Visual C++ 2010 Tutorial

Fall, 2011

Table of Contents

Program Development with Microsoft Visual C++ 2010

Introduction

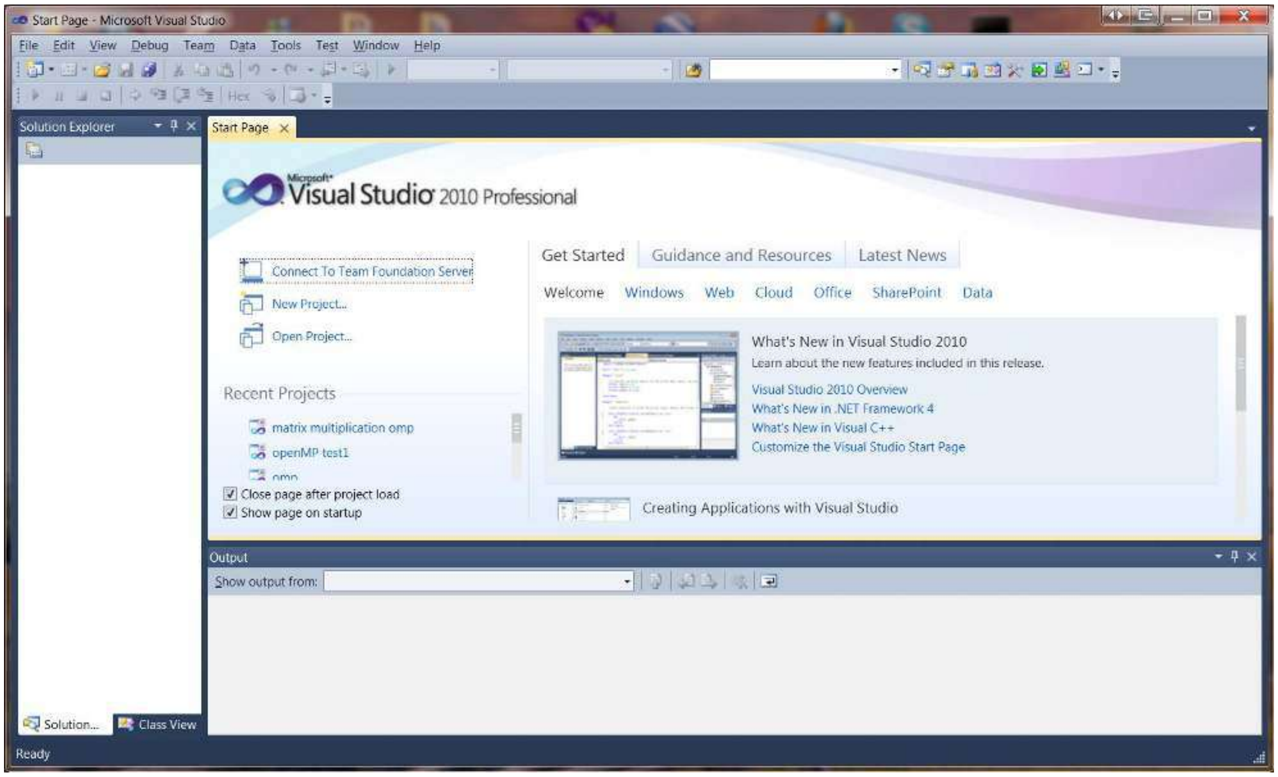
This tutorial is written to help those who are new to Visual C++. It introduces the **Integrated Development Environment** (**IDE**) of Microsoft Visual C++ 2010 and shows how to enter, edit, save, retrieve, compile, link, and run a C++ program in such an environment. It also shows a simple way to print the source code and the program output.

Two simple C++ projects, one single-file and the other multi-file are used as demo programs so that students can focus on Visual C++ 2010 IDE instead of getting distracted by language features. For the convenience of the reader, source code for both programs is listed in the Appendix.

If you have any comments and/or suggestions that might help improve the effectiveness of this tutorial, please forward them to either Dr. E. Hu at hue@wpunj.edu or Marvin Kiss at kissm@wpunj.edu.

Single-file Project: The *Hello World* Program

Step 1: Launch the MS Visual C/C++ 2010 software from task bar. The main window of Visual Studio 2010 should be similar to the below display:


Start Page - Microsoft Visual Studio
File Edit View Debug Team Data Tools Test Window Help
Solution Explorer
Start Page
Visual Studio 2010 Professional
Connect To Team Foundation Server
New Project...
Open Project...
Recent Projects
matrix multiplication omp
openMP test1
omp
Get Started
Guidance and Resources
Latest News
Welcome Windows Web Cloud Office SharePoint Data
What's New in Visual Studio 2010
Learn about the new features included in this release.
Visual Studio 2010 Overview
What's New in .NET Framework 4
What's New in Visual C++
Customize the Visual Studio Start Page
Creating Applications with Visual Studio
Output
Show output from:
Solution...
Class View
Ready

☑ Close page after project load
☑ Show page on startup

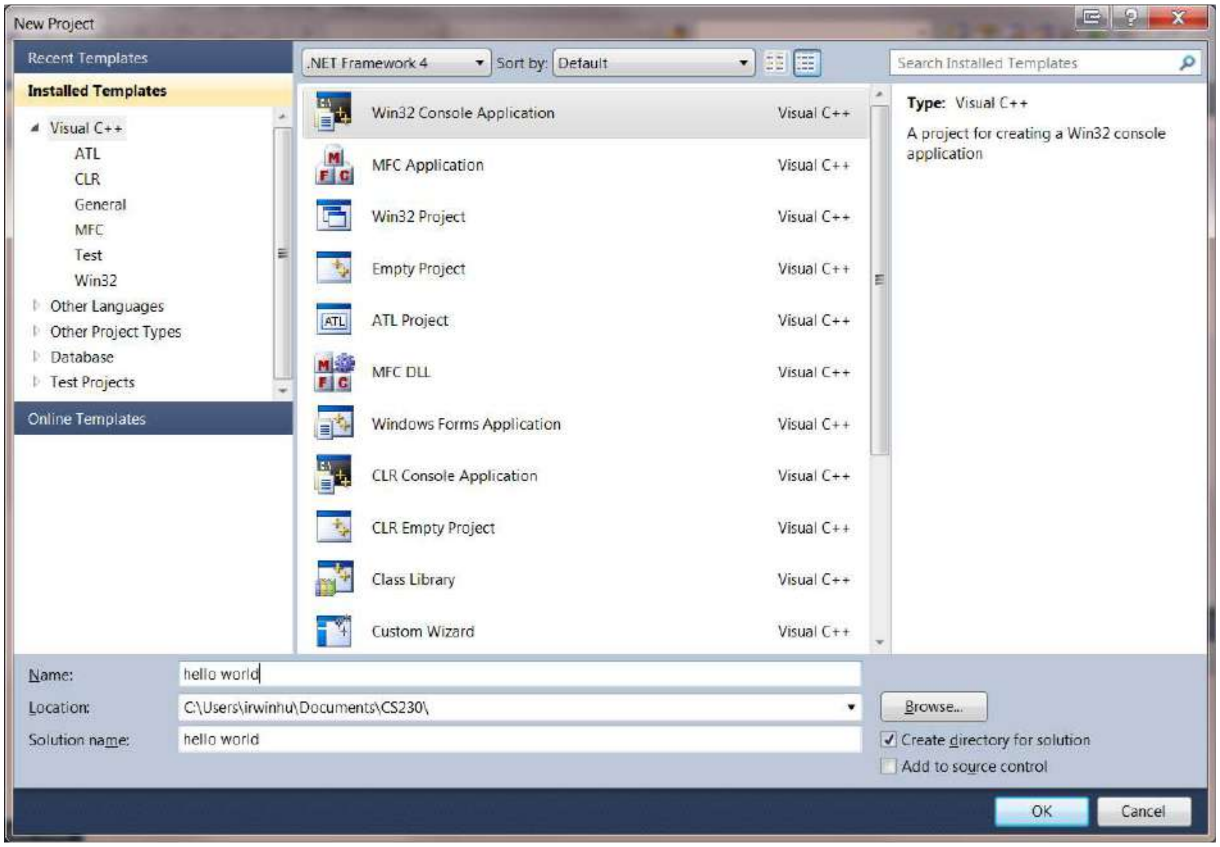
Note:

1. Hereafter, all system defined terms including menu items such as File will appear in bold and all entries made by programmers such as a filename are italicized.

2. If the Solution Explorer window on the left is not shown, click View in the menu bar and then click Solution Explorer to display it.

Step 2: In the menu bar, click **File →New → Projects…**to display the **New Project** dialog box shown below**:**

In the **New** dialog box shown below, select by clicking **Visual C++** in the **Installed Template** pane and **Win32 Console Application** in the middle pane. You then enter a name of your choice for the project (e.g., *hello world* as shown) in the Name box, select the location or folder in which you'd like to store project files (e.g., C:\Users\irwinhu\Documents\CS230\ as shown) by clicking the **Browse…** Note that there is no need to enter a name in the Solution name box; the system fills the project name in it by default.



.NET Framework 4 | Sort by: Default

Win32 Console Application	Visual C++
MFC Application	Visual C++
Win32 Project	Visual C++
Empty Project	Visual C++
ATL Project	Visual C++
MFC DLL	Visual C++
Windows Forms Application	Visual C++
CLR Console Application	Visual C++
CLR Empty Project	Visual C++
Class Library	Visual C++
Custom Wizard	Visual C++

Type: Visual C++

A project for creating a Win32 console application

Name: hello world

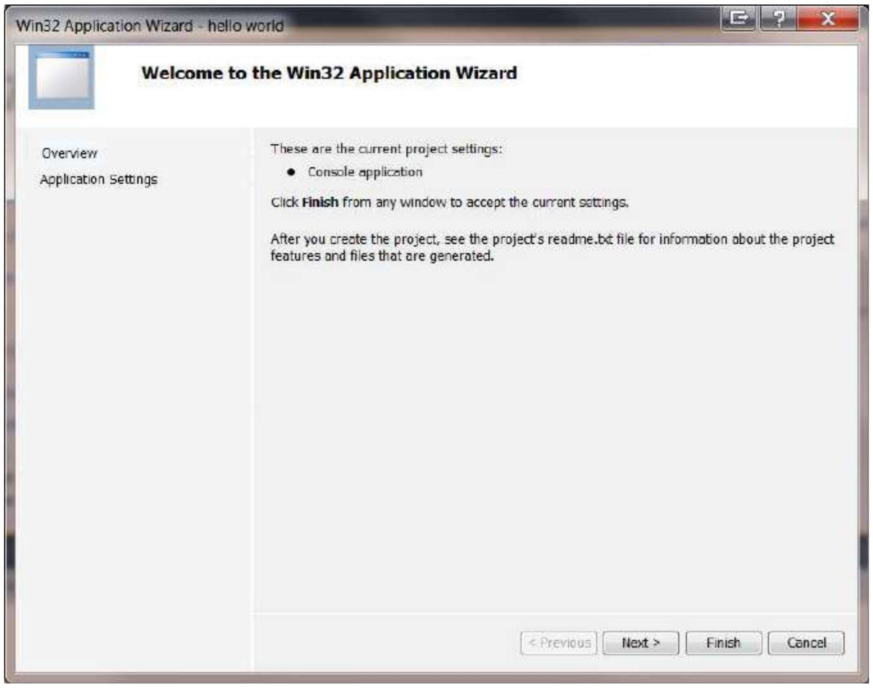
Location: C:\Users\irwinhu\Documents\CS230\

Solution name: hello world

[✓] Create directory for solution

[ ] Add to source control

Click on the **OK** button to display the **Win32 Application Wizard –** ***hello world*** window shown below:



Welcome to the Win32 Application Wizard

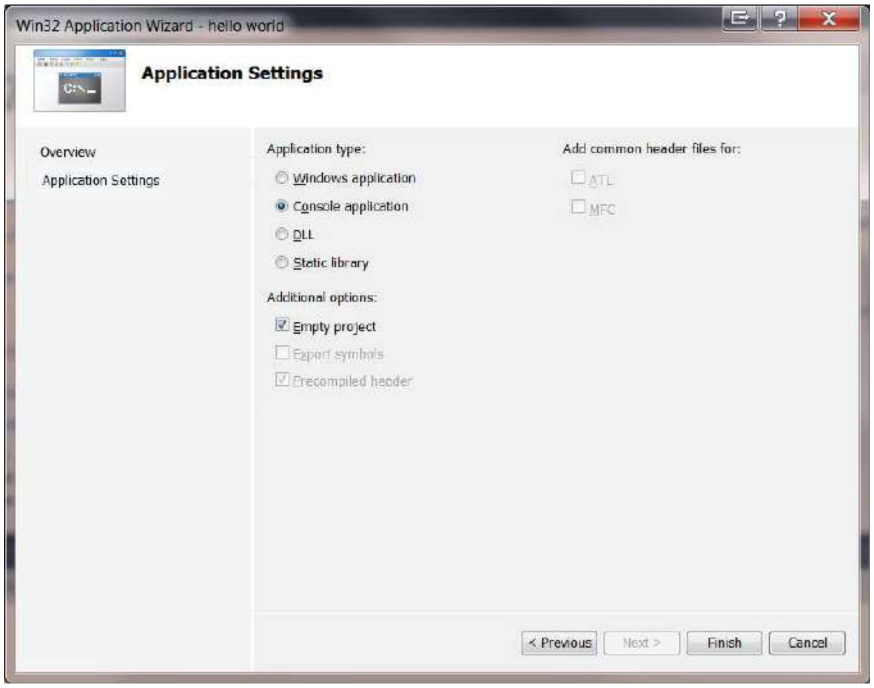
These are the current project settings:

- Console application

Click **Finish** from any window to accept the current settings.

After you create the project, see the project's readme.txt file for information about the project features and files that are generated.

Click the **Next** button to display the following dialog box:



Application Settings

Application type:

- Windows application
- [✓ Console application]
- DLL
- Static library

Additional options:

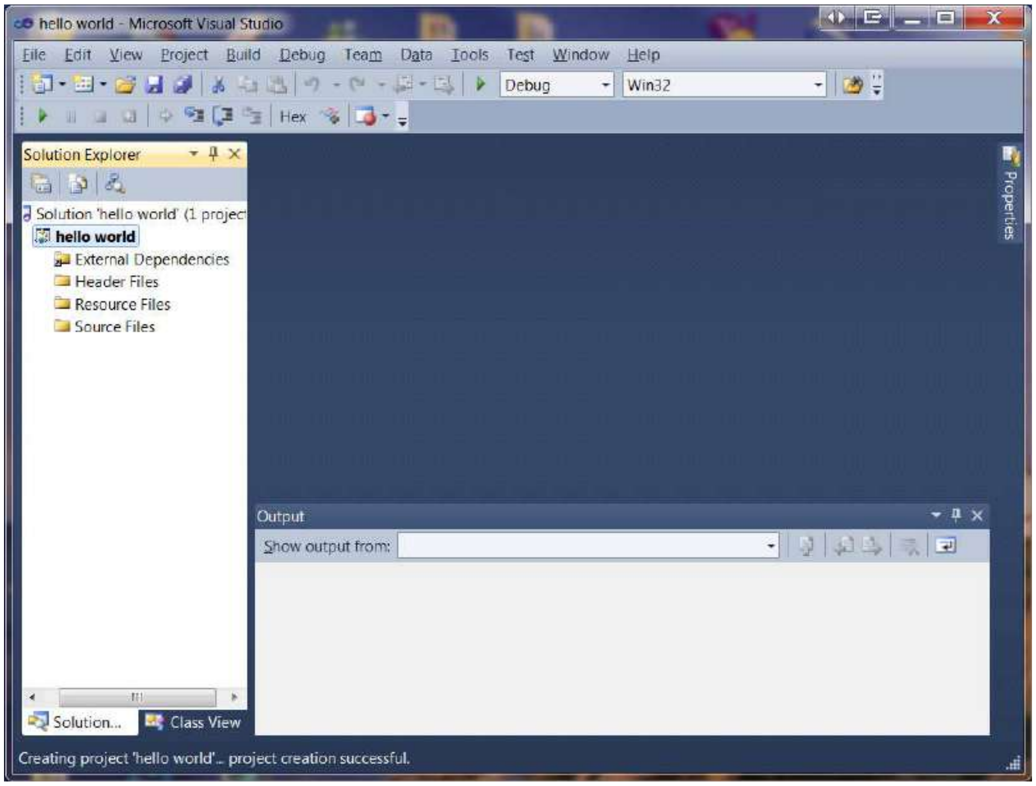
- [✓ Empty project]
- Export symbols
- [✓ Precompiled header]

Add common header files for:

- ATL
- MFC

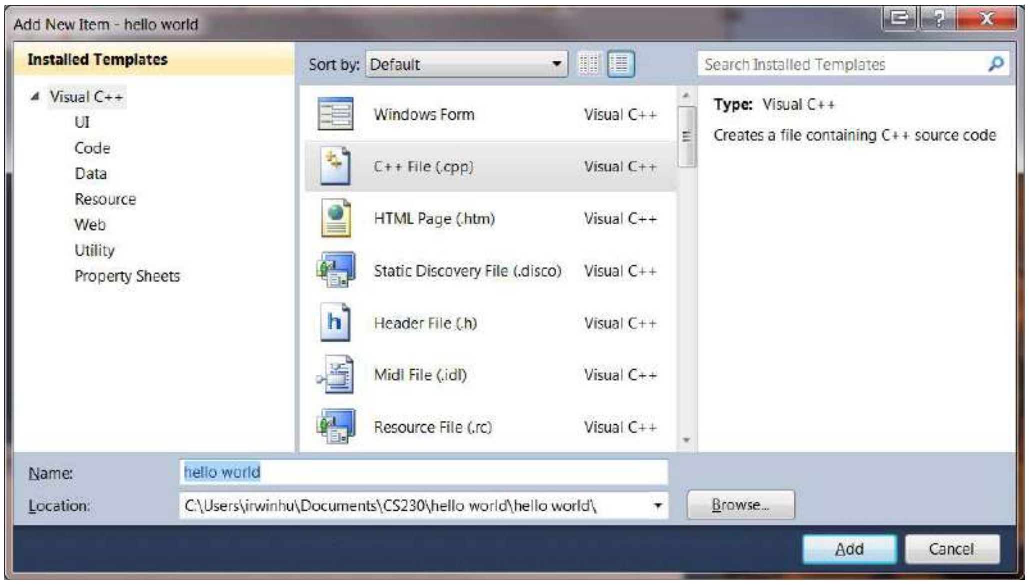
Check the **Empty project** box as shown above and click on the **Finish** button to proceed to the next step.

Step 3: Now the system displays the following window.



Creating project 'hello world'... project creation successful.

Right click on the **Source Files** folder in the **Solution Explorer** pane. In the popup menu, click **Add** then **New Item**… to display the following **Add New Item –** ***hello world*** dialog box:

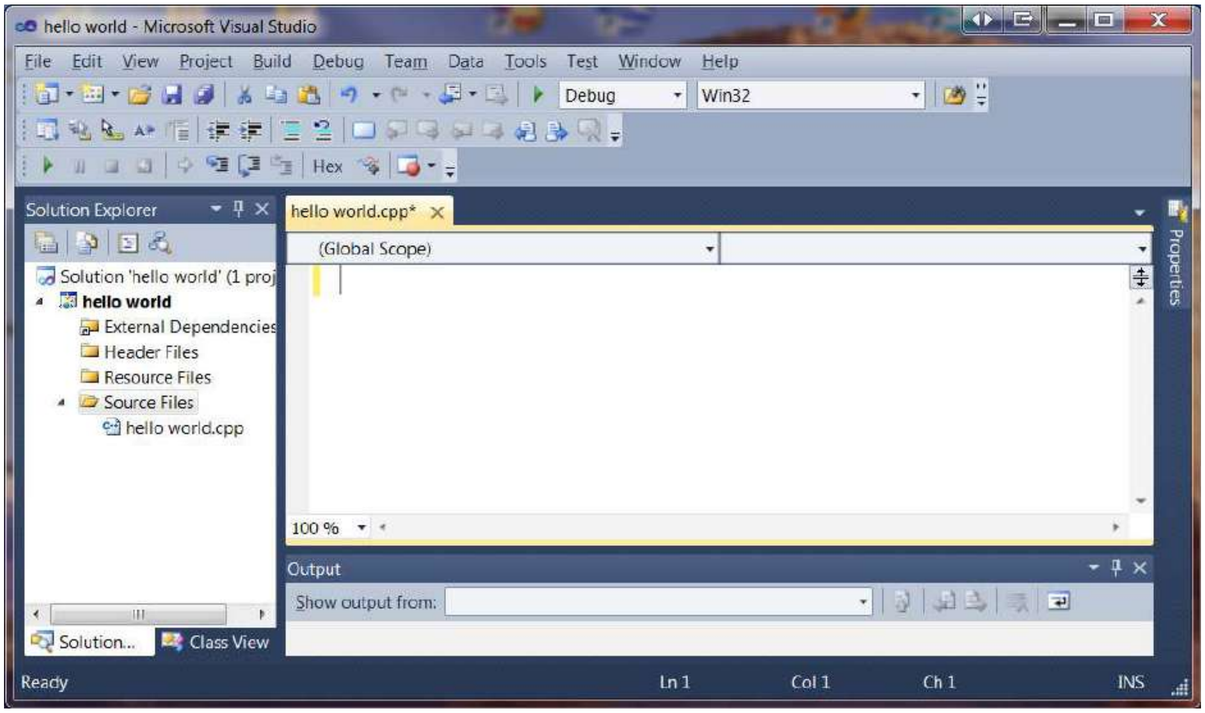

Add New Item - hello world
Installed Templates
Visual C++
UI
Code
Data
Resource
Web
Utility
Property Sheets
Sort by: Default
Search Installed Templates
Windows Form Visual C++
C++ File (.cpp) Visual C++
HTML Page (.htm) Visual C++
Static Discovery File (.disco) Visual C++
Header File (.h) Visual C++
Midl File (.idl) Visual C++
Resource File (.rc) Visual C++
Browse...
Add
Cancel

Type: Visual C++
Creates a file containing C++ source code

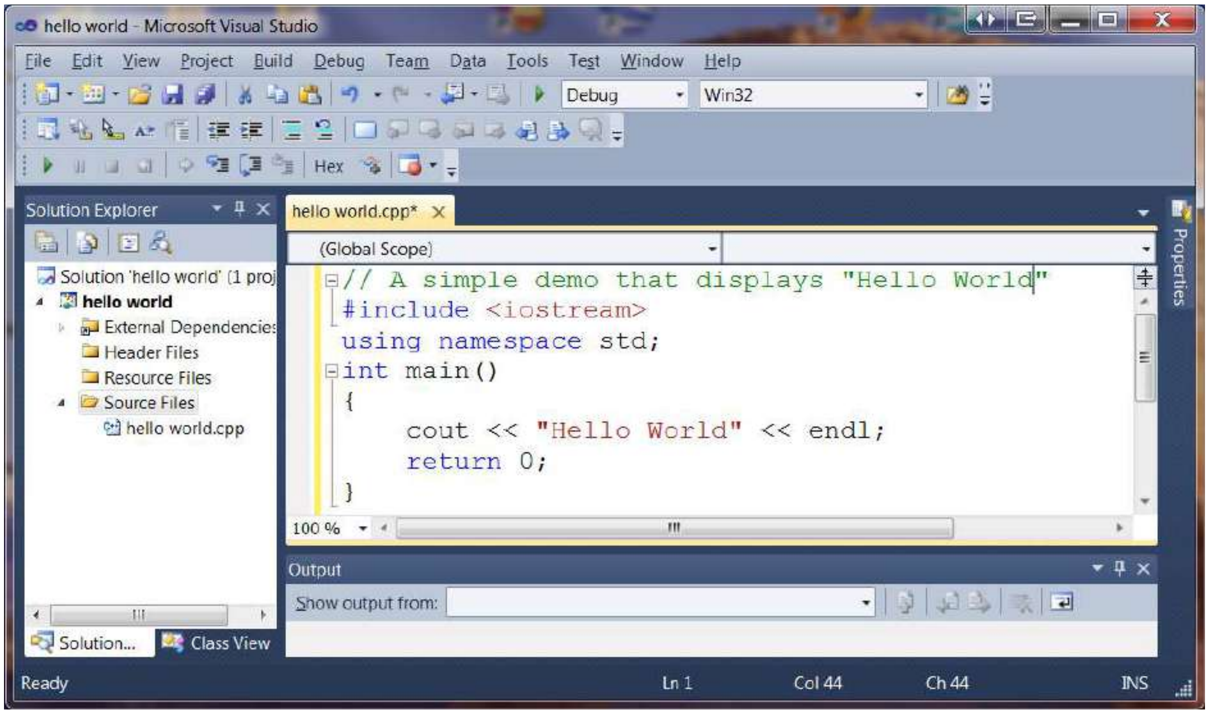
Name: hello world
Location: C:\Users\irwinhu\Documents\CS230\hello world\hello world\

Select **C++ Files (.cpp)** by clicking on it in the middle pane and enter an arbitrary file name (e.g., *hello world*, the same name we used for the project is chosen here). Click **Add** to proceed to the next step.

Step 4: The system displays the below window. Notice that 1) the **Source Folder** in **Solution Explorer** pane (make sure the **Solution Explorer** instead of **Class View** tab is selected) contains the *hello world.cpp* file that we just added and 2) the blank editing area/board is displayed with a *hello world.cpp* tab for you to enter your C++ source code.

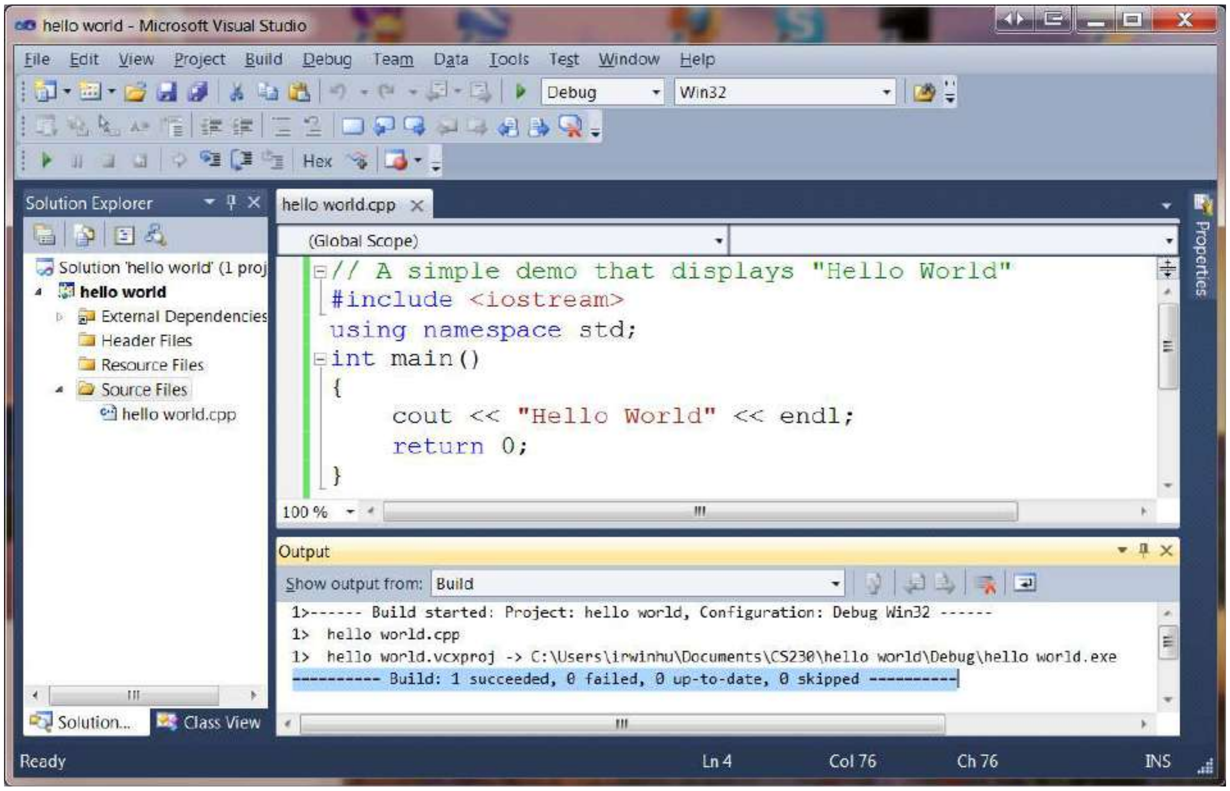


The same window after the source code is entered:



```
// A simple demo that displays "Hello World"
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello World" << endl;
    return 0;
}
```

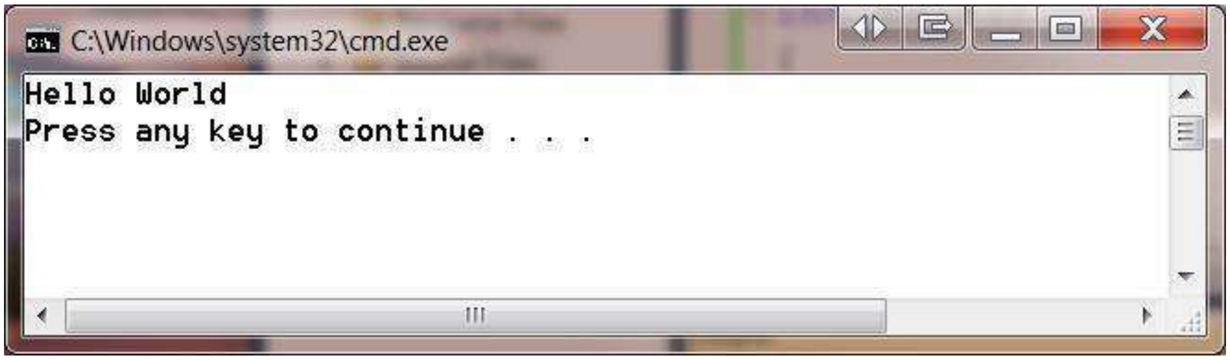
Step 5: To compile, link, load, and execute a program in a single step, click Debug in the menu bar and the click Start Without Debugging. If there is no error in your project, the below message `========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========` is displayed in the Output window as shown below.



```
// A simple demo that displays "Hello World"
#include <iostream>
using namespace std;
int main()
{
     cout << "Hello World" << endl;
     return 0;
}
```

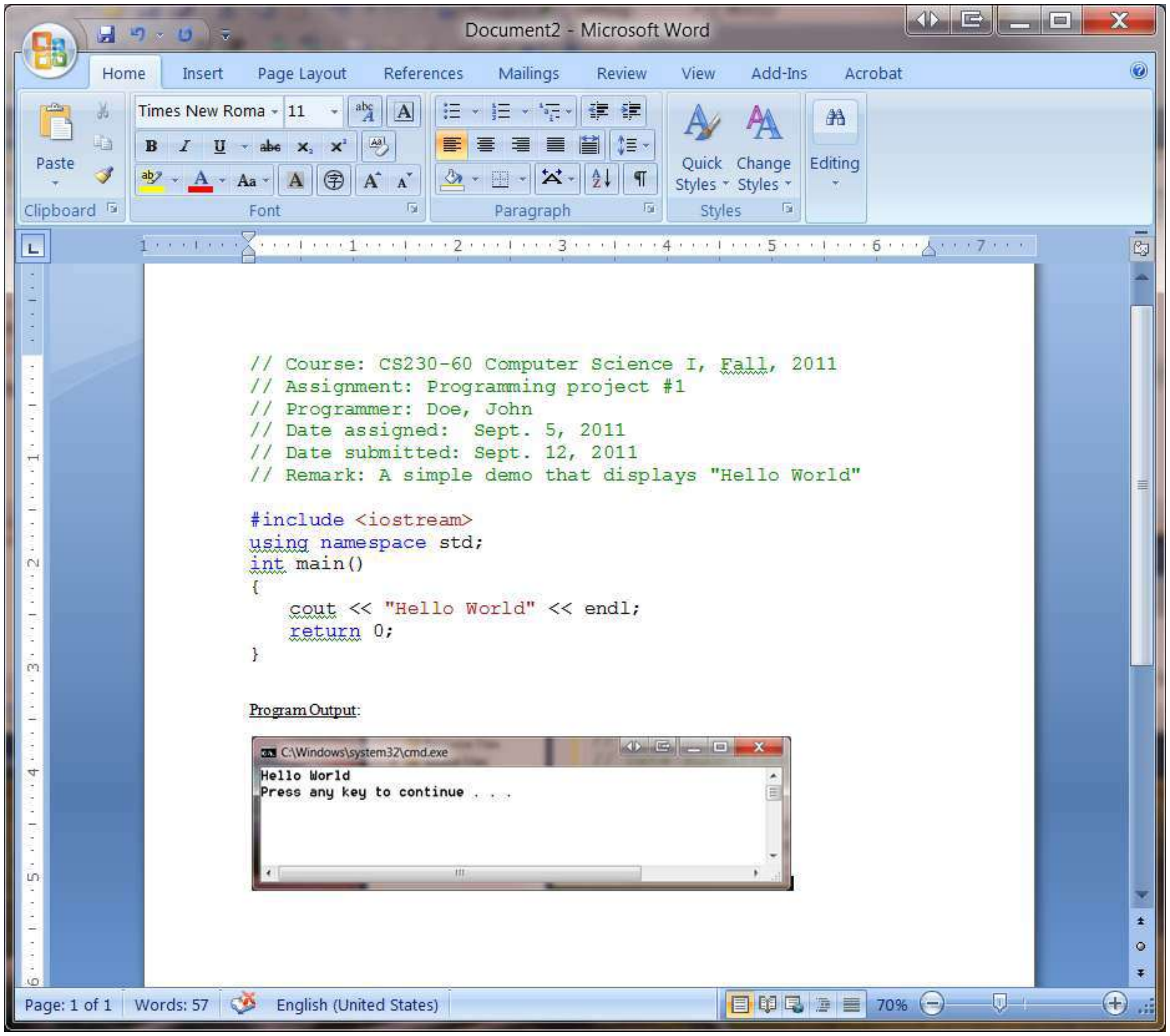
```
1>------ Build started: Project: hello world, Configuration: Debug Win32 ------
1>  hello world.cpp
1>  hello world.vcxproj -> C:\Users\irwinhu\Documents\CS230\hello world\Debug\hello world.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Also displayed in a separate window (C:\Windows\system32\cmd.exe) is the program output:



```
Hello World
Press any key to continue . . .
```

Step 6: You are normally required to turn in both the source code and the program output in printed form. One way to do it is to copy the source code from Visual C++ 2010 and paste it onto a Word document. Immediately after the source code, do a screen capture of the program output window (click on the window to activate it, hold down the **Alt** key and press the **Print Screen** key) and paste it on the Word document as shown below.



```
// Course: CS230-60 Computer Science I, Fall, 2011
// Assignment: Programming project #1
// Programmer: Doe, John
// Date assigned:  Sept. 5, 2011
// Date submitted: Sept. 12, 2011
// Remark: A simple demo that displays "Hello World"

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

Program Output:

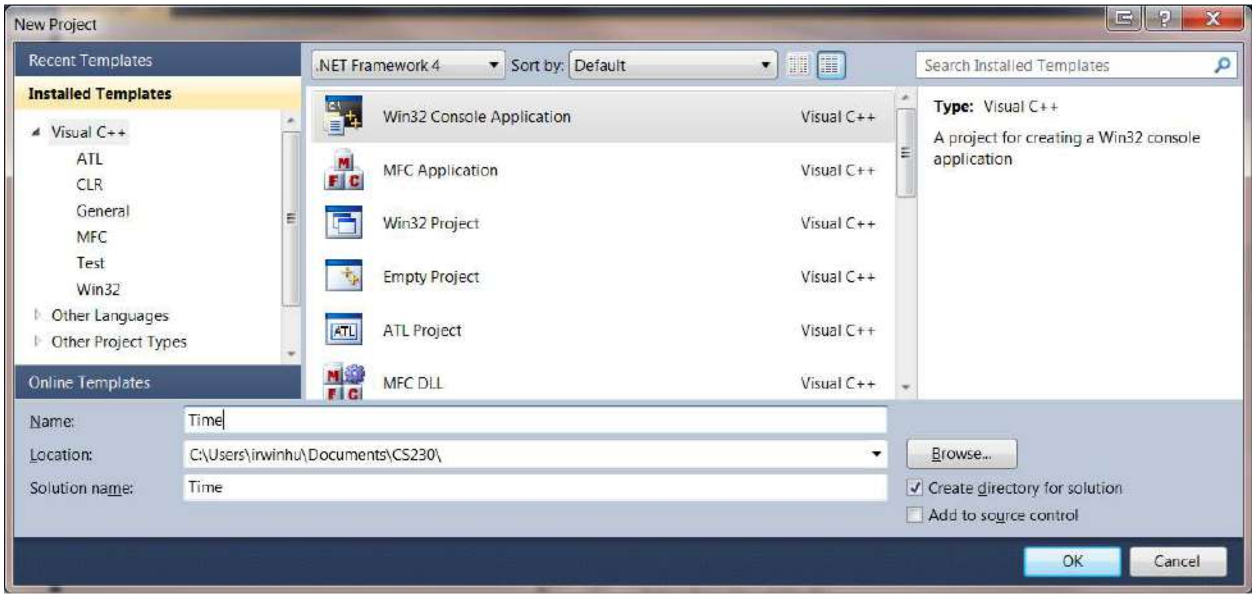
```
Hello World
Press any key to continue . . .
```

Note that comment statements have been added to the program code as part of the documentation requirement for a programming project. Check with your instructor regarding program documentation and submission rules.

A Multiple-File Project: The *Time* Program

In this demonstration, the project or program that prints a given time of the day in both military as well as standard time formats. The project contains three files created by the programmer: two source files (*timeDemo.cpp* and *Time.cpp*) and a header file (*Time.h*). Again we'll build the project from scratch.

Step 1: Launch Visual C++ 2010, click **File → New → Project…** to display the **New Project** dialog box. In the dialog box, select **Visual C++** in **Installed Template** pane and **Win32 Console Applications** in the middle pane. Enter project name (e.g., *Time* in this demo) in the **Name** box and select the folder in which you'd like files for this project to be stored (e.g., *C:\Users\irwinhu\Documents\CS230\* in this demo) by clicking the **Browse…** button as shown below:



Type: Visual C++

A project for creating a Win32 console application

Name: Time

Location: C:\Users\irwinhu\Documents\CS230\

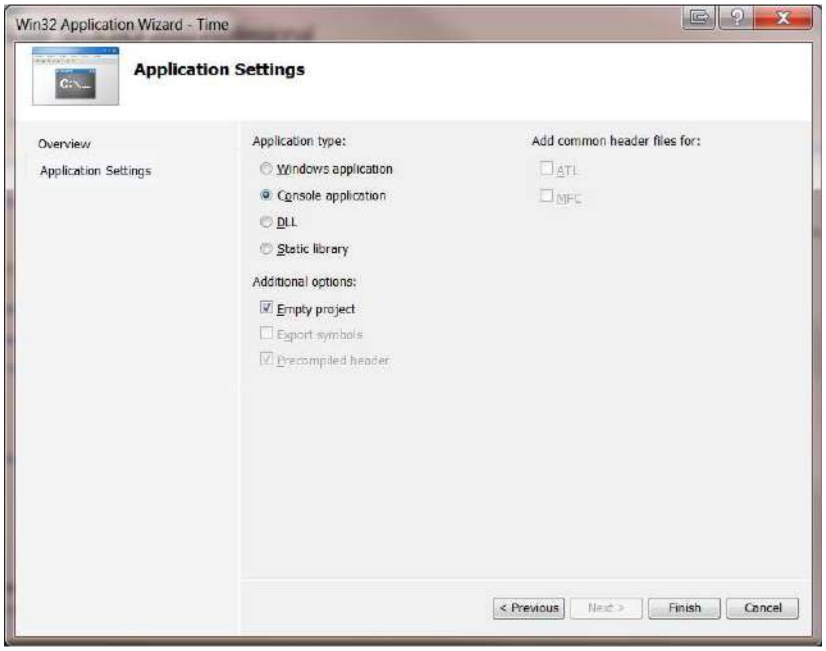
Solution name: Time

☑ Create directory for solution

☐ Add to source control

Click **OK** to display the **Win32 Application Wizard –** ***Time*** dialog box.

Step 2: In the dialog box, click **Next** (do not click **Finish**) and check the **Empty project** box as shown below:



Application Settings

Application type:

- Windows application
- [✓ Console application]
- DLL
- Static library

Add common header files for:

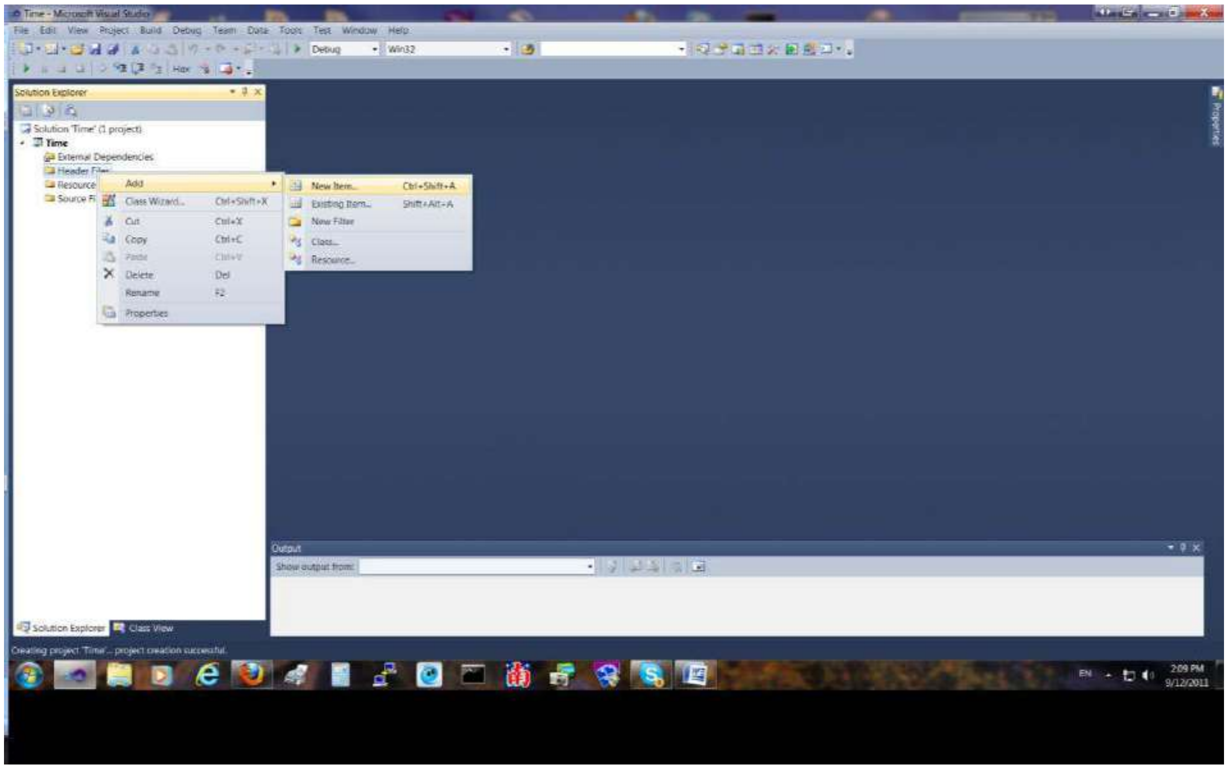
- ATL
- MFC

Additional options:

- [✓ Empty project]
- Export symbols
- [✓ Precompiled header]

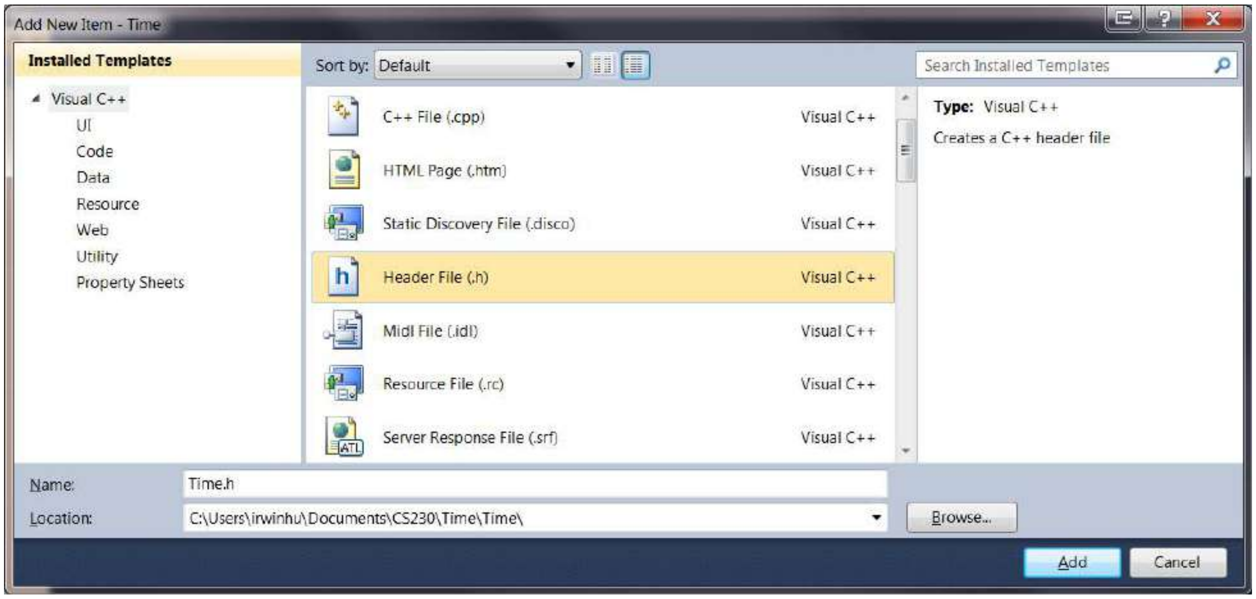
Click Finish to move on.

Step 3: Add header file to Header Files folder by right clicking the folder and then click **Add → New Item…**



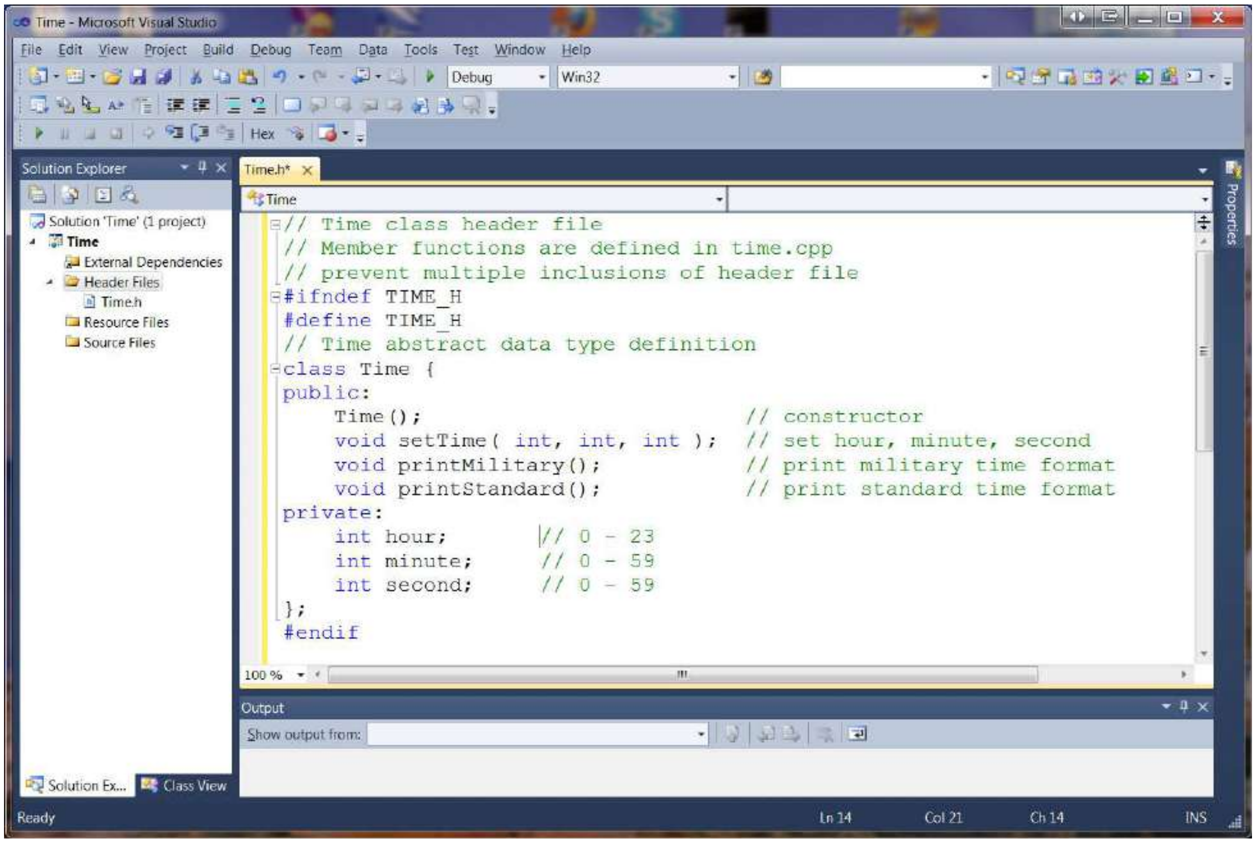
Creating project 'Time'... project creation successful.

In the **Add New Item –*Time*** dialog box, select **Header File (.h)** in the middle panel and enter an arbitrary name (e.g., Time in this demo) for the header file:


Add New Item - Time
Installed Templates
Visual C++
UI
Code
Data
Resource
Web
Utility
Property Sheets
Sort by: Default
Search Installed Templates
C++ File (.cpp) Visual C++
HTML Page (.htm) Visual C++
Static Discovery File (.disco) Visual C++
Header File (.h) Visual C++
Midl File (.idl) Visual C++
Resource File (.rc) Visual C++
Server Response File (.srf) Visual C++
Type: Visual C++
Creates a C++ header file
Browse...
Add
Cancel

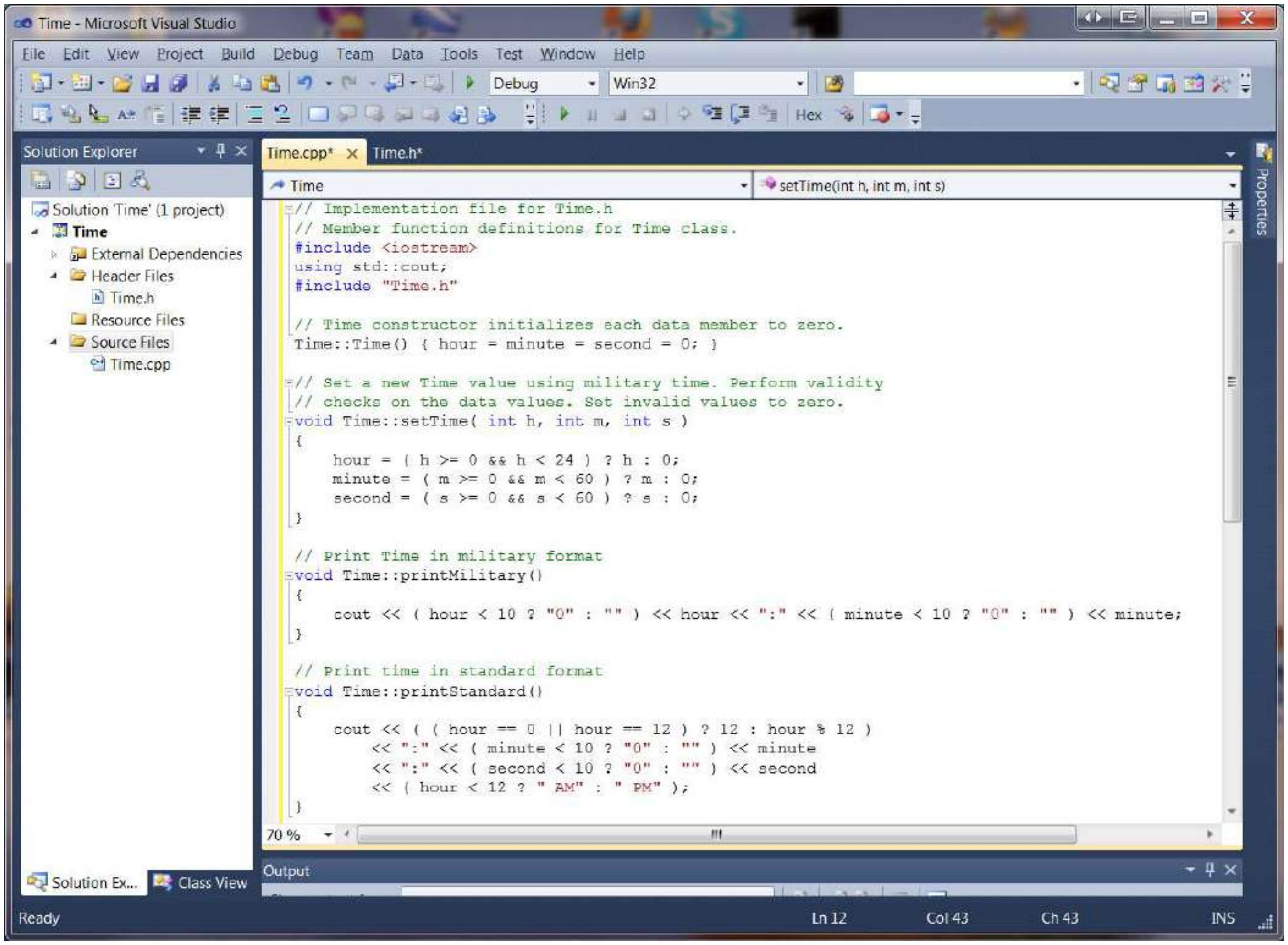
Name: Time.h
Location: C:\Users\irwinhu\Documents\CS230\Time\Time\

Click the **Add** button, the empty *Time.h* file appears in the **Header Files** folder in the **Solution Explorer** window. Enter source code in the editing area as shown:



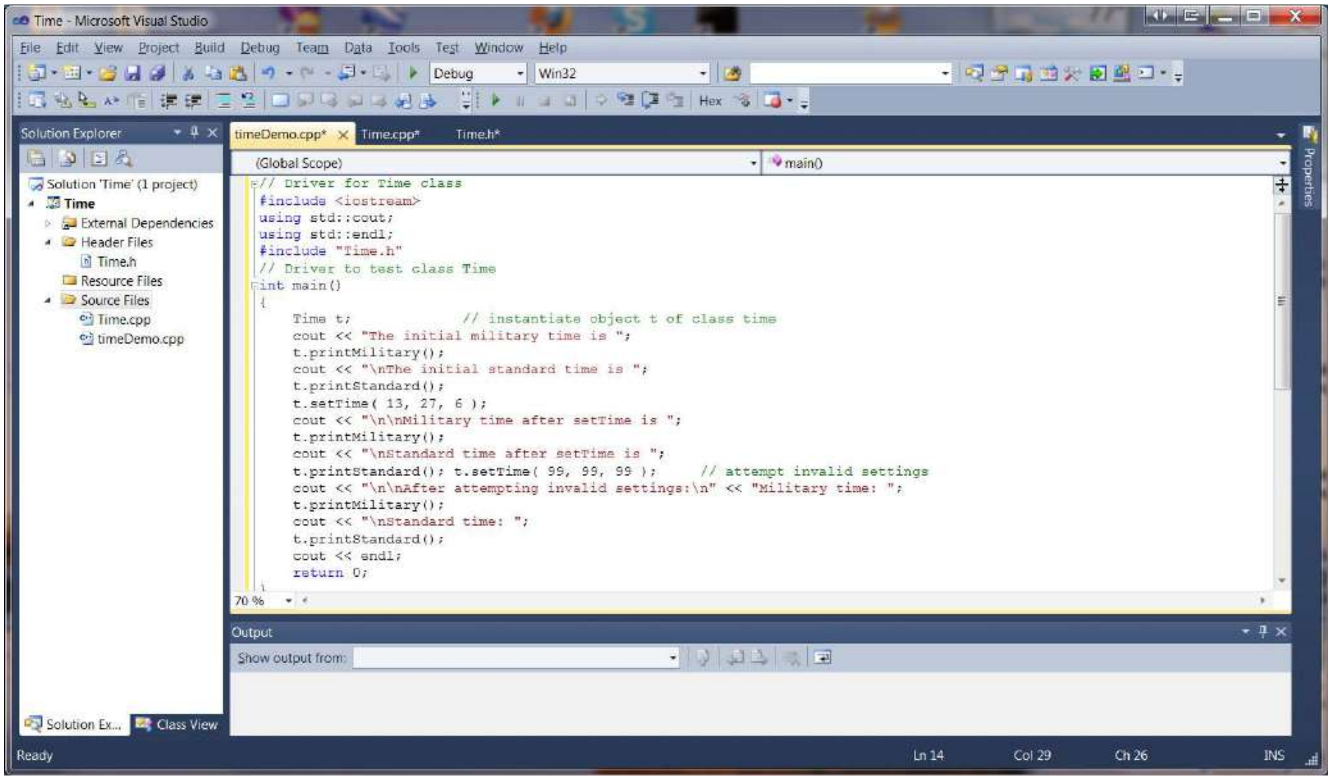
```
// Time class header file
// Member functions are defined in time.cpp
// prevent multiple inclusions of header file
#ifndef TIME_H
#define TIME_H
// Time abstract data type definition
class Time {
public:
    Time();                               // constructor
    void setTime( int, int, int );        // set hour, minute, second
    void printMilitary();                 // print military time format
    void printStandard();                 // print standard time format
private:
    int hour;       // 0 - 23
    int minute;     // 0 - 59
    int second;     // 0 - 59
};
#endif
```

Step 4: Similar to step 3 above, add two source files (must be *.cpp* type; but the file names can be arbitrary, e.g., *Time.cpp* and *TimeDemo.cpp* in this demo), one at a time, to the **Source Files** folder by right clicking the **Source Files** folder. The below screen shows the source code for *Time.cpp* file has been entered:



```
// Implementation file for Time.h
// Member function definitions for Time class.
#include <iostream>
using std::cout;
#include "Time.h"

// Time constructor initializes each data member to zero.
Time::Time() { hour = minute = second = 0; }

// Set a new Time value using military time. Perform validity
// checks on the data values. Set invalid values to zero.
void Time::setTime( int h, int m, int s )
{
    hour = ( h >= 0 && h < 24 ) ? h : 0;
    minute = ( m >= 0 && m < 60 ) ? m : 0;
    second = ( s >= 0 && s < 60 ) ? s : 0;
}

// Print Time in military format
void Time::printMilitary()
{
    cout << ( hour < 10 ? "0" : "" ) << hour << ":" << ( minute < 10 ? "0" : "" ) << minute;
}

// Print time in standard format
void Time::printStandard()
{
    cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
        << ":" << ( minute < 10 ? "0" : "" ) << minute
        << ":" << ( second < 10 ? "0" : "" ) << second
        << ( hour < 12 ? " AM" : " PM" );
}
```

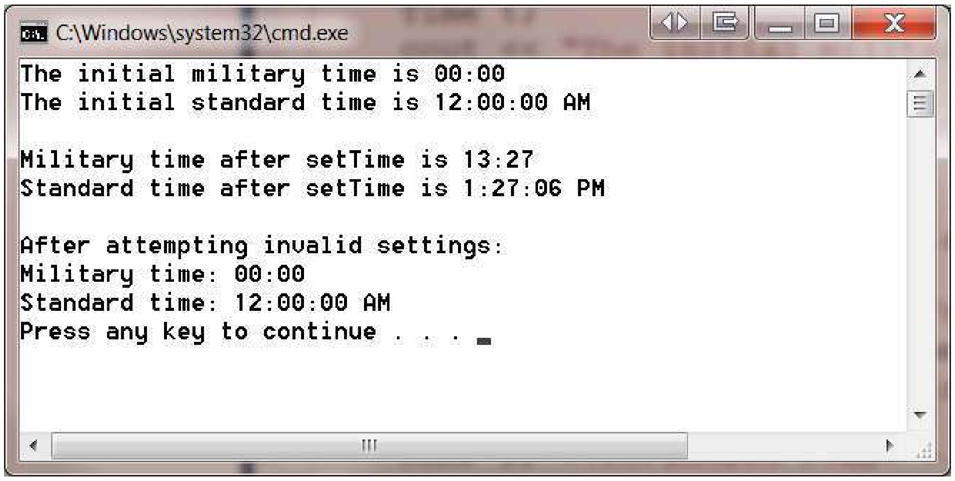
The below screen shows that the source file timeDemo.cpp (the main driver function) has been added:



```
// Driver for Time class
#include <iostream>
using std::cout;
using std::endl;
#include "Time.h"
// Driver to test class Time
int main()
{
    Time t;              // instantiate object t of class time
    cout << "The initial military time is ";
    t.printMilitary();
    cout << "\nThe initial standard time is ";
    t.printStandard();
    t.setTime( 13, 27, 6 );
    cout << "\n\nMilitary time after setTime is ";
    t.printMilitary();
    cout << "\nStandard time after setTime is ";
    t.printStandard(); t.setTime( 99, 99, 99 );     // attempt invalid settings
    cout << "\n\nAfter attempting invalid settings:\n" << "Military time: ";
    t.printMilitary();
    cout << "\nStandard time: ";
    t.printStandard();
    cout << endl;
    return 0;
}
```

Notice that in the source code editing area, there are three named tabs, each is associated with a file previously entered. To view the source of any of these files, just click the tab with desired name on it.

Step 5: To compile, link, load, and execute or run the program, click **Debug → Start Without Debugging.** Since there is no error or bug in the code, the output window displays the message `========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========`. In a separate window, the program output is display below.



```
The initial military time is 00:00
The initial standard time is 12:00:00 AM

Military time after setTime is 13:27
Standard time after setTime is 1:27:06 PM

After attempting invalid settings:
Military time: 00:00
Standard time: 12:00:00 AM
Press any key to continue . . .
```

Appendix: The Source Code

Single-file Demo: The Hello World Program

File name: Hello World.cpp

```
#include <iostream>
Using namespace std;
int main()
{
   cout << "Hello World!" << endl; return 0;
}
```

Multiple-file Demo: the Time Program

File 1: Time.h

```
// Time class header file
// Member functions are defined in time.cpp
// prevent multiple inclusions of header file
#ifndef TIME1_H
#define TIME1_H
// Time abstract data type definition
class Time {
public: Time();                       // constructor
void setTime( int, int, int );        // set hour, minute, second
void printMilitary();                 // print military time format
void printStandard();                 // print standard time format
private:
   int hour;                          // 0 - 23
   int minute;                        // 0 - 59
   int second;                        // 0 - 59
};
#endif
```

File 2: Time.cpp

```
// Implementation file for Time.h
// Member function definitions for Time class.
#include <iostream>
using std::cout;
#include "time.h"

// Time constructor initializes each data member to zero.
Time::Time() { hour = minute = second = 0; }

// Set a new Time value using military time. Perform validity
// checks on the data values. Set invalid values to zero.
```

```
void Time::setTime( int h, int m, int s )
{
   hour = ( h >= 0 && h < 24 ) ? h : 0;
   minute = ( m >= 0 && m < 60 ) ? m : 0;
   second = ( s >= 0 && s < 60 ) ? s : 0;
}
// Print Time in military format
void Time::printMilitary()
{
   cout << ( hour < 10 ? "0" : "" ) << hour << ":" << ( minute < 10 ? "0" :
"" ) << minute;
}

// Print time in standard format
void Time::printStandard()
{
   cout << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
      << ":" << ( minute < 10 ? "0" : "" ) << minute
      << ":" << ( second < 10 ? "0" : "" ) << second
      << ( hour < 12 ? " AM" : " PM" );
}
```

File 3: timeDemo.cpp (main function – the driver)

```
// Driver for Time class
#include <iostream>
using std::cout;
using std::endl;
#include "time.h"
// Driver to test class Time
int main()
{
   Time t;
   // instantiate object t of class time
   cout << "The initial military time is ";
   t.printMilitary();
   cout << "\nThe initial standard time is ";
   t.printStandard();
   t.setTime( 13, 27, 6 );
   cout << "\n\nMilitary time after setTime is ";
   t.printMilitary();
   cout << "\nStandard time after setTime is ";
   t.printStandard(); t.setTime( 99, 99, 99 );    // attempt invalid settings
   cout << "\n\nAfter attempting invalid settings:\n" << "Military time: ";
   t.printMilitary();
   cout << "\nStandard time: ";
   t.printStandard();
   cout << endl;
   return 0;
}
```