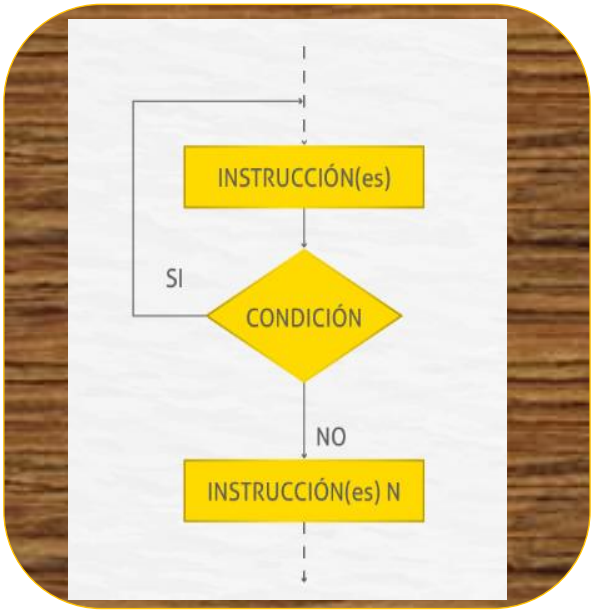
ESTRUCTURAS REPETITIVA DO … WHILE

La estructura repetitiva do-while es aquella en que el cuerpo del bucle se repite mientras que se cumple una determinada condición. En esta estructura, la condición del ciclo se evalúa al final, por lo que siempre se ejecutarán las instrucciones del ciclo por lo menos una vez. Si la condición se evalúa verdadera, se ejecuta nuevamente el cuerpo del bucle; si la condición es falsa, entonces sale y se sigue con el flujo normal del programa. Este proceso se repite una y otra vez hasta que la condición sea falsa.



Donde:

CONDICIÓN es la expresión lógica a evaluar.

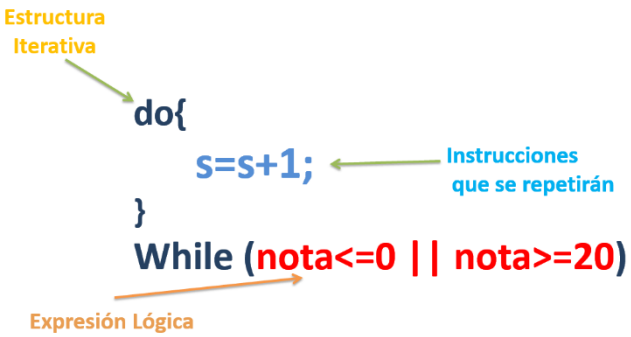
INSTRUCCIÓN (es) expresa las acciones que se van a realizar dentro del ciclo.

INSTRUCCIÓN (es) N exspresa las acciones que se van a realizar después de terminar la ejecución del ciclo.

- El bloque de operaciones se repite MIENTRAS la condición sea Verdadera.
- Si la condición retorna Falso el ciclo se detiene.
- En C++, todos los ciclos repiten por verdadero y cortan por falso.
- ***Es importante analizar y ver que las operaciones se ejecutan como mínimo una vez. En el caso del do … while***

Sintaxis del Ciclo while en C++:

```
do{
   <BLOQUE DE INSTRUCCIONES>
}while(expresión_lógica);
```



El bloque de instrucciones se realizará mientras la condición se cumpla. Es una estructura post-condición, pues, la expresión lógica se comprobará después de haber realizado por primera vez el bloque de instrucciones. En pocas palabras, siempre se realizará el bloque de instrucciones por lo menos una vez.

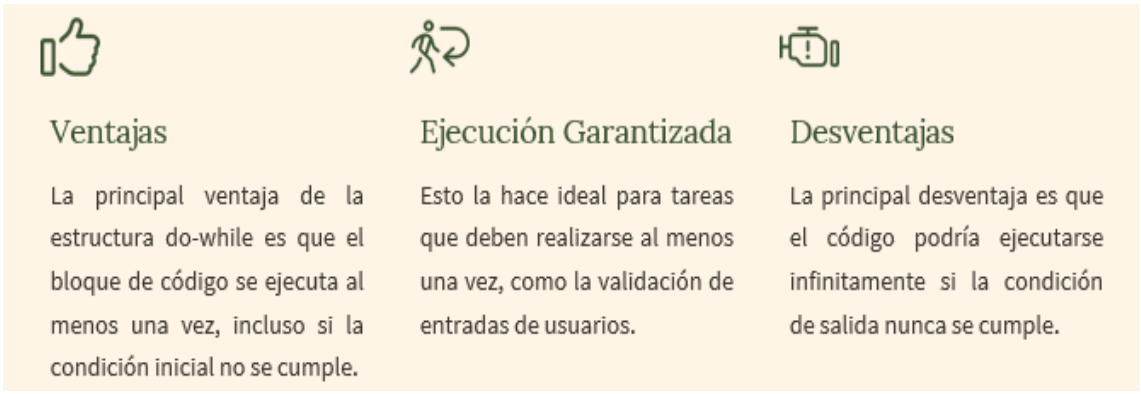
Ejemplos de uso de do-while

- ✓ La estructura de control do-while en C++ se utiliza comúnmente cuando se conoce que el bloque de código debe ejecutarse al menos una vez, independientemente de la condición de salida.

- ✓ Por ejemplo, se puede usar do-while para solicitar repetidamente la entrada de un usuario hasta que se proporcione un valor válido.

- ✓ También es útil en algoritmos que requieren una iteración inicial, como en la búsqueda binaria o el cálculo de raíces cuadradas.

Ventajas y desventajas de do-while



Ejemplos de instrucción while en C++.

Ejemplo1:

```c
//Programa que compara dos valores enteros usando do while
#include <stdio.h>
int main()
{
int clave=123;
int clave1;

printf("\n");
do
{
    printf("Digita la clave correcta:");
    scanf("%d",&clave1);
} while (clave!=clave1);
printf("\nLa clave ha sido digitada correctamente\n\n");
return 0;
}
```

Ejemplo2:

```cpp
// Sumar positivos
#include <iostream>

using namespace std;
int main ()
{
    cout << "Programa que pide números positivos al usuario y calcula la suma de todos ellos," <<endl;
    cout << "finalizando cuando se digite 0 o un número negativo" <<endl;
    double suma = 0;
    double dato;
    do
    {
        cout << "Introduce un dato (0 o negativo para salir) ";
        cin >> dato;
        if (dato > 0)
        {
            suma = suma + dato;
        }
    }
    while (dato > 0);
    cout << "Suma: " << suma << endl;
    return 0;
}
```

Ejemplo3: Escribir un programa que solicite la carga de un número entre 0 y 999, y nos muestre un mensaje de cuántos dígitos tiene el mismo. Finalizar el programa cuando se cargue el valor 0.

```cpp
#include<iostream>

using namespace std;

int main()
{
     int valor;
     do {
         cout <<"Ingrese un valor entre 0 y 999 (0 finaliza):";
         cin >>valor;
         if (valor >= 100)
         {
             cout <<"Tiene 3 dígitos.";
         }
         else
         {
             if (valor >= 10)
             {
                 cout <<"Tiene 2 dígitos.";
             }
             else
             {
                 cout <<"Tiene 1 dígito.";
             }
         }
         cout <<"\n";
     } while (valor != 0);
     return 0;
}
```

- Comprueba el ejemplo, y modificarlo para corregir las inconsistencias que tiene.
- Recuerda que sólo debe permitir números entre 0 y 999.
- Sólo termina el ciclo cuando se digita el cero.

Ejercicios propuestos:

I. Escribe el programa para resolver los siguientes problemas:

1. Realizar un programa que acumule (sume) valores ingresados por teclado hasta ingresar el 9999 (no sumar dicho valor, indica que ha finalizado la carga). Imprimir el valor acumulado e informar si dicho valor es cero, mayor a cero o menor a cero.

2. Escriba el programa para leer 15 números negativos y convertirlos a positivos e imprimir dichos números.

3. Escriba un programa que solicite ingresar diferentes números la vez y que pueda repetir indefinidamente si el usuario lo desea (que el ordenador pregunte ¿Quiere continuar? y la persona pueda responder S o N.

4. Escriba el programa que calcule la suma de los números impares mientras sean distintos de cero.

5. Escribe un programa que muestre los múltiplos de 3 y 5 simultáneamente a partir del 15 hasta el 45.

6. En un banco se procesan datos de las cuentas corrientes de sus clientes. De cada cuenta corriente se conoce: número de cuenta y saldo actual. El ingreso de datos debe finalizar al ingresar un valor negativo en el número de cuenta.
Se pide confeccionar un programa que lea los datos de las cuentas corrientes e informe:
a) De cada cuenta: número de cuenta y estado de la cuenta según su saldo, sabiendo que:
Estado de la cuenta:
'Acreedor' si el saldo es >0.
'Deudor' si el saldo es <0.
'Nulo' si el saldo es =0.
b) La suma total de los saldos acreedores.

7. Analiza el siguiente programa e indica su función:

```
#include <iostream>
using namespace std;
int main(){
  int cont = 1,num;
  cin>>num;

  do{
    cout<<num*cont<<endl;
    cont++;
  }while(cont<=10);

  return 0;
}
```