Autor: Simón, Angel
Carrera: Técnico superior en programación
Materia: Laboratorio de Computación I
Tema: Ciclo exacto **Tema nº:** 04

Ciclo exacto

Sentencia for

La sentencia for permite ejecutar la instrucción, o un conjunto de ellas, un cierto número de veces deseado.

La forma general de la sintaxis es la siguiente:

```
for(variable = valor; condicion; progresión-condición){
        instrucción1;
        instrucción2;
              .
              .
        instrucciónN;
}
```

La primera parte de la construcción for suele ser una asignación donde se inicializa alguna variable que controla el número de veces que debe ejecutarse el cuerpo del bucle. Esta sentencia se ejecuta en una sola ocasión, antes de entrar por primera vez al cuerpo del bucle.
La segunda parte corresponde a la condición que indica cuando finaliza el bucle, es responsabilidad del programador la de poner una condición que en algún momento deje de ser verdadera para que no se produzca un bucle infinito.
La tercera parte corresponde generalmente a una sentencia de incremento o decremento sobre la variable de control del bucle. Esta sentencia se ejecuta siempre después de la de la ejecución del cuerpo del bucle.

Ejemplo:
A partir de dos variables, una llamada base y otra llamada exponente. Se desea obtener la potencia $base^{exponente}$.

Resolución:

```
/*
  Archivo fuente: T04CF01.cpp
*/
#include <iostream>
using namespace std;

int main(void){
   int base, exponente, i;
   cout << "base: ";
   cin >> base;
   cout << endl << "exponente: ";
   cin >> exponente;
   int resultado = 1;
   for(i=1; i<=exponente; i++){
      resultado = resultado * base;
   }
   cout << endl << endl << base << " elevado a la " << exponente << " es: " << resultado;
   return 0;
}
```

T04CF01.cpp

Ejercicio alternativo:
A partir del código fuente anterior, mejorarlo para poder resolver potencias con exponentes negativos.
No se permite utilizar la función pow de la librería math.h.

El operador coma en el ciclo for

C permite la utilización de de más de una sentencia en la primera y tercera partes de la construcción for, así como más de una condición en la segunda parte. Por ejemplo, el siguiente bucle es válido:

```
#include<iostream>
using namespace std;
int main(void){
   int i,j;
   for(i = 0, j = 10; i < 10, j > 0; i++, j-=2){
     cout << "valor de i: " << i << endl;
     cout << "valor de j: "<< j << endl;
   }
}
```

Salida:

```
valor de i: 0
valor de j: 10
valor de i: 1
valor de j: 8
valor de i: 2
valor de j: 6
valor de i: 3
valor de j: 4
valor de i: 4
valor de j: 2
```

Así pues, las variables i y j se inicializan a 0 y 10, respectivamente, antes de comenzar la ejecución del bucle. En la segunda parte de la construcción, aparecen dos condiciones, i < 10 y j > 0. Si alguna de ellas es falsa, la ejecución del bucle se detiene. Finalmente, tras ejecutarse el cuerpo del bucle, i se incrementa en 1 y j se decrementa en 2, tras lo cual vuelven a comprobarse las condiciones, y así sucesivamente.