CONDICIONAL

En los lenguajes de programación es común el uso de condicionales que sirven para denotar diferentes alternativas que pueden llevarse a cabo dado el valor de una expresión lógica, el cual siempre será verdadero o falso.

La forma general que tiene un condicional (una pregunta) es la siguiente:

SI (expresión lógica es verdadera)
    Instrucciones que se realizan si la expresión lógica es verdadera
EN CASO CONTRARIO
    Instrucciones que se realizan si la expresión lógica es falsa

Ejemplo: Diseñar un algoritmo que lea un valor numérico y diga si dicho valor es mayor o no que 10.

Solución:

INICIO
    Leer A
    Si (A > 10)
        Imprimir “El valor ingresado es mayor que 10”
    De lo contrario
        Imprimir “El valor ingresado no es mayor que 10”
    Fin Si
FIN

Para realizar preguntas en cualquier lenguaje de programación es necesario el uso de operadores relacionales que permiten hacer comparaciones entre valores. En el lenguaje C estos operadores son:

Operador	Significado
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
= =	Igual a
!=	Diferente a

En muchos casos el condicional solo requiere de una alternativa por la parte verdadera, caso en el cual se omite toda lo de la parte falsa.

Ejemplo: Diseñar un algoritmo que lea un valor numérico y diga si dicho valor es igual a 25. también debe imprimir el cuadrado de ese valor.

Solución:

INICIO
    Leer X
    Si (X = = 25)
        Imprimir “El valor ingresado es 25”
    Fin Si
    Imprimir X*X
FIN

REPRESENTACIÓN

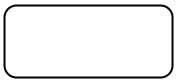
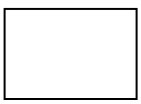
Hasta el momento, los algoritmos que hemos diseñado, han sido representados por medio de pseudocódigo el cual tiene las siguientes características:

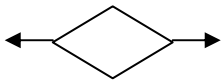
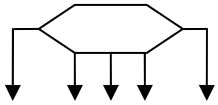
Es un lenguaje de especificación de algoritmos utilizando el lenguaje natural (ingles o español), que permite la traducción al lenguaje de programación con relativa facilidad.
No es ejecutable en ninguna maquina, su utilidad apunta a la planificación del programa.
Tiene la ventaja de que el programador debe preocuparse más por la lógica que por la sintaxis propia del lenguaje.
Permite identificar con facilidad errores de lógica antes de la codificación del algoritmo.

Sin embargo, existen otras formas de representación bastante comunes que se pueden encontrar en diversos textos de consulta.

Una de estas formas de representación es el diagrama de flujo el cual se caracteriza por utilizar un conjunto de símbolos gráficos normalizados y expresar de forma clara los posibles caminos de ejecución de las acciones (secuencia, condicional e iteración). La representación de un algoritmo mediante un diagrama de flujo se denomina ordinograma. En vez de caminos de ejecución se habla, con más propiedad, del flujo de control u orden lógico en el que se realizan las acciones de un algoritmo.

Los símbolos típicos de un diagrama de flujo son:

Símbolo	Significado
	**Punto de Inicio – Fin.** Representa el comienzo y el final de un algoritmo. Puede representar también una parada o interrupción programada que sea necesaria.
	**Entrada – Salida de datos.** Cualquier introducción de datos en la memoria desde el teclado o archivo de entrada, o presentación de resultados en la pantalla o archivo de salida.
	**Proceso.** Cualquier tipo de operación que pueda originar cambio de valor de una variable, operaciones aritméticas, de transferencia, etc.

	**Condicional.** Indica operaciones lógicas entre valores, y en función del resultado determina cuál de los distintos caminos alternativos se debe seguir.
	**Condicional múltiple.** En función del resultado de la comparación se seguirá uno de los diferentes caminos.
	Conector. Sirve para enlazar dos partes cualesquiera de un ordinograma

Otra forma de representación es el diagrama de caja, también conocido como de Chapin, el cual es como un diagrama de flujo en el que se omiten las flechas de unión, es decir, que cada acción se representa mediante cajas contiguas. Las acciones sucesivas se escriben en cajas sucesivas y, como en los diagramas de flujo, se pueden escribir diferentes acciones en una misma caja.

En el caso particular de la instrucción que estamos viendo en esta clase (el condicional) la representación en ambos diagramas es:

Diagrama de flujo:

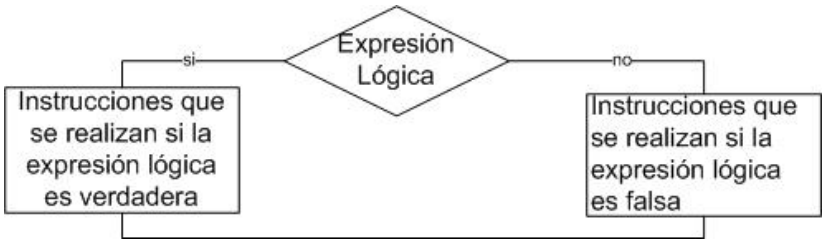
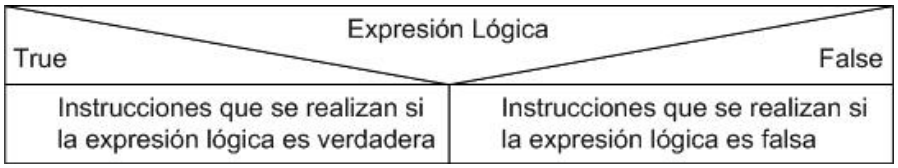


Diagrama de caja:



Ejemplo: representar el último ejemplo realizado tanto en diagrama de caja como en diagrama de flujo.

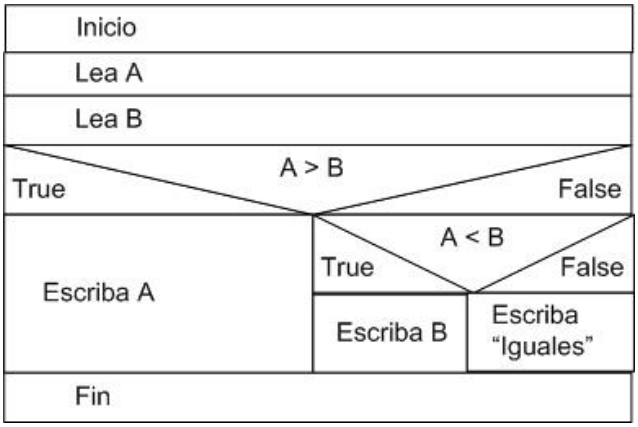
Importante: El estudiante puede usar cualquier forma de representación de algoritmos (pseudocódigo, diagrama de flujo, diagrama de caja), lo fundamental es que la lógica del algoritmo como tal esté bien.

VARIACIONES

Siempre es posible utilizar un condicional dentro de otro condicional. En programación, a este concepto se le conoce por el nombre de “anidar”.

Ejemplo: Diseñar un algoritmo para que lea dos valores numéricos y escriba el mayor de los dos o si son iguales.

Solución en diagrama de caja:



Solución en pseudocódigo:

```
INICIO
        Leer A
        Leer B
        Si   A > B
                Escribir A
        En caso contrario
                Si A < B
                        Escribir B
                En caso contrario
                        Escribir “son iguales”
                Fin Si
        Fin Si
FIN
```

Implementación en C:

```
#include <iostream.h>

void main()
{
float A, B;

cout<<”Ingrese A: “;
cin>>A;
```

```
cout<<"\nIngrese B: ";
cin>>B;

if (A > B)
{
        cout<<"\n"<<A;
}
else
{
        if (A > B)
        {
                cout<<"\n"<<B;
        }
        else
        {
                cout<<"\nSon iguales";
        }
}

}
```

Las preguntas que se utilizan en los condicionales pueden ser simples (sólo es necesario evaluar una pregunta) o compuestas (varias preguntas), para las cuales se deben utilizar operadores lógicos.

Operador	Accion	En C
Y	*P* Y *Q* es verdadera sólo si tanto *P* como *Q* son verdaderas, en caso contrario es falsa	&&
O	*P* O *Q* es verdadera si cualquiera de las dos es verdadera, y sólo es falsa si ambas son falsas	||
No	No *P* cambia el valor de *P*	!

Ejemplo: Si *A* es 8, B es 14 y C es -2 determine los valores de verdad de las siguientes expresiones lógicas.

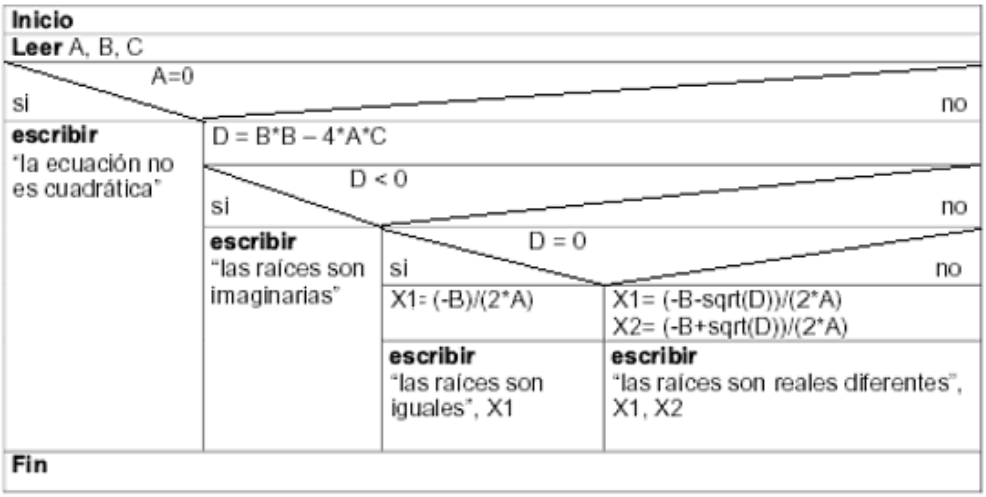
1. (*A* >= 10) || (*B* < 20)
2. !(B > 0) && (C = = -2)
3. (B != 4) || ((A >= 8) && (C < 0))

Ejemplo: Diseñar un algoritmo para resolver la ecuación cuadrática $AX^2 + BX + C = 0$

Recordar la solución general de esta ecuación es $X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$

Importante: Antes de “saltar” a la solución de este problema hay que recordar que a la hora de diseñar un algoritmo siempre hay que considerar todos los casos posibles que pueden presentarse y no debe caerse en el error de pesar que “eso se sabe” (LO SABE UNO PERO NO EL COMPUTADOR)

Solución en diagrama de caja:



El estudiante debe traducir este algoritmo a pseudocódigo y posteriormente a lenguaje C.