Programación de Ordenadores

Ingeniería Química

4

David Pelta

Depto de Ciencias de la Computación e I.A.

Universidad de Granada

Indice

- Estructura secuencial.
  - Ejemplos
- Estructuras condicionales.
  - Condicional Simple
  - Condicional Doble
  - Condicional Multiple
  - Ejemplos
- Estructuras repetitivas.
  - Bucles Controlados por condición
  - Bucles Controlados por contador
  - Ejemplos

Conceptos Básicos

Las estructuras de control de un lenguaje de programación se refieren al orden en que las instrucciones de un algoritmo se ejecutarán. El orden de ejecución de las sentencias o instrucciones determinán el flujo de control.

Las tres estructuras básicas de control son:

- ***secuencia***
- ***selección***
- ***repetición***

Conceptos Básicos

Bôhm y Jacopin (1996): *un programa propio puede ser escrito utilizando las tres estructuras de control básicas.*

Un programa se define como *propio* si cumple lo siguiente:

- Posee un solo punto de entrada y salida o fin para control del programa.
- Existen caminos desde la entrada hasta la salida que se pueden seguir y que pasan por todas las partes del programa.
- Todas las instrucciones son ejecutadas y no existen lazos o bucles infinitos.

Estructura Secuencial

Las sentencias se ejecutan sucesivamente, en el orden en que aparecen. No existen “saltos” o bifurcaciones.

```
// Calculo de Raices de una ecuación de grado 2.
#include <iostream>
#include <math>
int main(){
double a,b,c,r1,r2;
1 cout << "\nIntroduce coeficiente de 2o grado: ";
2 cin >> a;
3 cout << "\nIntroduce coeficiente de 1er grado: ";
4 cin >> b;
5 cout << "\nIntroduce coeficiente independiente: ";
6 cin >> c;
7 r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
8 r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
9 cout << "Las raíces son" << r1 << " y " << r2 << endl;
}
```

Estructura Secuencial

Si bien el programa anterior es correcto, no es capaz de manejar situaciones excepcionales.

Por ejemplo, que pasa en la sentencia:

```
r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
```

si ***a = 0?***

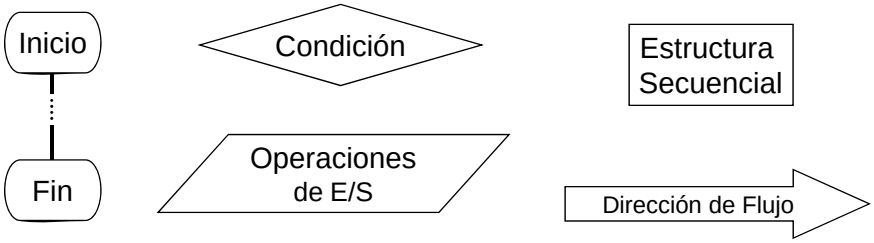
En este caso obtendríamos un error de ejecución por intentar hacer una división por cero.

Solución: estructuras condicionales

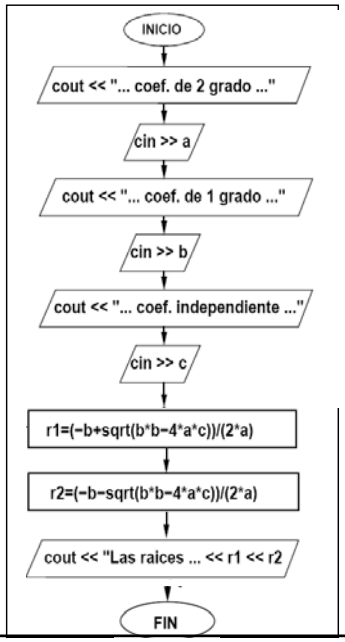
Pero antes....

Diagrama de Flujo

- Es una representación gráfica de un algoritmo
- Se construyen a partir de símbolos especiales que se conectan mediante flechas que indican el flujo de ejecución.



Ejemplo del Cálculo de Raíces



Estructura Condicional

Tambien recibe el nombre de *"estructura de selección"*

Permite elegir entre diferentes cursos de acción en función de condiciones.

```
Si la nota del examen es mayor o igual que 5
     mostrar "Aprobado"
```

Si la condición es *verdadera*, entonces se ejecuta la sentencia ***mostrar,*** y luego el programa continuaría en la sentencia siguiente al ***Si***

Si la condición es *falsa*, la sentencia **mostrar** se ignora y el programa continúa

Estructura Condicional Simple

La instrucción en pseudo código

```
Si la nota del examen es mayor o igual que 5
     mostrar "Aprobado"
```

Se traduce a C++ mediante una sentencia condicional simple:

```
if ( nota >= 5 )
   cout << "Aprobado";
```

La forma general es:

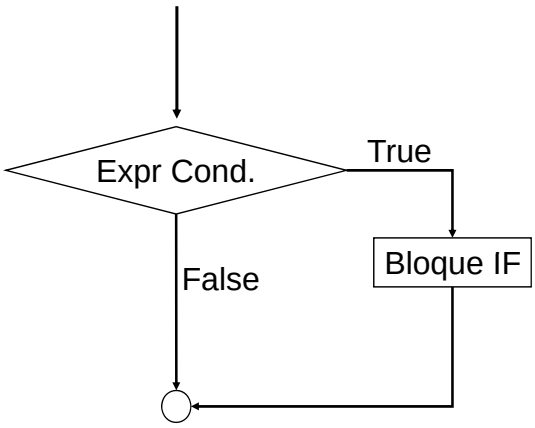
```
if (<condicion>)
   <bloque if>;
```

Una expresión lógica

Una sentencia o conjunto de sentencias (encerradas entre {})

Estructura Condicional Simple

El diagrama de flujo asociado es:



Cálculo de Raíces

```
// Calculo de Raices: control de la variable a
#include <iostream.h>
#include <math.h>
int main(){
double a,b,c,r1,r2;
1 cout << "\nIntroduce coeficiente de 2o grado: ";
2 cin >> a;
3 cout << "\nIntroduce coeficiente de 1er grado: ";
4 cin >> b;
5 cout << "\nIntroduce coeficiente independiente: ";
6 cin >> c;
7 if (a!=0) {
8     r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
9     r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
10    cout << "Las raíces son" << r1 << " y " << r2;
      }
}
```

Ejemplo

```
#include <iostream.h>

int main(){
int dato1,dato2;
char opcion;

cout << "Introduce el primer operando: ";
cin >> dato1;
cout << "Introduce el segundo operando: ";
cin >> dato2;
cout << "Selecciona (S) sumar, (R) restar: ";
cin >> opcion;

if (opcion == 'S')
      cout << "Suma = " << dato1+dato2 << endl;

if (opcion == 'R')
      cout << "Resta = " << dato1-dato2 << endl;

if ((opcion != 'R') && (opcion != 'S'))
      cout << "Ninguna operación\n";
}
```

Estructura Condicional Doble

Permite elegir entre 2 cursos de acción diferentes en función del valor de verdad de una expresión lógica.

Si la nota del examen es mayor o igual que 5
mostrar *"Aprobado"*
Si no
mostrar *"Desaprobado"*

En C++:

```
if ( nota >= 5 )
  cout << "Aprobado";

else

  cout << "Desaprobado";
```

La forma general es:

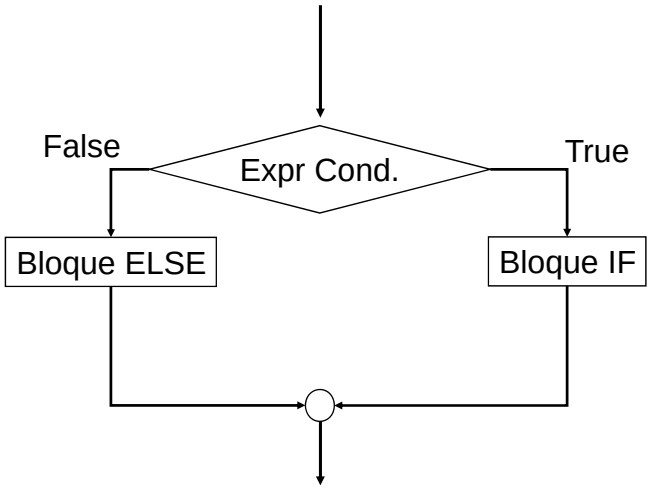
```
if( <condicion> )
 < bloque if >;

else

 < bloque else >;
```

Estructura Condicional Doble

El diagrama de flujo asociado es:

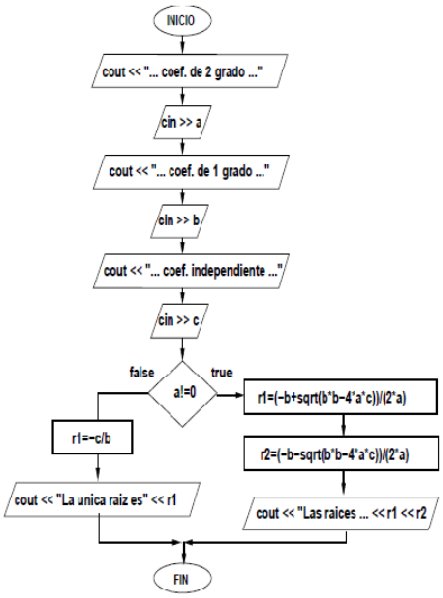


Cálculo de Raíces

En la ecuación de segundo grado, cuando **a = 0**, entonces la raíz es única y vale ***–c/b***

```
if (a!=0) {
   r1 = ( -b + sqrt( b*b-4*a*c ) ) / (2*a);
   r2 = ( -b - sqrt( b*b-4*a*c ) ) / (2*a);
   cout << "Las raíces son" << r1 << "y“ << r2;
     }
else {
     r1 = -c / b;
     cout << “La unica raiz es: “ << r1;
     }
```

Cálculo de Raíces



Anidamiento de Estructuras Condicionales

```
Si la nota es mayor o igual que 9
  imprimir "Sobresaliente"
  else
   Si la nota es mayor o igual
   que 7
       imprimir "Notable"
    else
   Si la nota es mayor o igual
   que 5
        imprimir"Aprobado"
      else
         imprimir"Suspenso"
```

```
if ( nota >= 9 )
    cout << "Sobre";

else if ( nota >= 7 )
     cout << "Notable";

else if ( nota >= 5 )
     cout << "Aprobado";

else
     cout << "Suspenso";
```

¿ Cuando se ejecuta cada instrucción ?

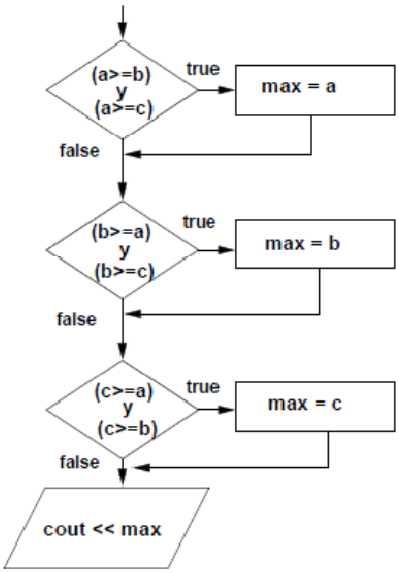
```
if (condic_1)
  {inst_1;
   if (condic_2)
     {inst_2;
     }
   else
    { inst_3;
    }
  inst_4;
  }
else
  {   inst_5;
  }
```

	condic_1	*condic_2*
inst_1	*true*	*independiente*
inst_2	*true*	*true*
inst_3	*true*	*false*
inst_4	*true*	*independiente*
inst_5	*false*	*independiente*

El mayor de 3 valores (I)

```
if ((a>=b) && (a>=c))
  max = a;
if ((b>=a) && (b>=c))
  max = b;
if ((c>=a) && (c>=b))
  max = c;
cout << "El mayor es " << max << endl;
```

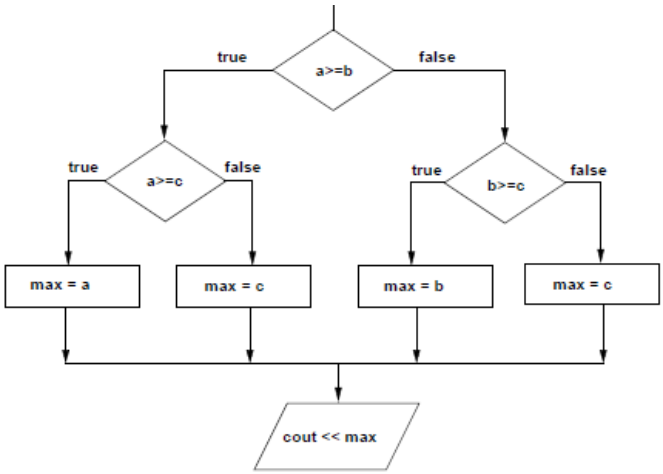
El mayor de 3 valores (I)



El mayor de 3 valores (II)

```
if (a>=b)
  if (a>=c)
     max = a;
  else
     max = c;
else
  if (b>=c)
     max = b;
  else
     max = c;
cout << "El mayor es " << max;
```

El mayor de 3 valores (II)



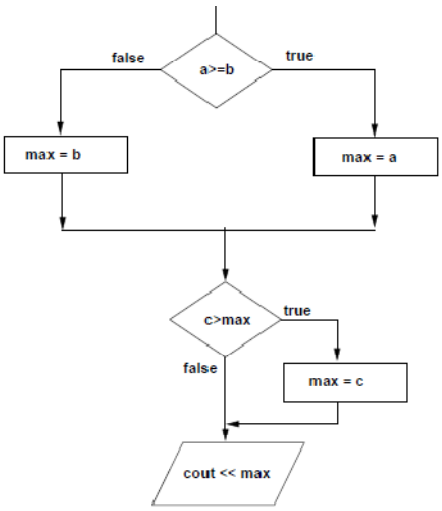
El mayor de 3 valores (III)

```
if (a >= b)
  max = a;
else
  max = b;

if (c > max)
  max = c;

cout << "El mayor es " << max;
```

El mayor de 3 valores (III)



Estructura Condicional Múltiple

Permite definir en una sola estructura, una serie de pares (condición, acción), con condiciones mutuamente excluyentes.

Muy utilizada en la construcción de menús.

```
switch (<expresión>) {
case<constante1>:<sentencias1>
                    break;
case <constante2>:<sentencias2>
                     break;
   ..........................
[default: <sentencias>]
}
```

Estructura Condicional Múltiple

- **<expresión>** es un expresión entera.
- **<constante1>** ó **<constante2>** es un literal tipo entero o tipo carácter.
- ***switch*** sólo comprueba la igualdad.
- No debe haber dos casos (**case**) con la misma ***<constante>*** en el mismo ***switch***. Si esto ocurre, sólo se ejecutan las sentencias correspondientes al caso que aparezca primero.
- El identificador especial ***default*** permite incluir un caso por defecto, que se ejecutará si no se cumple ningún otro. Se suele colocar como el último de los casos.

Ejemplo 1

```
#include <iostream.h>

int main(){
int dato1,dato2;
char opcion;

cout << "Introduce el primer operando: ";
cin >> dato1;
cout << "Introduce el segundo operando: ";
cin >> dato2;
cout << "Selecciona (S) sumar, (R) restar: ";
cin >> opcion;

switch (opcion){
 case 'S': {cout << "Suma = " << dato1+dato2 << endl;
            break;}

 case 'R': {cout << "Resta = " << dato1-dato2 << endl;
            break;}

 default: cout << "Ninguna operación\n";
      }
}
```

Ejemplo 2

```
cout << "Introduce el primer operando: ";
cin >> dato1;
cout << "Introduce el segundo operando: ";
cin >> dato2;
cout << "Selecciona (S) sumar, (R) restar: ";
cin >> opcion;

switch (opcion){
 case 's':
 case 'S': {cout << "Suma = " << dato1+dato2 << endl;
            break;}

 case 'r':
 case 'R': {cout << "Resta = " << dato1-dato2 << endl;
            break;}

 default: cout << "Ninguna operación\n";
      }
}
```

Ejercicios

- Supongamos que un producto cuesta 17345 euros. A cuantas pesetas equivale?. Implemente un programa que permita contestar a esta pregunta. (Recuerde 1 euro = 166.386 pesetas)
- Como debería hacer para implementar un programa que permita hacer la transformación de cualquier cantidad de euros ?
- Extienda el programa para que muestre un mensaje si el cliente es “millonario” en pesetas. En caso contrario, mostrará otro mensaje.
- Implemente un programa que “tire” un dado N veces y cuente las veces que salió cada número. Suponga que existe una función *tirarDado()*.

Estructuras Repetitivas

Estructuras Repetitivas

Las estructuras repetitivas son también conocidas como *bucles, ciclos o lazos.*

Una estructura repetitiva permite la ejecución de un conjunto de sentencias:

- hasta que se satisface una determinada condición (controladas por condición)
- un número determinado de veces (controladas por contador)

Bucles Controlados por Condición

Se ejecuta el conjunto de sentencias de interés mientras la condición sea verdadera.

Existen dos formas básicas de construcción:

Pre - Test

```
while (< condición>){
  <cuerpo del bucle>
}
```

Primero se pregunta y luego se ejecuta

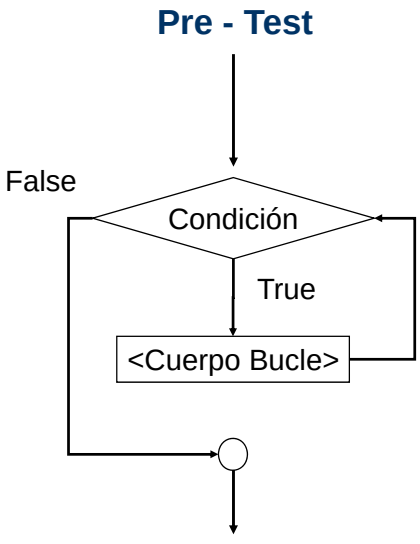
Post - Test

```
do{
 <cuerpo del bucle>
}while (< condición>)
```

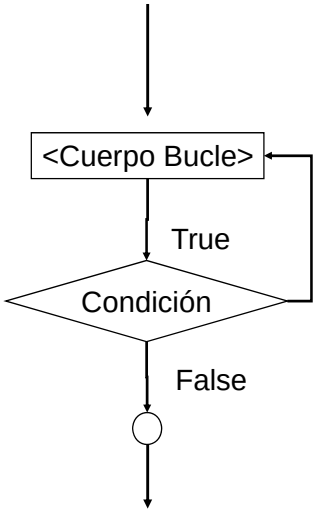
Primero se ejecuta luego se pregunta

Bucles (Pre / Post) Test

Pre - Test



Post - Test



Ejemplo

Escribir 10 líneas con 4 estrellas cada una

```
cont = 1;

do {
 cout << "****" << endl;
 cont = cont + 1;
}while (cont <= 10)
```

```
cont = 0;

do {
 cout << "****" << endl;
 cont = cont + 1;
}while (cont < 10)
```

```
cont = 1;

while (cont <= 10)
{cout << "****" << endl;
 cont = cont + 1;
}
```

```
cont = 0;

while (cont < 10)
{cout << "****" << endl;
 cont = cont + 1;
}
```

Ejemplo

Escribir *tope* líneas con 4 estrellas cada una

```
cin >> tope
cont = 0;

do {
 cout << "****" << endl;
 cont = cont + 1;
}while (cont < tope)
```

Problema si *tope = 0*

Solución

```
cin >> tope;
cont = 0;

while (cont < tope)
{cout << "****" << endl;
 cont = cont + 1;
}
```

Ejemplo

Escribir los números pares entre [0..*tope*]

```
cin >> tope
i = 0;
while (i <= tope)
  {cout << i << endl;
   i = i + 2;
  }
```

¿Qué ocurre si el código anterior se cambia por :

```
cin >> tope
i = 0;
while (i <= tope)
  {i = i + 2;
   cout << i << endl;
  }
```

¿Que hace este programa?

```
int main(){
 int i,j;

     i = 0;
     while(i < 4) {
       j = 0;
       while ( j < i){
         cout << i << "+" << j << "="
              << i+j << endl;
         j = j + 1;
       }
       i = i + 1;
     }
   return(0);
}
```

Ejemplo

1. Hacer un programa que muestre los pares (1,10), (2,9), (3,8)...
2. Tirar N veces un dado y contar las ocurrencias de cada valor.

Ejemplo: control por centinela

Sumar valores que se leen por teclado hasta que se lee -1. (el valor utilizado para detener el procesamiento se conoce como centinela)

```
suma = 0;
do
{cin >> valor;
 suma = suma + valor;
}while (valor != -1)
cout << "La suma es"
     << suma;
```

Se procesa el valor centinela. Esta clase de problemas se evita utilizando la técnica de *lectura anticipada*.

Lectura anticipada: se usa un bucle pre-test y se comprueba el primer valor.

```
suma = 0;
cin >> valor;
while (valor != -1)
 {suma = suma + valor;
  cin >> valor;
 }
cout << "La suma es"
     << suma;
```

Ciclos post-test como Filtros en la Entrada de Datos

Los ciclos post-test son útiles para forzar que los datos de entrada pertenezcan a un rango o conjunto de opciones predeterminados.

```
// filtro para valores positivos
do
 {cout << "Introduzca un valor > 0:";
  cin >> valor;
 }while (valor < 0)
```

```
// filtro para 's','S','n','N'
do
 {cout << "Desea Salir (s/n)?:";
  cin >> opc;
 }while((opc != 's')&&(opc !='S')
      &&(opc != 'n')&&(opc != 'N'))
```

Problema para Resolver

Tenemos que realizar un programa que calcule la nota promedio de un conjunto de notas. La cantidad de notas puede variar en cada ejecución del programa.

- Utilizaremos un valor "*centinela*" para indicar el fin de las notas.
- El proceso terminará cuando se ingrese dicho valor
- El valor concreto del centinela debe ser diferente al de cualquier posible entrada válida (en este caso podría ser -1)

Pseudo Código y Refinamientos Sucesivos

Nuestro problema es:

Determinar el promedio de las notas de un examen

Esta tarea la podemos subdividir en otras más pequeñas:

1. *Inicializar variables*
2. *Ingresar, Sumar y Contar las notas del examen*
3. *Calcular e imprimir el promedio*

La fase de inicialización *Inicializar variables* se puede refinar a:

Inicializar la variable total a cero
Inicializar el contador de notas a cero

La fase Ingresar, Sumar y Contar las notas del examen

se puede refinar a

Ingresar la primera nota (posiblemente el centinela)
Mientras el usuario no ingrese el valor centinela
sumar la nota al total
sumar uno al contador
Ingresar la siguiente nota (posiblemente el centinela)

La fase Calcular e imprimir el promedio se puede refinar a

Si la cantidad de notas es distinta de cero
el promedio es la suma de las notas dividido la cantidad
Imprimir el promedio
En caso contrario
Imprimir “No se ingresaron notas"

Resolución en Pseudo código

Inicializar la variable total a cero
Inicializar el contador de notas a cero
Ingresar la primera nota (posiblemente el centinela)

Mientras el usuario no ingrese el valor centinela
sumar la nota al total
sumar uno al contador
Ingresar la siguiente nota (el centinela ??)

Si la cantidad de notas es distinta de cero
el promedio es la suma de las notas dividido la cantidad
Imprimir el promedio
En caso contrario
Imprimir “No se ingresaron notas"

Resolución en C++

```
const double CENTINELA = -1.0;
int main()
{ double contador, suma, avg, nota;

 suma = 0; contador = 0;
 cout << "Ingrese la nota (-1 para fin) ";
 cin >> nota;

 while (nota != CENTINELA)
  {suma = suma + nota;
   contador = contador + 1;
   cout << "Ingrese la nota (-1 para fin) ";
   cin >> nota;
  }
 if(contador > 0)
   {avg = suma / contador;
    cout << " El promedio es " << avg;
   }
 else
  cout << " No se ingresaron notas ";
}
```

Ejecutar

Bucles controlados por Contador: FOR

Si conocemos exactamente la cantidad de veces que necesitamos repetir un conjunto de sentencias, entonces podemos usar un bucle FOR.

En general, los bucles controlados por contador requieren

- una variable de control o contador
- Un valor inicial para el contador
- Un valor final para el contador
- Una condición para verificar si la variable de control alcanzó su valor final.
- Un valor de incremento (o decremento) con el cual se modifica la variable de control en cada bucle

Bucles controlados por Contador: FOR

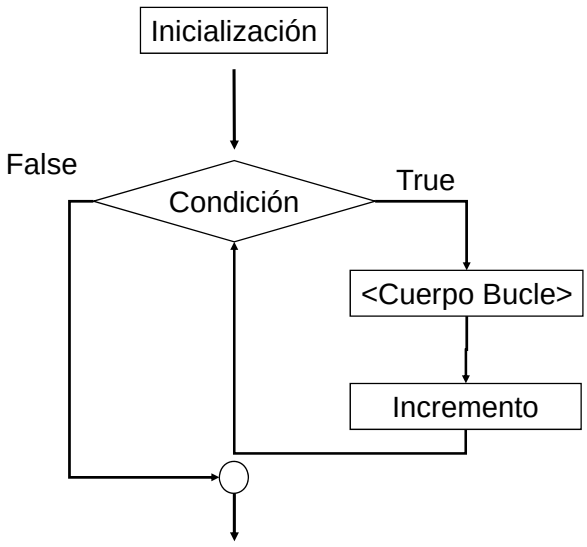
La forma general del bucle FOR es:

for (inicializacion; condicion continuación, incremento)
< conjunto de sentencias>

Ejemplo: imprimir los valores enteros entre 1 y 10

```
for( nro = 1; nro <= 10; nro = nro + 1)
     cout << nro << endl;
```

Diagrama de Flujo del FOR



Relación entre FOR y WHILE

Se debe notar que los bucles FOR se pueden reescribir como bucles WHILE

```
for ( inicializacion; condicion continuación, incremento )
  < conjunto de sentencias>
```

```
inicializacion
while (condicion continuacion)
  < conjunto de sentencias>
  incremento
```

Ejemplos

```
// tabla de multiplicar de un nro dado
int i,nro;
cin >> nro;
for(i = 1; i <= 10; i = i + 1)
 {cout << nro * i << endl;
  }
```

```
// sumar los nros pares entre 2 y 200
int i,suma;
suma = 0;
for(i = 2; i <= 200; i = i + 2)
 { suma = suma + i;
 }
cout << "la suma es " << suma << endl;
```

Ejercicio

- Hacer un programa que lea 10 valores enteros y muestre cual ha sido el maximo y cual el minimo.
- Hacer un programa que calcule la nota media para cada alumno de un conjunto, asi como la media de las notas de cada asignatura.
  - No se conoce el nro de alumnos,
  - Cada uno de ellos ha cursado 5 asignaturas.
  - Los alumnos se identifican por un número entre 0 y 10000