22 June 2020

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 22 June 2020 | Name: | Srinidhi J C |
| Course: | **C++ Programming** | USN: | 4al16ec078 |
| Topic: |  | Semester & Section: | 8th & b |
| Github Repository: | SrinidhiJC078 |  |  |
| FORENOON SESSION DETAILS | | | | | |
| Image of sessionA screenshot of a cell phone  Description automatically generated  A screenshot of a cell phone  Description automatically generated | | | | | |

Report:

A C++ program is a collection of commands or statements.  
  
Below is a simple code that has "Hello world!" as its output.

#include <iostream>  
using namespace std;  
  
int main()  
{  
cout << "Hello world!";  
return 0;  
}

Let's break down the parts of the code.#include <iostream>  
C++ offers various headers, each of which contains information needed for programs to work properly. This program calls for the header **<iostream>**.  
The **number sign (#)** at the beginning of a line targets the compiler's pre-processor. In this case, **#include** tells the pre-processor to include the **<iostream>** header.

Curly brackets { } indicate the beginning and end of a function, which can also be called the function's body. The information inside the brackets indicates what the function does when executed.

In C++, **streams** are used to perform input and output operations.  
In most program environments, the standard default output destination is the screen. In C++, **cout** is the stream object used to access it. **cout** is used in combination with the insertion operator. Write the insertion operator as **<<** to insert the data that comes after it into the stream that comes before.  
In C++, the **semicolon** is used to terminate a statement. Each statement must end with a **semicolon**. It indicates the end of one logical expression.

# Statements

A **block** is a set of logically connected statements, surrounded by opening and closing curly braces.  
For example: {  
cout << "Hello world!";   
return 0;  
}  
You can have multiple statements on a single line, as long as you remember to end each statement with a **semicolon**. Failing to do so will result in an error.

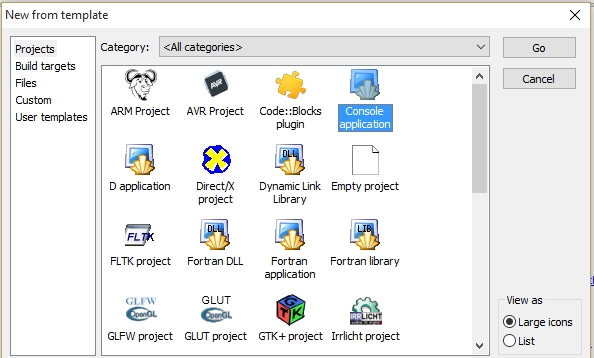
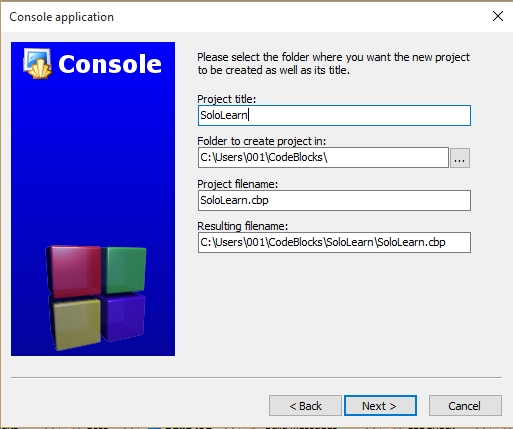
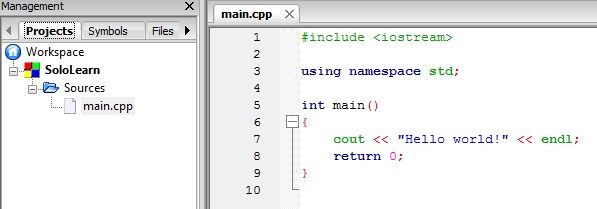
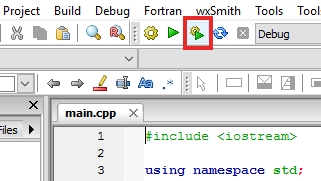
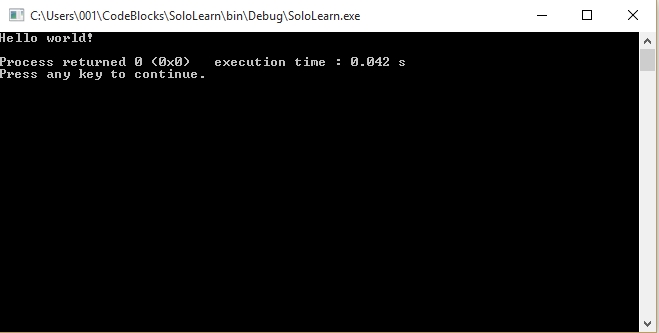
# Return

The last instruction in the program is the **return** statement. The line **return 0;** terminates the **main()** function and causes it to return the value 0 to the calling process. A non-zero value (usually of 1) signals abnormal termination.  
If the return statement is left off, the C++ compiler implicitly inserts "**return 0;**" to the end of the main() function.  
Tap **Continue** to learn more about functions, return, and other topics.

# Getting the Tools

You can run, save, and share your C++ codes on our **Code Playground**, without installing any additional software.  
Reference this lesson if you need to install the software on your computer.   
You need both of the following components to build C++ programs.  
1. **Integrated Development Environment (IDE)**: Provides tools for writing source code. Any text editor can be used as an IDE.  
2. **Compiler**: Compiles source code into the final executable program. There are a number of C++ compilers available. The most frequently used and free available compiler is the **GNU C/C++** compiler.  
Various C++ IDEs and compilers are available. We'll use a free tool called **Code::Blocks**, which includes both an IDE and a compiler, and is available for Windows, Linux and MacOS.  
To download Code::Blocks, go to http://www.codeblocks.org/, Click the **Downloads** link, and choose "**Download the binary release**".  
Choose your OS and download the setup file, which includes the C++ compiler (For Windows, it's the one with **mingw** in the name).  
Make sure to install the version that includes the **compiler**.

# Getting the Tools

To create a project, open Code::Blocks and click "**Create a new project**" (or File->New->Project).  
This will open a dialog of project templates. Choose **Console application** and click **Go**.   
Go through the wizard, making sure that C++ is selected as the language.  
Give your project a name and specify a folder to save it to.   
Make sure the **Compiler** is selected, and click **Finish**.  
**GNU GCC** is one of the popular compilers available for Code::Blocks.  
On the left sidebar, expand **Sources**. You'll see your project, along with its source files. Code::Blocks automatically created a **main.cpp** file that includes a basic Hello World program (C++ source files have .cpp, .cp or .c extensions).  
Click on the "**Build and Run**" icon in the toolbar to compile and run the program.  
A **console window** will open and display program output.   
Congratulations! You just compiled and ran your first C++ program!  
You can run, save, and share your C++ codes on our **Code Playground**, without installing any additional software.  
Reference this lesson if you need to install the software on your computer.

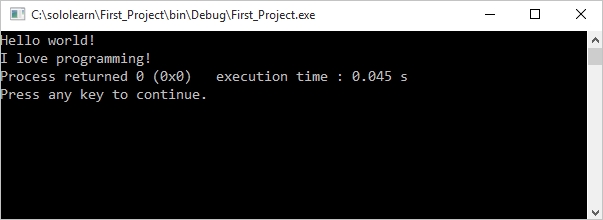
You can add multiple insertion operators after **cout**.

cout **<<** "This " **<<** "is " **<<** "awesome!";

# **Result:** New Line

The **cout** operator does not insert a line break at the end of the output.   
One way to print two lines is to use the **endl** manipulator, which will put in a line break.

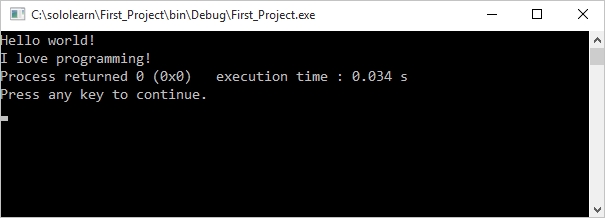
#include <iostream>  
using namespace std;  
  
int main()  
{  
cout << "Hello world!" << **endl**;  
cout << "I love programming!";  
return 0;  
}

The **endl** manipulator moves down to a new line to print the second text. 

# New Lines

The new line character **\n** can be used as an alternative to **endl**.  
The backslash (\) is called an **escape character**, and indicates a "special" character.  
  
**Example:**

#include <iostream>  
using namespace std;  
  
int main()  
{  
cout << "Hello world! **\n**";  
cout << "I love programming!";  
return 0;  
}

**Result:** 

# Declaring Variables

You have the option to assign a value to the variable at the time you declare the variable or to declare it and assign a value later.  
You can also change the value of a variable.  
**Some examples:** int a;   
int b = 42;  
  
a = 10;  
b = 3;

# User Input

To enable the user to input a value, use **cin** in combination with the extraction operator (**>>**). The variable containing the extracted data follows the operator.  
The following example shows how to accept user input and store it in the **num** variable: int num;  
**cin >> num;**

# Accepting User Input

The following program prompts the user to input a number and stores it in the variable **a**:

#include <iostream>  
using namespace std;  
  
int main()  
{  
int a;  
cout << "Please enter a number \n";  
**cin >> a;**  
  
return 0;  
}

When the program runs, it displays the message "Please enter a number", and then waits for the user to enter a number and press Enter, or Return.   
The entered number is stored in the variable **a**.

You can accept user input multiple times throughout the program:

#include <iostream>  
using namespace std;  
  
int main()  
{  
int a, b;  
cout << "Enter a number \n";  
cin >> a;  
cout << "Enter another number \n";  
cin >> b;  
  
return 0;  
}

Let's create a program that accepts the input of two numbers and prints their sum.

#include <iostream>  
using namespace std;  
  
int main()   
{  
int a, b;  
int sum;  
cout << "Enter a number \n";  
cin >> a;  
cout << "Enter another number \n";  
cin >> b;  
sum = a + b;  
cout << "Sum is: " << sum << endl;  
  
return 0;  
}

Let's create a program that accepts the input of two numbers and prints their sum.

#include <iostream>  
using namespace std;  
  
int main()   
{  
int a, b;  
int sum;  
cout << "Enter a number \n";  
cin >> a;  
cout << "Enter another number \n";  
cin >> b;  
sum = a + b;  
cout << "Sum is: " << sum << endl;  
  
return 0;  
}

The same shorthand syntax applies to the multiplication, division, and modulus operators.x \*= 3; // equivalent to x = x \* 3  
x /= 2; // equivalent to x = x / 2  
x %= 4; // equivalent to x = x % 4

# Increment Operator

The **increment** operator is used to increase an integer's value by one, and is a commonly used C++ operator.  
**x++;** //equivalent to x = x + 1

For example:

int x = 11;  
x++;  
cout << x;  
  
// Outputs 12

The increment operator has two forms, **prefix** and **postfix**.++x; // prefix  
x++; // postfix  
**Prefix** increments the value, and then proceeds with the expression.   
**Postfix** evaluates the expression and then performs the incrementing.  
  
**Prefix example:** x = 5;  
y = **++x**;  
// x is 6, y is 6  
**Postfix example:** x = 5;  
y = **x++**;  
// x is 6, y is 5

# Decrement Operator

The **decrement** operator (--) works in much the same way as the increment operator, but instead of increasing the value, it decreases it by one.--x; // prefix  
x--; // postfix