

# ESTRUCTURA DE LENGUAJE DE PROGRAMACIÓN C++

## Unidad 3

### Estructuras de condición en el lenguaje C++

## Tabla de Contenido

1. Introducción .....	3
2. Estructura de contenido .....	4
3. Estructuras de condición en el lenguaje C++ .....	5
3.1. Condiciones sencillas .....	5
3.2. Condiciones anidadas .....	7
3.3. Estatuto SWITCH .....	9
4. Material de apoyo .....	13
5. Glosario .....	14
6. Referencias bibliográficas .....	15
7. Control del documento .....	15
Créditos .....	16
Creative Commons .....	16



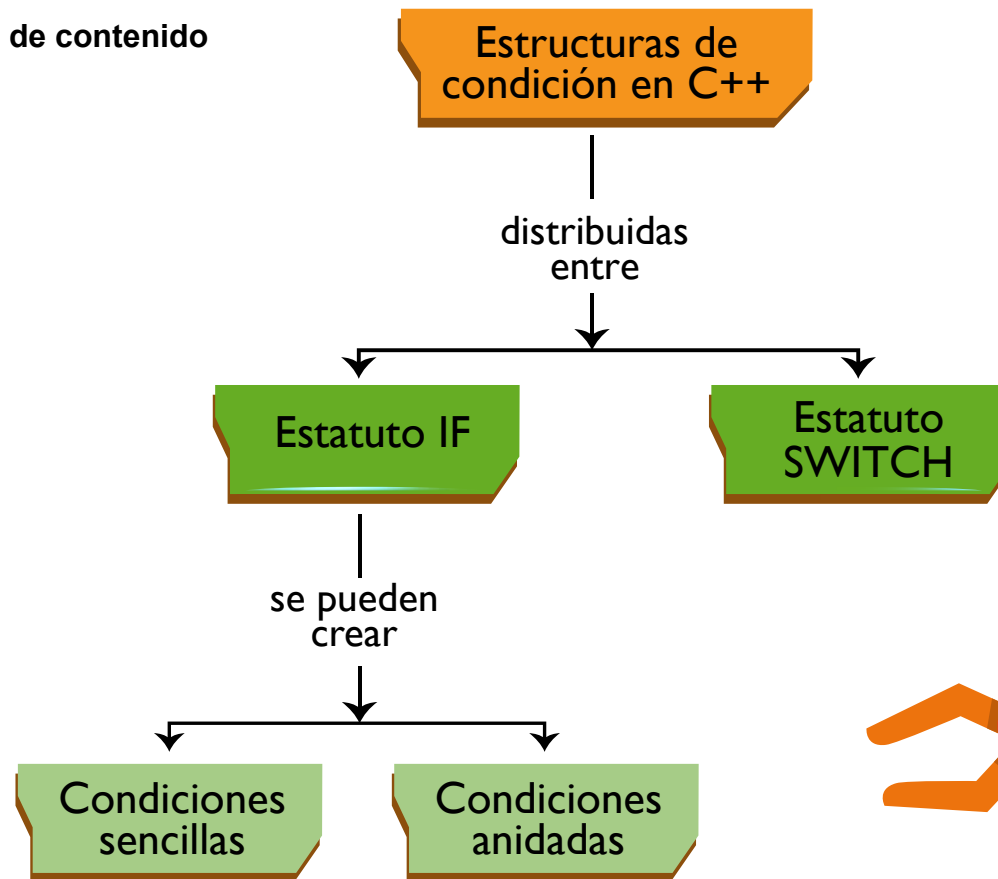
## 1. Introducción

La toma de decisiones en la programación de aplicaciones, está dada en gran medida por una serie de acontecimientos, los cuales, un programador debe anticipar para orientar a la máquina y evitar errores en tiempo de ejecución. Hasta el momento los ejercicios que se han desarrollado en el curso, funcionan de manera secuencial (la aplicación solicita un dato y después de digitado, ésta arroja un resultado). De ahora en adelante se tendrá un mayor control sobre las aplicaciones y dependiendo de la información digitada, el programa ejecutará determinado bloque de instrucciones.

C++ brinda al programador las estructuras de control if, if – else y switch que permiten direccionar el flujo de ejecución para dar solución a problemas de propósito general.



## 2. Estructura de contenido



### 3. Estructuras de condición en el lenguaje C++

#### 3.1. Condiciones sencillas

El estatuto if, permite cuestionar el valor de una expresión para ejecutar o no una serie de instrucciones, al evaluar dicha expresión, el valor generado es del tipo booleano; lo cual significa que puede obtener dos posibles valores: True o False. Es de anotar que la expresión puede ser del tipo numérico, relacional o lógico. Si la condición se cumple, se ejecuta una sentencia que va encerrada entre llaves { }. En caso de que no se cumpla, el programa sigue su curso de manera secuencial. La sintaxis de la sentencia if es la siguiente:

```
If(condición)
{
    Sentencia
}
```

El estatuto if – else, funciona igual que el anterior, con la única diferencia que permite establecer una sentencia en caso de que la condición no se cumpla. La sintaxis de la sentencia if – else es la siguiente:

```
If(condición)
{
```

```
    sentencia_1
}
else
{
    sentencia_2
}
```

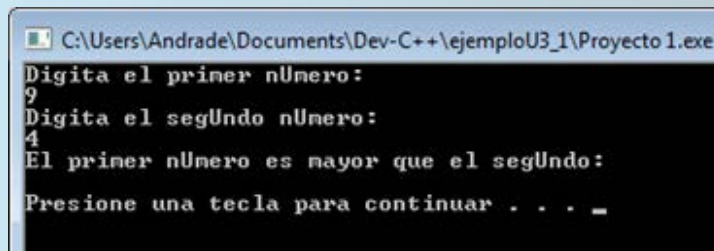
La estructura anterior se podría leer de la siguiente manera: “si la condición se cumple, entonces, ejecute la sentencia\_1. En caso de que no se cumpla, entonces, ejecute la sentencia\_2”.

A continuación, se mostrará un ejemplo que permitirá poner en práctica la sentencia de control if – else. Elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de sentencias\_1 y digite el siguiente código:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int n1, n2;
8
9     cout << "Digita el primer número: " << endl;
10    cin>>n1;
11    cout << "Digita el segundo número: " << endl;
12    cin>>n2;
13
14    if(n1 > n2)
15    {
16        cout << "El primer número es mayor que el segundo: " << endl << endl;
17    }
18
19    system("pause");
20    return EXIT_SUCCESS;
21 }
```

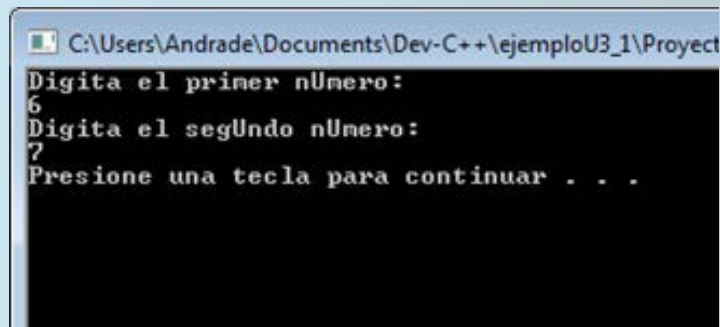


Como se observa, entre las líneas 14 y 17, se encuentra el código que hasta el momento es nuevo para el proceso de formación. Éste representa la ejecución de una sentencia que depende de una condición. Dicha condición se puede leer de la siguiente manera: “¿El primer número es mayor que el segundo número?”. En caso de cumplirse, el mensaje que se imprime en pantalla es el siguiente:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_1\Proyecto 1.exe
Digita el primer número:
9
Digita el segundo número:
4
El primer número es mayor que el segundo:
Presione una tecla para continuar . . . _
```

En caso de no cumplirse la condición, no se ejecuta nada y el resultado en pantalla es el siguiente:

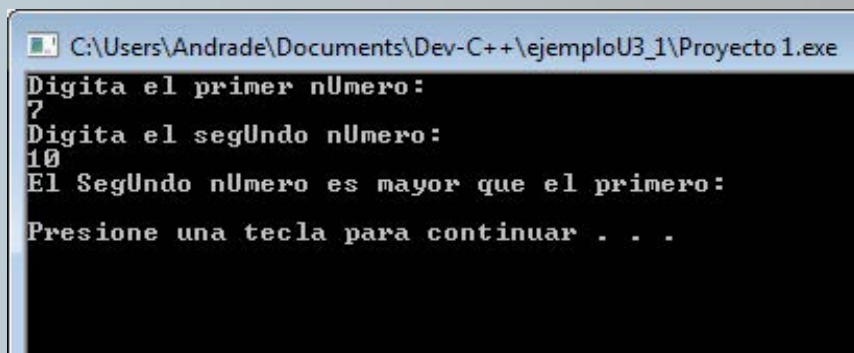


```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_1\Proyecto 1.exe
Digita el primer número:
6
Digita el segundo número:
7
Presione una tecla para continuar . . .
```

Ahora, realice los cambios en el código para agregar la sentencia en caso de que no se cumpla la condición, allí se podrá controlar el flujo de ejecución y mostrar un mensaje para que la aplicación quede completa. El código debe quedar de la siguiente manera:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int n1, n2;
8
9     cout << "Digita el primer número: " << endl;
10    cin>>n1;
11    cout << "Digita el segundo número: " << endl;
12    cin>>n2;
13
14    if(n1 > n2)
15    {
16        cout << "El primer número es mayor que el segundo: " << endl << endl;
17    }else{
18        cout << "El Segundo número es mayor que el primero: " << endl << endl;
19    }
20
21    system("pause");
22    return EXIT_SUCCESS;
23 }
```

Las líneas nuevas van desde la fila 17 hasta la 19. Al ejecutar de nuevo la aplicación y al digitar el segundo número mayor, la aplicación muestra lo siguiente:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_1\Proyecto 1.exe
Digita el primer número:
7
Digita el segundo número:
10
El Segundo número es mayor que el primero:
Presione una tecla para continuar . . .
```

Continuando con el ejercicio, se pedirá que realice una modificación en las sentencias para determinar si los números digitados son iguales. Para este caso se utilizará la estructura else if.

Después de digitar los números, la validación se hace de manera secuencial. Si la condición de la línea 14 no se cumple, el programa “salta” a la línea 18 para verificar que los números sean igual. En caso de que tampoco se cumpla, el programa salta a la línea 22 e imprime en pantalla el mensaje.

```
14  if(n1 > n2)
15  {
16      cout << "El primer número es mayor que el segundo: " << endl << endl;
17  }
18  else if (n1 == n2)
19  {
20      cout << "Los dos números son iguales: " << endl << endl;
21  }
22  else
23  {
24      cout << "El Segundo número es mayor que el primero: " << endl << endl;
25  }
```



### 3.2. Condiciones anidadas

Las condiciones anidadas se utilizan cuando se requiere más de una condición que se cumpla. Es decir, determinada condición se ejecuta solo si la anterior es verdadera. La sintaxis de una sentencia if anidada es la siguiente:

```
If(condición_1)
{
  If(condición_2)
  {
    Sentencia_1
  }
}
else
{
  Sentencia_2
}
```

A modo de ejemplo, elabore un nuevo proyecto en Dev C++, guárdelo con el nombre de **sentencias\_2** y digite el siguiente código:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float nota;
6     cout << "Digitar CalificaciOn" << endl;
7     cin >> nota;
8
9     //Si la nota es mayor que cero y menor que 5 es válida
10    //Si no, entonces genera un error
11    if(nota > 0.0 && nota < 5.0)
12    {
13        if(nota < 3)
14        {
15            cout << "Has REPROBADO la materia." << endl << endl;
16        }
17        else if(nota >= 3)
18        {
19            cout << "Has APROBADO la materia." << endl << endl;
20        }
21    }
22    else
23    {
24        cout << "La calificaciOn digitada no es válida." << endl << endl;
25    }
26
27    system("pause");
28    return EXIT_SUCCESS;
29 }
```

En este caso, el código de estudio se encuentra entre las líneas 9 y la 25. Cómo se observa en el comentario de las líneas 9 y 10, el objetivo del programa es recibir una calificación de una materia y mostrar como resultado si el estudiante aprobó o reprobó. Pero antes, es necesario validar que la calificación digitada es válida, es decir, que se encuentra entre el rango de 0.0 y 5.0. De lo contrario se determina que hay un error en su digitación.

```
11     if(nota > 0.0 && nota < 5.0)
12     {
13         if(nota < 3)
14         {
15             cout << "Has REPROBADO la materia." << endl << endl;
16         }
17         else if(nota >= 3)
18         {
19             cout << "Has APROBADO la materia." << endl << endl;
20         }
21     }
22     else
23     {
24         cout << "La calificaciOn digitada no es válida." << endl << endl;
25     }
```





En la línea 11 se comprueba si la nota es válida o no. El operador lógico and (&&) sirve para evaluar dos condiciones. Si las dos se cumplen, se da paso a las condiciones anidadas y si no, entonces el programa “salta” directamente a la línea 22 para mostrar en pantalla que la calificación no es válida. Los resultados en pantalla de la aplicación son los siguientes:

```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_2\sentencias_2.exe
Digitar CalificaciOn
6
La calificaciOn digitada no es vAlida.
Presione una tecla para continuar . . . _
```

```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_2\sentencias_2.exe
Digitar CalificaciOn
2.5
Has REPROBADO la materia.
Presione una tecla para continuar . . .
```

```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_2\sentencias_2.exe
Digitar CalificaciOn
4.5
Has APROBADO la materia.
Presione una tecla para continuar . . .
```

### 3.3. Estatuto SWITCH

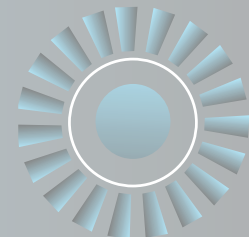
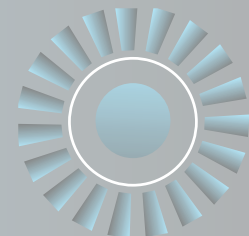
El estatuto switch, es una estructura que permite establecer los diferentes estados de una expresión y conforme a ello, ejecuta una sentencia para cada caso. Hay que tener en cuenta que solo se ejecuta la sentencia donde se cumple y después de finalizada, el programa sale del switch. La sintaxis es la siguiente:

```
switch(expresión)
{
    case expresión_valor_1:
        Sentencia_1
        break;

    case expresión_valor_2:
        Sentencia_2
        break;

    case expresión_valor_3:
        Sentencia_3
        break;

    default:
        Sentencia_4
        break;
}
```



A modo de ejemplo, se desarrollará el siguiente ejercicio que tiene como objeto determinar el nombre del día de la semana. Por consiguiente, elabore un nuevo proyecto, guárdelo con el nombre de sentencias\_3 y digite el siguiente código:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     int dia;
6     string nombre_dia;
7     cout << "Digita un día de la semana para determinar su nombre: " << endl << endl;
8     cin >> dia;
9
10    switch(dia)
11    {
12        case 1:
13            nombre_dia = "Domingo";
14            break;
15        case 2:
16            nombre_dia = "Lunes";
17            break;
18        case 3:
19            nombre_dia = "Martes";
20            break;
21        case 4:
22            nombre_dia = "Miércoles";
23            break;
24        case 5:
25            nombre_dia = "Jueves";
26            break;
27        case 6:
28            nombre_dia = "Viernes";
29            break;
30        case 7:
31            nombre_dia = "Sábado";
32            break;
33
34        default:
35            nombre_dia = "";
36            break;
37    }
38
39    if(nombre_dia != "")
40    {
41        cout << "El día es: " << nombre_dia << endl << endl;
42    }else
43    {
44        cout << "El número digitado no corresponde a días de la semana" << endl << endl;
45    }
46
47    system("pause");
48    return EXIT_SUCCESS;
49 }
```

A continuación, se explicarán las líneas que hasta el momento son nuevas para el proceso de formación.

La dinámica del ejercicio es la siguiente: se solicita al usuario digitar el día de la semana. Si digita un número entre 1 y 7 el switch evalúa el valor y dependiendo de éste, lo direcciona a una de las sentencias.

```
6      string nombre_dia;
```

En la línea 6 se define la variable nombre\_dia de tipo string. Se define de este modo porque permite asignar el valor sin necesidad de que sea digitado por el usuario. Ejemplo: nombre\_dia = "Domingo".

```
10     switch(dia)
11     {
12         case 1:
13             nombre_dia = "Domingo";
14             break;
```

En la línea 10 se define la estructura switch, y como expresión se establece la variable día, la cual ha recibido el número digitado por el usuario. Enseguida, el programa empieza a comparar de manera secuencial el valor de dicha variable en las cláusulas case.

Al leer la línea 12 y 13, éstas se puede expresar cómo: "En caso de que el día sea igual a uno (1), asigne a la variable nombre\_dia el valor 'Domingo'".

El break en la línea 14 le está diciendo al programa que ya cumplió con su objetivo, que no hay necesidad de validar los siguientes casos (case) y que debe salir del estatuto switch.

```
34         default:
35             nombre_dia = "";
36             break;
37     }
```

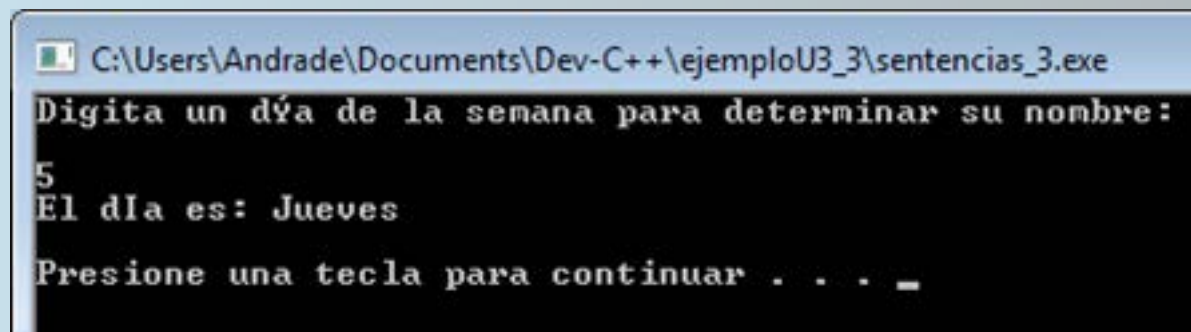
Antes de finalizar el estatuto switch, se debe establecer una opción por defecto (default). Esto es, en caso de que el valor digitado no corresponda a ninguna de las opciones. Entre la línea 34 y 36 se define la variable nombre\_dia cómo vacía al comprobar que el número digitado es inválido.

En esta instancia del programa la variable nombre\_dia ya tiene asignado un valor.

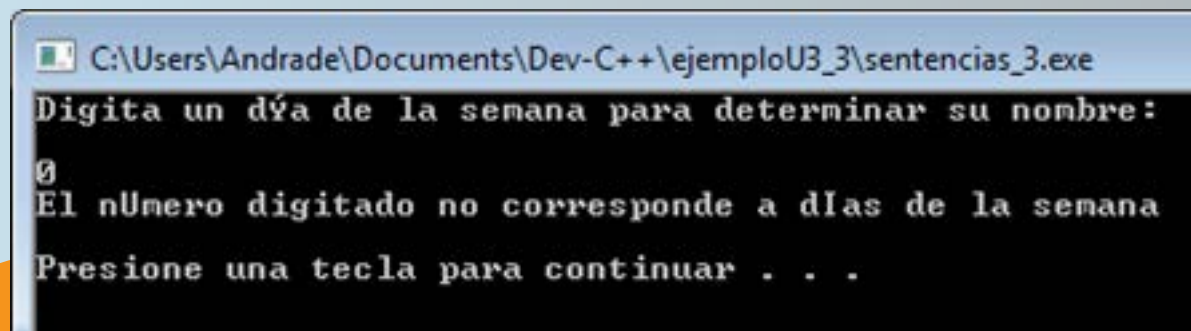
```
39     if(nombre_dia != "")
40     {
41         cout << "El día es: " << nombre_dia << endl << endl;
42     }else
43     {
44         cout << "El número digitado no corresponde a días de la semana" << endl << endl;
45     }
```



La condición de la línea 39 evalúa si la variable **nombre\_dia** es diferente (!=) de vacío. En caso de ser así, muestra en pantalla el valor que se asignó dentro del **switch**, de lo contrario, el mensaje para el usuario es de error:



```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_3\sentencias_3.exe
Digita un día de la semana para determinar su nombre:
5
El día es: Jueves
Presione una tecla para continuar . . . _
```

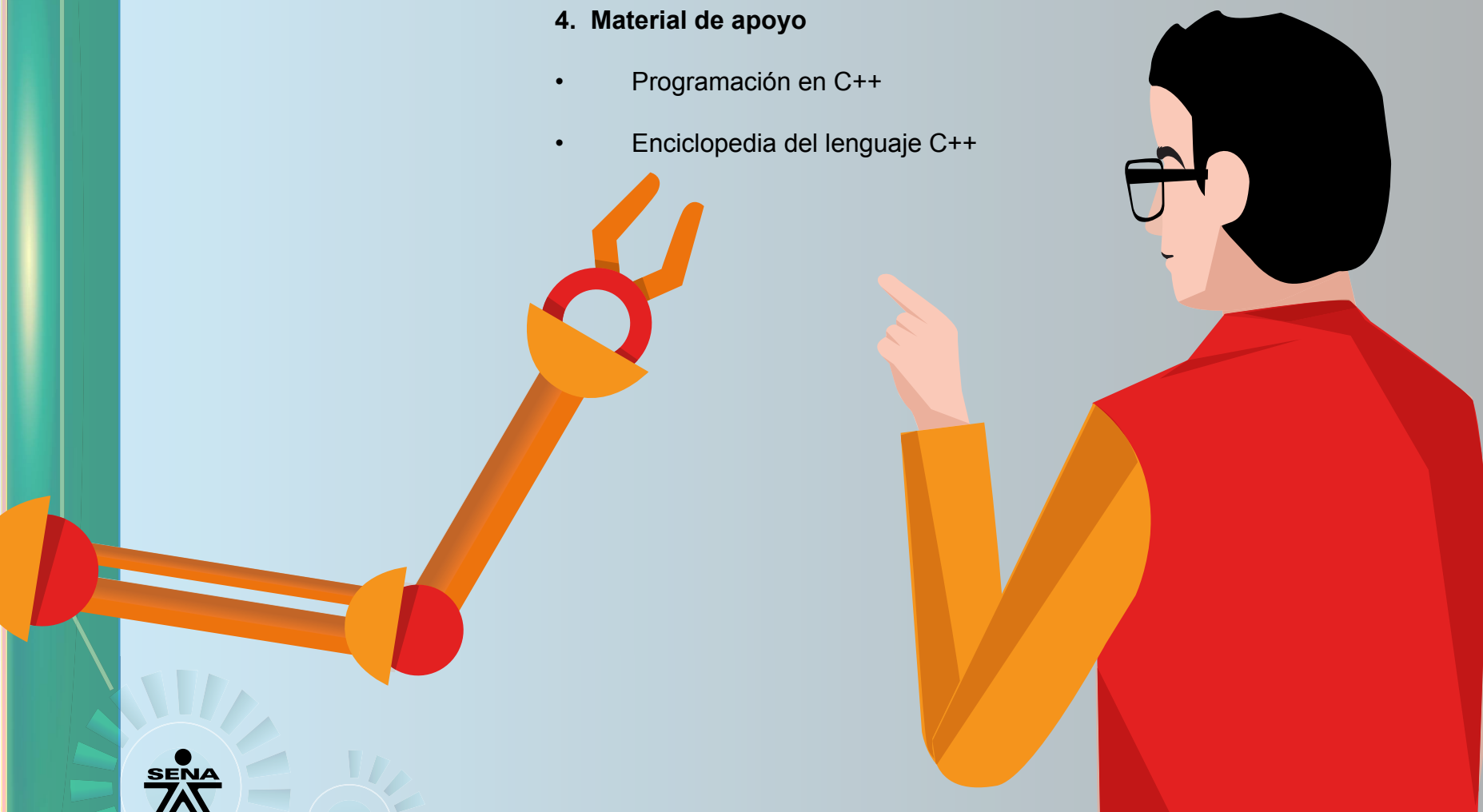


```
C:\Users\Andrade\Documents\Dev-C++\ejemploU3_3\sentencias_3.exe
Digita un día de la semana para determinar su nombre:
0
El número digitado no corresponde a días de la semana
Presione una tecla para continuar . . .
```



#### 4. Material de apoyo

- Programación en C++
- Enciclopedia del lenguaje C++





ABCD

## 5. Glosario

**Estatuto:** sinónimo de instrucción.

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

**Estatuto if:** una secuencia de instrucciones que se ejecuta o se evita dependiendo del valor (verdadero o falso) de una expresión booleana.

**Estatuto switch:** es una instrucción de decisión múltiple, donde el compilador prueba o busca el valor contenido en una variable contra una lista de constantes ints o chars, cuando el computador encuentra el valor de igualdad entre variable y constante, entonces ejecuta el grupo de instrucciones asociados a dicha constante, si no encuentra el valor de igualdad entre variable y constante, entonces ejecuta un grupo de instrucciones asociados a un default, aunque este último es opcional.

**Estructura de control:** las estructuras de control controlan el flujo de un programa o función. Permiten combinar instrucciones o sentencias individuales en una simple unidad lógica con un punto de entrada y un punto de salida.

**Sintaxis:** reglas que definen cómo se forman las instrucciones de un lenguaje de programación específico.



## 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
<b>Autor</b>	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.

