

ESTRUCTURA DE LENGUAJE DE PROGRAMACIÓN C++

Unidad 4

Estructuras de repetición en el lenguaje C++

Tabla de Contenido

1. Introducción	3
2. Estructura de contenido	4
3. Estructuras de repetición en el lenguaje C++	5
3.1. Estatuto WHILE	5
3.2. Estatuto DO WHILE	8
3.3. Estatuto FOR.....	11
3.4. Ciclos infinitos	14
4. Material de apoyo	16
5. Glosario	17
6. Referencias bibliográficas	18
7. Control del documento	18
Créditos	19
Creative Commons	19

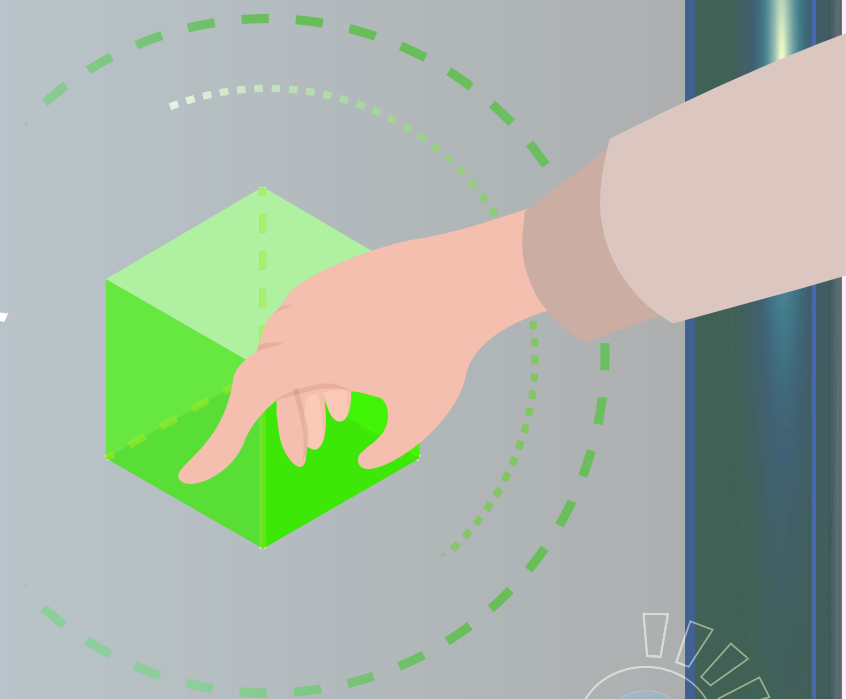


1. Introducción

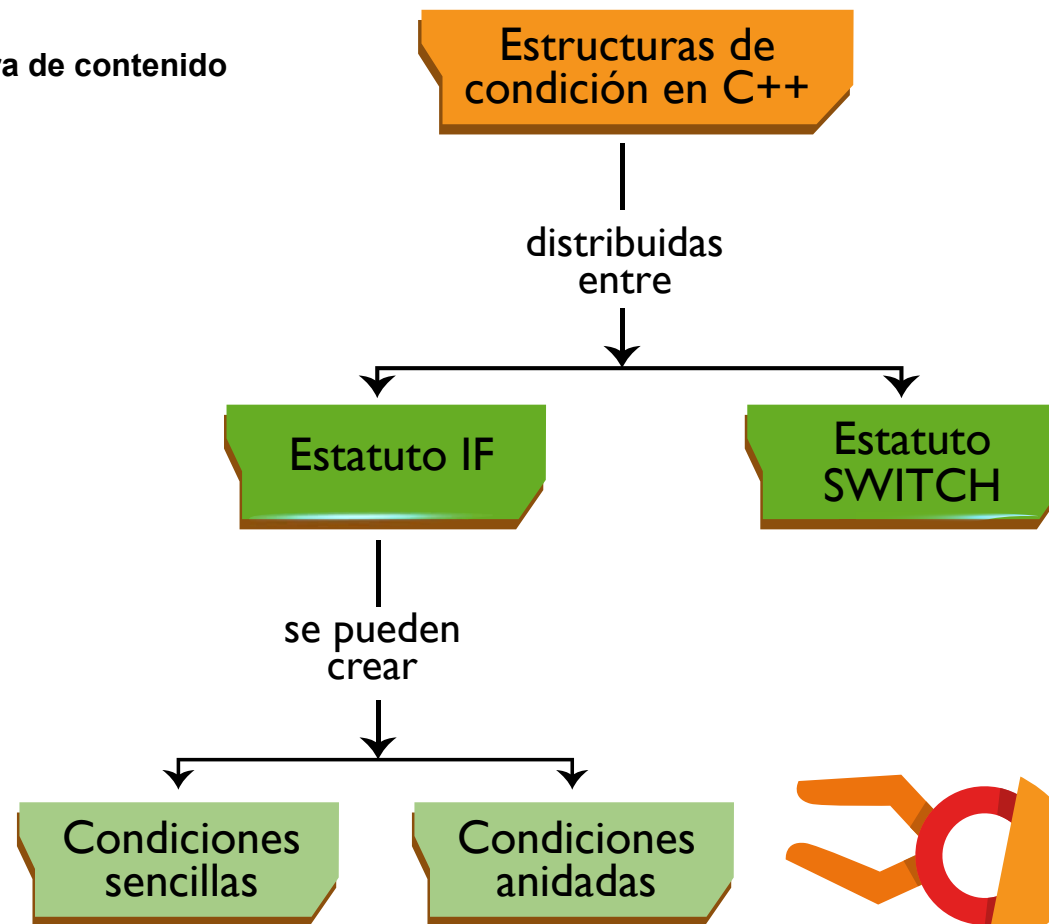
La implementación de las estructuras de repetición en el lenguaje C++ y en cualquier otro lenguaje de programación, conllevan a un dominio integral de una aplicación. Dado que ofrece al programador el control sobre el flujo de datos, tanto de entrada como de salida o simplemente para procesarlos sin que el usuario lo note.

Existen tres estructuras de repetición como lo son: while, do while y for. Cada una requiere de unos elementos y reglas específicas para su funcionamiento. En el desarrollo de soluciones se puede requerir la combinación de éstos y por consiguiente la lógica y el dominio del programador debe aumentar; lo cual solo se logra con la práctica.

En este resultado de aprendizaje, conocerá la sintaxis de cada bucle con ejemplos que le permitirán adquirir la competencia, además, se integrarán todos los temas vistos hasta el momento, lo cual le facilitará poner en práctica los conocimientos adquiridos.



2. Estructura de contenido



3. Estructuras de repetición en el lenguaje C++

3.1. Estatuto WHILE

El While establece la repetición de una serie de instrucciones si se cumple una condición. Por consiguiente, se valida una expresión del tipo booleano en la condición (true o false). Su sintaxis es como se muestra a continuación:

```
while(condición)
{
    sentencia
}
```

Si se quisiera leer la anterior estructura se expresaría como: “mientras la condición sea verdadera realice las siguientes instrucciones”.

Dentro del bloque de instrucciones, irá la variable que modificará dicha condición, pero antes de programar el ciclo, dicha variable debe ser inicializada con un valor que permita que la expresión sea verdadera, para que al menos ingrese una vez. En el momento en que esa variable no cumpla con la condición, el ciclo finaliza. Este bucle se utiliza independientemente si se conoce o no la cantidad de iteraciones que el programa requiere ejecutar.

A continuación, se presentará el primer ejemplo, donde se requiere hallar los múltiplos de un número digitado por el usuario hasta que llegue a 100. En Dev C++ elabore un proyecto nuevo, guárdelo con el nombre de ciclos_1 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int numero, resultado=0;
8
9     cout << "Digita un número para hallar los múltiplos: " << endl;
10    cin>>numero;
11    cout << "Los múltiplos de " << numero << " son: " << endl;
12
13    while(resultado < 100)
14    {
15        resultado = resultado + numero;
16        cout << resultado << endl;
17    }
18
19    system("pause");
20    return EXIT_SUCCESS;
21 }
22 }
```

Cómo se observa en la línea 7, se declaran dos variables del tipo entero, en la primera se guardará el número que digite el usuario y el resultado es el que se va a mostrar en pantalla, el cual se inicializa en cero.

13 while(resultado < 100)

En la línea 13 se declara el ciclo; y la condición establecida para que el programa ingrese a él, es que la variable resultado sea menor que 100.

```
15 resultado = resultado + numero;  
16 cout << resultado << endl;
```

En la línea 15, a la variable resultado se le agrega el valor del número digitado, para ser mostrada en pantalla en la línea 16. De este modo, la variable resultado se ha convertido en un “acumulador” y cuando se vuelvan a repetir estas instrucciones su valor incrementará. Por ejemplo, si se digitará el número 10, la variable resultado obtendría el valor de 10 en la primera iteración.

resultado = 0 + 10;
resultado queda valiendo 10.

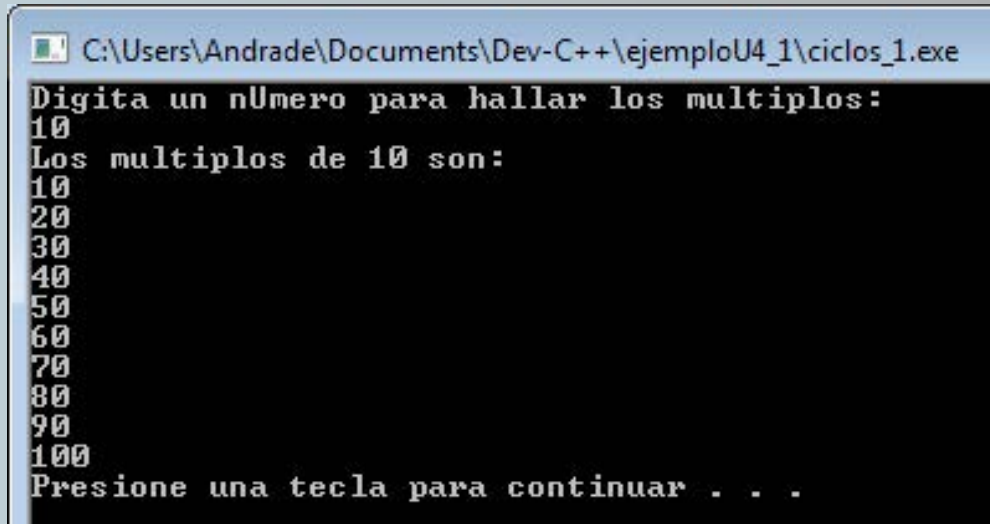
En la segunda iteración:

resultado = 10 + 10;
resultado queda valiendo 20.

En la tercera iteración:

resultado = 20 + 10;
resultado queda valiendo 30

El programa continúa con las iteraciones mientras que el resultado sea menor a 100, cuando resultado es mayor a 100 el programa sale del ciclo. La ejecución del programa es como se muestra a continuación:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU4_1\ciclos_1.exe  
Digita un número para hallar los multiplos:  
10  
Los multiplos de 10 son:  
10  
20  
30  
40  
50  
60  
70  
80  
90  
100  
Presione una tecla para continuar . . .
```

Teniendo en cuenta el siguiente enunciado, se desarrollará un segundo ejemplo.

“Un estudiante recibe cinco calificaciones y se debe determinar el promedio alcanzado por él.”



Elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de ciclos_2 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int contador_notas = 1;
8     float nota, promedio = 0.0, acumulador_notas = 0.0;
9
10    while(contador_notas <= 5){
11
12        cout << "Digita la nota " << contador_notas << ": ";
13        cin>>nota;
14        acumulador_notas = acumulador_notas + nota;
15        contador_notas++;
16
17    }
18
19    promedio = acumulador_notas / 5;
20
21    cout << "El promedio de las notas es: " << promedio << endl;
22
23    system("pause");
24    return EXIT_SUCCESS;
25 }
```

En esta ocasión, se nombrarán algunas variables que son necesarias para alcanzar el objetivo del programa.

```
7     int contador_notas = 1;
8     float nota, promedio = 0.0, acumulador_notas = 0.0;
```



En la línea 7 se declara la variable **contador_notas**, la cual indicará la cantidad de calificaciones digitadas por el usuario y se utilizará para definir la expresión que requiere el ciclo. En la línea 8 se declaran variables del tipo flotante, porque éstas pueden adquirir valores decimales. La variable **nota** guardará temporalmente el valor digitado por el usuario, la variable **promedio** se utilizará al final para hallar el promedio de las calificaciones y la variable **acumulador_notas** irá acumulando nota tras nota.

```
10    while(contador_notas <= 5){
```



En la línea 10 se declara el ciclo. La condición para que se ejecute el bloque de instrucciones es que la variable **contador_notas** sea menor o igual a 5.

```
12     cout << "Digita la nota " << contador_notas << ": ";
13     cin>>nota;
14     acumulador_notas = acumulador_notas + nota;
15     contador_notas++;
```

Después de recibir las notas digitadas por el usuario, en la línea 14, éstas se acumulan en la variable **acumulador_notas** y **contador_notas** incrementando en uno su valor. Recuerde que antes de entrar al ciclo, esta variable se inicializó con el valor de uno.

El programa solicita las notas mientras que **contador_notas** es igual o menor que 5. Cuando la condición no se cumpla el ciclo se termina.

```
19     promedio = acumulador_notas / 5;
```

En la línea 19 se halla el promedio dividiendo el total de las notas por la cantidad de notas digitadas. El resultado en pantalla de la aplicación es el siguiente:

C:\Users\Andrade\Documents\Dev-C++\ejemploU4_2\ciclos_2.exe

```
Digita la nota 1: 4.5
Digita la nota 2: 6.7
Digita la nota 3: 3
Digita la nota 4: 9
Digita la nota 5: 8.5
El promedio de las notas es: 6.34
Presione una tecla para continuar . . . _
```

3.2. Estatuto DO WHILE

El **do while** cumple una función parecida al estatuto while, con la diferencia de que el bloque de instrucciones dentro de él se ejecuta por lo menos una vez, así la variable iteradora no cumpla con la condición; dado que la expresión se evalúa al final.

Este bucle establece la repetición de una serie de instrucciones hasta que la condición sea falsa o no se cumpla. Su estructura se puede expresar como: **“realice las siguientes instrucciones mientras que la condición sea verdadera”**. Su sintaxis es la siguiente:

```
do
{
sentencia
}
while(condición)
```

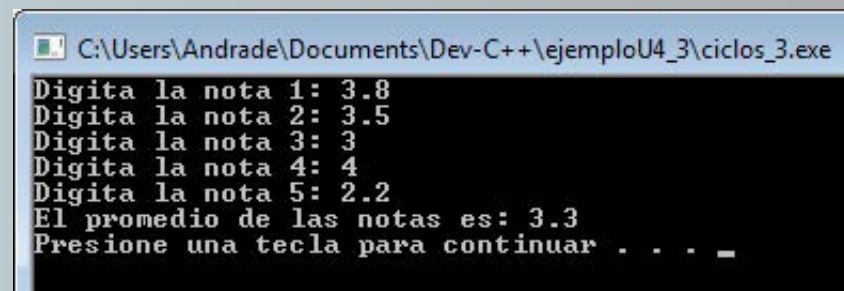


A continuación, elabore un nuevo proyecto en Dev C++, guárdelo como `ciclos_3`, copie y pegue el código del ejercicio `ciclos_2` y realice la siguiente modificación:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int contador_notas = 1;
8     float nota, promedio = 0.0, acumulador_notas = 0.0;
9
10    do{
11        cout << "Digita la nota " << contador_notas << ": ";
12        cin>>nota;
13        acumulador_notas = acumulador_notas + nota;
14        contador_notas++;
15    }while(contador_notas <= 5);
16
17    promedio = acumulador_notas / 5;
18
19    cout << "El promedio de las notas es: " << promedio << endl;
20
21    system("pause");
22    return EXIT_SUCCESS;
23 }
24 }
```

Como se observa el código quedó casi intacto, solamente se modificó la estructura del ciclo dándole apertura en la línea 10 con el **`do{`** y cerrándolo en la línea 16 con la expresión en el **`}while();`**

Al ejecutar el programa el resultado es el mismo:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU4_3\ciclos_3.exe
Digita la nota 1: 3.8
Digita la nota 2: 3.5
Digita la nota 3: 3
Digita la nota 4: 4
Digita la nota 5: 2.2
El promedio de las notas es: 3.3
Presione una tecla para continuar . . . _
```



Como segundo ejemplo del ciclo do while, se formula la siguiente aplicación que tiene como objeto identificar la mayor velocidad de un conjunto de 5 automóviles. Elabore un nuevo proyecto en Dev C++, guárdelo como ciclos_4 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int velocidad, carro = 1, mayor = 0;
8
9     do{
10         cout << "Digita la velocidad del auto nUmero " << carro << ": ";
11         cin>>velocidad;
12
13         if(velocidad > mayor){
14             mayor = velocidad;
15         }
16
17         carro++;
18     }while(carro <= 5);
19
20     cout << "La velocidad mayor es: " << mayor << endl;
21
22     system("pause");
23     return EXIT_SUCCESS;
24 }
25 }
```

Como se observa en la línea 7, se declaran las variables a utilizar y se asignan valores a **carro** y a **mayor**. La primera en **1** porque le indicará al usuario el número del vehículo; y la segunda variable en 0 porque es necesaria una base para comparar las velocidades digitadas.

```
13         if(velocidad > mayor){
14             mayor = velocidad;
15         }
```

Entre las líneas 13 y 15 se identifica la velocidad mayor. Al digitar la primera velocidad, ésta se convierte automáticamente en la mayor. Si después la velocidad digitada es mayor a la anterior, ésta ahora será la mayor y así sucesivamente **mientras que la variable carro sea menor o igual a 5**. El resultado en pantalla de la aplicación es el siguiente:

```
Digita la velocidad del auto nUmero 1: 80
Digita la velocidad del auto nUmero 2: 30
Digita la velocidad del auto nUmero 3: 45
Digita la velocidad del auto nUmero 4: 90
Digita la velocidad del auto nUmero 5: 55
La velocidad mayor es: 90
Presione una tecla para continuar . . .
```



3.3. Estatuto FOR

El ciclo **for** permite repetir un bloque de instrucciones una conocida cantidad de veces. Su estructura requiere de tres elementos: el primero representa la asignación de un valor a la variable iteradora, el segundo elemento es la expresión que evalúa el valor de dicha variable (condición) y el tercer elemento indica el incremento de la misma. Estos elementos deben ir separados con el carácter punto y coma (;). Su sintaxis es la siguiente:

```
for(variable = valor ; condición ; incremento  
variable)  
{  
sentencia  
}
```

Lo anterior quiere decir que el bloque de instrucciones se ejecuta hasta que la variable llegue a un valor asignado en la condición. Su estructura se puede expresar como: ***“para ‘x’ hasta que llegue a ‘y’, ejecute las siguientes instrucciones”***.

Lo anterior se comprende mejor a través de un ejemplo. Así que, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de **ciclos_5** y digite el siguiente código:

```
1 #include <iostream>  
2  
3 using namespace std;  
4  
5 int main()  
6 {  
7     int i;  
8     for(i=0; i <= 10; i++){  
9         cout << i << endl;  
10    }  
11  
12    system("pause");  
13    return EXIT_SUCCESS;  
14 }
```



El objetivo de este programa es mostrar en pantalla los números de cero a 10. En la línea 7 se declara la variable *i* que controlará el ciclo. En el **for** se le asigna el valor inicial de cero y la condición es que la variable *i* sea menor o igual a 10. El tercer elemento del **for** es el incremento de la variable *i*. En la línea 9 se va imprimiendo el valor de la variable *i* y el resultado en pantalla es el siguiente:

```
0
1
2
3
4
5
6
7
8
9
10
Presione una tecla para continuar . . . _
```

Como segundo ejemplo del estatuto **for** se propone desarrollar una aplicación que reciba el género de 5 personas y su edad, con la finalidad de hallar el promedio de edades del género femenino

y el género masculino. Por consiguiente, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de **ciclos_6** y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int edad, hombres = 0, mujeres = 0, edades_gm = 0, edades_gf = 0;
8     char genero;
9     float promedio_gm, promedio_gf;
10
11     int i;
12     for(i=0; i <= 4 ; i++)
13     {
14         cout << "Digite el genero de la persona M/F: ";
15         cin>>genero;
16         cout << "Digite la edad de la persona: ";
17         cin>>edad;
18
19         if(genero == 'M'){
20
21             hombres++;
22             edades_gm = edades_gm + edad;
23
24         }else if(genero == 'F'){
25
26             mujeres++;
27             edades_gf = edades_gf + edad;
28
29         }
30     }
31
32     cout << "Total de Hombres: " << hombres << endl;
33     cout << "Total promedio de edades de Hombres: " << edades_gm/hombres << endl;
34
35     cout << "Total de Mujeres: " << mujeres << endl;
36     cout << "Total promedio de edades de Mujeres: " << edades_gf/mujeres << endl;
37
38     system("pause");
39     return EXIT_SUCCESS;
40 }
```


Como se observa entre las líneas 7 y 9, se declaran las variables a utilizar. Las variables denominadas **hombres** y **mujeres** son contadores del género correspondiente. Las variables **edades_gm** y **edades_gf** son los acumuladores de las edades, necesarias para hallar el promedio (**promedio_gm** y **promedio_gf**).

```
11     int i;  
12     for(i=0; i <= 4 ; i++)  
13     {
```

En la línea 14 se solicita al usuario que digite la letra 'M' o 'F' para indicar masculino o femenino respectivamente.

```
19         if(genero == 'M'){  
20  
21             hombres++;  
22             edades_gm = edades_gm + edad;  
23  
24             }else if(genero == 'F'){  
25  
26                 mujeres++;  
27                 edades_gf = edades_gf + edad;  
28  
29             }
```

Entre las líneas 19 y 29 se identifica el carácter 'M' o 'F' y se asignan los nuevos valores a las diferentes variables en cada repetición.

```
32 cout << "Total de Hombres: " << hombres << endl;  
33 cout << "Total promedio de edades de Hombres: " << edades_gm/hombres << endl;
```

Al finalizar, se halla el promedio dividiendo el total de edades sobre la cantidad de personas del mismo género. El resultado en pantalla del programa es el siguiente:

```
Digite el genero de la persona M/F: M  
Digite la edad de la persona: 35  
Digite el genero de la persona M/F: M  
Digite la edad de la persona: 45  
Digite el genero de la persona M/F: F  
Digite la edad de la persona: 30  
Digite el genero de la persona M/F: F  
Digite la edad de la persona: 20  
Digite el genero de la persona M/F: M  
Digite la edad de la persona: 18  
Total de Hombres: 3  
Total promedio de edades de Hombres: 32  
Total de Mujeres: 2  
Total promedio de edades de Mujeres: 25  
Presione una tecla para continuar . . .
```



3.4. Ciclos Infinitos

Los ciclos infinitos se presentan mayormente cuando se comete un error al definir la condición del bucle, o cuando la variable iteradora nunca obtiene el valor para que dicha condición sea falsa. En la mayoría de los casos, los ciclos infinitos no son deseados por el programador, dado que consumen los recursos de la máquina y tienen un impacto negativo en su rendimiento; también se puede presentar un bloqueo en las aplicaciones.

Por ejemplo, la siguiente aplicación contará de 2 de en 2 de manera infinita:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int i=1;
8     while(i > 0){
9         i = i+2;
10        cout << i << endl;
11    }
12
13    system("pause");
14    return EXIT_SUCCESS;
15 }
```

Como se observa en la línea 7 se declara la variable iteradora en **1** y la condición del while es que sea mayor que **0**. Al ingresar al ciclo **i** aumenta cada vez en **2**, lo cual quiere decir que la condición nunca será falsa y el programa habrá que cerrarlo de forma arbitraria.

A continuación, se muestra el mismo ejemplo pero la condición será controlada por el usuario. Así que elabore un nuevo proyecto en Dev C++, guárdelo como **ciclos_7** y digite el siguiente código:

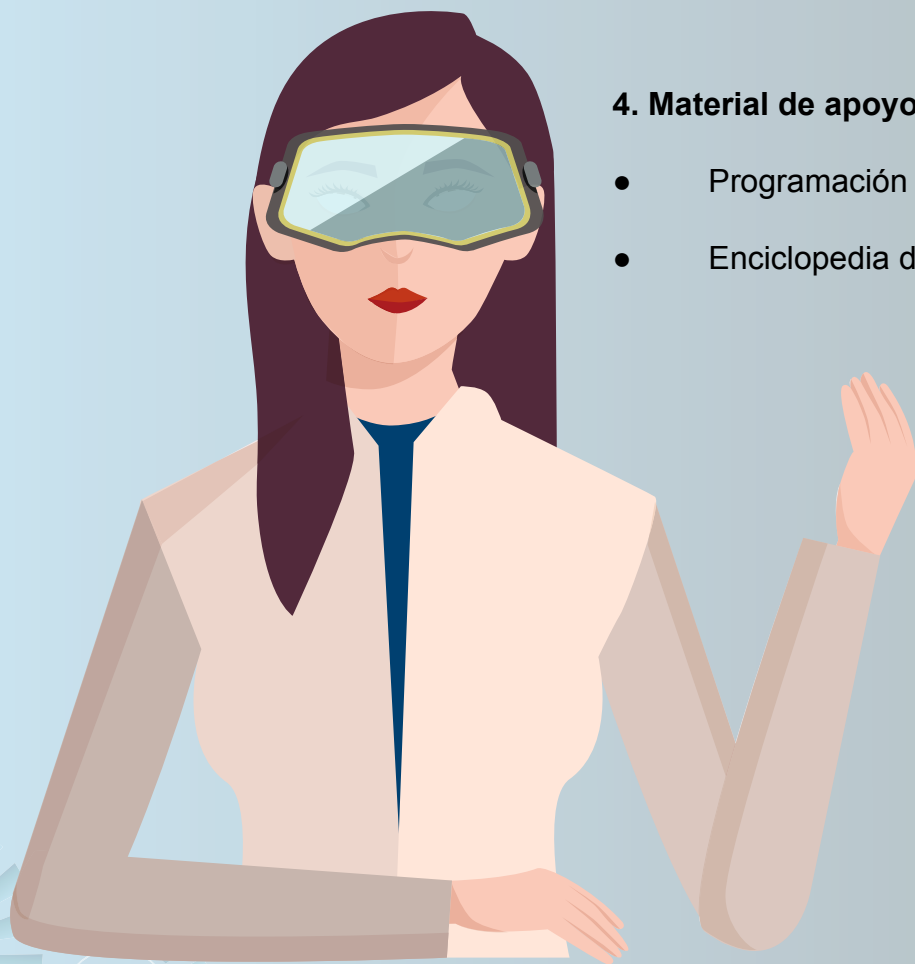
```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int i=0;
8     char continuar;
9
10    cout << "Desea contar nUmeros de dos en dos ? s/n: ";
11    cin>>continuar;
12
13    while(continuar != 'n'){
14        i = i+2;
15        cout << i << endl;
16        cout << "Desea continuar? digite 'n' para salir: ";
17        cin>>continuar;
18    }
19
20    cout << endl;
21    system("pause");
22    return EXIT_SUCCESS;
23 }
```



La variable que controlará el ciclo está definida en la línea 8 (continuar). En la línea 10 se pregunta al usuario si desea contar los números, si digita la letra 's' ingresa al ciclo y muestra el primer número. Antes de mostrar el siguiente número, se pregunta si desea continuar. En caso de que nunca digite la letra 'n' el ciclo será infinito. El resultado en pantalla es el siguiente:

```
Desea contar nUmeros de dos en dos ? s/n: s
2
Desea continuar? digite 'n' para salir: s
4
Desea continuar? digite 'n' para salir: s
6
Desea continuar? digite 'n' para salir: s
8
Desea continuar? digite 'n' para salir: n
Presione una tecla para continuar . . .
```





4. Material de apoyo

- Programación en C++: un enfoque práctico.
- Enciclopedia del lenguaje C++





ABCD

5. Glosario

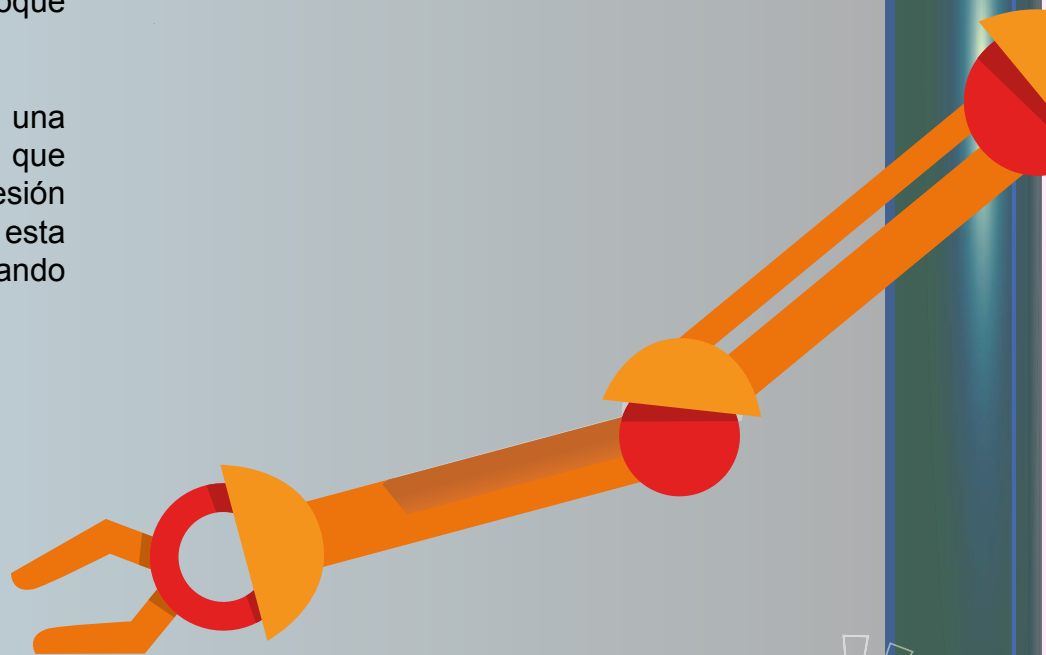
Estatuto: es un sinónimo de instrucción.

Estatuto break: instrucción que termina inmediatamente un ciclo o una instrucción switch.

Estatuto for: es un método para ejecutar un bloque de sentencias un número fijo de repeticiones.

Estatuto while: se usa para implementar una estructura de repetición (bucle while), en la que la repetición se controla mediante una expresión booleana y continúa ejecutándose mientras esta expresión permanece cierta, finalizando cuando se hace falsa.

Estatuto do/while: tiene un comportamiento similar a while, sólo que en este caso; primero se ejecuta el bloque de instrucciones y después se evalúa la condición. Con esto se asegura que el bloque se ejecutará al menos una vez.





6. Referencias bibliográficas

Ceballos, S. F. J. (2009). Enciclopedia del lenguaje C++ (2a. ed.). Madrid, ES: RA-MA Editorial.

Joyanes, L. Sánchez, L. (2006). Programación en C++: un enfoque práctico. España: McGraw-Hill.

Joyanes, L. Zahonero, I. (2007). Estructura de Datos en C++. España: McGraw-Hill.

10. Control del documento

	Nombre	Cargo	Versión	Fecha
Autor	Jorge Eliecer Andrade Cruz	Experto temático	1	Junio de 2017





Créditos

Equipo de Adecuación Gráfica
Centro de Comercio y servicios
SENA Regional Tolima
Línea de Producción

Director Regional

Félix Ramón Triana Gaitán

Subdirector de Centro

Álvaro Fredy Bermúdez Salazar

Coordinadora de formación profesional

Gloria Ines Urueña Montes

Experto temático

Jorge Eliecer Andrade Cruz

Senior equipo de adecuación

Claudia Rocio Varón Buitrago

Asesor pedagógico

Ricardo Palacio

Guionistas

Genny Carolina Mora Rojas
Jesús Bernardo Novoa Ortiz

Diseño y diagramación

Diana Katherine Osorio Useche
Pedro Nel Cabrera Vanegas
Ismael Enrique Cocomá Aldana

Programadores

Davison Gaitán Escobar
Héctor Horacio Morales García
Iván Darío Rivera Guzmán



Creatives commons

Atribución, no comercial, compartir igual.

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo comercial.

