UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA

APUNTES DE
COMPUTADORAS Y PROGRAMACION

RENATO DESCHAMPS ESQUIVEL
IGNACIO GUZMAN SPEZIALE
FERNANDO SOLORZANO PALOMARES
JULIO VARGAS RODRIGUEZ

DIVISION DE CIENCIAS BASICAS
DEPARTAMENTO DE MATEMATICAS APLICADAS

FI/DCB/86-020

INTRODUCCION

Desde la aparición del hombre sobre la Tierra, se ha tenido la necesidad de realizar diversos tipos de cálculos, lo que propició la invención de procedimientos e instrumentos. A medida que éstos evolucionaban, permitieron efectuar dichos cálculos, cada vez con mayor eficacia y rapidez.

Hoy en día, los instrumentos de cálculo poseen un alto grado de sofisticación, gracias al empleo de las técnicas electrónicas, las cuales permiten que dichos instrumentos también posean una gran rapidez y precisión. La aplicación de instrumentos como la computadora digital abarca casi todos los campos de la actividad humana, ya sean estos científicos, tecnológicos, administrativos o sociales.

En la ingeniería, la computadora ha llegado a ser una herramienta indispensable, lo cual obliga al profesional a un conocimiento tan amplio en este campo como sea posible. Por consiguiente el estudiante de ingeniería debe adquirir este conocimiento con el objeto de que en el futuro, su labor como profesionista sea realmente fructífera.

P R O L O G O

La computación electrónica ha acelerado el desarrollo de diversos aspectos de la tecnología contemporánea y constituye hoy en día una poderosa herramienta para el ejercicio profesional del ingeniero; por esta razón la Facultad de Ingeniería de la Universidad Nacional Autónoma de México ha incluido dentro de sus planes de estudio la asignatura Computadoras y Programación, a fin de que el estudiante se familiarice con los elementos fundamentales del campo de la computación electrónica.

Estos apuntes han sido elaborados como material de apoyo para los cursos que se imparten en la Facultad de Ingeniería, y pretenden ayudar al estudiante en su aprendizaje, proporcionándole un desarrollo de los temas que comprende el programa de la asignatura; sin embargo, se le recomienda consultar además otras fuentes de información sobre el tema, ya que del análisis de diversos puntos de vista sobre un tópico surge la síntesis que realiza el estudiante y, con ello, el verdadero aprendizaje. Como guía para este propósito se enlista al final de cada capítulo la bibliografía cuya consulta se considera necesaria para completar el estudio de estos apuntes.

En los dos primeros capítulos se desarrollan las bases para la comprensión de lo que es una computadora, así como sus aplicaciones y funcionamiento.

Posteriormente se exponen los conceptos fundamentales que permiten plantear el conjunto de instrucciones para resolver un problema con ayuda de la computadora, de manera que puedan ser traducidas a lenguajes como el FORTRAN y el BASIC. Finalmente se presentan los elementos fundamentales para el empleo de las llamadas calculadoras programables.

El mejoramiento de estos apuntes podrá lograrse con ayuda de las críticas y sugerencias de profesores y alumnos, por lo que agradeceremos las aportaciones que se hagan llegar a la coordinación de la materia con el objeto de mejorar las futuras ediciones.

Expresamos nuestro reconocimiento a los señores profesores:

Renato Deschamps Esquivel
Ignacio Guzmán Speziale
Fernando Solórzano Palomares
Julio Vargas Rodríguez

por su valiosa intervención en la elaboración de estos apuntes, así como a las licenciadas:

Irma Hinojosa Félix
María Cuairán Ruidíaz

por su colaboración en la adaptación pedagógica de los mismos.

Facultad de Ingeniería
División de Ciencias Básicas
Departamento de Matemáticas Aplicadas

CONTENIDO

1. HISTORIA DE LAS COMPUTADORAS

1.1 EVOLUCION DE LOS INSTRUMENTOS DE CALCULO Y DE LAS COMPUTADORAS

Las primeras operaciones aritméticas que el hombre realizó, tales como adiciones y sustracciones sencillas, pudieron llevarse a cabo con la ayuda de piedrecillas, varas y sus dedos, aumentando o disminuyendo el número de piedras, varas o dedos desplazados o marcados. Sin embargo, cuando las cantidades aumentaron, el hombre empezó a buscar nuevas técnicas.

El Abaco

Este instrumento fue inventado en China alrededor del año 2,600 A. de C. y posteriormente pasó a otros países como Grecia y Egipto.

El ábaco agrupa varias hileras de cuentas que se deslizan en alambres montados en un marco rectangular. El marco está dividido por un elemento transversal de manera que cada hilera tiene un sector con dos cuentas y otro con cinco cuentas, pudiendo llevarse a cabo operaciones de adición, sustracción, multiplicación y división.

A pesar de ser un instrumento tan simple, el ábaco permite realizar las operaciones aritméticas más eficazmente que con las cuentas sueltas.

Tabla de Logaritmos

Fue hasta 1614 cuando un escocés de nombre John Napier desarrolló las tablas de logaritmos, el cual es un sistema tabular de números con los que es posible simplificar muchos cálculos aritméticos.

Utilizando tablas de logaritmos, las operaciones de multiplicación y de división se pueden efectuar más fácilmente reduciéndolas respectivamente a sumas y restas.

Regla de Cálculo

En 1632 un matemático inglés de nombre William Oughtred inventó la regla de cálculo. La regla de cálculo consiste en dos reglillas movibles, cada una está marcada de tal forma que las distancias desde el principio son proporcionales a los logaritmos de los números marcados en la reglilla. Al deslizarlas se pueden efectuar rápidamente operaciones de multiplicación y división. En general los resultados que se obtienen con la regla de cálculo no son exactos pero tienen gran aproximación. En la actualidad han sido desplazadas por las pequeñas calculadoras electrónicas de bolsillo.

Máquina de Pascal

En 1642, Blaise Pascal, matemático francés construyó la primera máquina de sumar mecánica.

En la máquina de sumar de Pascal, los números del 0 al 9 estaban colocados en unas ruedas giratorias. Estas ruedas representaban unidades, decenas y centenas. Las subsiguientes divisiones estaban situadas una al lado de otra de modo semejante a las varillas del ábaco. Cuando una suma era realizada en alguna columna, esta rueda giraba por cada uno de los números que tenían que sumarse. El resultado se observaba en casillas colocadas sobre cada rueda de la máquina.

Máquina de Jacquard

En 1801 Joseph Marie Jacquard construyó la primera máquina de tarjetas perforadas, diseñada por él para tejer difíciles diseños de telas. En su telar automático, que revolucionó la industria textil, el tejido lo dirigía una tarjeta. Las perforaciones de la tarjeta proporcionaban las instrucciones que controlaban la selección de hilo y la ejecución de los diseños.

Charles Babbage

Charles Babbage era profesor de matemáticas de la Universidad de Cambridge y en el año de 1812 empezó a trabajar en la construcción de una máquina que permitiría calcular tablas matemáticas, a la cual él llamó *máquina de diferencias*. Después de casi 10 años de trabajar en esta máquina se interesó Babbage en un proyecto mucho más ambicioso, la *máquina analítica*. Esta máquina incluiría una unidad de almacenamiento de memoria que guardaría los datos en forma de perforaciones de tarjetas, además tendría una unidad aritmética en donde se efectuarían las operaciones fundamentales matemáticas, y una unidad de control cuya finalidad sería de dirigir las operaciones. Desgraciadamente la máquina de Babbage parecía capaz de todo, excepto de funcionar. Posiblemente se adelantó a su época puesto que muchos de los mecanismos planteados por Babbage no fueron resueltos sino después de muchos años. Podemos resumir diciendo que Babbage no inventó nada realmente complicado, sino sólo su máquina de diferencias en una forma muy rudimentaria, pero sus ideas influyeron en otras personas las cuales construyeron máquinas mucho más complejas.

Herman Hollerith

En América 20 años después de la muerte de Babbage hay un gran adelanto, se trata de la invención de las máquinas calculadoras de tarjetas perforadas de Herman Hollerith, estadístico de la oficina del censo en los Estados Unidos. Las leyes americanas exigían que se realizara un censo cada década. En el año de 1886 todavía no podían terminar con el censo de 1880 y era evidente que con el equipo que contaban no quedaría terminado en 1890, año en que tenía ya que iniciarse el nuevo censo. Hollerith pensó que el problema era por no tomar algunas medidas de mecanización y empezó la tarea de construir el equipo apropiado. Estaba familiarizado con los trabajos efectuados por Jacquard en los telares y en los cuales se habían utilizado tarjetas perforadas para automatizar los diseños de las telas.

El se dio cuenta que para la mayoría de las preguntas que se planteaban, las respuestas eran *sí* o *no*, lo cual podía ser representado en una tarjeta como la presencia o ausencia de una perforación en dicha tarjeta. Así mismo, que una cantidad numérica se podría representar a través de la posición que ocupase la perforación en cada columna de la tarjeta.

Hollerith, con dispositivos que funcionaban con el principio anterior, experimentó en la clasificación y recuento (principales operaciones censales) y algunas de sus máquinas se utilizaron ya en el censo de 1890.

En 1896 Hollerith organizó una compañía de máquinas tabuladoras para desarrollar sus máquinas y venderlas.

La Primera Computadora Electromecánica y las Primeras Computadoras Electrónicas

Fue hasta el año de 1937, cuando el profesor Howard Aiken de la Universidad de Harvard pensó utilizar los principios de Babbage y de Hollerith para construir un mecanismo automático de cálculo. En mayo de 1944 y con la corporación IBM se construyó una calculadora automática y de control de secuencias llamada *Mark I*. Esta era una máquina electromecánica formada de partes de equipo IBM, utilizaba relevadores y estaba controlada por una cinta de papel perforado. Finalmente se realizó el sueño de Babbage.

La *Mark I* dio comienzo a nuevas máquinas electromecánicas, como la *Mark II*, diseñada para los campos de tiro de la marina de los E.E.U.U.

Estas nuevas computadoras representaron un gran avance. Con la segunda guerra mundial la tecnología aumentó notablemente, pero como las máquinas eran dispositivos electromecánicos con relevadores o ruedas contadoras, su efectividad estaba restringida por la lentitud y las dificultades mecánicas de operación. Estos problemas se superaron con el siguiente desarrollo importante en la Historia de las técnicas de procesamiento de datos.

La primera máquina que utilizó tubos electrónicos al vacío para calcular fue la ENIAC, (*Electronic Numerical Integrator and Calculator*) desarrollada entre 1942 y 1946 en la Universidad de Pennsylvania por John W. Mauchly, J. Presper Eckert y sus asociados. La computadora ocupó todo el sótano de uno de los edificios de la Universidad (más de 150 m^2), y pesaba más de 30 toneladas, conteniendo más de 18,000 tubos electrónicos. La ENIAC podía completar en un día aquellos procesos que requerían 30 días en las computadoras electromecánicas.

Fue diseñada especialmente para resolver problemas de balística en los campos de pruebas de Aberdeen y se usó hasta 1955.

La computadora ENIAC era una máquina que usaba 20 acumuladores para almacenar datos, cada uno podía manejar 10 dígitos. Cada dos tubos representaba un dígito binario (*Binary Digit BIT*), la entrada y salida de información se realizaba por medio de tarjetas perforadas.

UNIVAC (*Universal Automatic Computer*) fabricada y diseñada por Sperry Rand Corporation se considera el primer paso hacia el procesamiento de

datos completamente automático. Fue una de las primeras máquinas que utilizaron la cinta magnética como mecanismo de entrada y salida de información.

En los años cincuenta se introdujo el tambor magnético, el cual era un nuevo dispositivo para almacenar información aumentando la capacidad de memoria de las computadoras, la primera máquina construida con este equipo fue SEC (*Simple Electronic Computer*) desarrollada en Londres.

Generaciones de Computadoras

La primera generación de computadoras digitales utilizó *tubos al vacío* como componentes electrónicos básicos, esto dio como resultado que el costo, volumen, consumo de fuerza, calentamiento, el retardo de lógica y la cantidad de fallas de cada red fueran muy elevadas comparadas con las redes de semiconductores utilizadas hoy en día.

La segunda generación de computadoras electrónicas, usa *transistores* y los componentes del diodo del semiconductor. Estos elementos junto con condensadores se montan en tarjetas de *circuitos impresos*.

La tercera generación utiliza *Circuitos Integrados* (Chip).

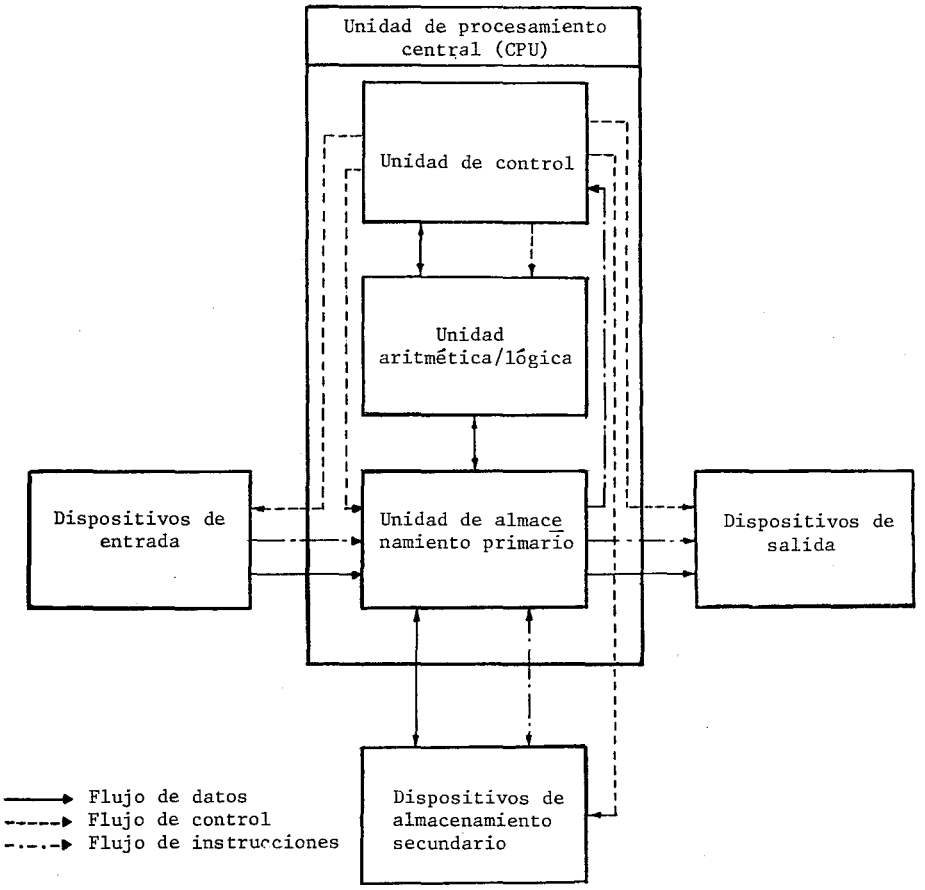
Este nuevo elemento es tan pequeño que un dedal de costura puede contener más de 50,000 elementos (Chips) y cada uno de ellos estar formado por miles de circuitos. Por esta razón también reciben el nombre de *microcircuitos*.

Las ventajas surgidas debido a los adelantos de la tercera generación son los siguientes:

- Se autogobierna y maneja sin detenerse, pasando de un trabajo a otro.
- El operador interviene sólo cuando hay problemas.
- La máquina se autodiagnostica e indica cuál es la anomalía.
- Todas las operaciones del sistema se realizan simultáneamente.
- Han evolucionado notablemente los lenguajes de programación.
- Gran autocontrol y autoverificación.
- Multiprogramación.
- Multiproceso.
- La velocidad entrada-proceso-salida se ha incrementado notablemente.

1.2 COMPONENTES Y FUNCIONAMIENTO DE LAS COMPUTADORAS DIGITALES

DIAGRAMA ESQUEMATICO DE UNA COMPUTADORA



Memoria o Almacenamiento de la Computadora

Consta de varias *charolas o tarjetas* en las cuales se encuentran los *núcleos*. Estos núcleos están acomodados en las tarjetas en un arreglo tal, que si imaginamos la tarjeta como una hoja cuadriculada, en cada cruce de dos líneas existe uno de estos núcleos.

Los núcleos que son fabricados de muy diversos materiales, son celdas que pueden mantenerse en uno de dos estados: prendido-apagado, magnetizado en un sentido o en sentido inverso, etc. La computadora puede revisar, en qué estado está determinado núcleo, o bien cambiarlo por el otro estado. Se considera a uno de los estados como 1 y al otro como 0, (prendido 1, apagado 0).

En ambos casos la información contenida en un núcleo es la unidad más pequeña que se puede tener de información y recibe el nombre de *BIT* (*Binary Digit*).

Un núcleo puede almacenar un dígito binario 0 y 1, pero un conjunto de núcleos pueden almacenarse y formar una palabra (*word*). Una palabra nos sirve para representar un valor de una variable.

Existen longitudes de palabra de 8, 12, 36, 48 ó más bits, un conjunto de 6 a 8 bits formarán un *Byte*.

Unidad de Control

Esta unidad tiene el papel supervisor para toda la máquina, toma las instrucciones de la memoria en la sucesión apropiada, interpreta las instrucciones y hace que las componentes apropiadas de la máquina efectúen las operaciones especificadas por las instrucciones. Como resultado de las operaciones lógicas en la unidad de procesamiento (aritmética y lógica), también puede, durante el proceso, seleccionar diferentes alternativas en las instrucciones.

Sistema Operativo

El sistema operativo es un programa que se encarga de administrar los recursos de la computadora.

Dentro de las ventajas surgidas del uso de los sistemas operativos se tienen las siguientes:

- Permite aumentar la velocidad de proceso, ejecutando programas independientes en forma consecutiva y sin interrupción.
- Incluye compiladores y ensambladores que facilitan la programación.
- La participación del operador se reduce al mínimo.
- Incluye programas de definición para ayudar al programador a descubrir y corregir errores de programación.
- Permite reducir el tiempo ocioso del CPU al atender por separado la ejecución de trabajos en el CPU y la ejecución de entradas y salidas.

El sistema operativo tiene el control de todos los programas. Esta es la tarea más importante del sistema.

Unidad Aritmética y Lógica

Esta unidad desarrolla todas las operaciones aritméticas y lógicas.

En las operaciones aritméticas su función es semejante a las de los registros de una calculadora de escritorio. Sin embargo, también puede ejecutar operaciones lógicas tales como comparar un número contra otro o comparar dos variables, etc.

Unidad de Entrada

Se usa esta unidad para introducir información a la computadora y puede consistir en lectoras de tarjetas perforadas, de cintas de papel o de cintas magnéticas, pantallas de rayos catódicos, lectoras opticas de caracteres, etc.

Unidad de Salida

Es a través de ésta que la computadora transmite información al usuario, por medio de cintas, impresoras, tubos de rayos catódicos. graficadores, etc.

Memoria Auxiliar

Este almacenamiento complementa al almacenamiento principal de la computadora y generalmente guarda cantidades mayores de datos. Los dispositivos de almacenamiento auxiliar pueden guardar desde varios cientos de miles hasta varios cientos de millones de caracteres de datos en forma secuencial o pseudoaleatoria.

Se tienen dos tipos de almacenamiento auxiliar, de acceso al azar y de acceso secuencial. Como ejemplo del primer tipo de acceso se tienen las unidades de disco y las unidades de cinta magnética como ejemplo del segundo tipo.

1.3 LENGUAJES UTILIZADOS EN LAS COMPUTADORAS

El cómo hacer que una computadora lleve a cabo la tarea de cálculo que deseamos, se logra mediante una serie de instrucciones que obedecen ciertas reglas y métodos utilizando por llamarle de alguna manera, un vocabulario propio.

De acuerdo al problema que se tenga (científico, técnico, administrativo) se empleará el lenguaje más apropiado.

El programa se debe realizar de acuerdo a las reglas del lenguaje para poder transmitir a la computadora las instrucciones que debe realizar exactamente, este conjunto de instrucciones se alimentan a la computadora por su unidad de entrada, las cuales son traducidas por el compilador al lenguaje de máquina.

Esto trae como consecuencia que para cada lenguaje diferente, debe haber un traductor o compilador que transforme nuestras instrucciones al lenguaje de máquina.

El compilador es un programa suministrado generalmente por el fabricante del equipo de cómputo.

El decidir qué lenguaje de programación es conveniente utilizar depende de algunos factores tales como características del problema a resolver, memoria disponible de la computadora en que se va a procesar, compiladores o traductores de que dispone dicha máquina, etc.

Por lo general es posible distinguir dos grupos de problemas, los que son técnicos o científicos y los que son administrativos.

En el grupo de los técnicos o científicos se observa que intervienen muchos cálculos y sin embargo, en el reporte de salida hay poca información. Por el contrario en el grupo de los administrativos, existen pocos cálculos y una amplia variedad de reportes de salida.

En el grupo de problemas técnicos y científicos los lenguajes más comúnmente utilizados son: FORTRAN, ALGOL, BASIC y PASCAL.

En el grupo de problemas administrativos, el lenguaje más utilizado es el COBOL y para reportes el RPG.

Los lenguajes antes mencionados, son los que reciben la denominación de lenguajes de alto nivel o superlenguajes.

Podemos decir en forma concreta, que un lenguaje de programación de alto nivel, es aquél que permite con una sola instrucción, hacer una serie de operaciones que en otros lenguajes (ensambladores) deben detallarse una por una.

En los lenguajes de alto nivel, por lo general se observan las siguientes ventajas:

- El tiempo de programación se reduce considerablemente.
- Es compatible con pequeños cambios de un sistema a otro.
- Se aprende con facilidad.
- Simplifica cambios o correcciones que sea necesario hacerle al programa.

BIBLIOGRAFIA

Luz Ma. Silva de Mejía
REALIDADES Y FANTASIAS DE LAS COMPUTADORAS
Universidad Nacional Autónoma de México
México, 1978

Arechiga G. Rafael
FUNDAMENTOS DE COMPUTACION
Segunda edición
Editorial Limusa
México, 1978

Awad Elias M.
PROCESAMIENTO AUTOMATICO DE DATOS
Primera edición
Editorial Diana
México, 1976

2. ESTRUCTURA Y CONFIGURACION DE SISTEMAS

2.1 REPRESENTACION DE NUMEROS EN LOS SISTEMAS BINARIO, OCTAL Y HEXADECIMAL

En el capítulo anterior al hablar de cómo almacena información la computadora, se mencionó que en los circuitos se observa uno de dos estados: prendido-apagado, magnetizando en un sentido o en sentido inverso, nivel alto de voltaje-nivel bajo de voltaje, etc.

La lógica de los circuitos de una computadora trabaja con el sistema binario. En dicho sistema hay solamente dos números aceptables, 0 y 1 (se llama binario debido a que significa *dos*). Usando estos dos símbolos es posible representar el equivalente de cualquier cantidad decimal combinando los dos símbolos (0 y 1).

Es trabajo de la computadora el convertir la información que se le suministra en sistema decimal a sistema binario para su proceso y nuevamente el convertir los resultados del sistema binario al decimal para su fácil interpretación.

Cabe mencionar que en algunos equipos de computación, además del sistema binario, se utilizan los sistemas octal y hexadecimal para transmitir mensajes al operador de la computadora.

Los elementos que integran a cada uno de los sistemas citados, son los siguientes:

Hexadecimal	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
Decimal	0,1,2,3,4,5,6,7,8,9
Octal	0,1,2,3,4,5,6,7
Binario	0,1

Se dice que los sistemas anteriores son de posición, debido a que el valor que se le asigna a una cantidad depende de la posición relativa de sus símbolos.

Los sistemas numéricos de posición quedan representados por la siguiente fórmula:

$$N_q = \sum_{i=n}^{-m} X_i \; q^i \qquad (2.1)$$

Donde:

N. Número representado

q. Base del sistema al cual pertenece dicho número.

i. Posición que ocupan los componentes.

n. Límite superior en la sumatoria de la ecuación.

-m. Límite inferior en la sumatoria de la ecuación.

X. Dígitos componentes del número.

Al desarrollar la ecuación anterior se tiene:

$$N_q = X_n q^n + X_{n-1} q^{n-1} + X_{n-2} q^{n-2} + \ldots + X_1 q^1 + X_0 q^0 + X_{-1} q^{-1} + X_{-2} q^{-2} + \ldots + X_{-m} q^{-m}$$

Los índices positivos corresponden a la parte entera mientras que los negativos corresponden a la parte fraccionaria de la cantidad a representar. El punto decimal debe considerarse inmediatamente antes del primer subíndice negativo.

Ejemplo

Sea:

$N_q = (1968.32)_{10}$

De acuerdo a la ecuación anterior nos queda:

$$(1968.32)_{10} = 1 \times 10^3 + 9 \times 10^2 + 6 \times 10^1 + 8 \times 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2}$$

$$(1968.32)_{10} = 1000 + 900 + 60 + 8 + \frac{3}{10} + \frac{2}{100}$$

$$(1968.32)_{10} = 1968 + \frac{32}{100} = 1968.32$$

En el ejemplo anterior se nota claramente la representación por posición.

Método de Conversión de Cantidades de Base Diez a Otra Diferente

Este método se lleva a cabo en dos partes, primero se considera la parte entera de la cantidad y posteriormente la parte fraccionaria.

Parte Entera

Sea primero la parte entera, la expresión nos queda de la siguiente manera:

$$\boxed{N = X_n q^n + X_{n-1} q^{n-1} + \ldots + X_3 q^3 + X_2 q^2 + X_1 q^1 + X_0 q^0} \qquad (2.2)$$

Tomemos primero la parte entera; el proceso consiste en dividir el número N en base diez, entre la base q a convertir, y considerando únicamente el cociente entero. De esta manera obtenemos al dividir un cociente Co y un residuo Ro.

$$\frac{N}{q} = Co + Ro$$

Este residuo multiplicado por q pasará a ser el coeficiente Xo, de la base q elevado a la potencia cero.

$$Xo = q \cdot Ro$$

Tomamos ahora el cociente Co como nuevo número N y repetimos el proceso, obteniendo un cociente C_1 y un residuo R_1, el cual multiplicado por q será el coeficiente X_1 de la base q elevada a la potencia uno.

$$X_1 = q \cdot R_1$$

El proceso se vuelve iterativo, pero no es infinito ya que se suspende cuando tenemos como cociente Cn el cero.

Cuando la base a convertir sea mayor que diez, puede ocurrir que algunos o todos los residuos multiplicados por q sean mayores que diez, en este caso se debe tomar como coeficiente el símbolo cuyo valor sea equivalente al residuo obtenido.

Ejemplo 1

Convertir el número 35 de base 10 al número correspondiente de base 2.

$$(35)_{10} = (?)_2$$

Aplicando el método tenemos:

$$\frac{35}{2} = 17 + \frac{1}{2} \ldots\ldots\ldots X_0 = 1$$

$$\frac{17}{2} = 8 + \frac{1}{2} \ldots\ldots\ldots X_1 = 1$$

$$\frac{8}{2} = 4 + \frac{0}{2} \ldots\ldots\ldots X_2 = 0$$

$$\frac{4}{2} = 2 + \frac{0}{2} \ldots\ldots\ldots X_3 = 0$$

$$\frac{2}{2} = 1 + \frac{0}{2} \ldots\ldots\ldots X_4 = 0$$

$$\frac{1}{2} = 0 + \frac{1}{2} \ldots\ldots\ldots X_5 = 1$$

La cantidad buscada será por lo tanto:

$$X_5\ X_4\ X_3\ X_2\ X_1\ X_0$$

$$1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1$$

$$(35)_{10} = (1\ 0\ 0\ 0\ 1\ 1)_2$$

Para comprobar apliquemos la ecuación (2.1) al número obtenido, tenemos:

$$(1\ 0\ 0\ 0\ 1\ 1)_2 = (1x2^5+0x2^4+0x2^3+0x2^2+1x2^1+1x2^0)_{10}$$

$$(1\ 0\ 0\ 0\ 1\ 1)_2 = (32 + 0 + 0 + 0 + 2 + 1)_{10}$$

$$(1\ 0\ 0\ 0\ 1\ 1)_2 = (35)_{10}\ \text{L.q.q.d.}$$

Ejemplo 2

Convertir el número 49 de base 10 al número correspondiente de base 2.

$$(49)_{10} = (?)_2$$

Aplicando el método tenemos:

$$\frac{49}{2} = 24 + \frac{1}{2} \quad \ldots\ldots\ X_0 = 1$$

$$\frac{24}{2} = 12 + \frac{0}{2} \quad \ldots\ldots\ X_1 = 0$$

$$\frac{12}{2} = 6 + \frac{0}{2} \quad \ldots\ldots\ X_2 = 0$$

$$\frac{6}{2} = 3 + \frac{0}{2} \quad \ldots\ldots\ X_3 = 0$$

$$\frac{3}{2} = 1 + \frac{1}{2} \quad \ldots\ldots\ X_4 = 1$$

$$\frac{1}{2} = 0 + \frac{1}{2} \quad \ldots\ldots\ X_5 = 1$$

La cantidad buscada será por lo tanto:

$$(49)_{10} = (1\ 1\ 0\ 0\ 0\ 1)_2$$

Comprobemos aplicando la ecuación (2.1)

$$(1\ 1\ 0\ 0\ 0\ 1)_2 = (1x2^5+1x2^4+0x2^3+0x2^2+0x2^1+1x2^0)_{10}$$

$$(1\ 1\ 0\ 0\ 0\ 1)_2 = (32 + 16 + 0 + 0 + 0 + 1)_{10}$$

$$(1\ 1\ 0\ 0\ 0\ 1)_2 = (49)_{10} \quad \text{L.q.q.d.}$$

Ejemplo 3

Convertir el número 35 de base 10 al número correspondiente de base 8.

$$(35)_{10} = (?)_8$$

Aplicando el método tenemos:

$$\frac{35}{8} = 4 + \frac{3}{8} \quad \ldots\ldots X_0 = 3$$

$$\frac{4}{8} = 0 + \frac{4}{8} \quad \ldots\ldots X_1 = 4$$

La cantidad buscada será por lo tanto:

$$(35)_{10} = (43)_8$$

Comprobemos aplicando la ecuación (2.1)

$$(43)_8 = (4x8^1+3x8^0)_{10}$$

$$(43)_8 = (4x8+3x1)_{10}$$

$$(43)_8 = (35)_{10} \quad \text{L.q.q.d.}$$

Ejemplo 4

Convertir el número 1750 de base 10 al número correspondiente de base 16.

$$(1750)_{10} = (?)_{16}$$

Aplicando el método tenemos:

$$\frac{1750}{16} = 109 + \frac{6}{16} \ldots\ldots\ldots X_0 = 6$$

$$\frac{109}{16} = 6 + \frac{13}{16} \ldots\ldots\ldots X_1 = 13\ (D)$$

$$\frac{6}{16} = 0 + \frac{6}{16} \ldots\ldots\ldots X_2 = 6$$

La cantidad buscada será por lo tanto:

$$(1750)_{10} = (6\ D\ 6)_{16}$$

Comprobemos aplicando la ecuación (2.1)

$$(6\ D\ 6)_{16} = (6x16^2+13x16^1+6x16^0)_{10}$$

$$(6\ D\ 6)_{16} = (6x256+13x16+6x1)_{10}$$

$$(6\ D\ 6)_{16} = (1536+208+6)_{10}$$

$$(6\ D\ 6)_{16} = (1750)_{10} \quad \text{L.q.q.d.}$$

Concluyendo, tenemos que para expresar una cantidad en una base dada a otra diferente, bastará con efectuar divisiones sucesivas de la cantidad dada sobre la base a la cual se desea pasar, y por cada división se tendrá una componente de la nueva cantidad, siendo ésta el residuo.

Parte Fraccionaria

Veamos ahora la parte fraccionaria, la expresión nos queda de la siguiente forma:

$$\boxed{N=X_{-1}q^{-1}+X_{-2}q^{-2}+\ldots.+X_{-m}q^{-m}} \qquad (2.3)$$

La expresión anterior se puede expresar también de la siguiente manera:

$$N = \frac{X_{-1}}{q} + \frac{X_{-2}}{q^2} + \ldots + \frac{X_{-m}}{q^m}$$

El método es similar en procedimiento al de la parte entera, sin embargo ahora en lugar de dividir la cantidad sobre la base a la cual se desea pasar, habrá que multiplicarla.

Por lo tanto tenemos:

$$N \cdot q = C_{-1} + F_{-1}$$

Obteniendo como resultado, un entero C_{-1} y una parte fraccionaria F_{-1}

El entero C_{-1} resultante del producto anterior es la componente X_{-1} de la cantidad buscada.

$$F_{-1} \cdot q = C_{-2} + F_{-2}$$

En la expresión anterior C_{-2} nos representa la siguiente componente de la cantidad buscada.

Prosiguiendo en forma iterativa el proceso anterior, obtenemos las componentes:

$$X_{-3}, X_{-4}, \ldots, X_{-m}$$

Como en el caso de la parte entera, cuando el valor C_{-n} sea mayor que diez, se pondrá el símbolo equivalente al valor C_{-n} encontrado. (Ver ejemplo 7).

Ejemplo 5

Convertir el número 0.75 de base 10 al número correspondiente de base 2.

$$(0.75)_{10} = (?)_2$$

Aplicando el método descrito tenemos:

$$0.75 \times 2 = 1 + 0.5 \ldots . X_{-1} = 1$$

$$0.5 \times 2 = 1 + 0.0 \ldots . X_{-2} = 1$$

Por lo tanto:

$$(0.75)_{10} = (0.11)_2$$

Comprobemos aplicando la ecuación (2.1)

$$(0.11)_2 = (1 \times 2^{-1} + 1 \times 2^{-2})_{10}$$

$$(0.11)_2 = (\frac{1}{2} + \frac{1}{4})_{10}$$

$$(0.11)_2 = (0.5 + 0.25)_{10}$$

$$(0.11)_2 = (0.75)_{10} \quad \text{L.q.q.d.}$$

Como se observa, al llevar a cabo las operaciones, éstas se suspenden cuando se tiene como parte fraccionaria el cero, sin embargo en algunas ocasiones no es posible llegar al cero por lo que el número de operaciones repetitivas se limita a un número establecido de componentes, o bien

en el momento en que la obtención de componentes se vuelve cíclica.

Ejemplo 6

Convertir el número (0.75) de base 10 al número correspondiente de base 8.

$$(0.75)_{10} = (?)_8$$

Aplicando el método tenemos:

$$0.75 \times 8 = 6 + 0.0 \quad X_{-1} = 6$$

Comprobación:

$$(0.6)_8 = (6 \times 8^{-1})_{10}$$

$$(0.6)_8 = \frac{6}{8} = \frac{3}{4} = (0.75)_{10} \qquad \text{L.q.q.d.}$$

Ejemplo 7

Convertir el número 0.18 de base 10 al correspondiente de base 16.

$$(0.18)_{10} = (?)_{16}$$

Aplicando el método tenemos:

$$0.18 \times 16 = 2.0 + 0.88 \ldots X_{-1} = 2$$

$$0.88 \times 16 = 14.0 + 0.08 \ldots X_{-2} = 14 \ (E)$$

$$0.08 \times 16 = 1.0 + 0.28 \ldots X_{-3} = 1$$

$$0.28 \times 16 = 4.0 + 0.48 \ldots X_{-4} = 4$$

$$0.48 \times 16 = 7.0 + 0.68 \ldots X_{-5} = 7$$

$$0.68 \times 16 = 10 + 0.88 \ldots X_{-6} = 10 \ (A)$$

Como se puede observar en este ejemplo, nunca llegaremos a tener como fracción el cero, sin embargo notamos que si continuamos adelante, la obtención de la componente ya es cíclica, ya que la fracción 0.88 inicial es igual a la última fracción obtenida.

Por lo tanto:

$$(0.18)_{10} = (.2\ E\ 1\ 4\ 7\ A)_{16}$$

Comprobación:

$$(.2\ E\ 1\ 4\ 7\ A)_{16} = (2x16^{-1}+14x16^{-2}+1x16^{-3}+4x16^{-4}+7x16^{-5}+10x16^{-6})_{10}$$

$$(.2\ E\ 1\ 4\ 7\ A)_{16} = \left(\frac{2}{16} + \frac{14}{16^2} + \frac{1}{16^3} + \frac{4}{16^4} + \frac{7}{16^5} + \frac{10}{16^6} \right)$$

$$= 0.125 + 0.0547 + 0.0002 \cong (0.18)_{10}$$

Existe una forma simple para pasar una cantidad que se encuentra en sistema binario a un sistema octal o hexadecimal y viceversa.

El método consiste en considerar el hecho de que tres dígitos binarios, representan un dígito octal, así tenemos:

Ejemplo 8

Convertir la cantidad 101101111 que está en binario a la correspondiente en el sistema octal.

Bastará dividir en cifras de 3 de derecha a izquierda.

$$\underbrace{1\ 0\ 1}\ \underbrace{1\ 0\ 1}\ \underbrace{1\ 1\ 1}$$

$$(1\ 0\ 1)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

$$(1\ 0\ 1)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

$$(1\ 1\ 1)_2 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 7$$

Por lo tanto:

$$(1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1)_2 = (557)_8$$

Para la parte fraccionaria se hace igualmente la división en cifras de 3, pero iniciando de izquierda a derecha, sea:

Ejemplo 9

Convertir la cantidad 0.101011011 que está en binario a la correspondiente en base ocho.

$$0.\ \underbrace{1\ 0\ 1}\ \underbrace{0\ 1\ 1}\ \underbrace{0\ 1\ 1}$$

$$(1\ 0\ 1)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

$$(0\ 1\ 1)_2 = 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3$$

$$(0\ 1\ 1)_2 = 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3$$

$$(0.\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1)_2 = (0.533)_8$$

Por lo que respecta ahora al sistema hexadecimal se considera que un dígito en este sistema está representado por cuatro dígitos binarios.

El método es idéntico al anterior, es decir para la parte entera se divide la cantidad binaria de 4 en 4 dígitos de derecha a izquierda.

Para la parte fraccionaria también se divide la fracción binaria de 4 en 4 dígitos, pero de izquierda a derecha.

Ejemplo 10

Dada la siguiente cantidad en el sistema binario 1010110101, pasar a la correspondiente en el sistema hexadecimal.

$$\underline{1\ 0}\ \underline{1\ 0\ 1\ 1}\ \underline{0\ 1\ 0\ 1}$$

Si al dividir la cantidad en grupos de cuatro dígitos el grupo de la extrema izquierdo, en el caso de los enteros, resulta incompleto habrá que considerar como ceros el complemento.

De igual manera se deberá proceder cuando el grupo de extrema derecha, en el caso de las fracciones, resulta incompleto.

$$(0\ 0\ 1\ 0)_2 = 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 2$$

$$(1\ 0\ 1\ 1)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11\ (B)$$

$$(0\ 1\ 0\ 1)_2 = 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

De donde:

$$(0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1) = (2\ B\ 5)$$

Veamos ahora un ejemplo para la parte fraccionaria:

Ejemplo 11

$$0.\ \underline{1\ 0\ 1\ 0}\ \underline{1\ 1\ 0\ 1}\ \underline{0\ 1}$$

$$(1\ 0\ 1\ 0)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^0 + 0 \times 2^0 = 10\ (A)$$

$$(1\ 1\ 0\ 1)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13\ (D)$$

$$(0\ 1\ 0\ 0)_2 = 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$$

Por lo tanto:

$$(0.\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1)_2 = (0.\ A\ D\ 4)_{16}$$

Otro método para pasar de una cantidad binaria a otra octal es teniendo en cuenta la siguiente tabla, en la cual se observa que para cada dígito octal se tienen tres dígitos binarios.

TABLA DE CONVERSION DEL SISTEMA OCTAL AL BINARIO

Octal	Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Así del ejemplo 8 anterior, tenemos:

$$(\underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 1\ 1}_{7})_2 = (557)_8$$

y del ejemplo 9, tenemos:

$$(0.\ \underbrace{1\ 0\ 1}_{5.}\ \underbrace{0\ 1\ 1}_{3}\ \underbrace{0\ 1\ 1}_{3})_2 = (0.533)_8$$

En forma similar al método anterior, podemos emplear ahora la tabla mostrada a continuación para pasar de una cantidad binaria a una hexadecimal, considerando que por cada dígito hexadecimal se tienen cuatro dígitos binarios.

TABLA DE CONVERSION DEL SISTEMA HEXADECIMAL AL BINARIO

Hexadecimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Aplicando esta tabla al ejemplo 10 anterior, tenemos:

$$\underbrace{(1\ 0}_{2}\ \underbrace{1\ 0\ 1\ 1}_{B}\ \underbrace{0\ 1\ 0\ 1)_2}_{5} = (2\ B\ 5)_{16}$$

Del ejemplo 11

$$(0.\ \underbrace{1\ 0\ 1\ 0}_{A}\ \underbrace{1\ 1\ 0\ 1}_{D}\ \underbrace{0\ 1)_2}_{4} = (0.\ A\ D\ 4)_{16}$$

Si se tiene una cantidad en el sistema octal o en el hexadecimal y se desea pasar al decimal, se recomienda primero pasarlo a binario y de binario a decimal.

Ejemplo 12

Convertir el número 557 de base 8 a su correspondiente de base 10.

$$(5\ 5\ 7)_8 = (\underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 1\ 1)_2}_{7}$$

$$(1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1) = (1x2^8 + 0x2^7 + 1x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0)_{10}$$

$$= (256 + 64 + 32 + 8 + 4 + 2 + 1)_{10}$$

$$= (3\ 6\ 7)_{10}$$

Por lo tanto:

$$(5\ 5\ 7)_8 = (3\ 6\ 7)_{10}$$

Ejemplo 13

Convertir el número 2 B 5 de base 16 a su correspondiente de base 10.

$$(2\ B\ 5)_{16} = (\underbrace{1\ 0}_{2}\ \underbrace{1\ 0\ 1\ 1}_{B}\ \underbrace{0\ 1\ 0\ 1}_{5})_2$$

$$(1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1) = (1x2^9 + 0x2^8 + 1x2^7 + 0x2^6 + 1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0)_{10}$$

$$= (512 + 128 + 32 + 16 + 4 + 1)_{10}$$

$$(2\ B\ 5)_{16} = (6\ 9\ 3)_{10}$$

Como ejercicio y a modo de comprobación, hagamos el proceso inverso, es decir pasemos del sistema decimal al octal en un caso y de decimal a hexadecimal en el otro.

Ejemplo 14

Convertir el número 367 de base 10 a su correspondiente de base 8.

$$(3\ 6\ 7)_{10} = (?)_8$$

Primero lo pasamos a binario

$$(3\ 6\ 7)_{10} = (?)_2$$

$$\frac{367}{2} = 183 + \frac{1}{2} \quad \ldots\ldots X_0 = 1$$

$$\frac{183}{2} = 91 + \frac{1}{2} \quad \ldots\ldots X_1 = 1$$

$$\frac{91}{2} = 45 + \frac{1}{2} \quad \ldots\ldots X_2 = 1$$

$$\frac{45}{2} = 22 + \frac{1}{2} \quad \ldots\ldots X_3 = 1$$

$$\frac{22}{2} = 11 + \frac{0}{2} \quad \ldots\ldots X_4 = 0$$

$$\frac{11}{2} = 5 + \frac{1}{2} \quad \ldots\ldots X_5 = 1$$

$$\frac{5}{2} = 2 + \frac{1}{2} \quad \ldots\ldots X_6 = 1$$

$$\frac{2}{2} = 1 + \frac{0}{2} \quad \ldots\ldots X_7 = 0$$

$$\frac{1}{2} = 0 + \frac{1}{2} \quad \ldots\ldots X_8 = 1$$

Por lo tanto:

$$(3\ 6\ 7)_{10} = (1\ 0\ 1\ 1\ 0\ 1\ 1\ 1)_2$$

Recordando que cada dígito octal corresponde a tres dígitos binarios, agrupando de derecha a izquierda ya que la cantidad es entera y aplicando la tabla de conversión, tenemos:

$$(\underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 0\ 1}_{5}\ \underbrace{1\ 1\ 1}_{7})_2 = (5\ 5\ 7)_8$$

De aquí que:

$$(3\ 6\ 7)_{10} = (5\ 5\ 7)_8$$

Si aplicamos la ecuación 2.1

$$(3\ 6\ 7)_{10} = (?)_8$$

$$\frac{367}{8} = 45 + \frac{7}{8} \quad \ldots\ldots X_0 = 7$$

$$\frac{45}{8} = 5 + \frac{5}{8} \quad \ldots\ldots X_1 = 5$$

$$\frac{5}{8} = 0 + \frac{5}{8} \quad \ldots\ldots X_2 = 5$$

Por lo tanto:

$$(3\ 6\ 7)_{10} = (5\ 5\ 7)_8$$

Ejemplo 15

Convertir el número 693 de base 10 a su correspondiente de base 16.

$$(6\ 9\ 3)_{10} = (?)_{16}$$

Primero lo pasamos a binario:

$$(6\ 9\ 3)_{10} = (?)_2$$

$$\frac{693}{2} = 346 + \frac{1}{2} \quad \ldots\ldots X_0 = 1$$

$$\frac{346}{2} = 173 + \frac{0}{2} \quad \ldots\ldots X_1 = 0$$

$$\frac{173}{2} = 86 + \frac{1}{2} \quad \ldots\ldots X_2 = 1$$

$$\frac{86}{2} = 43 + \frac{0}{2} \quad \ldots\ldots X_3 = 0$$

$$\frac{43}{2} = 21 + \frac{1}{2} \quad \ldots\ldots X_4 = 1$$

$$\frac{21}{2} = 10 + \frac{1}{2} \quad \ldots\ldots X_5 = 1$$

$$\frac{10}{2} = 5 + \frac{0}{2} \quad \ldots\ldots X_6 = 0$$

$$\frac{5}{2} = 2 + \frac{1}{2} \quad \ldots\ldots X_7 = 1$$

$$\frac{2}{2} = 1 + \frac{0}{2} \quad \ldots\ldots X_8 = 0$$

$$\frac{1}{2} = 0 + \frac{1}{2} \quad \ldots\ldots X_9 = 1$$

Por lo tanto:

$$(6\ 9\ 3)_{10} = (\underbrace{1\ 0}_{2}\ \underbrace{1\ 0\ 1\ 1}_{B}\ \underbrace{0\ 1\ 0\ 1}_{5})_2$$

$$(6\ 9\ 3)_{10} = (2\ B\ 5)_{16}$$

Si aplicamos la ecuación 2.1

$$(6\ 9\ 3)_{10} = (?)_{16}$$

$$\frac{693}{16} = 43 + \frac{5}{6} \quad \ldots\ldots \quad X_0 = 5$$

$$\frac{43}{16} = 2 + \frac{11}{16} \quad \ldots\ldots \quad X_1 = 11\ (B)$$

$$\frac{2}{16} = 0 + \frac{1}{16} \quad \ldots\ldots \quad X_2 = 2$$

$$\therefore (6\ 9\ 3)_{10} = (2\ B\ 5)_{16}$$

2.2 PROBLEMAS DE PRECISION QUE SE ORIGINAN INTERNAMENTE

Existen tres clases de errores inevitables que ocurren en los procesos numéricos ejecutados con las computadoras. Estos son: los inherentes, los de truncamiento y los de redondeo.

Errores Inherentes

Los errores inherentes son errores en los datos de entrada y en las mismas constantes. Pueden ser debido a incertidumbre en la medición de los datos, o al hecho de que los datos no se pueden representar exactamente con los dígitos disponibles para representar el número.

Ejemplo

$$1/3 = 0.33333 \ldots\ldots$$
$$\pi = 3.1415926 \ldots\ldots$$
$$e = 2.7182818 \ldots\ldots$$

Errores de Truncamiento

Los errores de truncamiento o corte se producen cuando un proceso matemático infinito se interrumpe en un momento dado.

Ejemplo

El cálculo de la raíz cuadrada por el método de Newton.

$$X_{i+1} = \frac{1}{2}\left(X_i + \frac{N}{X_i}\right)$$

Errores de Redondeo

Cuando el resultado de una operación debe almacenarse con un número limitado de dígitos significativos, y este resultado tiene un número mayor de dígitos, deberá redondearse.

Ejemplo

$$5/3 = 1.666667$$

$$8/9 = 0.888889$$

2.3 CONFIGURACION DE SISTEMAS

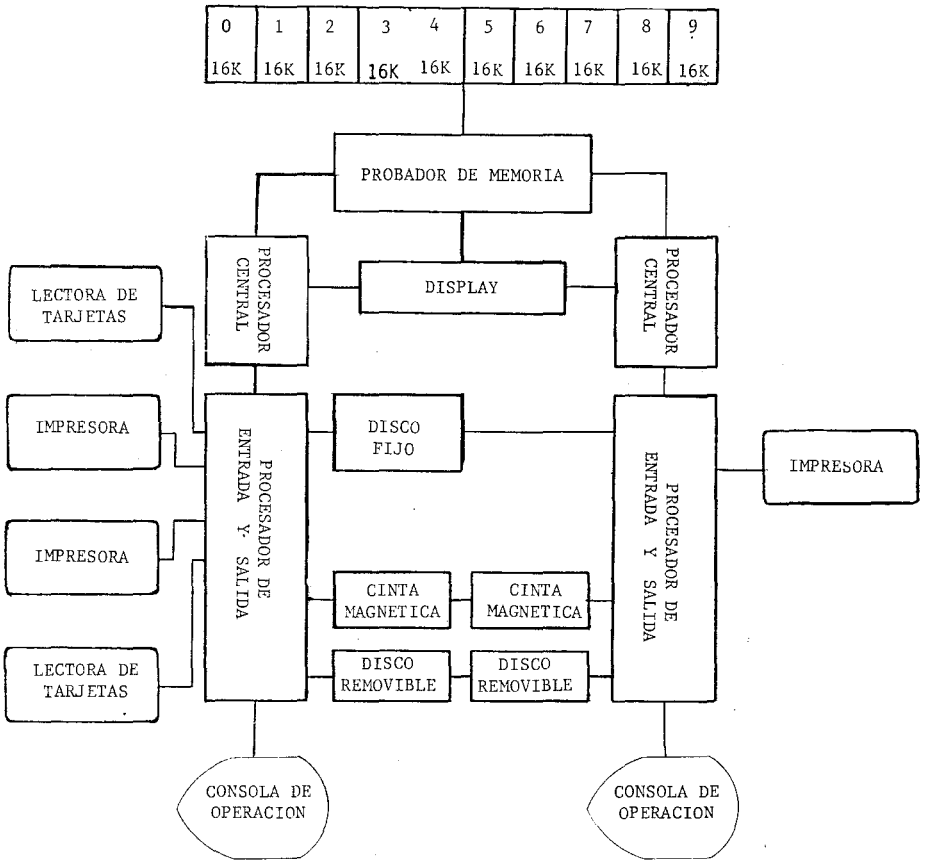
Lo que se pretende al hablar de configuración de un sistema de cómputo, es explicar qué componentes integran dicho sistema y si es posible representar en forma esquemática todas y cada una de dichas componentes y la interrelación entre ellas, mencionando las características.

Por lo tanto, en una configuración de un sistema deben aparecer descripciones tales como:

- Número de procesadores y características de los mismos.
- Capacidad total en memoria y características de la misma.
- Número de unidades de disco, cuántos son fijos y cuántos removibles así como la capacidad de cada uno de ellos.
- Número de unidades de cinta indicando el número de canales y densidad de grabación.
- Número de lectoras de tarjetas y velocidad de lectura de cada una de ellas.
- Número de impresoras y velocidad de impresión de cada una de ellas.
- Número de consolas de operación.
- Número de terminales de teleproceso.
- Descripción de cómo se integra la palabra, es decir cuántos bits.
- El código utilizado para representar datos o instrucciones en la computadora como por ejemplo: EBCDIC, ASCII, etc.

A continuación se presenta un ejemplo de la configuración de un sistema de cómputo.

MODULOS DE MEMORIA



Tiempo Compartido

Cuando en un sistema de cómputo, un usuario utiliza como medio de entrada y salida una terminal, se dice que está trabajando de manera interactiva con la computadora. Esto representa algunas ventajas ya que el usuario puede obtener resultados parciales y dados éstos, alimentar a la computadora con los datos adecuados o hacer que el proceso realice determinada acción sin necesidad de terminar el proceso y tenerlo que volver a reanudar.

Por supuesto las diferentes componentes de la computadora son mucho más rápidas que la capacidad de reacción del ser humano y por tanto ésta tiene oportunidad de atender a varios usuarios en lo que parece ser el mismo tiempo; los usuarios, por otra parte, tienen la impresión de ser cada uno de ellos el único en utilizar la computadora; esto es lo que se conoce como *tiempo compartido*.

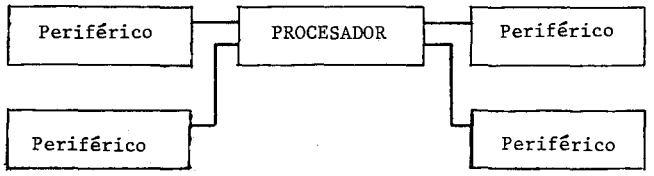
Equipo Periférico

Las componentes de toda computadora pueden ser clasificadas en dos categorías: equipo central y equipo periférico.

El equipo central consiste de procesador (CPU), memoria principal y procesador de entrada y salida (multiplexor). En caso de existir éste último, todo el resto del equipo que esté directamente conectado al equipo central, se considera como equipo periférico.

El equipo periférico puede, a su vez, ser de dos tipos: memoria auxiliar y entrada/salida. De entre la gran variedad de periféricos podemos destacar, del tipo de memoria auxiliar, unidades de disco magnético fijo y removible y unidades de cinta magnética. Estos mismos periféricos pueden actuar como unidades de entrada/salida así como los siguientes:

Lectora óptica de caracteres (OCR)
Lectora de caracteres magnéticos (MICR)
Terminal de rayos catódicos (CRT)
Lectora de tarjetas perforadas (PCR)
Graficadora (PLOTTER)
Lectora de cinta perforada (PTR)
Terminales punto-de-venta (POS)



Capacidad de Memoria

La memoria de una computadora está formada por circuitos integrados, midiéndose su capacidad por el número máximo de palabras que puede almacenar.

Una palabra de computadora (*WORD*) es una cadena de bits los cuales son considerados como una unidad.

En general, una palabra es la mínima unidad de almacenamiento en memoria que puede ser direccionable, es decir normalmente es posible leer o escribir todos sus bits al mismo tiempo. Asimismo una palabra tiene una dirección única en memoria, lo cual permite un acceso directo a ella. La longitud de la palabra viene dada por el número de bits que la componen, de esta forma, dependiendo de la instalación, se tienen palabras de 8, 16, 24, 48, 64, etc., bits.

De lo anterior, la capacidad de almacenamiento en una palabra está determinada por la longitud de la misma.

Es común que la capacidad de memoria de alguna computadora en particular se mencione en bytes en lugar de palabras. Un byte es un agrupamiento lógico de bits generalmente distinto de la palabra. Sera necesario considerar esta diferencia al tratar de comparar la capacidad de una máquina con respecto a otra.

2.4 VENTAJAS Y DESVENTAJAS DEL USO DE LOS DIFERENTES EQUIPOS DE COMPUTO

En muchas ocasiones cuando se utiliza un equipo de cómputo, éste se usa generalmente porque es el único con el que se dispone en el lugar donde se labora, o bien porque dada la urgencia de tener resultados no se lleva a cabo una reflexión, de si es ventajoso o no el emplear dicho equipo.

Muchos de los problemas que se presentan es posible resolverlos mediante otros equipos más pequeños como puede ser el uso de una máquina calculadora programable de bolsillo, un microprocesador, una minicomputadora, etc.

Por lo tanto, habría que ponderar en un momento dado qué tan ventajoso o desventajoso resulta el hecho de utilizar un sólo equipo de cómputo para cualquier tipo de problema.

Por lo anterior se puede pensar en que para decidir qué equipo conviene utilizar, hay que tomar en cuenta algunas de las siguientes consideraciones:

- Tiempo de espera de resultados.
- Memoria.
- Rapidez de cálculo y de obtención de resultados o tipo de salida.
- Forma de presentación de resultados.
- Cantidad de datos de entrada.
- Número de cálculos a efectuar.
- Rapidez en la programación.
- Precio.

BIBLIOGRAFIA

James A. Saxon
MATEMATICAS BASICAS PARA PROCESO DE DATOS
Editorial Prentice Hall Int.
Madrid, España 1973

Siegel Paul
COMPUTADORAS DIGITALES
Primera edición, cuarta impresión.
Editorial CECSA
México, 1975

Awad Elias M.
PROCESAMIENTO AUTOMATICO DE DATOS
Primera edición, segunda impresión
Editorial Diana
México, 1976

Arechiga G. Rafael
FUNDAMENTOS DE COMPUTACION
Segunda edición
Editorial Limusa
México, 1978

3. ALGORITMOS Y DIAGRAMAS DE FLUJO

3.1 CONCEPTO DE ALGORITMO

> Un algoritmo es un conjunto de acciones que determinan la secuencia de los pasos a seguir para resolver un problema específico.

Por las características del problema específico que se plantea, es posible distinguir dos tipos de algoritmo:

a) Algoritmo numérico

b) Algoritmo no numérico

Características que deben cumplir los algoritmos:

Finitud

El procedimiento para la solución de un problema dado, se debe determinar en un número finito de pasos. De aquí que no es posible establecer un número infinito de pasos para la solución de un problema propuesto, al construir el algoritmo.

Definición

Durante el desarrollo del algoritmo, los pasos deben estar definidos con precisión. Por lo tanto no deben existir especificaciones cuya interpretación sea ambigua y den origen a elegir una decisión que no es la deseada.

Entrada

Se considera como entrada el conjunto de datos o información requerida para resolver un problema dado. Esto es, cuando se establece un algoritmo para encontrar la solución a un problema, los datos deben cumplir con las características propias a dicho problema, por lo que no cualquier grupo de datos se puede considerar como entrada en el procedimiento señalado.

Salida

Recibe el nombre de salida, el resultado o conjunto de resultados que se obtienen al desarrollar un algoritmo utilizando los datos de entrada.

Efectividad

El desarrollo del algoritmo que se proponga nos debe conducir a la solución del problema planteado, es decir que al ejecutar o realizar los pasos señalados, el procedimiento nos conduzca al final del mismo a obtener el resultado buscado.

De lo anterior podemos pensar que cuando se propone un problema después de analizarlo y ver la conveniencia de utilizar la computadora para su solución, entonces habrá que visualizar dicho problema planteado de modo tal que sean definidas perfectamente todas las operaciones o decisiones, estableciendo una secuencia entre ellas que conduzca a los resultados esperados, es decir habrá que plantear el problema como un algoritmo o una serie de ellos.

3.2 CONCEPTO DE SECUENCIA

Entendemos por secuencia la ejecución o realización de pasos o acciones en el desarrollo de un algoritmo, siguiendo el orden establecido.

3.3 CONCEPTO DE PROGRAMA

Se le llama programa a la serie de instrucciones escritas en alguno de los lenguajes disponibles en la instalación de cómputo, por medio de las cuales se logra que la computadora realice todas las operaciones o decisiones señaladas en dichas instrucciones.

Podemos distinguir dos tipos de programa:

Programa Fuente

Generalmente recibe el nombre de programa fuente el conjunto de instrucciones escritas en algún lenguaje de computadora, las cuales han sido perforadas o transcritas para ser interpretadas por algún dispositivo de lectura de la computadora.

Programa Objeto

Recibe este nombre el conjunto de instrucciones que componen un programa fuente y que han sido traducidas al lenguaje de máquina por medio del compilador correspondiente.

3.4 CONSTRUCCION DE ÀLGORITMOS

Algoritmos no Numéricos

En algunas de nuestras actividades diarias, desarrollamos labores que requieren necesariamente se realicen siguiendo una secuencia de pasos bien definidos y reglamentados, lo cual cumple con las características de algoritmo.

Ejemplo 1

Supongamos que se requiere cambiar el neumático del coche que se va conduciendo y que éste dispone de la llanta de refacción y herramienta necesaria.

1. Detener el coche.
2. Bajar la herramienta y llanta de refacción.
3. Levantar el coche.
4. Desmontar el neumático sujeto de reemplazo.
5. Montar la llanta en el lugar correspondiente.
6. Bajar el coche.
7. Guardar la herramienta y el neumático dañado.
8. Continuar el viaje.

Ejemplo 2

Consideremos que se trata de trasladarse de la casa a la escuela, en autobús, suponiendo que éste pasa cerca de la casa y escuela.

Descripción de los pasos a seguir:

1. Verificar que se dispone del dinero suficiente para el transporte o no se hará el viaje.
2. Llegar al lugar donde pasan los autobuses.
3. Si es el autobús indicado abordarlo, si no es así continuar esperando que pase.
4. Pagar el pasaje, recogiendo el comprobante y conservarlo.
5. Si hay asientos vacíos ocupar alguno, si no viajar de pie.
6. Si está próximo el lugar de destino, aproximarse a la salida y tocar el timbre, en caso contrario, continuar el viaje.
7. Una vez detenido el autobús, bajarse y caminar hacia la escuela.

Algoritmos Numéricos

Son aquéllos que están orientados hacia problemas de ingeniería, científicos, etc., y en general en los que se vean involucrados cálculos matemáticos.

Ejemplo 1

Supongamos que se nos solicita que se establezca un procedimiento para convertir una cantidad decimal entera a su equivalente en binario.

Pensando que un algoritmo es un conjunto de reglas que determinan la secuencia de los pasos a seguir para la solución del problema propuesto, apliquemos este concepto para establecer el procedimiento solicitado y el método de divisiones sucesivas, para realizar los siguientes pasos:

1. A la cantidad de base 10 suministrada, dividirla entre la base de la cantidad a la cual se desea pasar, para este ejemplo será el número 2.

2. El residuo que resulte de la división, pasa a formar parte de la cantidad buscada, para este problema el residuo será uno o cero.

3. En el resultado que se obtiene al llevar a cabo la división, la parte entera es la nueva cantidad de base 10 suministrada.

4. Si la nueva cantidad de base 10 es cero, habrá concluido el algoritmo y se pasará al inciso quinto, en caso contrario, regresar al inciso primero.

5. Para formar la cantidad resultante con los residuos obtenidos en el procedimiento anterior, agrúpese en forma tal que el primer residuo obtenido ocupe el lugar de la extrema derecha y el último obtenido el de la extrema izquierda.

   Para verificar el algoritmo descrito pensemos que la cantidad suministrada de base 10 es el número 217 y se pide encontrar su equivalencia de base 2.

Tenemos:

$$(217)_{10} = (?)_2$$

$$\frac{217}{2} = 108 + \frac{1}{2} \qquad X_0 = 1$$

$$\frac{108}{2} = 54 + \frac{0}{2} \qquad X_1 = 0$$

$$\frac{54}{2} = 27 + \frac{0}{2} \qquad X_2 = 0$$

$$\frac{27}{2} = 13 + \frac{1}{2} \qquad X_3 = 1$$

$$\frac{13}{2} = 6 + \frac{1}{2} \qquad X_4 = 1$$

$$\frac{6}{2} = 3 + \frac{0}{2} \qquad X_5 = 0$$

$$\frac{3}{2} = 1 + \frac{1}{2} \qquad X_6 = 1$$

$$\frac{1}{2} = 0 + \frac{1}{2} \qquad X_7 = 1$$

La cantidad de base 2 buscada, será por lo tanto:

1 1 0 1 1 0 0 1

Entonces:

$$(217)_{10} = (11011001)_2$$

Ejemplo 2

Establecer un procedimiento para obtener la suma de todos los números enteros impares positivos desde el uno hasta un número impar proporcionado como dato.

Pasos a seguir:

1. Verificar que el número proporcionado sea impar, entero y positivo, si no lo es, dar por concluido el procedimiento.

2. Sumar el número proporcionado en una variable que llamaremos SUMA, la cual inicialmente deberá valer cero.

3. Restar al número proporcionado dos unidades, obteniendo un nuevo número proporcionado.

4. Si el nuevo número proporcionado es negativo, dar por concluido el procedimiento y pasar al inciso quinto, en caso contrario, regresar al inciso segundo.

5. Escribir el resultado obtenido, el cual estará alojado en la variable que llamamos SUMA.

Verifiquemos el algoritmo anterior mediante un ejemplo numérico, supongamos que el número proporcionado es el número 5, y la variable llamada SUMA vale cero.

$\frac{5}{2} = 2 + \frac{1}{2}$	1. Residuo y parte entera mayor que cero por lo tanto es número impar entero positivo.
SUMA = SUMA + 5	2. SUMA tiene ahora el valor 5
5 - 2 = 3	3. Tres, nuevo número proporcionado.
3 > 0	4. Como el nuevo número proporcionado es positivo repetimos el procedimiento desde el inciso segundo.
SUMA = SUMA + 3	2. Como SUMA tenía el valor 5, al sumarle el número 3, tendrá ahora el valor 8.
3 - 2 = 1	3. Uno, nuevo número proporcionado.
1 > 0	4. Como el nuevo número proporcionado es positivo repetimos nuevamente el procedimiento desde el inciso segundo.
SUMA = SUMA + 1	2. Como SUMA tenía el valor 8, al sumarle el número 1, tendrá el valor 9.
1 - 2 = - 1	3. Menos uno, nuevo número proporcionado.
-1 < 0	4. Como el nuevo número proporcionado es negativo, se ejecutará ahora el paso quinto.
Resultado = 9	5. El número 9 fue el último valor que quedó alojado en SUMA.

Ejemplo 3

Algoritmo de Euclides

Dados dos números naturales 'a' y 'b' encontrar su máximo común divisor.

Establezcamos el siguiente procedimiento por pasos, para resolver este sencillo problema.

1. Comparar los dos números del problema, 'a' y 'b'.
2. Si los números observados son iguales, cada uno de ellos es el resultado buscado y el proceso termina escribiendo el resultado.
3. Si el número 'a' es mayor que el número 'b' continuar en el paso quinto.
4. Intercambiar los valores de 'a' y 'b' de tal manera que 'a' tenga el valor de 'b' y consecuentemente 'b' el de 'a'.
5. Restar el número 'b' del número 'a' asignando como nuevo valor de 'a' el resultado de la resta, y regresar al paso primero.

3.5 DIAGRAMAS DE FLUJO Y SU SIMBOLOGIA

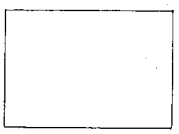
Un diagrama de flujo es la representación gráfica de un algoritmo, en el cual se contemplan los siguientes elementos:

1. Inicio.
2. Especificación de los datos de entrada.
3. Operaciones a realizar con los datos o decisiones a tomar.
4. Especificación de la salida (resultados).
5. Fin.

Hasta ahora no existen reglas o estándares que indiquen claramente la interpretación o uso que deba darse a todas las figuras geométricas que usualmente se utilizan para la elaboración de un diagrama de flujo.

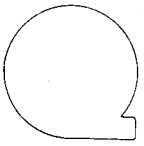
Sin embargo, las figuras geométricas más comúnmente utilizadas, así como su interpretación son las que a continuación se indican:

Esta figura en forma de óvalo se utiliza para indicar el inicio o fin de un procedimiento.

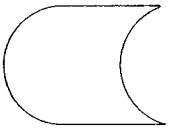


El rectángulo nos sirve para indicar cualquier operación que se tenga que realizar en el procedimiento.

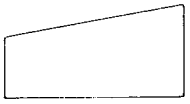
El rectángulo con un corte en el margen superior izquierdo indica una operación de entrada o salida a través de tarjetas perforadas.



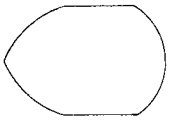
El círculo se emplea para representar una cinta magnética e indica entrada de datos almacenados en ella o bien salida que quedará grabada en dicha cinta.



Este símbolo se utiliza para representar un disco magnético, e indica entrada de datos almacenados en él o bien salida que quedará grabada en dicho disco.

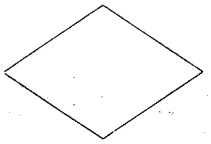


Este símbolo es empleado para representar transmisión de información a través de un teletipo.

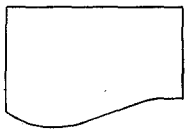


Este otro representa una terminal de video.

Un hexágono se usa para indicar el inicio y el fin de un proceso iterativo.



Un rombo se utiliza para indicar una decisión, es decir elegir una alternativa entre dos o tres que se presenten.



Este símbolo se utiliza para denotar que los resultados en una operación de salida aparecerán en hoja impresa (listado).

Este pequeño círculo, llamado conector se utiliza para indicar cambios en la secuencia del procedimiento.

Este pequeño símbolo, también llamado conector, se utiliza para indicar la continuación de un diagrama de hoja a hoja.

Las flechas se utilizan para indicar hacia dónde se dirige el flujo del proceso.

Ejemplo 1

Calcular:

$$y = 2x + 3 \qquad \text{si } x > 2$$
$$y = 4x - 2 \qquad \text{si } x \leq 2$$

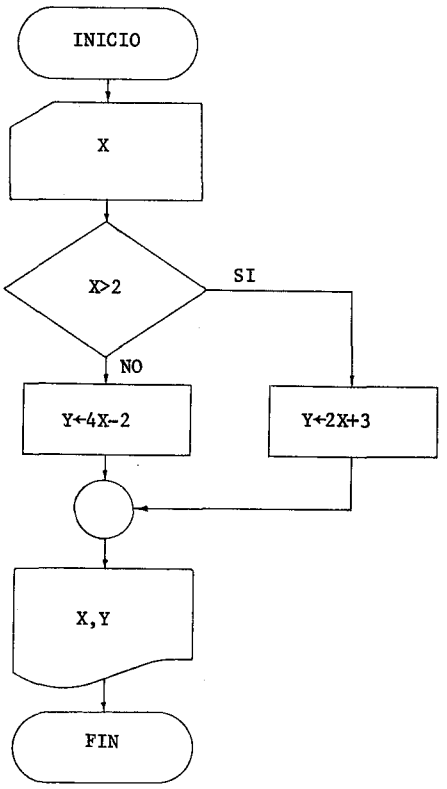
El procedimiento a seguir lo podemos indicar mediante los siguientes pasos:

1. Conocer el valor de la variable x.
2. Comparar el valor de la variable x contra la constante 2.
3. Si el valor de la variable x resulta ser mayor que la constante 2, calcular $y = 2x + 3$ y pasar al inciso 4, en caso contrario:

   Calcular: $y = 4x - 2$ continuar.

4. Escribir el valor de x así como el de y.
5. Terminar.

Si este mismo problema ahora lo tratamos en cuanto a pasos a seguir mediante métodos gráficos, es decir, si elaboramos un diagrama de flujo tenemos:



Como se puede observar, mediante el diagrama de flujo se expresa en forma gráfica la secuencia de pasos ordenados que se deben realizar para obtener la solución al problema planteado.

Por lo tanto, un diagrama de flujo es una importante herramienta en la programación, ya que permite al programador planear la secuencia de operaciones en un programa antes de escribirlo.

En un diagrama de flujo se debe tener cuidado de expresar en forma clara todas y cada una de las operaciones y/o decisiones a realizar, con el fin de que cualquier persona, aun ajena a la programación pueda interpretarlo.

Podemos señalar como ventajas al utilizar un diagrama de flujo, las siguientes:

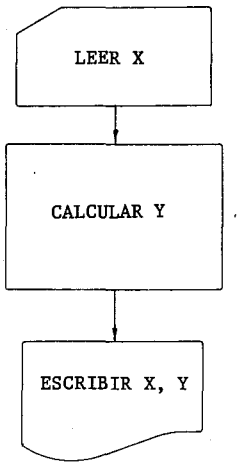
- Permite en forma gráfica planear las operaciones y/o decisiones de un programa antes de escribirlo.

- Representa ayuda visual para observar las correlaciones que se presentan entre operaciones o decisiones de un programa.
- Facilita la interpretación.
- Ayuda a la comunicación entre programadores.
- Forma parte de la documentación de un programa.
- Es independiente del lenguaje de programación a emplear.

En algunos problemas elaborados, muchas veces la programación de los mismos, se realiza por más de un programador, motivo por el cual el diagrama de flujo se presenta como ayuda importante en la planeación de la programación, permitiendo que en su realización intervengan varias personas.

Cabe señalar que en muchas ocasiones se utiliza el término diagrama de bloque como sinónimo de diagrama de flujo, sin embargo, por lo general la acepción de diagrama de bloque es utilizada para indicar operaciones o decisiones agrupándolas por objetivos.

Al hacer la representación del ejemplo anterior como diagrama de bloques, se tiene:



3.6 CONCEPTO DE ITERACION

En algunos procesos es necesario llevar a cabo varias veces una misma acción o serie de acciones antes de continuar adelante. Esta acción repetitiva recibe el nombre de iteración.

Así por ejemplo, supongamos que se nos pide elaborar un procedimiento para obtener la suma total de una serie de datos que se nos proporcionen. Uno de los pasos dentro del procedimiento que se señale, podría ser el de sumar el primer dato con el segundo obteniendo un resultado, después a ese resultado obtenido sumarle el tercer dato arrojando un nuevo resultado, repitiendo tantas veces la acción de sumar como datos se suministren, es esta repetición la que recibe el nombre de iteración.

3.7 CONCEPTO DE ARREGLO

Existen algunos problemas para los cuales es deseable que la información que se proporciona pueda ser identificada mediante una sola variable, esto es factible si a dicha variable se le asocia un número llamado índice.

Así por ejemplo, si tenemos una lista de cinco valores y queremos identificarlos mediante una sola variable, esto lo logramos asignándole a dicha variable cinco índices.

Sean los siguientes cinco números:

325; 8; 24; 3; 97

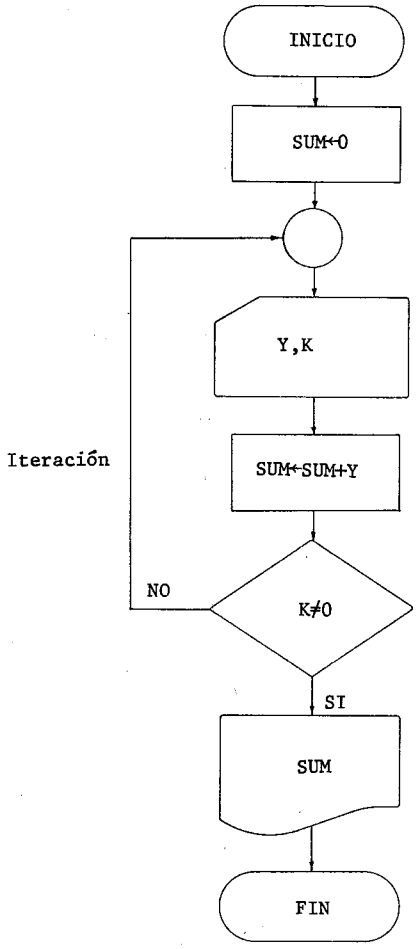
En la lista de valores deseamos identificarlos con una sola variable, pensemos que sea la variable Y, entonces habrá que asignarle a ésta, cinco índices, así tenemos:

```
Y(1) contiene el valor de 325

Y(2) contiene el valor de   8

Y(3) contiene el valor de  24

Y(4) contiene el valor de   3

Y(5) contiene el valor de  97
```

> Un arreglo es un conjunto de elementos los cuales dependen del nombre del arreglo y de un índice.

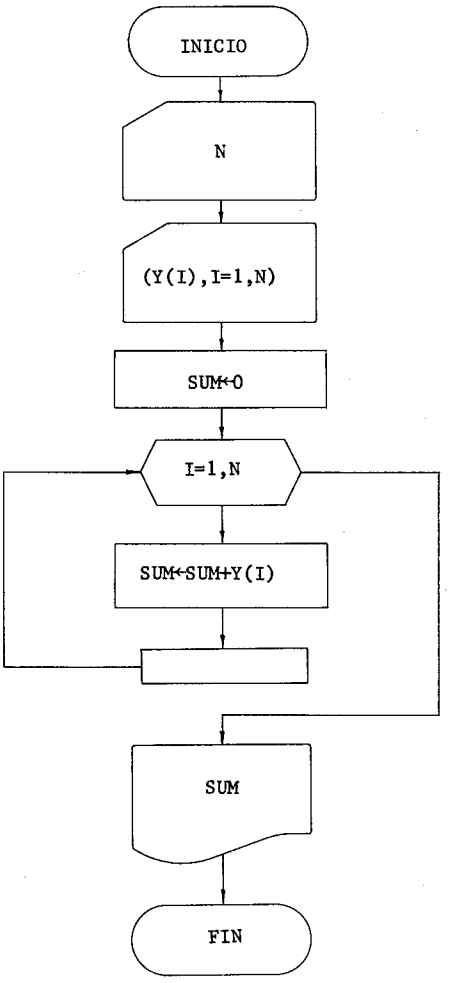
Es recomendable usar arreglos para facilitar la programación de aquellos problemas en los que en el proceso de solución se presentan iteraciones en las que se ven involucrados datos, para los cuales es posible realizar un agrupamiento.

Como un ejemplo para ilustrar en forma gráfica el concepto de iteración, elaboremos el diagrama de flujo, que corresponda a la suma de n números interesando el valor total de dicha suma:



En este diagrama de flujo la variable K, se utiliza para indicar el fin de los datos. Para el último valor de la variable Y, la variable K deberá tener algún valor positivo o negativo.

Elaborando el diagrama de flujo para el mismo ejemplo anterior, pero utilizando el concepto de arreglo, tenemos:



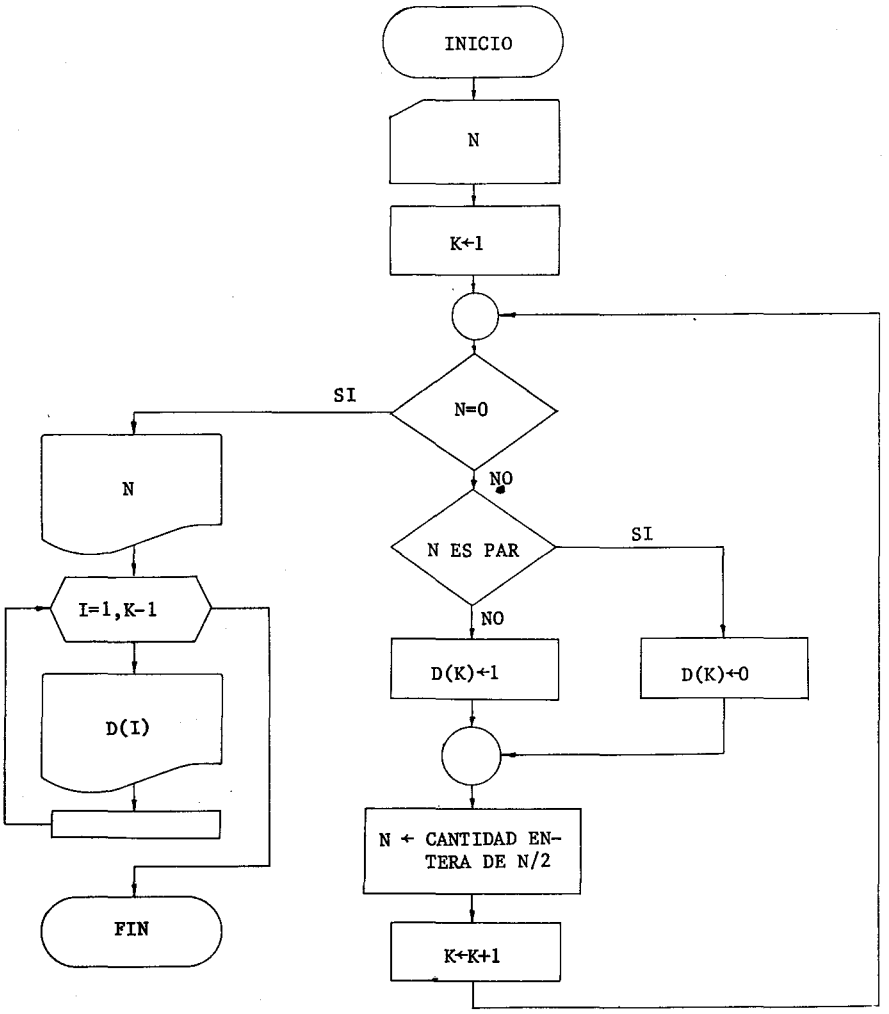
En este diagrama de flujo tenemos el arreglo Y_i y la variable N, con la cual se denota el número total de datos que estarán comprendidos en el arreglo Y_i.

3.8 EJEMPLOS SOBRE DIAGRAMAS DE FLUJO

Ejemplo 1

Del ejemplo uno de algoritmos numéricos, en el cual se solicitó se estableciera un procedimiento para convertir una cantidad decimal entera a su equivalente en binario.

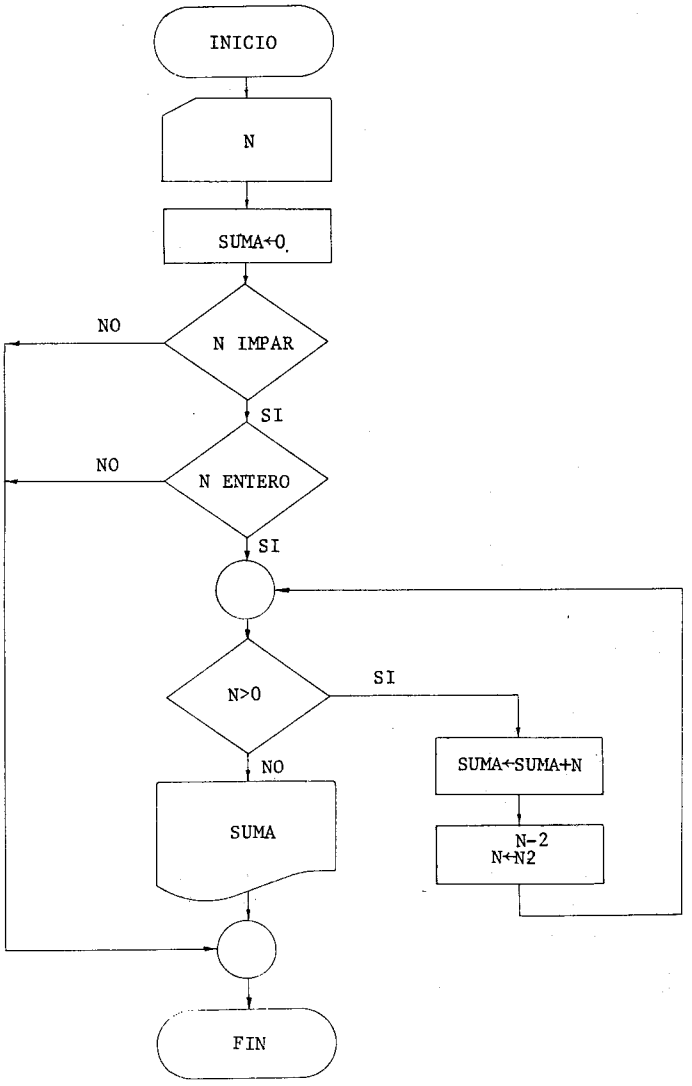
Sea N la cantidad proporcionada en base 10 y D(I) el equivalente binario.



Ejemplo 2

Del ejemplo número dos correspondiente a algoritmos numéricos en el que se pidió se estableciera un procedimiento para obtener la suma de todos los números enteros impares positivos desde uno hasta el número impar proporcionado.

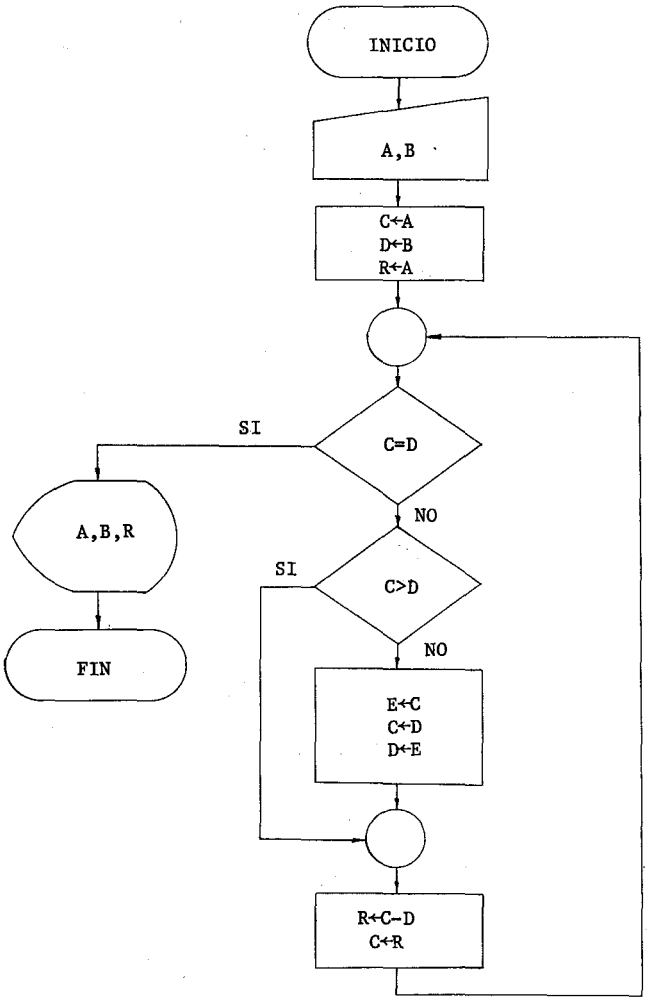
Sea N el número dado y SUMA el resultado pedido.



Ejemplo 3

Del ejemplo número tres de algoritmos numéricos en el cual se pide encontrar el máximo común divisor para dos números naturales dados.

Sean A y B los números naturales proporcionados y R su máximo común divisor.



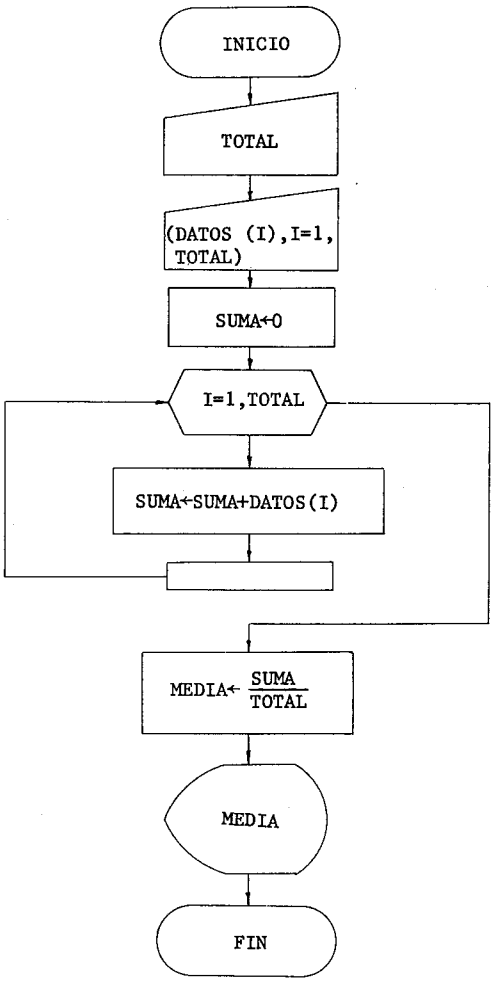
Ejemplo 4

Dado un conjunto de datos numéricos, construir el diagrama de flujo para obtener el promedio de los mismos.

Sea DATOS la variable utilizada como arreglo que identifica al conjunto de datos proporcionados.

SUMA la variable que almacenará la suma de todos los datos.

TOTAL la variable que contiene el número de datos.



Ejemplo 5

Dado un conjunto de datos y el número total de ellos se pide:

a) Establecer un procedimiento para obtener la desviación estandar indicando los pasos a seguir.

b) Construir el diagrama de flujo correspondiente al inciso anterior.

- Algoritmo:

1. Disponer de todos los datos y del número total de ellos.

2. Sumar todos los datos proporcionados, guardando el resultado de la suma en la variable R.

3. Dividir R entre el número total de datos almacenando dicho resultado en la variable M.

4. Restar cada uno de los datos proporcionados a la variable M, guardando el resultado obtenido en cada resta en la variable D, obteniendo por lo tanto tantas D como datos suministrados.

5. Elevar al cuadrado cada una de las D obtenidas, almacenando los valores resultantes en la variable D nuevamente.

6. Sumar todas las D obtenidas, guardando el resultado en la variable SU.

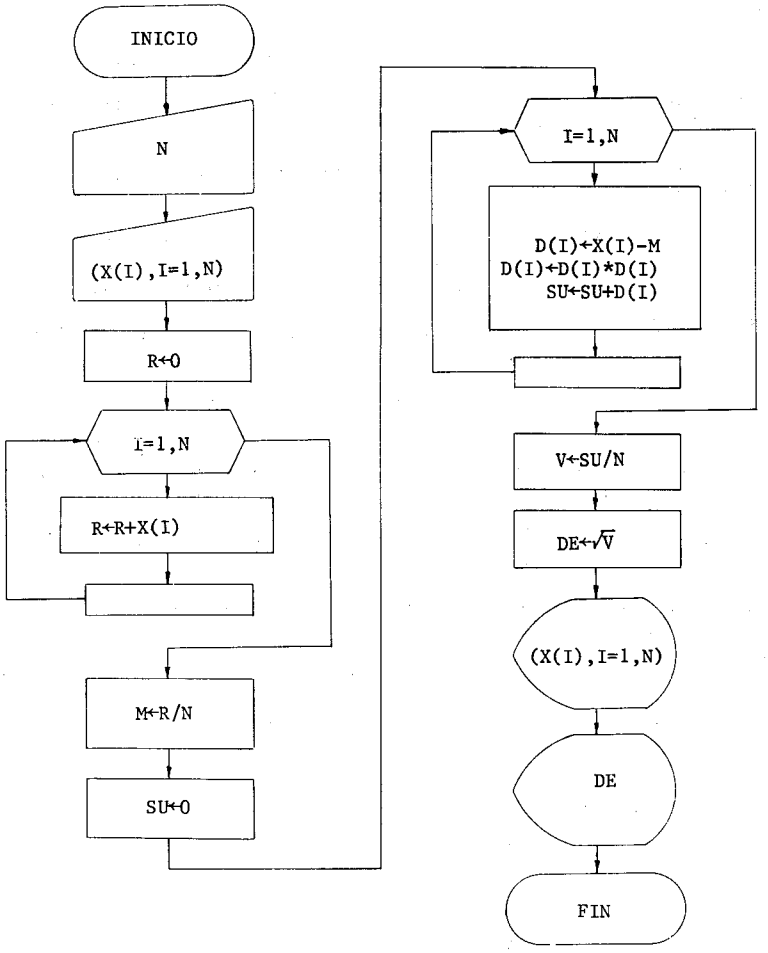
7. Dividir SU entre el número total de datos, guardando el resultado en la variable V.

8. Extraer la raíz cuadrada a V, almacenando el resultado en la variable DE.

9. Escribir los datos proporcionados y el resultado que se encuentra almacenado en la variable DE.

- Diagrama de flujo:

Sea X(I) el arreglo que contiene al conjunto de datos y N la variable que guarda el número total de ellos.



BIBLIOGRAFIA

Forsythe, Keenan, Organick, Stenberg
LENGUAJES DE DIAGRAMAS DE FLUJO
Primera edición, cuarta reimpresión
Editorial Limusa
México, 1973

Awad Elias M.
PROCESAMIENTO AUTOMATICO DE DATOS
Primera edición, segunda impresión
Editorial Diana
México, 1976

Arechiga G. Rafael
FUNDAMENTOS DE COMPUTACION
Segunda edición
Editorial Limusa
México, 1978

Mario V. Farina
DIAGRAMAS DE FLUJO
Editorial Diana
México, 1979

4. LENGUAJE FORTRAN

4.1 CONSTANTES Y VARIABLES

En la actualidad FORTRAN(*Fórmula Translation*), es uno de los lenguajes de alto nivel más utilizados en la solución de problemas científicos y de ingeniería.

Los elementos de programación FORTRAN y en general de cualquier otro superlenguaje, son los siguientes:

1. Constantes
2. Variables
3. Funciones

Constantes

Las constantes son cantidades numéricas o alfanuméricas (cadenas de caracteres) que no cambian su valor durante el proceso.

- Constantes
  - Numéricas
    - Enteras
    - Reales
      - Sin exponente
      - Con exponente
  - Alfanuméricas

Las constantes enteras son cantidades numéricas sin punto decimal.

Ejemplo 1

-2345	9075
+727	-86

El número máximo para una constante entera varía de computadora a computadora.

Ejemplo 2

En una computadora el rango es de -32768 a +32767 y en otra es de ±549755813887.

En este tipo de constantes es válido omitir el signo +.

Las constantes reales sin exponente son cantidades con punto decimal.

Ejemplo 3

32.5, -0.17, +355571.3, .12, etc.

Las constantes reales con exponente tienen el significado de números multiplicados por una potencia de 10.

Ejemplo 4

$0.314159x10^{1}$	0.314159E01
$0.15x10^{3}$	0.15E+03
$-16.5342x10^{-5}$	-16.5342E-5

La forma general de estas últimas es:

a E x

Donde a es una constante real sin exponente y x es una constante entera (con o sin signo).

En este tipo de constantes no son válidas letras, comas o dobles signos a excepción de un punto decimal y de la letra E (notación exponencial).

El mayor valor depende también de la computadora que se maneje.

Ejemplo 5

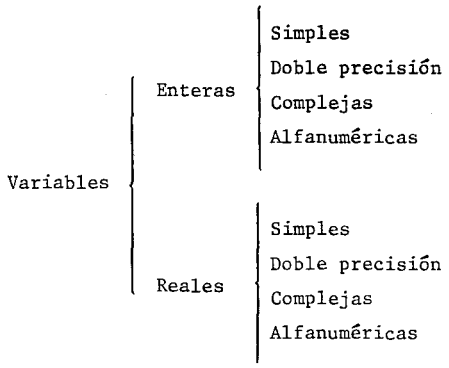
En una computadora varía de $-0.17x10^{-39}$ a $0.17x10^{39}$ en otra de $8.7581154021x10^{-47}$ a $4.3135914667x10^{68}$ etc..

Las constantes de tipo alfanumérico son aquellas cadenas de combinaciones de números, letras y caracteres especiales encerrados entre comillas.

Variables

Las variables son aquellos nombres alfanuméricos (formados por una letra y varios caracteres alfabéticos o numéricos), que tomarán distintos valores en el proceso.

La longitud de una variable (su tamaño en caracteres), varía de computadora a computadora, por lo que se recomienda limitarla a 5 caracteres.



Entera Simple

Sólo recibe valores enteros y debe empezar su nombre con alguna de las letras: I, J, K, L, M, N:

Ejemplo

NUM	MAS
I	N
K	NI

Real Simple

Es aquélla que recibe valores reales o que se le asignarán valores que deseamos tengan punto decimal. Su nombre puede empezar con cualquiera de las letras de la A a la H y de la O a la Z.

Ejemplo

X	RADIO	A1
Y	AREA	B2
Z	SIMP	F10

En las variables no es válido incluir caracteres especiales como /, +, %, etc..

De Doble Precisión

Una variable entera o una real puede admitir valores numéricos de doble precisión (se almacenan en dos palabras de computadora), siempre y cuando se incluya una declaración DOUBLE PRECISION al principio del programa. La escritura para las constantes de doble precisión es la misma que se explicó con anterioridad, con excepción de las constantes reales con exponente en las cuales se sustituye la letra E por una letra D.

Complejas

Una variable entera o una variable real puede recibir valores numéricos complejos (del tipo a+bi), siempre y cuando se incluya una declaración COMPLEX al principio del programa.

Alfanuméricas

Una variable entera o una variable real puede adquirir valores alfanuméricos (letras y símbolos) mediante lectura o asignación.

4.2 OPERADORES Y SUS PRIORIDADES

Las operaciones básicas en FORTRAN son: suma, resta, multiplicación, división y exponenciación; sus símbolos son los siguientes:

Exponenciación	**
Multiplicación	*
División	/
Suma	+
Resta	-

La jerarquía de las operaciones es, primero exponenciaciones, luego multiplicaciones y/o divisiones y finalmente sumas y restas. En caso de existir en una expresión varias operaciones de igual jerarquía, se ejecutarán en el orden que aparecen en la expresión, de izquierda a derecha; excepto en el caso de varias exponenciaciones que son ejecutadas de derecha a izquierda.

Es posible modificar dicha jerarquía con el uso de paréntesis, primero se realizan las operaciones dentro del paréntesis más interno y así sucesivamente y siempre de izquierda a derecha.

Estos símbolos nos permiten operar tanto con constantes como con variables.

Ejemplo

ARITMETICA ENTERA

3/5 produce como resultado 0

5/3 produce como resultado 1

I/K produce como resultado un entero

ARITMETICA REAL

3.0/5.0 produce como resultado 0.6

5./3. produce como resultado 1.66666

A/B produce como resultado un valor real

4.3 EXPRESIONES Y FUNCIONES ARITMETICAS

Expresiones

Una expresión aritmética es una relación matemática entre constantes y variables, constantes y funciones o únicamente constantes o únicamente variables o únicamente funciones, por medio de los operadores aritméticos y el uso de paréntesis.

Ejemplo 1

Expresión matemática	Expresión FORTRAN
$1.5 + x$	1.5+X
$1.5x$	1.5*X
$\frac{1.5}{x}$	1.5/X
$a + r^{-2}$	A+R**(-2)
$x^2 + y^2$	X**2+Y**2
$x^5 - y^{-2}$	X**5-Y**(-2)
$a + \frac{b}{c} + d$	A+B/C+D
$\frac{(a + b)}{c} + d$	(A+B)/C+D
$a + \frac{b}{c + d}$	A+B/(C+D)
$\frac{a + b}{c + d}$	(A+B)/(C+D)

El uso de paréntesis cambia notablemente el significado de las expresiones en caso de que no sea empleado correctamente.

En la escritura de expresiones en FORTRAN no es válido emplear dos operadores juntos.

Ejemplo 2

A**-2 produce error y es necesario emplear paréntesis: A**(-2).

En el caso de exponentes se recomienda que éstos sean enteros (una expresión como X**2.0 se evalúa por series mientras que X**2 se calcula como producto X*X).

Funciones

Las funciones son subprogramas que forman parte del lenguaje, de manera que basta únicamente hacer referencia a la función deseada dentro del programa para que la computadora ejecute la serie de instrucciones correspondientes a dicha función.

La parte que se encuentra encerrada dentro del paréntesis recibe el nombre de argumento, el cual puede ser una constante o variable numérica, o bien una expresión matemática.

El valor del argumento para las funciones trigonométricas tales como seno, coseno, etc. se debe proporcionar en radianes.

DESCRIPCION	FUNCION
Seno trigonométrico de X	SIN(X)
Coseno trigonométrico de X	COS(X)
Angulo cuya tangente es X	ATAN(X)
Valor absoluto de X	ABS(X)
Exponencial de X	EXP(X)
Logaritmo natural de X	ALOG(X)
Raíz cuadrada de X	SQRT(X)
Tangente de X	TAN(X)
Cotangente de X	COT(X)
Logaritmo en base 10 de X	ALOG10(X)
Tangente hiperbólica de X	TANH(X)
Conversión de tipo real a entero	IFIX(X)
Conversión de tipo entero a real	FLOAT(I)
Generador de números pseudoaleatorios	RANDOM(X)

En todos estos casos X es una variable real o una constante real o una expresión aritmética real e I es una variable entera o una expresión entera.

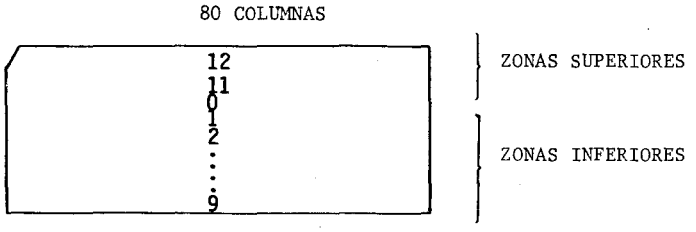
4.4 CONCEPTO DE CODIFICACION

Codificar significa traducir los pasos indicados en un algoritmo o diagrama de flujo a un lenguaje de programación, respetando las reglas de éste.

Una vez codificado en algún lenguaje, por ejemplo FORTRAN, es necesario suministrárselo a la máquina por alguna unidad de entrada.

En algunas computadoras se utiliza principalmente como unidad de entrada la lectora de tarjetas.

Descripción de una tarjeta



Las instrucciones se perforan en dichas tarjetas de acuerdo a un código, (código Hollerith), los números se representan mediante una sola perforación.

0 Se tiene una perforación en zona 0
1 Se tiene una perforación en zona 1
2 Se tiene una perforación en zona 2
:
9 Se tiene una perforación en zona 9

Las letras, por lo general, son las que presentan dos perforaciones en la misma columna.

A perforaciones en 12 y 1
B perforaciones en 12 y 2
C perforaciones en 12 y 3
:
Z perforaciones en 0 y 9

Existen además 10 caracteres especiales como +, -, *, (,), =, etc., para éstos el código marca una o dos o tres perforaciones por columna.

Ejemplo

\- (menos) perforación en zona 11
/ perforación en 0 y 1
* perforación en 11, 4 y 8

Se sugiere perforar una tarjeta e identificar el código completo.

En una tarjeta de un programa fuente se emplean las columnas como sigue:

Columna 1 Cuando se perfora una letra C, indica que esa tarjeta contiene un comentario.

Columnas 1 a 5 Para perforar números de proposiciones o de formatos.

Columna 6 Cuando la proposición o instrucción no se alcanzó a perforar en una tarjeta, entonces se usa una tarjeta adicional en la cual se perfora un carácter distinto de cero, con lo que se indica que es continuación de la instrucción anterior.

Columnas 7 a 72 Para perforar la instrucción o proposición en FORTRAN.

Las columnas 73 a 80 no son procesadas, por lo que su uso depende del programador (seriar tarjetas, poner ciertos caracteres para visualizar instrucciones, etc.).

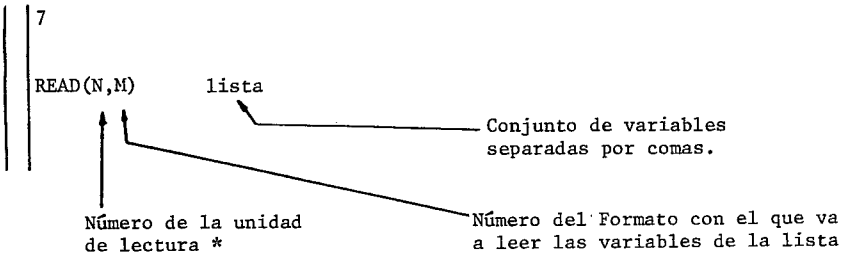
Las tarjetas que corresponden a la parte de datos de un programa emplean las 80 columnas.

Generalmente todas las tarjetas son perforadas tomando las instrucciones señaladas en una hoja de codificación. Una hoja de codificación reúne, por lo general, un total de 25 tarjetas.

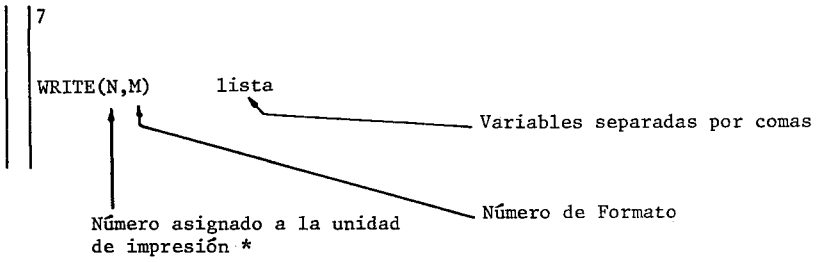
4.5 INSTRUCCIONES DE ENTRADA Y SALIDA

Instrucciones READ y WRITE

Forma general de una lectura



Forma general de una salida por impresora.



* Debe asignarse una constante entera la cual varía de computadora a computadora.

Instrucciones INPUT y PRINT

Las instrucciones INPUT y PRINT son casos particulares de READ y WRITE y se utilizan para transferir información desde o hacia un teletipo.

Al estar siempre en relación con el uso de teletipos, las instrucciones INPUT y PRINT no necesitan una lista de control como en el caso de READ y WRITE. Asimismo, el formato puede ser el predeterminado por el compilador FORTRAN.

Ejemplo 1

```
10 INPUT,A,B
80 PRINT,A,B
```

Ejemplo 2

```
10 INPUT,X
20 PRINT,'EL VALOR DE X ES',X
```

Declaración FORMAT

Los formatos se identifican con un número arbitrario positivo menor o igual a 99999. La información que contienen indicará a la computadora la forma en cómo va a leer o a imprimir los resultados. En el caso de lectura indica mediante especificaciones de campo, las columnas en las cuales se encuentran los valores de las variables a leer.

Especificaciones de Campo

Las especificaciones de campo le indican a la computadora los campos (conjunto de columnas), en los que se deberán leer o escribir los valores de las variables, (en el caso de lectura se cuenta con 80 columnas por tarjeta; en el caso de escritura se cuenta en general con 132 columnas, además del control de carro).

Especificaciones para Lectura o Escritura de Valores de Variables Enteras

La forma general de lectura o escritura de valores de variables enteras es:

In

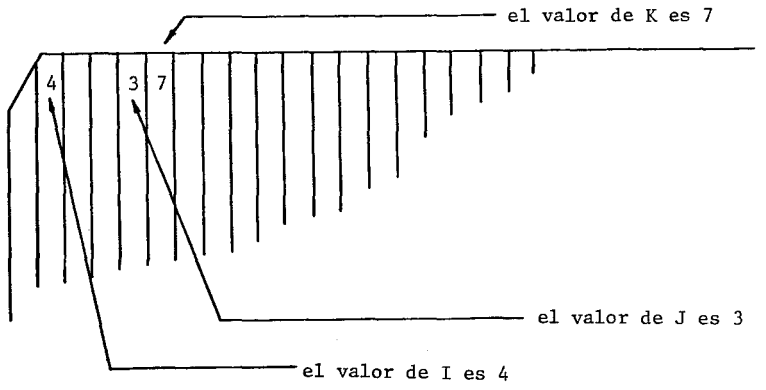
n indica el total de columnas del campo.

Ejemplo

```
      READ(2,10)I,J,K

10    FORMAT(I2,I3,I1)
```

Indica que el valor de la variable I se encuentra en las primeras dos columnas de la tarjeta, el valor de J en las columnas tres, cuatro y cinco, y el valor de K en la columna seis; nótese que la variable I no tiene relación en su nombre, con la definición de la especificación In.

Los valores enteros siempre deben indicarse hasta la extrema derecha del campo. Esto es:



Especificación para Lectura o Escritura de Valores de Variables Reales

La forma general de lectura o escritura de valores de variables reales es:

Fw.d

w es el total de columnas del campo y d el número posible de decimales dentro de ese campo.

Ejemplo 1

```
      |7
      |READ(2,20)A
20    |FORMAT(F4.2)
```

Indica que el valor de A se encuentra en las primeras cuatro columnas de la tarjeta y que es posible que tenga dos decimales.

Cuando se trata de escribir resultados, es conveniente separarlos entre sí; esto se logra empleando la especificación nX, en la que n es un número entero que indica el total de columnas de separación.

Para imprimir letreros es necesario indicarlos entre apóstrofes o mediante especificaciones de campo Hollerith.

Ejemplo 2

```
      7
      WRITE(6,100) Z
100   FORMAT(10X, 'EL VALOR DE Z ES',
     *2X,F10.4 )
      WRITE(6,200)Z
200   FORMAT(10X,16HEL VALOR DE Z ES,
     *2X,F10.4)
```

Ambos formatos son equivalentes, el segundo de ellos incluye la especificación Hollerith que indica el total de espacios que deben escribirse 16 H, estos espacios se cuentan inmediatamente después de la H.

Al imprimir resultados es necesario incluir el control de carro como primera instrucción del formato:

1H1	ó	'1'	para ubicar el carro de impresión en la parte alta de la siguiente hoja.
1H0	ó	'0'	salta una línea antes de escribir (espacio doble).
1H+	ó	'+'	evita el salto a la siguiente línea (reimprime sobre la línea anterior).
1H	ó	' '	para imprimir a renglón seguido.

También es válido emplear nX al principio. Esto equivale a la última instrucción para el control de carro (n debe ser un entero positivo).

Veámos un último ejemplo:

```
      7
      WRITE(6,200)A,B,Z
      FORMAT(1H0,'A=',F10.4,2X,
     *'B=',F8.2,3X,F12.4)
```

68

Especificación para Lectura o Escritura de Valores de Variables Alfanuméricas .

La forma general de lectura o escritura de valores alfanuméricos es:

```
mAn
```

m número entero positivo que indica el total de campos de tamaño n que se tiene en el formato.

Ejemplo

```
      |7
      |READ(5,500)A,B,C,D,E
500   |FORMAT(5A4)
      |WRITE(6,800)A,B,C,D,E
800   |FORMAT(1HO,5A4)
```

Veamos como ejemplo, la lectura y escritura de la siguiente tarjeta.

```
1                   19
DATOS ALFANUMERICOS
```

```
      |7
      |READ(5,100)T1,T2,T3,T4,T5
100   |FORMAT(4A4,A3)
      |WRITE(6,200)T1,T2,T3,T4,T5
200   |FORMAT(1HO,4A4,A3)
```

El programa anterior para una computadora, emplea cuatro especificaciones alfanuméricas de cuatro columnas de ancho y una de tres columnas. El mismo programa para otra:

```
      |7
      |READ(5,100)T1,T2,T3,T4
100   |FORMAT(3A6,A1)
      |WRITE(6,200)T1,T2,T3,T4
200   |FORMAT(1HO,3A6,A1)
```

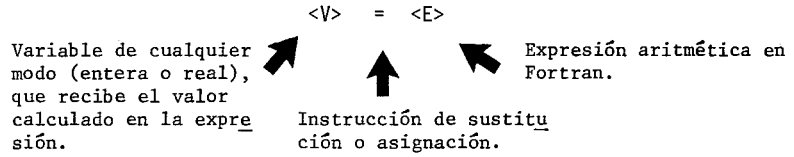
Por supuesto en este último, el propósito es minimizar el uso de memoria, ya que de antemano sabemos que el primer programa también funciona en otras computadoras con las respectivas tarjetas de control.

4.6 INSTRUCCIONES DE ASIGNACION Y DE FIN DE PROGRAMA

Instrucción de asignación

En FORTRAN una instrucción se evalúa de izquierda a derecha, ejecutándose primero las operaciones agrupadas por paréntesis y observando rigurosamente las prioridades en los operadores.

El resultado de la expresión generalmente se almacena en alguna variable para un uso posterior. La forma general de una instrucción de asignación es la siguiente:



Por lo tanto <V>=<E> es una proposición de asignación aritmética, es decir FORTRAN realiza primero la expresión a la derecha del signo igual y el valor obtenido lo asigna a la variable que está a la izquierda de dicho signo.

Instrucción de sustitución	Valor asignado a la variable
A = 3.5+SIN(0.)	3.5
B = 2*3/4	1.0
C = 2*(3/4)	0.0
D = 3.**2	9.0
E = 3**2.0	8.9999
F = 1/3+SIN(3.141592)**2	1.0
G = (6.0**2+3**2)**2	2025
H = SIN(0.0)**2+COS(0.0)**2	1.0
I = 3.5+SIN(0.0)	3
J = 2*3/4	1
K = 2*(3/4)	0
L = 3.0**2.0	8
M = 2./7.+3./5.+4./3.	2

Instrucción de Fin de Programa

La instrucción de fin de programa en FORTRAN es la instrucción END.

4.7 INSTRUCCIONES DE TRANSFERENCIA DE CONTROL

GO TO Simple

La instrucción de transferencia de control GO TO n donde n es un número entero positivo, al ejecutarse altera la secuencia de ejecución del programa desviándola a la instrucción numerada con n.

Ejemplo 1

```
      7
      .
      .
      .
      GO TO 98
```

GO TO Calculado

El GO TO calculado transfiere el control a dos o más instrucciones dependiendo del valor que tenga una variable entera.

Su forma general es:

```
      7
      .
      .
      .
      GO TO(n1,n2,n3,..np),I
```

Los números encerrados entre paréntesis y separados por comas, corresponden a números de instrucciones ejecutables a las que se transferirá el control dependiendo del valor de la variable I. Los números entre paréntesis no requieren ir en orden ascendente e inclusive se pueden repetir.

La variable que sigue a la coma siempre debe ser de modo entero y su valor transfiere el control a aquella instrucción que corresponda con la posición de los números entre paréntesis.

Ejemplo 2

```
      7
      I=2
      .
      .
      .
      GO TO(15,12,13),I
```

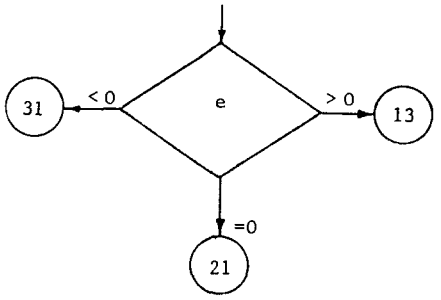
En este caso transfiere el control a la instrucción número 12 ya que está indicada como segundo argumento del GO TO calculado.

Generalmente I puede tener cuando menos los valores de 1 a 10. Evidentemente no es válido darle valores mayores que el número de argumentos o valores negativos, cuando esto se hace, la computadora ejecuta la instrucción que sigue al GO TO calculado.

IF Aritmético

La proposición IF se utiliza para preguntar por el valor numérico de una expresión (proposición condicional).

Diagrama de flujo



Cuando e es negativo se transfiere a la instrucción 31.

Cuando e es cero a la 21.

Cuando e es positivo a la 13.

La forma general de escritura del If Aritmético es:

```
  7
IF(e) n1,n2,n3
```

Donde n1,n2,n3 son números arbitrarios de instrucciones ejecutables, e es una expresión cualquiera.

Ejemplo 3

```
  7
X=1.0
B=2.0
IF(X-B**2)3,2,5
```

En este caso como 1-4=-3 el control se envía hacia la instrucción número 3.

IF Lógico

Se utiliza para preguntar por el valor de una expresión (proposición condicional). Su forma general es:

```
  7
IF (e1 . operador relacional . e2)   instrucción ejecutable
```

e_1 y e_2 son las expresiones que se van a comparar.

Los operadores relacionales son los siguientes:

.LT.	Menor que	<
.LE.	menor o igual	≤
.EQ.	igual	=
.GE.	mayor o igual	≥
.GT.	mayor que	>

En las expresiones lógicas se utilizan, además de los operadores relacionales, los operadores lógicos:

.OR.	o
.AND.	y
.NOT.	no

Ejemplo 4

```
      7
      IF((A.EQ.B).AND.(B.EQ.C))WRITE(3,100)
100   FORMAT(2X,"A=B=C")
      B=2.*A
```

Ejemplo 5

```
      7
      IF((A.EQ.B).OR.(A.EQ.D))C=A+B
      E=C+D
```

En el primer caso se deben cumplir las dos condiciones para que se ejecute el WRITE(3,100), en caso contrario continúa a la siguiente instrucción ejecutable (B=2.*A).

En el segundo caso basta que se cumpla alguna de las dos condiciones.

4.8 INSTRUCCION ITERATIVA DO

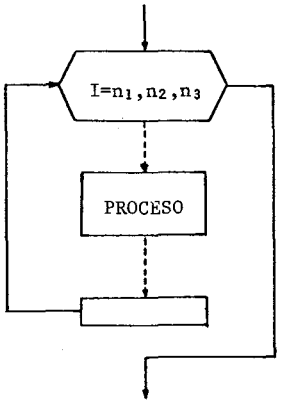
La instrucción iterativa en FORTRAN es la instrucción DO. La forma general de esta instrucción es la siguiente:

```
     DO n I=n1,n2,n3
       Instrucciones
       Ejecutables
n    CONTINUE
```

n es un número positivo que corresponde con la última instrucción ejecutable en el ámbito del DO. Instrucciones no ejecutables no son válidas como límite del rango de un DO (ejemplo: FORMAT DIMENSION, Comentarios).

I es una variable de tipo entero que variará de su valor inicial n_1 a su valor final n_2, en incrementos de valor n_3. Si el incremento es unitario se puede omitir n_3.

Símbolo de diagrama de flujo

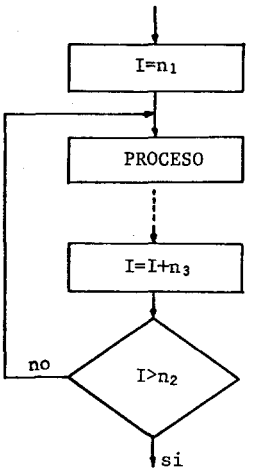


n_1, n_2, n_3 son variables o constantes de tipo entero, sin embargo, algunas computadoras permiten valores fraccionarios o variables reales.

El DO equivale en principio a los siguientes elementos:

1 contador
1 pregunta
1 transferencia de control

El diagrama equivalente sería:



Ciclos Anidados

Se llaman ciclos anidados aquellos ciclos iterativos que dentro de su ámbito contienen a otro u otros ciclos iterativos.

El emplear varios ciclos iterativos en forma anidada (un DO dentro del ámbito de otro DO), nos permite programar en forma más fácil, por ejemplo el manejo de variables de 2 y 3 dimensiones, en lectura, etc.

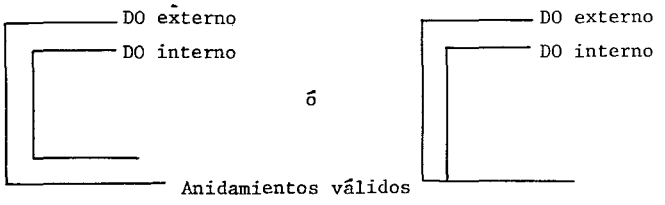
Al utilizar ciclos anidados se deben observar las siguientes reglas:

1. Todas las proposiciones de un DO interior deben estar contenidas dentro del DO exterior, el ámbito de los 2 o más DO'S pueden terminar con la misma proposición.

2. La última proposición en el ámbito de un DO no debe ser un GO TO, un IF, un RETURN, un STOP, un PAUSE, u otro DO.

3. No es válido dentro del ámbito de un DO, utilizar proposiciones que alteren alguno de los parámetros de índice del DO (I, n_1, n_2, n_3).

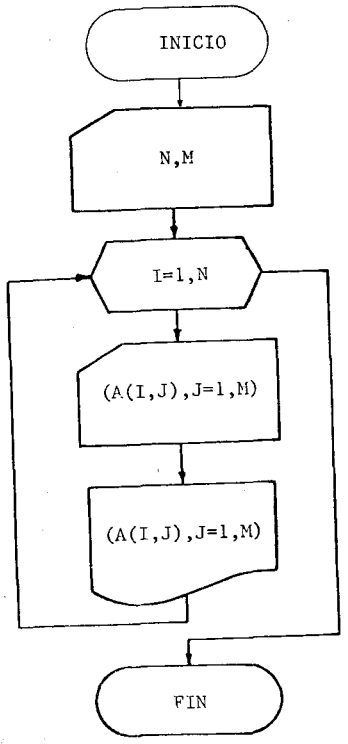
4. No es válido realizar una transferencia de control al interior del ámbito de un DO; sin embargo, sí es válido una transferencia de control de un DO interior hacia el ámbito de un DO exterior.



Ejemplo 1

Lectura y escritura de un arreglo bidimensional por renglones.

```
      |7
      |DIMENSION A(10,10)
      |READ (5,100)N,M
      |DO 2 I=1,N
      |DO 2 J=1,M
      |READ (5,200) A(I,J)
    2 |WRITE (6,300) A(I,J)
  100 |FORMAT (2I2)
  200 |FORMAT (F8.2)
  300 |FORMAT (2X,F8.2)
      |CALL EXIT
      |END
```

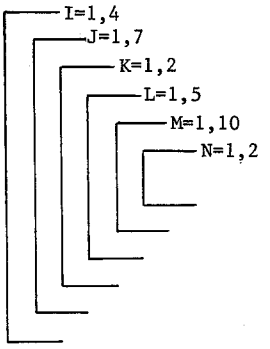


```
      |7
      |DIMENSION A(10,10)
      |READ (5,100) N,M
      |DO 2 I=1,N
      |READ (5,200) (A(I,J),J=1,N)
    2 |WRITE (6,300) (A(I,J),J=1,M)
  100 |FORMAT (2I2)
  200 |FORMAT (10F8.0)
  300 |FORMAT (10(2X,F8.2))
      |CALL EXIT
      |END
```

En este último caso los DO'S internos se encuentran de manera implícita en las instrucciones READ y WRITE.

En los ciclos anidados, el ciclo más interno se repite por cada variación exterior.

Ejemplo 2



En este caso N tomará los valores 1 y 2 un total de 2800 veces.

4.9 VARIABLES CON INDICE. DECLARACION DIMENSION

En FORTRAN se puede manejar arreglos de una , dos o tres dimensiones, el único requisito es declararlos al inicio de un programa, esto es, indicarle a la computadora el máximo número de elementos de cada arreglo. Esto se logra mediante la declaración DIMENSION.

Ejemplo 1

```
      |7
      |DIMENSION A(10),B(4,5),C(3,3,2)
```

Indica que manejamos un vector de 10 elementos, un arreglo bidimensional de 20 elementos y un arreglo tridimensional de 18 elementos.

En la declaración DIMENSION, los índices de las variables, deberán ser constantes enteras, separadas por comas cuando haya más de una.

Para identificar algún elemento en especial se emplea la variable que define el arreglo, pero como índice siempre se utiliza una variable de modo entero.

Ejemplo 2

```
      7
      DIMENSION A(7,7),Z(5,1,4),L(3,2)
      A(3,5)=2.0
      Z(1,1,1)=-35.7
      L(2,2)=84
```

Esto es, podemos abarcar los valores de cada elemento del arreglo en el plano cartesiano o en el espacio tridimencional.

4.10 DECLARACIONES

Declaración DATA

Se utiliza para inicializar un grupo de variables. Su forma general es:

```
      7
      DATA A,B,C,..Z/ valores de cualquier tipo (separados por comas)/
```

La declaración DATA se indica al principio del programa (después de DIMENSION, COMMON, INTEGER), y es válida sólo al inicio de la ejecución. Por supuesto, las variables pueden cambiar sus valores con el proceso, pero no es válido tratar de ejecutar una segunda vez esta declaración.

Ejemplo 1

```
      7
      DATA AB,C,D,I/'ALTA',-14.5,0.15E-12,7/
```

Ejemplo 2

```
      7
      DIMENSION K(12)
      DATA K/12*2/
```

En este último ejemplo se ha inicializado el vector K con sus 12 elementos por efecto del * y el valor entero que le precede.

Ejemplo 3

```
      7
      DIMENSION N(8)
      DATA N/8*'Ø'/
```

En el que se han inicializado los elementos del vector N en el valor alfanumérico 'Ø'.

Ejemplo 4

```
      7
      DIMENSION Z(7),B(4),X(3)
      DATA X(3),Z(4),B(3),B(1)/3.5,4.2,0.0,-1.0/
```

Notése que la lista de variables se indica según se desee.

Declaraciones INTEGER y REAL

En FORTRAN se identifica el modo de las variables por la primera letra de sus nombres; sin embargo usando las declaraciones INTEGER o REAL, puede modificarse dicha identificación.

```
      7
      INTEGER A,B,C,X
      REAL I,J,K,L
```

Estas declaraciones pueden usarse junto con DIMENSION.

Ejemplo 1

```
      7
      DIMENSION R(10)
      INTEGER R
```

equivale a:

```
      7
      INTEGER R(10)
```

Ejemplo 2

```
      7
      DIMENSION T(7),Y(4),I(3)
      INTEGER T,Y,I
      REAL L(8),MAS(12)
```

equivale a:

```
      7
      INTEGER T(7),Y(4),I(3)
      REAL L(8),MAS(12)
```

Se observa que las declaraciones REAL, INTEGER también tienen la función de reservar memoria para arreglos.

Declaración COMMON

Permite un área común de memoria para el programa principal y los subprogramas. Asimismo, nos permite no indicar argumentos en subprogramas.

Ejemplo

```
      7
      COMMON A,B,C,D(10)
      DATA D/10*10.0/
      A=1.5
      B=0.0
      C=-3.5
      CALL SUBA
      WRITE (6,300)A,B,C,(D(I),I=1,10)
300   FORMAT (13(1X,F6.2))
      CALL EXIT
      END
```

```
      |7
      |SUBROUTINE SUBA
      |COMMON X,Y,Z,T(10)
      |X=X+Y+Z
      |Y=X-Y-Z
      |.
      |.
      |.
      |DO 12 I=1,10
   12 |T(I)=X+Y
      |RETURN
      |END
```

Como se observa, la subrutina SUBA necesitaría un total de 13 argumentos si no se usara COMMON. Las variables en COMMON ocupan la misma localización de almacenamiento, de ahí que se les pueda dar nombre en el programa principal y otro en el subprograma. Esto es, la primera localidad de almacenamiento se llama A en el programa principal y X en el subprograma, la segunda B y Y, la tercera C y Z y de igual forma el vector D y T con una equivalencia elemento a elemento. Realmente lo necesario es respetar el modo (entero o real) de las variables y la posición de éstas.

Declaración COMPLEX

Permite definir variables complejas (del tipo a+bi), para tratarlas en esa forma durante el proceso de nuestro programa. La forma general de la declaración COMPLEX es la siguiente:

```
      |7
      |COMPLEX  lista de variables
```

La lista de variables deberá ir separada por comas. La declaración COMPLEX puede tener, agregada a ella, la función de reservar memoria.

Ejemplo 1

```
      |7
      |COMPLEX A(20),T(5),Z(5,5)
      |COMPLEX H,W
```

Como se observa en la primera declaración se tienen arreglos de una y dos dimensiones. En la segunda declaración, sólo variables sin índice (notése que es posible omitir la segunda declaración incluyendo toda la lista en la primera declaración).

Ya en el proceso de nuestro programa, a fin de formar nuevos valores complejos, podemos emplear la función CMPLX.

Ejemplo 2

```
      7
      COMPLEX T(2)
      A=3.5
      B=1.0
      T(1)=CMPLX(A,B)
      T(2)=CMPLX(A,-B)
      WRITE (6,100)T(1),T(2)
100   FORMAT (5X,2G15.7,5X,2G15.7)
```

Evidentemente T(1) y T(2), están recibiendo valores complejos y por lo tanto requieren cada uno de ellos de dos especificaciones de campo para su escritura.

4.11 SUBPROGRAMAS

Concepto de Subprograma

Un subprograma es un conjunto de instrucciones en secuencia que aparecen una sola vez fuera del programa principal y se utiliza para evitar una repetición de ese mismo conjunto de instrucciones en diferentes lugares de dicho programa principal.

En FORTRAN se emplean funciones y subrutinas, el subprograma para definir una función se conoce como subprograma FUNCTION, el subprograma para definir una subrutina es el subprograma SUBROUTINE.

Subprograma FUNCTION

La utilidad del FUNCTION es evidente; nos permite construir funciones especiales, según nuestras propias necesidades de programación.

La llamada de un FUNCTION es igual a la llamada de funciones de biblioteca como es el caso de SIN (X), ABS (X), esto es, dado un valor al argumento, el subprograma FUNCTION nos calcula un valor que corresponde a la función en sí, asignando dicho valor al nombre propio de la función.

Ejemplo 1

Función que evalúa un polinomio de segundo orden.

Veamos primeramente el programa principal

```
C     PROGRAMA PRINCIPAL
      X=3.0
      Y=F(X,A,B,C)
C     LA FUNCION ESTA DEFINIDA EN EL
C     SUBPROGRAMA. SU NOMBRE ES F
C     SUS ARGUMENTOS SON X,A,B,C
      .
      .
      .
      END
```

Veamos ahora el subprograma FUNCTION

```
      7
      FUNCTION F(X,A,B,C)
      F=A*X**2+B*X+C
      RETURN
      END
```

Un subprograma FUNCTION puede manejar varios argumentos que deben corresponder en número y modo con los definidos en el programa que lo va a usar, (en este caso 4 argumentos reales); la característica distintiva es que el FUNCTION regresa sólo un valor como resultado y dicho valor se asocia al nombre del FUNCTION mismo (en este caso F con lo que se sabe es una función real).

El subprograma FUNCTION recibe los valores de los argumentos al mencionarse la función en el programa principal, inmediatamente evalúa el proceso indicado en el FUNCTION y regresa por efecto del RETURN a la instrucción misma que lo llamó para de ahí continuar con el programa.

Subprograma SUBROUTINE

Este subprograma involucra también un nombre y argumentos pero a diferencia del anterior, no importa el modo (entero o real) del nombre de la subrutina. Para llamar a una subrutina se emplea la instrucción CALL.

Ejemplo 2

```
      7
      CALL AREA (A,B,C)
```

Los argumentos permiten enviar valores de variables a la subrutina y también recibir resultados de ella para cada variable enviada.

```
     7
     SUBROUTINE AREA (X,Y,Z)
     X=0.5*Y*Z
     RETURN
     END
```

Evidentemente la variable X del subprograma corresponde con la variable A del programa que llama, Y con B, y Z con la variable C, en esto es definitivamente necesario respetar el modo de las variables; el nombre de éstas pueden cambiarse al gusto.

4.12 INTRODUCCION AL MANEJO DE ARCHIVOS

Cuando se maneja gran cantidad de información, no es conveniente la manipulación de ella en la memoria principal de la computadora.

En este caso conviene mandarla a una memoria secundaria, como pueden ser discos o cintas, y de ahí recuperarla o grabarla, conservando en la memoria de alta velocidad sólo lo indispensable.

Veremos como ejemplo el manejo de archivos en una computadora IBM 1130.

Cada disco contiene 200 cilindros; cada cilindro está formado por 2 pistas (una inferior y otra superior); cada pista se divide en 4 sectores; cada sector permite 320 palabras para almacenamiento de información.

El disco cuenta con 1600 sectores numerados del 0 al 1599 en la primera palabra de cada uno de ellos. Con esto se dispone de 512000 palabras de computadora. La información se almacena a partir del principio de cada sector ocupando siempre un número entero de palabras. El archivo está formado por un conjunto de registros. Un registro contiene toda la información relativa a un concepto (por ejemplo los registros de alumnos para préstamo de libros en una biblioteca).

Esto es, un registro puede ser: una tarjeta perforada, una línea impresa, etc..

Para la computadora es necesario indicar el tamaño del registro (en palabras de computadora). La computadora del ejemplo asigna el siguiente tamaño a las variables:

v. enteras	1 palabra
v. reales	2 palabras

Si se emplea alguna instrucción para el compilador FORTRAN es posible asignar dos palabras a variables enteras y tres a variables reales.

El tamaño de cada registro es importante dado que:

- En un sector sólo se almacena un número entero de palabras.

- Con el total de registros definiremos el tamaño de nuestro archivo.

Para el manejo en disco del archivo cuyos registros contengan el nombre, carrera, número de cuenta y edad de 1000 alumnos de la Facultad haremos:

NOM	-	arreglo de 7 elementos	14 palabras
CARR	-	arreglo de 6 elementos	12 palabras
NOCTA	-	arreglo de 2 elementos	4 palabras
N	-	variable entera (edad)	1 palabra

```
     |7
     |DIMENSION NOM(7),CARR(6),NOCTA(2)
C    |DECLARACION DEL ARCHIVO
     |DEFINE FILE 1(1000,31,U,KRO)
C    |LECTURA DE DATOS DE TARJETA
     |NREG=0
   1 |READ(2,10,END=7)NOM,CARR,NOCTA,N
     |NREG=NREG+1
C    |GRABADO A DISCO
     |WRITE (1'NREG)(NOM(K),K=1,7),CARR,NOCTA,N
     |GO TO 1
C    |RECUPERACION DE DISCO E IMPRESION
C    |DE REGISTROS EN PAPEL
   7 |DO 2 MREG=1,NREG
     |READ (1'MREG)NOM,CARR,NOCTA,N
   2 |WRITE (3,11)NOM,(CARR(L),L=1,6)NOCTA,N
     |WRITE (3,12)
  10 |FORMAT(I2,7A4,2X,6A4,6X,2A4,2X,I2)
  11 |FORMAT(5X,7A4,3X,6A4,4X,2A4,3X,I2)
  12 |FORMAT(//////,5X,77(1H$),///)
   * |34X,15H FIN DE ARCHIVO,//)
     |CALL EXIT
     |END
```

Para almacenar permanentemente en disco el archivo, deberemos indicar las siguientes tarjetas:

```
// JOB
// FOR
*  SAVE
*  IOCS (DISK)
}  PROG. FUENTE
// XEQ
}  DATOS
// DUP
*  STORE   WS   UA   EJEMP   100
```

Como observamos cada registro tiene un tamaño de 31 palabras, y por tanto 320/31=10 registros por sector con un total de 10 palabras no empleadas. A su vez esto implica que usaremos 1000/10=100 sectores.

El archivo está identificado por el número arbitrario 1 en la declaración de archivo DEFINE FILE; en ésta la U (3er. elemento), no nos interesa mas que indicarla y su significado es SIN FORMATO en disco.

El cuarto elemento es una variable entera arbitraria que lleva un control automático de registros.

Por último es importante notar que las lecturas y escrituras a disco, en la computadora que hemos tomado para el ejemplo, se caracterizan por indicar el número de identificación del archivo y el número del registro ligados por un apóstrofo, lo cual varía de computadora a computadora.

BIBLIOGRAFIA

Donald D. Spencer
SOLUCION DE PROBLEMAS CON FORTRAN
Editorial Prentice - Hall Int.
España, 1980

Seymour Lipschutz, Arthur Poe
PROGRAMACION CON FORTRAN
Primera edición
Editorial Mc Graw Hill, Serie Schaum
Colombia, 1979

Arechiga G. Rafael
FUNDAMENTOS DE COMPUTACION
Segunda edición
Editorial Limusa
México, 1978

Joan Kirkby Hughes
PROGRAMACION DEL SISTEMA IBM 1130
Primera edición
Editorial Limusa - Wiley
México, 1973

C S C - U.N.A.M.
BURROUGH'S 6700, BASIC REFERENCE MANUAL
México, 1979

5. PAQUETES DE BIBLIOTECA

5.1 DEFINICION DE PAQUETE DE BIBLIOTECA

En una máquina computadora es posible almacenar en ciertos dispositivos, programas elaborados por una persona o conjunto de personas quedando agrupados de manera que formen lo que se denomina una biblioteca. Un usuario de alguno de los programas existentes en dicha biblioteca, sólo necesita suministrar el nombre del programa que desea utilizar conjuntamente con los datos para que obtenga resultados.

Reciben el nombre de paquetes de biblioteca los programas que se encuentran almacenados en dispositivos tales como discos magnéticos, diskettes, etc. y que pueden ser usados por cualquier persona.

5.2 MANEJO DE INSTRUCTIVOS PARA ENTRADA DE DATOS Y OBTENCION DE RESULTADOS

Es necesario que el usuario de paquetes de biblioteca, una vez que ha verificado que el paquete que desea se encuentra disponible en la computadora que va a utilizar, conozca la forma en la cual debe alimentar dicha información. En este sentido, deberá proveerse de la documentación correspondiente del paquete a utilizar. Dicha documentación recibe el nombre de instructivo de entrada de datos.

En la elaboración de un instructivo figuran principalmente los siguientes puntos:

1. OBJETIVO

Se trata de describir la finalidad del paquete, así como sus limitantes.

2. DATOS

En este inciso se explica cómo deben ir los datos de entrada, es decir las especificaciones de campo y el significado de las variables que intervienen en él.

3. TARJETAS DE CONTROL

Es necesario que aparezcan codificadas las tarjetas de control de la computadora que se va a utilizar.

4. EJEMPLO

Para mayor claridad se recomienda incluir un ejemplo en el que se detalle el orden en que deben ir las tarjetas de control y los datos.

5. RESULTADOS

El instructivo debe tener información acerca de la manera en que produce los resultados el paquete.

Es conveniente también tener como parte de la documentación un ejemplo de los resultados que se obtuvieron al utilizar la información que sirvió para probar el paquete.

Tarjetas de Control

Para lograr que el paquete que se vaya a emplear se encuentre en la memoria de la máquina computadora para su ejecución se emplean generalmente tarjetas llamadas de control, las cuales difieren de una máquina a otra.

5.3 USO DE PAQUETES E INTERPRETACION DE RESULTADOS

En muchos casos no es fácil interpretar adecuadamente los resultados, sobre todo si el paquete utilizado es técnico y la persona usuaria de dicho paquete no está familiarizada con la nomenclatura o terminología utilizada en los resultados obtenidos, por lo cual se recomienda un análisis o una investigación a fin de conocer la terminología y el significado de los parámetros utilizados para una interpretación adecuada y correcta.

En muchas ocasiones se plantea la necesidad de complementar la salida de resultados de un paquete con información adicional, la cual no se contempla en dicho paquete.

Una forma de lograrlo es elaborar un programa que procese de acuerdo a los requerimientos la información adicional que se proporciona y además considere como subprograma el paquete de biblioteca con el fin de obtener un reporte que cubra los requerimientos que se solicitan.

También es común que un paquete de biblioteca llame a un subprograma el cual es elaborado por el usuario, motivo por lo que en el instructivo de uso del paquete en cuestión se debe describir el nombre de la subrutina y variables que intervienen además de la información ya descrita anteriormente.

Lo más frecuente al utilizar un paquete de biblioteca es que no sea necesario considerarlo como subprograma de un programa principal ni que éste sea considerado programa principal al cual se debe adicionar una subrutina.

En general, el utilizar un paquete de biblioteca por sí solo es suficiente para satisfacer los requerimientos del problema que se desea resolver.

Ejemplo 1

A continuación se muestra el instructivo para el uso del paquete AKD91 cuya finalidad es la de obtener la gráfica hasta de cinco funciones, las cuales se proporcionarán mediante una subrutina llamada A7AZU.

A K D 9 1

OBJETIVO

Graficar hasta cinco funciones.

DATOS (Sólo una tarjeta)

Variable	Formato	Columnas	Descripción
M	I1	Col. 10	Número de funciones ≤ 5
G1	I1	Col. 20	= 0 Todas juntas = 1 Separadas
M1	I1	Col. 30	= 0 Gráfica entre 0 y CRIT = 1 Gráfica entre $\frac{-CRIT}{2}$ a $\frac{CRIT}{2}$
CRIT	F10.5	Cols. 31 a 40	Si = 0 $\Rightarrow$ CRIT = 4π

Donde CRIT son los radianes a graficar ($0 < CRIT < 4\pi$)

Se grafican únicamente 101 puntos, ya sea en una sola hoja o en hojas separadas.

NOTA:

Es necesario generar una subrutina llamada A7AZU(X,F1,F2,F3,F4,F5) donde X es el valor de la abscisa para el cual quieren conocerse las funciones de X; las M funciones se codifican y las restantes se igualan a 0.

Las gráficas de las funciones F1,F2,F3,F4 y F5 serán representadas por 1,2,3,4 y 5 respectivamente. Si algunas de ellas son iguales a cero no se graficarán.

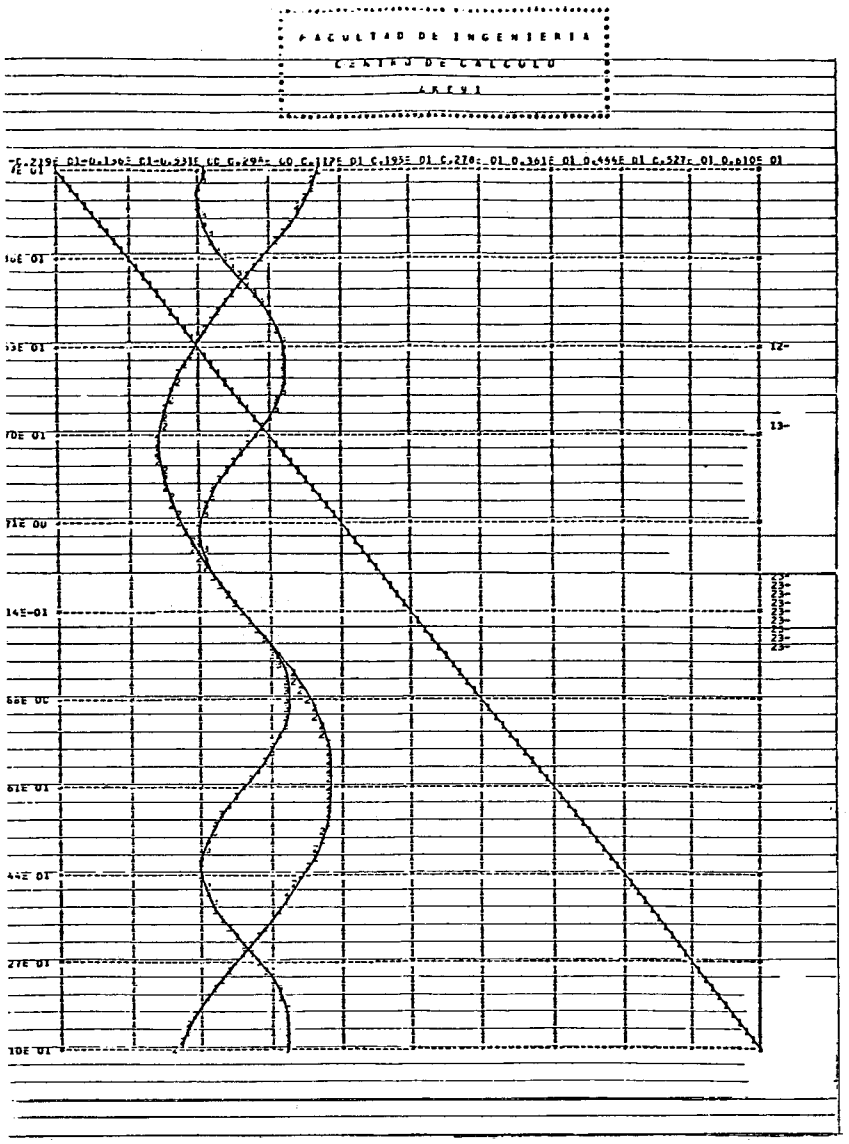
EJEMPLO

Si M = 3 se codificaría como sigue, a partir de la columna 7.

```
      SUBROUTINE A7AZU(X,F1,F2,F3,F4,F5)
      F1=X+2.0
      F2=SIN(X)
      F3=SIN(X)*COS(X)
      F4=0.0
      F5=0.0
      RETURN
      END
```

Las tarjetas de control, de la subrutina y de datos, quedarán organizadas como sigue:

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890123
// JOB T
// FOR
** NOMBRE, NUM. DE CUENTA, FECHA, ETC.
      SUBROUTINE A7AZU(X,F1,F2,F3,F4,F5)
      F1=X+2.0
      F2=SIN(X)
      F3=SIN(X)*COS(X)
      F4=0.0
      F5=0.0
      RETURN
      END
// XEQ AKD91
        3          0        1  8.3
/*
12345678901234567890123456789012345678901234567890123
         1         2         3         4         5
```


FACULTAD DE INGENIERIA
CENTRO DE CALCULO
UNAM
12-
13-

Ejemplo 2

Se presenta el instructivo para el uso del paquete EC2GR cuya finalidad es resolver ecuaciones cuadráticas de la forma $AX^2 + BX + C = 0$

EC2GR

OBJETIVO

Resolver ecuaciones cuadráticas. Las raíces de la ecuación se determinan mediante:

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$ Si las raíces son reales

$$X_{1,2} = \frac{-b}{2a} \pm \frac{\sqrt{b^2 - 4ac}}{2a}$$ Si las raíces son complejas

DATOS (dos tipos de tarjetas)

1ª Tarjeta:

Variable	Formato	Columnas	Descripción
NOM	15A2	Cols. 1 a 30	nombre del usuario
N	I2	Cols. 31 y 32	número de ecuaciones a resolver $N \leq 99$

2ª(s) Tarjeta(s):

Variable	Formato	Columnas	Descripción
A	F10.3	Cols. 1 a 10	coeficiente de X^2
B	F10.3	Cols. 11 a 20	coeficiente de X
C	F10.3	Cols. 21 a 30	término independiente

NOTA:

Se pueden tener hasta 99 tarjetas de tipo 2.

TARJETAS DE CONTROL

Las tarjetas de control para este paquete son las siguientes:

```
// JOB T
// FOR
//* COMENTARIOS
// XEQ EC2GR
   } DATOS
/*
```

EJEMPLO

Resolver cuatro ecuaciones cuadráticas.

1. $X^2 - X - 2 = 0$
2. $2X^2 + 7X + 6.999 = 0$
3. $12X^2 + 3X = 0$
4. $48X^2 = 0$

Las tarjetas de control y de datos quedarán organizadas como sigue:

```
         1         2         3         4
1234567890123456789012345678901234567890
-----------------------------------------
// JOB T
// FOR
// * NOMBRE,NUM.DE CUENTA,FECHA,ETC.
// XEQ EC2GR
DOMINGUEZ LOPEZ JUAN JORGE   04
    1.0      -1.0      -2.0
    2.0       7.0       6.999
   12.000     3.000     0.000
   48.0       0.0       0.0
/*
-----------------------------------------
1234567890123456789012345678901234567890
         1         2         3         4
```

RESULTADOS

Los resultados que se obtienen son: el nombre del usuario, los coeficientes de la ecuación, el tipo de raíces y el valor de las raíces.

BIBLIOGRAFIA

CECAFI CSC - U.N.A.M.
INSTRUCTIVOS DE PAQUETES DE BIBLIOTECA
México, 1979

6. LENGUAJE BASIC

6.1 USO DE MINICOMPUTADORAS Y MICROCOMPUTADORAS

El adelanto tecnológico de los últimos años ha permitido que cada vez en menor espacio, se tenga una gran capacidad de cómputo.

Algunas aplicaciones complejas requieren todavía de grandes computadoras para solucionar sus problemas; sin embargo existen muchas aplicaciones menos complejas que satisfacen sus necesidades de cómputo con las llamadas microcomputadoras y minicomputadoras. Se trata de computadoras de tamaño pequeño que, en muchos casos, no requieren de instalaciones especiales ni de grandes áreas para colocarlas, pero que son capaces de resolver problemas de razonablemente gran complejidad.

Sus limitaciones principales son capacidad en memoria y tiempo de proceso.

En muchas microcomputadoras y minicomputadoras se ha implantado el lenguaje BASIC como principal, y algunas veces como único compilador.

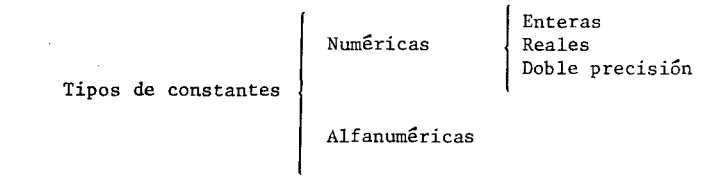
6.2 VENTAJAS Y DESVENTAJAS DEL LENGUAJE BASIC

Entre las principales ventajas que tiene este lenguaje se consideran su simplicidad, la facilidad de aprenderlo, la ausencia de formatos de lectura y escritura, estas mismas ventajas podrían ser consideradas como desventajas, excepto la facilidad de aprendizaje, así como la ausencia de instrucciones para manejo de archivos, la falta de estructuras complejas, etc.

6.3 ELEMENTOS DEL LENGUAJE BASIC

Constantes

Las constantes son representaciones simbólicas de valores fijos que aparecen en un programa. Existen varios tipos de constantes:



El tipo es el atributo asociado con una constante que especifica de qué manera esta constante será almacenada, interpretada u operada.

En la escritura de constantes de tipo numérico se deben observar las siguientes reglas:

a) No deben aparecer símbolos especiales tales como, * ¡ " ... etc. excepto el punto decimal y los signos + ó - precediendo a la constante.

b) Una constante se puede expresar en forma exponencial utilizando la letra E que corresponde al número 10, así por ejemplo la cantidad 2.5 x 10^{-5} se puede escribir en BASIC como 2.5E-5

   El exponente puede variar desde 38 hasta -38 aunque el rango puede variar de una computadora a otra.

   Además el exponente debe ser un número entero positivo o negativo pero no decimal.

c) En la mayoría de las versiones del lenguaje BASIC se pueden tener números de hasta ocho o nueve cifras significativas.

Ejemplo 1

Escritura válida de constantes en BASIC.

25	-286.50	3.1415926
+25	-2.8650E2	3.1415926D0
2.5E+1	-286.50E0	.31415926D1
0.2E+2	-28650.E-2	.031415926D2

Las constantes de tipo alfanumérico son aquellas cadenas de combinaciones de números, letras y caracteres especiales encerrados entre comillas.

Ejemplo 2

"MARIA" "A7$W" "JOSE LUIS"

Variables

Una variable es un nombre que se utiliza para representar constantes.

- Variables
  - Numéricas
    - Enteras
    - Reales
    - Doble precisión
  - Alfanuméricas

Una variable alfanumérica representa una serie de caracteres los cuales pueden consistir de letras; letras y números o bien letras, números y caracteres especiales.

El número máximo de caracteres que puede contener una variable alfanumérica varía de versión a versión de BASIC en algunas puede contener hasta 15 caracteres en otras hasta 4095.

Es de tomar en cuenta que, a pesar de que una variable alfanumérica contenga únicamente números, ésta en realidad no puede considerarse como variable numérica.

Para escribir una variable numérica se debe emplear una letra o bien una letra seguida por dígito.

Una variable alfanúmerica se debe escribir como una letra seguida por el signo de pesos $.

Algunas versiones del lenguaje BASIC permiten escribir el nombre de una variable numérica como letra seguida de letra; además las variables alfanuméricas pueden ser escritas como letra seguida de dígito, seguida del signo de pesos $, o bien como letra seguida de letra, seguida del signo de pesos $.

Ejemplo 1

Las siguientes variables pueden representar una cantidad numérica.

A, B, J, K, AI, KI, A3, IN, B8

Ejemplo 2

Las siguientes variables pueden representar una cantidad no numérica (serie de caracteres).

A$, B$, J$, K$, K3$, BP$

Operadores y sus Prioridades

En BASIC para indicar operaciones aritméticas, se utilizan los siguientes símbolos llamados operadores:

Exponenciación	↑
Multiplicación	*
División	/
Suma	+
Resta	-

Los operadores se utilizan para unir constantes o variables numéricas formando expresiones aritméticas o fórmulas.

Una expresión aritmética o fórmula puede estar formada por una constante, una variable numérica o una combinación de constantes, variables numéricas y operadores, sin embargo es necesario que cualquier variable numérica que aparezca en una fórmula se le haya asignado previamente un valor numérico.

Prioridades entre operadores al ejecutar operaciones:

1. Se ejecutan las operaciones de exponenciación.
2. La multiplicación y división, dentro de una fórmula en la que aparecen multiplicación y división, se ejecutan según su orden de aparición ya que ambas tienen la misma jerarquía.
3. Suma y resta, al aparecer en una fórmula operaciones de suma y resta, también se ejecutan según su orden de aparición ya que tienen la misma jerarquía.

Por lo tanto en una expresión en la que se tengan operaciones de exponenciación, multiplicación, división, suma, resta, éstas se ejecutarán de acuerdo a las prioridades señaladas, es decir primero las exponenciaciones, luego las multiplicaciones y/o divisiones y por último las sumas y/o restas.

Cabe aclarar que las operaciones se ejecutan de izquierda a derecha dentro de un mismo grupo jerárquico.

Algunos ejemplos de expresiones aritméticas en BASIC y sus equivalentes algebraicos son los siguientes:

Expresión BASIC	Expresión Algebraica
A/B*C	$\frac{a}{b}c$
B↑2-4*A*C	$b^2 - 4ac$
2*C*X/Y*N	$\frac{2cx}{y}n$
Z*3↑N/2	$\frac{z\ 3n}{2}$
3*X↑3+2*X↑4+2*X↑5-4*X↑2/7*N	$3x^3 + 2x^4 + 2x^5 - \frac{4x^2}{7}n$
N*X/Y-2	$\frac{nx}{y} - 2$

Una forma de alterar la jerarquía normal de ejecución de las operaciones, es mediante el uso de paréntesis ya que en una fórmula en la que

aparezcan paréntesis, siempre por pares (abre, cierra), se realizan primero las operaciones que se encuentran dentro de los paréntesis observando la jerarquía ya establecida antes.

Ejemplos

Expresión en BASIC	Expresión Algebraica
(B↑2-4*A*C)/(2*A)	$\frac{b^2 - 4ac}{2a}$
(4*X+Y)/((3*A+2*B)/Z)	$\frac{4x + y}{\frac{3a + 2b}{z}}$
(A+B)/2	$\frac{a + b}{2}$
5*(M+N+O)	$5(m + n + o)$
(A+B)/(X-Y)	$\frac{a + b}{x - y}$

Funciones

En el lenguaje BASIC al igual que en otros lenguajes es posible evaluar funciones matemáticas, teniendo acceso a la función mediante la referencia de su nombre. Las funciones matemáticas más comúnmente utilizadas son las siguientes:

DESCRIPCION	FUNCION
valor absoluto de x	ABS(x)
ángulo cuya tangente es x	ATN(x)
coseno de x	COS(x)
cotangente de x	COT(x)
exponencial de x	EXP(x)
asigna el mayor valor entero de x	INT(x)
logaritmo natural de x	LOG(x)
determina el signo de x	SGN(x)
seno de x	SIN(x)
raíz cuadrada de x	SQR(x)
tangente de x	TAN(x)

Las funciones matemáticas descritas son subprogramas que forman parte integral del lenguaje, de tal manera que basta hacer referencia a la función deseada dentro del programa para que ejecute la serie de instrucciones correspondientes a dicha función.

> La parte que se encuentra encerrada dentro del parêntesis recibe el nombre de argumento.
>
> El valor del argumento para las funciones trigonométricas como seno, coseno, etc., se debe proporcionar en radianes.

Algunos ejemplos de funciones en lenguaje BASIC y sus equivalentes algebraicos se dan a continuación:

FUNCION	EQUIVALENCIA
SIN(X)	sen x
ABS(-3)	$\lvert -3 \rvert$
ATN(3*X+Y)	ang tan (3x + y)
COS((X-3)/A)	$\cos \frac{x - 3}{a}$
COT(2/(X+Y+Z))	$\cot \frac{2}{x + y + z}$
EXP(X↑2)	e^{x^2}
SQR(B↑2-4*A*C)	$\sqrt{b^2 - 4ac}$

6.4 INSTRUCCIONES DE ENTRADA Y SALIDA

Como ya se ha visto, un programa consiste de una serie de proposiciones colocadas en el orden en que deben ser ejecutadas. Al escribir un programa en lenguaje BASIC se deben observar las siguientes reglas:

1. Cada proposición debe aparecer en un renglón separado y no debe exceder de la longitud del renglón que generalmente es de 72 caracteres, aunque existen algunas versiones que admiten hasta 255 caracteres.

2. Cada proposición se inicia con un número entero o número de proposición, el cual no debe aparecer en dos o más proposiciones.

3. La numeración debe ser creciente.

4. A continuación del número de proposición debe ir la instrucción que indica la operación a ejecutar.

Proposición READ y Declaración DATA

La proposición READ se utiliza para proporcionar datos a un programa que los requiera, y está formada por un número de proposición, la palabra READ y una lista de variables separadas por comas.

La proposición READ se utiliza siempre en combinación con la declaración DATA; esta última se forma con un número de proposición, la palabra DATA y una lista de constantes numéricas o alfanuméricas separadas por comas.

La declaración DATA sirve para colocar los datos que serán asignados mediante la proposición READ; puede ser colocada en cualquier punto del programa, aunque se acostumbra colocarla al final del mismo.

Con la proposición READ los datos forman parte del programa fuente; si se desean cambiar se debe sustituir la línea o líneas de DATA que los contengan.

Aunque por facilidad es costumbre que para cada READ se escriba un DATA correspondiente, esto no es necesario pues es posible tener varias proposiciones READ y una sola proposición DATA, con la condición de que los datos que figuran en ella deben ser tantos como variables aparezcan en la proposición READ.

La forma de utilizar la proposición DATA es escribir en primer lugar el número de proposición, en seguida la palabra DATA y a continuación los valores numéricos o alfanuméricos de las variables que figuran en la o las proposiciones READ, separados por comas. Cuando se trata de un dato alfanumérico, y éste contiene comas, punto y coma o espacios en blanco, deberá estar delimitado por comillas.

Ejemplo 1

Supongamos que las siguientes proposiciones forman parte de un programa en BASIC.

```
 10 READ X,Y,Z
 20 READ W,A$,B$
 30 READ P(1),P(2),P(3),P(4)
 - - - - - - - - - - - - - -
140 DATA 5,9,-4
150 DATA 100,ALFA,BETA
160 DATA 30,40,50,60
```

Los valores que son asignados a las diferentes variables con estas proposiciones son:

a X,5; a Y,9; a Z,-4; a W,100; a A$,ALFA; a B$,BETA; a P(1), 30;

a P(2), 40; a P(3), 50 y a P(4), 60.

En lugar de tres proposiciones DATA podría ponerse una sola en la siguiente forma:

```
140 DATA 5,9,-4,100,ALFA,BETA,30,40,50,60
```

Proposición RESTORE

La proposición RESTORE sirve para indicar que se trata de leer nuevamente un dato o grupo de ellos previamente leído, para ello se utiliza dicha proposición la cual consta de un número seguido de la palabra RESTORE y la siguiente proposición será un READ.

Ejemplo 2

Se tiene el siguiente grupo de proposiciones en un programa BASIC.

```
 10 READ A,B$,C,D$
 - - - - - - - - -
 80 RESTORE
 90 READ W,X$,Y,Z$
150 DATA 1,PRIMERO,2,SEGUNDO
```

En este ejemplo la proposición 10 (READ) hace que la variable A se le asigne el valor 1, a la variable B$ se le asigne el valor alfanumérico PRIMERO y así sucesivamente, siguiendo la secuencia de instrucciones nos encontramos con la proposición 90 que es otro READ la cual de no estar precedida por la proposición 80 RESTORE implicaría un nuevo juego de datos, sin embargo la proposición RESTORE antes del segundo READ provoca que se vuelvan a asignar los valores 1, PRIMERO, 2, SEGUNDO pero ahora a las variables W, X$, Y, Z$.

Instrucción PRINT

Para transmitir los datos o resultados de salida se utiliza la proposición PRINT por medio de la cual se pueden mandar a impresión variables y/o constantes numéricas o alfanuméricas. La forma de esta instrucción es el número de proposición, la palabra PRINT y la lista de variables que se desean imprimir.

En la lista de variables que se desean imprimir pueden existir además de variables y/o constantes numéricas o alfanuméricas fórmulas y títulos o series de caracteres.

Cuando se envía a impresión un título o serie de caracteres éste debe estar entre comillas.

Las variables numéricas, alfanuméricas y títulos que aparecen en la lista que sigue a la proposición PRINT se deben separar por comas o por puntos y comas.

Cuando en una proposición PRINT no existe la lista de salida, el resultado es un renglón en blanco.

Cuando se utilizan comas para separar las variables o títulos que aparecen en la lista de impresión, el renglón de impresión se divide en cinco zonas de igual longitud (en algunos casos son cuatro zonas) y se imprime un valor en cada zona.

Cuando se desea imprimir los resultados más separados entre sí se pueden utilizar comas consecutivas, sin que existan variables entre ellos.

Si por el contrario, el deseo es imprimir los resultados más cercanos entre sí, sin que intervengan las zonas, se utiliza el punto y coma.

Cuando se desea la impresión de una variable numérica que almacena una cantidad entera que consta de seis dígitos o menos, la computadora la imprimirá como tal, pero si dicha cantidad tiene más de seis dígitos entonces la computadora la redondeará a seis dígitos y aparecerá como un número decimal con exponente.

Ahora, si la cantidad que se desea imprimir es decimal y tiene más de seis dígitos entonces la computadora redondea a seis dígitos y la cantidad aparece con exponente.

Si en una proposición PRINT después de la última variable o título que aparecen en la lista de impresión se coloca un punto y coma, esto indicará que los siguientes resultados que se envíen a impresión aparecerán en los espacios disponibles del mismo renglón de salida.

Ejemplo 1

Suponga que se tiene la siguiente serie de instrucciones de un programa en BASIC.

```
10 PRINT "UNAM","FACULTAD DE INGENIERIA"
20 PRINT
30 PRINT "RESULTADOS"
40 READ A,B,C,D,E,F,G,H
50 PRINT
60 PRINT A,B,C,D,E
70 PRINT F,,,G,H
80 DATA 10,12,8,-4,20,100,38,6
```

Al ejecutarse esta serie de instrucciones, se tiene como salida lo siguiente:

```
UNAM        FACULTAD DE INGENIERIA

RESULTADOS

 10         12        8       -4       20
100                             38        6
```

Si ahora en la proposición 60 cambiamos comas por puntos y comas y eliminamos la proposición 70.

```
60 PRINT A;B;C;D;E;F;G;H
```

Los resultados numéricos aparecerán como:

```
10   12   8   -4   20   100   38   6
```

Ejemplo 2

Sea el siguiente grupo de instrucciones en un programa BASIC

```
20 READ A,B,C,D
- - - - - - - -
100 PRINT A,B,
110 PRINT C,D
- - - - - - - -
200 DATA 826325,-54.80,326425672,0.0008645
```

Se tiene como salida en impresión lo siguiente:

```
826325      -54.80      3.2642E+8      8.64500E-4
```

Si ahora en el mismo ejemplo eliminamos la última coma que aparece en la proposición 100, es decir codificamos 100

```
100 PRINT A,B
```

Entonces los resultados aparecerán de la siguiente forma:

```
826325      -54.80
3.26425E+8      8.64500E-4
```

nótese que:

```
100 PRINT A,B,
110 PRINT C,D
```

Equivale a:

```
100 PRINT A,B,C,D
```

Eliminando la proposición 110

Ejemplo 3

Considere el siguiente grupo de instrucciones de un programa en BASIC.

```
 10 READ A$,B$,C
 - - - - - - - -
 90 PRINT "NOMBRE",A$,"ESCUELA",B$
100 PRINT "CALIFICACION",C
110 DATA JUAN,UNAM,8
```

Al ejecutarse esta serie de instrucciones del programa BASIC, los resultados aparecerán como:

```
NOMBRE  JUAN  ESCUELA  UNAM
CALIFICACION  8
```

Si en la proposición 90 en lugar de comas ponemos puntos y comas e incluimos la lista de la proposición 100:

```
90 PRINT "NOMBRE";A$,"ESCUELA";B$;"CALIFICACION";C
```

se tiene como resultado:

```
NOMBRE JUAN ESCUELA UNAM CALIFICACION 8
```

6.5 INSTRUCCION DE ASIGNACION LET

En la secuencia de instrucciones de un programa en BASIC frecuentemente se desea asignar a una variable el valor de una expresión. La proposición que se utiliza en BASIC para asignar valores numéricos o alfanuméricos a una variable, es la llamada LET, la cual se forma con un número de proposición, la palabra LET a continuación una variable, un signo = y una expresión. La forma general de la proposición de asignación es:

```
LET <v> = <e>
     ↑     ↑
variable  expresión
```

En casi todas las versiones del lenguaje BASIC se puede omitir la palabra LET.

Siempre a la izquierda del signo = en una proposición de asignación deberá ir una variable y a la derecha del signo = podrá ir otra variable, o bien una expresión algebraica, siendo factible también tener una serie de caracteres alfanuméricos debiendo ir delimitada con comillas.

Al escribir una proposición de asignación se debe tener cuidado de que si la variable a la izquierda del signo = es numérica, también debe ser numérica la expresión de la derecha, o bien si es alfanumérica la variable a la izquierda del signo = también deberá ser alfanumérica la expresión de la derecha del signo =

Ejemplo

```
10 LET A=30
20 LET B=A
30 LET C=A+B*30
40 LET A$="FIN"
50 LET B$=A$
```

En este ejemplo a cada una de las variables que se encuentran a la izquierda del signo = se les asigna el valor de la expresión que está a la derecha de dicho signo.

En la mayoría de las versiones del lenguaje BASIC se puede escribir el mismo ejemplo de la siguiente manera:

```
10 A=30
20 B=A
30 C=A+B*30
40 A$="FIN"
50 B$=A$
```

Proposición INPUT

Cuando se está trabajando en tiempo compartido y el volumen de los datos que requiere un programa es pequeño, se puede utilizar la proposición INPUT en lugar de la proposición READ.

La proposición INPUT cumple la misma función de la proposición READ, aunque de diferente manera, ya que los datos no están incluidos en el programa fuente sino que son proporcionados por el programador o usuario del programa al momento de ejecución.

La proposición INPUT se forma con un número de proposición, la palabra INPUT y una lista de variables separadas por comas.

Ejemplo

```
10 INPUT X,Y,X
20 INPUT A$,X1,B$,YI
30 INPUT A(I),Q(I),I$(I)
```

Cuando se utiliza la proposición INPUT, durante la ejecución del programa, aparece el signo de interrogación, el cual indica solicitud de datos, quedando suspendida la ejecución hasta recibir los datos solicitados.

Esta proposición INPUT es muy útil en programas conversacionales. Al suministrar los datos se deben seguir las siguientes reglas:

1. Debe haber correspondencia entre los datos suministrados y la lista de variables que aparecen en la proposición INPUT.

Si la variable es numérica los datos deberán ser numéricos. Si la variable es alfanumérica los datos también deberán ser alfanuméricos.

2. Los datos deben ir separados por comas.

3. Si los datos son alfanuméricos y contienen espacios en blanco, comas o dos puntos se deben utilizar comillas para delimitar los grupos de datos, de no ser así se pueden omitir dichas comillas.

4. Sólo se debe suministrar el valor de las variables que se incluyen en la proposición INPUT.

Ejemplos

Proposición BASIC	?	Datos que se pueden proporcionar al aparecer una interrogación.
INPUT X	?	13.2
INPUT Y	?	18
INPUT X,Y	?	13.2,18
INPUT N$	?	SALVADOR PADILLA GOMEZ
INPUT B,N$	?	-.245,GONZALEZ LOPEZ
INPUT X$,A7	?	FERNANDO HIGUERAS,17.9
INPUT A$,Y	?	EL VALOR ES,27.2
INPUT A$,Y	?	"EL VALOR ES, REDONDEANDO:",27

En este último ejemplo es necesario delimitar el texto entre comillas ya que contiene coma y dos puntos.

6.6 TRANSFERENCIAS DE CONTROL

Proposición Incondicional GOTO

Como se ha visto anteriormente, en un programa las proposiciones se ejecutan en el orden que aparecen una tras otra, sin embargo en algunas ocasiones es necesario modificar el orden de ejecución de las proposiciones. Lo anterior se logra mediante el uso de la proposición GOTO, con esta proposición es posible modificar el orden de ejecución, sin que para ello medie ninguna condición, razón por la que recibe el nombre de proposición de transferencia de control incondicional.

La proposición GOTO está formada por un número de proposición, seguido de la palabra GOTO y del número de proposición a la cual se desea transferir el control.

Ejemplo

Sea una parte de un programa en BASIC.

```
10 INPUT A,B
20 LET C=A+B
30 PRINT "VALOR DE C=";C
40 GOTO 10
```

En este ejemplo se van ejecutando las proposiciones en el orden en que aparecen, pero al llegar a la proposición 40 se transfiere el control de ejecución a la proposición 10, volviendo nuevamente a ejecutarse las proposiciones que siguen a la proposición 10.

Proposiciones Condicionales ON GOTO, IF THEN

Es frecuente en la elaboración de un programa el tratar de que no todas las proposiciones se ejecuten en el orden en el que aparecen. Sino que dependiendo de alguna condición, el control del programa se transfiera a otra proposición la cual pudo ya haberse ejecutado, o bien no sea la que se encuentra a continuación sino que se localice en lugar alejado a la instrucción que se estaba ejecutando. Existen en el lenguaje BASIC dos proposiciones para efectuar esta transferencia. La primera de ellas es la proposición ON GOTO.

La proposición ON GOTO se dice que es una proposición de transferencia de control múltiple. La forma general de esta proposición es un número de proposición seguido de la palabra ON después una variable numérica o una expresión, enseguida GOTO y a continuación dos o más números de proposiciones de tal forma que dependiendo del valor de la variable numérica o de la expresión, se transferirá el control a la proposición que tenga el número que figura después del GOTO y que corresponda en orden al valor de la variable numérica o expresión.

Ejemplo

Sea el siguiente conjunto de proposiciones de un programa en BASIC.

```
10 READ A,B
20 LET C=A+B
30 ON C GOTO 40,10,70
40 PRINT "VALOR DE C=";C
50 PRINT "VALOR DE A=";A;"VALOR DE B=";B
60 GOTO 150
70 LET C=C/2.+A*2.
- - - - - - - - -
- - - - - - - - -
- - - - - - - - -
200 DATA 2,1
```

En este ejemplo al ejecutarse la proposición 30, se transfiere el control a la proposición 70 ya que la variable C tiene el valor de 3.

Si la variable numérica o expresión que figura en una proposición ON GOTO no es un valor entero, se trunca la parte decimal sin redondeo. Así, si en el ejemplo anterior A vale 1 y B vale 0.6 entonces la variable C tiene el valor de 1.6 y al llegar a la proposición 30 la transferencia de control se dirige a la proposición 40 ya que al truncarse el valor de la variable C en lugar de 1.6 considera para efectos de esta proposición el valor 1.

La segunda transferencia de control consta de un número de proposición, enseguida la palabra IF con la condición que se prueba y a continuación la palabra THEN con el número de proposición que corresponda a la que se desee se transfiera el control de ejecución en caso de que se cumpla la condición de que se está probando.

En algunas versiones de BASIC se puede omitir la palabra THEN.

En algunas ocasiones cuando una serie de instrucciones dentro de un programa BASIC se desea que estas se ejecuten varias veces, es decir que haya iteraciones, se hace uso del IF THEN con el GOTO.

Ejemplo

Sea el siguiente conjunto de instrucciones de un programa BASIC.

```
10 READ X,A
20 IF A>=2 THEN GOTO 50
30 LET Y=2.*X+4
40 GOTO60
50 LET Y=4.*X-2
60 PRINT "VALOR DE Y =";Y
- - - - - - - - - - - - - -
- - - - - - - - - - - - - -
- - - - - - - - - - - - - -
200 DATA 5,1
```

En este ejemplo se trata de obtener el valor de Y, calculado éste dependiendo de que una variable numérica tenga un valor menor que dos o bien mayor o igual que dos.

si $A < 2$ $\qquad Y = 2x + 4$

si $A \geq 2$ $\qquad Y = 4x - 2$

Se observa que en la condición interviene lo que se denomina un operador de relación.

Operadores de relación:

Igual a	=
Diferente a	<> ó ><
Menor que	<
Menor o igual que	<=
Mayor que	>
Mayor o igual que	>=

6.7 VARIABLES CON INDICE. DECLARACION DIM

Es frecuente en la elaboración de un programa, por facilidad o conveniencia el trabajar con variables con índice, es decir utilizar arreglos; esto es posible en BASIC mediante la proposición DIM.

Desde luego se trabajarán arreglos cuando las características del problema lo permitan, tales como lista de datos o resultados que se quieren asignar a una sola variable.

En BASIC se utiliza la proposición DIM para dimensionar las variables que se trabajan como arreglos en un programa. Esta proposición está formada por el número de proposición seguido de la palabra DIM, y la lista de variables que figuran como arreglos separadas por comas y entre paréntesis el número de localidades para cada variable. Si el arreglo es de dos o más dimensiones, los números de localidades deberán estar separados por comas.

Ejemplos

```
 5 DIM A(10)

10 DIM V(20),A(10,10)
```

Al utilizar la proposición DIM la computadora aparta un conjunto de localidades de la memoria, en las cuales se alojarán los elementos del arreglo.

Aunque la proposición DIM puede estar colocada en cualquier lugar del programa con la condición de que figure antes de la instrucción donde los arreglos son utilizados, conviene por claridad y facilidad escribirla al inicio del programa fuente.

En BASIC se asigna automáticamente 11 localidades a todas las variables que se vayan a trabajar como arreglos de una sola dimensión y 121 localidades (11x11) a todas las variables que se tengan con dos dimensiones, de esta forma cada índice puede variar de 0 hasta 10.

Esto es que si una variable que se trabaja como arreglo de una sola dimensión no excede el número de localidades que el sistema le asigna en forma automática, no es necesario utilizar la proposición DIM para asignarle localidades en memoria; sin embargo es conveniente utilizar la proposición DIM, aunque el arreglo no exceda de 11 elementos, debido a que de esta manera se ahorra memoria.

Cuando se tienen arreglos con dos dimensiones, se debe tener cuidado de no considerar el valor de 121 localidades como número total ya que 121 se obtiene del producto de 11 renglones por 11 columnas; y si alguna de las dimensiones excede de 11 aunque la otra dimensión sea menor a 11 y el producto sea menor que 121, habrá que utilizar necesariamente la proposición DIM.

Ejemplo

```
10 DIM A(25),B(20,5),C$(15)
```

Esta proposición reserva 26 localidades para A;(21,6) localidades para B y 16 localidades para C$.

El número de localidades es mayor que el que figura en las variables de la proposición DIM debido a que se considera el índice cero.

Siempre que se empiece con índice uno el número de localidades en memoria para cada variable será la que esté indicada, 25 para A;(20,5) para B y 15 para C$.

6.8 INSTRUCCION ITERATIVA FOR TO. INSTRUCCION NEXT

La instrucción iterativa en BASIC es la instrucción FOR TO. Esta instrucción especifica el número de veces que se va a repetir un grupo de instrucciones.

Para marcar el fin de dicho grupo se utiliza la instrucción NEXT.

Entre las palabras FOR y TO se debe incluir una variable numérica sin índice cuyo valor varía cada vez que se ejecuta el grupo de proposiciones.

El número de veces que se ejecuta el grupo de instrucciones está determinado por el valor inicial y final de la variable que figura entre FOR y TO.

Ejemplo

```
- - - - - - - - -
- - - - - - - - -
 80 FOR I=1TO10
 90 LET A=A+I
100 PRINT "VALOR DE A =";A
110 NEXT I
- - - - - - - - - -
- - - - - - - - - -
```

En este ejemplo las instrucciones 90 y 100 se repetirán diez veces; es decir la variable I se incrementará en una unidad, iniciándose con el valor uno hasta llegar al valor diez.

Si se desea incrementar la variable en valor diferente de uno se agrega la palabra STEP a la proposición FOR TO.

Ejemplo

```
 80 FOR I=1TO11STEP2
 90 LET A=A+I
100 PRINT "VALOR DE A =";A
110 NEXT I
```

En este ejemplo las proposiciones 90 y 100 se ejecutarán seis veces para valores de I = 1,3,5,7,9, y 11.

La variable que figura entre FOR y TO no tiene que ser necesariamente un entero positivo, sino que éste puede tomar valores negativos y aún fraccionarios; aún más, el valor inicial y final puede expresarse como variable o como fórmula. Esto es igual para la palabra STEP.

Ejemplo

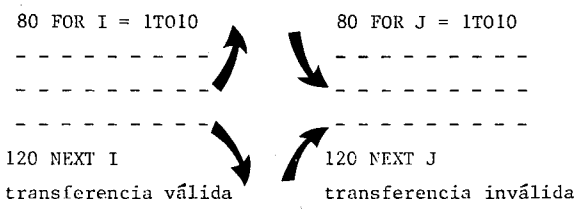
```
 80  FOR X = A/2 TO B↑2+C STEP D+E
120  NEXT X
```

Para indicar el fin del grupo de instrucciones que se están repitiendo se utiliza la proposición NEXT precedida por un número de proposición y seguida por la variable que figura en la proposición FOR TO.

Las reglas que se deben cumplir cuando se utilizan las proposiciones FOR-TO y la instrucción NEXT, en un grupo de instrucciones que se van a repetir varias veces son las siguientes:

1. La variable que figura entre FOR y TO puede aparecer dentro del grupo de instrucciones que se van a repetir, sin embargo no es válido modificar su valor.

2. Se puede transferir el control desde el grupo de instrucciones que se están repitiendo hacia afuera de dicho grupo, pero no desde afuera hacia adentro.



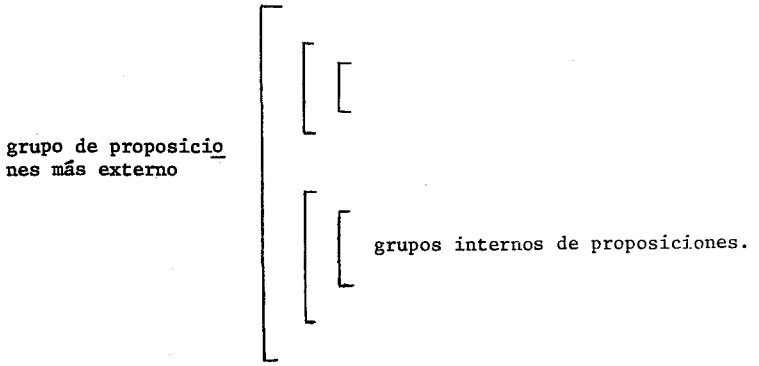
Es posible que un grupo de instrucciones que se va a repetir varias veces contenga otro subgrupo o subgrupos que también se van a repetir, si éste es el caso se deben observar las siguientes reglas:

1. Cada grupo debe empezar con su propia proposición FOR TO y terminar con su propia proposición NEXT.

2. Un grupo externo y otro interno no pueden tener la misma variable en el FOR TO.

3. Cada grupo interno debe quedar completamente contenido dentro del grupo externo.

En forma gráfica:



Ejemplo

Elaborar un programa en BASIC que obtenga las sumas de los valores de cada renglón del siguiente arreglo bidimensional.

$$\begin{bmatrix} 2 & 4 & 6 \\ 8 & 2 & 5 \\ 1 & 9 & 3 \end{bmatrix}$$

Se debe tener como resultado:

$$\begin{bmatrix} 12 \\ 15 \\ 13 \end{bmatrix}$$

```
 10 DIM A(3,3),R(3)
 20 FOR I=1TO3
 30 FOR J=1TO3
 40 READ A(I,J)
 50 R(I) =R(I)+A(I,J)
 60 NEXT J
 70 NEXT I
 80 PRINT "VECTOR COLUMNA ES"
 90 PRINT
100 FOR K=1TO3
110 PRINT R(K)
120 NEXT K
130 DATA 2,4,6,8,2,5,1,9,3
140 END
```

6.9 MANEJO DE ARREGLOS. ENUNCIADOS MAT READ Y MAT PRINT

El manejo de arreglos matriciales es posible hacerlo mediante variables con índice previamente dimensionadas; sin embargo existe en BASIC la proposición MAT la cual puede ser comprendida como una función que sirve para realizar operaciones con matrices tales como lectura, impresión, suma, resta, producto, transposición, inversión, etc., de matrices aunque nos limitaremos sólo a tratar las proposiciones MAT READ Y MAT PRINT.

La forma de la proposición MAT READ consta de un número de proposición seguida de las palabras MAT READ y enseguida los nombres de los arreglos que fueron previamente dimensionados.

La proposición MAT PRINT se escribe en forma similar a la proposición anterior.

Ejemplo

```
 10 DIM X(3,3), Y(3,3)
 20 MAT READ X,Y
 30 PRINT "MATRIZ X"
 40 MAT PRINT X
 50 PRINT
 60 PRINT "MATRIZ Y"
 70 MAT PRINT Y
 80 DATA 1,4,9,7,2,3,8,6,6
 90 DATA 20,30,40,10,20,30,15,30,45
100 END
```

En este ejemplo en la proposición 20 tenemos que los arreglos que leen son:

$$X = \begin{bmatrix} 1 & 4 & 9 \\ 7 & 2 & 3 \\ 8 & 6 & 5 \end{bmatrix}$$

$$Y = \begin{bmatrix} 20 & 30 & 40 \\ 10 & 20 & 30 \\ 15 & 30 & 45 \end{bmatrix}$$

Y los arreglos que se imprimen son:

MATRIZ X			MATRIZ Y		
1	4	9	20	30	40
7	2	3	10	20	30
8	6	5	15	30	45

Es conveniente hacer notar que la proposición MAT no puede ser usada en algunas computadoras.

6.10 SUBPROGRAMAS

Subprograma GOSUB

En algunas ocasiones al elaborar un programa se observa que es necesario repetir un mismo grupo de proposiciones en diferentes lugares del programa, para evitar el escribir dicho grupo de proposiciones en todos y cada uno de los lugares donde se requiera, se aconseja escribirlo una sola vez formando lo que se conoce con el nombre de subprograma o subrutina, siendo suficiente que en cada lugar del programa donde debiera ir, escribir una sola proposición cuya función es transferir el control del programa al grupo de proposiciones que constituyen el subprograma. El regreso a la siguiente proposición inmediata al lugar desde el cual se invocó el subprograma se logra mediante otra proposición.

La proposición para invocar la subrutina se llama GOSUB la cual consiste en un número de proposición, la palabra GOSUB y el número de la primera proposición de dicha subrutina.

La proposición para regresar el control a la proposición inmediata siguiente a la del GOSUB que invocó dicha subrutina, es RETURN.

Es válido invocar una subrutina desde otra subrutina es decir se puede invocar una subrutina desde el programa principal y en la subrutina invocada llamar a ejecución a otra u otras subrutinas.

Ejemplo

Sea un programa que contiene las siguientes proposiciones:

```
 50 GOSUB 150
 - - - - - - -
140 END
150 READ A,B
- - - - - - -
190 GOSUB 400
- - - - - - -
220 RETURN
- - - - - - -
400 FOR I=1TON
- - - - - - -
440 NEXT J
- - - - - - -
500 RETURN
```

Nôtese que una subrutina puede empezar con cualquier proposición, sin embargo la última proposición debe ser una proposición RETURN , como se observa, consiste de un número de proposición y la palabra RETURN la cual hace que se transfiera el control de regreso a la siguiente proposición de la que invocó a dicha subrutina.

Subprograma DEF

De la misma manera que en algunos programas se requiere repetir una serie de proposiciones en varios lugares de dicho programa; también en otros es posible que se necesite utilizar los mismos cálculos en repetidas ocasiones.

Cuando la proposición se va a utilizar en repetidas ocasiones, se puede emplear lo que se conoce con el nombre de función, en forma similar al uso de las llamadas funciones matemáticas.

Una función consta de una sola instrucción, mientras que una subrutina puede constar de varias instrucciones.

En BASIC una función se forma con un número de proposición, la palabra DEF y la definición de dicha función.

La definición de la función consta del nombre de la función seguida de un signo = y éste a su vez seguido de una constante, una variable o una

expresión; si son necesarios argumentos estos deben aparecer inmediatamente después del nombre de la función encerrados entre paréntesis y separados por comas. Sólo es permisible utilizar variables sin índice como argumentos en la definición de una función.

Mediante la proposición DEF se pueden definir funciones numéricas o alfanuméricas.

Si la función es numérica el nombre de la función debe consistir de tres letras, las primeras dos de ellas deben ser FN, por lo tanto puede haber hasta veintiseis nombres diferentes de funciones numéricas (FNA,FNB...)

Ejemplo

(función numérica) 50 DEF FNA(Z)=3*Z↑2+4

(función alfanumérica) 30 DEF FNA$="FACULTAD DE INGENIERIA"

Aunque una proposición DEF puede aparecer en cualquier lugar de un programa se recomienda colocarlas al inicio del programa.

Una vez que se ha definido una función; para evaluarla es necesario hacer referencia al nombre de la función en el lugar o lugares donde se utilice.

Ejemplo

```
10 DEF FNA(X,Y)=B*X+E*Y↑2
- - - - - - - - - - - - -
- - - - - - - - - - - - -
- - - - - - - - - - - - -
50 LET B=5
60 LET C=10
70 LET D=FNA(S,T)
```

La proposición 70 equivale a:

```
70 LET D=5*S+10*T↑2
```

BIBLIOGRAFIA

Gottfried Byron S.
PROGRAMACION BASIC
Primera edición
Editorial Mc Graw Hill, Serie Schaum's
México, 1977

C S C - U.N.A.M.
BURROUGH'S 6700, BASIC REFERENCE MANUAL
México, 1979

7. PROGRAMACION DE CALCULADORAS DE BOLSILLO

7.1 USO DE LA MAQUINA COMO CALCULADORA

Existen en el mercado dos tipos de calculadoras cuyo modo de operación difiere una de otra. El primero utiliza el sistema normal o estándar y el segundo el sistema conocido como *notación polaca*. Cada calculadora en particular cuenta con muchas ventajas que las hacen muy versátiles.

Para conocer esas ventajas, así como para aclarar cualquier duda que pueda surgir, es necesario consultar el o los manuales que siempre acompañan a las calculadoras.

En general todos los manuales presentan una introducción que es indispensable leer antes de referirse a alguna sección en particular.

Sistema normal

En el sistema normal existen los siguientes grupos de teclas:

- Numerales (0-9)
- Operacionales (+, -, x, ÷)
- Igual (=)
- Funciones de un solo número (seno, coseno, logaritmo, exponencial, cambio de signo, etc.)
- Funciones de dos números (y^x, $\sqrt[x]{y}$, etc.)
- Manejo de memoria (almacenamiento, recuperación, aritmética, borrado)
- Especiales para programación (variable de acuerdo a la marca y modelo de la máquina).
- De borrado (de la pantalla, general, etc.)
- Control de la pantalla (FIX, ENG, EE, etc.)

Muchas calculadoras cuentan con una o dos teclas cuyo objeto es duplicar o triplicar las funciones del resto de las teclas.

Para realizar las operaciones básicas suma, resta, multiplicación y división basta teclar el primer número, el operador (+, -, x, ÷), el segundo número y la tecla del signo igual (=).

Ejemplo 1

expresión $\frac{18}{6} = ?$

teclear: 1
8
÷
6
=

Para efectuar las funciones de un solo número, después de teclear el número, se oprime la tecla de la función; para las funciones de dos números se teclea el primer número (Y), se oprime la tecla de la función, se teclea el segundo número (X) y se oprime la tecla igual (=).

Ejemplo 2

expresión $e^{7.22} = ?$

teclear: 7
.
2
2
e^x

Ejemplo 3

expresión $18^4 = ?$

teclear: 1
8
y^x
4
=

En la mayoría de las máquinas se pueden hacer operaciones en cadena, en ese caso sólo es necesario oprimir la tecla igual al final de la serie de operaciones.

Si se desea evaluar expresiones que contengan paréntesis es necesario calcular primero la expresión entre paréntesis y posteriormente las exteriores; algunas máquinas cuentan con teclas de paréntesis, en cuyo caso las expresiones pueden ser evaluadas tal como están escritas.

Ejemplo 4

expresión: 9X(3 + 2) = ?

máquina sin paréntesis	máquina con paréntesis
teclear: 3	teclear: 9
+	X
2	(
X	3
9	+
=	2
	)
	=

NOTA:

Algunas máquinas contienen sistemas que imponen jerarquías a las operaciones, de manera que, en expresiones de dos o más operaciones, se efectúan primero las de mayor jerarquía sin importar el orden en que fueron tecleadas. Esta jerarquía generalmente realiza primero multiplicaciones y divisiones y después sumas y restas. Se recomienda probar con una expresión sencilla si la máquina que se está utilizando impone jerarquías a las operaciones.

Sistema de Notación Polaca (Inversa).

Los grupos de teclas generalmente son los mismos que en el sistema estándar, con la excepción de la tecla igual, la cual es sustituida por la tecla ENTER (SAVE o simplemente ↑ en algunos casos); sin embargo el sistema operacional en sí, es muy diferente y requiere de un poco de estudio para su comprensión y total aprovechamiento.

Estas máquinas cuentan con una escala numérica, usualmente de cuatro registros llamados, de abajo a arriba, X,Y,Z,T. Estos registros pueden guardar un número cada uno.

Cuando un número se teclea, éste es automáticamente colocado en el registro X.

La tecla ENTER sirve para *hacer subir* a los números por la escala numérica. Cada vez que es oprimida el número en el registro Z es colocado en el T, el del registro Y es colocado en Z y el del registro X es copiado en Y. Si al momento de oprimir la tecla ENTER existe un número en el registro T, éste desaparece.

Para efectuar operaciones aritméticas con dos números es necesario que el primero se encuentre en el registro Y y el segundo en el X.

El proceso es el siguiente:

1. Se teclea el primer número (es colocado en X)
2. Se oprime ENTER (copia el primer número en Y)
3. Se teclea el segundo número (es colocado en X)
4. Se oprime la tecla de la operación.

El resultado es depositado en el registro X.

Ejemplo 1

Expresión $\frac{18}{6} = ?$

teclear: 1
8
ENTER
6
÷

Al terminar la operación el registro Z es colocado en el Y y el T copiado en Z.

Cuando un nuevo número es tecleado inmediatamente después de una operación, automáticamente se realiza la función ENTER sin necesidad de oprimir la tecla, facilitando las operaciones en cadena.

Ejemplo 2

expresión 7x5x2x9 = ?
teclear: 7
ENTER
5
x
2
x
9
x

Las funciones de un solo número se realizan igual que en las calculadoras de tipo estándar: se teclea el número y se oprime la tecla de la función. Para las funciones de dos números es necesario seguir el siguiente proceso: se teclea el primer número (Y), se oprime ENTER, se teclea el segundo número (X) y se oprime la tecla de la función.

Ejemplo 3

expresión	$18^4 = ?$
teclear:	1
	8
	ENTER
	4
	y^x

Aunque en general es sencillo evaluar expresiones que contengan paréntesis utilizando la escala numérica, es recomendable al principio evaluar primero la expresión entre paréntesis.

Ejemplo 4

expresión: $9x(3+2) = ?$

sin utilizar la escala		utilizando la escala	
teclear:	3	teclear:	9
	ENTER		ENTER
	2		3
	+		ENTER
	9		2
	x		+
			x

7.2 USO DE MEMORIAS

En muchas ocasiones, al resolver un problema, es necesario calcular resultados parciales que se utilizan en una etapa posterior de la resolución. Si para resolver el problema se cuenta con una calculadora electrónica, es muy conveniente que ésta tenga uno o más registros en donde se puedan almacenar los resultados parciales para su utilización posterior. Las calculadoras con dichos registros se llaman *con memoria*.

Las calculadoras con memoria(s) disponen de teclas especiales para el manejo de las mismas; dos de ellas siempre deben existir: para almacenar y para recuperar.

Para almacenar un número en alguna de las memorias, éste debe estar en pantalla, ya sea como resultado de alguna operación o tecleado directamente, después se debe oprimir la tecla de almacenamiento (generalmente STO), si la calculadora tiene más de un registro de memoria es necesario indicarle en cuál de ellos deseamos almacenar la información, tecleando el número de dicho registro.

Ejemplo 1

Almacenar 18.23 en el registro 6

Calculadora 'A', teclear:

1 8 · 2 3 STO 0 6

Calculadora 'B', teclear:

1 8 · 2 3 STO 6

Para recuperar el número almacenado en algún registro se debe oprimir la tecla de recuperación (generalmente RCL) y el número del registro en caso de existir más de uno.

Ejemplo 2

Recuperar el contenido del registro 6 de la memoria.

Calculadora 'A', teclear:

RCL 0 6

Calculadora 'B', teclear:

RCL 6

Algunas calculadoras permiten efectuar operaciones aritméticas entre un registro de la memoria y el número en pantalla. Para lograrlo, una vez teniendo en pantalla el número deseado, se oprime la tecla o teclas que realizan la operación y el número del registro de memoria con que se desea operar.

Ejemplo 3

Dividir el contenido de la memoria 6 entre el número en pantalla 5.2

Calculadora 'A', teclear:

5 · 2 INV 2nd Prd 0 6

teclas para dividir

Calculadora 'B', teclear:

5 · 2 STO ÷ 6

teclas para dividir

7.3 ELABORACION Y ALMACENAMIENTO DE PROGRAMAS

Algunas veces se presentan problemas para los cuales es necesario realizar una misma secuencia de instrucciones manejando diferentes grupos de datos, sería conveniente que una calculadora *recordara* dicha secuencia de manera que sea posible utilizarla cambiando únicamente los grupos de datos.

Esta es precisamente la característica que hace distintas a las calculadoras *programables* de aquéllas que no lo son.

Las calculadoras programables cuentan, para cumplir su función, con un grupo de teclas especiales; de éstas las más importantes son:

- La tecla que sirve para indicar a la computadora que empieza o termina la secuencia de pasos de nuestro programa (LRN, switch PRGM/RUN, etc.).
- Tecla para ejecutar o detener la ejecución de los pasos del programa (R/S).
- Tecla para colocar la calculadora al principio de la memoria del programa (RST, RTN, etc.).

Para programar y ejecutar un programa simple es necesario hacer lo siguiente:

A. Colocar la calculadora en el modo que le permite registrar los pasos del programa.

B. Teclear la secuencia de pasos para resolver el problema, tal como se haría manualmente; el último paso debe ser, generalmente, R/S (RUN/STOP) para detener la ejecución del programa.

C. Salir del modo de programación.

D. Colocar la calculadora al principio de memoria del programa.

E. Si el programa requiere un dato, teclearlo.

F. Oprimir R/S para empezar la ejecución de los pasos del programa.

Para ilustrar mejor lo anterior, daremos un ejemplo tal como se haría para dos calculadoras, de las más comunes en el mercado: la calculadora 'A' de tipo estándar y la calculadora 'B' de tipo notación polaca.

Ejemplo 1

Para resolver un problema necesitamos encontrar:

$$\frac{\sin(e^{x}\ \pi)}{\mathrm{Ln}(\sqrt{4x})}$$ donde x es un dato de entrada

Para programarlo en la calculadora 'A'

A. Oprimir LRN para colocar la calculadora en modo de programación.

B. Teclear la siguiente secuencia de pasos: (el número de instrucción es colocado automáticamente por la calculadora; sólo se debe teclear lo que está en la columna *teclear*).

Número de instrucción.	Teclear	Comentarios
000	STO	El número tecleado es
001	00	almacenado en 00
002	x	Se calculará el denominador primero
003	4	
004	=	
005	$\sqrt{x}$	
006	lnx	
007	STO	El denominador es
008	01	almacenado en 01
009	RCL	Se calcula el
010	00	numerador
011	INV	Calcula la inversa del
012	lnx	logaritmo natural (e^x)
013	x	
014	2nd π	
015	=	
016	2nd SIN	Calcula el seno
017	÷	Divide entre
018	RCL	el
019	01	denominador
020	=	
021	R/S	Detiene la ejecución, el resultado queda en la pantalla.

C. Oprimir LRN para salir del modo de programación.

D. Oprimir RST para colocar la calculadora al principio del programa.

E. Teclear el dato de entrada.

F. Oprimir R/S para iniciar la ejecución.

Si el dato de entrada es 5 el resultado debe ser .6409329949

Si se desea ejecutar nuevamente el programa con otro dato se deben repetir los pasos D, E y F.

Para programarlo en la calculadora 'B':

A. Poner el switch PRGM/RUN en la posición PRGM para colocar la calculadora en modo de programación.

B. Teclear la siguiente secuencia de pasos.

Número de instrucción	Teclear	Comentarios
01	STO 0	El número tecleado es almacenado en 0.
02	ENTER	El número tecleado es copiado en el registro Y.
03	4	
04	x	
05	f$\sqrt{x}$	
06	f LN	
07	STO 1	El denominador es almacenado en 1.
08	RCL 0	
09	g e^x	
10	g π	
11	x	Se multiplican e^x y π
12	f SIN	
13	RCL 1	
14	÷	Se divide numerador entre denominador.
15	R/S	

C. Se coloca el switch PRGM/RUN en RUN para salir del modo de programación.

D. Oprimir g RTN para colocar la calculadora al principio del programa.

E. Teclear el dato de entrada.

F. Oprimir R/S para iniciar la ejecución.

Si el dato de entrada es 5, el resultado al igual que en la otra calculadora, es .6409329949; para ejecutar nuevamente con otro dato se repiten los pasos D a F.

Supongamos ahora que se cuenta con más de un dato de entrada, y así mismo con dos o más resultados de salida. Para lograr que la calculadora acepte más de un dato de entrada y/o proporcione más de un resultado de salida, utilizaremos en repetidas ocasiones la tecla R/S.

Para ejemplificar lo anterior pensemos en el problema de convertir de coordenadas cartesianas a polares, en el cual existen dos datos de entrada (x,y) y se obtienen dos resultados de salida (r,θ).

Ejemplo 2

Convertir (x,y) de coordenadas cartesianas a coordenadas polares. (r,θ).

Para la calculadora 'A' la secuencia de pasos sería:

Número de instrucción	Teclear	Comentarios
000	STO	Almacena X en el registro 00
001	00	
002	R/S	Detiene la ejecución para poder teclear Y
003	STO	Almacena Y en el registro 01
004	01	
005	x^2	Eleva Y al cuadrado
006	+	
007	RCL	Recupera X
008	00	
009	x^2	Eleva X al cuadrado
010	=	Completa $x^2 + y^2$
011	$\sqrt{x}$	Calcula $\sqrt{x^2 + y^2}$
012	R/S	Detiene la ejecución para visualizar el valor de r.
013	RCL	Recupera Y
014	01	
015	÷	
016	RCL	Recupera X
017	00	
018	=	Completa $\frac{Y}{X}$
019	INV	Calcula ángulo cuya tangente es $\frac{v}{X}$
020	2nd TAN	
021	R/S	Detiene la ejecución, se obtiene θ

Para cargar y ejecutar el programa anterior se debe:

A. Oprimir LRN

B. Teclear la secuencia de pasos anterior

C. Oprimir LRN

D. Oprimir RST

E. Teclear X

F. Oprimir R/S

G. Teclear Y

H. Oprimir R/S

I. Después de obtener r, oprimir nuevamente R/S para continuar la ejecución y obtener θ.

Para correr con otros datos repetir los pasos D a I.

La instrucción 012 puede ser sustituida por:

012	2nd PAUSE	Se detiene unos instantes, permitiendo visualizar r, y reanuda la ejecución automáticamente sin necesidad de oprimir R/S.

Por supuesto al sustituir de esta manera la instrucción 012 se debe omitir el paso I, pues resulta innecesario.

Para resolver este problema con la calculadora 'B' la serie de instrucciones puede ser:

Número de instrucción	Teclear	Comentarios
01	STO 0	Almacena X en el registro 0
02	R/S	Detiene la ejecución para permitir teclear Y
03	STO 1	Almacena Y en el registro 1
04	gx^2	Calcula Y^2
05	RCL 0	Recupera X
06	gx^2	Eleva X al cuadrado
07	+	Suma $X^2 + Y^2$
08	$f\sqrt{x}$	Calcula $\sqrt{X^2 + Y^2}$
09	R/S	Detiene la ejecución para permitir visualizar r.
10	RCL 1	Recupera Y
11	RCL 0	Recupera X
12	÷	Divide $\frac{Y}{X}$

13	g TAN^{-1}	Calcula ángulo cuya tangente es $\frac{Y}{X}$
14	R/S	Detiene la ejecución proporciona θ

Para cargar y ejecutar estos pasos es necesario:

A. Poner el switch PRGM/RUN en PRGM

B. Teclear los pasos del programa

C. Colocar el switch PRGM/RUN en RUN

D. Oprimir g RTN

E. Teclear X

F. Oprimir R/S

G. Teclear Y

H. Oprimir R/S

I. Cuando se obtiene el resultado r, oprimir R/S para reanudar la ejecución y obtener θ.

Al igual que en la calculadora 'A', se puede sustituir la instrucción 09 por:

09 f PAUSE

Con iguales resultados y debiendo también omitir el paso I.

Para repetir el programa con diferentes datos efectuar los pasos D a I.

Transferencias de control

En los programas anteriores la ejecución de los pasos ha sido secuencial, sin embargo existen muchos problemas en los que es necesario repetir una serie de pasos mientras exista cierta condición o en tanto ésta no exista. Una vez hecho lo anterior se debe continuar con la secuencia lógica del programa.

Pensemos en el caso del factorial; utilizaremos para encontrarlo el siguiente algoritmo:

P_0: Inicio
P_1: lee el número n ($n \geq 0$)
P_2: el factorial es 1
P_3: ¿es n = 0?
    sí: ve a fin
    no: continúa

P_4: Multiplica n por el factorial; asígnalo como el factorial.

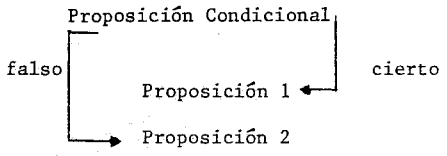
P_5: Decrementa n en 1

P_6: ve al paso P_3

P_2: termina

Vemos que el paso P_3 contiene una condición y que el paso P_6 una transferencia de control al paso P_3

Para programar lo anterior las calculadoras cuentan con varias teclas que sirven para probar condiciones. El esquema general de estas proposiciones es el siguiente:



Al ejecutarse la proposición condicional se obtiene un resultado que sólo puede ser cierto o falso; si el resultado es cierto se ejecuta la proposición 1, en caso contrario, se omite ésta y se ejecuta la proposición 2, continuando de ahí la secuencia lógica del programa.

NOTA:

En algunas calculadoras (en particular la calculadora 'A') la proposición 1 debe ser un número de instrucción o una etiqueta, entendiéndose ésto como una transferencia de control a la instrucción o etiqueta contenida en dicha proposición 1.

La calculadora 'A' cuenta con las siguientes teclas para pruebas condicionales:

$x \leftrightarrows t$ intercambia el contenido de la pantalla con el registro de prueba t.

$x=t$ compara el contenido de la pantalla con el contenido del registro t; produce *cierto* en caso de ser estrictamente iguales.

$x \geq t$ En forma similar al anterior, produce *cierto* cuando el contenido de la pantalla es estrictamente mayor o igual a t.

Con el uso de la tecla INV se puede obtener también:

INV $x=t$ produce *cierto* cuando el contenido de la pantalla y el de t son diferentes.

INV $x \geq t$ produce *cierto* cuando el contenido de la pantalla es estrictamente menor que el de t.

La calculadora 'B', con la ayuda de las teclas f y g, tiene las pruebas siguientes:

$x \leq y$; $x>y$; $x \neq y$; $x=y$; $x<0$; $x>0$; $x \neq 0$; $x=0$

En las cuatro primeras se compara el contenido del registro x con el registro y de la escala numérica; en las cuatro últimas se compara el contenido del registro x con el cero; en ambos casos se produce *cierto* cuando se cumple la condición.

Para efectuar una transferencia de control, ambas calculadoras cuentan con una tecla GTO, ésta al presionarse como secuencia de un programa produce una instrucción, que cuando se ejecuta cambia la secuencia lógica al número de instrucción tecleado a continuación de GTO. El número de instrucción es de tres cifras para la calculadora 'A' y de dos para la 'B'

Ejemplo 3

Calcular el factorial de un número.

La secuencia de pasos del programa para la calculadora 'A' sería:

Número de instrucción	Teclear	Comentarios
000	STO	Almacena n en el registro 00
001	00	
002	1	Almacena 1 en el registro 01
003	STO	(el registro 01 contiene el factorial)
004	01	
005	RCL	Recupera n
006	00	
007	2nd x = t	Compara x con t; como no hemos afectado t, éste contiene cero. En caso de que x = t (n igual a cero) va a la instrucción 023
008	0	
009	23	
010	x	Multiplica n por el factorial.
011	RCL	
012	01	
013	=	
014	STO	Almacena el resultado como nuevo factorial
015	01	
016	1	Resta uno a n
017	INV	
018	SUM	
019	00	
020	GTO	Transfiere la secuencia lógica al paso 005
021	0	
022	05	
023	RCL	Recupera el factorial

024	01	
025	R/S	Detiene la ejecución

Una vez más la secuencia de pasos a seguir para almacenar y ejecutar este programa sería:

A. Oprimir LRN

B. Teclear los pasos del programa

C. Oprimir LRN

D. Oprimir RST

E. Teclear n ($0 \leq n \leq 69$)

F. Oprimir R/S

para ejecutar el programa con otro dato se deben repetir los pasos D a F.

La secuencia de pasos para la calculadora 'B' sería:

Número de instrucción	Teclear	Comentarios
01	STO 0	Almacena n en el registro 0
02	1	Almacena 1 en el registro 1
03	STO 1	(el registro 1 contiene el factorial)
04	RCL 0	recupera n
05	g x = 0	Compara x con cero
06	GTO 13	Transfiere la ejecución a la instrucción 13 en caso de que x sea igual a cero
07	RCL 1	Recupera el factorial
08	x	Multiplica n por el factorial
09	STO 1	El resultado lo almacena como nuevo factorial
10	1	Resta uno al registro 0
11	STO - 0	
12	GTO 04	Transferencia a la instrucción 04
13	RCL 1	Recupera el factorial
14	GTO 00	Transferencia a la instrucción 00

Para cargar y ejecutar el programa se debe:

A. Colocar el switch PRGM/RUN en PRGM

B. Teclear la secuencia de pasos del programa

C. Colocar el switch PRGM/RUN en RUN

D. Oprimir g RTN

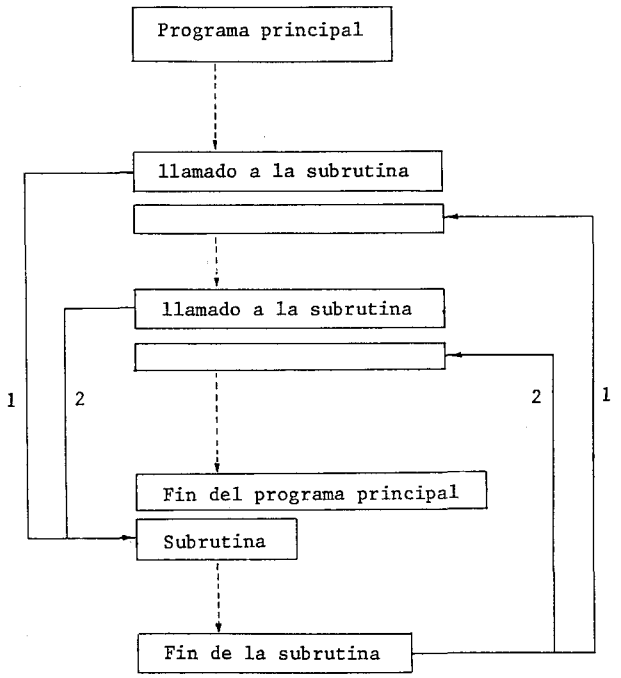
E. Teclear n ($n \leq 0 \leq 69$)

F. Oprimir R/S

Nótese que la última instrucción del programa NO es R/S sino GTO 00; en la mayoría de los casos es mejor utilizar esto último (sólo en la calculadora 'B') ya que GTO 00 transfiere la secuencia a la instrucción 00 y detiene la ejecución. Para ejecutar el programa con otro dato, sólo es necesario repetir los pasos E y F.

Subrutinas

En muchas ocasiones, al resolver un problema, es necesario repetir un grupo de instrucciones cierto número de veces. Para evitar tener que programar varias veces este grupo de instrucciones, la mayoría de las calculadoras programables cuentan con posibilidades de manejar subrutinas, es decir grupos de instrucciones que es necesario repetir varias veces. El esquema general es el siguiente:



Es posible tener más de una subrutina y en algunos casos llamados a otra subrutina desde una subrutina.

La calculadora 'A' cuenta con la tecla SBR para hacer llamados a subrutinas; las teclas INV SBR marcan el fin de la subrutina. La calculadora 'B' tiene la tecla GSB para el llamado y las teclas g RTN para marcar el fin de las subrutinas.

Para ejemplificar el uso de subrutinas utilizaremos el problema de encontrar las combinaciones de n en k ($n \geq k$) mediante la fórmula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

La fórmula incluye el cálculo de tres factoriales, anteriormente se elaboró el programa para el cálculo del factorial de un número, ahora, utilizaremos éste como subrutina.

La secuencia de pasos para la calculadora 'A' podría ser:

Número de instrucción	Teclas	Comentarios
000	STO	Almacena n en el registro de memo-
001	02	ria 02
002	R/S	
003	STO	Almacena k en el registro 03
004	03	
005	SBR	Llama a la subrutina (factorial)
006	0	que empieza en la instrucción 050
007	50	Calcula k!
008	STO	Almacena k! en el registro 04
009	04	
010	RCL	
011	02	
012	-	
013	RCL	
014	03	
015	=	Completa n - k
016	SBR	Llama a la subrutina para calcular
017	0	(n-k)!
018	50	
019	x	
020	RCL	
021	04	

022	=	Multiplica k!(n-k)!
023	STO	Almacena k!(n-k)! en el
024	04	registro 04
025	RCL	Recupera n!
026	02	
027	SBR	Llama a la subrutina para
028	0	calcular n!
029	50	
030	÷	
031	RCL	
032	04	
033	=	Divide $\frac{n!}{k!(n-k)!}$
034	R/S	
050	STO	Empieza la subrutina para calcular el factorial
051	00	
052	1	
053	STO	
054	01	
055	RCL	
056	00	
057	2nd x=t	
058	0	
059	73	
060	x	
061	RCL	
062	01	
063	=	
064	STO	
065	01	
066	1	
067	INV	
068	SUM	
069	00	
070	GTO	
071	0	
072	55	
073	RCL	
074	01	
075	INV SBR	

El colocar a la máquina en la instrucción número 050 después de teclear el programa principal, se puede hacer de dos maneras:

a) Encontrándose la calculadora en el modo de programación, oprimir la tecla SST repetidas veces hasta colocar el programa en la instrucción 050.

b) Oprimir LRN para salir del modo cargar programa, oprimir GTO y 050; después oprimir LRN para continuar cargando el programa.

La secuencia de pasos para la calculadora 'B' sería:

Número de instrucción	Teclear	Comentarios
01	STO 2	Almacena n en el registro 2
02	R/S	
03	STO 3	Almacena k en el registro 3
04	GSB 21	Llama a la subrutina que empieza en la línea 21 (factorial), calcula k!
05	STO 4	Almacena k! en 4
06	RCL 2	Recupera n
07	RCL 3	Recupera k
08	-	Resta n-k
09	GSB 21	Llama a la subrutina
10	RCL 4	Recupera k!
11	x	Multiplica k !(n-k)!
12	STO 4	Almacena k !(n-k)! en 4
13	RCL 2	Recupera n
14	GSB 21	Llama a la subrutina
15	RCL 4	Recupera k !(n-k)!
16	÷	Divide n/k !(n-k)!
17	GTO 00	
21	STO 0	Empieza la subrutina factorial
22	1	
23	STO 1	
24	RCL 0	
25	gx=0	
26	GTO 33	
27	RCL 1	
28	x	
29	STO 1	
30	1	

31	STO-0
32	GTO 24
33	RCL 1
34	g RTN

Después de teclear el programa principal, existen dos formas para encontrar el número de instrucción 21.

a) Dejando el switch PRGM/RUN en la posición PRGM, oprimir la tecla SST sucesivas veces hasta colocar la calculadora en la instrucción 20 y empezar a teclear la subrutina. Se debe colocar la calculadora en la instrucción 20 ya que, a diferencia de la calculadora 'A', siempre muestra la instrucción ya tecleada, no la que se va a teclear.

b) Colocar el switch PRGM/RUN en la posición RUN y teclear GTO 20; se pone el switch PRGM/RUN en PRGM y se teclea la subrutina.

7.4 PROGRAMAS DE BIBLIOTECA

Algunas calculadoras programables cuentan con módulos o tarjetas magnéticas programadas, conteniendo programas de interés general. El objetivo de éstos es ahorrar trabajo al usuario de la calculadora, proporcionándole la solución a problemas comunes.

Es necesario consultar el manual, para saber cuales son los programas de biblioteca con que cuenta una calculadora determinada, y el modo de utilizarlos.

Ejemplo 1

La calculadora 'A' cuenta con un módulo con 25 programas; uno de ellos permite encontrar los elementos de un triángulo, dados tres de ellos, así como su área.

El procedimiento a seguir es el siguiente:

1. Se selecciona el programa para solución de triángulos

   Oprimir 2nd Pgm 12

2. Para encontrar dos lados y un ángulo, conociendo un lado y los ángulos que forman con los otros dos.

   Teclear el lado; oprimir A
   Teclear un ángulo; oprimir B
   Teclear otro ángulo; oprimir C

Para encontrar el tercer ángulo, oprimir 2nd A

Para encontrar el segundo lado, oprimir D

Para encontrar el tercer lado, oprimir E

3. Para encontrar dos lados y un ángulo, conociendo un lado, el ángulo opuesto y otro ángulo.

   Teclear el lado; oprimir A

   Teclear el ángulo opuesto; oprimir B

   Teclear otro ángulo, oprimir C

   Para encontrar el tercer ángulo, oprimir 2nd B'

   Para encontrar el segundo lado, oprimir D

   Para encontrar el tercer lado, oprimir E

4. Para calcular el área del triángulo obtenido en los pasos dos o tres:

   Oprimir 2nd C'

5. Para calcular el área de un triángulo cualquiera, conocidos sus lados:

   Teclear el 1er lado; teclear STO 01

   Teclear el 2o. lado; teclear STO 02

   Teclear el 3er lado; teclear STO 07

   Oprimir 2nd C'

BIBLIOGRAFIA

Texas Instruments, Hewlett Packard
MANUALES PARA CALCULADORAS PROGRAMABLES
U.S.A. 1977, 1978