Lab 01

Objectives

After performing this lab, students will be able to

- Install Dev-C++ IDE
- Configure DEV C++ Environment
- Create new Program or Project
- Write the first C++ program and understanding its different parts
- Use the comments in their programs
- Use escape sequence in their programs

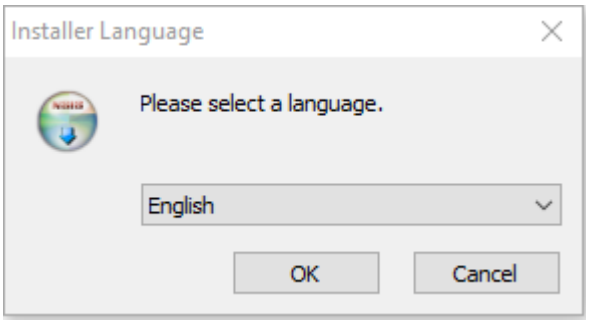
Installing Dev-C++ IDE

Dev-C++ is a full-featured C and C++ Integrated Development Environment (IDE) for Windows platforms that is distributed under GNU general public license. It was originally developed by Colin Laplace and first released in 1998.

An integrated development environment provides comprehensive facilities to programmers. It normally consists of source code editor, build automation tools and debugger. Most modern IDEs offer intelligent code completion features. Follow the below given steps to install the DevC++.

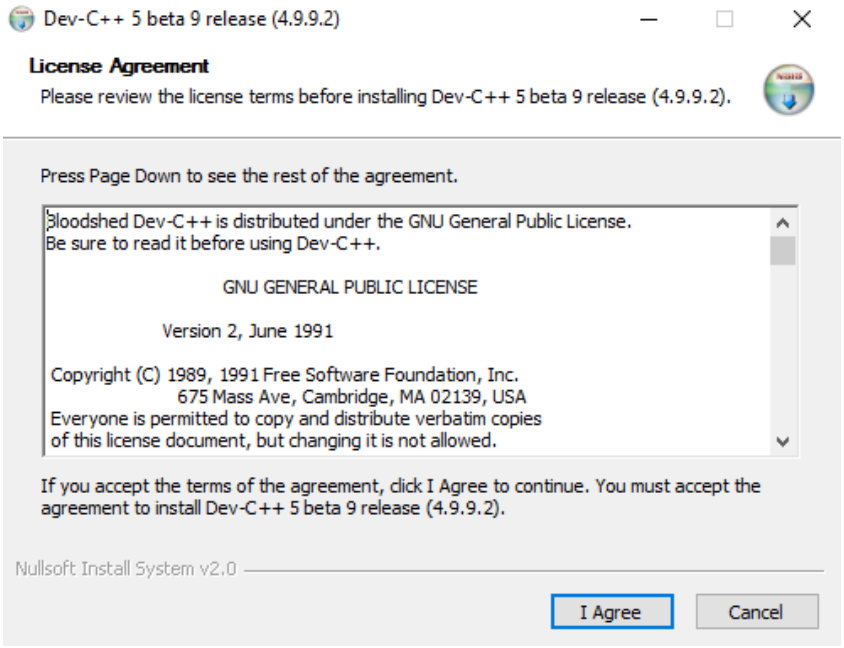
To install Dev-C++, you first need to have its setup. You can download the Dev-C++ setup from the URL: https://sourceforge.net/projects/orwelldevcpp/

Once the setup file is downloaded, you can start the installation process. Open the setup file and you will see a screen similar to shown below:

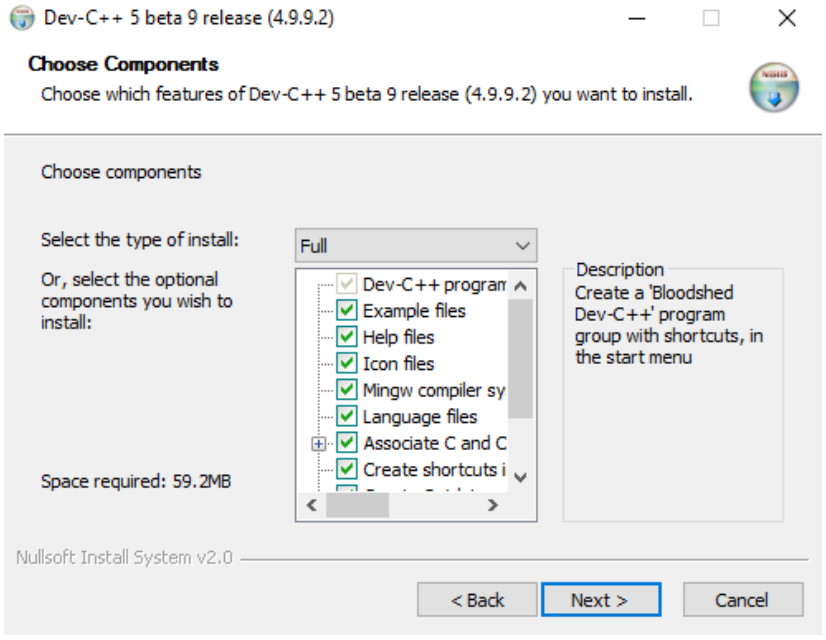


Select the English language from the drop-down menu (or leave it to default if it is already selected) and click on the OK button.

After that, a window (like the one given below) will appear asking you to agree to the License Agreement.

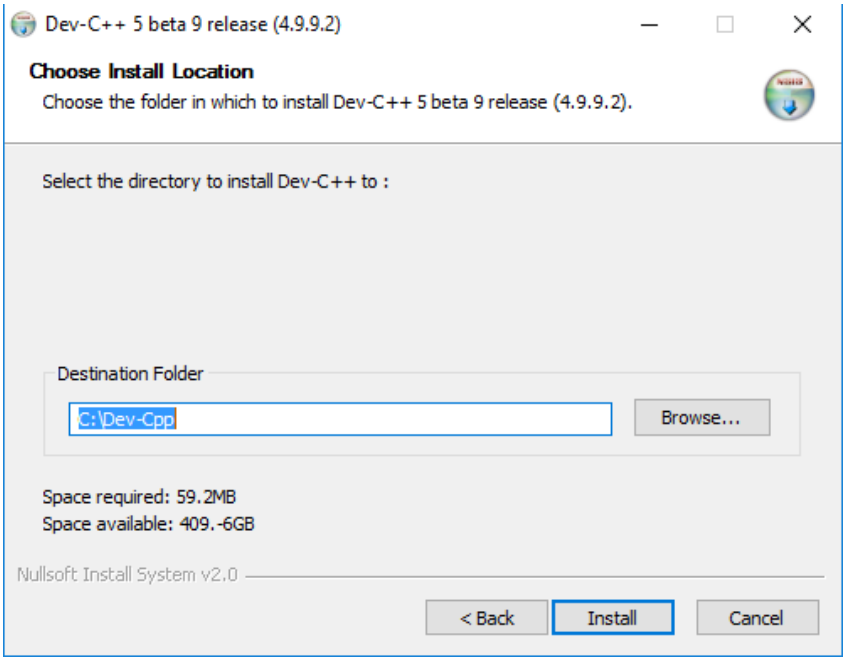


Click the I Agree button. Then, the following window will appear



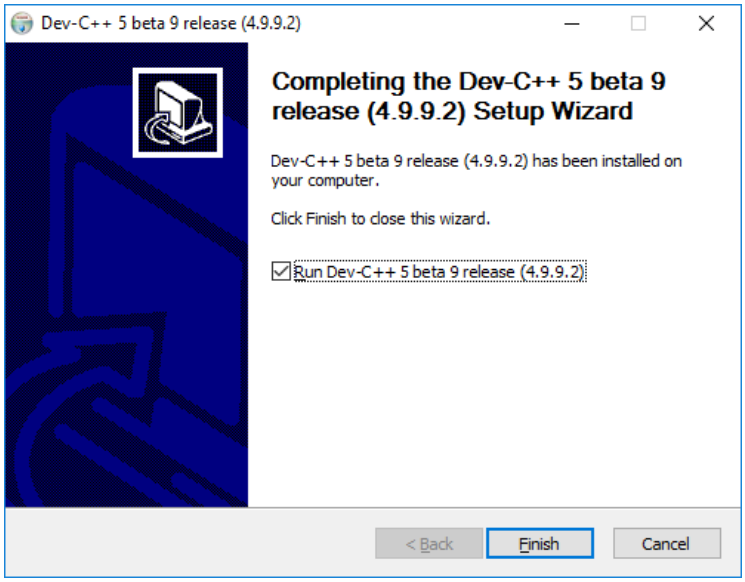
Leave the default option selected/checked and click the Next button.

Then, the next window will ask the location where you want to install the Dev-C++ as shown in the following figure



Leave it to default or change the location by clicking the Browse button and selecting your desired location, and click the Install button.

Then, the installation process will start and will be finished after some time. Lastly, click the Finish button.

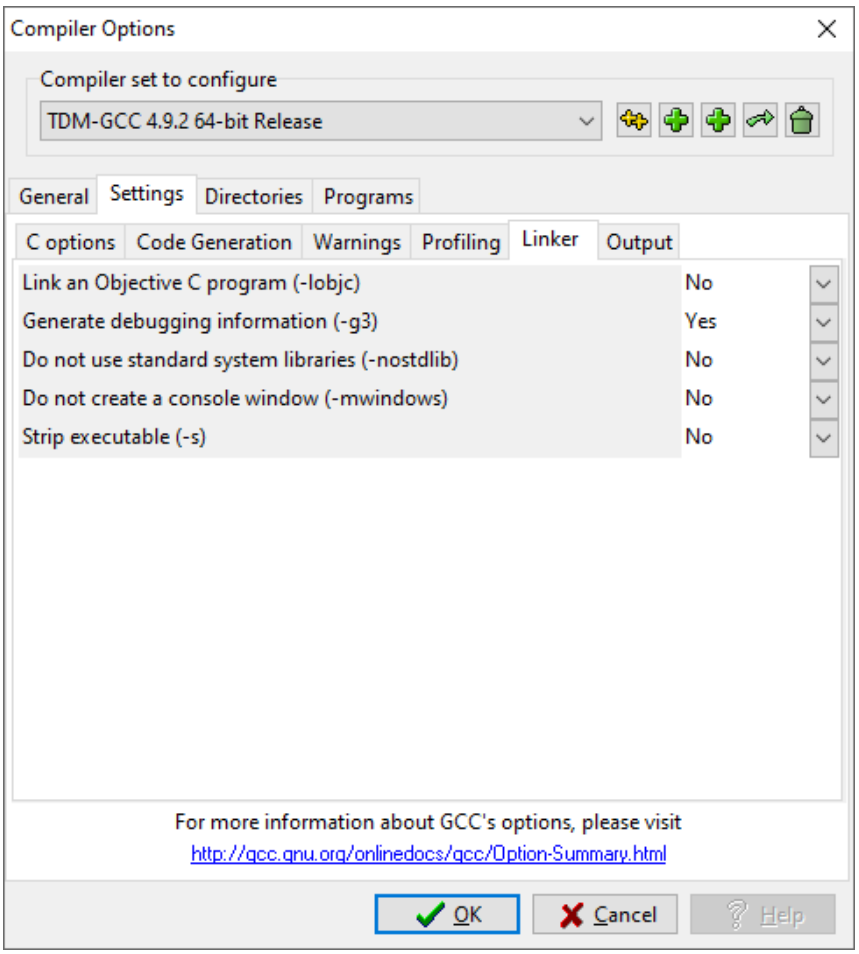


Configuring Dev-C++ Environment

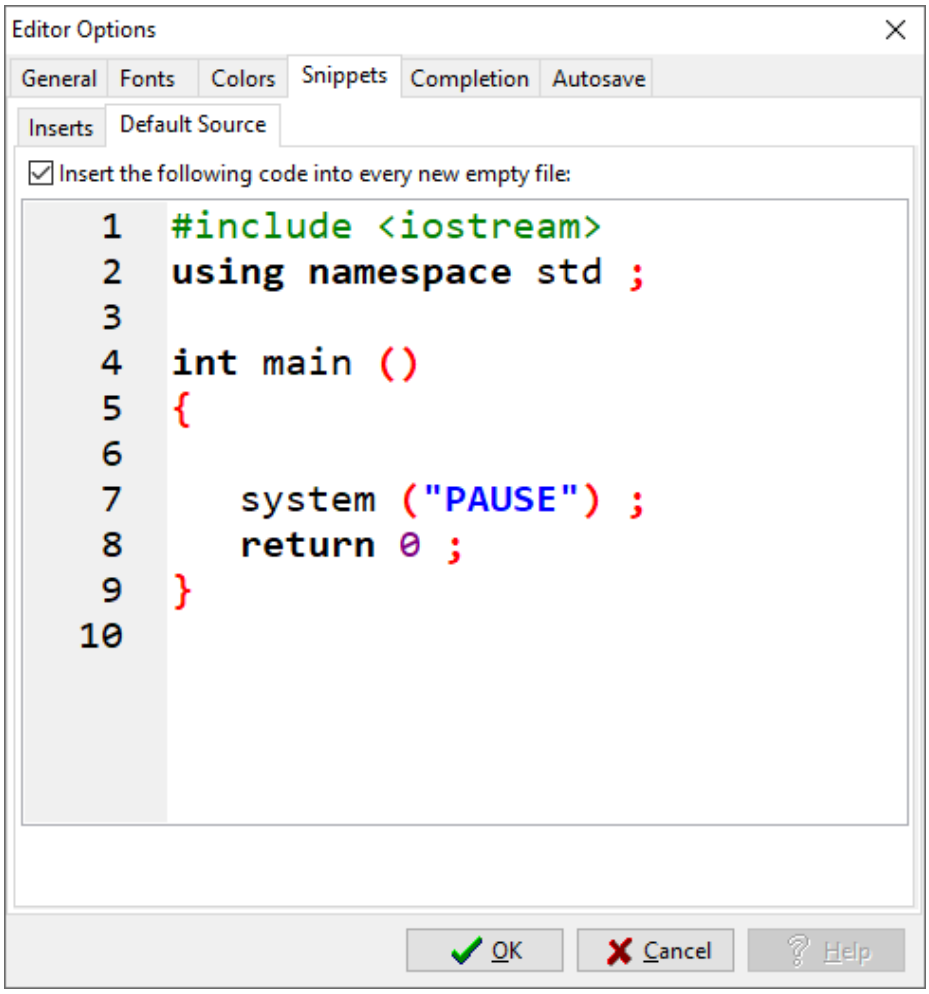
Open DevC++ from start menu and go to

1. Tools→Compiler Options→Settings→Linker→Set Generate Debugging information bit to

Yes from ***No***.



2. Tools→Environment Options→Directories tab→set user default directories settings (Here your project file will be saved by default).
3. Tools→Editor Options→Snippets→Default Source →Check/select the "Insert the following code into every new empty file" option and click "Ok" (Whenever you create a new file you will see following default code in that file. We will discuss this code in subsequent sections in this lab)



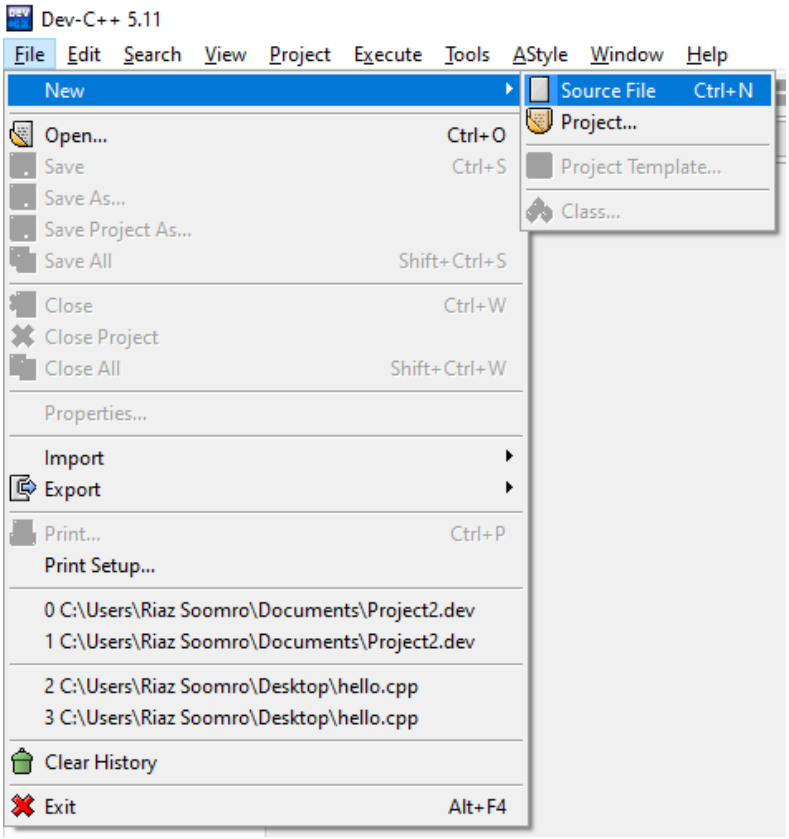

```
#include <iostream>
using namespace std;

int main ()
{

   system ("PAUSE") ;
   return 0 ;
}
```

Creating a new program or project:

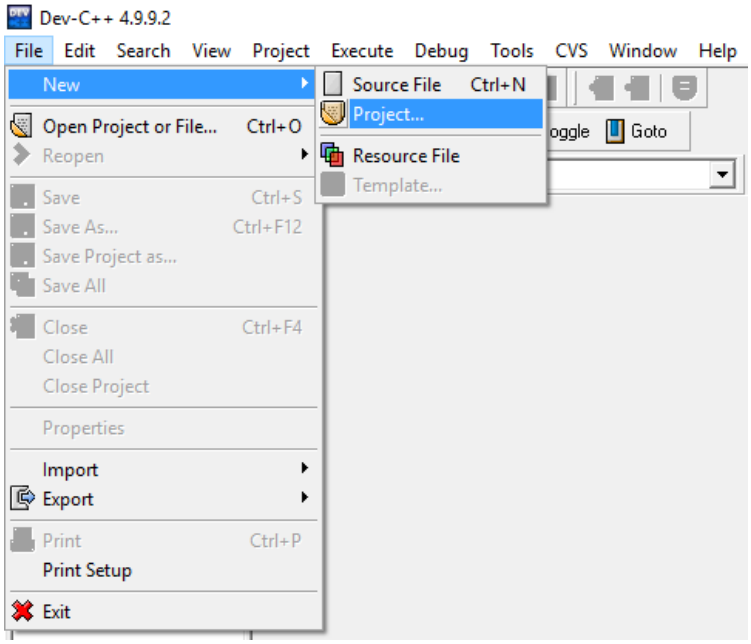
For creating a new program you have to either open a new source file or new project. If you want to create a single file the best choice is to open a new source file. If you want to create a project which may have multiple files then opening a new project is good choice.

Opening a new source file:

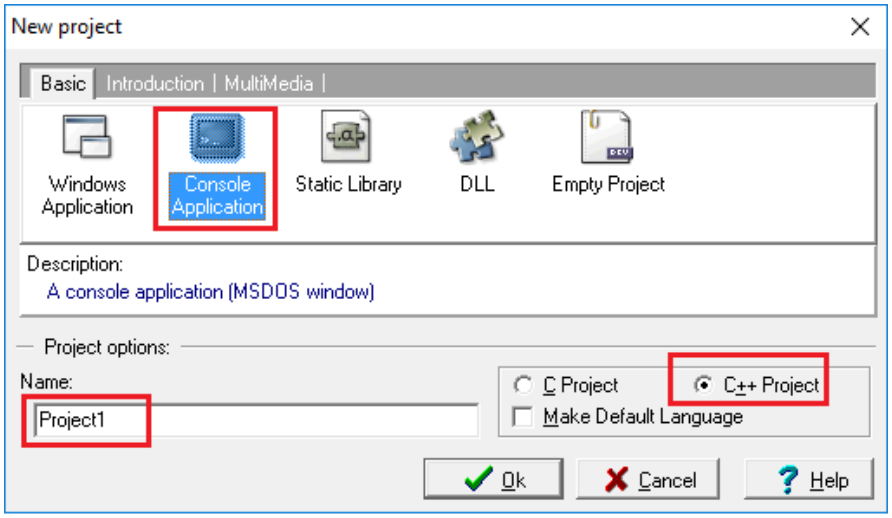
Go to file→New→Source File→Save the source file with the meaningful name→Start writing your code



Opening a new project:
Go to file→New→Project→Create the folder→save project inside the folder→Also save main file inside the folder→Ok

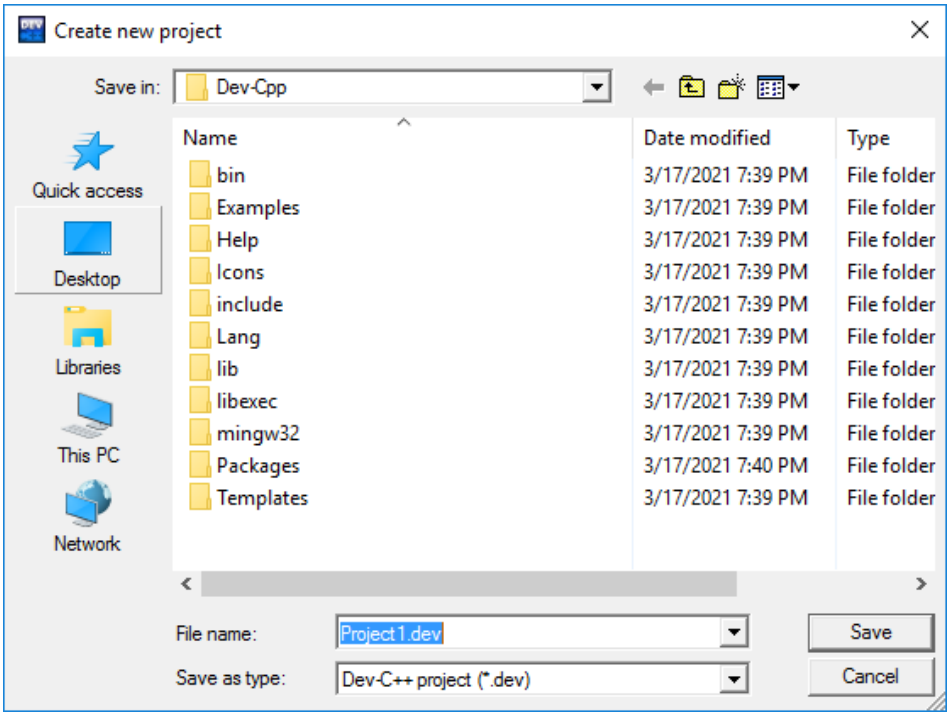


Then the following window will appear



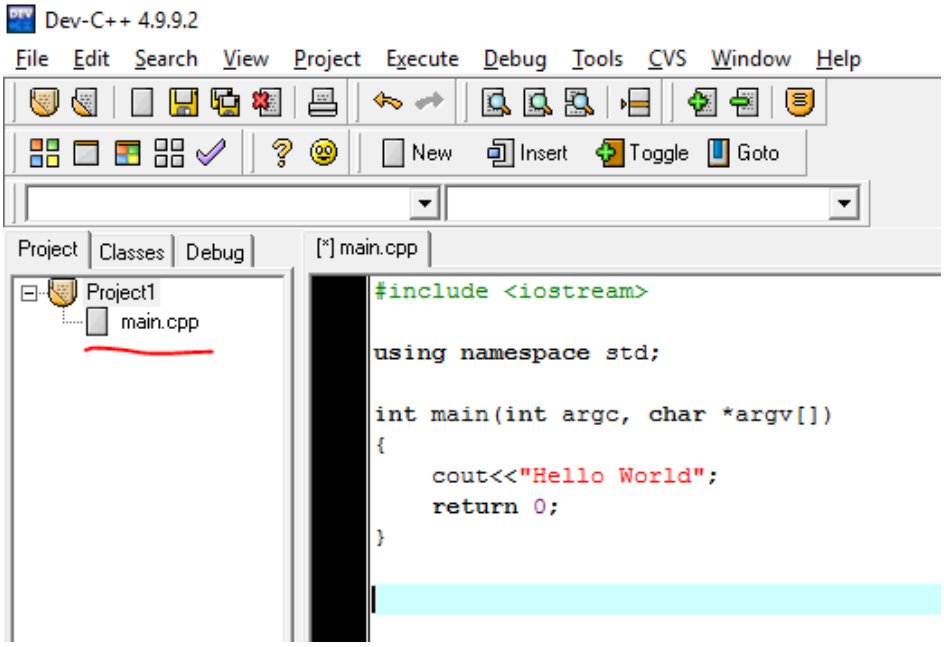
Select the "Console Application" option as the type of the project. Then, give some name to your project. By default, Dev-C++ gives it the name "Project1" (as highlighted with the red box in the above-given figure) but you can change it any name you want. Also, select "C++ Project" as the type of project to indicate that you will use C++ language for this project. Finally, click the OK button.

After that, the following window will be shown asking you to select the location where the project will be saved.



Select any location of your choice and click the Save button.

After indicating the folder where the project configuration file (.dev) will be saved, the IDE generates a basic source code file (by default, main.cpp), adds some code as shown below. These files are not saved in the project folder until the programmer saves or compiles the program.



Sometimes, the IDE may have written **int argc, char *argv[]** inside the brackets of the main () function, like shown below.

```
int main(int argc, char** argv) {
    return 0;
}
```

You can remove the "**int argc, char *argv[]**" written inside the brackets of the main () function

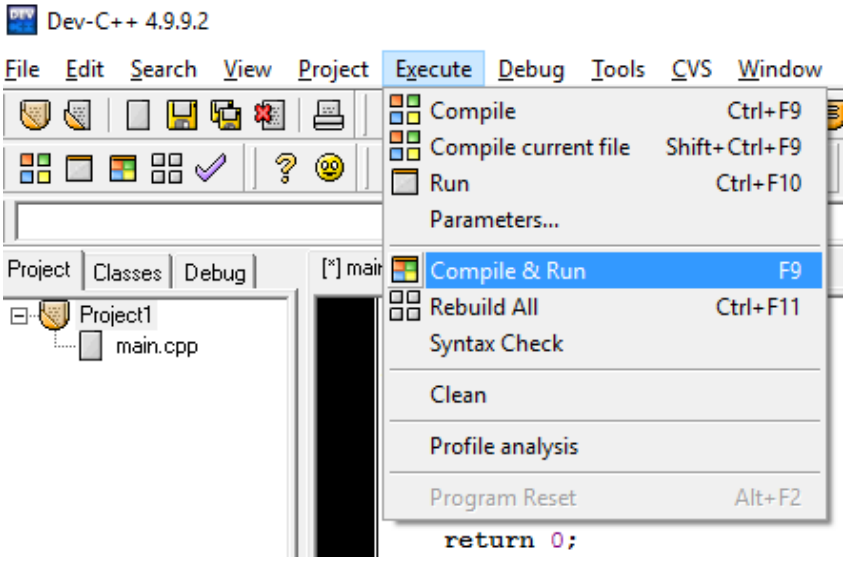
Writing the first C++ program and understanding its different parts

For now open a new source file (if it not already opened), save it with the name of HELLOWORLD and type following program.

```
#include <iostream>
using namespace std ;

int main ()
{
  cout <<"Hello World.. This is my first program in C++" ;
  system ("PAUSE") ;
  return 0 ;
}
```

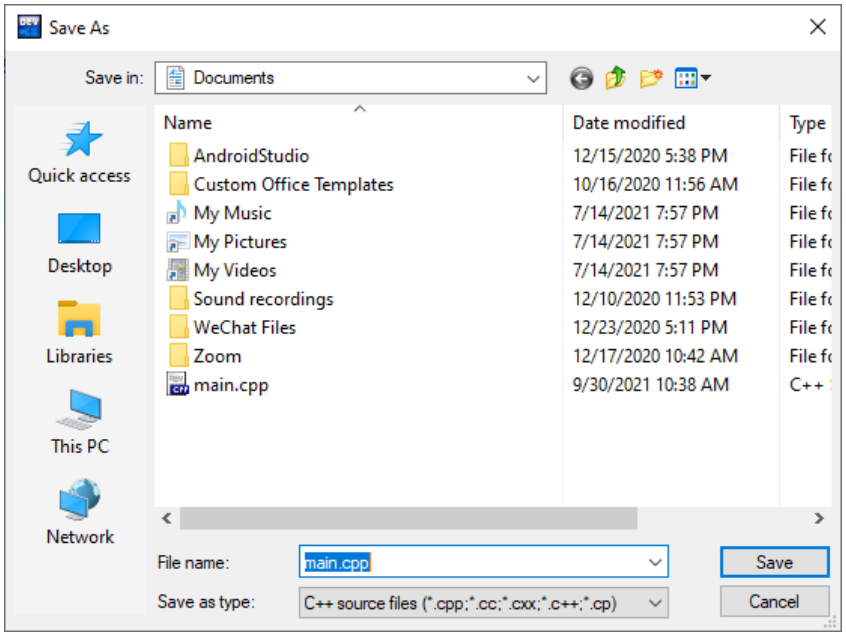
Compiling and executing the project

To compile the project, click **Execute -> Compile** (or press **Ctrl+F9** key on the keyboard). Once the project is compiled successfully, we need to run it. To run or execute Click on **Execute -> Run** (or press **Ctrl+F10** key). You can perform the both operations (compiling and running the code) with a single shortcut key **F9**.



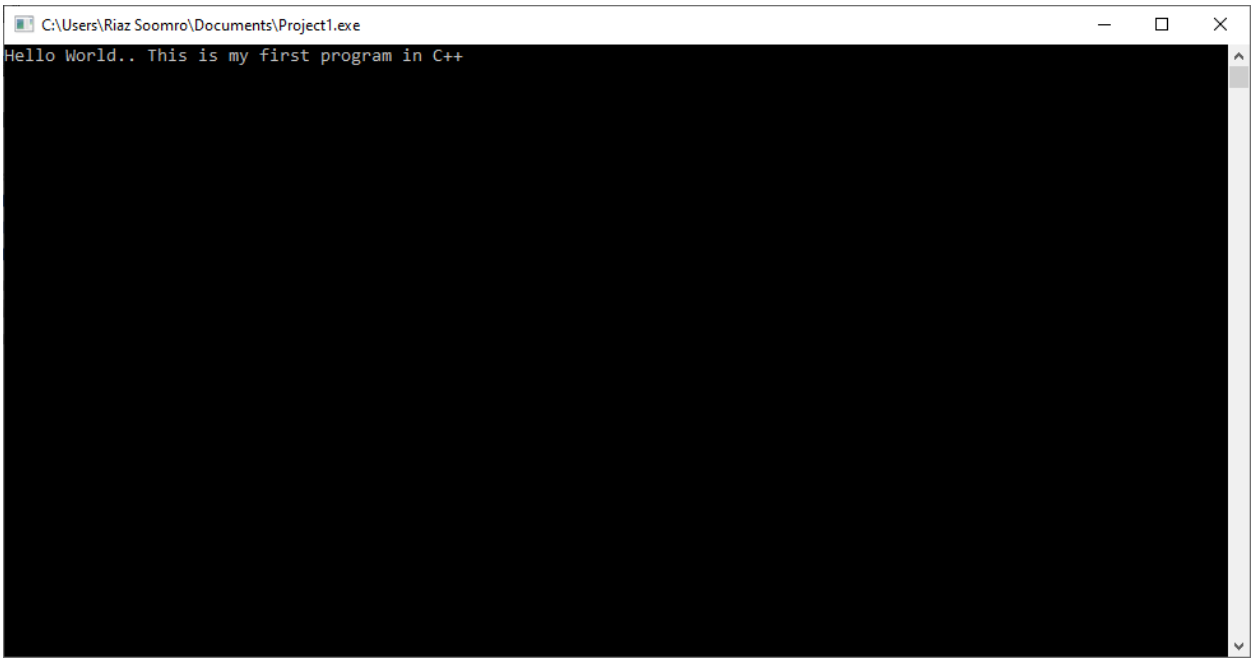
Note: The keyboard shortcut keys may be different for different versions of the Dev-C++. So, please check according to the version installed on your system.

You will have to save your .c++ file first in order to compile and run it. So, the IDE may show you the Save (or Save As) dialog box (as shown below)



You should save the file at your desired location with a meaningful name.

If you have followed the step correctly and everything goes right, then you will see the following screen showing the output of your program.



Structure of a C++ Program

#include <iostream>

Lines beginning with a hash sign (#) are called preprocessor directives. They are special lines interpreted before the compilation of the program itself begins. In this case, the directive #include <iostream>, instructs the preprocessor to include a section of standard C++ code, known as *header iostream* that allows to perform standard input and output operations, such as writing the output of this program to the screen.

using namespace std

It tells the compiler that we use the namespace named std. "**std**" is an abbreviation for standard. So that means we use all the things within "std" namespace. If we don't want to use this line of code, we can use the things in this namespace like ***cout*** as **std::cout** every time we want to use it.

A blank line

Blank lines have no effect on a program. They simply improve readability of the code.

int main ()

This line initiates the declaration of the main function. Functions will be discussed in detail in the later lectures. The function named ***main*** is a special function in all C++ programs; it is the function (or part of code) that is first called/executed when the program is run. The execution of all C++ programs begins with the ***main*** function, regardless of where the function is actually located within the code.

{ and }

The opening brace "**{**" indicates the beginning of main function's definition, and the closing brace "**}**", indicates its end. Everything between these braces is the function's body that defines what happens when main function is called/executed. All functions use braces to indicate the beginning and end of their definitions.

cout

cout is the "**c**haracter **out**put stream". It is pronounced "see-out". It is used for displaying (also called as printing) something on a monitor. You can consider the ***cout*** as the name for the monitor/screen.

"**<<**" is an operator that points to where the data is to end up (or in other words, where the string of characters enclosed in double quotes coming after it is to be sent/displayed). In this case, the phrase ***Hello World.. This is my first program in C++*** will be printed/shown on the screen/monitor. This operator << indicates that data (which is on the right side) goes towards the ***cout*** (or the monitor/screen).

"Hello World.. This is my first program in C++" is what's being outputted here. You need double quotes around text. Whatever is written inside the double quotes, will be displayed on the screen/monitor. Semicolon ";" is a punctuation that tells the computer that you are at the end of a statement. It is similar to a period (full stop) in a sentence in English.

Activity 1

Write a C++ program that prints/displays your name on the screen.

Making the text appear on the new line

By default, all the text written through the ***cout*** statements will be printed/displayed on the same line. If you want the text written in the second line of code to be displayed on the next line of the screen, then you can do that using one of the following ways.

1. Write **<<endl** at the end of the first line (before the semi colon ;).

```
cout<<"This is the first line."<<endl;
cout<<"This is the second line.";
```

2. Write \n (backslash n) at the end of the first line (inside the double quotes).

```
cout<<"This is the first line.\n";
cout<<"This is the second line.";
```

In both of the cases mentioned above, the second line of text will be displayed/printed on the second line of the screen as shown below

```
This is the first line.
This is the second line.
```

Activity 2

Write a C++ program that prints your details, like name, student ID, batch and the department, each on a separate line.

Comments

A comment is a programmer-readable note that is inserted directly into the source code of the program. Comments are ignored by the compiler and are for the programmer's use only.

In C++ there are two different styles of comments, both of which serve the same purpose: to help programmers document the code in some way.

Single-line comments

The // symbol begins a C++ single-line comment, which tells the compiler to ignore everything from the // symbol to the end of the line. For example:

```
std::cout << "Hello world!"; // Everything from here to the end of the line is ignored
```

Typically, the single-line comment is used to make a quick comment about a single line of code.

Having comments to the right of a line can make both the code and the comment hard to read, particularly if the line is long. If the lines are fairly short, the comments can simply be aligned (usually to a tab stop), like so:

```
std::cout << "Hello world!\n";                // std::cout lives in the iostream library
std::cout << "It is very nice to meet you!\n"; // this is much easier to read
std::cout << "Yeah!\n";                       // don't you think so?
```

However, if the lines are long, placing comments to the right can make your lines really long. In that case, single-line comments are often placed above the line it is commenting:

```
// std::cout lives in the iostream library
std::cout << "Hello world!\n";

// this is much easier to read
std::cout << "It is very nice to meet you!\n";

// don't you think so?
std::cout << "Yeah!\n";
```

Multi-line comments

The /* and */ pair of symbols denotes a C-style multi-line comment. Everything in between the symbols is ignored.

```
/* This is a multi-line comment.
   This line will be ignored.
   So will this one. */
```

Since everything between the symbols is ignored, you will sometimes see programmers "beautify" their multi-line comments:

```
/* This is a multi-line comment.
 * the matching asterisks to the left
 * can make this easier to read
 */
```

Multi-line style comments cannot be nested. Consequently, the following will have unexpected results:

```
/* This is a multi-line /* comment */ this is not inside the comment */
// The above comment ends at the first */, not the second */
```

When the compiler tries to compile this, it will ignore everything from the first / to the first /. Since "this is not inside the comment */" is not considered part of the comment, the compiler will try to compile it. That will inevitably result in a compile error.

Escape Sequence

The escape sequences are special non-printing characters that are used to control the printing behavior of the output stream objects (such as 'cout'). These characters are not displayed in the output. An escape sequence is prefixed with a backslash (\) and a coded character is used to control the printing behavior. The backslash (\) is called an escape character. So the escape sequence looks like two characters.

The escape sequence is used inside a string constant or independently. These are written in single or double-quotes. The escape sequence can be inserted in any position of the string such as:

- At the beginning of the string.
- In the middle of the string.
- At the end of the string etc.

For example, you want to put a line break in the output of a C++ statement then you will use "\n" character which is an escape sequence itself (it is an alternate of ***endl***).

New line (\n)

When a new line is necessary in the output, then this escape sequence is used. For example:

```
cout<<"COMPUTER\nSCIENCE";
```

First of all, "COMPUTER" is printed and "\n" shifts the cursor to the next line. Then "SCIENCE" is printed on second line. A screenshot of output is shown below:

```
COMPUTER
SCIENCE
```

Tab (\t)

A TAB is equal to approximately eight spaces. Whenever TAB button is pressed from the keyboard, then 8 spaces are left blank. This escape sequence performs the functionality of TAB key in the output stream. The code given below will insert a TAB between two words.

```
cout<<"COMPUTER\tSCIENCE";
```

```
COMPUTER        SCIENCE
```

As you can see after printing "COMPUTER", 8 spaces are left blank and then "SCIENCE" is printed on the screen.

Alert Bell (\a):

This escape sequence is used to play beep during execution. For example:

```
cout<<"COMPUTER\aSCIENCE";
```

First of all, “COMPUTER” is printed then a beep is played and after that “SCIENCE” is printed.

Backspace (\b):

Whenever we want to delete a single character, we press the button “backspace” from our keyboard. The same functionality can be achieved in C++ output with this escape sequence. For example:

```
cout<<"COMPUTER\bSCIENCE";
```

```
COMPUTESCIENCE
```

First of all, “COMPUTER” is printed and after that “\b” comes, which deletes the last character i.e. “R”. After that, “SCIENCE” is printed.

Single Quote (\’):

To insert a single quote in the output, this escape sequence is used. Look at the code written below:

```
cout<<"\’COMPUTER SCIENCE\’";
```

This code prints single quotes at the start and end of the string.

```
'COMPUTER SCIENCE'
```

Note: Similar to single quote escape sequence, you can use the double quote escape sequence (\”) to insert the double quote around any text in the output.

Carriage Return (\r):

This escape sequence moves the cursor at the beginning of current line. For example:

```
cout<<"COMPUTER\rSCIENCE";
```

First of all, “COMPUTER” is printed and after that “\r” comes which moves the cursor at the beginning of the line and “SCIENCE” is printed which overwrites the word written before.

```
SCIENCEER
```

Activity 3

Write a C++ program to print numbers from 1 to N (where N is the number of characters in your first name) on the same line with each pair of adjacent numbers separated by space in the following ways:

Suppose your full name is Ahmad Mujtaba hence, your first name is Ahmad which contains five characters.

1 2 3 4 5

Write a program to achieve the aforementioned task in following three ways.

a. Using one statement with one stream insertion operator.
b. Using N statements
c. Using one statement with N stream insertion operators.

Solution (a)

```
// Suppose name is Ahmad Mujtaba
#include <iostream>
using namespace std;
int main ()
{
        cout<<"1 2 \t 3 4 \t 5 \n" ;
        system("pause");
        return 0;
}
```

Solution (b)

```
// Suppose name is Ahmad Mujtaba
#include <iostream>
using namespace std;
int main ()
{
 cout<<"1" ;
 cout<<"  2" ;
```

```
  cout <<"\t 3" ;
  cout<<"  4" ;
  cout<<"\t 5 \n" ;
  system("pause");
  return 0;
}
```

Solution (c)

```
// Suppose name is Ahmad Mujtaba
#include <iostream>
using namespace std;
int main ()
{
  cout<<"1"  <<"  2"  <<"\t 3"  <<"  4"  <<"\t 5 \n" ;
  system ("PAUSE") ;
  return 0;
}
```

Activity 4

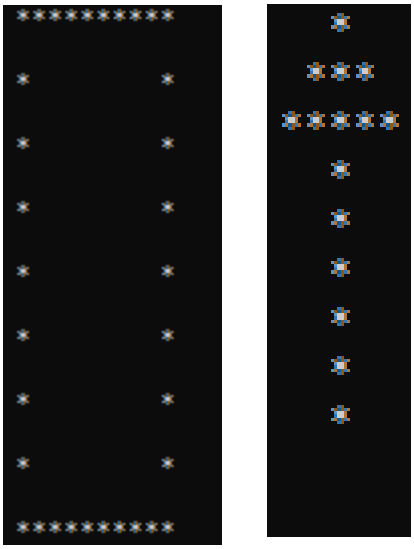
Write a program that displays your favorite poem.

Solution:

```
#include <iostream>
using namespace std;
int main ()
{
  cout<<"\t \t \t \t \t  poem \n" ;
  cout<<"\t \t \t I wandered lonely as a cloud \n" ;
  cout<<"\t \t \t That floats on high o'er vales and hills \n" ;
  cout<<"\t \t \t When all at once I saw a crowd \n" ;
  cout<<"\t \t \t A host, of golden daffodils \n" ;
  cout<<"\t \t \t Beside the lake, beneath the trees \n" ;
  cout<<"\t \t \t Fluttering and dancing in the breeze \n\n\n\n\n\n\n\n\n\n\n" ;
  system ("PAUSE") ;
  return 0;
}
```

Activity 5

Write a program to print a box, an arrow of asterisks using escape sequences:



Solution (box)

```
#include <iostream>
using namespace std;
int main ()
{
	cout<<" **********\n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n *\t  * \n" ;
	cout<<"\n **********\n" ;
	system("pause");
	return 0;
}
```

Solution (Arrow)

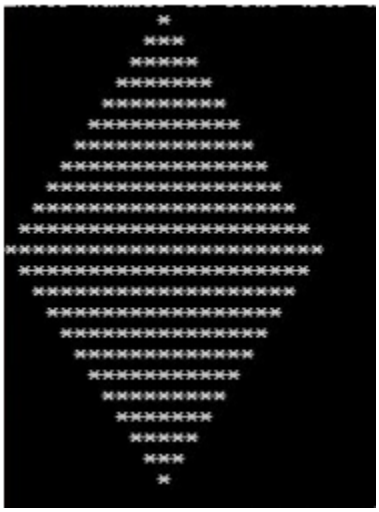
```
#include <iostream>
using namespace std;
int main ()
{
	cout<<" \t\t*" ;
	cout<<"\n\t      ***" ;
	cout<<"\n\t     *****" ;
	cout<<"\n \t\t*" ;
```

```
        cout<<"\n \t\t*" ;
        cout<<"\n \t\t*" ;
        cout<<"\n \t\t*" ;
        cout<<"\n \t\t*" ;
        cout<<"\n \t\t*" ;
        cout<<"\n\n\n\n\n\n\n\n" ;
        system("pause");
        return 0;
}
```

Exercises

Question 1:

Write a program to print a diamond using escape sequences.



Question 2:

Write a program to print your first name with stars (*) using escape sequences.