Objetivo de la práctica:
- Realizar programas con estructuras iterativas.

Conceptos básicos: ESTRUCTURAS ITERATIVAS

Sentencia while

```
while (expresión lógica)
{
    sentencia1;
    sentencia2;
}



Ejemplo:

cin >> n;
num = 1;
while (num <= n)
{
    cout << num << endl;
    num++;
}
```

Sentencia do..while

```
do
{
    sentencia1;
    sentencia2;
}
while (expresión lógica);

Ejemplo:

cin >> n;
num = 1;
do
{
    cout << num << endl;
    num++;
}
while (num <= n)
```

Sentencia for

```
for(inic; cond; increm)
{
    sentencia1;
    sentencia2;
}


Ejemplo:

cin >> num;
for(n = 1; n <= num; n ++)
    cout << n << endl;
```

COMENTARIOS DE INTERÉS

1. Las sentencias **while**() y **for**() no finalizan con '**;**', mientras que la sentencia **do..while**(); sí.

2. En el cuerpo de un bucle **for**, no debemos variar el valor de las variables que forman parte de la condición de fin, puesto que pueden producirse efectos inesperados.

3. ¿Cuándo utilizar un bucle u otro?
- La sentencia **while** suele utilizarse cuando no se conoce el número de iteraciones del bucle, pudiendo ser éste mayor o igual a 0.
- La sentencia **do..while** suele utilizarse cuando no se conoce el número de iteraciones del bucle, pudiendo ser éste mayor o igual a 1.
- La sentencia **for** suele utilizarse cuando se conoce exactamente el número de iteraciones del bucle.

4. Técnicas de control de los bucles:
- **Bucles controlados por indicadores (banderas):**
  Se utiliza una variable "bandera" de tipo bool, de cuyo valor depende la terminación del bucle.

```
bool continuar;

continuar = true;  // Inicializacion del indicador
while (continuar)
{
    ...
    if (condición para acabar)
```

(*) Práctica obligatoria.
(**) Práctica obligatoria en la que hay que entregar el diagrama de flujo completo.

```
            continuar = false;
        ...
}
```

- **Bucles controlados por centinela:**
  En un proceso de introducción de datos, el centinela es el valor cuya lectura como dato, indica la finalización del proceso.

```
suma = 0;
cout << "Introduce números a sumar, 0 para acabar";
cin >> num;
while (num != 0)
{
    suma = suma + num;
    cout << "Introduce números a sumar, 0 para acabar";
    cin >> num;
}
cout << suma;
```

PROBLEMAS

(*) 1. Escribir un programa para mostrar la tabla de multiplicar del 7 utilizando un bucle for, la del 11 con un bucle while y la del 13 con un bucle do-while. Como segunda parte, hacer que la salida de las tablas por pantalla sea la siguiente:

7x1=7	11x1=11	13x1=13
7x2=14	11x2=22	13x2=26
...		
7x10=70	11x10=110	13x10=130

(**) 2. Escribir un programa que genere y muestre x ($1 < x < 6$) números aleatorios no repetidos enteros entre 0 y 9 y calcule cúal es el máximo, cual es el mínimo y cual es la media de los números. El valor de x se introducirá por teclado al iniciar la ejecución del programa.

(*) 3. Escribir un programa que lea un número entero por teclado, equivalente a una cantidad de dinero en céntimos de euro, y determine la cantidad mínima y la cuantía de las monedas de curso legal (2 y 1 euro, 50, 20, 10, 5, 2 y 1 céntimos) que se necesitan para obtener la cantidad introducida. Solo se pueden utilizar sumas y restas en el programa para realizar los cálculos.

(**) 4. Escribir un programa para escribir las tablas de multiplicar (aprovechar el ejercicio 1). El programa mostrará un menu para indicar si se desea ver la tabla de un solo número, todas las tablas del 1 al 10, o terminar la ejecución del programa. Según la opción que introduzca el usuario, se deberá realizar la tarea seleccionada.

5. Escribir un programa para jugar al master mind, de forma simplificada. El jugador 1 introduce una combinación de cuatro números del 1 al 4, sin posibilidad de repetirlos. Posteriormente se borra la pantalla, y el jugador 2 debe introducir combinaciones hasta acertar la combinación original introducida por el jugador 1. Tras la introducción de cada combinación por el jugador 2, se debe mostrar el número de aciertos obtenidos (es decir, cuantas posiciones han sido acertadas). Al

(*) Práctica obligatoria.
(**) Práctica obligatoria en la que hay que entregar el diagrama de flujo completo.

terminar el programa (el jugador 2 ha acertado la combinación, o sea, hay 4 aciertos), se debe mostrar como resultado el número de intentos empleados en resolverlo.

6. Utilizando la serie: $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ... = 2{,}71828^x ...$, hacer un programa para calcular el número e hasta una precisión en el valor de i introducida al inicio del programa.

(*) Práctica obligatoria.
(**) Práctica obligatoria en la que hay que entregar el diagrama de flujo completo.

Nota sobre el uso de funciones aleatorias:

La generación de un número entero aleatorio se consigue con la función srand(); el número generado está entre 0 y MAX_RAND.

La generación de números de forma aleatoria es en realidad pseudo-aleatoria. Se puede conseguir una mejora en la generación de números aleatorios a través de la funcion srand(<semilla>). La semilla se puede conseguir de forma más o menos aleatoria invocando a la función time() con parámetro NULL (es decir time(NULL)), y es necesario incluir en este caso la cabecera time.h.

(*) Práctica obligatoria.
(**) Práctica obligatoria en la que hay que entregar el diagrama de flujo completo.