Laboratorio 02

Bucles

1. Competencias

1.1. Competencias del curso

Conoce, comprende e implementa programas usando las estructuras básicas del lenguaje de programación C++.

1.2. Competencia del laboratorio

Conoce, comprende e implementa programas usando las estructuras básicas del lenguaje de programación C++.

2. Equipos y Materiales

- Un computador.
- IDE para C++.
- Compilador para C++.

3. Marco Teórico

3.1. Bucles

Un ciclo o bucle permite repetir una o varias instrucciones cuantas veces lo necesitemos, por ejemplo, si quisiéramos escribir los números del uno al cien no tendría sentido escribir cien líneas mostrando un numero en cada una, para esto y para muchísimas cosas más, es útil un ciclo, permitiéndonos hacer una misma tarea en una cantidad de líneas muy pequeña y de forma prácticamente automática.

Existen diferentes tipos de ciclos o bucles, cada uno tiene una utilidad para casos específicos y depende de nuestra habilidad y conocimientos poder determinar en qué momento es bueno usar alguno de ellos. Tenemos entonces a nuestra disposición los siguientes tipos de ciclos en C++:

- Bucle for
- Bucle while
- Bucle do-while
- Bucle anidado

Bucle o ciclo for

Los ciclos for son lo que se conoce como estructuras de control de flujo cíclicas o simplemente estructuras cíclicas, estos ciclos, como su nombre lo sugiere, nos permiten ejecutar una o varias líneas de código de forma iterativa, conociendo un valor especifico inicial y otro valor final, además nos permiten determinar el tamaño del paso entre cada "giro" o iteración del ciclo.

En resumen, un ciclo for es una estructura de control iterativa, que nos permite ejecutar de manera repetitiva un bloque de instrucciones, conociendo previamente un valor de inicio, un tamaño de paso y un valor final para el ciclo.

La sintaxis de un ciclo for es simple en C++, en realidad en la mayoría de los lenguajes de alto nivel es incluso muy similar, de hecho, con tan solo tener bien claros los 3 componentes del ciclo for (inicio, final y tamaño de paso) tenemos prácticamente todo hecho

```
for(int i = valor inicial; i <= valor final; i = i + paso)
{
   Bloque de Instrucciones....
}
```

Bucle o ciclo while

Los ciclos while son también una estructura cíclica, que nos permite ejecutar una o varias líneas de código de manera repetitiva sin necesidad de tener un valor inicial e incluso a veces sin siquiera conocer cuando se va a dar el valor final que esperamos, los ciclos while, no dependen directamente de valores numéricos, sino de valores booleanos, es decir su ejecución depende del valor de verdad de una condición dada, verdadera o falso, nada más. De este modo los ciclos while, son mucho más efectivos para condiciones indeterminadas, que no conocemos cuando se van a dar a diferencia de los ciclos for, con los cuales se debe tener claro un principio, un final y un tamaño de paso.

La sintaxis de un ciclo while es incluso más simple y "legible" que la del ciclo for en C++, pues simplemente requerimos tener clara una condición de parada.

```
while(condición de finalización)
{
   Bloque de Instrucciones....
}
```

Bucle o ciclo do-while

Los ciclos do-while son una estructura de control cíclica, los cuales nos permiten ejecutar una o varias líneas de código de forma repetitiva sin necesidad de tener un valor inicial e incluso a veces sin siquiera conocer cuando se va a dar el valor final, hasta aquí son similares a los ciclos while, sin embargo el ciclo do-while nos permite añadir cierta ventaja adicional y esta consiste que nos da la posibilidad de ejecutar primero el bloque de instrucciones antes de evaluar la condición necesaria, de este modo los ciclos do-while, son más efectivos para algunas situaciones específicas.

La sintaxis de un ciclo do-while es un tanto más larga que la del ciclo while en C++, sin embargo, no se hace más complicado, de hecho, con tan solo tener bien clara una condición de finalización para el ciclo tendremos prácticamente todo terminado.

```
do
{
   Bloque de Instrucciones....
}
while(condición de finalización);
```

Bucle o ciclo anidado

Los ciclos anidados no son una estructura de control por sí mismos, son un indicativo de que quizá deberíamos plantear la solución a algún problema si nos vemos obligados a usar ciclos anidados y más aún si es más de uno, sin embargo, debemos saber que a veces son indispensables.

La sintaxis es sencilla, un ciclo con otro adentro, y lo que nos haga falta, pues podemos poner varias sentencias adicionales al interior de cualquiera de los dos ciclos.

```
for(int i = valor inicial; i < valor final; i++)
{
      Bloque de Instrucciones....
   for(int j = valor inicial; j < valor final; j++)
   {
```

```
        Bloque interno de Instrucciones....
    }
    //Acá puede haber más instrucciones
}
```

4. Ejercicios

Resolver los siguientes ejercicios planteados:

1. Sumar todos los enteros pares desde 2 hasta 100.
2. Calcule los primeros 50 números primos y muestre el resultado en pantalla.
3. Escribir un programa que visualice en pantalla los números múltiplos de 5 comprendidos entre 1 y 100.
4. Escriba un código que solicite ingresar dos números x y y, tal que $x < y$. Muestre todos los números primos que se encuentren entre el rango de valores, de no encontrarse, mostrar el primo más cercano a x o y.
5. Elabore un programa que lea n números y determine cuál es el mayor, el menor y la media de los números leídos.
6. Elabore un programa que calcule la serie de Fibonacci. La serie de Fibonacci es la sucesión de números: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Cada número se calcula sumando los dos anteriores a él.
7. Calcula el promedio de 3 notas para n estudiantes.
8. Escribir un programa que genere la tabla de multiplicar de un número introducido por el teclado.
9. Escribir un programa que pida al usuario un número entero y muestre por pantalla un triángulo rectángulo como el de más abajo, de altura el número introducido.

   ```
   *
   **
   ***
   ****
   *****
   ```
10. Escribir un programa que pida al usuario una palabra y luego muestre por pantalla una a una las letras de la palabra introducida empezando por la última.

5. Entregables

Al final estudiante deberá:

1. Compactar el código elaborado y subirlo al aula virtual de trabajo. Agregue sus datos personales como comentario en cada archivo de código elaborado.
2. Elaborar un documento que incluya tanto el código como capturas de pantalla de la ejecución del programa. Este documento debe de estar en formato PDF.
3. El nombre del archivo (comprimido como el documento PDF), será su LAB02_GRUPO_A/B/C_CUI_1erNOMBRE_1erAPELLIDO.

   (Ejemplo: LAB02_GRUPO_A _2022123_PEDRO_VASQUEZ).
4. Debe remitir el documento ejecutable con el siguiente formato:

   LAB02_GRUPO_A/B/C_CUI_ EJECUTABLE_1erNOMBRE_1erAPELLIDO

   (Ejemplo: LAB02_GRUPO_A_EJECUTABLE_2022123_PEDRO_VASQUEZ).

En caso de encontrarse trabajos similares, los alumnos involucrados no tendrán evaluación y serán sujetos a sanción.