III. METODOLOGÍA DE LA PROGRAMACION

III.1 DIAGRAMAS DE FLUJO

Un diagrama de flujo es la representación gráfica de cada una de las acciones para resolver un problema. Su uso facilita a terceros la comprensión y/o modificación de la solución planteada para resolver un problema y sirve como documentación formal del programa de computadora que se ha elaborado. Es sin lugar a dudas una ayuda visual independiente del lenguaje de programación y representa una posible solución a un problema, en términos de lógica expresada mediante una secuencia de símbolos gráficos.

La evolución de los diagramas ha ido de la mano con el desarrollo del software y es posible que con la tendencia hacia interfases gráficas y lenguaje hablado para comunicarnos con la computadora, sean de los únicos medios para comunicación que continúen perfeccionando y aplicando para el futuro. A pesar de lo antes mencionado no se cuenta con estándares mundiales para utilizar simbología que, sin embargo, ha aumentado, mejorado y simplificado para convertirse en herramienta formal que explica la solución de un problema dado. Recientemente la ANSI ha emitido ciertas reglas cuyo objetivo es normalizar el uso de los diagramas de flujo.

Es importante remarcar que todo diagrama de flujo obtenido a partir del análisis de un problema involucra un proceso de información, es documentación formal de los programas de computadora.

III.1.1 Un programa es una secuencia de instrucciones mediante las cuales se ejecutan diferentes acciones de acuerdo con los datos que se estén procesando para que la computadora calcule la solución, tomando en cuenta todas las alternativas de proceso a partir de los datos de entrada. Este conjunto de instrucciones se denomina programa fuente cuando el programa cuando el lenguaje de programación es de los denominados de alto nivel. Cuando las instrucciones anteriores se han traducido a lenguaje de máquina (compiladas), se obtiene el programa objeto (en lenguaje de máquina) y el programa ejecutable (al nombrarse se ejecuta).

Un algoritmo es una secuencia ordenada y cronológica de pasos que llevan a la solución de un problema o la ejecución de una tarea . Los pasos deben ser claros y exactos y seguir un orden lógico y, además, tener un principio y un fin.

III.I.I.2 Existen algoritmos numérico y no numérico
Un algoritmo no numérico es el conjunto de instrucciones ordenadas, que indican paso a paso, como obtener el resultado de un problema que no involucra proceso matemáticos. Por ejemplo las instrucciones para cambiar un foco fundido

Pasos
1.- Quitar el foco fundido
2.- Colocar el foco nuevo
Esto parece resolver el problema, pero supóngase que estamos tratando de entrenar un robot para que realice esta tarea en tal caso no funcionará lo anterior, y tendremos que ser mas específicos y claros; por ejemplo:

1. Colocar la escalera debajo del foco fundido
2. Elegir un foco nuevo de la potencia adecuada

3. Subir por la escalera hasta alcanzar el foco fundido
4. Girar el foco fundido hacia la izquierda hasta que esté suelto
5. Ubicar el foco sustituto en el mismo lugar
6. Darle la vuelta hacia la derecha hasta que quede apretado
7. Bajar la escalera
8. Fin

Aun así puede suceder que la precisión no sea suficiente; es probable que algunos pasos deban detallarse de manera tal que las instrucciones sean más simples y precisas.
El algoritmo para reemplazar el foco fundido podrá quedar así:

1.- Colocar la escalera debajo del foco fundido
2.-Elegir un posible sustituto
Si la potencia no es igual a la del foco fundido, repítase el siguiente procedimiento hasta que lo sea.
Descartar el foco escogido
Elegir uno nuevo
3.- Repetir el siguiente procedimiento hasta que se alcance el foco fundido
Subir un peldaño
4.- Repetir el siguiente procedimiento hasta que el foco fundido esta suelto
Girar el foco hacia la izquierda
5.-Ubicar el nuevo foco en el mismo lugar
6.-Repetir el siguiente procedimiento hasta que el foco este apretado.
Girar el foco hacia la derecha
7.- Bajar la escalera
8.-Fin

Observemos que un algoritmo no numérico puede hacerse tan de talladamente o general como se requiera para obtener el resultado deseado.

Un algoritmo no numérico es el conjunto de instrucciones ordenadas para resolver un problema que involucra procesos matemáticos (con calculo de fórmulas de manera repetida).Este tipo de algoritmos no admite ambigüedades y debe darse cada uno de los pasos para su solución. Este tipo de algoritmos se analizarán con detalle.

III.I.2

Aunque la idea de solución de un problema paso a paso mediante símbolos gráficos se hace pensando en no particularizar en algún lenguaje de programación, es necesario conocer la interpretación y uso de los elementos de programación que en su definición de comportamiento, se diferencian de lo que podría entenderse desde el punto de vista algebraico o matemático resultando útil conocer algunas de las reglas de funcionamiento lógico y de interpretación comunes a la mayoría de los lenguajes de alto nivel , con el fin de manejar adecuadamente la prioridad de operaciones y agrupamiento en las instrucciones ejecutables en la computadora.

III.1.2 ELEMENTOS DE UN LENGUAJE DE PROGRAMACIÓN

Los elementos esenciales de todo lenguaje de programación son:

OPERADORES
DECLARACIONES
CONSTANTES
VARIABLES
FUNCIONES DE BIBLIOTECA

Es mediante ellos, y algunos otros elementos de programación básicos, que podemos definir correctamente un buen número de instrucciones o pasos para la solución de un problema con la computadora, desde la entrada de información hasta la salida de resultados.

III.1.2.1OPERADORES

Un operador es un símbolo que indica al indica al compilador que lleve a cabo específicas manipulaciones matemáticas o lógicas. C tiene 3 clases generales de operadores: aritméticos, relacionales, lógicos y a nivel de bits. Además C tiene operadores especiales para determinadas tareas.

Operadores Aritméticos:

+ SUMA
- RESTA (EN C TAMBIEN MENOS MONARIO)
- - DECREMENTO
++ INCREMENTO
MOD RESIDUO DE DIVISIÓN ENTERA
* MULTIPLICACION
\ DIVISIÓN ENTERA (EN C SE USA EL CARÁCTER %)
/ DIVISIÓN
^ EXPONENCIACION (EN C SE USA UNA FUNCION)

El menos monario hace el efecto de multiplicar su único operando por –1. O sea cualquier número precedido por un signo menos, cambia de signo.
Operaciones relacionales y lógicos

En el término operador relacional la palabra relacional se refiere a la relación entre unos valores y otros. En el término operador lógico la palabra lógico se refiere a las formas en que esas relaciones pueden conectarse entre sí.

Operadores Relacionales

Operador	Acción
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
= =	Igual
¡=	No igual

Operadores Lógicos

Operador	Acción
&&	(Y)
II	(O)
¡	(NO)

Tanto los operadores relacionales como los lógicos tienen un nivel de precedencia menor que los operadores aritméticos.

La Tabla que sigue la precedencia relativa entre los operadores relacionales y lógicos:

mayor ¡
>>=<<=
= = ¡ =
&&
menor II

Como en el caso de las expresiones aritméticas, es posible usar paréntesis para alterar el orden natural de evaluación en una expresión relacional o lógica por ejemplo:

0II!0&&1

es true por que el ! se evalúa primero y luego se evalúa el && antes que el II. Sin embargo, cuando se añaden paréntesis ala misma expresión, como se muestra a continuación el resultado es falso.

(0II!0)&&1

Todas las expresiones relacionales o lógicas producen como resultado un 0 o un 1. Por tanto, el siguiente fragmento de programa no es solo correcto , sino que imprimirá el número 1 en la pantalla:

```
int x;
x= 100;
printf("%d", x>10);
```

Operadores a nivel de bits

Las operaciones a nivel de bits se refieren a la comprobación asignación o desplazamiento de los bits reales que componen un byte o una palabra las operaciones sobre bits son frecuentes en aplicaciones de controladores de dispositivos –programas de modem, rutinas de disco de archivo, y rutinas de impresora- debido a que permiten enmascarar ciertos bits, como el de paridad(el byte de paridad se utiliza para confirmar que el resto de los bits del byte no han cambiado.)

La prioridad de los operadores en casi todos los lenguajes para la evaluación de las expresiones aritméticas, involucrando funciones del programa y de biblioteca, de derecha a izquierda es la siguiente:

FUNCIONES
EXPONENCIACION

MULTIPLICACIÓN O DIVISIÓN DE REALES
DIVISIÓN ENTERA
MOD
SUMA O RESTA

Los paréntesis redondos se utilizan como elementos de agrupamiento y siempre que existe un par o más de dichos paréntesis, el contenido de éstos se evalúa primero, desde el más interno, hasta el mas externo. Un ejemplo de aplicación es cuando en una expresión existe un numerador y un denominador que constan de más de un término.

	→
Mayor	() []
	! ~ + + -- - (tipo)* & sizeof
	* / %
	+ -
	< < > >
	<<=> >=
	= = ! =
	&
	^
	&&
	?
	= + = - = *=/ =
Menor	,

DECLARACIONES

Las instrucciones de un programa se dividen en ejecutables y no ejecutables. Estas ultimas las declaraciones no ejecutables proveen a la computadora la información necesaria sobre las características de las variables, especificaciones para formatos de salida etc.

CONSTANTES

Las constantes en C se refieren a valores fijos que no pueden ser alterados por el programa. Pueden ser cualquier tipo de datos y se dividen en:

ENTERAS
REALES
ALFANUMERICAS

A su vez, dentro de los dos primeros grupos, las constantes pueden catalogarse en:

DE PRECISION SENCILLA
DE DOBLE PRECISION

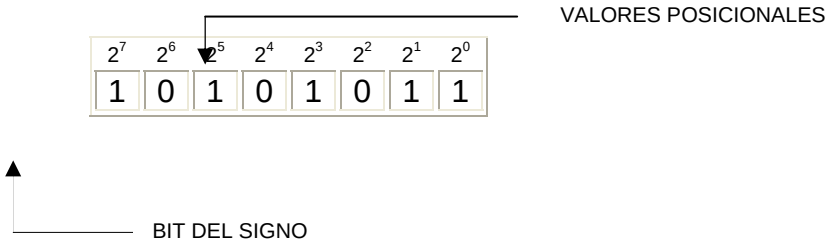
DEF

CONSTANTE ENTERA: Cantidad que carece de punto y dígitos decimales, solo admite signo (positivo o negativo).

CONSTANTE REAL: Cantidad que incluye punto y dígitos decimales, sólo admite signos positivo y negativo.

Las constantes reales permiten representar algunos números racionales o cantidades con fracción definida exactamente, en razón de las limitaciones de su memoria y la forma en la que se almacenan . Curiosamente una fracción real se almacena en forma parecida a una cantidad representada en notación exponencial, es decir, hay una parte entera y un exponente. La precisión simple de las cantidades reales garantiza por lo general seis dígitos decimales exactos, la doble precisión el doble mas o menos.

Una computadora almacena datos en su memoria primaria utilizando dígitos binarios (cero o uno) agrupados en bytes para formar con eso, a su vez, palabras de computadora. De esta manera, una constante entera requiere de uno o más bytes para la representación de su valor. Es así que la cantidad puede imaginarse representada por un conjunto de focos que representan individualmente uno de dos estados prendido o apagado es decir uno o cero. De este grupo de bytes la primera posición de la izquierda está reservada para indicar si la cantidad es negativa (valor unitario) o positiva (valor nulo), los demás bits indican un valor posicional.



III.1.2.4 VARIABLES

Una variable es un nombre que empieza con una letra y puede estar formada con varios caracteres más (por lo general sólo son válidos dígitos, letras y el subguión_)
Un primer paso, en C consiste en declarar el tipo de todas las variables empleadas.
En C existen cuatro tipos de variables básicas:

TIPO	MEMORIA	RANGO
Char (tipo carácter)	8 bits	De 0 a 255
Int (variable tipo entera)	16 bits	-32768 a 32767
Float (tipo punto flotante)	32 bits	
Double (doble magnitud)	64 bits	1.7E-308 a 1.7E +308

Estas variables básicas pueden ser alteradas por medio de modificadores, los cuales, combinados con ellas, generan nuevos tipos de datos.

La variable tipo char se utiliza para almacenar caracteres tipo letras, símbolos puntuación etc.

Inicialización de una variable
El término inicialización significa asignar a la variable un valor inicial.
FUNCIONES DE BIBLIOTECA
Una función de biblioteca es un proceso subprogramado , identificado con un nombre de una letra de uno a cinco caracteres más seguido de argumentos entre paréntesis que asocia a él los valores definido por argumentos un valor unico en respuesta al llamado de la función

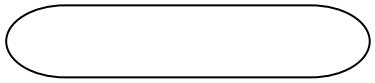
III.1.3 SIMBOLOGIA BASICA DE LOS DIAGRAMAS DE FLUJO

Todo tipo de algoritmos se puede representar gráficamente mediante símbolos previamente definidos que indican las características básicas de un algoritmo:

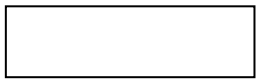
1.Inicio
2.-Datos de Entrada
3.Proceso de Información de Entrada
4.-Salida
5.-Fin

Todo algoritmo debe de estar perfectamente definido, con un número finito de pasos y que obtenga la solución del problema.

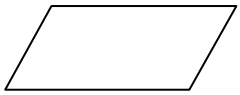
Este tipo de símbolos se asocia con la idea general de la programación estructurada en la que la solución de un problema se muestra con una trayectoria de solución del inicio al fin es decir en diseño ascendente o top- down y , de esta manera se plantea de forma más fácil la solución de un problema.



Marca el inicio y fin de un programa



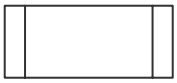
Proceso. Son acciones que el programa tiene que realizar



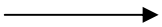
Entrada / Salida. Nos permite ingresar datos, de un periférico, así como mostrarlos

Salida. Es usado para mostrar datos o resultados

Conector. Se coloca al principio y fin de un pedazo de programa, enlaza dos partes cualquiera de un programa.



Llamado a subrutina(una subrutina es un modulo independiente del programa

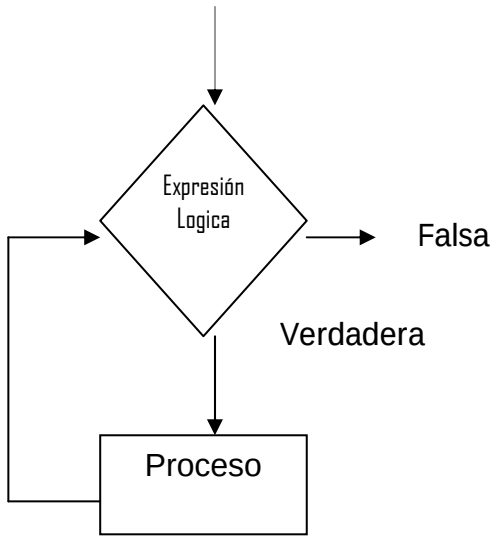


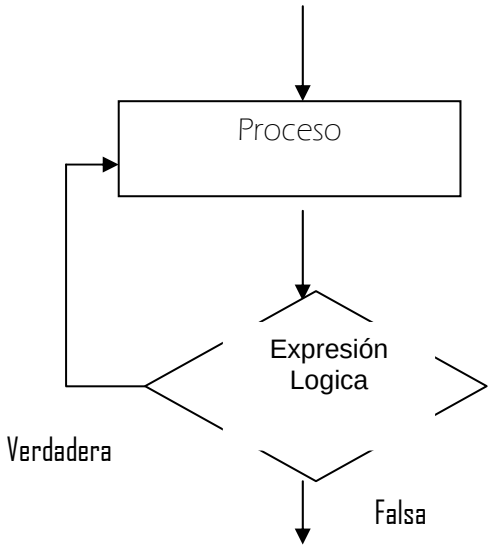
Línea de flujo o indicador de dirección

III.1.4 EL CICLO ITERACTIVO

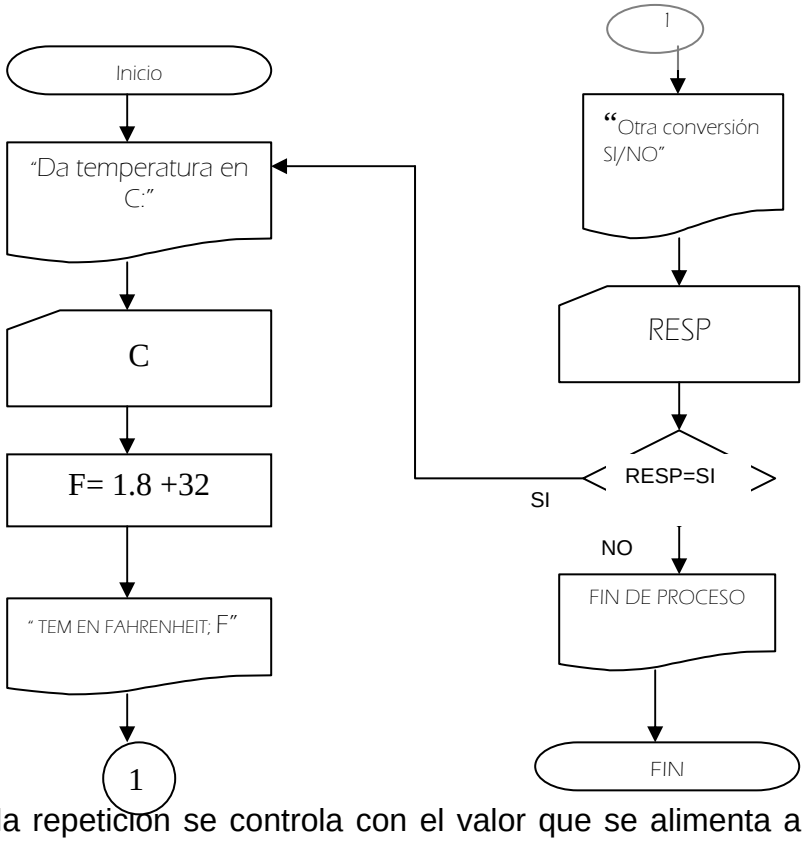
Se entiendo por ciclo iterativo la unión de símbolos gráficos que en su conjunto indica como repetir cierto proceso; las estructuras básicas más comunes son:

Observamos





Observamos que mediante una pregunta, se opta por repetir n veces el proceso que se indica en el rectángulo, es evidente que en las labores humanas existe la necesidad de repetir una y otra vez una serie de actividades en forma cíclica. Usar un ciclo es básico en la solución de múltiple problemas que repiten procesos



En este caso la repeticion se controla con el valor que se alimenta a la variable alfanumérica RESP cuando se lee algo distinto de si el proceso termina.

III.1.5APLICACIONES DEL CICLO ITERATIVO
EL CONCEPTO DE CONTADOR

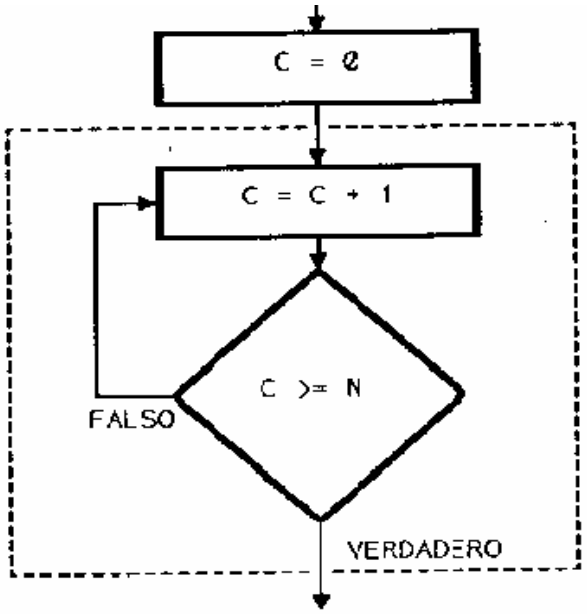
Es una variable de tipo entero que tiene como función contener el numero de veces que esa variable se procesa.

De la interpretación que se hace de la instrucción de reemplazo: calcular el valor de la expresión y almacenar la variable indicada a la izquierda del signo igual, se genera el concepto de contador o expresión que permite contar (que consta de un ciclo iterativo que ejecuta varias veces la instrucción de reemplazo: C=C+1):

C=0 (El marco que delimita la instrucción representa un ciclo que implica repetir lo que contiene)

C=C+1

A lo que corresponde el siguiente diagrama de flujo, elaborando con simbología tradicional o básica:



Considere un caso particular con N=4, es decir, con cuatro iteraciones donde notacionalmente se indica subrayado el valor anterior del almacén y enseguida el nuevo término que se agrega en cada vuelta (controlada por una condición).

Análisis del funcionamiento o prueba de escritorio

C=0 (VALOR INICIAL)

Primera iteración (primera vez que se ejecuta el ciclo)

C = 0 + 1 = 1 (Se almacena el valor anterior de C(cero) más la constante 1.Es decir C=C+1)

Segunda iteración:

C= 1 + 1 = 2 (Se almacena el valor que se acaba de calcular de C más la constante 1, es decir C=C+1).

Tercera iteración

C = 2+ 1 = 3 (Se almacena el valor imediato de C más la constante 1, es decir C=C+1).

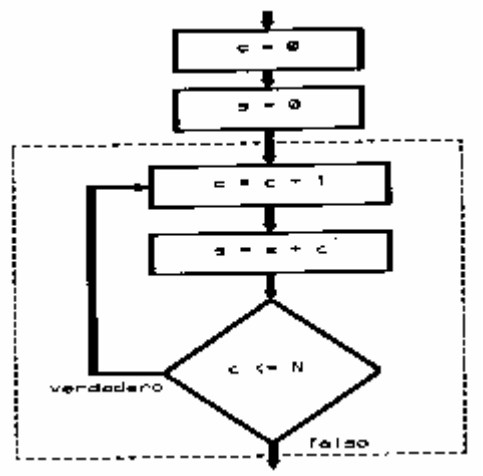
Cuarta iteración

C = 3+ 1 = 4 (valor anterior de C más la constante 1: C=C+1).

III.1.5.2 SEGUNDA APLICACIÓN EL CONCEPTO DE ACUMULADOR

Es una variable de tipo numerico cuya función es contener la suma de un determinado conjunto de datos.

La forma de operar es la siguiente. Al principio se inicia con cero, dentro del ciclo se incrementa con lo que contenga la variable que contiene el dato a acumular, después del fin del ciclo el contenido del acumulador que es el total de la suma.



C=0
S=0

C=C+1
S=S+C

La definición de acumulador estructuralmente es similar a la del contador, con la diferencia de que ahora se acumula en cada iteración un nuevo término dado por una variable o contador (considérese el ciclo iterativo que permite calcular la sumatoria siguiente: S = 1+2+3+4; :

Valores iniciales para entrar al ciclo iterativo

C=0 el contador no ha contado nada
S=0 el valor inicial de la suma

Entra a ejecutar el ciclo (primera iteración)
C = 0 + 1 = 1 (se cuenta la primera vuelta: C = C + 1)

S= 0 + 1 (valor anterior de la suma más el primer término: S = S + primer término)

Entra a la segunda ejecución (segunda iteración)
C = 1 + 1 = 2 (se cuenta la primera vuelta: C = C + 1)

S= 1 + 2 (se acumula el segundo término S= S + C)

Entra a la tercera ejecución (tercera iteración)

C = 2 + 1 = 3 (se cuenta la tercera vuelta: C = C + 1)

S= 1 + 2 + 3 (Valor anterior de la suma más tercer término: S = S +C)

Entra a la cuarta ejecución (cuarta iteración)

C = 3 + 1 = 4 (se cuenta la cuarta vuelta: C = C + 1)

S= 1 + 2 + 3 + 4 (Valor anterior de la suma más el cuarto término: S = S +C)

De igual manera se sigue repitiendo (iterando) para calcular n términos, según se indicó mediante el diagrama de flujo anterior.

III..5.3 TERCERA APLICACIÓN: EL CONCEPTO DE MULTIPLICACIÓN REITERADA FACTORIALES

Otro elemento útil para al cálculo de fórmulas es el que resulta de aplicar el concepto de reemplazo de valor de la variable a lo que se llamará una multiplicación reiterada, la cual permite obtener factoriales de tipo descendente o ascendente. Por ejemplo:

Obtener el factorial de 5!

Se sabe que su factorial ascendente es: 1x2x3x4x5

De dicha multiplicación se observa la presencia del elemento llamado contador, que multiplica su valor por el de la variable del factorial obtenido en el paso anterior, en efecto:

F= 1 (VALOR INICIAL DE LA VARIABLE DEL FACTORIAL)

F= 1x2 (VALOR ANTERIOR DE F MULTIPLICADO POR 2, ES DECIR, F =FX2)

F=1X2X3 (VALOR ANTERIOR DE F MULTIPLICADO POR 3, ES DECIR, F = F X 3)

F=1x2x3x4(VALOR ANTERIOR DE F MULTIPLICADO POR 4, ES DECIR, F =F X 4)

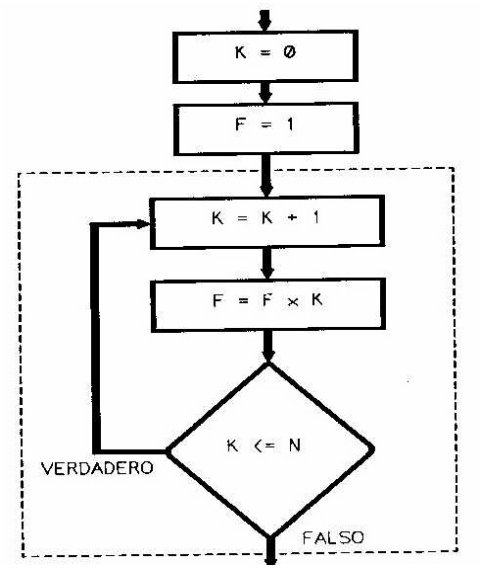
F=1x2x3x4x5 (VALOR ANTERIOR DE F MULTIPLICADO POR 5, ES DECIR, F= FX5)

Apoyándose en el concepto de almacén y en la estructura previamente obtenida para el contador, se obtiene el algoritmo para la multiplicación sucesiva o factorial para enteros positivos a partir de la unidad, que en forma simplificada se representa:

K=0
F=1

(Indicamos que el proceso se ejecutará n veces al iterar)
K=K+1
F=FxK

Lo cual representado en diagrama de flujo queda:



En esta aplicación elemental el ciclo iterativo básico siempre involucra dos elementos esenciales:

Un contador que controla el número de vuelta realizadas.
Una pregunta que relaciona al contador con una variable que indica el número máximo de vueltas que deberá realizar el ciclo, esto permite un regreso o GOTO o una terminación del proceso de acumulación.

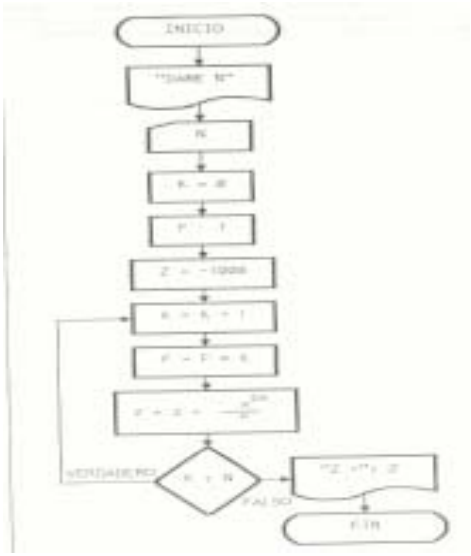
Otra observación consiste con base en la pregunta se catalogan dos tipos de ciclos: el controlado por contador y el ciclo iterativo controlado por una iteración de tipo alfanumérica, este último nos permite elaborar diagramas para procesos conversacionales de computadora.

Utilizando estos conceptos básicos, se facilita la solución de problemas como el cálculo de la serie:

Donde se observan los siguiente elementos de programación

* Un contador (para generar los exponentes y para contar los N términos de la serie que se calculan)
* Una multiplicación sucesiva para obtener el factorial
* Un sumador

El diagrama de flujo que considera en ese orden de aparición los elementos mencionados para calcular la serie es el siguiente :



Con base en la estructura de ciclo iterativo y el concepto de almacén, el contador, el factorial y el sumador llevan a efecto su tarea; veámoslo como una prueba de escritorio

N=4

Valores iniciales para entrar al ciclo iterativo

K=0 (el contador no ha contado nada)
F=1 (valor inicial de la variable del factorial)
Z= 1000 (valor inicial de la suma)

Notacionalmente se indica subrayado el valor anterior del almacén y enseguida la operación que lo modifica:

Se ejecuta el ciclo (primera iteración)
K= $\underline{0}$+1 (se cuenta la primera vuelta: K = $\underline{K}$ + 1)
F= $\underline{1}$*1 (se calcula el primer factorial: F = $\underline{F}$ * K)

Z= -$\underline{1000}$ + $\frac{X^{2(1)}}{1!}$ (valor anterior de la suma más el segundo término: Z= Z+ segundo término)

Segunda ejecución (segunda iteración)
K= $\underline{1}$ + 1 (se cuanta segunda vuelta: K = $\underline{K}$ + 1)
F = $\underline{1}$ * 2 (segundo factorial: F = $\underline{F}$ * K)

Z= -1000 + $\frac{X^{2(1)}}{1!}$ + $\frac{X^{2(2)}}{2!}$ (se agrega a la suma el tercer término: Z = $\underline{Z}$ + tercer término)

Tercera ejecución (tercera iteración)
K= $\underline{2}$ + 1 (se cuenta la tercera vuelta: K = $\underline{K}$ + 1)
F = $\underline{1 * 2}$ * 3 (tercer factorial: F = F * K)

Z= -1000 + $\frac{X^{2(1)}}{1!}$ + $\frac{X^{2(2)}}{2!}$ + $\frac{X^{2(3)}}{3!}$ (agregamos el cuarto término: Z = $\underline{Z}$ + cuarto término)

De igual manera se puede continuar repitiendo la operación para calcular n términos.
La simbología elemental de diagramas de flujo permite presentar paso a paso todas las instrucciones necesarias para entender los conceptos de :
Contadores
Sumatorias
Factoriales

Con la utilización de ciclos iterativos elementales que se han formado mediante el uso de instrucciones especificas de regreso conocidas en la programación como GOTOS.

III.I.6 DIAGRAMACION ESTRUCTURADA

CICLO ITERATIVO CONTROLADO POR UN CONTADOR (FOR)

Este tipo de repetición es útil para controlar ciclos en los que se conoce de antemano el número de veces, que se deberán ejecutar las acciones que están dentro del ciclo. Esto es por que se controla con un contador, que toma desde un valor inicial, hasta un valor final con un incremento.

Ejemplo elaborar un algoritmo que imprima los números del 1 al 10.

Este es un problema natural para al FOR, por que se conoce cuántas veces se repetirá.

Algoritmo imprime 1 –10

1. Declaraciones
    a. Variables
    b. I:Entero
2. FOR I= 1,10,1}
    a. Imprimir I

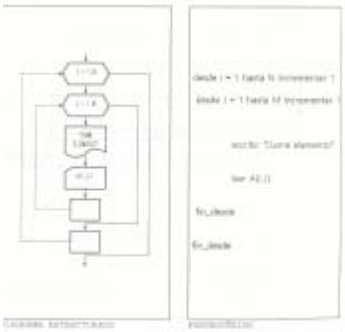
3.ENDFOR

4.FIN

Se plantea el ciclo FOR desde I =1 hasta 10 con incrementos de 1; cada vez que entra al ciclo imprime el valor de I, entonces imprimirá: 1 2 3 4 5 6 7 8 9 10.

III.1.6.1.2 CICLOS ANIDADOS CON CONTADOR INCLUIDO
MANEJO DE ARREGLOS DE DOS DIMENSIONES

Cuando una de estas estructuras iterativas contiene a otra, se dice que son ciclos anidados. En ellos, por cada valor de la variable del ciclo externo, la variable del ciclo interno varía completamente, generándose dos tipos de variación en los contadores: la externa o lenta y la interna o rápida. Este tipo de situación resulta adecuada para el manejo de arreglos de varias dimensiones, por ejemplo la lectura de un arreglo matricial por renglones:



I = 1,3
J=1,4

A(I;J)

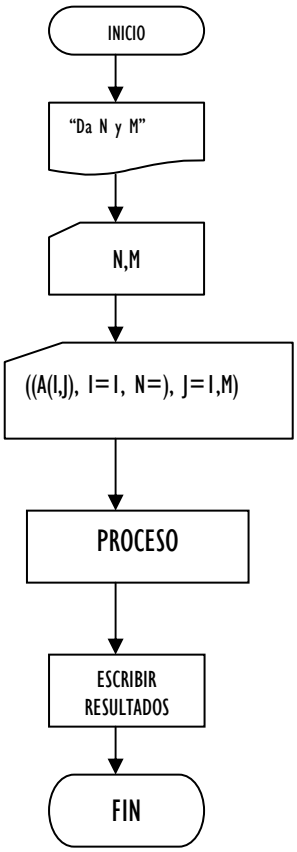
En este caso como ya se menciono, por cada valor de I , la variable J toma los valores de 1 a 4, es decir se tiene:

Con I=1, el valor de J varía de 1 a 4 y se leen los elementos:
A(1,1), A(1,2), A(1,3), A(1,4),

Con I=2, el valor de J varía de 1 a 4 y se leen los elementos del segundo renglón:
A(2,1), A(2,2), A(2,3), A(2,4),

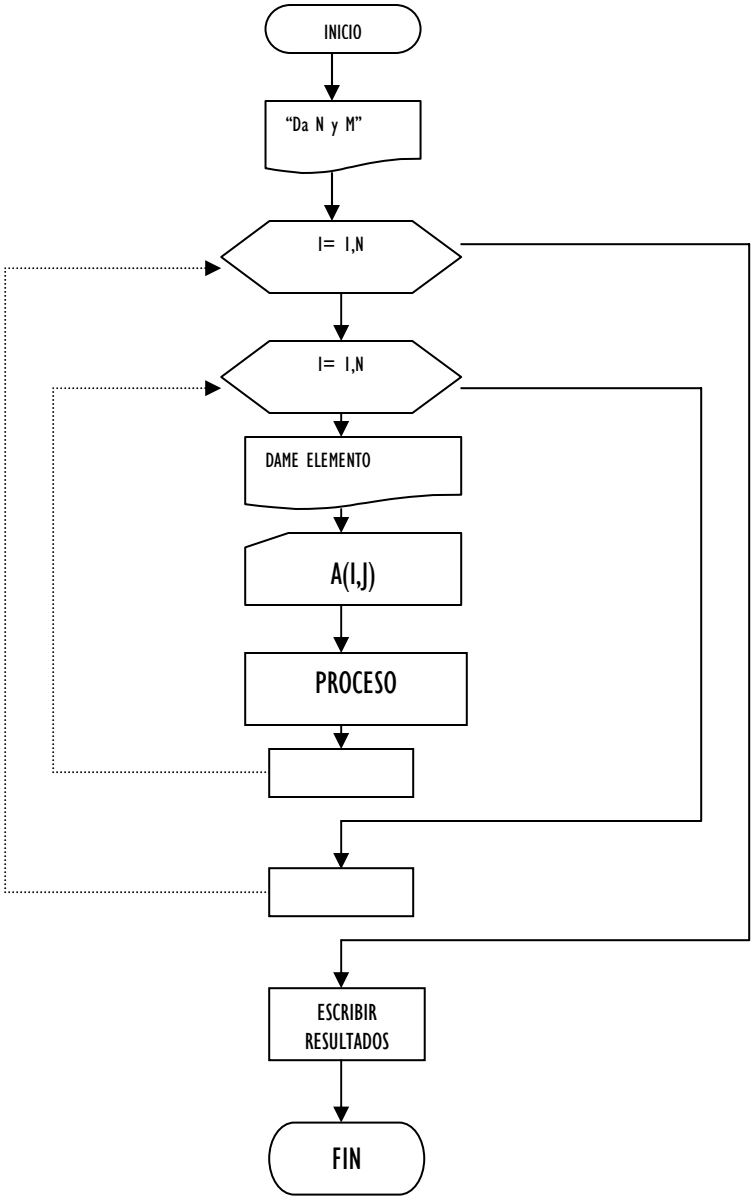
Con I=3, el valor de J varía de 1 a 4 y se leen los elementos del segundo renglón:
A(3,1), A(3,2), A(3,3), A(3,4),

De esa manera, puede pensarse que los contadores de los ciclos han generado dos tipos de variaciones entre sí:
-Una lenta asociada a la variable I
-Una rápida asociada a la variable J
Aunque la mayoría de lenguajes requiere de dos explícitos para la lectura de este tipo de arreglos, en la simbología de diagramas de flujo se opta por indicarlos de manera implícita.
La forma implícita más compacta que se emplea es la siguiente:



Es importante tener en cuenta que, en la mayoría de los lenguajes de programación, debe codificarse conforme a la sintaxis de los ciclos anidados explícitos.

La representación gráfica de la lectura de un arreglo de dos dimensiones mediante ciclos iterativos explícitos queda:



El uso de ciclos iterativos anidados resulta esencial para la solución de problemas con arreglos en los que se involucra dos o más variaciones de índices, por ejemplo en el caso de la multiplicación de matrices:

Dos matrices pueden multiplicarse si el número de elementos por renglón de la primera es igual al número de elementos por columna de la segunda. Con estas condiciones considere el producto de dos matrices de 2 x 3 y de 3 x 3.

$$\begin{bmatrix} A(1,1) & A(1,2) & A(1,3) \\ A(2,1) & A(2,2) & A(2,3) \end{bmatrix} \times \begin{bmatrix} B(1,1) & B(1,2) \\ B(2,1) & B(2,2) \\ B(3,1) & B(3,2) \end{bmatrix} =$$

Los elementos del primer renglón de la matriz resultando están formados por las sumatorias

C(1,1) = A(1,1) X B(1,1)+ A(1,2) x B(2,1) + A(1,3) x B(3,1)
C(1,2) = A(1,1) X B(1,2)+ A(1,2) x B(2,2) + A(1,3) x B(3,2)

Los elementos del segundo renglón:

C(2,1) = A(2,1) X B(1,1)+ A(2,2) x B(2,1) + A(1,3) x B(3,1)
C(2,2) = A(2,1) X B(1,2)+ A(2,2) x B(2,2) + A(2,3) x B(3,2)

De este desarrollo se observa la presencia de tres contadores y, por lo tanto, la necesidad de usar tres ciclos iterativos anidados para obtener:

-una variación lenta de un índice si éste está fijo en todas las sumatorias o elementos de un renglón y se modifica al cambiar de renglón.
-una variación intermedia si un índice permanece fijo entre los elementos de una sumatoria y se modifica la cambiar de sumatoria
-una variación rápida cuando el índice cambia para definir las variables que determinan los productos de la sumatoria.

De la inspección de las sumatorias, se observa que en todo el primer renglón, el primer índice de A permanece fijo en 1 y cambia a 2 para el segundo renglón. Esto implica una variación lenta; de igual forma el primer índice de C:

Primer renglón de la matriz:
A(1,) X B(,)+ A(1,) x B(,)+A(1,) x B(,) A(1,) X B(,)+ A(1,) x B(,)+A(1,) x B(,)
Segundo renglón
A(2,) X B(,)+ A(2,) x B(,)+A(2,) x B(,) A(2,) X B(,)+ A(2,) x B(2,)+A(2,) x B(,)
Los índices de variación intermedia son los que varían de sumatoria a sumatoria en el mismo renglón. Estos son el segundo índice de B y el segundo de C;:

Primer renglón:
A(,) X B(,1)+ A(,) x B(, 1)+A(,) x B(, 1) A(,) X B(,2)+ A(,) x B(, 2)+A(,) x B(,2)
Segundo renglón
A(,) X B(, 1)+ A(,) x B(, 1)+A(,) x B(, 1) A(,) X B(,2)+A(,) x B(,2)+A(,) x B(,2)

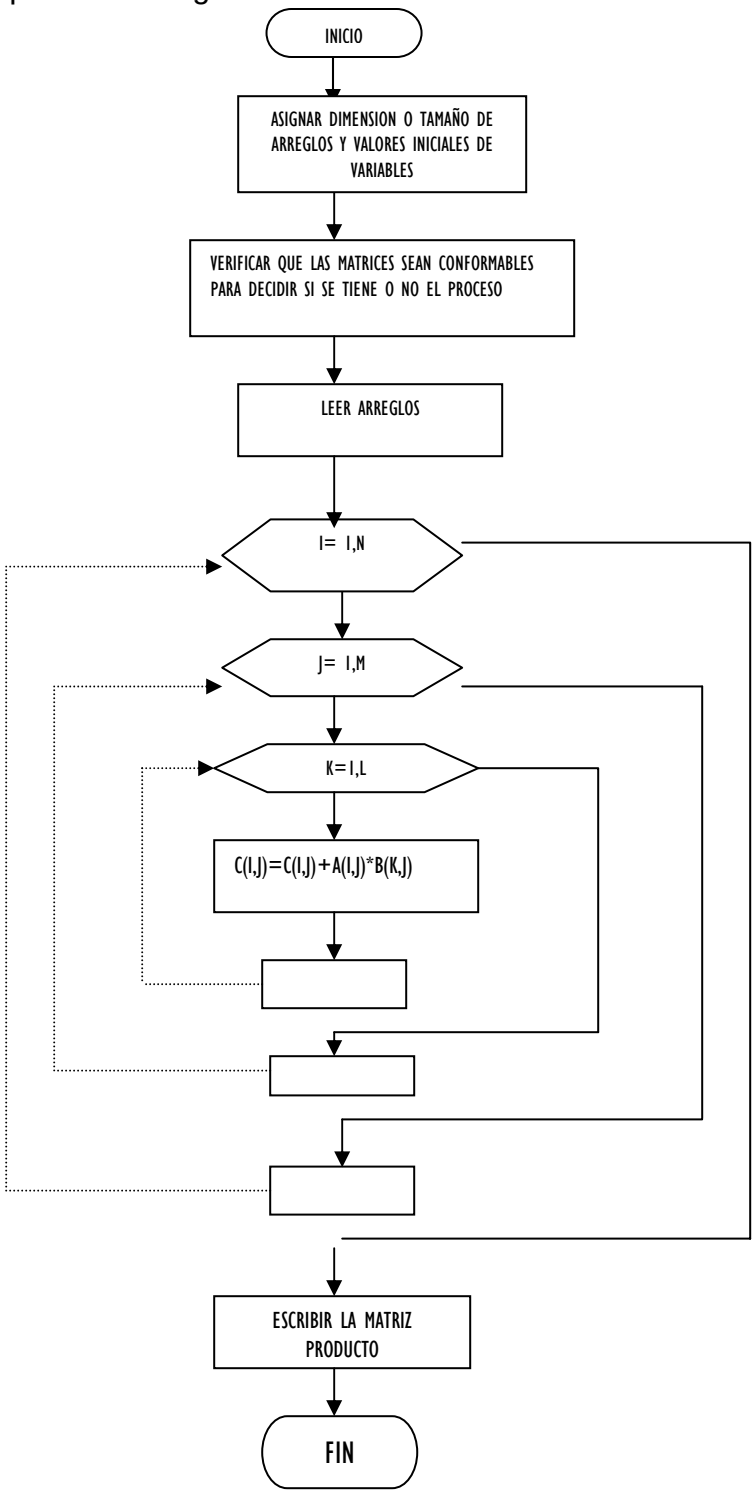
Por ultimo se observa que le Segundo índice de A cambia de 1 a 3 en cada una de las sumatorias, al igual que el primer índice de B; esto implica una variación rápida:

Primer renglón:
A(, 1) X B(1 ,)+ A(, 2) x B(2 ,)+A(, 3) x B(3 ,) A(, 1) X B(1,)+ A(, 2) x B(2,)+A(,3) x B(3 ,)
Segundo renglón
A(,1) X B(1,)+ A(,2) x B(2 ,)+A(,3) x B(3,) A(,1) X B(1,)+A(,2) x B(2,)+A(, 3) x B(3 ,)

Es más o menos obvio que con el ciclo que presenta la variación rápida, también se obtiene la sumatoria con la cual cada elemento de la matriz resultante, por lo tanto el logaritmo que define la multiplicación es:

I=1,2
J=1,2
K=1,3
C(I,J) =C(I,J) +A(I,J) * B(K,J)

Por lo que el diagrama de flujo para el caso general es:



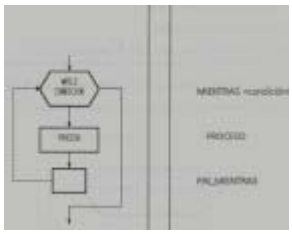
III.1.6.1.3CICLOS ITERATIVOS CONTROLADOS POR UNA CONDICIÓN

Los ciclos iterativos son la herramienta de programación que permite repetir procesos. En la práctica muchos procesos deben repetirse MIENTRAS se cumpla una condición, en cambio otros procesos deben repetirse HASTA que se cumpla una condición.

La primera opción implica usar una condición antes de realizar el proceso, la segunda implica realizar la condición al final del proceso

CICLO CON CONDICION AL INICIO (CICLO WHILE O MIENTRAS)

Este tipo de ciclo sólo se ejecuta cuando se cumple la condición de la pregunta implícita colocada al inicio. Su forma es la siguiente:



El diagrama de la izquierda indica el proceso enmarcado en el rectángulo se ejecutará mientras se cumpla la condición señalada al principio del ciclo iterativo y que el ámbito del ciclo esta delimitado por la instrucción dada en el último rectángulo, que en diagramas de flujo siempre se dejará en blanco (fin del ámbito del WHILE)

En este caso, el último recuadro o última instrucción del ciclo, esta asociada con lai nstrucción específica que define el ciclo WHILE)

Comparando con el diagrama de la derecha, se nota una menor cantidad de símbolos gráficos y, por lo tanto creemos que es más fácil de emplear este diagrama estructurado para el ciclo. EN dicho símbolo la línea punteada de la izquierda que el ciclo se repite automáticamente, mientras que en el diagrama tradicional es necesario indicar explícitamente que el proceso se repite

De forma reducida podemos indicar lo anterior como:

WHILE condición

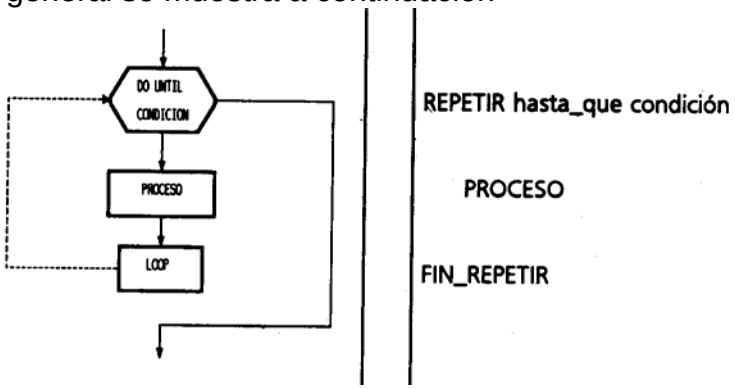
PROCESO

La condición por lo general involucra dos variables que permiten romper el ciclo, por ejemplo un contador incluido en el ámbito del ciclo, cuyo valor se compara con un valor dado: WHILE i < n.
De esa manera se controla cuantas veces se realiza el proceso. Una variante es emplear en la condición de variables de tipo alfanumérico para controlar si se sigue ejecutando el ciclo iterativo.

CICLO CON CONDICION WHILE/UNTIL

En algunos lenguajes de alto nivel, se cuenta con elementos estructurados para ciclo iterativos y en ellos se permite colocar las situaciones WHILE o mientras y la UNTIL o hasta al inicio o al final del ciclo, con lo que brindan mayo flexibilidad a la programación.

El diagrama del caso general se muestra a continuación



Primera opción (WHILE o mientras al inicio)

El ciclo iterativo con la opción WHILE al inicio del ciclo permite que el ciclo no se ejecute ni una sola vez, cuando no se cumple la condición.

La figura para ese caso indica que el proceso enmarcado en el rectángulo se ejecutará mientras se cumpla la condición señalada al principio del ciclo iterativo y que el ámbito del ciclo está delimitado por la instrucción dada en el último rectángulo (LOOP)

De forma reducida se indica lo anterior como:

```
DO WHILE condición
PROCESO
LOOP
```

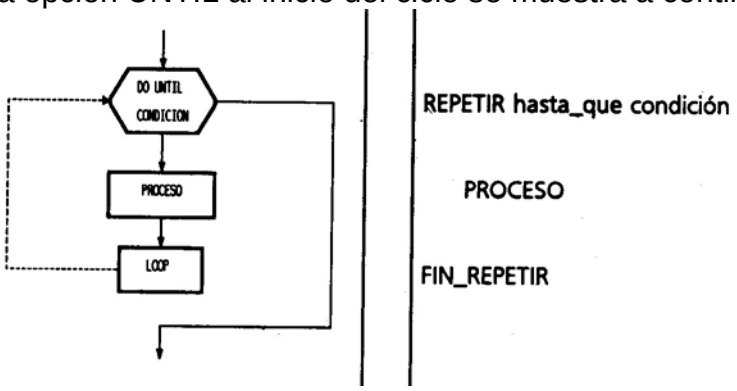
En resumen el ciclo DO LOOP, con la condición WHILE, se interpreta como : hacer el ciclo MIENTRAS se cumpla la condición.

La condición por lo general involucra dos variables que permiten romper el ciclo: DO WHILE i<n. De esta manera se controla las veces que se realiza el proceso.

Ejemplo de aplicación:
Considere el diagrama para calcular el factorial descendente de un número positivo no nulo.

Segunda opción (DO UNTIL-LOOP)

El ciclo iterativo con la opción UNTIL al inicio del ciclo se muestra a continuación:



Esto indica que le proceso en marcado en el rectángulo se ejecutará hasta que se cumpla la condición señalada al principio del ciclo iterativo y que el ámbito del ciclo está delimitado por la instrucción dada en el último rectángulo (LOOP).

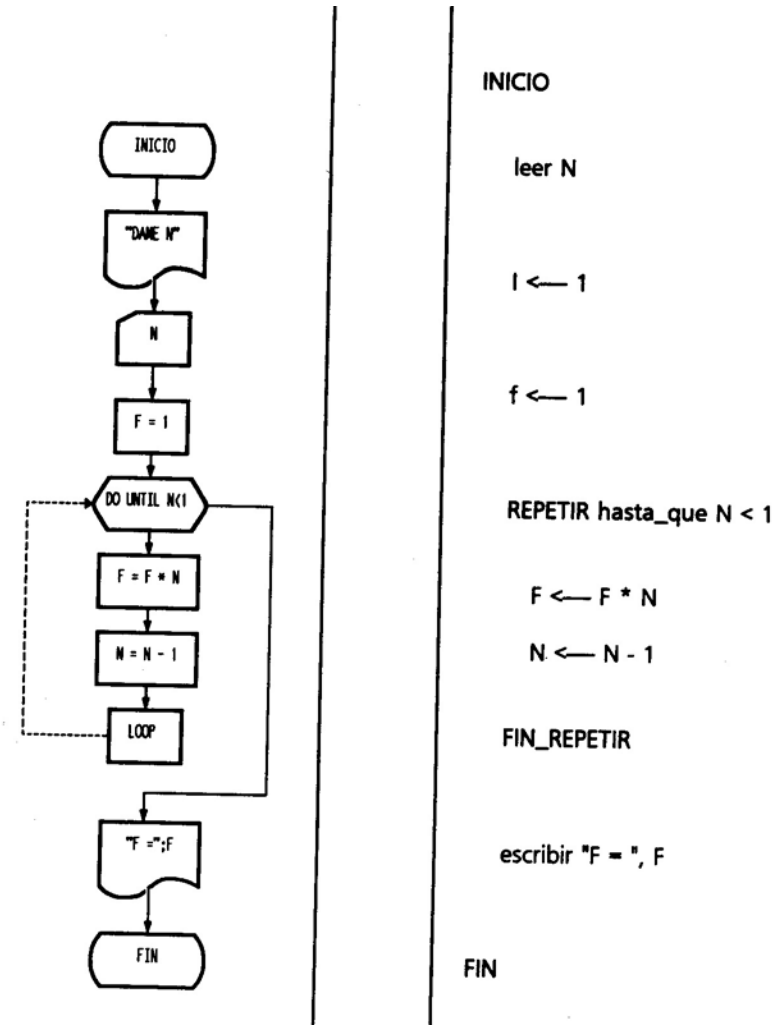
En forma reducida se indica lo anterior como

```
DO UNTIL condición
PROCESO
LOOP
```

El ciclo DO LOOP, con la opción UNTIL, puede interpretarse como: hacer ciclo HASTA QUE se cumpla la condición

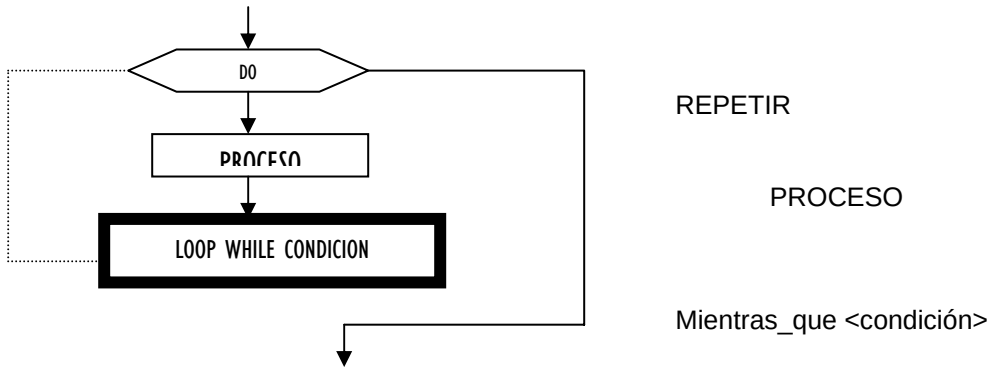
De forma similar a la anterior, la condición por lo general involucra dos variables que permiten romper el ciclo como se sugirió para el caso anterior: DO UNTIL i=n (hacer el ciclo hasta que i sea igual a n), así se controla cuantas veces se realiza el proceso.

Veamos el caso del ejemplo anterior resuelto con UNTIL.



Tercera opción (DO-LOOP WHILE)

La tercera forma simplificada para un ciclo iterativo es la opción WHILE al final del ciclo DO-LOOP y se muestra a continuación.



La fig indica que el proceso enmarcado en el rectángulo se ejecutará al menos una vez y se repetirá mientras se cumpla la condición indicada al final del ciclo iterativo y que ámbito del ciclo está delimitado precisamente por esa instrucción que involucra la condición (LOOP WHILE)

En forma reducida se indica lo anterior como:

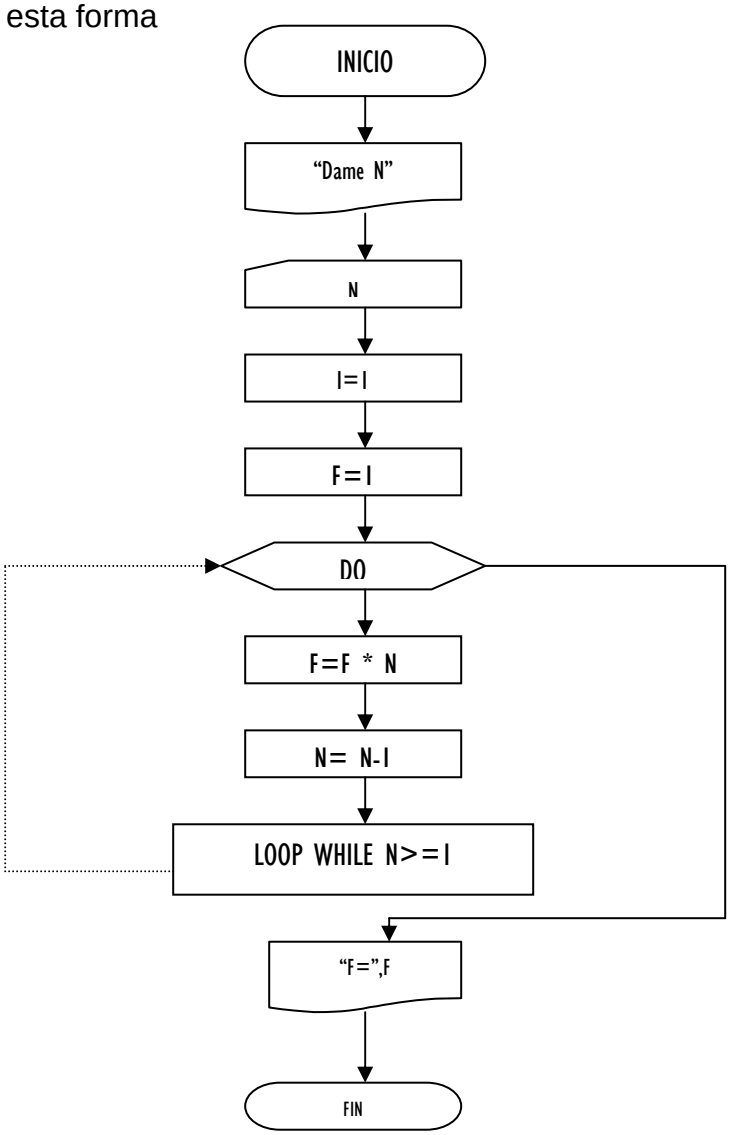
```
DO
PROCESO
LOOP WHILE CONDICION
```

El ciclo DO-LOOP WHILE se interpreta como hacer ciclo MIENTRAS que se cumpla la condición.
De forma similar a la anterior, la condición comúnmente involucra dos variables que permiten romper el ciclo, como se sugirió para los casos anteriores: por ejemplo una variable alfanumérica que se lee dentro del ciclo y que se compara con un valor alfanumérico dado (LOOP WHILE res$ =”n”).

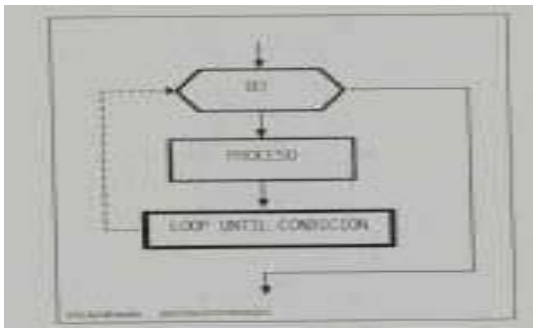
Para el caso del lenguaje C, la sintaxis elimina la palabra LOOP y la opción abreviada puede indicarse:

```
do
proceso
while (condición);
```

Ejemplo anterior resuelto de esta forma



Cuarta opción (DO-LOOP UNTIL)
La cuarta forma simplificada del diagrama para un ciclo iterativo corresponde a la opción UNTIL al final del ciclo DO_LOOP y se muestra a continuación:



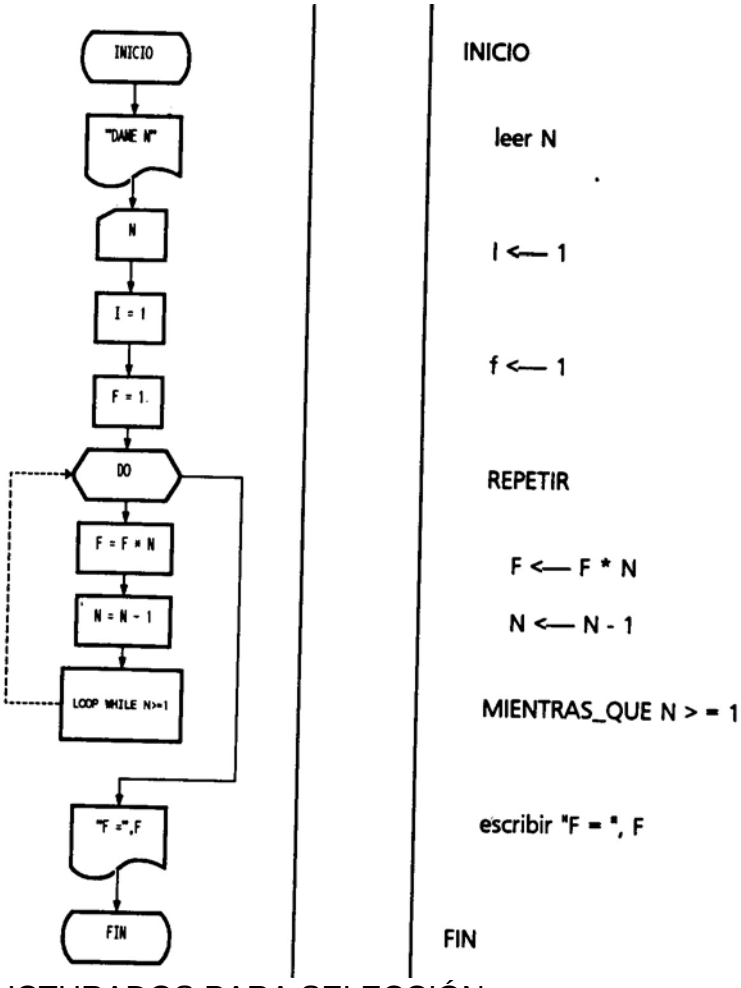
En la fig se indica que el proceso enmarcado en el rectángulo se ejecutará hasta que se cumpla la condición indicada al final del ciclo iterativo y que el ámbito del ciclo está delimitado precisamente por esa instrucción que involucra la condición (LOOP UNTIL)

```
do
proceso

loop until condicion
```

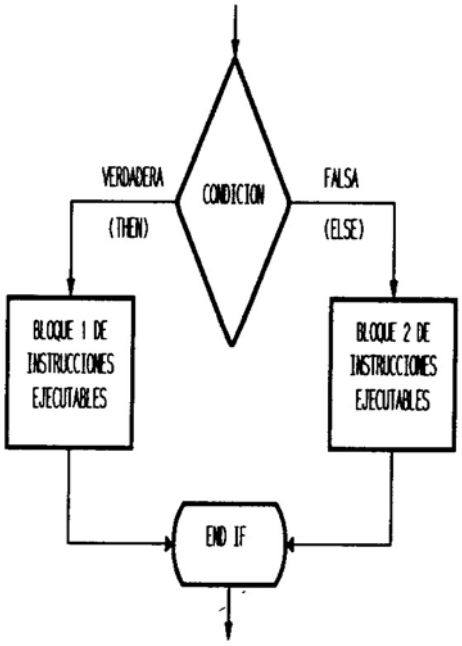
EL ciclo DO-LOOP UNTIL se interpreta como: hacer ciclo HASTA que se cumpla la condición.

La condición comúnmente involucra dos variables que permiten romper el ciclo, como se sugirió para los dos caso anteriores: por ejemplo una variable alfanumérica que se lee dentro del ciclo y que se compara con un valor alfanumérico dado (LOOP UNTIL resp$ = “n”)
Fig 212



DIAGRAMAS ESTRUCTURADOS PARA SELECCIÓN
DIAGRAMA ESTRUCTURADO IF –THEN ELSE

Cuando el resultado de una pregunta o IF considera por un lado ejecutar un conjunto de acciones específicas y en caso contrario ejecutar otro conjunto de instrucciones, se emplea un diagrama que corresponde al elemento IF--ELSE : La figura es:



SI <condición>

ENTONCES

bloque de instrucciones 1

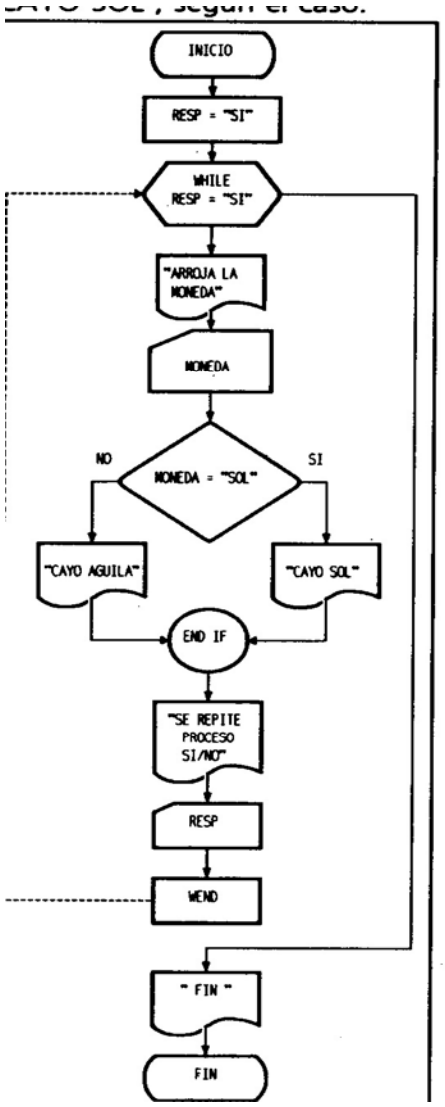
EN_CASO_CONTRARIO

bloque de instrucciones 2

FIN_SI

Ejemplo de la aplicación:

Un estudiante lanza al aire una moneda n veces, cada vez que cae águila se indica como un 1 y cuando el sol cae se indica como un 2. Elabore un diagrama de flujo que muestre como salida en la pantalla el letrero “Aguila” o el letrero Sol según sea el caso.




```
INICIO

  RESP <—— "SI"

  MIENTRAS RESP = "SI"

      imprimir "ARROJA LA MONEDA"

      leer MONEDA

      SI MONEDA = "SOL"
         ENTONCES

            imprimir "CAYO SOL"

         EN_CASO_CONTRARIO
            imprimir "CAYO AGUILA"
      FIN_SI
      imprimir "SE REPITE EL PROCESO SI/NO"

      leer RESP

  FIN_MIENTRAS

  imprimir "FIN"

FIN
```

III.1.6.2.2
ELEMENTO ESTRUCTURADO DE RAMIFICACIÓN ENCADENADO IF-THEN-ELSE IF-THEN-ELSE IF-THEN-ELSE

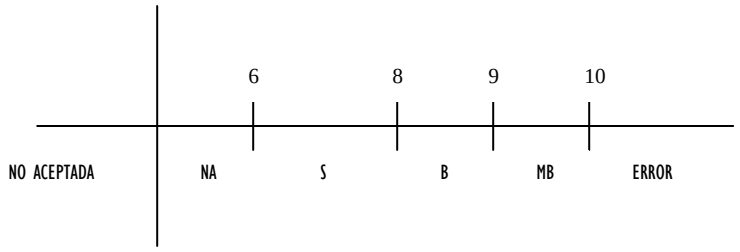
Cuando se presentan preguntas en serie sobre una expresión, se pueden establecer categorías, intervalos o clases que separen en distintas clases los valores por la posición que ocupan en un rango de comparación para una expresión dada. La primera pregunta captura los datos de la primera categoría, la segunda establece el segundo intervalo con valores mayores y acotados por la condición establecida y así sucesivamente.
Por esta razón se dice que se tiene una estructura de preguntas encadenadas o de selección de clases.

Veamos el siguiente ejemplo:

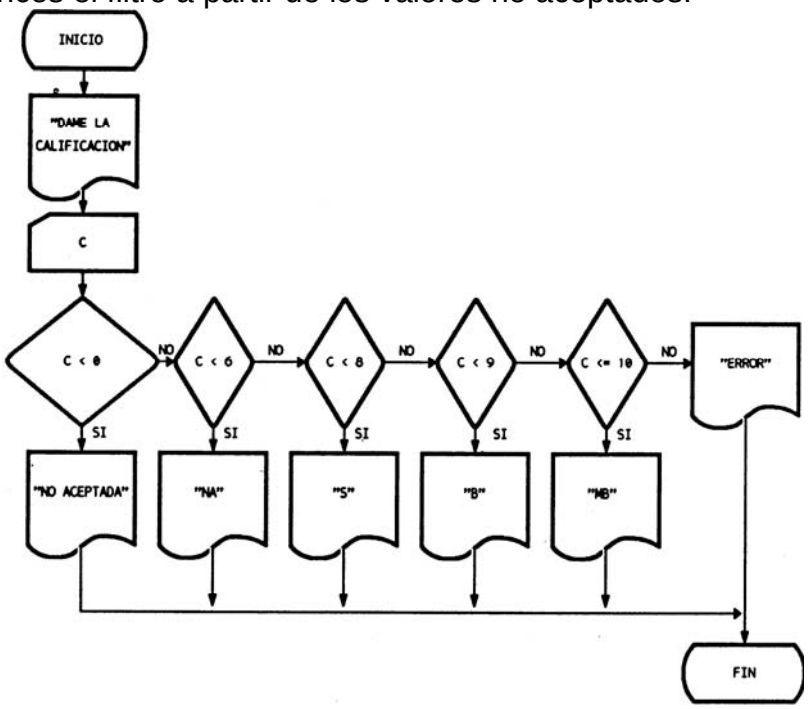
Considerar el problema de asignar la calificación del examen de un alumno, en escala de 0 a 10 puntos, con el siguiente criterio:

No son validas las calificaciones negativas

Na si la calificación es menor de 6
S si obtiene como mínimo 6 y menos de 8
B si obtiene como mínimo 8 y menos de 9
MB si obtiene de 9 a 10
Las calificaciones mayores de 10 se consideran como errores



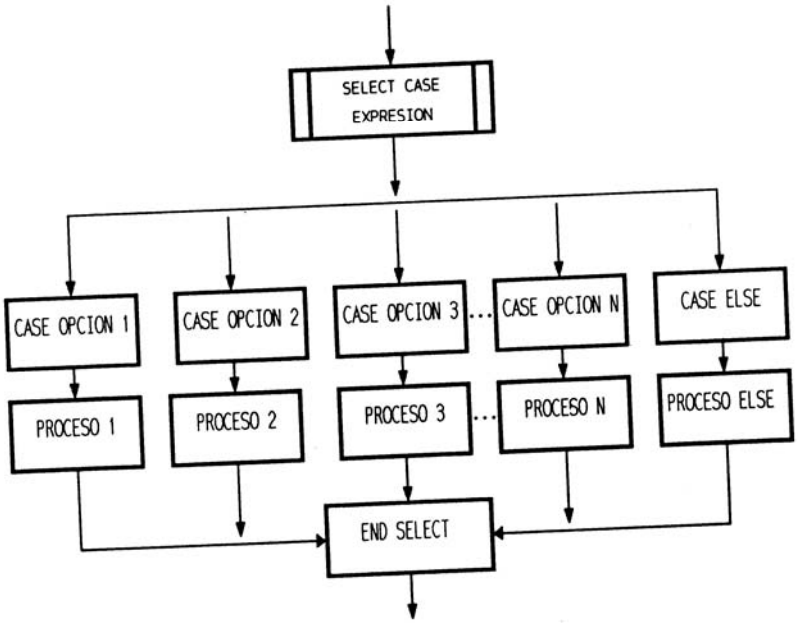
Si se desea filtrar las calificaciones realizando preguntas en cadena, se inicia por uno de los extremos del rango de calificaciones, es decir de menor a mayor o viceversa. Preguntar en cada cadena puede ejemplificarse diciendo que si la calificación no fue menor que 6, hay que ver si fue menor que 8 y si no fue ninguna de esas dos ver si fue menor que 9 y cuando no fue ninguna de las anteriores ,verificar si corresponde a lo; por ultimo en caso contrario ubicarlo como error.
Consideremos entonces el filtro a partir de los valores no aceptados:



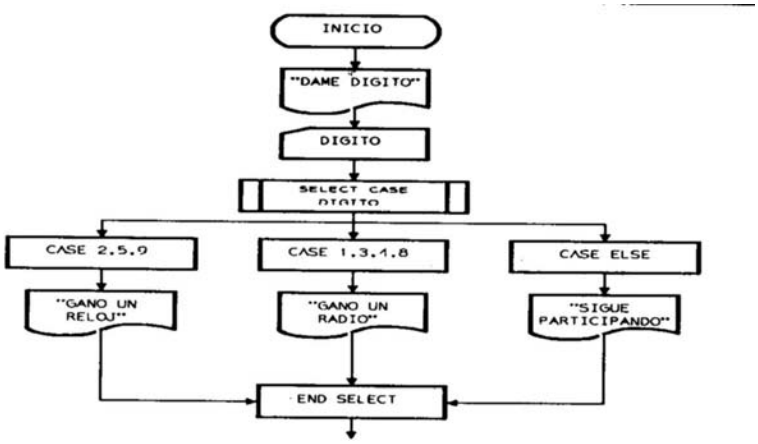
Observe cómo las preguntas en cadena sobre una misma expresión permiten filtrar la información por intervalos, y como las preguntas van de menor o mayor o viceversa, no hay necesidad de acotar ambos extremos y solo se delimita uno de ellos . De esta manera cada pregunta establece una magnitud limite para cada intervalo y establece un filtro para las preguntas encadenadas que restan (en este caso las preguntas previas capturan los valores menores y sólo dejan pasar los valores de los intervalos de las siguiente preguntas que acturan de la misma manera hasta el fin del encadenamiento.

II.1.6.2.3 SELECCIÓN DE CASOS ESPECIFICOS CASE

El diagrama que indica selección de un caso especifico de un conjunto de opciones dadas, se considera similar al caso de las preguntas encadenadas y corresponde a SELECT CASE y switch en lenguaje C. En el diagrama se pregunta por una expresión única que se compara con N+1 opciones posibles (empezando de izquierda a derech y se seleciona aquel caso especifico que le corresponde.



En lenguaje C la expresión sólo se puede comparar con un valor dado
Ejemplo
Ejemplo de aplicación: considere la asignación de premios en un sorteo donde se extrae un dígito.
-si se extrae alguno de los dígitos 2,5,9 se asigna como premio un reloj
-si se extrae algunos de los dígitos 1,3,4,8 se asigna como premio una radio portátil.

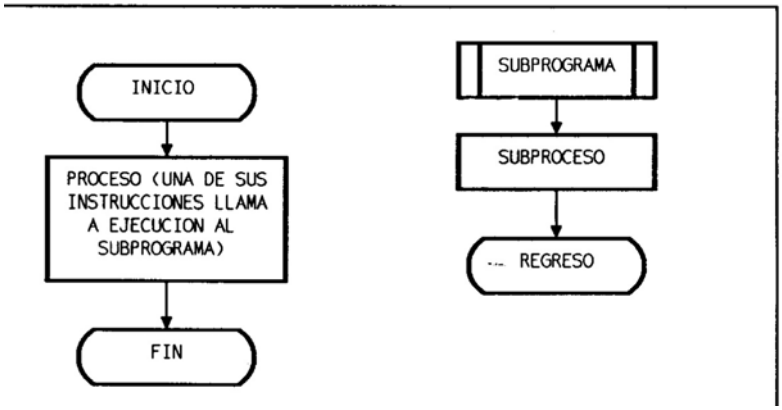


Ejemplo de aplicación: considere la asignación de calificaciones en un examen con el siguiente criterio:

-Las fracciones más pequeñas son décimas
-Si se obtiene un puntaje menor de 6 se asigna NA
-Si se obtiene un puntaje de 6.0 a 8 se asigna S
-Si se obtiene un puntaje menor de 9.3 se asigna B
-Si se obtiene un puntaje de 9.3 a 10 se asigna MB

III.1.6.3 SUBPROGRAMAS

Un subprograma es un módulo o conjunto de instrucciones agrupadas que por lo general se consideran separadas del programa o procedimiento principal. La ejecución de dichas instrucciones se realiza mediante un llamado del subprograma en algún lugar del programa principal, con lo cual el control de ejecución se transfiere al subprograma.
A continuación una vez realizado el proceso del subprograma, se devuelve el control de ejecución al programa principal. El grupo de instrucciones del programa principal termina con una llave (}) en lenguaje C, mientras que el conjunto de instrucciones del subprograma termina con un REGRESO (RETURN END SUB END FUCTION ETC.)



Por lo tanto los subprogramas facilitan la programación al dividir un problema extenso en pequeñas partes, además si el programa requiere repetir en varios lugares una serie de instrucciones, es conveniente definirlas dentro de un subprograma y llamarlo a ejecución donde se necesite.

Dependiendo del lenguaje de programación, hay distintos nombres para los subprogramas, pero consideraremos dos:

FUNCIONES
SUBRUTINAS

III..1.6.3.1
FUNCIONES

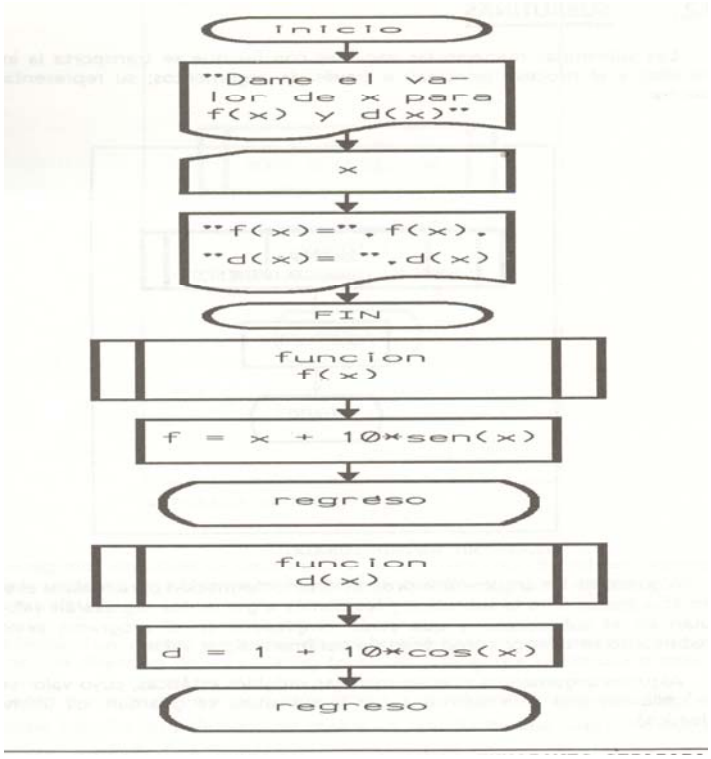
Una función es un subproceso que asocia un valor al nombre propio de la función, una vez que han sido dados valores a el o los argumentos de la función tal y como en el caso de la función trigonométrica sen(30°). Es decir, dado el valor del argumento 30° asociamos ala función el valor 0.5.

Una subrutina es un proceso que, a diferencia de la función que sólo calcula un valor único como resultado, puede manejar varios argumentos o variables como salida. Los subprogramas emplean básicamente los siguientes tipos de variables:

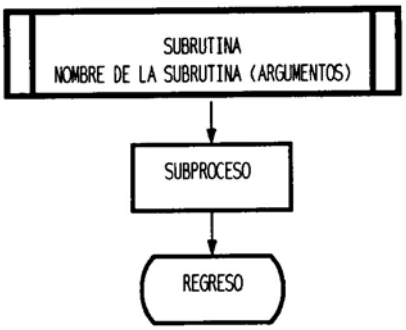
a)Comunes o globales, cuyos valores son válidos tanto en programa como subprogramas

b) Locales, cuando sus valores están delimitados al proceso o subproceso donde fueron declaradas.
c)Estáticas que se emplean en subprogramas y su valor permanece entre llamadas sucesivas.

En el caso del lenguaje C los programas se definen exclusivamente por medio de funciones. Se cuenta con una función principal que equivale al proceso principal; además es posible definir funciones adicionales que pueden considerarse como módulos separados o subprocesos de la función principal. Ejemplo considere el diagrama de flujo para el calculo de una función algebraica y su derivada.



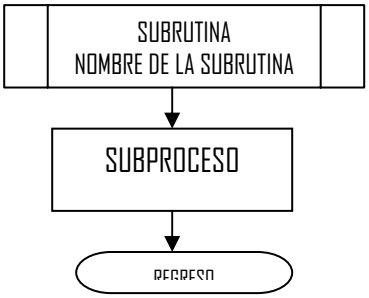
III.1.6.3.2 SUBRUTINAS
Las subrutinas manejan las variables con las que se transporta la información entre ellas y el proceso principal, a través de argumento su representación es la siguiente:



Algunos de los argumentos proporcionan información para realizar el subproceso al llamar a ejecución a la subrutina y los demás argumentos regresarán valores que se calculan en el subproceso y que serán empleados en el programa principal. Este procedimiento se conoce como **paso de parámetros de valor**.

Algunos argumentos pueden manejar variables estáticas, cuyo valor se conserva entre cada llamada ala subrutina (en la subrutina se guardan los ultimos valores empleados)

La subrutina puede manejar algunas variables en memoria común o de forma global, de tal manera que los valores que cambie el subproceso, automáticamente están cambiados para su uso en el programa principal y viceversa, sin necesidad de mencionarse como argumentos; esta opción se presenta a continuación:



III.2 SEUDOCODIGO

CARACTERÍSTICAS GENERALES DEL SEUDOCODIGO

Toda instrucción de lenguajes estructurados se expresa mediante órdenes escritas que corresponden a acciones indicadas en un lenguaje parecido a nuestro modo de hablar cotidiano (seudocódigo)

La idea del seudocódigo fue originalmente desarrollada en ingles, sin embargo, en nuestro caso expondremos su equivalente en español.

La representación de algoritmos mediante seudocódigo y diagrama de flujo se considera útil desde la etapa de depuración del proceso, debido a que en ellos se facilita la prueba de escritorio y su modificación para corresponder de manera eficiente ala solución planteada, mediante instrucciones simplificadas de control comunes a cualquier lenguaje de alto nivel, sin que importe la sintaxis o escritura especifica de alguno de ellos. En esta etapa primaria lo

importante es concentrarse en la lógica de la solución, expresando ideas tal y como las expresamos y decimos, y a continuación probándolas. Además una vez afinada la lógica de la solución, el planteamiento así expresado puede constituir parte de la documentación final del programa que se elabore.

Todo algoritmo indicado mediante pseudocódigo incluirá las instrucciones INICIO Y FIN, que delimitan el ámbito del conjunto de acciones del algoritmo; todo grupo de instrucciones secuenciales usará cierta sangría a la izquierda, la cual aumentará para las instrucciones dentro del os diagramas estructurados (CICLOS ITERATIVOS Y PREGUNTAS DE SELECCIÓN)