EEE102 C++ Programming and Software Engineering II

Lab Practice 1

Using Microsoft Visual C++ for C++ Programmes

Notice:

- Read though the scripts and understand how a simple C++ program can be created and run in Microsoft Visual Studio.
- Ask the demonstrators for help if you have any difficulties.
- No submission for this week's lab.

1. Header files

C++ is a huge language (not necessarily complicated). It is a set of instructions that you give to the computer to perform an operation and give you a result. The language is so huge that it uses various sets of instructions from different parts to do its work. Some of these instructions come in forms of files that you simply "put" in your program. These instructions or files are in the form of libraries, and that's what we will call them, libraries. To make your job easier, some of these libraries have already been written for you so that as you include them in your program, you already have a good foundation to continue your construction. Yet, some of these libraries have their limitation, which means you will expand them by writing or including your own libraries.

As noted already, there are libraries previously written for you. One of them asks the computer to receive keyboard strokes from you the user (when you press a key) and another asks the machine (the computer performing some operations) to give back a result. These libraries are files that you place at the beginning of your program as if you were telling the computer to receive its preliminary instructions from another program before expanding on yours. The libraries are called ***header files***. An example would be **`iostream`**, or **`stdlib.h`**. As you see, they could have any name, with or without the extension **`.h`**; when you start creating your own libraries, you will give your files appropriate and recognizable names, and always defines as **`xxx.h`**. The libraries are stored in the *include* folder under the installation directory, such as:
D:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\

The first library we will be interested in is called the **`iostream`** library. It contains the objects (functions) making the computer to display things on the monitor or accepting input from the keyboard.

Since this is a computer language, the computer will follow particular instructions to perform appropriately, which will make this language distinct from the everyday

languages. C++ has some words it treats specially and some that will completely depend on you the programmer. For example, the word "include" could be a special word used by C++ or a regular you want to use in your program. In this particular situation, if you want the compiler to "know" that the word "include" means, "I want to include the following library", you will have to append a special sign to it. The pound sign "#" will do just that. Therefore, to include a library, you need to write a line as the following:

```
#include <iostream>
```

which is usually called a pre-processor line.

There are usually two kinds of libraries or files you will use in your programs: libraries that came with the compiler, and those that you write. The compiler likes to know which ones are which.

To include your own library, you would enclose it between double quotes "", like this

```
#include "book.h"
```

When you include a library that came with the compiler, you enclose it between angle brackets < and > as follows:

```
#include <iostream>
```

Following this same technique, you can add as many libraries as you see fit. Before adding a file, you will need to know what that file is and why you need it. This will mostly depend on your application. For example, you can include the I/O library inherited from C I/O file `stdio.h` like this:

```
#include <cstdio>
```

** The older version of C++ using the similar header files like C, always having the extension ".h". However, the libraries of C++ were standardised in 1998 to improve the program robustness. They are compared as follows:

	C	Pre-standard C++	Standard C++
Standard I/O	`stdio.h`	`stdio.h`	`cstdio`
	--	`iostream.h`	`iostream`
Character string	`string.h`	`cstring.h`	`cstring`
	--	`string.h`	`string`
Math operations	`math.h`	`math.h`	`cmath`
Compatibility	/	Old version compiler: VC6.0	New version compiler: MS 2005, 2008, 2010
Usage suggestion	/	**No**	**Yes**

Since the components (objects) of the standard libraries are a part of the standard namespace **std**, to use the components, programmer needs to claim the using of the std namespace as follows:

```
using namespace std;
```

Therefore, the beginning of a simple program involving the standard screen output could be:

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

2. C++ Instructions

C++ works by giving (separate) instructions to the computer. These instructions can be treated as assignments. In C++, such an assignment will be called a function. The primary function used in C++ is called `main`. To distinguish a function from the other types of things you will be using in your programs, a function's name is followed by an opening and a closing parenthesis. For example, the main function will always be written at least as `main()`.
When a program is written and you ask the computer to "execute" it, the first thing the compiler will look for is the `main()` function. This means that every C++ program should have the `main()` function. Because a function is an assignment, in order to perform its job, a function has a body; this is where the behaviour (assignment) of the function would be "described". The body of a function starts with an opening curly bracket "`{`" and closes with a closing curly bracket "`}`". Everything in between belongs to, or is part of, the function. Therefore, the `main()` function can be written as

```
int main(void)
{
    ……
    ……
    return 0;
}
```

The return type of `main()` must always be an `int`, this allows a return code to be passed to the invoker. For the reasons for why not `void main()` and `main()`, please reference the following link:
http://faq.cprogramming.com/cgi-bin/smartfaq.cgi?id=1043284376&answer=1044841143

As we learned that we should (must) always include the libraries that we would need, our program now would include `main()`. Whenever you create a program, it is important to

isolate any inclusion of a library on its own line. Here is an example:

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

int main(void)
{
   ……
   return 0;
}
```

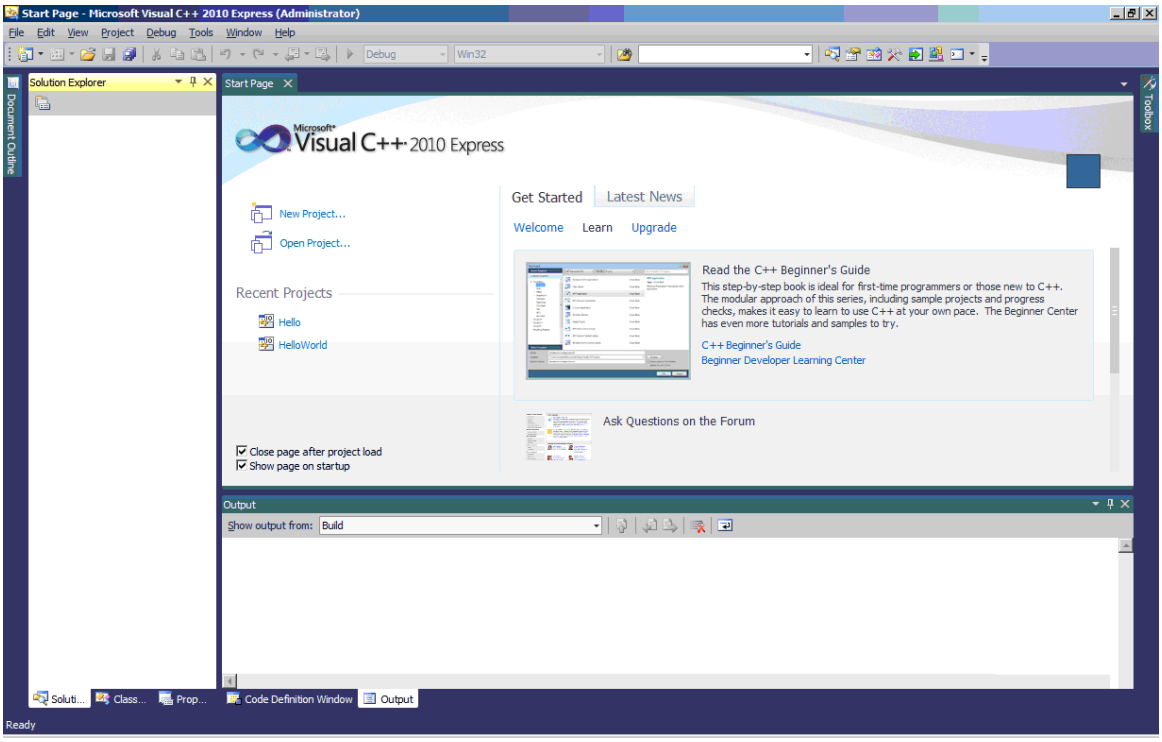
3. The first program using MVC++ 2010: “Hello World”

C++ is a very universal language; it can be used to write programs for Linux, MS Windows, Macintosh, BeOS, etc. C++ is very powerful and can be used to create other compilers or languages; it can also be used to write an operating system. This means that you can use C++ to create/write your own computer language. You can also use C++ to create/write your own compiler; this means that, using C++, you can create your own implementation of C++, Pascal, Basic, Perl, or any other existing or non-existing language.

There are many products you can use to create a program in C++. Before a program is made available, it is called a project because you are working on it. Although in the beginning you will usually be working alone, most programs involve a lot of people. That is why during the development of a program or software product, it is called a project. Each one of the available environments provides its own technique(s) of creating a C++ program or working on a C++ project. Therefore, the person who, or the company that, made the environment available to you must tell you how to use that environment (it is neither your responsibility, nor the C++ Standard job to tell you how to create a program or how to start a project).

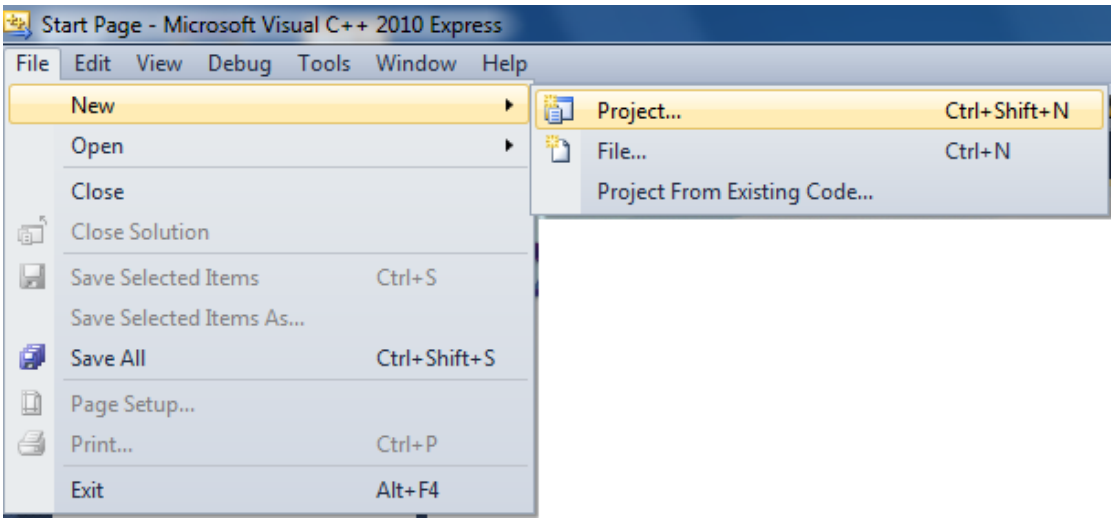
The programs we will be creating in the first half of this module are called console applications in MicroSoft Windows. The compiler we are using is Microsoft Visual C++ 2010 Express, which can be found in the program list of the start menu.

1. Open the Microsoft Visual C++ 2010 Express. You will get a window like this:

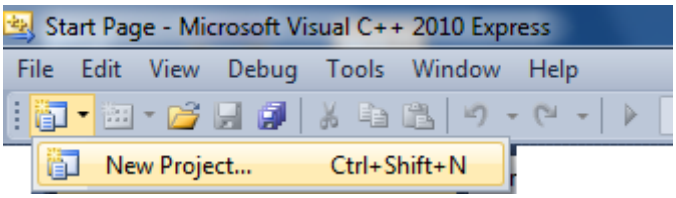


In the following we are using the ***Win32 Application Wizard*** to create an empty project.

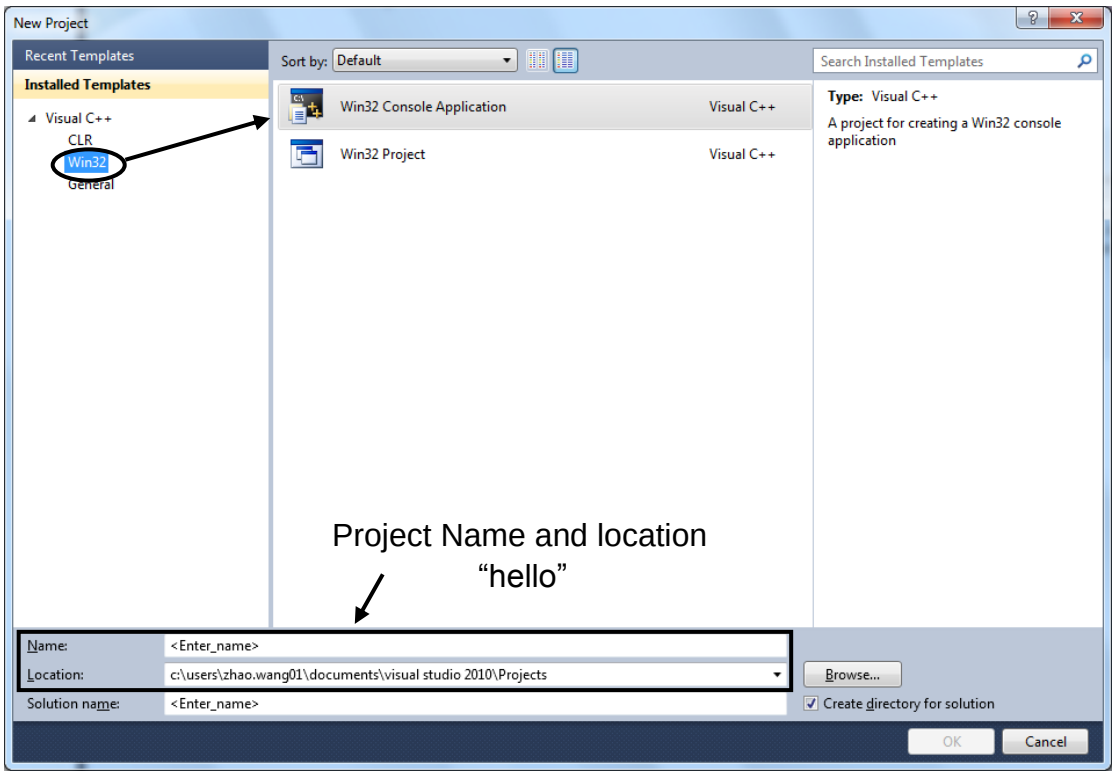
2. To create a **New Project**.
   There are two ways to create a new project:
   a) File -> New -> Project…



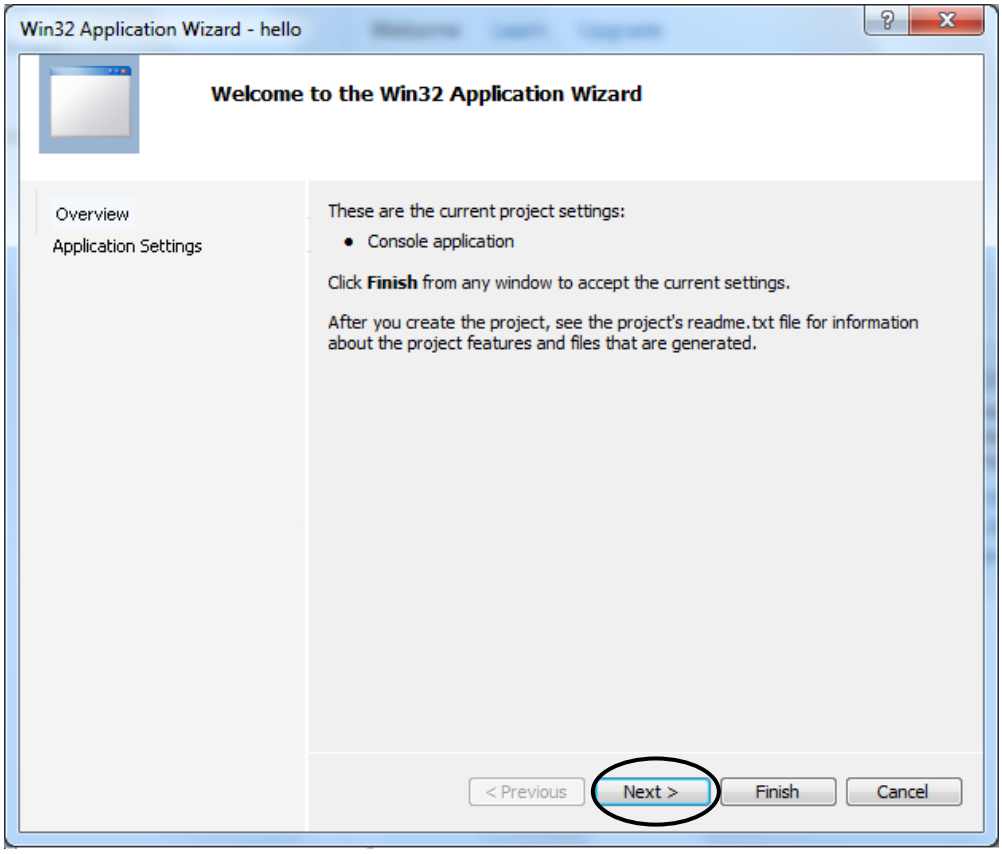
   b) Using the shortcut  on the toolbar:

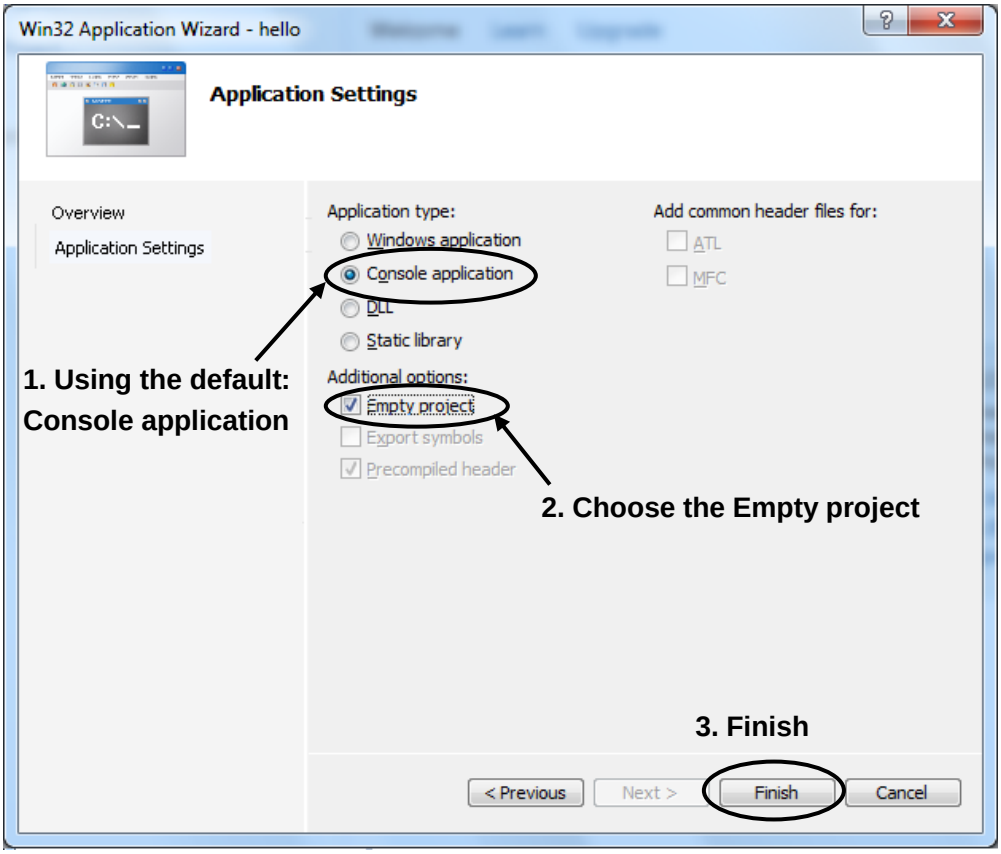


3. Choose the project type: **Win32 -> Win32 Console Applications**
   Assign the **location** and the **name** of the program (project).

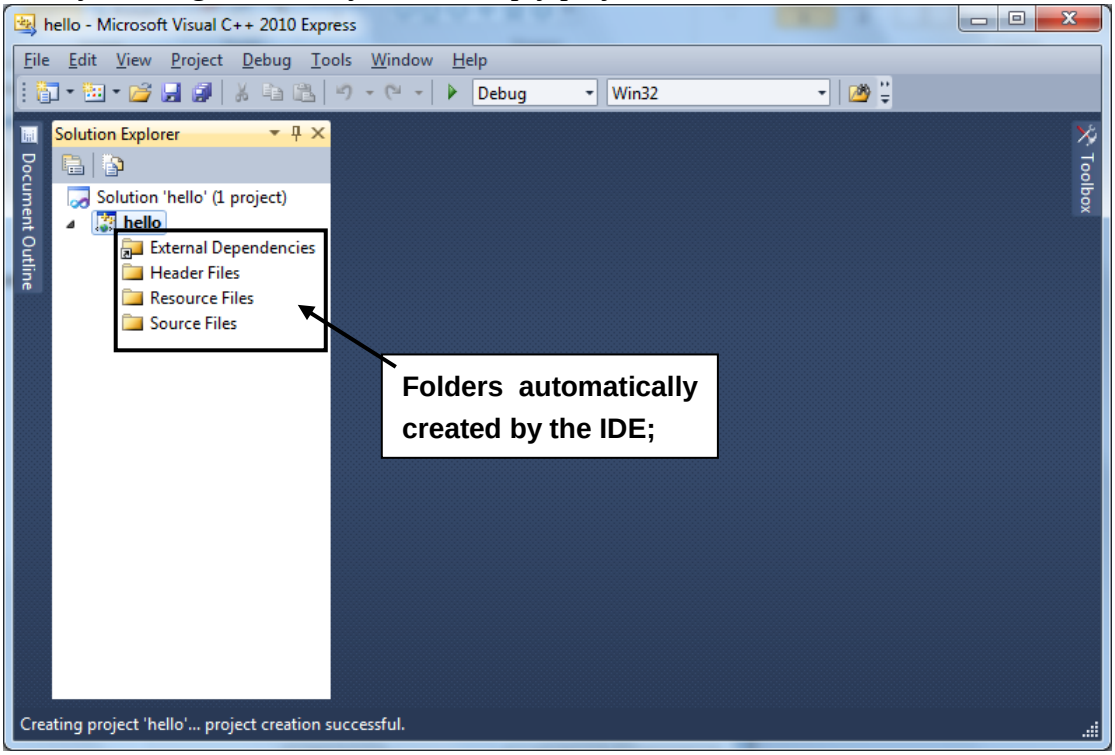


Then the ***Win32 Application Wizard*** will be opened to help you create the project.

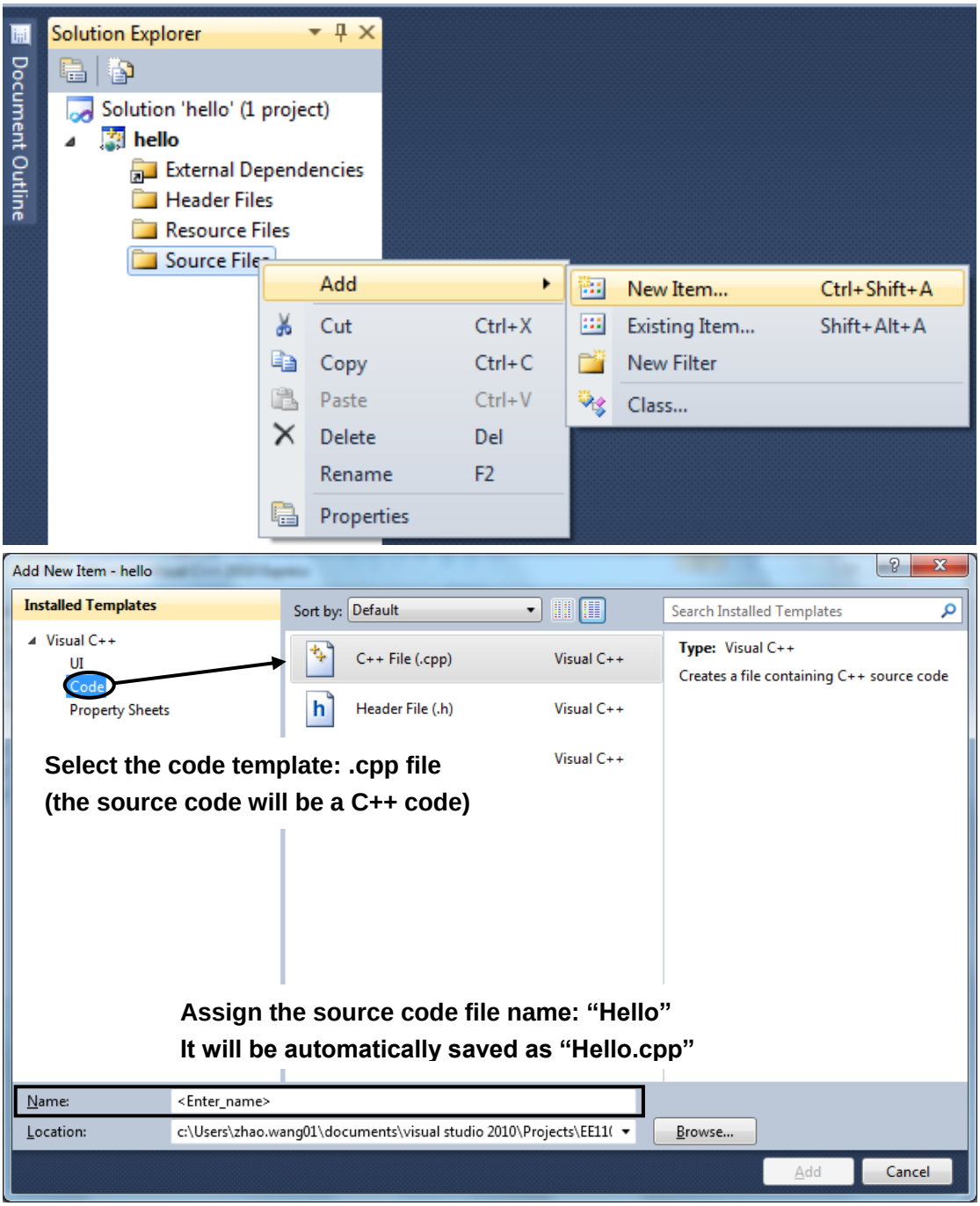




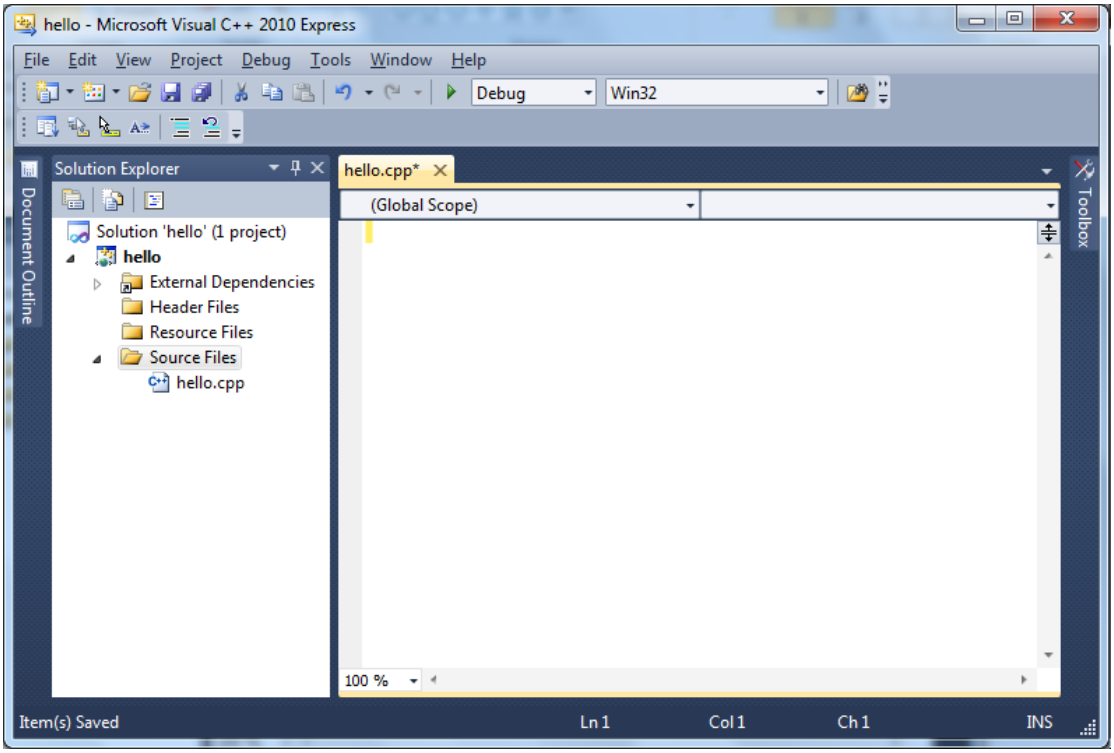
Then you will get the newly created empty project:



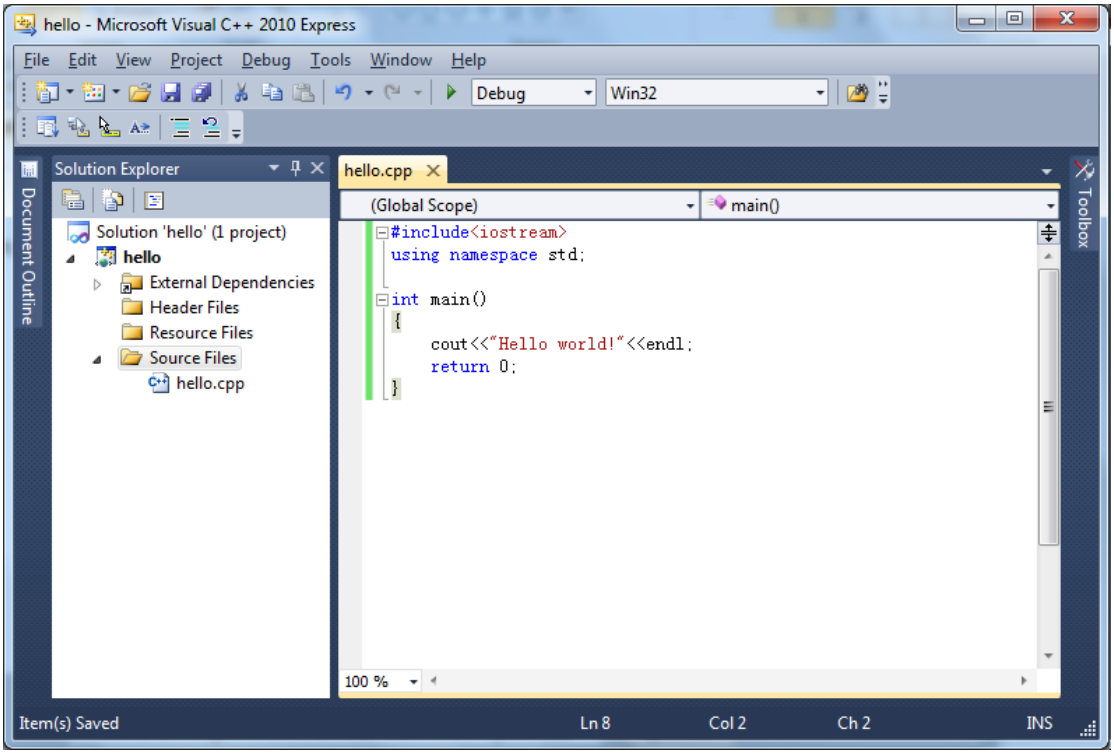
4. Add a piece of C++ source code to the project
   Right-click the folder “Source Files”, select **Add -> New Item…**



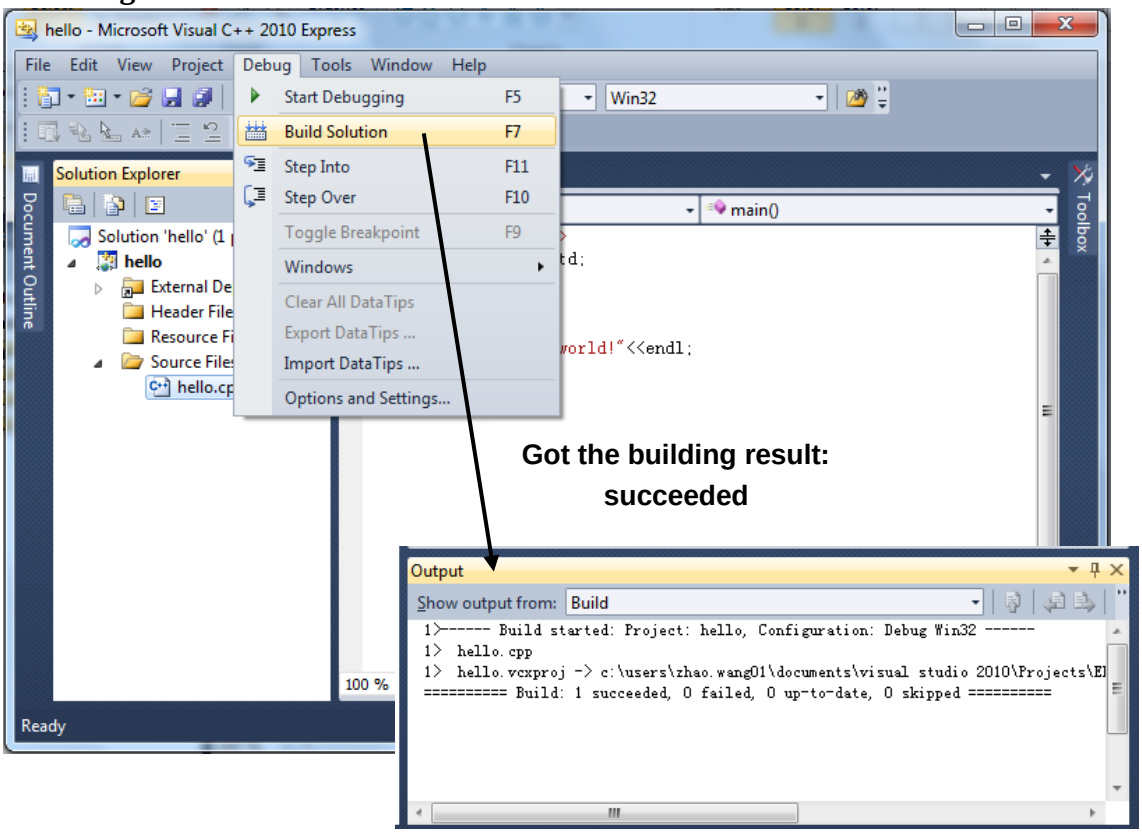
Then the text editor for source code will be opened:



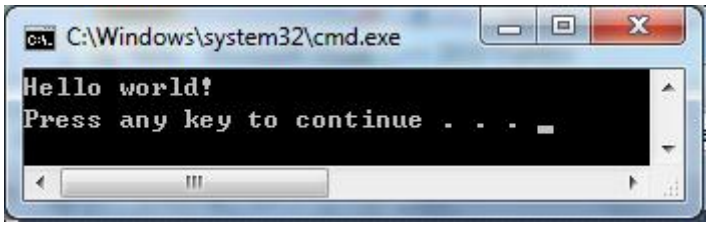
5. Input your source code and save:



6. Compile, build and run the program
   The "Build" operation includes the "compiling" and "building", select from **debug -> Build Solution**



Then run the program (without debug) by shortcut: **ctrl-F5**



Exercise 1: Try to execute this program. What is the screen display?

```
// example 1.1
#include <iostream>
using namespace std;
int main(void)
{   // Opening Brace
    int counter = 0; // Note variables defined at the the begining of main
    cout << "This is an example program\n"; // Prints header message
    while(counter < 5)
    {
        cout << "Current value of the counter: " << counter << "\n";
```

```
        counter++;     // Increments counter
    }
    cout << "This is the end of the example program";
    return 0;
}   // Closing Brace
```

4. Data Output in C++

There are various ways data get in your program. The first means of entering data in a C++ program is by typing it from the keyboard. Another way would be to instruct the program to receive data from another program then process it. A program can also receive its data or part of it from other hardware devices such as a CD-ROM, a DVD-ROM, a modem, a video camera, etc.

To display stuff on the monitor, C++ uses operators. The operator used to display something on the screen is the `cout` operator (pronounce C - out) (actually, `cout` is a class and not an operator, for the time being we call it an operator). The `cout` word is followed by the extraction operator <<, then some simple rules to display anything. For example, to display a word or sentence, you include it in double quotes " and ".

While you are giving these instructions, you type them in your program. Each C++ instruction is terminated by a semi-colon "`;`".

We have already seen that a program is a set of instructions you give to the computer. These instructions are given inside of functions. This means that an instruction is part of a function. The thing we want to display in our program will be performed by the `main()` function. In other words, the instruction to display something will be given in main.

Here is an example:

```
// example 1.2
#include <iostream>
using namespace std;
int main(void)
{
    cout << "Computer Programming With C++ \n";
    return 0;
}
```

**** Comments**

Documenting your work is very important in the area of programming. Sometimes you might get hired by a company and the first assignment is to "read" a program that was left by some developer before you. The most basic documentation you will (have to) perform is to put comments as much as you can. Comments help you and other people who read

your code to figure out what you were doing.

Comments are not read by the compiler while executing your program. That means you write them in everyday conversation. There are two usual ways of including comments in your program. To comment the content of one line, you start it with double forward slashes like this `//`. An alternative is to start the beginning of the commented paragraph with `/*` and end the commented section with `*/`.

Here is an example:

```
// example 1.3
// Include the first library
#include <iostream.h>
using namespace std;
main()
{
    // Here is a simple sentence
    cout << "Hi, this is my program!";
}
// The end of my program
```

Note: Examples 1.3, 1.4, 1.5 are not guaranteed to be correct. Please observe carefully and find out the problems of them. Try to compile, build and run all the examples in MS VC++, and do some modifications if they are needed. Google for explanation if needed!

Here is another commented version of our program:

```
// example 1.4
// Include the iostream library
#include <iostream.h>
using namespace std;
void main()
{
    /* Here is a simple sentence that I want to display
       when the program starts. It doesn't do much.
       I am planning to do more stuff in the future. */
    cout << "Hi, this is my program!";
}
// The end of my program
```

**** Escape sequence**

C++ allows you to use escape sequences to make the computer do something such as getting to the next line, performing a TAB action, producing a quick sound (beep). When you write your program, you include the (right) escape sequence and C++ will take care of the rest. Various escape sequences behave differently, but they are always written with

a backslash (\) followed by a particular letter or character. Computers and compilers that follow the American Standard Code Information Interchange (ASCII) (pronounce asky) recognize these escape sequences that have a corresponding ASCII value.
Here are the mostly used escape sequences:

Escape Sequence	Name	ASCII value	Description
`\a`	Bell (alert)	`007`	Makes a sound from the computer
`\b`	Backspace	`008`	Takes the cursor back
`\t`	Horizontal Tab	`009`	Takes the cursor to the next tab stop
`\n`	New line	`010`	Takes the cursor to the beginning of the next line
`\r`	Carriage return	`012`	Takes the cursor to the beginning of the current line
`\"`	Double Quote	`034`	Displays a quotation mark (")
`\'`	Apostrophe	`039`	Displays an apostrophe (')
`\?`	Question mark	`063`	Displays a question mark
`\\`	Backslash	`092`	Displays a backslash (\)
`\0`	Null	`000`	Displays a null character

Whenever you use any of these characters, if they are used inside of a sentence, then you can just include them; if they are stand-alone, and then include them in double quotes.
The most popular escape sequences you will be using are the new line (`\n`) and the horizontal tab (`\t`). The new line can also be substituted with `endl` and the result will be the same.

Here is a simple example of use of the escape sequences:

```
//example 1.5
#include <iostream>
using namespace std;

int main()
{
    // Create some space with double \n
    cout << "Here are some companies and their products\n\n";
    cout << "Company\t\tSoftware\n";
    cout << "Borland\t\tC++ Builder\n";
    cout << "Microsoft\tVisual C++\n";
    cout << "Corel\t\tWord Perfect\n";

    // Another list
    cout << "\nHere are a few countries and their continents\n\n";
    cout << "Continent\tCountry\tCapital\n";
    cout << "America\t\tMexico\tMexico City\n";
    cout << "Europe\t\tItaly\tRome\n";
    cout << "Africa\t\tLybia\tTripoli\n";
    cout << "Asia\t\tIndia\tNew Delhi\n";
```

```
    cout << "\nPress any key to continue...";
    return 0;
}
```

The running result is: