UNIVERSIDAD NACIONAL DE INGENIERIA

CTIC UNI

CENTRO DE TECNOLOGIAS DE INFORMACION Y COMUNICACIONES

Programación en C++

Sesión 5: Estructura de Control for

Unidad

Sesión 5: Estructuras de Iteración

Contenidos

a) Instrucción for.
b) Mecánica del lazo for.
c) Modificaciones al lazo for.

Aprendizajes esperados

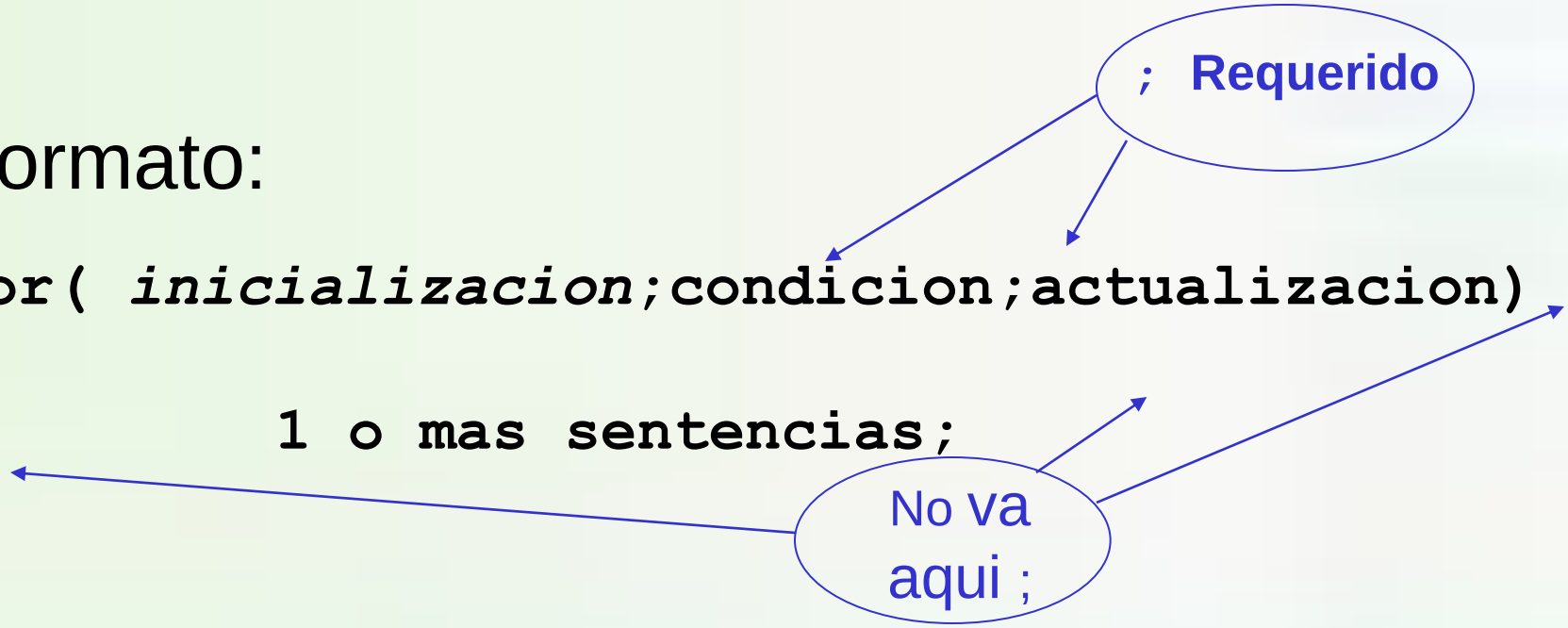
- Comprende el uso de la estructura for para repetir procesos

El lazo **for**

- Es aquel lazo que se ejecuta cero o mas veces
- Util para lazos controlados por contador.

- Formato:

```
for( inicializacion;condicion;actualizacion)
{
        1 o mas sentencias;
}
```



Mecánica del lazo for

```
for(inicializacion; condicion; actualizacion)
{                          // {} se pueden omitir
   sentencia(s);           // si el cuerpo del loop
}                          // tiene 1 instruccion
```

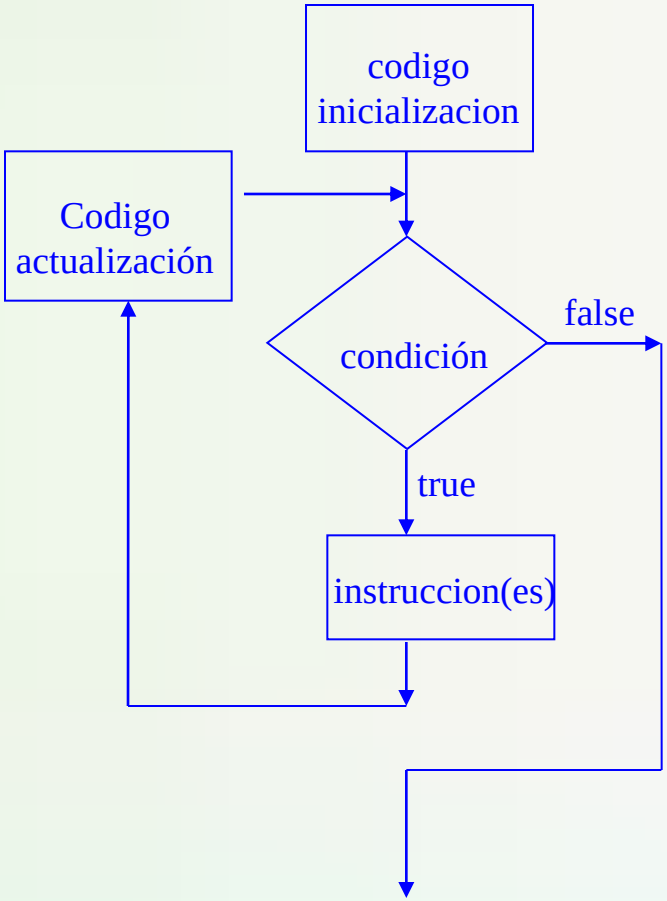
1) Realiza ***inicializacion.***
2) Evalúa expresion ***condicional***:

   3a) Si es **`true`**, ejecuta ***sentencia(s)***

   3b) Si es **`false`**, termina la ejecucion del loop

4) Ejecuta la ***actualización***, luego reevalúa expresión en la ***condición.***

Flujograma del Lazo `for`



Ejercicio:

a) Imprimir los valores del 1 al 100 inclusive, con incremento de 1.

b) Imprimir los valores de 100 a 1 inclusive, con incremento de -1 (decremento)

Ejercicio:

c) Imprimir los valores de 7 a 77 inclusive, con incremento de 7.

d) Imprimir los valores de 20 a 2 inclusive, con incremento de -2 (decremento)

Ejercicio:

e) Imprimir la siguiente secuencia:

0, -1, -2, -3, -4, -5, -6, -7, -8, -9

f) Imprimir la siguiente secuencia de valores y hallar su suma:

2, 5, 8, 11, 14, 17, 20.

g) Imprimir la siguiente secuencia y hallar su suma:

99, 88, 77, 66, 55, 44, 33, 22, 11, 0.

Ejemplo Lazo **for**

```
int suma = 0, num;
for (num = 1; num <= 10; num++)
   suma += num;
cout << "Suma de numeros 1 - 10 es "
     << suma << endl;
```

Modificaciones Lazo for

- Se puede definir variables en el código de inicialización.
  - Su ambito es el lazo **`for`**
- Los códigos de inicialización, condición, o códigos de actualización pueden contener mas de una instrucción
  - Separados con comas.
- Ejemplo:

```
for (int suma = 0, num = 1; num <= 10; num++)
    suma += num;
```

Más Modificaciones al Lazo for
(Estas no se recomiendan)

- Puede omitir la ***inicializacion*** si ya la hizo

```
int suma = 0, num = 1;
for (; num <= 10; num++)
    suma += num;
```

- Puede omitir la ***actualización*** si la hizo en el lazo

```
for (suma = 0, num = 1; num <= 10;)
    suma += num++;
```

- Puede omitir la ***condición*** – pero puede causar un lazo infinito

```
for (suma = 0, num = 1; ; num++)
    suma += num;
```

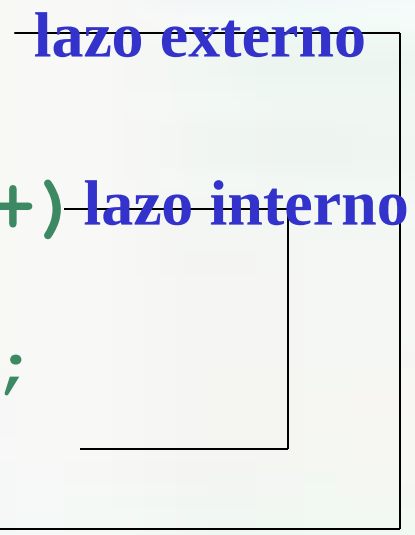
Decidiendo que lazo usar

- **`while`**: evalúa primero la condición (el cuerpo del lazo puede que nunca se ejecute)
- **`for`**: el cuerpo del lazo puede que nunca se ejecute); tiene inicialización y código de actualización; es útil con contadores o si es conocido el número de repeticiones.

Lazos Anidados

- Un lazo anidado es un lazo al interior del cuerpo de otro lazo
- Ejemplo:

```
for (f = 1; f <= 3; f++)
{
   for (col = 1; col <= 3; col++)
   {
      cout << f * col << endl;
   }
}
```



Notas acerca de los Lazos Anidados

- El lazo interior se recorrre totalmente para cada valor de la variable del lazo exterior.
- Las repeticiones del lazo interior se completan más pronto que las del lazo exterior.
- El número total de repeticiones del lazo for interior es producto del número de repeticiones de los dos lazos.

Saliendo de un Lazo

- Puede usar **`break`** para terminar la ejecución de un lazo.

- Cuando es usado en un lazo interior, sólo termina ese lazo y vuelve al lazo exterior.

La sentencia **`continue`**

- Puede usar **`continue`** para ir al final del lazo y prepararse para la siguiente repetición.
  - **`while`** va a la condición y repite el lazo si la condición se evalúa a true.
  - el lazo **`for`** va al paso de actualización, luego evalúa la condición, y repite el lazo si la condición es true.

Usando Lazos para Validación de Datos

Los lazos son la estructura más apropiada para validar datos ingresados por el usuario

1. Pedir y leer los datos.
2. Usar una condición de un lazo para evaluar si el dato es válido.
3. Ingresar al lazo sólo si el dato no es válido.
4. Al interior del lazo, pedir al usuario reingresar los datos.
5. No se podrá salir del lazo hasta que se ingrese un dato válido.

Ejemplo de Validación de Datos

```
cout << “Ingrese un número (1-100)“
cin  >> num;

while (num < 1 || num > 100)
{  cout << "Numero debe estar entre 1 y 100."
        << " Reingrese su número. ";
   cin  >> num;
}
/* Aquí va el código a usar para el número
   válido   */
```

Ejercicio:

1. Hallar la suma siguiente:

$S = 1 + 1/2 + 1/3 + ... + 1/n.$

Ejercicio:

2. Leer del teclado 20 números y para cada número imprimir el mismo número, el número elevado al cubo y el número elevado a la cuarta.

Ejercicio:

3. Elabore un programa que muestre un saludo indefinidamente.

Ejercicio:

4. Escriba un programa que determine la cantidad de pares, la cantidad de impares y la cantidad de ceros de una lista de n números ingresados por el usuario(pueden ser positivos, negativo o ceros).

Ejercicio:

5. Hallar la suma de los términos de la siguiente serie:

$S = 1 - 1/2 + 1/3 – 1/4 + ...+/- 1/N.$

Práctica #1

- Elabore un programa que pida al usuario un entero positivo *n*. Luego, el programa debe pedir al usuario *n* enteros usando un ciclo. El programa deberá encontrar el mayor, menor, y promedio de los valores dados por el usuario e imprimirlos al final.

Ejercicio:

6. Escriba un algoritmo para calcular el promedio de una serie de números impares introducidos por el usuario.

Terminar cuando introduzca el primer número par.

Ejercicio:

7. Escriba un programa que pida un número de cualquier cantidad de cifras y responda si es o no múltiplo de 3 y
si es o no múltiplo de 9.