- Unidad 3

INSTRUCCIONES DE REPETICION

(Capitulo 5 bibliografía)

Estructuras de ciclos – Ciclos FOR

» Ventre, Luis O.

Estructuras Básicas de ciclos

- Construir una sección de código repetitiva requiere cuatro elementos:

1) **Instrucción de repetición**
2) **Condición de repetición**
3) **Instrucción Condición de inicio**
4) **Instrucción de Salida.**

1) **<u>Instrucción de repetición:</u>**

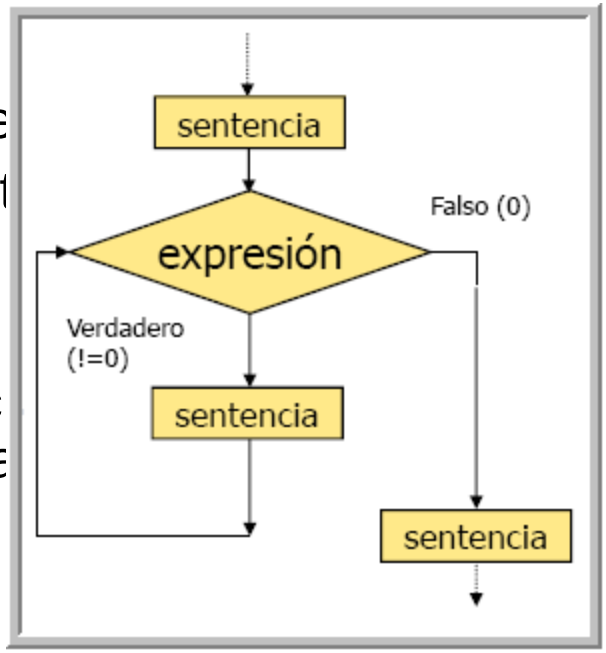
Esta define los **limites** que contienen la *<u>sección de código</u>* repetitiva. Y controla si el código se ejecutara o no. C++ incorpora 3 diferentes:

WHILE - **FOR** - **DO WHILE**

2) Condición de repetición:

Es la condición que debe evaluarse para
condiciones válidas son iguales a las de las e

Si la condición es verdadera el ciclo es ejecut



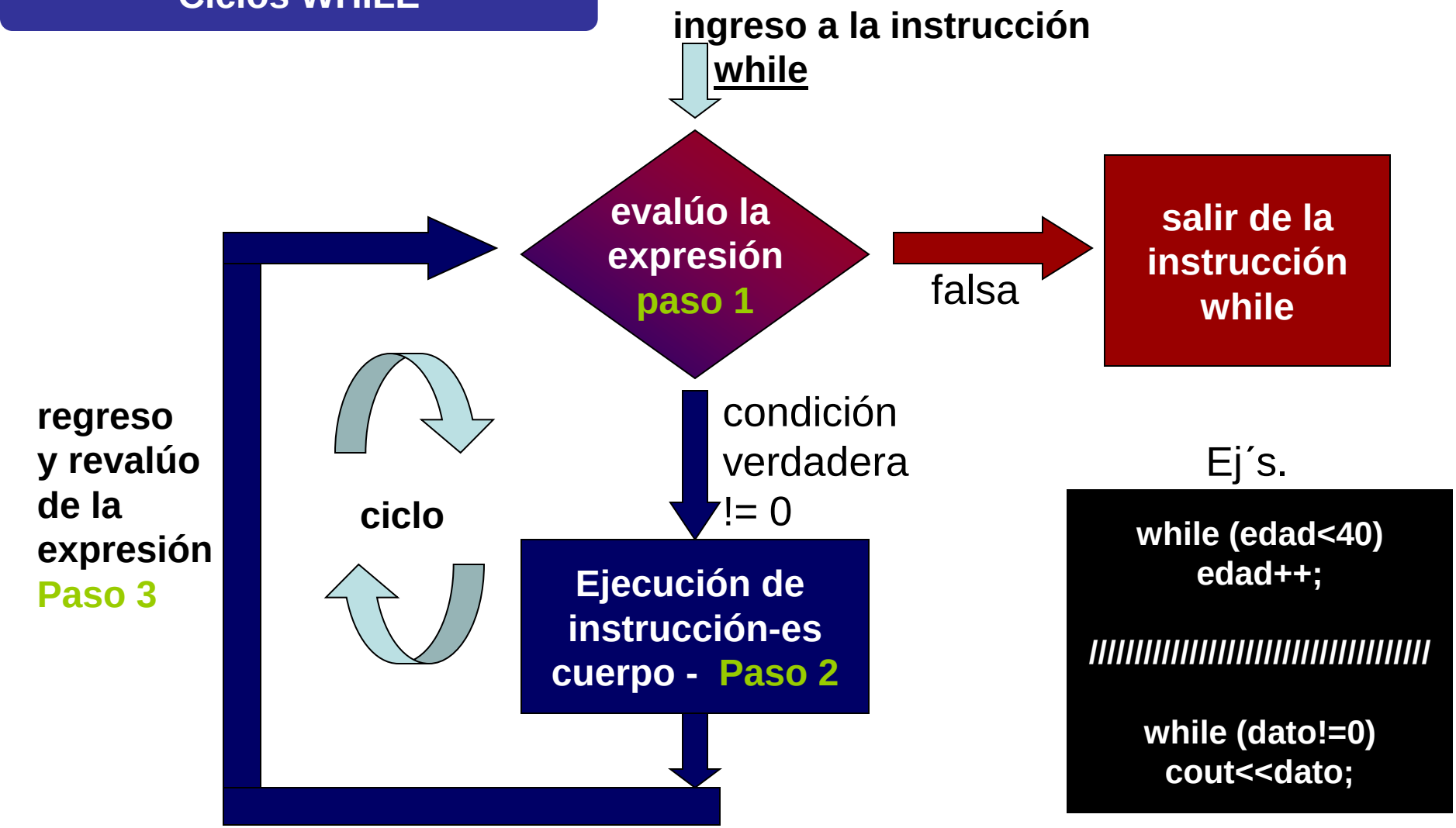
3) Instrucción Condición de inicio:

Instrucción que establece la condición al inic
siempre antes de que la condición sea eva
para asegurar la ejecución correcta del ciclo.

4) Instrucción de Salida:

Debe existir en el ciclo, una instrucción que permita volver falsa la condición de ejecución del ciclo. Esto es necesario con el objetivo de poder detener las repeticiones en algún momento.

Ciclos WHILE



Ej´s.

```
while (edad<40)
edad++;

//////////////////////////////////

while (dato!=0)
cout<<dato;
```

Ciclos WHILE

- **Veamos un ejemplo:**

```
while (cuenta<=10)
cout<<cuenta;
```

- **Dos posibles errores los ve?**
- **Debo inicializar la variable cuenta:**

```
cuenta=1;
while (cuenta<=10)
cout<<cuenta;
```

- **Y debo poder hacer falsa la condición**
  para esto puedo utilizar una instrucción compuesta en lugar de una simple

```
cuenta=1;
while (cuenta<=10)
{
cout<<cuenta;
cuenta++;
}
```

Ciclos WHILE Interactivos

- En los programas vistos, todos los **ciclos son de cuenta fija**, ya que se utiliza un **contador** para determinar el fin del ciclo

- En muchos programas es necesario construir **ciclos de condición variable.**

- Existen casos donde es necesario ingresar datos de manera continua, en ellos se ingresan datos hasta que un determinado valor, denominado "**centinela**" es ingresado y este detiene el ciclo.

- Evidentemente un requisito fundamental es que el valor centinela no pueda confundirse con los valores esperados en el programa.

- Veamos un ejemplo

Ciclos WHILE Interactivos

- Desarrolle un programa llamado SUMASADO. El mismo tendrá los siguientes objetivos:

- 1-Hay un numero determinado de comensales. Suponga 5.
- 2-Debe solicitar a los comensales, que cada uno ingrese por teclado el importe que gasto.
- 3- Al finalizar el programa, debe indicar el costo total del asado.

- Desarrolle una nueva versión de SUMASADO. El mismo tendrá los siguientes objetivos:

- 1-Hay un numero determinado de comensales. Suponga 5.
- 2-Debe solicitar a los comensales, que cada uno ingrese por teclado el importe que gasto.
- 3- Al finalizar el programa, debe indicar el costo total del asado y cuanto debe pagar cada uno, entendiendo que todos paga lo mismo.

Ciclos WHILE Interactivos

- Desarrolle una nueva versión de SUMASADO. El mismo tendrá los siguientes objetivos:

- 1-El numero de comensales se solicita que se ingrese por teclado.
- 2-Debe solicitar a los comensales, que cada uno ingrese por teclado el importe que gasto.
- 3- Al finalizar el programa, debe indicar el costo total del asado y cuanto debe pagar cada uno, entendiendo que todos paga lo mismo.
- Desarrolle la ultima versión de SUMASADO. El mismo tendrá los siguientes objetivos:

- 1-El numero de comensales se desconoce.
- 2-Debe solicitar a los comensales, que cada uno ingrese por teclado el importe que gasto, para finalizar el ingreso se debe colocar -99.
- 3- Al finalizar el programa, debe indicar el costo total del asado y cuanto debe pagar cada uno, entendiendo que todos paga lo mismo.

Ciclos WHILE Interactivos

- Suponga que debe implementar una versión de SUMASADO. La cual acepta datos y los acumula para el cálculo del costo total HASTA que ingresa el valor -99. En ese momento se corta el ingreso e indica costo total y promedio por persona:

```
 9    cout<<"Ingrese el costo de cada persona, para finalizar la suma ingrese -99"<<endl;
10    cin>>costounit;
11    while(costounit!=-99)
12    {
13       suma+=costounit;
14
15       cantidadpers++;
16
17       cout<<"Ingrese el costo de cada persona, para finalizar la suma ingrese -99"<<endl;
18       cin>>costounit;
19
20       }
```

Intro FOR

- Como hemos visto la construcción de ciclos en C++ puede realizarse de varias formas. Una de estas es utilizando una **instrucción FOR**, cuya traducción puede interpretarse como **“para” o “para cada”.**

- En muchas situaciones, en particular aquellas que se necesita de una **cuenta fija,** o sea se conoce en un comienzo la cantidad de iteraciones a realizar es mas fácil el **formato de la instrucción FOR.**

  - **La sintaxis de la instrucción FOR tiene la siguiente forma:**

```
for (lista de inicialización; expresión; lista de alteración)
    instrucción;
```

Ciclos FOR

- Aunque esta instrucción **parezca compleja**, es bien simple si analizamos sus **componentes por separado**.

- La sintaxis de esta instrucción muestra entre paréntesis 3 elementos OPCIONALES, **separados por ";";** aunque los elementos pueden faltar, los ";" deben estar SIEMPRE.

```
for (lista de inicialización; expresión; lista de alteración)
      instrucción;
```

- **lista de inicialización:** en su forma mas común, consiste de una sola instrucción usada para **establecer el valor inicial de un contador**.

OJO

- Al finalizar la instrucción FOR, NO se debe colocar ;

```
for (lista de inicialización; expresión; lista de alteración)
      instrucción;
```

- Al finalizar la instrucción WHILE, NO se debe colocar ;

```
while (expresion)
      instrucción;
```

Ciclos FOR

```
for (lista de inicialización; expresión; lista de alteración)
instrucción;
```

- **expresión**: contiene el valor **máximo o mínimo** que puede tener el contador y determina **cuando se termina el ciclo**.

- **lista de alteración:** proporciona el valor de incremento que se **suma o resta del contador cada vez que se ejecuta el ciclo**.

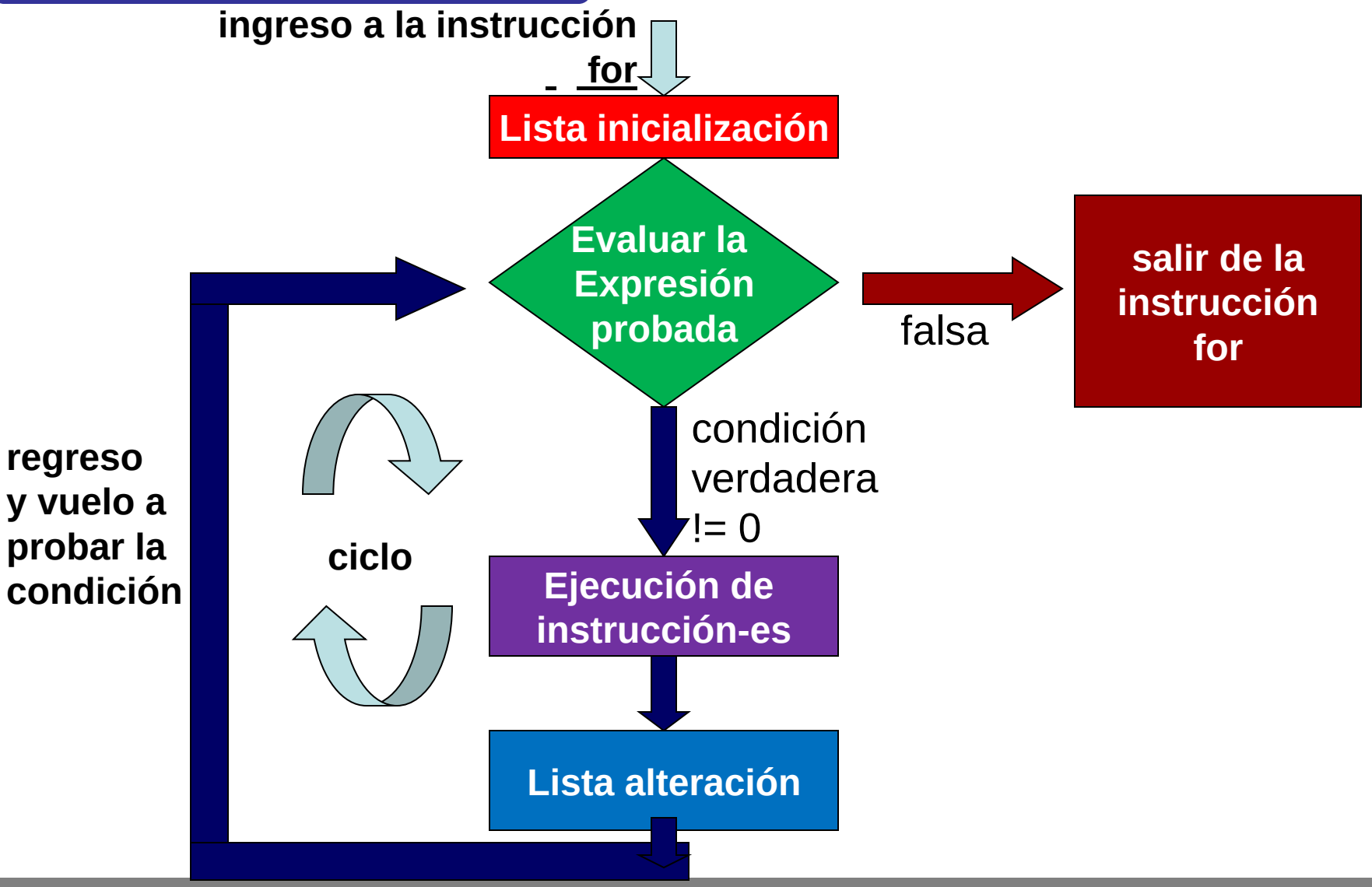
- **Ej. simples de esta instrucción son:**

```
for (cuenta=1   ; cuenta <10   ; cuenta = cuenta +1)
      cout<<cuenta;
```

```
for (i=1; i <=15; i = i +2)
      cout<<i;
```

```
for (lista de inicialización; expresión; lista de alteración)
    instrucción-es;
```

Diagrama de Flujo - FOR



Ciclos FOR

```
for (cuenta=1; cuenta <10; cuenta = cuenta +1)
cout<<cuenta;
```

- En esta instrucción, la **variable contadora debe haber estado DECLARADA anteriormente**, y se llama cuenta. La misma es inicializada en valor 1, y se repite el ciclo incrementándose en 1 siempre y cuando sea menor que 10.

```
for (i=5; i <=15; i = i +2)
cout<<i;
```

- **En esta instrucción, la variable contadora debe haber estado DECLARADA anteriormente, y se llama i. La misma es inicializada en valor 5, y se repite el ciclo incrementándose en 2 siempre y cuando su valor sea menor o igual a 15.**

Ciclos FOR

- **Ciclo WHILE equivalente**

```
cuenta=1;
while(cuenta<=MAXCUENTA)
{cout<<setw(4)<<cuenta
      <<setw(15)<<sqrt(double(cuenta)<<endl;
cuenta++;
}
```

```
for( [expre1]; [expre2]; [expre3] )
        sentencia;

for( [expre1]; [expre2]; [expre3] )
{
      sentencia 1;
       .....
      sentencia n;
}
```

Obsérvese que esto equivale a:

```
      expre1;
      while (expre2)
      {
        sentencia;
        expre3;
      }
```

Ciclos FOR ANIDADOS

- Realice un programa que cumpla los siguientes requisitos:
  - Suponga que un alumno rinde 6 Evaluaciones Conceptuales.
  - Ud. Debe ingresar las 6 notas del alumno por teclado.
  - El programa debe imprimir en pantalla la nota promedio del alumno en las EC.
  - Debe implementarlo CON UN BUCLE **FOR.**
  - **No es de interés en esta etapa validar las notas ingresadas.**

Ciclos FOR ANIDADOS

- Ahora necesitamos optimizar su programa para un curso con 2 alumnos:
  - **Cada uno rinde las 6 evaluaciones conceptuales.**
  - **Debe solicitarse el ingreso de las notas de cada alumno por teclado.**
  - **Debe imprimirse en pantalla las notas promedio de cada alumno.**
  - **IMPLEMENTARLO en Dev C++ para ver que debo agregar al programa anterior para lograr esto…**
  - **Y si los alumnos ahora fueran 5?** **Y si fuera la comisión 5.2?**

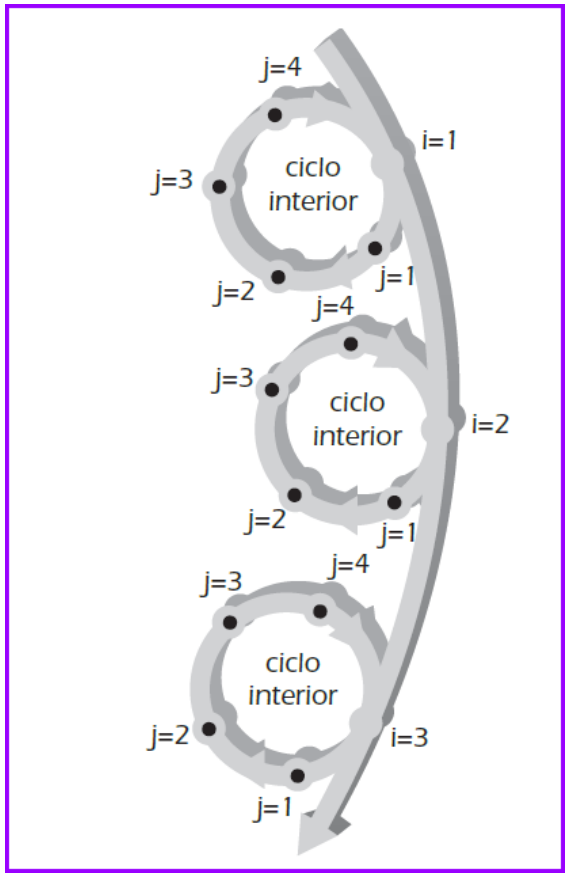
Ciclos FOR ANIDADOS

- En muchas situaciones es necesario y conveniente utilizar un ciclo contenido dentro de otro ciclo. Esto es posible y los ciclos resultantes se conocen como **ciclos anidados.**

- El primer ciclo se llama **CICLO EXTERIOR**, el segundo es llamado **CICLO INTERIOR**. Generalmente todas las instrucciones del CICLO INTERIOR están **contenidas en el cuerpo del ciclo EXTERIOR**.

Ciclos FOR ANIDADOS

- En CADA **ITERACION DEL CICLO EXTERIOR se PRODUCE todas las iteraciones** DEL **CICLO INTERIOR**.



- **Se usan DIFERENTES VARIABLES PARA EL CONTROL DE LAS ITERACIONES DE CADA CICLO.**

Ciclos FOR ANIDADOS

Cual será la salida?

```
#include <iostream>
using namespace std;

int main()
{
  const int MAXI = 5;
  const int MAXJ = 4;
  int i, j;

  for(i = 1; i <= MAXI; i++)
  {
    cout << "\ni es ahora " << i << endl; //
                                          //
    for(j = 1; j <= MAXJ; j++)    // inicio ciclo interior
      cout << "  j = " << j;      // fin ciclo interior
  }                               // fin ciclo exterior   <-------+

  cout << endl;

  system("PAUSE");
  return 0;
}
```

```
i es ahora 1
  j = 1  j = 2  j = 3  j = 4
i es ahora 2
  j = 1  j = 2  j = 3  j = 4
i es ahora 3
  j = 1  j = 2  j = 3  j = 4
i es ahora 4
  j = 1  j = 2  j = 3  j = 4
i es ahora 5
  j = 1  j = 2  j = 3  j = 4
Presione una tecla para continuar . . .
```

Ciclos FOR ANIDADOS

Para observar la utilidad de un ciclo anidado, veremos un ejemplo mas concreto, suponga que se necesita un programa en c++ para calcular la **calificación final para cada estudiante** en una clase de **20** alumnos. Cada alumno ha presentado **cuatro** exámenes durante el semestre. La calificación final es el **promedio** de las 4 calificaciones obtenidas.

El pseudocódigo seria:

Hago un ciclo que ***recorra los 20 alumnos***
    Establezco total de calificaciones *del alumno en 0*
    Hago un ciclo para las ***4 calificaciones de cada alumno***
        introducir calificación
        acumular calificación al total
    Fin del ciclo de calificaciones
    Calculo promedio *e imprimo nota final del alumno*
Fin del ciclo de alumnos.

Ciclos FOR ANIDADOS

```
int main()
{

  const int NUMEROCALIFICACIONES = 4;
  const int NUMEROESTUDIANTES    = 20;
  int i,j;
  double calificacion, total, promedio;

  for (i = 1; i <= NUMEROESTUDIANTES; i++) // inicio ciclo exterior
  {
    total = 0;                          // limpia el total para este estudiante
    for (j = 1; j <= NUMEROCALIFICACIONES; j++) // inicio ciclo interior
    {
      cout << "Ingese la calificacion del examen ara este estudiante: ";
      cin >> calificacion;
      total = total + calificacion;  // agregar la calificacion al total
    }                                // fin del ciclo interior
    promedio = total / NUMEROCALIFICACIONES;    // calcula el promedio
    cout << "\nEl promedio para el estudiante " << i
         << " es " << promedio << "\n\n";
  }                                  // fin del ciclo exterior

  system("PAUSE");
```

Es importante notar el lugar donde se coloca total=0 y donde se calcula el promedio.

Ciclos FOR ANIDADOS

La salida generada será:

```
Ingese la calificacion del examen ara este estudiante: 45
Ingese la calificacion del examen ara este estudiante: 68
Ingese la calificacion del examen ara este estudiante: 58
Ingese la calificacion del examen ara este estudiante: 98

El promedio para el estudiante 1 es 67.25

Ingese la calificacion del examen ara este estudiante: 45
Ingese la calificacion del examen ara este estudiante: 35
Ingese la calificacion del examen ara este estudiante: 46
Ingese la calificacion del examen ara este estudiante: 98

El promedio para el estudiante 2 es 56

Ingese la calificacion del examen ara este estudiante: 25
Ingese la calificacion del examen ara este estudiante: 78
Ingese la calificacion del examen ara este estudiante: 98
Ingese la calificacion del examen ara este estudiante: 65

El promedio para el estudiante 3 es 66.5

Ingese la calificacion del examen ara este estudiante: _
```

```
examen ara este estudiante: 58
examen ara este estudiante: 46
examen ara este estudiante: 46
examen ara este estudiante: 76

nte 18 es 56.5

examen ara este estudiante: 98
examen ara este estudiante: 48
examen ara este estudiante: 54
examen ara este estudiante: 65

El promedio para el estudiante 19 es 66.25

Ingese la calificacion del examen ara este estudiante: 54
Ingese la calificacion del examen ara este estudiante: 57
Ingese la calificacion del examen ara este estudiante: 98
Ingese la calificacion del examen ara este estudiante: 64

El promedio para el estudiante 20 es 68.25

Presione una tecla para continuar . . . _
```

- Puede implementar este programa con bucles **while** anidados?...

ERRORES COMUNES DE PROGRAMACION

Algunos de los errores mas comunes de programación con ciclos se listan a continuación:

* Uno de los mas conocidos es la **“falla por uno”;** tener bien en cuenta los limites de ejecución de los ciclos por ej:

La instrucción for(i=1;i<11;i++)

se ejecuta 10 veces y no 11, observa el porque?.

Y esta instrucción seria equivalente a for(i=1; i<=10; i++)?

Además usando una instrucción similar for(i=0;i<11; i++) esta si se ejecutara 11 veces. Por lo tanto **MAXIMA ATENCION en los LIMITES DE CONTROL DEL CICLO.**

* Otro error común, es el uso del = y no del operador de igualdad “==“ dentro de la expresión a probar ej WHILE. **Resultado BUCLE INFINITO**

ERRORES COMUNES DE PROGRAMACION

* Es bastante usual equivocarse y colocar un “;” al final de la línea de instrucción FOR, esto lleva a que el **ciclo NO HAGA NADA**, ya que considera que en el cuerpo de la instrucción FOR hay una **instrucción NULA,** por lo tanto solo itera incrementando la variable y haciendo **nada** en cada iteración.

* Otro error muy frecuente es utilizar dentro de la expresión de la instrucción FOR, **en vez de “;” utilizar “,”** esto produce **un error de COMPILACION**.

* Finalmente un error común, es acostumbrarse al uso de la instrucción **WHILE, la cual no lleva “;”** y luego al utilizar la instrucción **DO WHILE, no colocar el ; del final de instrucción**.