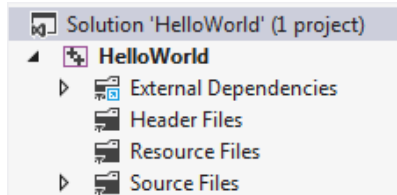


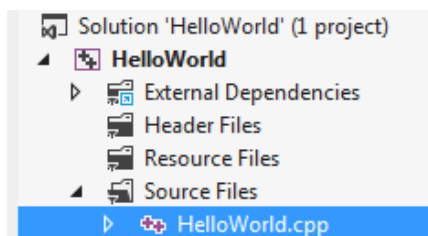
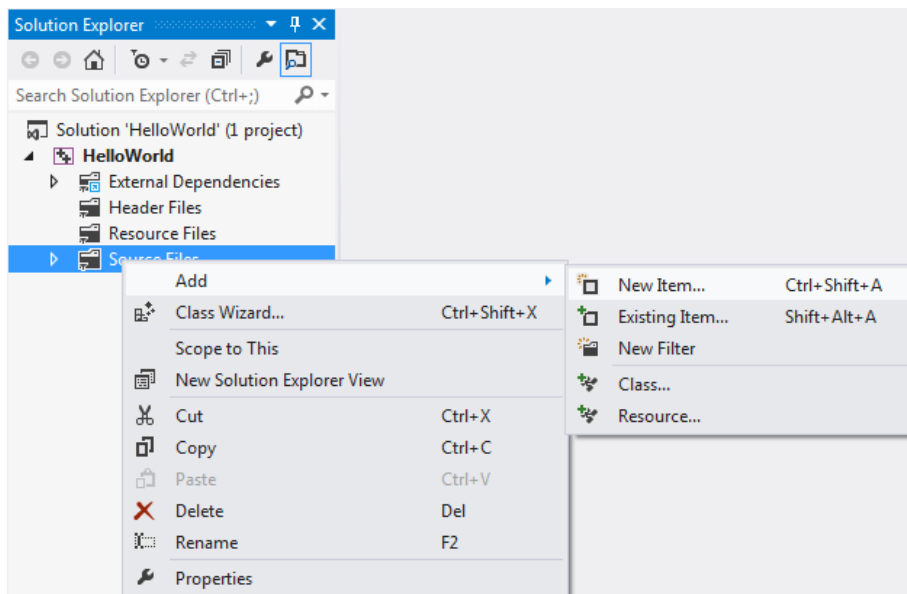
Programming Process

Develop an algorithm: Think about the problem before coding. Create a flowchart, storyboard, or pseudo code to represent a solution to the problem.

Create Project: Create a new project in Visual Studio 2012. The Visual Studio environment is known as our **IDE** (Integrated Development Environment). In Visual Studio, when you create a project, it creates a folder structure. The following is an example of a folder structure for a HelloWorld project created in Visual Studio.



Code: In order to type the C++ code, we first need to add a **cpp file** to our Source Files folder. You can do this by right clicking on the **Source Files folder**, selecting **Add**, then **New Item...**



Once you have the cpp file, you can begin typing your C++ code (source code).

Compile: Before compilation the preprocessor searches for any directives (lines with hash mark #). The directives tell the preprocessor to modify the source code. The compiler then checks the preprocessed source code and translating this into machine code. If the program is free of syntax errors, then the translated machine code is stored in an object file. The object code is then joined with the C++ library of pre-existing code to create an executable file. This library of pre-existing code has been written to save the programmer time.

Run: Show the program results.

Program Errors

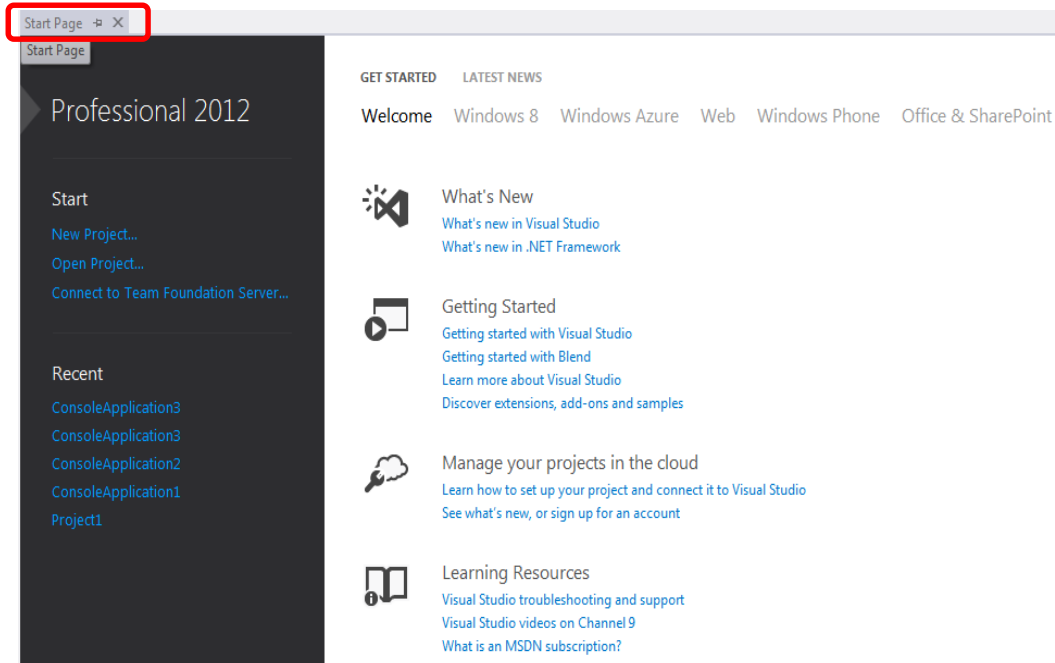
There are 3 different types of programming errors: compiler, run-time, and logic errors.

- **Syntax errors** are caused when the user writes code that is not understood by the compiler. A syntax error can be caused by incorrect capitalization or spelling mistakes.
- **Run-time errors** are caused by invalid data. Run-time errors do not affect the compilation of your program thus the program will compile and execute, but it may crash or hang after execution. If you try to divide 12 by 0 you would get a run-time error because you cannot divide by 0.
- **Logic errors** are caused by mistakes that do not defy the rules of the language and do not crash or hang the program, but instead yield incorrect results. The user may not understand the problem that the program is trying to solve and therefore uses the wrong equation, wrong strategy, etc. An example of a logic error would be moving left instead of right.

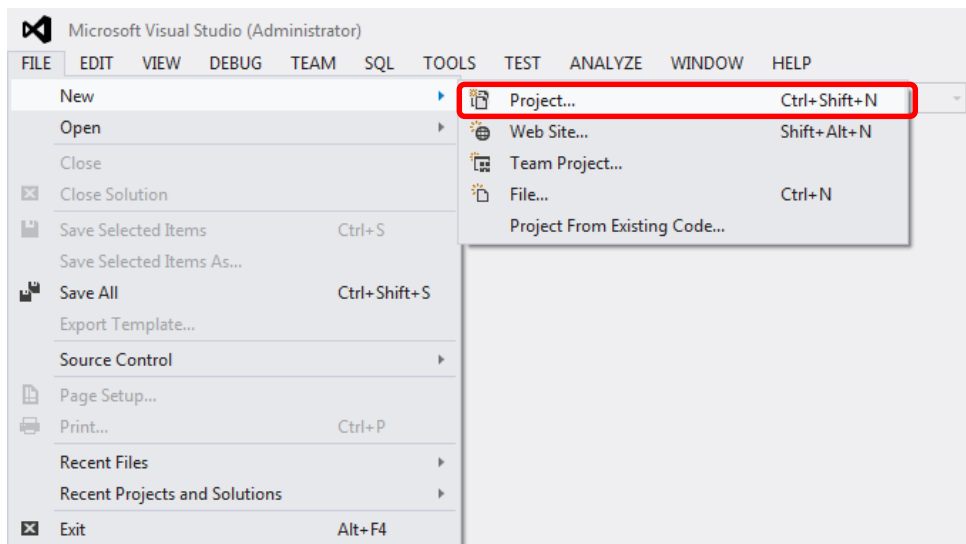
Exercises

Hello World Exercise

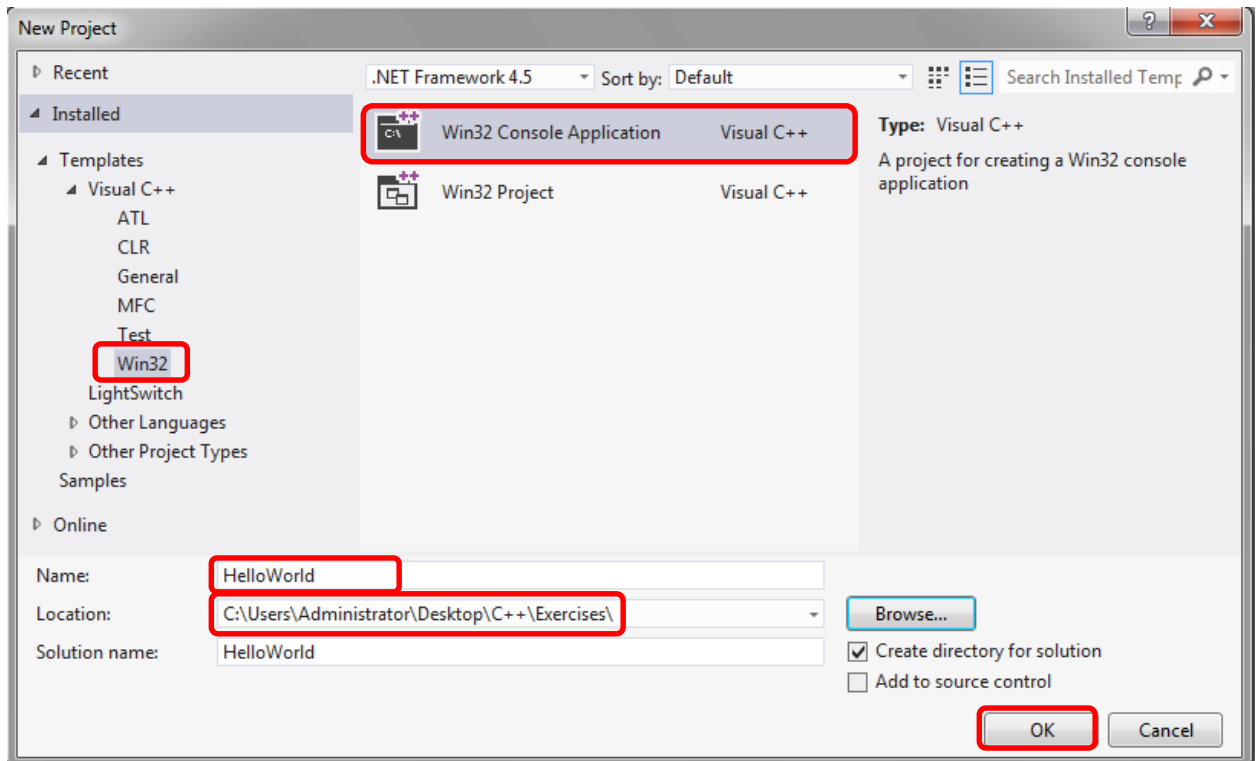
1. Open up the **Visual Studio 2012** environment.
2. You can close the **Start Page**. The tutorials provided in the Start Page can be confusing for a first timer.



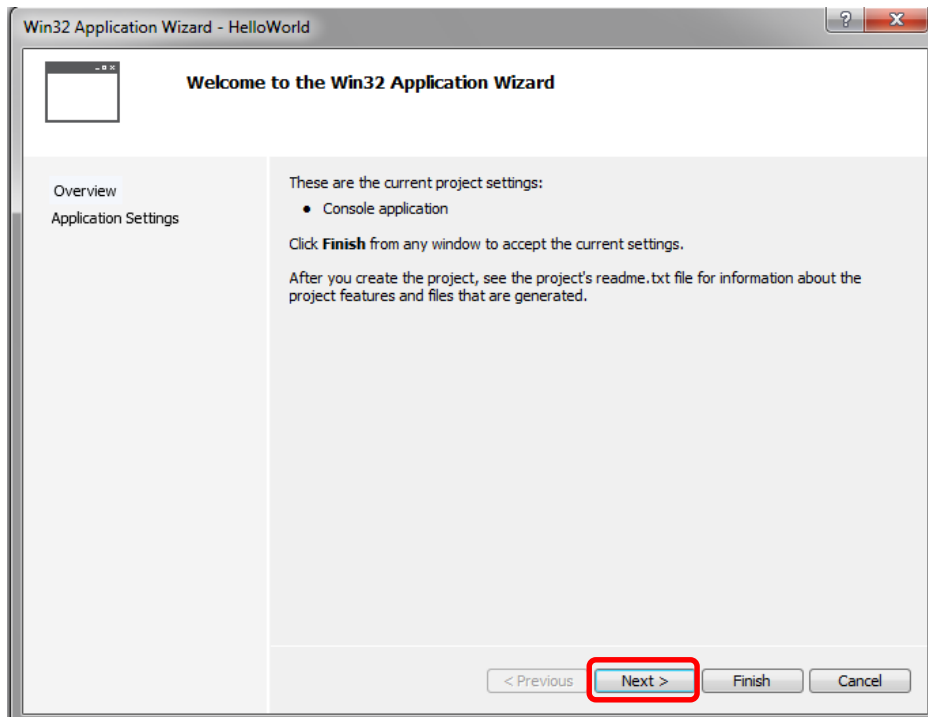
3. Select the **File** menu and then choose **Project**.



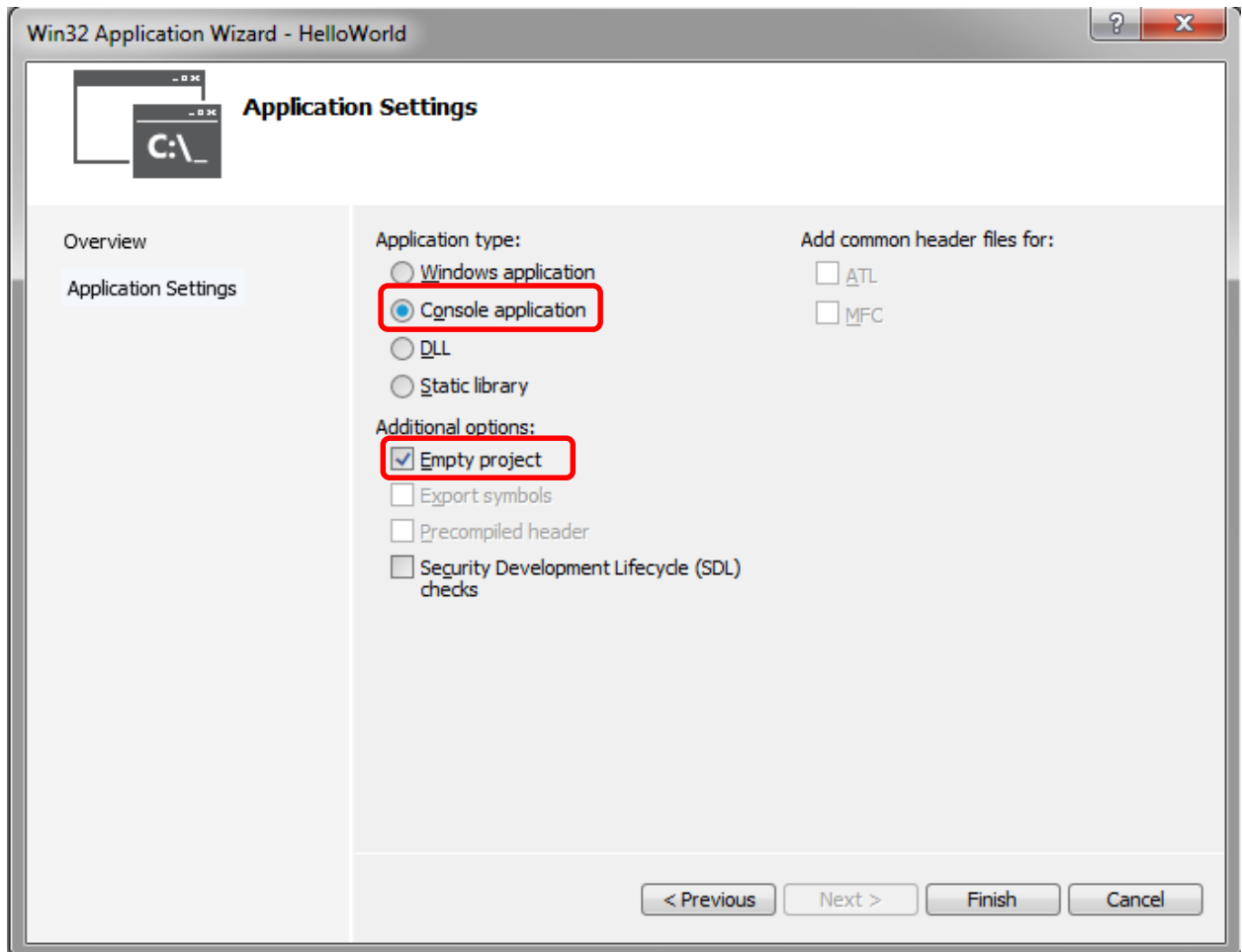
4. Select **Win32 Console Application**, name this project **HelloWorld**, change the location of where you are saving your project, click **OK**.



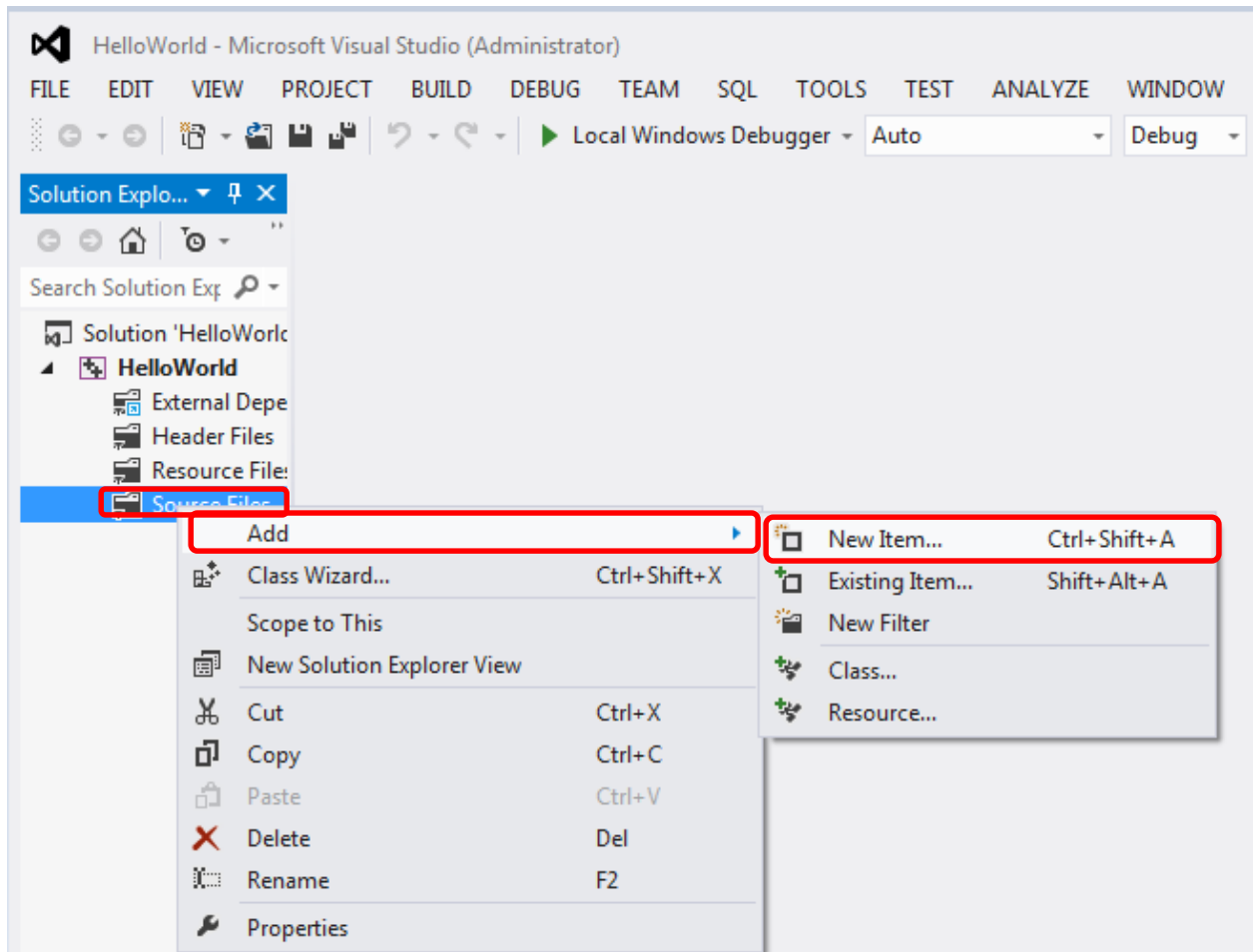
5. Click **Next**



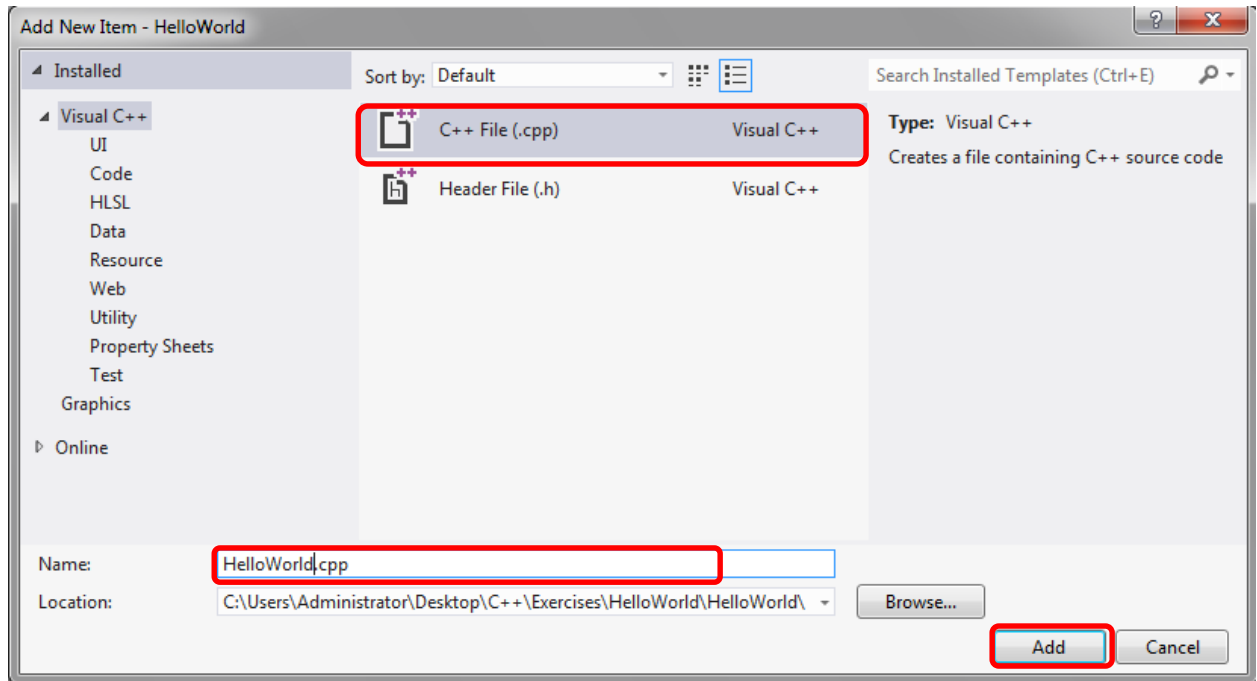
6. Make sure that you have the **following options** selected and click **Finish**.



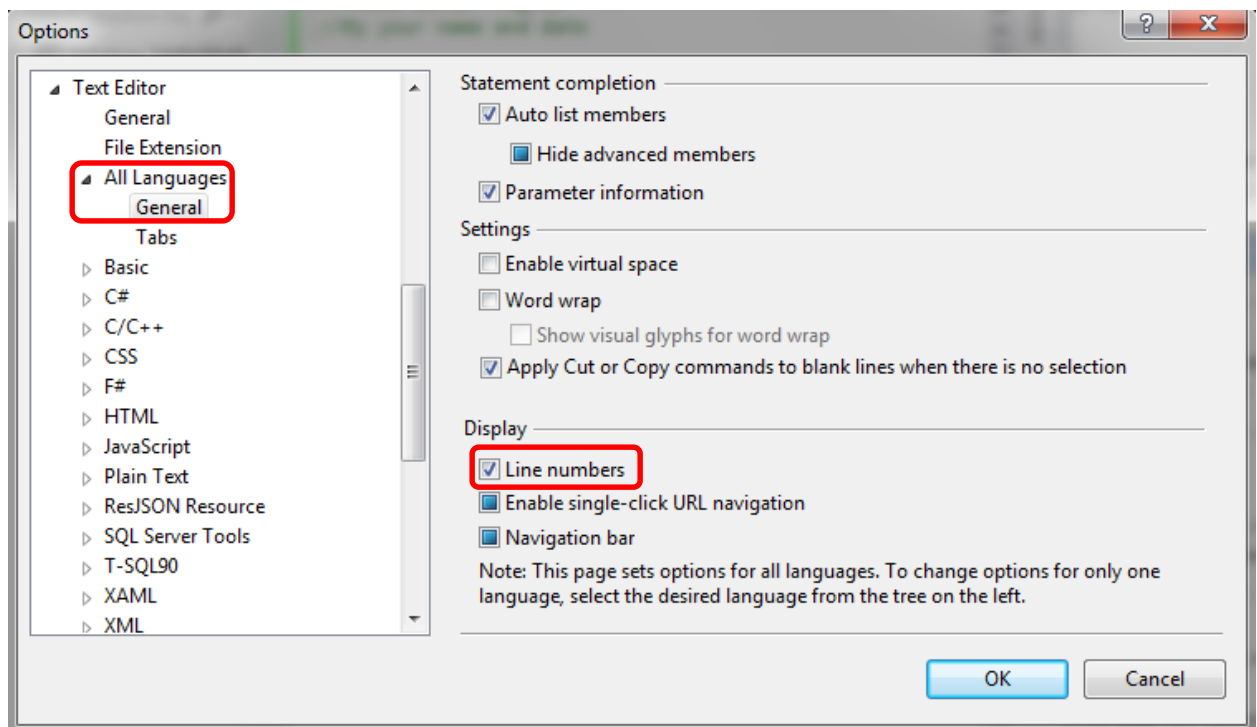
6. Right click on the **Source Files** folder under the **Solution Explorer**. Choose **Add**, then **New Item...**



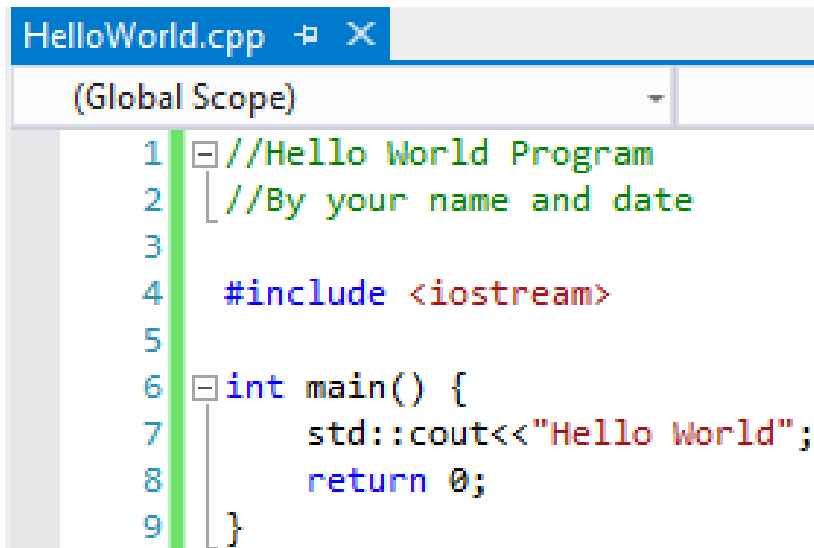
7. Select **C++ File (.cpp)** and rename the file **HelloWorld.cpp** and click **Add** as shown below:



8. Select **Tools** from the menu and then **Options**. Under **All Languages** and **General**, check the **Line numbers** box.



9. Type in the following C++ program. Be careful to make your program look exactly as shown below. Try to keep your statements on the same lines as those shown below and also try to use the same approximate indentation to make your program more understandable.

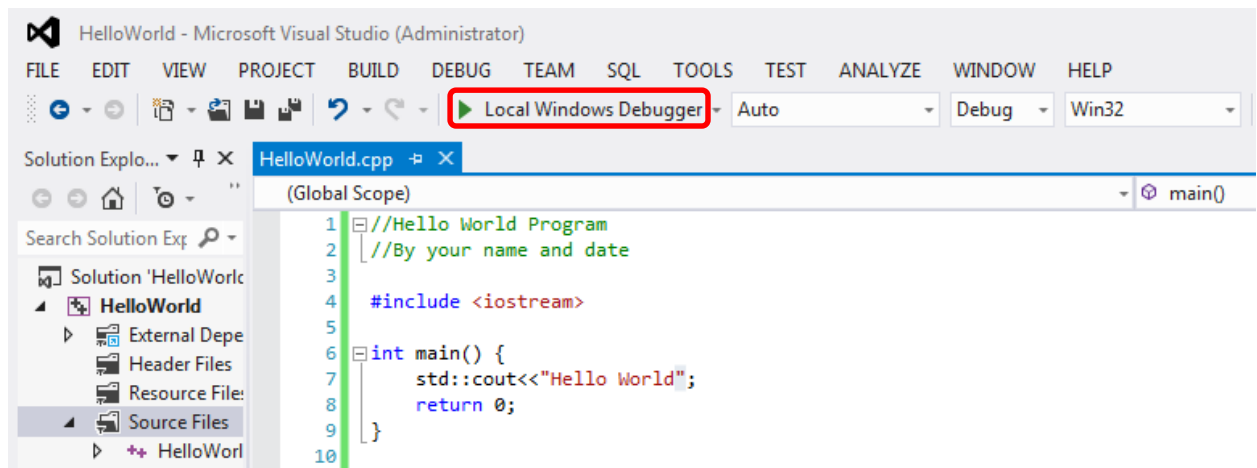


```
1 //Hello World Program
2 //By your name and date
3
4 #include <iostream>
5
6 int main() {
7     std::cout<<"Hello World";
8     return 0;
9 }
```

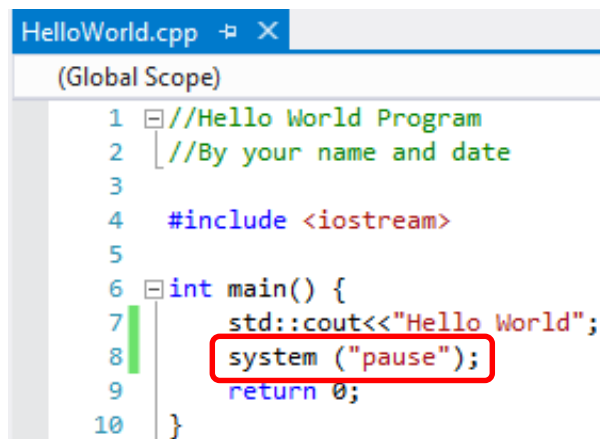
What does the above program do? You will not understand everything about this program YET. However, here is a brief explanation line by line:

- 1-2) Lines 1-2 are known as comments. Comments are used to document code so that other people reading our code can understand our logic. Comments are useful for adding extra information to our programs that we don't necessarily want to show up in the output of our program such as: author, date, program description, etc. Also, it is a good idea to comment your programs extensively when you are just starting out so that you have well-documented examples.
- 3) Blank line for readability purposes. Does nothing. (not necessary)
- 4) This line is known as a directives and are to be handled by the preprocessor (before compilation). This line instructs the preprocessor to include the header `iostream` which is used to perform standard input and output.
- 5) Blank line for readability purposes. Does nothing. (not necessary)
- 6) This is the start of a function named "main". Functions have a type (in this case "int"), a name (in this case "main"), parameters needed for the function are enclosed within parathesis (in this case none), and the statements within the function are enclosed within curly braces (one to begin and one to end).
- 7) The statement of `std::cout<<"Hello World";` identifies the standard character output device (screen) and inserts the words in quotes into this output stream to be printed to the screen.
- 8) Since functions must return a value, it is good practice to add a return statement that returns 0 to end this function.
- 9) A right curly brace to end the main function.

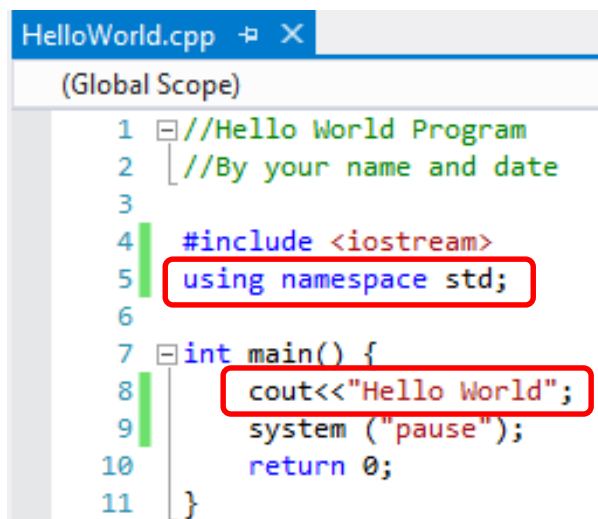
10. This Java program needs compiled. Compiling a program will have the computer look at each line of your program for syntax errors such as typos, mispunctuation, etc. To compile your program, click on green run arrow. The compiler will check this file for syntax errors and let you know on what lines you made errors. If you have errors, correct your typos on the top of the screen and compile again. If you do not have any errors, you should see a black console window open with the text Hello World and then quickly close.



11. To force the console to stay on the screen so that we can see the results of our program, you will need to add the following line of code after your **cout** statement. This line of code causes the console to stay on the screen.

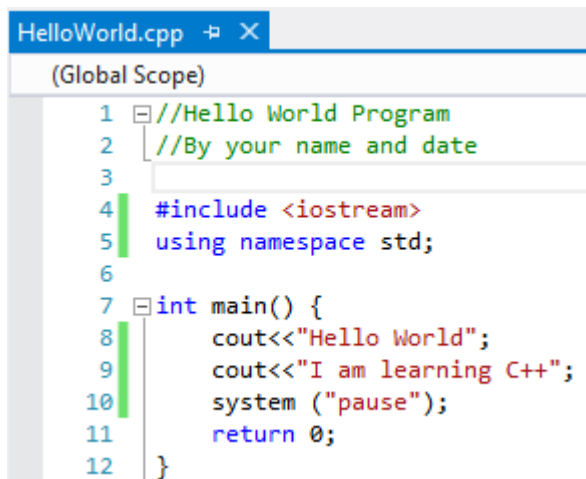


12. Instead of having to refer to all elements in the std namespace before using them, you can introduce this namespace at the top of your program as shown below:



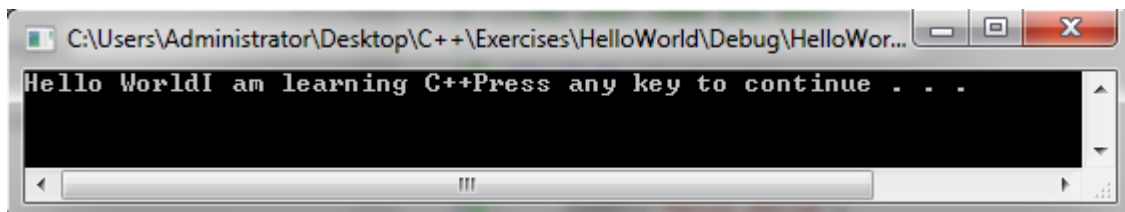
```
1 //Hello World Program
2 //By your name and date
3
4 #include <iostream>
5 using namespace std;
6
7 int main() {
8     cout<<"Hello World";
9     system ("pause");
10    return 0;
11 }
```

13. If we wanted to print out another line, we could add another **cout** statement as shown below:



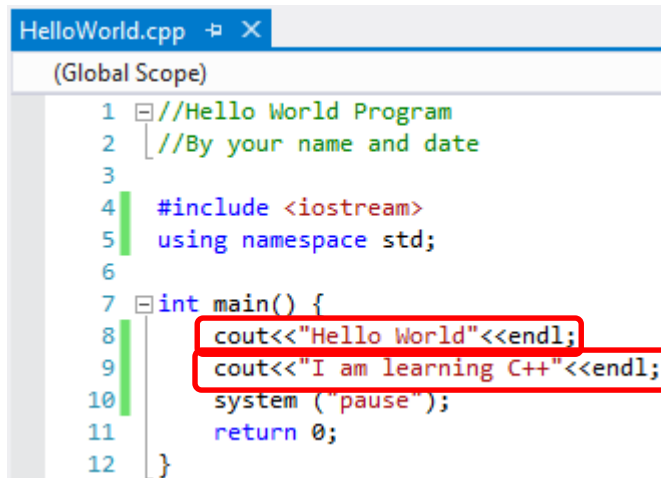
```
1 //Hello World Program
2 //By your name and date
3
4 #include <iostream>
5 using namespace std;
6
7 int main() {
8     cout<<"Hello World";
9     cout<<"I am learning C++";
10    system ("pause");
11    return 0;
12 }
```

The output should look as follows:



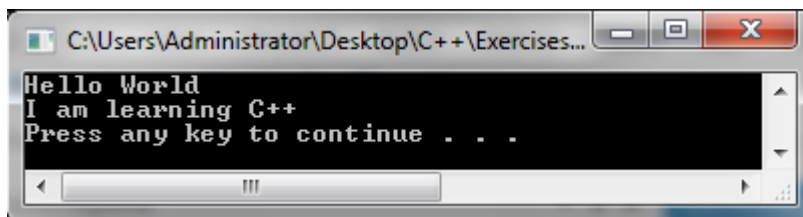
```
C:\Users\Administrator\Desktop\C++\Exercises\HelloWorld\Debug\HelloWor...
Hello WorldI am learning C++Press any key to continue . . .
```

14. You may notice that both of the output lines are joined onto one line which makes it difficult to read. If you want the output on separate lines, then you can add the following to the end of the line:



```
1 //Hello World Program
2 //By your name and date
3
4 #include <iostream>
5 using namespace std;
6
7 int main() {
8     cout<<"Hello World"<<endl;
9     cout<<"I am learning C++"<<endl;
10    system ("pause");
11    return 0;
12 }
```

Which would yield the following output:



```
C:\Users\Administrator\Desktop\C++\Exercises...
Hello World
I am learning C++
Press any key to continue . . .
```

There are a list of escape sequences that can be used to adjust output.

Escape Sequence	Name	Description
\n	Newline	Skip to a new line for output
\t	Horizontal tab	Skip to new tab stop
\\	Backslash	Print a backslash
\'	Single quote	Print a single quotation mark
\"	Double quote	Print a double quotation mark

Example:

The following code segments produce the same results:

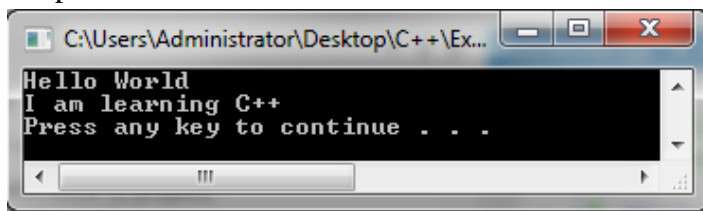
```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hello World"<<endl;
    cout<<"I am learning C++"<<endl;
    system ("pause");
    return 0;
}
```

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

int main() {
    cout<<"Hello World\nI am learning C++\n";
    system ("pause");
    return 0;
}
```

Output:



Example:

If you wanted to put the “I am learning C++” in quotations, you would write the code as follows:

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

int main() {
    cout<<"Hello World\n\"I am learning C++\"\n";
    system ("pause");
    return 0;
}
```

Output:

