PRÁCTICA 4: ESTRUCTURAS DE CONTROL REPETITIVAS

Objetivo de la práctica:
- Aprender el uso de estructuras iterativas.

CONCEPTOS BÁSICOS

- Las estructuras de control repetitivas rompen la secuencialidad en la ejecución de los programas, generando repeticiones de sentencias o bucles.
- Las tres estructuras DO-WHILE, WHILE y FOR son equivalentes e intercambiables.
- Las llaves {} sólo son necesarias en el caso de que el bloque esté formado por más de una sentencia.

A continuación se presenta un cuadro resumen:

Estructura control repetitiva	**Sintaxis**	**Ejecución**	**Orientación sobre su uso**
do-while	do { Bloque de sentencias; } while (condición);	(1) Se ejecuta el bloque de sentencias. (2) Se evalúa la expresión condicional: • Si es falsa, sale del bucle. • Si es verdadera, vuelve a (1).	Cuando lo que se conoce es la condición que ha de cumplirse para que el bloque de sentencias se repita. Sí aseguramos que el bloque de sentencias se ejecutará al menos 1 vez [ejecución 1 ó más veces].

while	while (condición)
	{
	Bloque de sentencias;
	}
		(1) Se evalúa la expresión condicional:
		• Si es falsa, sale del bucle.
		• Si es verdadera, continúa en (2).
		(2) Se ejecuta el bloque de sentencias.
		(3) Vuelve a (1).
			No aseguramos que el bloque de sentencias se ejecute alguna vez [ejecución 0 ó más veces]

for	for (inicio_contador/es; condición; actualiz_contador/es)
	{
	Bloque de sentencias;
	}
		(1) Se inicia/n contador/es.
		(2) Se evalúa la expresión condicional:
		• Si es falsa, sale del bucle.
		• Si es verdadera, continúa en (3).
		(3) Se ejecuta el bloque de sentencias.
		(4) Se actualiza/n contador/es.
			Cuando lo que se conoce es el número de veces que el bloque de sentencias debe repetirse.
			Este número de veces se controla mediante el incremento del contador.

OTRAS CARACTERËSTICAS DEL BUCLE FOR:

- El bucle for se ejecuta siempre que la condición sea verdadera.
- Puede tener una o varias expresiones de inicialización y varias expresiones de incremento. En este caso, estas expresiones se separan mediante comas: for(x=1, y=1; x<=100; x++, y++).
- No puede haber más de una expresión de condición.
- Puede tener alguna o todas las expresiones vacías: for (; ;)
- Si la condición está vacía, tenemos un bucle infinito. Sólo puede terminar con la ejecución de una sentencia de salto dentro del bucle.

BLOQUE DE EJERCICIOS

Ejercicio1: el siguiente programa pretende calcular la potencia y^x

```
#include <iostream>
using namespace std;
int main()
{
    int res, x, y, i;
    cout << "dame y" << endl;
    cin >> y;
    cout << "dame x" << endl;
    cin >> x;
    if (x == 0)
        cout << "La potencia es 1" << endl;
    else
    {
        res = y;
        for (i = 0; i <= x; i++)
            res = res * y;
        cout<<"La potencia es  " << res;
    }
  system("pause");
  return 0;
}
```

Escribe el anterior programa en el ordenador. ¿Funciona? ¿Por qué? ¿Cómo lo arreglarías?

Ejercicio 2: Escribe un programa que le pida al usuario dos enteros, de manera que el primero sea menor que el segundo, sino se cumple esta condición los volveremos a pedir hasta que se cumpla. Una vez introducidos correctamente mostraremos la suma de todos los enteros comprendidos entre ambos números incluidos ellos. Por ejemplo para los números 3 y 7, la suma seria 25.

Ejercicio 3: Escribe un programa que le pida al usuario un numero entre el 1 y el 9 - pediremos al usuario dicho número hasta que cumpla la condición- una vez introducido correctamente el programa debe escribir la tabla de multiplicar de ese número usando un bucle for, después de escribir la tabla le preguntaremos ¿quieres introducir otro número? S/N si pulsa S, volveremos a pedirle otro número si pulsa N saldrá un mensaje dándole las gracias por usar nuestro programa y finalizara la ejecución, las tablas de los números que introduzca tendrán el siguiente formato de salida:
7X1=7
7X2=14
.....
7X9=63

Ejercicio 4: Escribe un programa que lea números enteros positivos hasta que se introduzca un 0. El programa deberá mostrar por pantalla la cantidad de números leídos, el mayor, el menor y la media de los números leídos.

Ejercicio 5: La conjetura de Ulam afirma que dado un entero y siguiendo los pasos siguientes siempre obtenemos un 1.

- Si el número es par se divide por 2.
- Si es impar se multiplica por 3 y se suma 1.

Escribe un programa que le pida al usuario un número entero y que compruebe si la conjetura de Ulam es cierta, el programa deberá escribir toda la secuencia hasta llegar al uno. Por ejemplo si el usuario introduce un 5 la secuencia sería: 5, 16, 8, 4, 2, 1.

Ejercicio 6: Escribe un programa que calcule el factorial de un número introducido por teclado. Para todo número natural **n**, se llama **n factorial** o **factorial de n** al producto de todos los enteros entre **1** y **n**:

$$n! = 1 \times 2 \times 3 \times 4 \times ... \times (n-1) \times n$$

Ejercicio 7: Escribe un programa que solicite al usuario un valor no negativo n (si introduce un valor negativo le volveremos a pedir el numero hasta que introduzca uno positivo), y visualice la siguiente salida para (n=6).

123456
12345
1234
123
12
1

Ejercicio 8: Dos números a y b se dice que son amigos si la suma de los divisores de a (salvo él mismo) coincide con b y viceversa. Diseña un programa que tenga como entrada dos números naturales y que indique mediante un mensaje si son amigos o no.

OPCIONALES

Opcional1: El valor de e^x se puede aproximar por la suma $e^x = \sum_{i=0}^{n} \frac{x^i}{i!}$

Escribe un programa que le pida al usuario el valor de x y el valor de n y muestre por pantalla el valor de la aproximación de e^x para el x y el n introducidos.

Opcional2: Escribe un programa que muestre por pantalla todos los números de tres cifras tales que la suma de los cuadrados de sus dígitos es igual al cociente de la división entera del número entre 3.