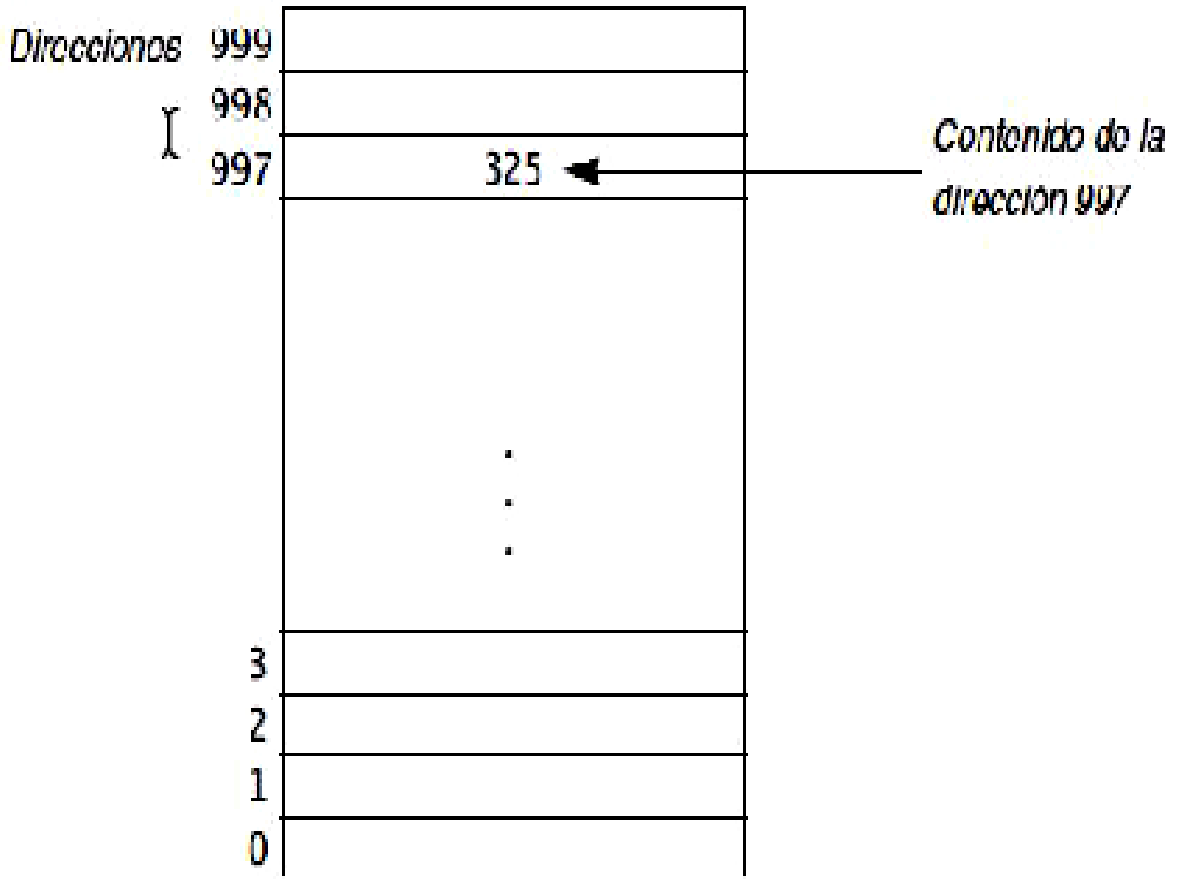
Elementos de un Programa

Diseño de Algoritmos

ELEMENTOS DE UN PROGRAMA



Variables

Direcciones de memoria con un valor: un número, una letra, o valor nulo.

Diseño de Algoritmos

Variables

Son elementos de almacenamiento de datos

Es un nombre que se asocia con una porción de la memoria del ordenador, en la que se guarda el valor asignado a dicha variable.

Hay varios tipos de variables que requieren distintas cantidades de memoria para guardar datos.

Una variable es un grupo de bytes asociado a un nombre o identificador, y a través de dicho nombre se puede usar o modificar el contenido de los bytes asociados a esa variable.

En una variable se puede almacenar distintos tipos de datos.
De acuerdo al tipo de dato, definido para cada lenguaje de programación, será la cantidad de bytes que ocupa dicha variable en la memoria.

Diseño de Algoritmos

Variables

Delante del nombre de cada variable se ha de especificar el tipo de variable

Para algunos lenguajes de programación es posible especificar si las variables numéricas tendrán o no signo (es decir que puedan tomar valores negativos y positivos, o sólo positivos).
La elección de tener o no signo (definidas con la palabra reservada unsigned) depende del significado de la variable ya que al no tenerlo podremos almacenar el doble de valores, por ejemplo, una variable short sin signo posee valores desde el 0 al 65535.

Diseño de Algoritmos

Variables

Las variables son uno de los elementos básicos de un programa, y se deben

Declarar:

Para poder utilizar una variable en un programa, primero tenemos que declararla.
Declarar una variable significa asignarle un nombre y un tipo.

Por ejemplo:

```
int a;   // declaramos la variable a de tipo int
```

Inicializar:

A las variables declaradas dentro de un método no se les asigna un valor automáticamente. Es nuestra responsabilidad asignarles valores iniciales.

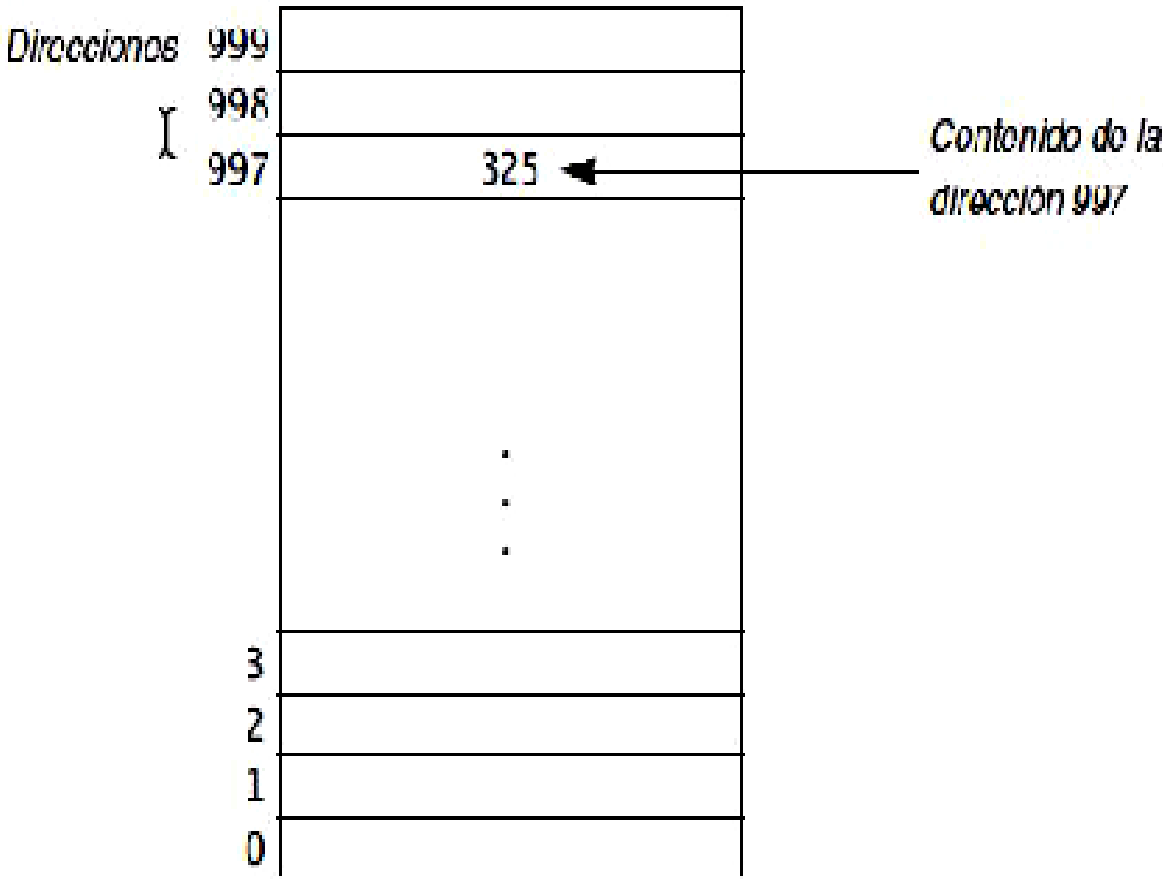
```
int contador = 0; // declara la variable contador de tipo int y se le asigna el valor 0
```

Usar:

Si declaramos una variable debemos utilizarla. (ocupan memoria)

Diseño de Algoritmos

ELEMENTOS DE UN PROGRAMA



Constantes

Direcciones de memoria con un valor que ***no varía*** durante la ejecución del programa.

Diseño de Algoritmos

Constantes

Los programas de ordenador contienen ciertos valores que no deben cambiar durante su ejecución. **Estos valores se llaman constantes.**

Podemos decir que una constante es una posición de memoria que se referencia con un identificador, conocido como nombre de la constante, donde se almacena el valor de un dato que no puede cambiar durante la ejecución del programa.

Una constante en Java se declara de forma similar a una variable, anteponiendo la palabra **final**

Constantes

La estructura sería:

final nombreConstante = valor;

De esta forma si queremos definir las constantes DIAS_SEMANA ó DIAS_LABORABLES, que sabemos que son variables que no cambiarán su valor a lo largo del programa, generaremos el siguiente código:

```
final int DIAS_SEMANA = 7;
final int DIAS_LABORABLES = 5;
```

Si queremos utilizar una constante Java, simplemente deberemos de utilizar su nombre. Así, si queremos utilizar las anteriores constantes, lo haremos de la siguiente forma:

```
System.out.println("El número de días de la semana son " + DIAS_SEMANA);
System.out.println("El número de días laborables de la semana son " + DIAS_LABORABLES);
```

TIPOS DE DATOS

Byte

- 13 grados

Short

- 12 años

Entero

- 111000 programadores

Long

- 30546789165 - Nº de Cuit

Flotante

- 590.90 $

Decimal

- 54378981,9181

Booleano

- True / False

TIPOS DE DATOS

Tipo de Datos	Significado	Ejemplos de uso
Byte	Número entero de 8 bits. Con signo	Temperatura de una habitación en grados Celsius
Short	Número entero de 16 bits. Con signo	Edad de una Persona
Int	Número entero de 32 bits. Con signo	Distancia entre localidades medida en metros
Long	Número entero de 64 bits. Con signo	Producto entre dos distancias almacenadas en variables tipo int como la anterior
Float	Número Real de 32 bits.	Altura de algún objeto
Double	Número Real de 64 bits.	Proporción entre dos magnitudes
Boolean	Valor lógico: true (verdadero) o false (falso)	Almacenar si el usuario ha aprobado un examen o no

OPERADORES ARITMÉTICOS

Suma +	Resta -	Producto *
entero a = 5+6 = 11;	entero b = 15-4=11;	entero c = 3*3=9;

División /	Resto % Resto de la división entera
flotante d = 40/10 = 4.0;	entero e= 50 % 8=2;

OPERADORES UNITARIOS

Los operadores unitarios requieren sólo un operando; que llevan a cabo diversas operaciones, tales como incrementar/decrementar un valor de a uno, negar una expresión, o invertir el valor de un booleano.

Operador	Descripción	Ejemplo	Resultado
++	operador de incremento; incrementa un valor de a 1	int suma=20; suma++;	suma=21
--	operador de decremento; Reduce un valor de a 1	int resta=20; resta--;	resta=19
!	operador de complemento lógico; invierte el valor de un valor booleano	boolean a=true; boolean b= !a;	b=false

OPERADORES UNITARIOS

Los operadores unitarios requieren sólo un operando; que llevan a cabo diversas operaciones, tales como incrementar/decrementar un valor de a uno, negar una expresión, o invertir el valor de un booleano.

++	- -	!

entero a = 5+6 = 11;
a++;
a=12
	entero b = 15-4=11;
	entero --;
	b=10
		booleano b=true;
		a = !b;
		a= false

OPERADORES CONDICIONALES

Relacionales - Lógicos

==	!=	> "o" >=	< "o" <=
Igual a	Distinto de	Mayor o Mayor e Igual	Menor o Menor e Igual
Ejemplos			
15==14 -> false 3==3-> true	15!=14-> true 3!=3-> false	8 > 3-> true 3 >= 3 -> true 3 > 8 -> false	4 < 8 -> true 8 <= 8 -> true 8 <= 4 -> false

OPERADORES CONDICIONALES

Relacionales - **Lógicos**

&&	||
AND	**OR**
Ambas son verdaderas	Al menos una es verdadera

Ejemplos	
SI (temperatura > 26 && haySol==true) **MOSTRAR:** **“El clima está soleado y caluroso”**	**SI (pagoEfectivo == true || pagoContado ==true)** **MOSTRAR:** **“Su compra tiene un descuento del 10% por la forma de pago realizada.”**

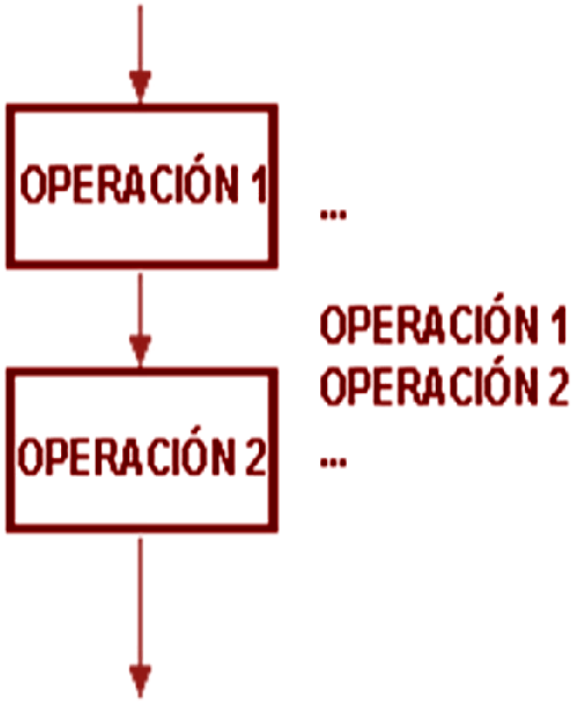
ESTRUCTURAS DE CONTROL

- Secuenciales
- Selectivas o De Decisión
- Repetitivas

ESTRUCTURAS DE CONTROL

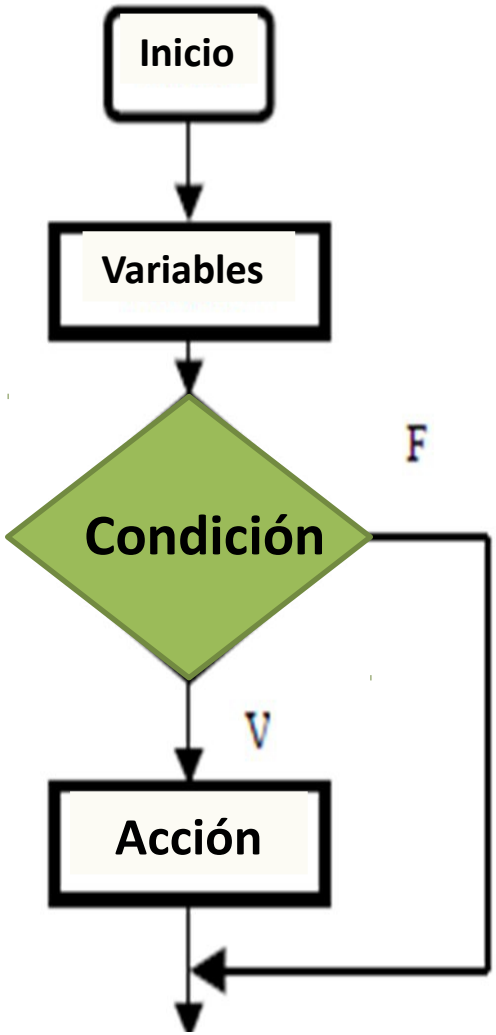
Secuenciales

```
INICIO calcularDescuento
    flotante precioProducto = 450.80;
    flotante porcDescuento = 0.10;

    flotante descuento =
    precioProducto*porcDescuento;

    flotante nuevoPrecio =
    precioProducto- descuento
FIN
```



ESTRUCTURAS DE CONTROL

Selectivas o De Decisión ➡ *Alternativa Simple*



¿Qué es una instrucción de control alternativa simple?

Una instrucción alternativa simple es una variante (más sencilla) de una instrucción alternativa doble.
En pseudocódigo, para representarla se utiliza la sintaxis:

```
si ( <expresión_lógica> )
   <bloque_de_instrucciones>
fin_si
```

Ejemplo: Se quiere diseñar el algoritmo de un programa que:

1º) Pida por teclado la nota (dato real) de una asignatura.
2º) Muestre por pantalla:

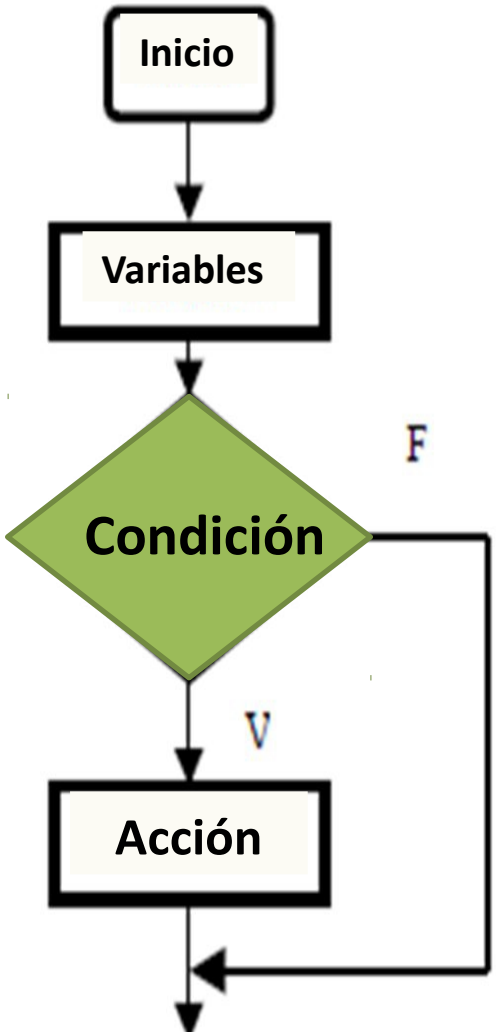
"APROBADO", en el caso de que la nota sea mayor o igual que 5.

ESTRUCTURAS DE CONTROL

Selectivas o De Decisión

➡ *Alternativa Simple*

```
algoritmo Calificacion_segun_nota

variables
   int nota

inicio
   escribir( "Introduzca nota (real): " )
   leer( nota )

   si ( nota >= 5 )
      escribir( "APROBADO" )
   fin_si
fin
```

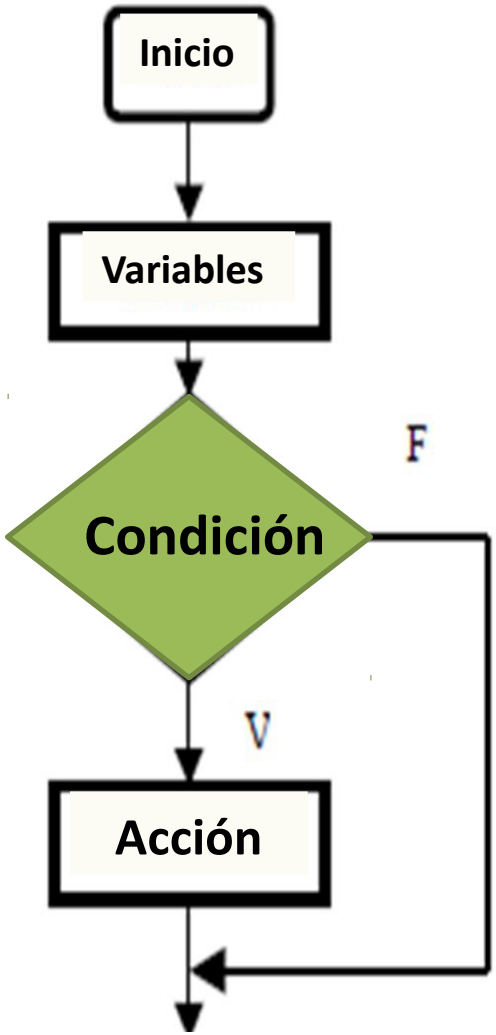


ESTRUCTURAS DE CONTROL

Selectivas o De Decisión

➡ *Alternativa Simple*

```
INICIO aplicarDescuento
  flotante precioProd = 450.80;
  flotante porcDesc = 0.10;
  Booleano tieneDescuento=true;
   SI (tieneDescuento==true)
       ENTONCES flotante descuento =
   precioProd*porcDesc
       flotante nuevoPrecio = precioProd –
   descuento
   FIN SI
FIN
```

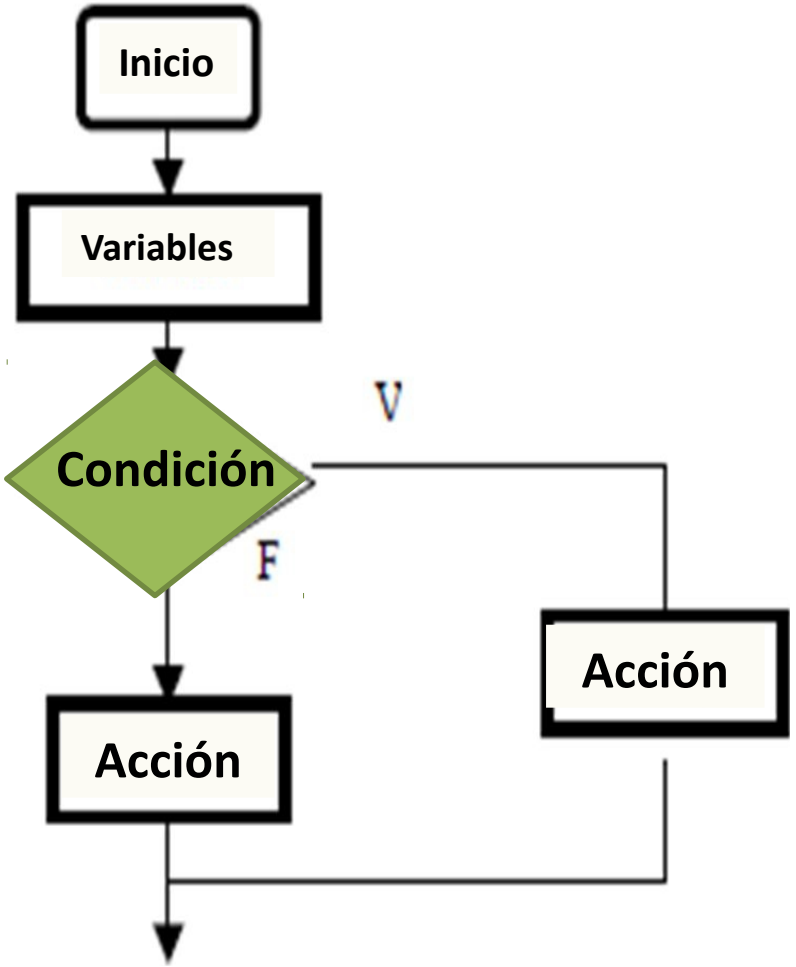


ESTRUCTURAS DE CONTROL

Selectivas o De Decisión

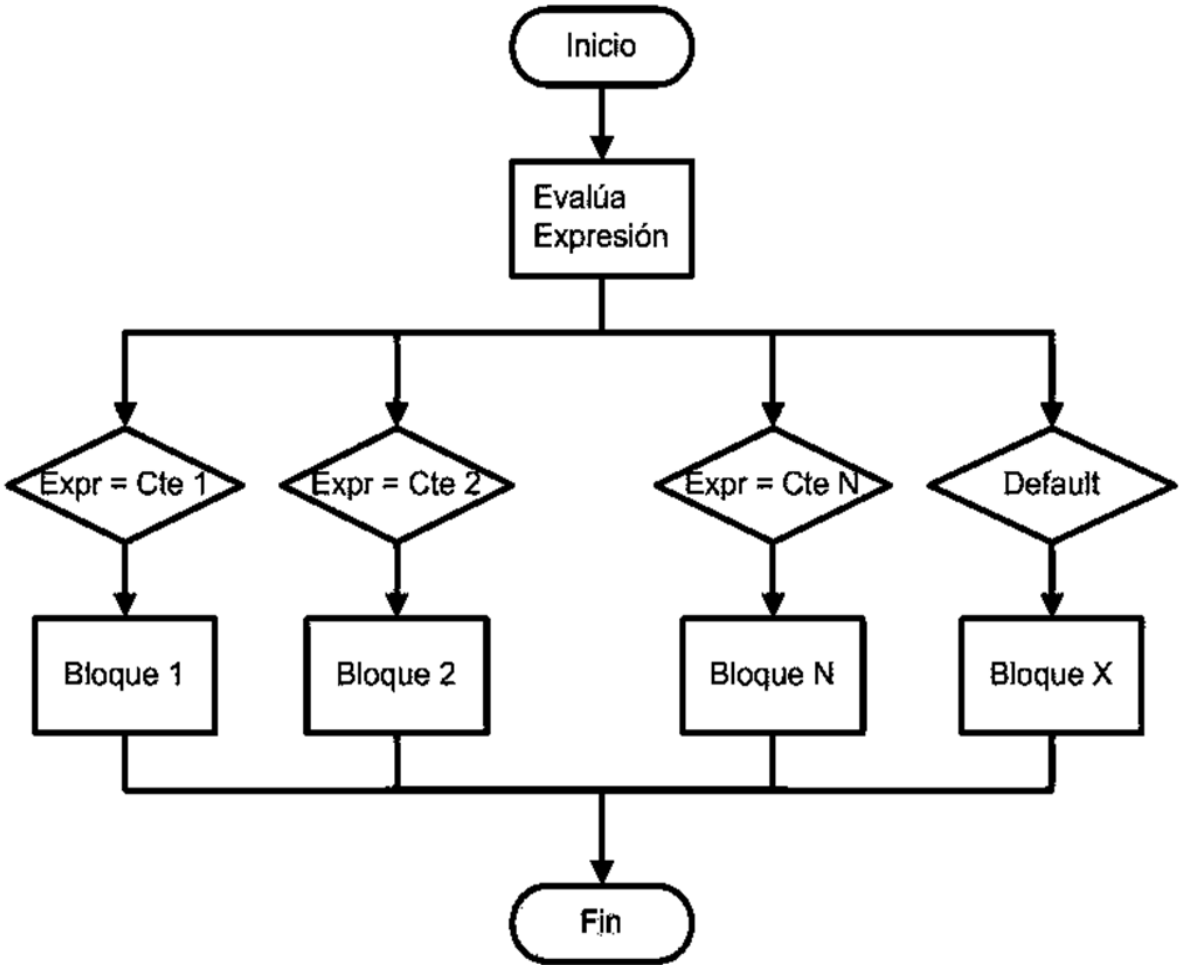
Alternativa Doble

```
INICIO aplicarDescuento
   flotante precioProd = 450.80;
   flotante porcDesc = 0.10;
   Booleano tieneDescuento=true;
    SI (tieneDescuento == true)
    ENTONCES
        flotante desc = precioProd*porcDesc;
        flotante nuevoPrecio = precioProd - des
        MOSTRAR “El precio nuevo es:” +
    nuevoPrecio
    SINO
        MOSTAR “No aplica descuento”
    FIN SI
FIN
```



Selectivas o De Decisión ➡ *Alternativa Múltiple*

```
INICIO
ENTERO posicionDeLlegada = 3
    SEGUN SEA posicionDeLlegada
    1: entregar medalla de oro
    2: entregar medalla de plata
    3: entregar medalla de
    bronce
    otro: entregar mención de
                participación
FIN
```



Preguntas?

Ejercicios

Alternativa Simple

Escribir un algoritmo que permita loguearse (registrarse) a un sistema, ingresando un nombre de usuario y la contraseña adecuada. Considerar que tanto el usuario como la contraseña están formados sólo por letras. El sistema deberá validar que el usuario y la contraseña sean correctas, comparándolas con lo que es sistema tiene registrado para ese usuario.

***Aclaración, en los sistemas reales, el inicio de sesión es mucho más complejo que lo que se muestra a continuación. Se ha simplificado el proceso, abstrayendo la validación a una función denominada esValido() que resuelve la verificación del usuario y su contraseña.*

RUTINAS

- ✓ **BLOQUE:** Conjunto de Sentencias.
- ✓ Puede tener: **valor de retorno** y **parámetros**.
- ✓ **Funciones**: Rutina con parámetros.

Ejemplo

```
Flotante SumarPrecioProductos(precioProducto1,
precioProducto2)
    flotante suma = precioProducto1 + precioProducto2;
    return suma
```