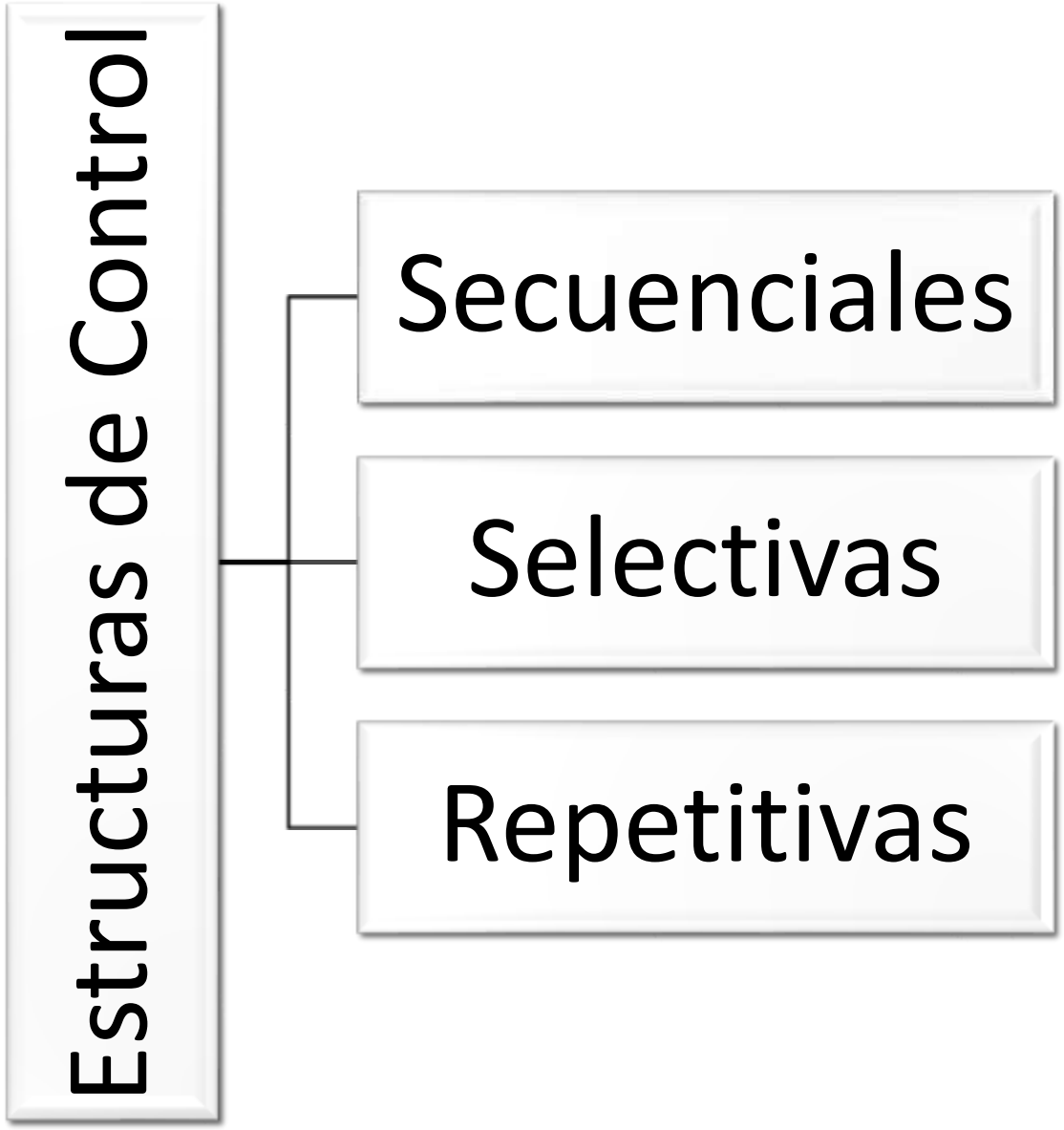
Estructuras de Control

FUNDAMENTOS DE PROGRAMACIÓN

Introducción

En un programa los enunciados son ejecutados uno después del otro, en el orden en que aparecen escritos. Sin embargo, habrá momentos en que el programa debe ejecutar determinadas partes dependiendo del estado en el que se encuentre. Esto permite modificar el orden de la ejecución para adaptarse al estado del programa.

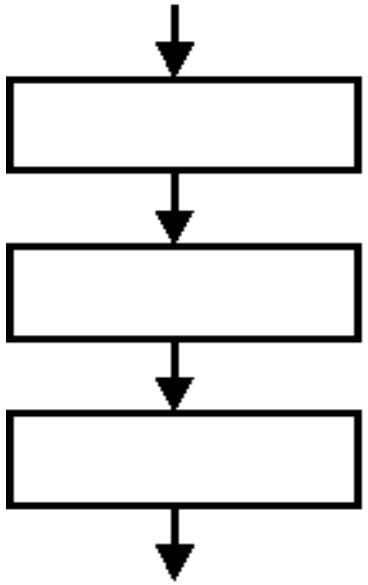
Las *sentencias de control* son la esencia de cualquier lenguaje de programación, ya que gobiernan el flujo de la ejecución del programa.


Estructuras de Control
Secuenciales
Selectivas
Repetitivas

Estructuras secuenciales

Las instrucciones se ejecutan en el mismo orden en que ellas aparecen en el programa.

Una acción sigue a otra en secuencia.

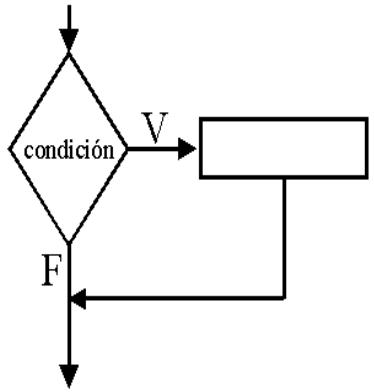


Estructuras selectivas

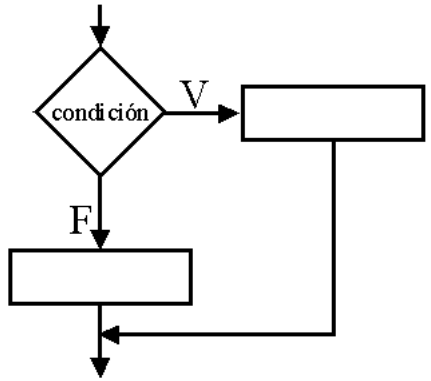
Se utilizan para tomar decisiones lógicas.

Se evalúa una condición y en función del resultado de la misma se realiza una opción u otra.

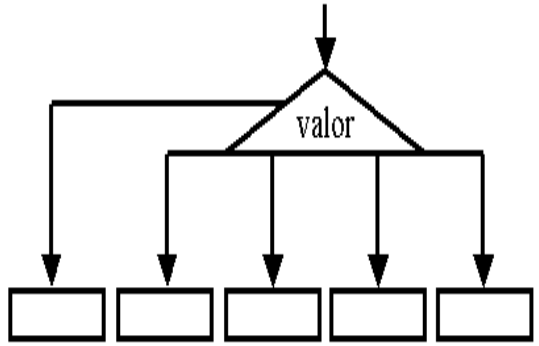
Las condiciones se especifican utilizando *expresiones lógicas* y *relacionales.* La condición puede resultar verdadera o falsa, indicando qué decisión o acción se debe tomar.



a) Simple



b) Bicondicional



c) Condición múltiple

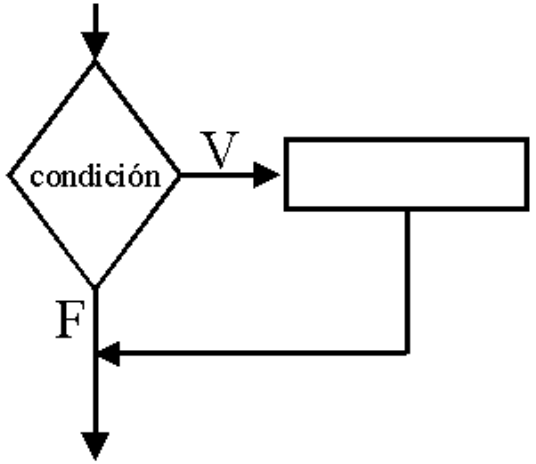
Estructura selectiva if (condición)

La sentencia se ejecuta solo si el resultado de evaluar la *condición* es verdadero, caso contrario no se toma ninguna acción.

NOTA: Si el número de sentencias a desarrollar en el cuerpo de if es mayor a 1 se utilizan { }.

Sintaxis en C++ de la sentencia if:

```
if (condición) sentencia;
```

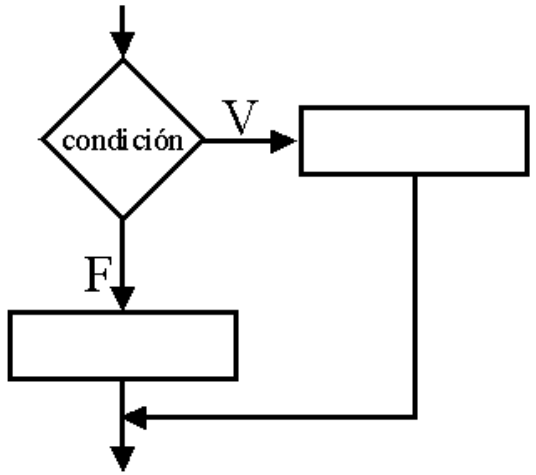


Estructura selectiva if (condición) else

Permite que se ejecuten acciones distintas cuando la *condición* es verdadera que cuando la *condición* es falsa.

Sintaxis en C++ de la sentencia if-else:

```
if (condición) sentencia;
else
   sentencia;
```



Estructura selectiva if/else anidadas

Prueban para muchos casos, colocando estructuras *if/else* dentro de estructuras *if/else.*

Sintaxis en C++:

```
if (condición)
    sentencia;
else
    if (condición)
        sentencia;
    else
        if (condición)
            sentencia;
        else
            sentencia;
```

Estructura selectiva switch-case

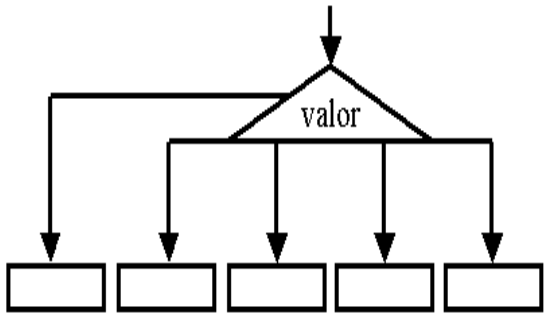
La sentencia múltiple *switch ( )* está formada por una serie de etiquetas *case* y un caso opcional *default*.

Sirve para agrupar varias sentencias if en una sola, en el caso particular en el que una variable es comparada a diferentes valores, todos ellos constantes y que realiza acciones si coincide con ellos.

La expresión solo puede ser de tipo entero y de un solo caracter, al igual las constantes que se colocan.

Sintaxis en C++:

```
switch (expresion)
{
   case constante1:
      sentencia1;
      break;
   case constante2:
      sentencia2;
      break;
   case constante_n:
      sentencia_n;
      break;
  default:
      sentencias;
      break;
 }
```

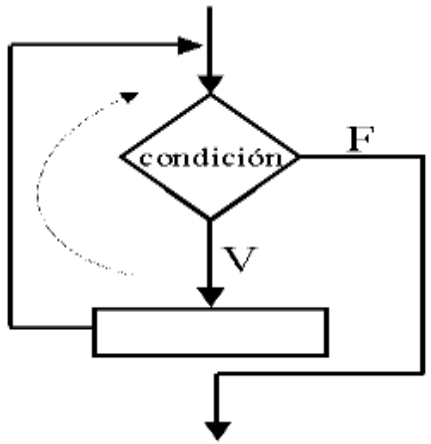


Estructuras repetitivas

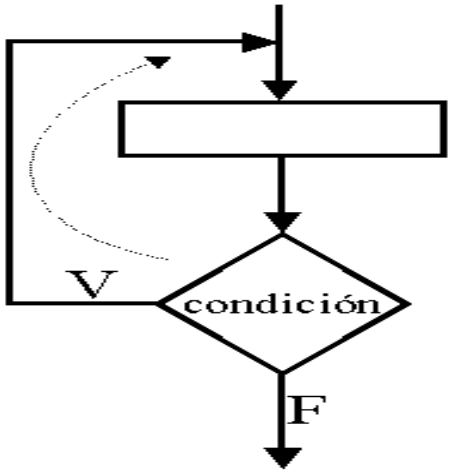
Repiten una secuencia de instrucciones un número determinado de veces, en tanto cierta condición se mantenga verdadera.

Se conocen como *bucles / iteración.*

a) for **b) while**



c) do- while

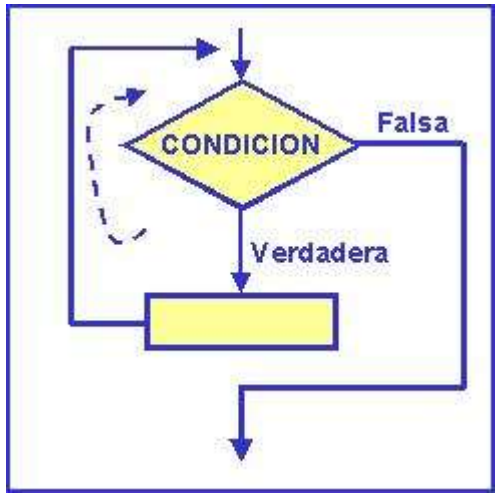


Estructuras repetitivas for y while

La estructura de repetición **for** y **while,** repiten una secuencia de instrucciones un número determinado de veces, en tanto cierta condición sea verdadera. El bucle itera mientras la condición sea verdadera. Cuando llega a ser falsa, el control del programa pasa a la línea que sigue al bucle.

Sintaxis en C++ del ciclo for:

```
for (inicialización; condición; incremento/decremento)
    sentencia;
```



Estructuras repetitivas for y while

En C/C++, se utiliza el bucle **while, con la siguiente sintaxis:**

```
while ( condición )
sentencia;
```

Ejemplo, utilizando el ciclo for:

```
    int sum=0, numero;
    for (numero=2; numero<=100; numero+=2)
        sum=sum+numero;
    cout<<“La suma es ”<<suma<<endl;
    return 0;
}
```

Ejemplo, utilizando el bucle while:

```
    int product;
product = 2;
while ( product <= 1000 )
        product = 2 * product;
```

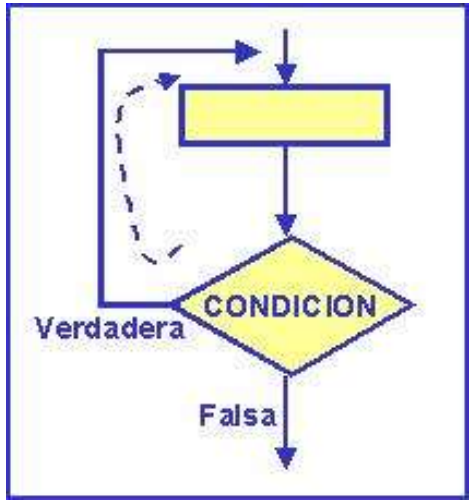
Estructura repetitiva do-while

Esl funcionamiento de esta estructura es similar al del bucle while, salvo que la expresión de control se evalúa al final del bucle, por lo tanto, el cuerpo del ciclo se ejecutará por lo menos una vez.

Cuando termina, la ejecución continuará con el enunciado que aparezca después de la cláusula **while.**

Sintaxis en C++ para el bucle do-while

```
do {
   sentencia;
 }while (condición);
```



Por ejemplo:

```
int num;
do{
    cin >> num;
}while ( num > 100 );
```