



# Sentencia for

### ¿Para qué sirve la sentencia for?

Sirve para realizar un conjunto de operaciones un determinado número de veces

Permite repetir una instrucción o una instrucción compuesta un número especificado de veces

permite repetir varias instrucciones (pasos) un cierto número de ocasiones (por ejemplo, 10 veces). Se emplea en el recorrido de vectores, matrices y estructuras, entre otros.

## ¿Cómo funciona la sentencia for?

1

Se inicializa la variable de control y se evalúa la condición-bucle

2

Si condición es verdadera:

- Se ejecuta el cuerpo del bucle
- Se incrementa la variable y se evalúa la condición

3

- Si condición-bucle es falsa:
  - Se termina el ciclo

4

- \* Condiciones:
- \* Cuantitativas

Sintaxis:

```
for(inicio;exp_boolana;incremento){  
    Sentencias;  
}
```

### Sintaxis del ciclo for

Sintaxis:

```
for(inicio;exp_booleana;incremento){  
    Sentencias;  
}
```

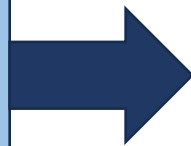
Si la condición es verdadera, se ejecutan las sentencias; si es falsa no se ejecutan y se termina el ciclo

También puede ser decremento

## Ejemplo: ¿Cómo escribo un programa que imprima los números del 1 al 100?

Sin usar ciclo for, una de las posibilidades es: escribir 100 veces un:

```
cout<<"1"<<endl;  
cout<<"2"<<endl;  
cout<<"3"<<endl;  
...  
cout<<"4"<<endl;
```



Con el ciclo for, usaríamos lo siguiente:

```
for(int x=1;x<=100;x++)  
    cout<<x<<endl;
```

La variable x se puede declarar al principio, o bien dentro del for

Inicio

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int x=1;x<=100;x++)
5         cout<<x<<endl;
6 }
```

Expresión  
booleana

Incremento  
de 1 en 1

Variable x, se  
imprime

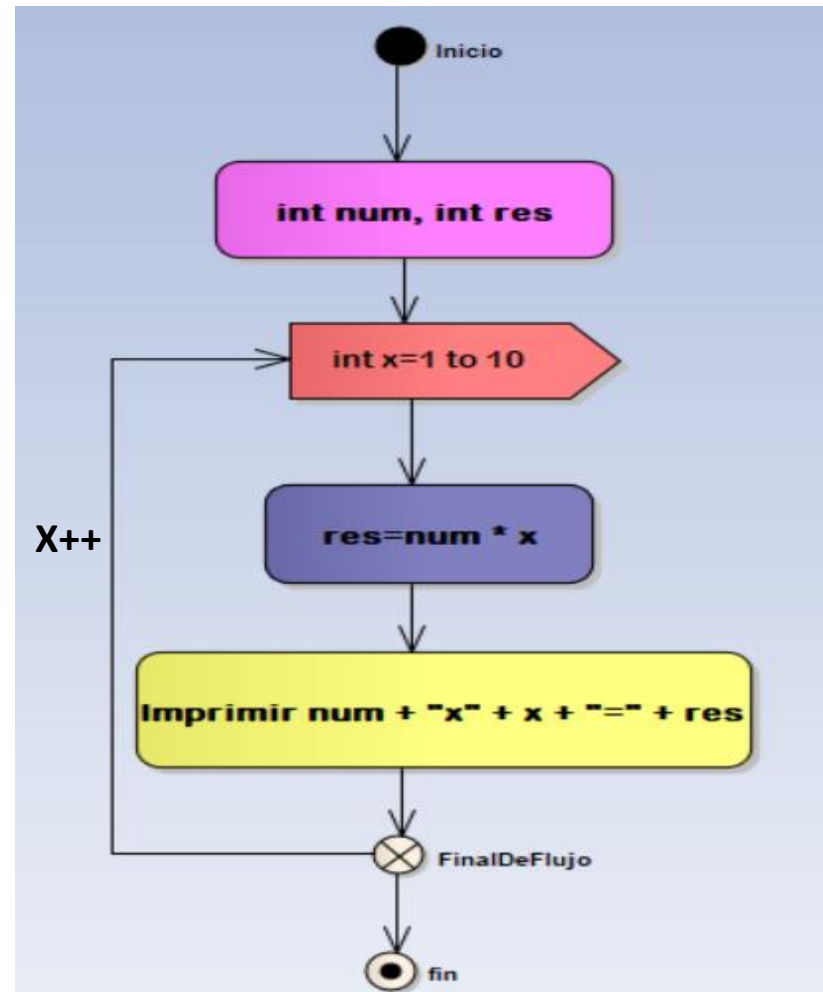
Puede incrementar o  
disminuir y no  
necesariamente de 1 en 1

## Requerimiento:

*Se ocupa un programa para que muestre la tabla de multiplicar de cualquier número entero ingresado por el teclado, exceptuando números negativos y cero. Se debe programar en una función.*

## Algoritmo

Declarar librerías  
Declarar variables: int **num**, int **res**  
Pedir al usuario que escriba el número  
Leer el número  
Crear un ciclo con un iterador que recorra del 1 al 10 y en cada vuelta multiplique **num \* el iterador**, guardando el resultado en **res**  
Imprimir en cada vuelta (**num x iterador=res**)  
Finalizar ciclo  
Finalizar programa

Diagrama  
de flujo

### Código fuente

```
1  #include<iostream>
2  using namespace std;
3  main(){
4      int num, res;
5      cout<<"ingrese un numero"<<endl;
6      cin>>num;
7      for(int x=1;x<=10;x++){
8          res=num*x;
9          cout<<num<<"x"<<x<<"="<<res<<endl;
10     }
11 }
```

### Detalles con el ciclo for

No son necesarias las llaves, si solo se desea ejecutar una instrucción debajo del for

La variable que trabaja dentro del ciclo for se llama **“iterador”**

El for no debe llevar ; al final. Si se le coloca, no ejecutará nada

Los ciclos infinitos suceden cuando la expresión booleana no cambia de verdadero a falso

El di4blo  
está en los  
detalles



No son necesarias las llaves, si solo se desea ejecutar una instrucción debajo del for

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int x=100;x<=200;x++)
5         cout<<x<<endl;
6 }
```

Este programa  
imprime los  
números del 100 al  
200

No son  
necesarias las  
llaves

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int x=100;x<=200;x++){
5         cout<<x<<endl;
6     }
7 }
```

Si se ponen no  
pasa nada

La variable que trabaja dentro del ciclo for se llama **“iterador”**

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int i=100;i>=1;i--)
5         cout<<i<<endl;
6 }
```

Iterador

Se puede usar cualquier nombre para el iterador, por lo regular muchos programadores usan la variable **“i”**

Este programa imprime los números del 100 al 1

El for no debe llevar ; al final. Si se le coloca, no ejecutará nada

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int i=100;i>=1;i--);
5     cout<<i<<endl;
6 }
```

Al ponerle ; marcaría un error dado que en la línea 5 se hace referencia al iterador "i" pero no será reconocido dado que solo vive en el for



```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int i=100;i>=1;i--);
5     cout<<"Hola a todos"<<endl;
6 }
```

Sólo imprime una vez  
la frase

Este programa pretende imprimir 100 veces en pantalla la frase "hola a todos". Al compilarlo no marcaría error, solo que imprimirá una sola vez la frase ya que el for tiene al final un ; que detiene su trabajo

Los ciclos infinitos suceden cuando la expresión booleana no cambia de  
verdadero a falso

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     for(int i=100;i>=1;i++)
5         cout<<"Hola a todos"<<endl;
6 }
```

La expresión booleana no  
cambia nunca a falso

En este ejemplo, la expresión booleana nunca cambia su valor de verdadero a falso, ya que el iterador va **aumentando**; por lo tanto **i** siempre será **mayor que 100**.

Esto provoca que la frase “**Hola a todos**” se imprima de forma **infinita**.

### Ciclos for anidados

Los ciclos for se pueden **anidar** dependiendo del requerimiento a resolver. Todas las sentencias de **control** pueden ser **anidadas**

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     int res;
5     for(int i=1;i<=10;i++){
6         for(int j=1;j<=10;j++){
7             res=i*j;
8             cout<<i<<"X"<<j<<"="<<res<<endl;
9         }
10    cout<<"\n";
11 }
12 }
```

For externo  
(trabaja de 1 a 10)

For interno(trabaja  
de 1 a 10)

Este programa imprime las tablas  
del 1 al 10.

### Contadores y acumuladores

Son variables que se utilizan de forma “especial” para poder sumar valores, acumular o bien para contar un número de sucesos o repeticiones dentro de un programa o aplicación.

Se declaran como cualquier otra variable, normalmente se inicializan al momento de declararlos

```
int acum=0;  
Int cont=0;
```

## Acumuladores

### Requerimiento:

Hacer un programa que solicite diez números enteros  
Por el teclado y que al final imprima la suma de los mismos

Para este requerimiento se necesita un **ACUMULADOR**

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     int num;
5     int acum=0;
6     for(int i=0;i<10;i++){
7         cout<<"ingrese un numero"<<endl;
8         cin>>num;
9         acum=acum+num;
10    }
11    cout<<"La suma de los diez numeros es: "<<acum;
12 }
```

Se declara el acumulador

Aquí se acumula el valor (se hace la suma)

## Contadores

### Requerimiento:

Hacer un programa solicite 10 números enteros por teclado y que indique **cuántos** de esos diez números son pares.

Para este requerimiento se necesita un **CONTADOR**

```
1 #include<iostream>
2 using namespace std;
3 main(){
4     int num;
5     int contador=0;
6     for(int i=0;i<10;i++){
7         cout<<"ingrese un numero"<<endl;
8         cin>>num;
9         if(num%2==0)
10            contador=contador+1;
11     }
12     cout<<"Hay "<<contador<<" | numeros Pares";
13 }
```

Variable contador, inicializada en cero

Se incrementa el valor del contador de 1 en 1