

Sean E[1],E[2],...,E[n] los elementos de la estructura y K[ ],K[2],...,K[n] sus respectivas llaves . El elemento buscado se representa por X .

```
INICIO .
For i = 1 hasta n
        if K[i] = x then
                Escribe("BUSQUEDA EXITOSA")
                salida
        endif
endfor
Escribe ("búsqueda sin exito,elemento no hallado")
FIN .
```

Una sencilla modificación a este algoritmo lo hace más eficiente y consiste en colocar al elemento X en la posición K[n+1] como elemento "centinela".

```
INICIO .
i = 1
K[n+1] = x
Do while K[i] <> x
        i = i + 1
enado
if i=n+1 then
        Escribe("búsqueda sin éxito")
else
        Escribe("BUSQUEDA EXITOSA")
endif
FIN .
```

El colocar a X en la posición n+1 indica que siempre se va a encontrar a el elemento en la estructura y,por lo tanto,solo resta observar en que posición se encuentra¿si donde lo pusimos,que indica una búsqueda no exitosa o bien en otra posición,que nos indica una búsqueda exitosa.

B I N A R I A .

La búsqueda binaria o por bisección se basa en la repetida bipartición del intervalo de búsqueda dentro de la estructura de datos.

Este método es solo aplicable en estructuras ordenadas.

El algoritmo consiste en hallar el elemento central de la tabla,comparar este elemento con el buscado,si son iguales la búsqueda termina ¿si el elemento buscado es mayor (menor) que el elemento central,se desecha la primera (segunda) mitad de la estructura y se considera esta nueva mitad como la nueva estructura de búsqueda.

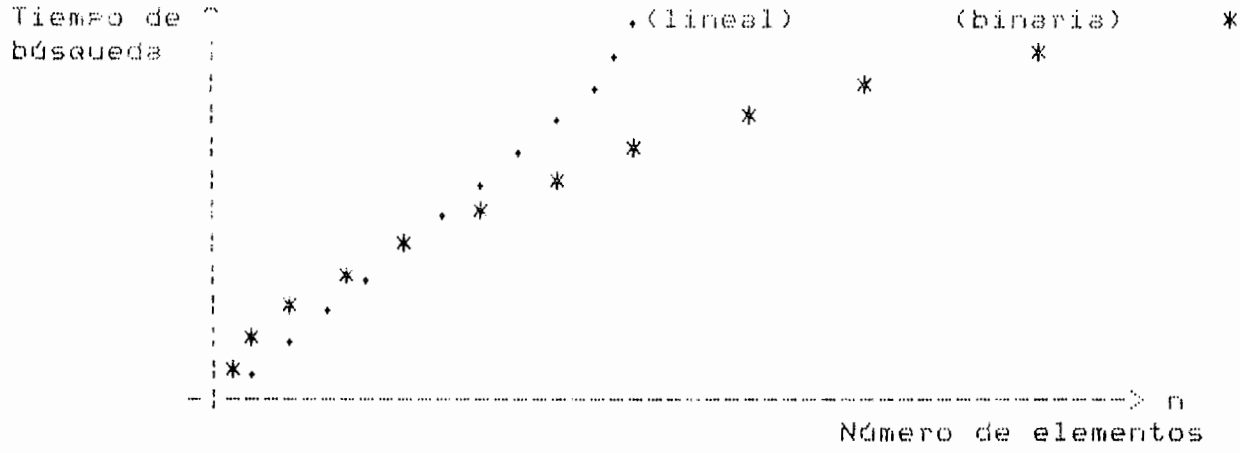
Algoritmo .

Sean E[i] los elementos de la estructura,K[i] sus llaves y X el elemento buscado,donde i [1...n] . La operación TRUNC se refiere a truncar el número real para entregar un número entero .

```
INICIO .
inf = 1
sup = n
Do while inf <= sup
        medio = trunc ( (inf+sup) / 2 )
        if x < K [ medio ] then
                sup = medio - 1
        else
                if x > K[medio] then
                        inf = medio + 1
                else
                        Escribe ( "BUSQUEDA EXITOSA" )
                        Salida
                endif
        endif
enddo
Escribe ( "búsqueda sin exito" )
FIN .
```

La eficiencia de este método,comparada con el de la búsqueda lineal mejora entre mayor es n. El número de comparaciones en promedio es de TRUNC(log2(n))-1 y en el peor caso de TRUNC(log2(n))+1 .

Una gráfica comparativa del comportamiento de ambos métodos es la siguiente :



E j e r c i c i o :

Al menú del programa de nómina agregar una rutina de búsqueda binaria que presunte la llave que se desea localizar en el archivo . Las llaves pueden ser : la clave del empleado,nombre,fecha de ingreso,categoría . La práctica debe de presentarse acompañada de un listado mostrando al archivo maestro original y de al menos tres búsquedas de llaves distintas .

A P E N D I C E S .

Como hacer uso de la Radio Shack .

En este apéndice se verán los pasos básicos para hacer uso de la microcomputadora Radio Shack nivel II .

El diskette que usaremos será de 8 pulgadas,soft sector.

El primer paso es meter el disco en el drive , localizado al lado derecho de la pantalla . Para cargar el sistema , el disco que introduzcamos deberá de tenerlo previamente .

Entonces la computadora automáticamente carga el sistema operativo del disco a la memoria de la computadora . Esta acción se realiza también cuando se manipula la palanca de RESET .

Una vez cargado,la computadora pide la fecha en el formato MM/DD/YYYY (mes/día/año) y la hora HH:MM:SS .

Una vez realizado esto aparece en la pantalla :

TRSDOS READY
.....

Si tu diskette es virgen,primero se le tiene que formatear,para esto se usa el programa de utilería FORMAT que borra e inicia el diskette . La sintaxis de la instrucción es :

FORMAT drive (opciones)

En nuestro caso , el drive es cero,se pone 0 .

Algunas opciones son :

ID=nombre-diskette - Asigna el nombre al diskette , por omisión es TRSDOS .

PW=password . Asigna una contraseña al diskette , por omisión toma PASSWORD.

Una vez formateado se deberán copiar los programas del sistema al diskette,para lo que se usa el programa de utilería BACKUP , que duplica , copia un diskette .

La sintaxis es la siguiente :

BACKUP fuente TO destino (opciones)

Fuente y destino son los números de los drives involucrados en el proceso , si se omiten,TRSDOS los presunta después .

Algunas opciones son :

ID=nombre del diskette. Asigna nombre al diskette destino . Si se omite,TRSDOS asigna el que tiene el diskette fuente .

PW=password del fuente . Se da el password del diskette fuente,si no es correcto la duplicación no es realizada . Si se omite,TRSDOS asume que es PASSWORD .

SYS . Indica al TRSDOS que sólo copie los archivos del sistema . Si es omitido todos los archivos serán copiados (o sujetos a selección por PROMPT) .

PROMPT . Dice al TRSDOS que mande un prompt al usuario antes de que un archivo que no sea del sistema vaya a ser copiado,dando de esta manera la capacidad de selección de copiado .

Sistema operativo TRSDOS .

El sistema operativo es un programa que se encarga de administrar los recursos de la máquina,como son el procesador,en el caso de la Radio Shack un microprocesador Z-80 , la memoria , los periféricos como la unidad del diskette,el teclado y la pantalla , etc .

Algunos de los principales comandos del sistema operativo TRSDOS son :

AGAIN .
Indica al TRSDOS que vuelva a ejecutar el último comando introducido .

CLOCK .
Despliega el reloj en la pantalla (CLOCK ON) o lo quita (CLOCK OFF).

CLS .
Borra la pantalla .

DIR .
Despliega los archivos contiene nuestro diskette .

KILL archivo .
Borra un archivo de nuestro diskette .

LIST archivo .
Lista el contenido de un archivo .

RENAME viejo nombre TO nuevo nombre .
Cambia el nombre a un archivo .

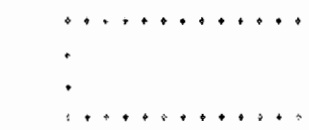
G U I A S D E P R O G R A M A C I O N .

P r o g r a m a c i ó n E s t r u c t u r a d a .

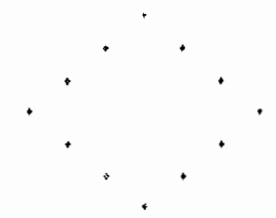
La programación estructurada surge con la necesidad de reducir la complejidad de los grandes programas,que con frecuencia presentan los problemas de no resolver las necesidades del usuario,su desarrollo no es predecible,contienen errores,cuestan mas de lo estimado y su mantenimiento es difícil dada su poca flexibilidad.
La programación estructurada intenta dar la mejor solucion a estos problemas.

Una de las varias herramientas que usa la programación estructurada son los diagramas de flujo estructurados .
En estos diagramas podemos distinguir las siguientes figuras básicas :

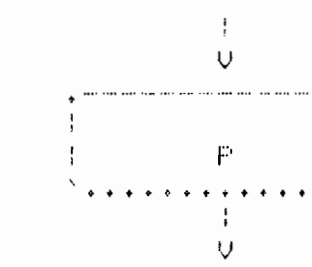
Esta figura indica el inicio o fin de un proceso .



Esta figura indica un proceso .



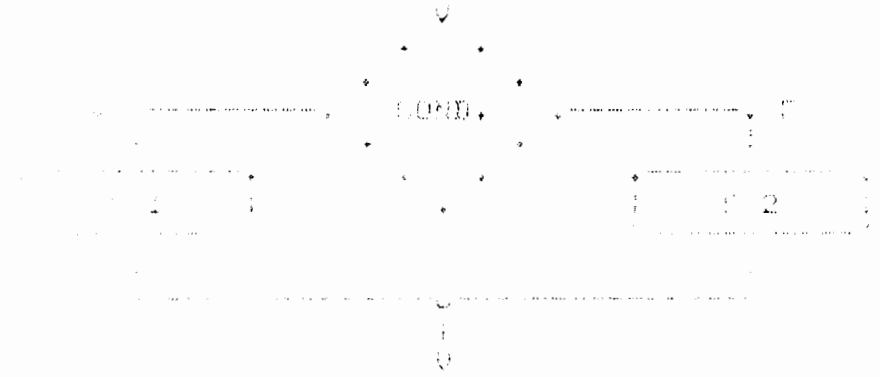
Esta figura indica una decisión .
Las figuras estructuradas con sus respectivas implementaciones en Basic son las siguientes :



Esta es una secuencia,tiene una entrada,una salida y realiza el proceso P .

```
Basic :
100 P r o c e s o
110   . . . . .
```

2.2 IF-THEN-ELSE.



El diagrama representa un IF-THEN-ELSE cuando la condición COND es verdadera se realiza el proceso P1 si es falsa se realiza el proceso P2 . El terminar ambos procesos vuelven a unirse con lo que la figura tiene una entrada y una salida y puede considerarse como la figura principal de la SECUENCIA .

Ejemplo :

```
100 IF "COND" THEN 110 ELSE 200
110             Se pone aquí el proceso P1
120             GOTO 300
200 ELSE
210             Se pone aquí el proceso P2
300 ENDIF
```

La implementación para la figura modificada de [Pseudo] es casi que solo se hace un proceso en el caso verdadero (THEN) mientras en el caso falso el flujo sigue normalmente y en [illegible]

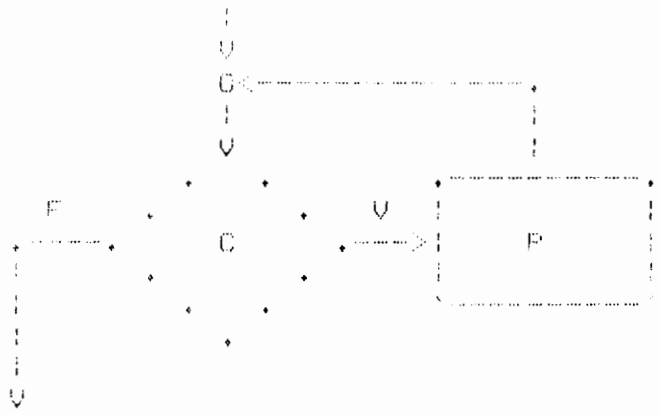
```
100 IF "Condición inicial negada" THEN 120
110             Proceso a realizarse .
120 ENDIF
```

La "condición inicial negada" se refiere por ejemplo a que si la condición inicial era " A>B " ahora será

Figuras iterativas.

Las siguientes dos figuras son iterativas, esto es, dependiendo de una condición realizará un proceso varias veces y cuando la condición sufra un cambio pasará el control del programa a el siguiente proceso.

Do - While.



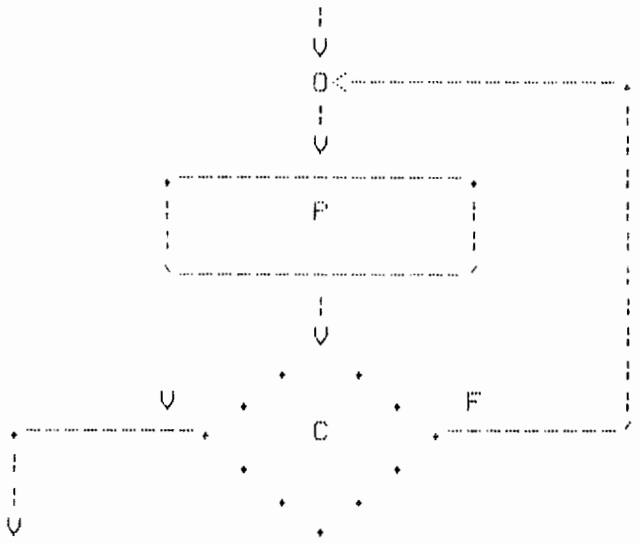
Esta iteración es un DO-WHILE, "haz mientras". Mientras la condición 'C' sea verdadera realizará el proceso "r", cuando la condición sea falsa saldrá de la iteración. Casos muy usados de esta figura serían "haz mientras haya datos que leer", "haz mientras no haya error", etc. El "for" de Basic es un caso del "do-while".

Esta figura tiene una entrada y una salida, por lo que puede ser encerrado en un bloque y considerarse un proceso.

Basic :

```
100 IF "C" THEN 200 ELSE 300
200             Se pone el proceso "P" aquí
...                 . . .
...             GOTO 100
300 'ENDWHILE
```

D o - U n t i l .



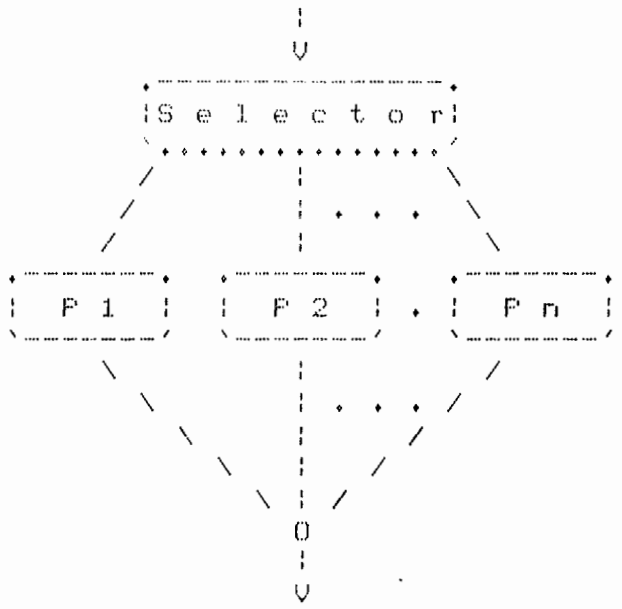
Esta figura es un DO-UNTIL,"Haz-hasta". Se realiza el proceso "P" hasta que la condición "C" se torna verdadera . Al menos una vez el proceso "P" se realiza. La figura tiene una entrada y una salida . El "do" de Fortran es un caso del "do-until".

Basic :

```
100 'DO
...    Aquí se pone el proceso "P"
...
... IF "C" THEN 200 ELSE 100
200 'ENDDO
```

C a s e .

La siguiente figura es de "decisión" :



Se trata de un CASE,"caso". Dependiendo del valor que tome la variable "selector" el flujo del programa enfilará a una de los "n" procesos "Pi". Es factible que algun valor del selector no tenga un proceso,por lo que es recomendable tener un "else". En basic la instrucción "ON ... GOTO" corresponde a esta figura.

Basic :

```
010 ON "selector" GOTO 100,200,...
020 'ELSE
...       Proceso que se realiza en caso
...       de que no haya proceso para el
...       valor asumido por el selector.
...       GOTO 1000
100       Proceso P1
...       GOTO 1000
200       Proceso P2
...       GOTO 1000
...       . . .
1000 'ENDCASE
```

R e c o m e n d a c i o n e s :

1. Programa claramente,no sacrifiques la claridad por la "eficiencia" .

2. Sustituye las expresiones repetidas por llamadas a funciones o subrutinas .

3. Modulariza tu programa en subrutinas .

4. Da nombres claros y explícitos a las variables .

5. Usa los GOTOs solo para implementar las estructuras fundamentales.

6. Da una adecuada sangría a cada nivel lógico del programa , formatéalo.

7. Es mejor rehacer una mala codificación que tratar de remendarla.

8. Escribe y prueba un programa grande en pedazos pequeños .

9. Verifica los valores de entrada de tu programa,usa procesos de validación de entrada .

10. Inicia todas las variables antes de usarlas.

11. Verifica manualmente algunas respuestas .

12. Haz tu programa claro antes de hacerlo "rápido".

13. Comenta tu programa,señala que representan las principales variables,explica los algoritmos usados,qué entradas necesita tu programa y qué resultados entresa.

14. Haz uso de los comentarios "endif","endfor","enddo","endwhile",etc. Asegúrate que los comentarios concuerden con el programa .

15. Documenta tu programa .

E D I T O R .

El Basic nos provee de un editor,que nos sirve para modificar líneas sin tenerlas que teclear de nueva cuenta . El editor tiene efecto únicamente sobre la línea que se le especifique y,por tanto,se dice que es un editor de línea .

Para tener acceso al editor existen dos formas,pulsar la tecla <F1> antes del <ENTER> si se está tecleando la línea ; se activa entonces el editor y podemos hacer uso de sus distintos comandos .

La otra manera es tecleando :

* EDIT número de línea <ENTER>

Si en vez de poner un "número de línea" ponemos un punto,nos estaremos refiriendo a la última línea que fue introducida,modificada o bien la línea donde ocurrió el último error del programa.

Por ejemplo,vamos a teclear la siguiente línea :

100 FOR I=1 TO 5 STEP 0.5: PRINT I:NEXT J

Si tecleamos :

EDIT . <--- O bien : EDIT 100

La máquina responde :

100_

Que significa que estamos en modo editor.

Los comandos del editor son los siguientes :

* <ENTER>

Al pulsar esta tecla en el modo de edición,la máquina guarda todas las modificaciones que se le hayan hecho a la línea y vuelve al modo de comandos de basic .

* n <BARRA ESPACIADORA>

Recorre "n" espacios a la derecha sobre la línea que estamos editando ,por omisión de "n" es uno . En nuestro ejemplo,si damos :

3 <BARRA ESPACIADORA>

La máquina responderá :

100 FOR_

* n<BACKSPACE>

El cursor se recorre a la izquierda "n" carácteres . Por omisión de "n" es uno .Con 3<BACKSPACE> nuestro ejemplo quedaría :

100_

* <ESC>

Presionando esta tecla nos salimos del modo de inserción al que se llega mediante los comandos I,H,X . Quedamos en el modo de edición de nuevo,si tecleamos <ENTER> también podemos salir del modo de inserción,pero llegamos al modo de comandos de Basic .

* L (lista la línea editada)

Al dar L se despliega la línea que estemos editando,en nuestro ejemplo,al dar L queda :

100 FOR I=1 TO 5 STEP 0.5: PRINT I:NEXT J

100_

De esta forma podemos ver la línea que estemos editando.

* X (extiende la línea editada)

Al dar X entramos al modo de inserción justo al final de la línea;en nuestro ejemplo,si queremos agregar ":PRINT "HECHO" ,tecleamos X y:

100 FOR I=1 TO 5 STEP 0.5: PRINT I:NEXT J : PRINT "HECHO" <ESC>

Se usa el <ESC> para salir del modo de inserción X .

* I (inserta)

Al dar I podemos insertar carácteres en donde se halle el cursor;en nuestro ejemplo queremos que el incremento del FOR sea de 0.25 ,entonces ponemos el cursor en el 5 y tecleamos la I ,a continuación tecleamos el 2 y <ESC> para salirnos y queda entonces :

100 FOR I=1 TO 5 STEP 0.2<ESC>

Presionando L nos queda :

100 FOR I=1 TO 5 STEP 0.25: PRINT I:NEXT J : PRINT "HECHO"

100_

* A (cancela cambios y reempieza)

Al dar A,cancelamos todos los cambios que hayamos realizado y la máquina sitúa el cursor al principio de la línea .

* E (salva cambios y sale)

Al dar E ,salvamos todos los cambios que hayamos hecho y termina la edición de la línea,saliendo al modo de comandos de basic .

* Q (cancela cambios y sale)

Al dar Q,cancelamos todos los cambios que hayamos hecho y termina la edición de la línea,saliendo al modo de comandos de basic .

* H (borra e inserta)

Al dar H,le decimos a la máquina que borre los carácteres remanentes a partir de donde se sitúa el cursor y que los sustituya por los que le vamos a dar . En nuestro ejemplo,suponsamos que deseamos borrar el PRINT "HECHO" y sustituirlo por un END . Situamos nuestro cursor en la P y damos H,a continuación tecleamos el END,para salirnos usamos <ESC> . Nos queda :

100 FOR I=1 TO 5 STEP 0.25: PRINT I:NEXT J : END<ESC>

* nD (borra)

Este comando le indica a la computadora que borre "n" carácteres a partir de donde esta situado el cursor , por omisión de "n" es uno . En nuestro ejemplo suponsamos que deseamos borrar el "0.2" para que el incremento de nuestro FOR sea de 5 . Situamos entonces el cursor en el 0 y tecleamos 3D ,quedara :

100 FOR I=1 TO 5 STEP \0.2\

Tecleando una L nos desplegará :

100 FOR I=1 TO 5 STEP 5: PRINT I:NEXT J : END

* nC (cambia)

Este comando le indica a la computadora que deseamos cambiar "n" carácteres a partir de donde esta situado el cursor . Por omisión de "n" es uno . En nuestro ejemplo,suponsamos que nos dimos cuenta que es "NEXT I" y no "NEXT J" la instrucción correcta,deseamos cambiar la J por I . Entonces nos situamos en J y oprimimos C,lo que le indica a la computadora que deseamos hacer un cambio,queda:

100 FOR I=1 TO 5 STEP 5: PRINT I:NEXT I

Presionando L :

100 FOR I=1 TO 5 STEP 5: PRINT I:NEXT I : END

* nSc (busca)

Le indica a la computadora que busque la enésima ocurrencia del carácter "c" y mueva el cursor a esa posición.
Si no se indica "n" se busca la primera ocurrencia. En nuestro ejemplo suponsamos que el incremento FOR se va a modificar,entonces para situar el cursor en ese punto podemos dar el comando : 2S5 y queda :

100 FOR I=1 TO 5 STEP _

El cursor,de esta forma,ya esta situado donde deseabamos y ya podemos realizar las modificaciones que queramos .

* nKc (busca y borra)

Este comando borra todos los caractres hasta la enésima ocurrencia del carácter "c". En nuestro ejemplo suponsamos que sólo queremos dejar el END, entonces tecleamos 3K: y queda :

100\FOR I=1 TO 5 STEP 5: PRINT I:NEXT I :\ END

Si damos L :

100 END

Mensajes de error.

Código	Abreviatura	Explicación
1	NF	NEXT SIN FOR . Este error puede ocurrir si las variables del next están invertidas en un anidamiento de iteraciones FOR .
2	SN	SINTAXIS . Ocurre generalmente por puntuación equivocada,carácteres ilegales o instrucciones mal escritas .
3	RG	RETURN SIN GOSUB . Se encontró un return sin haberse ejecutado antes un gosub .
4	OD	INSUFICIENCIA DE DATOS . Un read fue ejecutado con datos no disponibles . La DATA está incompleta o ya fue leída en su totalidad .
5	FC	LLAMADO ILEGAL DE FUNCION . Se hizo el llamado a una función con parámetros ilegales,por ejemplo la raíz cuadrada de un número negativo .
6	OV	FUERA DE RANGO . La magnitud del número es muy grande para el tipo con que se declaró .
7	OM	FUERA DE MEMORIA . Toda la memoria disponible fue usada o reservada. Puede ocurrir con arreglos muy grandes o transferencias anidades como GOSUB e iteraciones FOR / NEXT .
8	UL	LINEA INDEFINIDA . Se está haciendo referencia a una línea inexistente .
9	BS	INDICE INVALIDO . El índice de un arreglo está fuera del rango dimensional declarado .
10	DD	DIMENSIONAMIENTO REPETIDO . Se trató de dimensionar un arreglo ya definido. Se debe de usar antes ERASE.
11	/0	DIVISION POR CERO . Se usó un denominador nulo .
12	ID	DIRECTA ILEGAL . Se trató de usar una instrucción de sólo programa,como INPUT,en una línea inmediata (no de programa).
13	TM	CONFLICTO DE TIPOS . Se trató de asignar un número a una variable cadena o a la inversa .
14	OS	FUERA DE ESPACIO DE CADENAS . La acumulación usada por las cadenas supera la memoria asignada. Usar CLEAR para aumentar el espacio .
15	LS	CADENA MUY LARGA . Una cadena ha excedido el máximo de 255 carácteres .

Mensajes de error.

Código	Abreviatura	Explicación
16	ST	CADENA MUY COMPLEJA . Una operación de tipo cadena es demasiado compleja,debe de frasmentarse en etapas.
17	CN	NO SE PUEDE CONTINUAR . Un comando CONT fue dado en un punto donde no puede tener efecto .
18	UF	FUNCION DE USUARIO INDEFINIDA . Se trató de llamar una función USR sin haber definido primero su punto de entrada vía la instrucción DEFUSR .
19	NR	NO REANUDACION . Durante una rutina de atención de errores,BASIC encontró el fin del programa sin haber detectado un RESUME .
20	RW	REANUDACION SIN ERROR . Un RESUME se encontró sin que hasa habido un error . Se necesita poner un END o un GOTO antes de la rutina de atención a errores.
21	UE	ERROR INDEFINIDO . Esta reservado para definir errores futuros (pesimistas!) .
22	MO	OPERANDO EXTRAVIADO . Se esta tratando de hacer una operacion a la que le falta un operando .
23	BO	BUFFER SOBREPASADO . Se trata de dar entrada a una linea de datos con demasiados carácteres para ser tomados por el buffer de línea .
24-49	UE	ERROR INDEFINIDO . Igual al 21 .
50	FO	CAMPO SOBREPASADO . Se está tratando de poner en un campo FIELD más carácteres de los que la longitud de registro del archivo de acceso directo proveé. Esta longitud se asigna en el OPEN . El default es de 256 carácteres .
51	IE	ERROR INTERNO . También indica que se trató de usar un EOF en un archivo no abierto .
52	BN	NUMERO DE ARCHIVO ERRONEO . Se trató de hacer una operación con un número que especifica un archivo no abierto o mayor al número de archivos que se especificaron al llamar al basic .
53	FF	ARCHIVO NO ENCONTRADO . Una referencia con LOAD, KILL u OPEN se refiere a un archivo que no existe.
54	BM	MODO DE ARCHIVO ERRONEO . Se trató de hacer un acceso directo a un archivo abierto para acceso secuencial o viceversa .

Mensajes de error.

Código	Abreviatura	Explicación
55	AO	ARCHIVO YA ABIERTO . Se intenta abrir un archivo que ya ha sido abierto . También ocurre si se da un KILL a un archivo abierto .
56	IO	ERROR DE ENTRADA/SALIDA DEL DISCO . Se detecta durante la lectura de disco .
57	FE	INDEFINIDO PARA EL BASIC MODEL II .
58	UE	ERROR INDEFINIDO . Igual al 21 .
59	DF	DISCO LLENO . Todo el espacio para almacenamiento del disco ha sido usado . Borrar archivos no útiles o usar un disco no lleno completamente .
60	EF	FIN DE ARCHIVO . Se trata de leer después de haber pasado el fin de archivo .
61	RN	NUMERO DE REGISTRO ERRONEO . En un PUT o GET, el número de registro es mayor al máximo permitido o es cero .
62	NM	INDEFINIDO PARA EL BASIC MODEL II .
63	MM	CONFLICTO DE MODOS . Un OPEN secuencial fue dado para un archivo ya existente en disco como de acceso directo o viceversa .
64	UE	ERROR INDEFINIDO . Igual al 21 .
65	DS	DECLARACION DIRECTA . Una declaración directa fue encontrada durante la carga de un programa en formato ASCII . La carga es terminada .
66	FL	INDEFINIDO PARA EL BASIC MODEL II .

B A S I C E N A L T O S .

La computadora Altos 586 es un sistema que puede manejar hasta 6 usuarios simultáneamente. Es compacta y rápida .

El microprocesador de la Altos 586 es el Intel 8086.

El Basic que tiene es el Basic-86 de Microsoft.

Las principales diferencias con el Basic de la Radio Shack son :

-En el open para archivos directos se pone "R" de random,el valor por omisión de longitud de registro es de 128 bytes .

-La forma de llamar a basic es :

BASIC [<ARCHIVO>] [/F:n] [/S:m] [/M:h]

Si "archivo" esta presente entonces ese programa se ejecuta .

La opción "F" indica número de files que se pueden manejar simultáneamente,por omisión su valor es de tres .

La opción "M" coloca la más alta dirección de memoria que podrá ser usada por el Basic.

La opción "S" nos ayuda a especificar la máxima longitud de registro de los archivos,por omisión su valor es de 128 bytes .

-La instrucción CLEAR [,[exp1][,exp2]] que además de las funciones ya conocidas,cierra todos los archivos abiertos. La "exp1" se refiere a la máxima dirección accesible para el Basic;la "exp2" incrementa el espacio de "stack"para Basic.Por omisión es de 256 bytes . El Basic asigna el espacio para cadenas ,"strings",dinámicamente .

-La instrucción CLOSE,GET,OPEN,PUT pueden llevar antes del número del file el simbolo "#".

-La figura del WHILE está implementada,con el formato siguiente :

```
WHILE <expresión>
        Proceso
WEND
```

Mientras la "expresión" sea verdadera el "proceso" se estará ejecutando iterativamente,hasta que sea falsa la "condición" y la iteración termine .

-La instrucción CLS no existe .

-Los carácteres de control son los siguientes :

control-A . Para entrar al editor en la línea que se tecleó inmediatamente antes .

control C . Interrumpe la ejecución del programa y retorna al nivel de comandos de Basic .

control G . Toca la campana en la terminal .

control H . Backspace .

control I . Tab .

control O . Detiene la salida de un programa mientras la ejecución continua . Para restablecer la salida se oprime de nueva vez "control O".

control R . Despliega de nuevo la línea que se está tecleando .

control S . Suspende la ejecución del programa .

control Q . Reanuda la ejecución del programa después de "control S" .

control U . Borra la línea que en ese momento se esté tecleando .

-Otras instrucciones y comandos del Basic-86 son :

CHAIN . Llama a un programa y le pasa variables desde el programa actual .

COMMON . Para pasar variables a un programa de CHAIN .

NULL . Para poner carácteres nulos que se imprimirán al final de cada línea .

OPTION BASE . Para declarar el valor mínimo del índice de un arreglo,cero o uno. Por omisión es cero,igual que en la Radio Shack .

La instrucción < SYSTEM "comando" > no está implementada,sólo SYSTEM , para regresar al sistema .

Consultar la sintaxis completa en el manual de "Microsoft Basic . User's Guide" disponible en el laboratorio .

N O M I N A .

Descripción de un sistema de nómina .

El sistema de nómina proporciona a una empresa los recursos y controles necesarios para manejar la información sobre los empleados , de una forma rápida y eficiente al conjuntarse con el uso de la computadora .

Un sistema de nómina deberá lograr mantener el flujo de información de los empleados que laboran en las distintas areas de la empresa , simplificando el acceso a la información actualizada y oportuna , en el momento en que se requiera.

El sistema de nómina asiliza y simplifica el proceso para determinar el monto de la remuneración a los empleados , debido a que se valida la información cuando es introducida a la computadora , permitiendo realizar las correcciones correspondientes en ese mismo momento , dando de esta manera mayor seguridad al proceso .

Caracteristicas .

Un sistema de nómina deberá constar de cuatro áreas básicas:

-Creación y mantenimiento de archivos .

-Movimiento de deshajos y el cálculo correspondiente .

-Cálculo de la nomina y emisión de nómina , recibos y/o cheques .

-Procesos mensuales,bimestrales y anuales .

La introducción de datos se realizará en forma interactiva,validando en cada caso los datos,marcando al operador los errores y pidiéndole la corrección .

La organización de los procedimientos de operación del sistema se realiza mediante "menús",que facilitan seleccionar la tarea que se desea realizar .

Los datos de percepciones y deducciones del empleado en cada periodo se guardan para hacer posteriormente las declaraciones y reportes necesarios para control lesal y empresarial del empleado .

Para el cálculo y declaraciones de tarifas legales se deben de tomar en cuenta las siguientes leyes :

-Ley del Seguro Social .

-Ley Federal del Trabajo .

-Ley del Impuesto Sobre la Renta .

Información proporcionada por el sistema de nómina .

Los reportes que se deben de obtener son :

Cada período de pago:

Nómina .

Recibos de pago .

Mensualmente:

Declaraciones de créditos de FONACOT .

Declaraciones del impuesto retenido y uno por ciento sobre remuneraciones pagadas .

Bimestralmente:

Reporte de créditos e importes totales al INFONAVIT .

Reporte de cuotas retenidas, referente al IMSS .

Anualmente:

Declaración de ISPT .

Reparto de utilidades .

Datos para el sistema .

Un sistema de nómina funciona a base de :

1.-Datos generales del empleado,como son:Nombre,Número de afiliación del Seguro Social,Tipo de empleado,etc .

2.-Información sobre percepciones y deducciones fijas del empleado,como son:Salario,Cuota sindical,Cuota de caja de ahorro,etc .

3.-Información sobre percepciones y deducciones variables de los empleados,como son:Horas Extra , Faltas , Incapacidades , etc .

Toda esta información permitirá hacer el cálculo del impuesto a retener, la cuota del Seguro Social y el total a pagar a cada empleado según las disposiciones legales,aplicables conforme a la legislación mexicana.

Los períodos de pago pueden ser semanal,decenal,quincenal y mensual , con percepciones y deduciones como: horas extra , gratificaciones, incentivos, vacaciones,premios,faltas,etc .

P R O Y E C T O D E N O M I N A P R O P U E S T O .

El proyecto de nómina que se propone es el siguiente :
Al programa que se ha ido realizando desde la práctica de archivos directos se le agregarán los siguientes procedimientos :

-Captura quincenal .
-Nómina (cheques) .
-Mantenimiento del archivo maestro .
-Reportes varios .

Captura quincenal .

Se cargarán los siguientes datos en un archivo :
[clave del empleado][categoría][horas normales][horas extra]
Se tendrá la opción de emitir el reporte del contenido de este archivo .

Nómina .

Se emitirán cheques con el siguiente formato :

```
+-------------------------------------------------------------------------+
| Compañía CHAFAMEX,S.A. de C.V.     fecha dd / mm / aaaa                 |
| ++++++++++++++++++++++++++++++     Cheque no. xxxx                      |
| Clave          Nombre del Beneficiario          RFC                     |
| xxxxx     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx     xxxx/xxxxxx              |
|                                                                         |
| Descuentos    Importe          Categorías    Importe                  |
| - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -     |
| Infonavit     xxxxx.xx         Horas norm.   xxxxxx.xx                  |
|          ------------------------ etc ------------------------          |
| Total descuentos xxxxxx.xx Total percepciones xxxxxx.xx                 |
| Importe neto = xxxxxx.xx ( con letra importe )                          |
|                                  Firma                                  |
+-------------------------------------------------------------------------+
```

Los distintos porcentajes son : para el fondo de ahorro 5%,para la couta sindical un 1%,para el IMSS de acuerdo a una tabla en que las primera 4 categorías son 4%,las siguientes 3 el 6% y las últimas 3 el 7%,para el ISPT(impuesto sobre el producto del trabajo)será 1%,para el Infonavit serán 10 porcentajes distintos:2,3,5,6,8,9,9,10,11,13 por ciento de acuerdo a la categoría del empleado.

Las horas extra se pasarán al doble de las normales .

Mantenimiento del archivo maestro .

Tendrá un menú con los siguientes procedimientos :

- Altas
- Bajas
- Modificaciones,que es el procedimiento ya realizado en la práctica de archivos directos .

Reportes varios .

Serán los siguientes :
-Reporte de los datos contenidos en el archivo maestro (ya realizado).
-Reporte de nómina conteniendo los mismos la clave,nombre,categoría,rfc,total de percepciones y de descuentos,importe neto,número de cheque y espacio para firma.
-Reporte con clave,nombre,categoría,rfc,total de importes del IMSS .
-Reporte similar al anterior pero del Infonavit .

N O T A S .
==========

Para la carga del RFC al archivo maestro se usará una rutina que haga el RFC,usando el nombre del empleado y preguntando su fecha de nacimiento .

Al realizar el cheque se deberá tener la rutina que escriba el importe total con letra a partir de la cantidad numérica .

El proyecto se deberá entregar con el listado del programa,un reporte escrito describiendo su operación,los diagramas de flujo o seudocódigos del programa,documentación acerca de los archivos y variables usadas. Se entregarán listados de resultados de cada procedimiento de la nómina,en caso de que el procedimiento altere al archivo maestro deberá entregarse un reporte del contenido previo y posterior del mismo.

A p l [illegible]

Se describirán algunas de las muchísimas aplicaciones que tiene la computadora .

Sistema de Contabilidad General .

Un sistema de este tipo debería proporcionar a la empresa los recursos y controles necesarios para manejar la información contable con elementos modernos para una oportuna toma de decisiones .

La contabilidad es el área donde se concentra toda la información referente a las distintas operaciones realizadas por la empresa con el propósito de controlar la situación financiera para la obtención del máximo rendimiento con el mínimo de recursos , reflejando en períodos determinados el avance obtenido por la empresa en el logro de este objetivo .

Un sistema de contabilidad deberá de proporcionar la siguiente información :

a) Listado diario de movimientos en forma de Libro Diario .

b) Libro Mayor mensual por mes en forma de Estado de Cuenta [illegible] en el cual a su nivel todos los movimientos de cada cuenta .

c) Balance de Comprobación a cualquiera de los cuatro niveles de [illegible] cuentas .

d) [illegible] de Pérdidas y Ganancias .

e) [illegible] .

f) Listados de Pólizas Mensuales .

g) Relación de Cuentas .

h) Catálogo de cuentas .

Sistema de Control de Ordenes y Facturación .

Un sistema de este tipo permite el fácil ingreso de pedidos a través de una pantalla,procesa la información e imprime las facturas . El obtener la facturación oportunamente y sin errores es una necesidad vital.

El sistema debe poder hacer el ingreso de pedidos, validación y corrección interactivos en línea con validación de existencia e inventario . La prefacturación o posfacturación . Devoluciones y cancelaciones . Remisión factura . Control de pedidos pendientes . Flexibilidad en la fijación de precios y descuentos . Manejo de artículos sustitutos y entregas parciales . Manejo y control de notas de crédito . Consulta interactiva por cliente , pedidos abiertos , etcetera .

Sistema de Cuentas por Cobrar .

Permite un sistema de este tipo averiguar quién debe , cuanto debe y desde cuando . Algunas características comunes de estos sistemas son :

Ingreso , validación y corrección interactivos de pagos y ajustes en cuentas de clientes .

Edad automática de las facturas de clientes .

Verificación de límites de crédito .

Soporte de cuentas consolidadas para compañías .

Impresión de estancia en tiempo de saldos detallados o resumidos .

Estados de cuenta mensuales o en otros períodos .

Sistema de Control de Inventario .

Provee un control para evitar las pérdidas de inventario debido a salidas no registradas , mediante la validación y registro de todos los movimientos . Ayuda a saber qué es lo que se tiene en existencia en el momento que se desee , comprobando además los límites de existencia .

Sus características principales son :

Costos alternos para valorización de inventarios , revisión del estado del inventario a solicitud , soporte de múltiples almacenes , cálculo del lote económico y de puntos de reorden , informes de análisis de inventario tipo financiero o de consumos .

Sistema de Análisis de Ventas .

Este sistema debe dar tratamiento a los resultados de ventas por producto y representante y debe de permitir analizar, identificar las tendencias de compra de los clientes. Será un soporte para definir las políticas de ventas de acuerdo al comportamiento de los artículos en el mercado .

Este sistema concentra y mantiene la información de facturas, notas de crédito, devoluciones, cancelaciones y remisiones de factura .

El sistema puede emitir reportes tales como : Ventas por almacén y artículo , Ventas por agente/cliente/línea de producto, Ventas por territorio/zona/agente, Contribución de cada producto al total de ventas, Ventas comparativas respecto a otros períodos, Recapitulación diaria de las ventas .

Procesadores de palabras .

Permiten capturar, editar, corregir, insertar información, [illegible], Manejo de bloques de información (copiarlos, borrarlos, moverlos, enviarlos a otro archivo, etc) . Manejo plano o de corte de hoja, control de márgenes, tamaño de hoja, tabuladores, cabeza y pie de página automático y una revisión ortográfica y gramatical .

B i b l i o g r a f i a .

BRABB , GEORGE . " Computadoras y sistemas de información en los negocios " . Ed. Interamericana .

DIEHR , GEORGE . " Programación Basic para la administración ". Editorial CECSA . México 1977 .

EUAN , JORGE . CORDERO , LUIS . " Estructuras de datos " . Facultad de Ingeniería , UNAM . México 1982 .

GOTTFRIED , BYRON . " Programación Basic " . Serie Schaum, McGraw-Hill . México 1975 .

GROSS , PAUL . SMITH , ROBERT . "System Analysis & Design for management".Dun-Donnelley . EUA 1976 .

KERNIGHAN ,BRIAN . PLAUGER ,P.J."Elementos de estilo de programación" Editorial Diana . México 1980 .

KOCH , A . K . "Sistemas de producción basados en computadora" . Editorial CECSA .

ORILIA , LAWRENCE . "Introduction to business data processing" . McGraw-Hill . Japón 1979 .

TREMBLAY , JEAN-PAUL . BUNT , RICHARD . "An introduction to computer science . An algorithmic approach " . McGraw-Hill . Japón 1981 .

P r e s e n t a c i ó n d e l a s p r á c t i c a s

1 .- Qué es un [illegible] ? En la Radio Shack cuántos tipos de [illegible] existen ? Cuáles son los tipos en que [illegible] ?

2 .- Se trata de un algoritmo de CRIPTOANALISIS que contempla la posibilidad de que el [illegible] caracteres sobre la que sobre este llena [illegible] archivos ,Datos .

3 .- [illegible] INSTR para una rutina que localice la primera aparición de una cadena de caracteres de [illegible] , En "RECINCUL" la rutina nos dobera de [illegible] .

4 .- [illegible] de 1,1 menos que en vez de [illegible] tres , [illegible] llave: la [illegible] .

5 .- Ordenar [illegible] una secuencia dada de numeros .

6 .- Qué [illegible] tiene el vector de incrementos en el SHELLsubiis [illegible] de una rutina generadora de este vector .

7 .- Cómo se puede [illegible] un archivo secuencial ? Problema de algoritmo .

8 .- Similar a la anterior pero con archivos directos .

9 .- Como se ordenan un archivo directo ?

10 .- Ventajas [illegible] archivos secuenciales y directos .

11 .- Muestre el funcionamiento del algoritmo de búsqueda binaria en una tabla de datos .

12 .- Porque [illegible] realizar procesos tales como [illegible] Dar un diagrama general de bloques [illegible] . Que salidas ?

13 .- [illegible] que sea conveniente [illegible] un programa general [illegible] . (ej . control [illegible]

14 .- [illegible] de Basic?ejemplo [illegible]

15 .- [illegible] un compilador y un intérprete

16 .- [illegible]

17 .- [illegible] computadora ? Que función [illegible] tipos de memoria hay ?

18 .- [illegible] IF-THEN-ELSE y su implementación en BASIC .

19 .- [illegible] ? Explique los distintos [illegible]

20 .- [illegible] que se usen en el manejo de [illegible]

21 .- Como [illegible] concatenación de cadenas en Basic ? Cuál es [illegible] una cadena en Basic ?

22 .- Cómo [illegible] Basic o Shack en el manejo de archivos ?

23 .- Diga [illegible] para el mantenimiento [illegible] archivos en [illegible]

TRSDOS™

Commands and Utilities

Braces { } are required only to resolve ambiguity.

AGAIN Tells TRSDOS to re-execute the most recently entered command.
AGAIN

APPEND *file-1* TO *file-2* Appends *file-1* onto the end of *file-2*.
APPEND NEWNAMES/DAT TO NAMES/DAT

ATTRIB Changes protection of specified file.
ATTRIB FILE/DAT ACC = QUICK, UPD = MOUSE, PROT = READ

AUTO *command* Automatically executes the specified command each time TRSDOS starts up. (AUTO by itself erases the automatic command.)
AUTO FORMS W = 80 *AUTO DO START* *AUTO*

BACKUP Calls utility program to copy all data from one formatted diskette to another formatted diskette.
BACKUP

BUILD Creates a file of commands which are automatically executed.
BUILD STARTUP

CLEAR Clears and unprotects user memory; clears screen; resets stack.
CLEAR

CLOCK Starts or stops real-time clock display.
CLOCK ON *CLOCK* *CLOCK OFF*

CLS Clears the Video Display.
CLS

COPY *oldfile* TO *newfile* Copies a file. {ABS} absolutely copies *oldfile* to *newfile* even if *newfile* already exists.
COPY FILE1/BAS TO FILE2/BAS *COPY FILE/A TO FILE/B:1 ABS*

CREATE Creates a file with pre-allocated space. {NGRANS = *granules* NRECS = *records*, LRL = *record length*, TYPE = V or F} NGRANS and NRECS, LRL are mutually exclusive.
CREATE FILE1 NRECS = 300, LRL = 126

DATE *newdate* Displays or resets today's date and time.
DATE *DATE 07/18/1979*

DEBUG Starts debugger utility. See DEBUG Subcommands.
DEBUG OFF *DEBUG ON* *DEBUG*

DIR :*d* Lists the diskette directory on drive d.
DIR *DIR :1* *DIR {SYS,PRT}*

DO Executes a JCL file.
DO STARTER

DUMP Dumps machine language program in RAM into a disk file. {START = *hex addr*, END = *hex addr*, TRA = *hex addr*, RELO = *dddd*, RORT = R or T}
DUMP PAY/CMD START = 5200 END = 5F00

ERROR Lists, or displays message associated with specified error code.
ERROR *ERROR 40* *ERROR 25*

FORMAT Calls a utility program which organizes a diskette into tracks and sectors. {D = *diskette drive*, PW = *password*, FULL or QUICK or NONE}
FORMAT *FORMAT D = ACCOUNTS, PW = mouse, FULL*

FORMS Sets printer driver parameters {P = *page size*, L = *lines*, W = *width*, C = *columns* (1 or 2), P (parallel), T (top of form)}
FORMS *FORMS W = 80* *FORMS L = 56*

FREE Displays the free space map.
FREE *FREE (START*

I Tells TRSDOS that a diskette has been changed; must be used after swapping diskettes in any drive.
I

KILL Deletes file from directory; frees space allocated to that file.
KILL FILE/BAS

LIB Lists all library commands.
LIB

LIST Lists contents of a file. {PRT, SLOW, R = *record number*, A}
LIST DATA/BAS *LIST FILE/1 SLOW*

LOAD Loads machine language file into memory.
LOAD PAY/CMD

PAUSE Causes TRSDOS to pause until operator presses ENTER; used include a "Prompt."
PAUSE *PAUSE Insert disk #21*

PURGE Allows rapid deletion of files. {SYS, PROG, DATA, ALL}
PURGE *PURGE :1*

PROT Changes file and diskette passwords.
PROT :1 {ON OFF PASSWORD, PW, NEW=H20}

RENAME Renames a file.
RENAME Mary/Smith Mary/Brown

SETCOM Sets RS232 communications. (*baud*, *word-length*, *parity*, *stop bits*)
SETCOM B *SETCOM B,D,N,1* *SETCOM OFF*

TIME *newtime* Displays time and date or resets time.
TIME *TIME 13:20:00*

VERIFY Checks disk surface integrity after write operation.
VERIFY ON *VERIFY OFF* *VERIFY*

BASIC Functions

Argument ranges are indicated below by special symbols:

- x: [−1 × 10^38, −1 × 10^−38], [1 × 10^−38, 1 × 10^38]
- n: [0,255]
- i: [−32768, 32767]
- r: [0,15]
- str: string argument
- var: variable name

ABS(*x*) Computes absolute value.
Y = ABS(X)

ASC(*str*) Returns ASCII code of first character of string.
A = ASC(T$)

ATN(*x*) Computes arctangent; value returned in radians.
Y = ATN(X/3)

CDBL(*x*) Converts to double-precision.
*X# = CDBL(N*3)*

CHR$(*c*) Returns character for ASCII, control, or graphics code.
P$ = CHR$(T)

CINT(*n*) Returns largest integer not greater than *n*.
PRINT CINT(15.0075)

COS(*x*) Computes cosine; angle must be in radians.
Y = COS(X)

CSNG(*x*) Converts to single-precision.
FC = CSNG(TM#)

CVD(*x*) Converts to double-precision after GET.
A# = CVD(GRSPAY$)

CVI(*n*) Converts to integer after GET.
PRINT CVI(I1$)

CVS(*x*) Converts to single-precision after GET.
FK = CVS(TS)

DATE$ Gets today's date as 18-character string.
D$ = DATE$

EOF(*b*) End-of-file detector for buffer *b*.
IF EOF(3) THEN CLOSE 3

ERL Returns the line number in which an error has occurred.
PRINT ERL

ERR Returns the error code of the most recent error.
IF ERR=7 THEN 650 ELSE 800

EXP(*x*) Computes natural antilog.
Y = EXP(X)

FIX(*x*) Truncates all digits to right of decimal point.
Y = FIX(X)

FRE (*numeric*) Finds amount of free memory.
F = FRE(X) *PRINT FRE(10)*

FRE (*string*) Finds free string space in bytes.
F = FRE("X") *PRINT FRE (A$)*

HEX$(*n*) Computes hexadecimal value and returns it as a string.
HX$ = HEX$(DEC)

INKEY$ Gets keyboard character if available.
A$ = INKEY$

INSTR (*pos, mainstr, substr*) Returns number which indicates the position in the main string where the substring begins; if no substring in main string, 0 is returned. If *pos* is omitted, *pos* = 1.
PRINT INSTR(S$,"VA") *X=INSTR(S$,Q$)* *Y1=INSTR(8,S$,Q$)*

INT(*x*) Returns largest whole number not greater than *x*.
Y = INT(X)

LEFT$(*str,c*) Returns left portion of string.
P$ = LEFT$(M$,7)

LEN(*str*) Returns the number of characters in a string.
X = LEN(SEN$)

LOC(*n*) Determines current record number of specified file.
PRINT LOC(1)

LOF(*n*) Determines number of last (highest-numbered) record in specified file.
Y = LOF(5)

LOG(*x*) Computes natural logarithm
Y = LOG(X)

MEM Finds amount of free memory.
PRINT MEM

MID$(*string, pos, len*) Returns a substring of another string. If length option is omitted, the entire string right of *pos* is returned.
PRINT MID$(A$,3,2) *F$=MID$(A$,3)*

MID$(*old, pos, len*)=*repl* Replaces one portion of a string with another. If length option is omitted, same number of characters in the old string will be changed as the number of characters in the replacement string.
MID$(A$,3,4)="USAFX" *MID$(A$,5)="01"*

MKD$(*x*) Makes double-precision number ready for disk write (direct access).
LSET AVG$=MKD$(3000.00001) *LSET D$=MKD$(X#)*

MKI$(*n*) Makes integer number ready for disk write (direct access).
LSET AVG$=MKI$(3000) *LSET Y$=MKI$(Y%)*

MKS$(*x*) Makes single-precision number ready for disk write (direct access).
LSET AVG$=MKS$(3000.1) *LSET M$=MKS$(N)*

OCT$(*n*) Computes octal value and returns it as a string.
OCS=OCT$(DEC)

POS(*c*) Returns column position of cursor.
PRINT TAB (40) POS(0)

RIGHT$(*str, c*) Returns right portion of string.
ZIP$ = RIGHT$(AD$,5)

RND(*n*) Generates a pseudorandom number between 1 and *n* if *n* > 1; between 0 and 1 if *n* = 0.
Y = RND(100) *PRINT RND(0)* *R = RND(X)*

ROW(c) Gets row number (0-23) where cursor is positioned.
X = ROW(Y)

SGN(x) Returns sign: −1, 0, 1 if *x* is negative, zero, positive.
*X = SGN(A*B)*

SIN(x) Computes sine; angle must be in radians.
Y = SIN(X)

SPACE$(c) Returns a string of spaces.
A$ = SPACE$(6)

SQR(x) Computes square root.
Y = SQR(A+B)

STR$(x) Converts a numeric expression to a string.
S$ = STR$(X)

STRING$(*l*, *c*) Returns string of characters of length *l*. Character *c* can be specified as an ASCII code or as a string.
B$ = STRING$ (125, "?") *B$ = STRING$ (125, 63)*

TAN(x) Computes tangent; angle must be in radians.
Y = TAN(X)

TIME$ Returns the time in 24-hour format.
A$ = TIME$

USR*n*(*x*) Calls any one of up to 10 machine-language subroutines, *n* = 0 through 9. If *n* is omitted, USR0 is assumed.
PRINT USR(-1) *X=USR5(Y)*

VAL(*str*) Evaluates a string as a number.
V% = VAL("100 DOLLARS")

VARPTR(*var*) Gets address where variable contents are stored.
Y = USR1 (VARPTR(X)) *PRINT VARPTR(X)*

VARPTR(#*b*) Returns address of data buffer *b*.
PRINT VARPTR(#5)

BASIC Edit Commands

A	Cancels changes and starts over.
***n*C**	Changes *n* characters.
***n*D**	Deletes *n* characters.
E	Ends editing and saves all changes.
H	Hacks line and inserts at end.
I	Inserts characters.
n*K*c	Kills all characters up to *n*th occurrence of *c*.
L	Lists the line.
Q	Quits edit mode and cancels all changes.
n*S*c	Searches for *n*th occurrence of *c*.
X	Extends line (inserts at end).
BACKSPACE	Backspaces cursor.
n SPACE BAR	Advance cursor *n* spaces to the right.
CLEAR	Escapes from edit subcommand (stops inserting).
ENTER	Ends editing and saves all changes.

BASIC Error Messages

Code	Abbreviation	Explanation
1	NF	NEXT without FOR
2	SN	Syntax error
3	RG	RETURN without GOSUB
4	OD	Out of data
5	FC	Illegal function call
6	OV	Overflow
7	OM	Out of memory
8	UL	Undefined line
9	BS	Subscript out of range
10	DD	Redimensioned array
11	/0	Division by zero
12	ID	Illegal direct
13	TM	Type mismatch
14	OS	Out of string space
15	LS	String too long
16	ST	String formula too complex
17	CN	Can't continue
18	UF	Undefined user function
19	NR	No RESUME
20	RW	RESUME without error
21	UE	Undefined error
22	MO	Missing operand
23	BO	Line buffer overflow
24-49	UE	Undefined error
50	FO	Field overflow
51	IE	Internal error
52	BN	Bad file number
53	FF	File not found
54	BM	Bad file mode
55	AO	File already open
56	IO	Disk I/O error
57	FE	File already exists
58	UE	Undefined error
59	DF	Disk full
60	EF	End of file
61	RN	Bad record number
62	NM	Not defined in Model II BASIC
63	MM	Mode mismatch
64	UE	Undefined error
65	DS	Direct statement in file
66	FL	Not defined in Model II BASIC

BASIC
Video Control Codes

Dec	Hex	PRINT CHR$(*code*)
01	01	Turns on blinking cursor
02	02	Turns off cursor
04	04	Turns on steady cursor
08	08	Backspaces cursor and erases character
09	09	Tabs cursor
10	0A	Line feed
13	0D	Carriage return
23	17	Erases to end of line
24	18	Erases to end of screen
25	19	Sets white-on-black mode
26	1A	Sets black-on-white mode
27	1B	Clears screen, homes cursor
30	1E	Clears Display, sets 80-character mode
31	1F	Clears Display, sets 40-character mode
252	FC	Moves cursor left
253	FD	Moves cursor right
254	FE	Moves cursor up
255	FF	Moves cursor down

Control Keys

CTRL J	Line feed.
CTRL O	Video output on-off switch.
CTRL R	Retypes the line being typed.
BACKSPACE	Cancels last character typed; moves cursor back one space.
BREAK	Interrupts anything in progress and returns to command level.
ENTER	Signifies end of current line.
ESC	Exits from subcommand.
F1	Starts edit mode (immediate line).
HOLD	Causes pause in execution.
SPACE BAR	Enters a space (blank) character and moves cursor one space forward.
TAB	Advances cursor to next tab position.

BASIC Statements

AUTO ***start, increment*** Numbers lines automatically.
AUTO *AUTO 150, 20* *AUTC*

CLEAR ***n, mem*** Reserves *n* bytes of string storage space; zeroes variables, *mem* resets memory protection.
CLEAR *CLEAR 100, 32000* *CLEAR*

CLOSE ***b*** Closes all open file-buffers or specified buffer(s) *b*.
CLOSE *CLOSE 1,2,8* *CLOS*

CLS Clears the Display.
CLS

CONT Continues execution of program after BREAK or STOP.
CONT

DATA Stores data to be accessed by a READ statement.
DATA "Lincoln, A.", 1861, Illinois

DEFDBL Defines variables as double-precision.
DEFDBL V, X-Z

DEF FN Defines a user-created function.
DEF FNL$(X) = STRING$(X,45)

DEFINT Defines variables as integer type.
DEFINT A, I-N

DEFSNG Defines variables as single-precision.
DEFSNG I, W-Z

DEFSTR Defines variables as string type.
DEFSTR C, L-Z

DEFUSR*n* Defines entry point for machine-language subroutine ca
by USR*n*, *n* = 0 through 9.
DEFUSR3=&H7D00

DELETE Erases program lines from memory.
DELETE 1205 *DELETE -80* *DELE*

DIM Dimensions an array.
DIM R(75),W(40) *DIM AR$(8, 25)* *DIM L%(3,1*

EDIT Puts Computer into edit mode for specified line; F1 does same current line if ENTER hasn't been pressed. See **BASIC Edit Comma**
EDIT 100 *EDIT.*

END Ends program execution.
END

ERASE Deletes array.
ERASE R,W,AR$

ERROR Simulates the specified error.
ERROR 1

TRS-80™ MODEL

FIELD Organizes a direct file-buffer into fields.
FIELD 1, 128 AS A$, 128 AS B$.

FOR...TO...STEP/NEXT Opens a program loop.
FOR I = 1 TO 81 (. . .), NEXT I FOR CI=0 TO 5 STEP .2(. . .), NEXT CI

GET *b, record-number* Gets specified or next record from a disk file (direct access); stores it in buffer *b*.
GET 1 *GET 1, 25*

GOSUB Transfers program control to the specified subroutine.
GOSUB 250

GOTO Transfers program control to the specified line.
GOTO 180

IF...THEN...ELSE Tests conditional expression.
IF P = Q THEN 200 *IF N%<0 THEN 150 ELSE N% = N% + 1*

INPUT Inputs data from keyboard.
INPUT X# *INPUT L, M, N* *INPUT "NEXT";N$*

INPUT$(*n*) Inputs *n* characters from the keyboard.
A$ = INPUT$(9)

INPUT$(*n,b*) Inputs *n* characters from disk into buffer *b* (sequential access).
A$ = INPUT$(12,2)

INPUT#*b,* Inputs data from buffer *b* (sequential access).
INPUT#1, A,B

KILL Deletes a disk file.
KILL "PRG/BAS" *KILL "TEXT.3"*

LET Assigns value to variable (optional).
LET X = 7.05 *LET R2 = R1* *LET C$ = "RED"*

LINE INPUT Line inputs from keyboard. ENTER ends input.
LINE INPUT A$* *LINE INPUT "ENTER YOUR NAME? ";N$

LINE INPUT#*b* Line inputs from disk into buffer *b*; X'0D', EOF, or 255th character ends input.
LINE INPUT#1, A$

LIST Lists program lines to the Video Display.
LIST *LIST 50-85* *LIST-100* *LIST 80*

LLIST Lists program lines to the Line Printer.
LLIST *LLIST.* *LLIST 50-*

LOAD Loads program file from disk. R option causes program to run and leaves open files open.
LOAD "PRG/BAS" *LOAD "TEXT.SPC.1"* *LOAD "PROG", R*

LPRINT Prints an item or list of items on the printer.
LPRINT CAP$; "is the capital of "; ST$

LPRINT TAB Moves printer carriage to specified position.
LPRINT TAB(25) "NAME"

LPRINT USING Prints formatted strings and numbers on the printer. See PRINT USING for list of field specifiers.
LPRINT USING "####";1234

LSET Left-justifies data into a direct access field.
LSET CITY$ = "DULUTH" *LSET A$=MKI$(N)*

MERGE Merges ASCII program file with resident program.
MERGE "A:B/BAS

NEW Erases program from memory; initializes all variables.
NEW

ON ERROR GOTO Sets up an error-handling routine.
ON ERROR GOTO 2100

ON ERROR GOTO 0 Disables an error-handling routine.
ON ERROR GOTO 0

ON...GOSUB Multi-way branch to specified subroutines.
ON Y GOSUB 50, 100, 150, 200

ON...GOTO Multi-way branch to specified lines.
ON X GOTO 130, 200, 210

OPEN *mode, b, file, rl* Opens file; assigns mode (I = input, O = o D = direct); assigns buffer number *b*; for direct access, record-l option may be added (default value *rl* = 256).
OPEN "O", 1, "CLIENTS/TXT" *OPEN "D", 5, "TEST/BAS",*

PRINT Prints an item or list of items on the Display at current c position.
PRINT X1 + Y1 *PRINT N,N^2* *PRINT "U.*

PRINT@ *n,* Prints beginning at *n*, *n* = 0 to 1919.
PRINT@ 930, "CENTER"

PRINT@ (*row, column*), Prints beginning at specified row and co (*row* = 0-23, *column* = 0-79).
PRINT@(12,38) "CENTER"

PRINT#*b* Writes data to file-buffer *b* (sequential access).
PRINT#1, A

PRINT SPC *n* Prints a line of *n* blanks.
PRINT SPC(25); "Hello"

PRINT TAB Moves cursor to specified tab position.
PRINT TAB(20) "NAME"

PRINT USING Prints formatted strings and numbers:

Formats numbers.
PRINT USING "#####"; 66.2

. Decimal point.
PRINT USING "##.#"; 58.76

, Displays comma to left of every third digit.
PRINT USING "####,"; 1234

****** Fills leading spaces with asterisks.
*PRINT USING "**####"; 44.0*

$$ Floating dollar sign.
PRINT USING "$$##.##"; 118.6735

****$** Floating dollar sign; fills leading spaces with asterisks.
*PRINT USING "**$#.##"; 8.333*

^^^^ Exponential format.
PRINT USING "#### ^^^^"; 8527100

+ Causes sign to be printed.

QUICK REFEREN

PRINT USING "+###"; -216
Minus sign after negative numbers, space after positive.
PRINT USING "####.#-"; -8124.420
Returns first string character.
PRINT USING "!"; "YELLOW"

paces \ String field, length of field is number of spaces + 2.
PRINT USING " \ \"; "BLUE"

UT *b, record number* Moves data from file-buffer *b* into the specified record. If *record number* is omitted, current record number is used.
PUT 1, 25 *PUT I, N* *PUT 1*

ANDOM Reseeds the random number generator.
RANDOM

EAD Reads a value from a DATA statement.
READ T *READ S$* *READ NM$, AGE*

EM Remark, instructs Computer to ignore rest of line. ' is an abbreviation for REM.
REM. PLACE COMMENTS HERE *'HERE TOO*

ENUM *new, start, increment* Renumbers resident program.
RENUM *RENUM 6000, 5000, 100* *RENUM,,5*

ESTORE Resets data pointer to first item in first data line.
RESTORE

ESUME Ends an error-handling routine by specifying where normal execution is to resume.
RESUME *RESUME 40* *RESUME NEXT*

ETURN Returns from subroutine to statement following GOSUB.
RETURN

SET Right-justifies data into a direct access field.
RSET CITY$ = "SPOKANE" *RSET B$=C$+D$*

UN Executes disk or resident program. , R option leaves open files open. Specify starting line number if desired.
RUN *RUN "PROG/BAS"* *RUN 100*

AVE Saves BASIC program on disk. , A causes file to be stored in ASCII format.
SAVE "FILE1/BAS" *SAVE "PRG/BAS.PW"* *SAVE "PR/TXT", A*

TOP Stops execution of a program.
STOP

WAP Exchanges values of two variables.
SWAP P, Q

YSTEM Returns to TRSDOS or executes a TRSDOS command and returns to BASIC.
SYSTEM *SYSTEM "FORMS T"* *SYSTEM "FREE"*

ROFF Turns off the trace.
TROFF

RON Turns on the trace.
TRON

E CARD

TRS-80™ MODEL II MICRO COMPUTER SYSTEM

Start-Up

1. **First, be sure all disk drives are empty. Then turn on all peripherals.**
2. Turn on the computer.
3. Insert a system diskette into the built-in drive (drive 0) and close the drive door. The system will initialize and TRSDOS will start up.
4. Answer DATE and TIME questions.
5. When you see the words TRSDOS READY and a blinking cursor on the display, you can type in a TRSDOS command.
6. Before using your printer, you may have to initialize with the FORMS command.
7. Whenever you insert a different diskette, you must type:
   I ENTER
   immediately after changing diskettes.
8. To start BASIC, type:
   BASIC -F:3 ENTER
   This will allow you to have up to three files open at once. The general form for starting BASIC is:
   BASIC *progname* -F:*files* -M:*lastaddress*
   where *progname* is a BASIC program file; *file* tells how many concurrent files you want; and *lastaddress* is the last (highest) address BASIC will use during program execution. Defaults are:

progname	None
-F:	Zero
-M:	Use all memory available from TRSDOS

Radio Shack®
The biggest name in little computers™

La impresión se realizó en la
Unidad de Difusión de la Facultad de Ingeniería