

Supplement II.A: Visual C++ 2005 Tutorial

For Introduction to C++ Programming
By Y. Daniel Liang

1 Introduction

Visual C++ is a component of Microsoft Visual Studio .NET for developing C++ programs. A free version named *Visual C++ 2005 Express Edition* is included in the book's Companion CR-ROM. This section introduces how to create a project, create a program, compile and run the program.

2 Getting Started with Visual C++

Visual C++ is easy to install. If you need help on installation, please refer to *VC++ Tutorial* in the supplements.

Suppose you have installed Visual C++ 2005 Express Edition. You can launch VC++ from Windows Start button by choosing *All Programs, Visual C++ 2005 Express Edition, Microsoft Visual C++ 2005 Express Edition*. The Visual C++ 2005 Express Edition user interface appears, as shown in Figure 1.

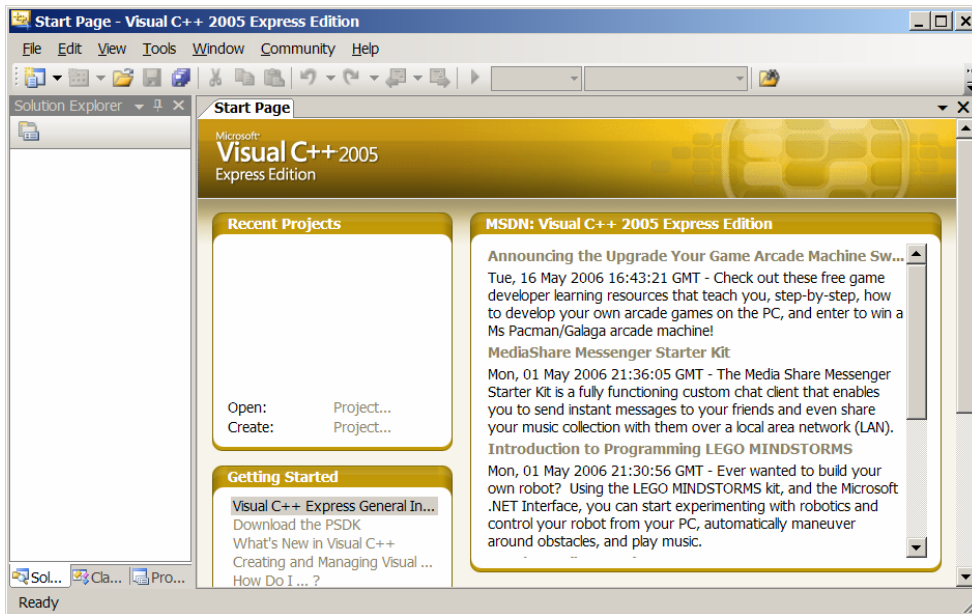


Figure 1

The VC++ user interface is a single window that performs editing, compiling, debugging, and running programs.

3 Creating a Project

To create C++ programs in VC++, you have to first create a project. A project is like a holder that ties all the files together. Here are the steps to create a project:

1. Choose *File, New, Project* to display the New Project window, as shown in Figure 2.
2. Choose *Win32* in the project types column and *Win32 Console Application* in the Templates column. Type bookexample in the Name field and c:\smith in the Location field. Click *OK* to display The Win32 Application Wizard window, as shown in Figure 3.
3. Click *Next* to display the application settings window, as shown in Figure 4.
4. Select *Console application* in the Application type section and check *Empty project* in the Additional options section. Click *Finish* to create a project. You will see the project named bookexample in the solution explorer, as shown in Figure 5.

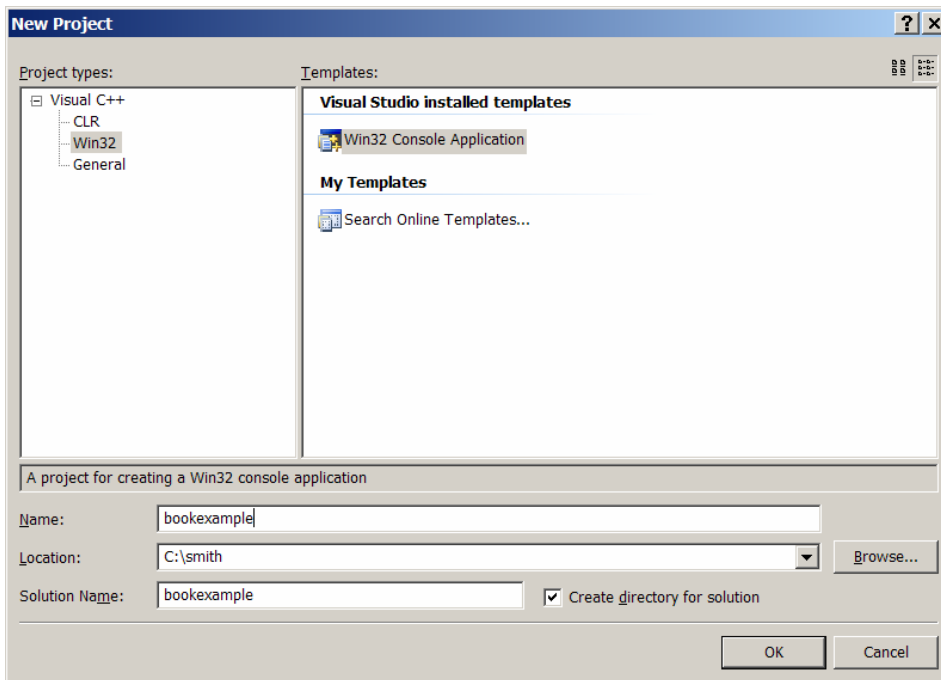


Figure 2

You need to create a project before creating programs.



Figure 3
Win32 Application Wizard creates a project for Win32 applications.

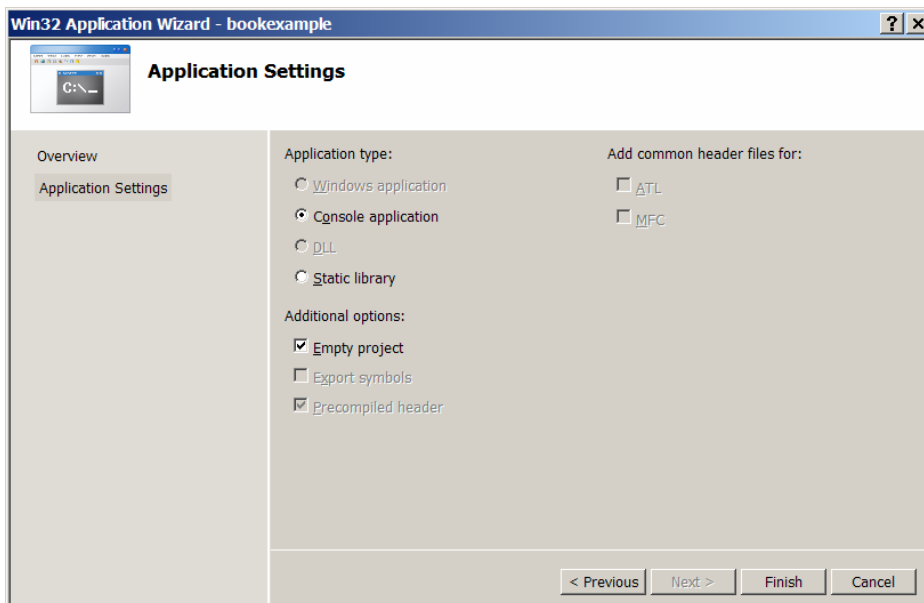


Figure 4
Win32 Application Settings window lets you set the application type.

Solution explorer
shows →

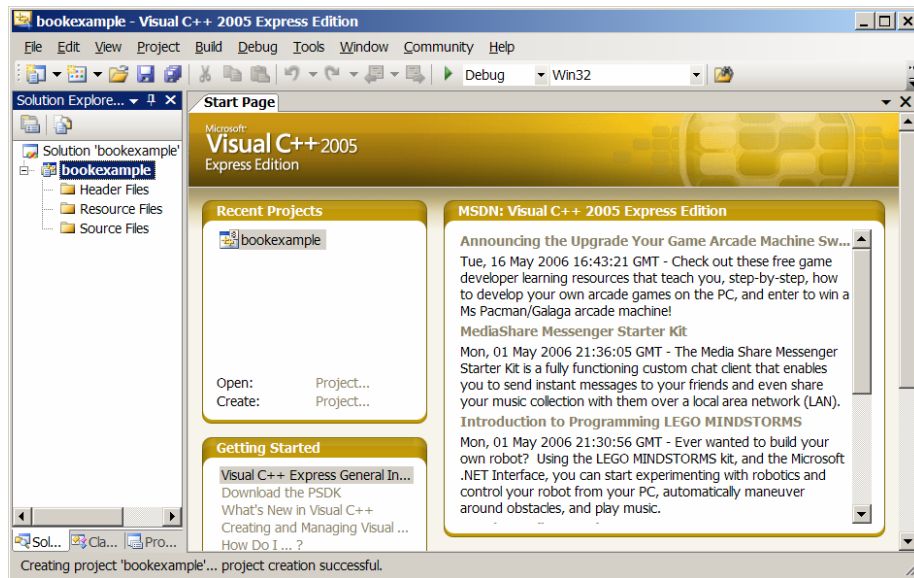


Figure 5

A project is created for C++ console applications.

4 Creating a C++ Program

After you created a project, you can create programs in the project. Here are the steps to create a C++ program for Listing 1.1:

1. Choose *Add, Add New Item* from the context menu of the bookexample project (see Figure 6) to display the Add New Item window, as shown in Figure 7.
2. Choose Code in the Categories column and C++ File (.cpp) in the Templates column. Enter Welcome in the Name field and c:\smith\bookexample\bookexample in the Location field. Click *Add* to create the file, as shown in Figure 8.
3. Enter the code for Welcome.cpp exactly from Listing 1.1, as shown in Figure 9.

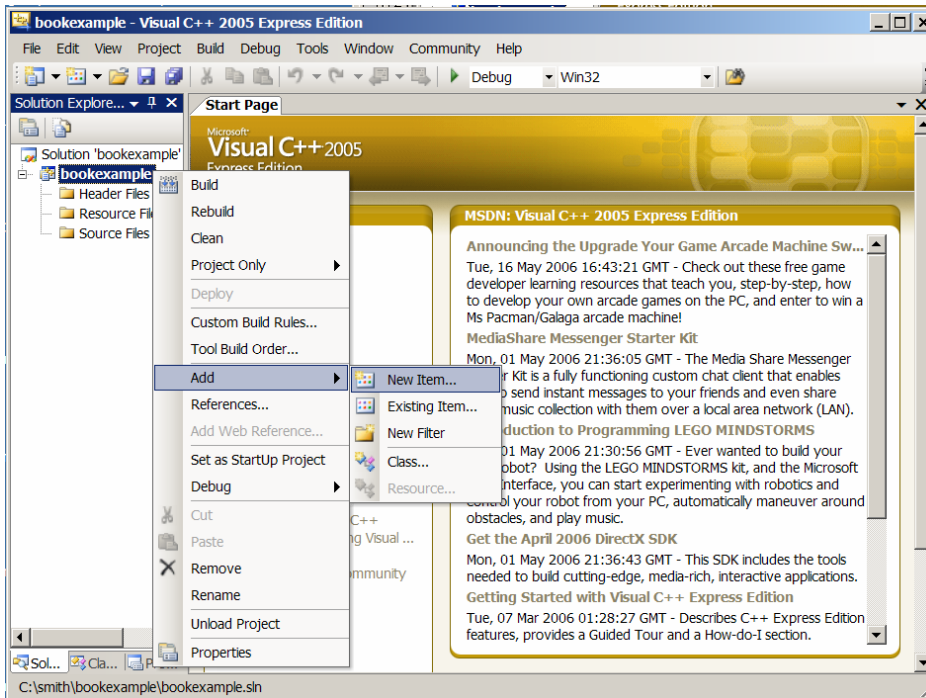


Figure 6

You can open the Add New Item window from the project's context menu.

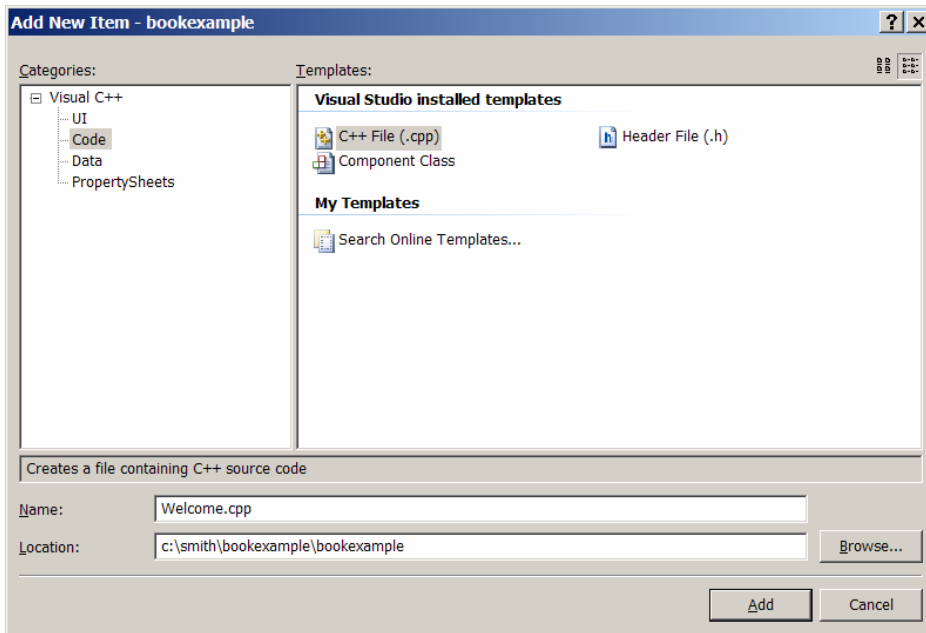


Figure 7

You can specify the file type, name, and location to create a file.

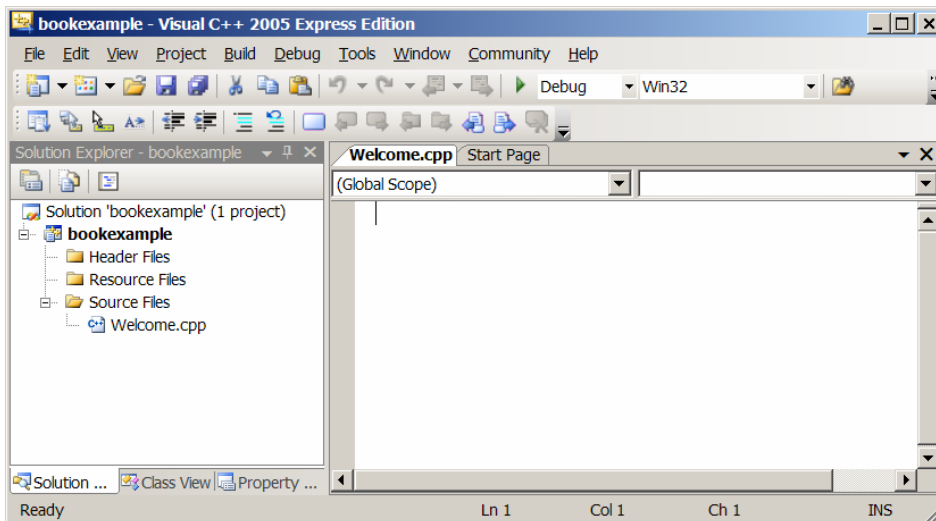


Figure 8
Welcome.cpp is created in the project.

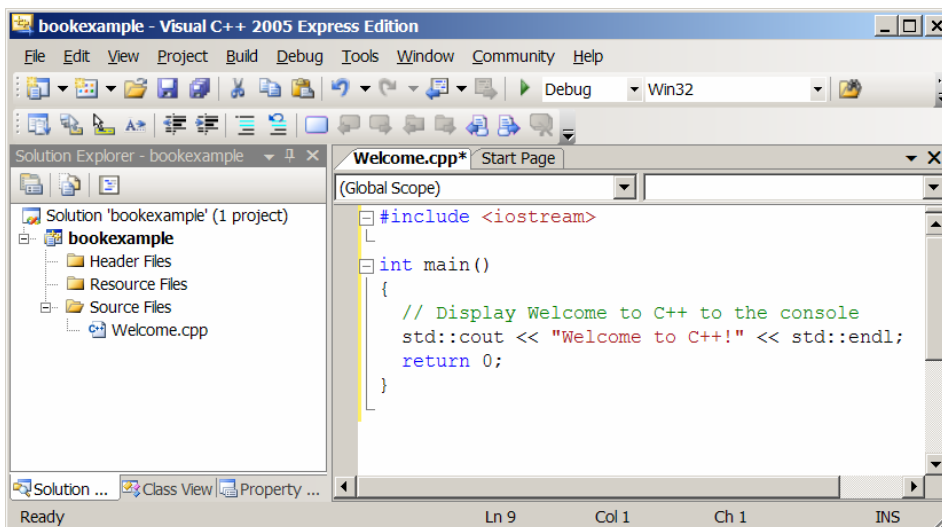


Figure 9
The source code for Welcome.cpp is entered.

5 Compiling a C++ Program

After you created a program, you can compile it. You may compile it by choosing *Build*, *Compile*, or press *Ctrl+F7*, or choose *Compile* in the context menu for *Welcome.cpp*, as shown in Figure 10.

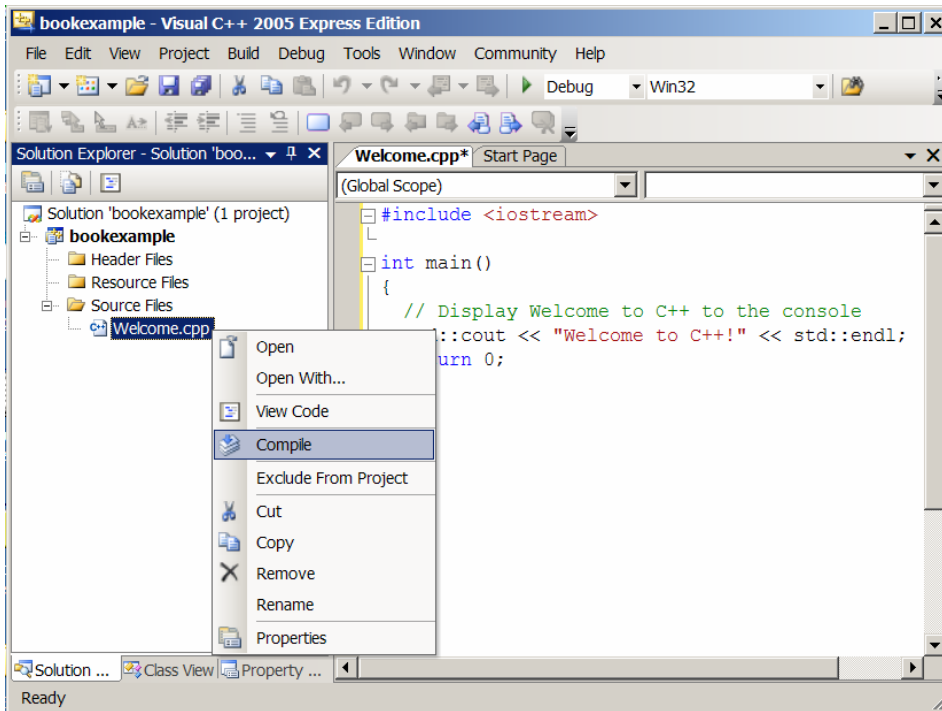


Figure 10
 Choose the *Compile* command to compile the program.

6 Running a C++ Program

To run the program, choose *Debug, Start Without Debugging*, or press *Ctrl+F5*. You will see a dialog box, as shown in Figure 11(a). Click *Yes* to continue. You will see the output is displayed in a DOS window, as shown in Figure 11(b).

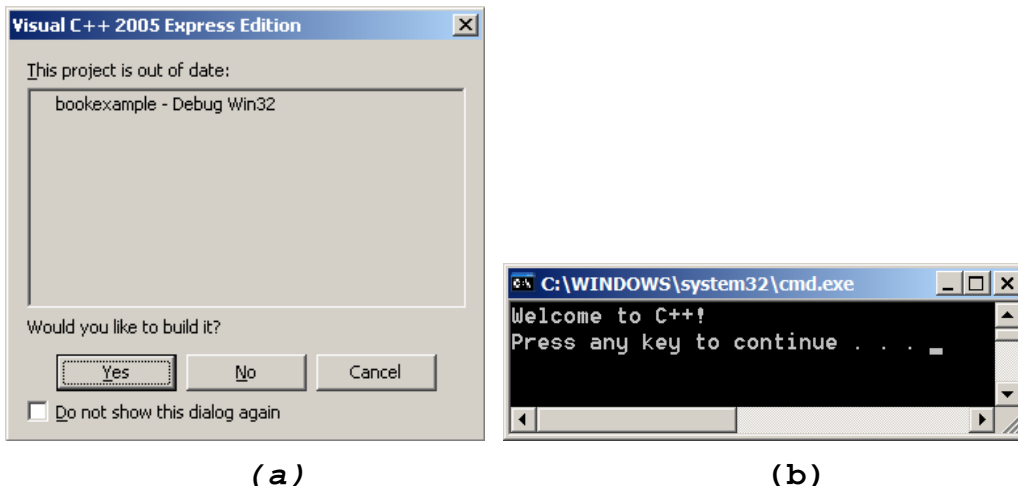


Figure 11
 The output is displayed in a DOS window.

NOTE:

<Side Remark: compile and run>

The Run command invokes the Compile command if the program is not compiled or was modified after the last compilation.

NOTE:

<Side Remark: one main function>

Each project can have only one file that contains a main function. If you need to create another file with a main function, you have two options:

- Remove the current file that contains a main function from the project by choosing *Remove* from the context menu of the program, as shown in Figure 12. (Note that you can add an existing file to the project by choosing *File, Add Existing Item*.)
- Create a new project for the new program.

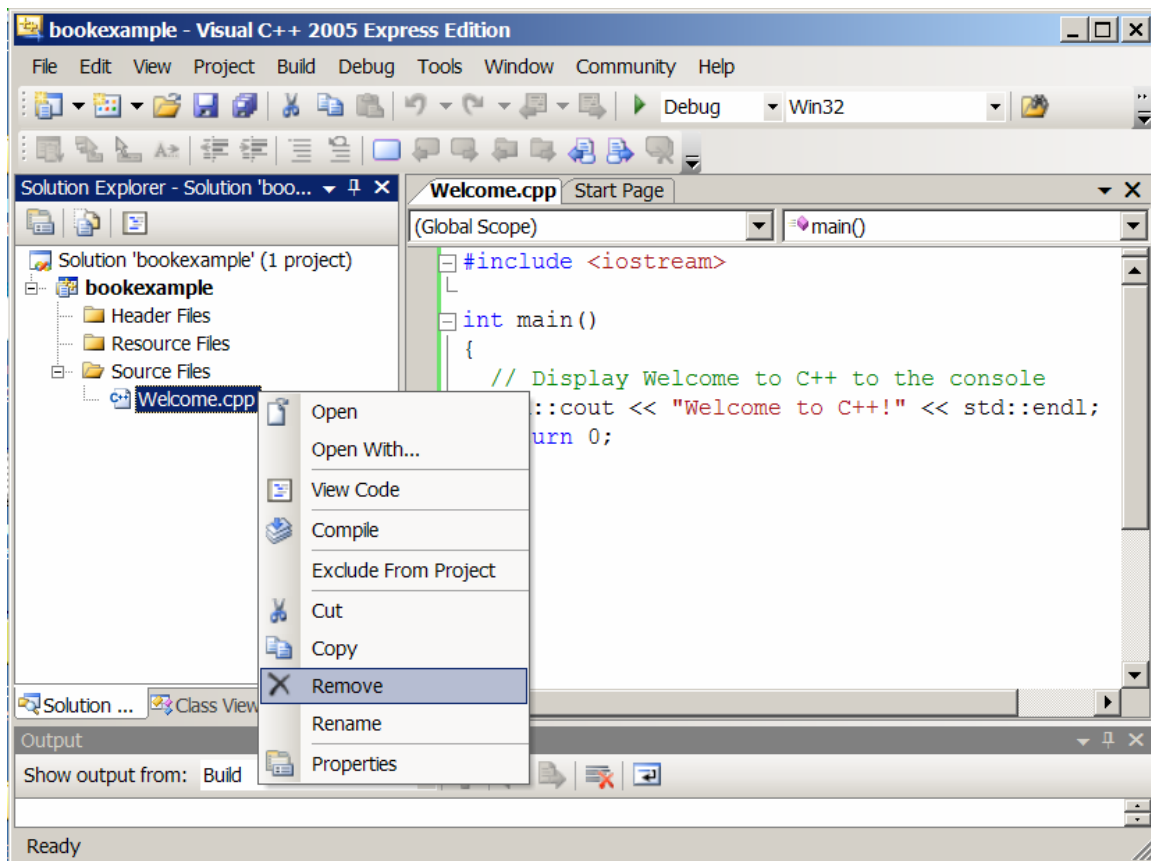


Figure 12

You can remove a file from a project.

*****END NOTE**

7 Debugging in VC++

The debugger utility is integrated in VC++. You can pinpoint bugs in your program with the help of the VC++ debugger without leaving the IDE. The VC++ debugger enables you to set breakpoints and execute programs line by line. As your program executes, you can watch the values stored in variables, observe which functions are being called, and know what events have occurred in the program. Let us use Listing 2.11, *ShowCurrentTime.cpp*, to demonstrate debugging. Create a new program named *ShowCurrentTime.cpp*. The source code for *ShowCurrentTime.cpp* can be obtained from the book.

7.1 Setting Breakpoints

You can execute a program line by line to trace it, but this is time-consuming if you are debugging a large program. Often, you know that some parts of the program work fine. It makes no sense to trace these parts when you only need to trace the lines of code that are likely to have bugs. In cases of this kind, you can use breakpoints.

A *breakpoint* is a stop sign placed on a line of source code that tells the debugger to pause when this line is encountered. The debugger executes every line until it encounters a breakpoint, so you can trace the part of the program at the breakpoint. Using the breakpoint, you can quickly move over the sections you know work correctly and concentrate on the sections causing problems.

There are several ways to set a breakpoint on a line. One quick way is to click the cutter of the line on which you want to put a breakpoint. You will see the line highlighted, as shown in Figure 13. You also can set breakpoints by choosing *Run, Add Breakpoint*. To remove a breakpoint, simply click the cutter of the line.

As you debug your program, you can set as many breakpoints as you want, and can remove breakpoints at any time during debugging. The project retains the breakpoints you have set when you exit the project. The breakpoints are restored when you reopen it.

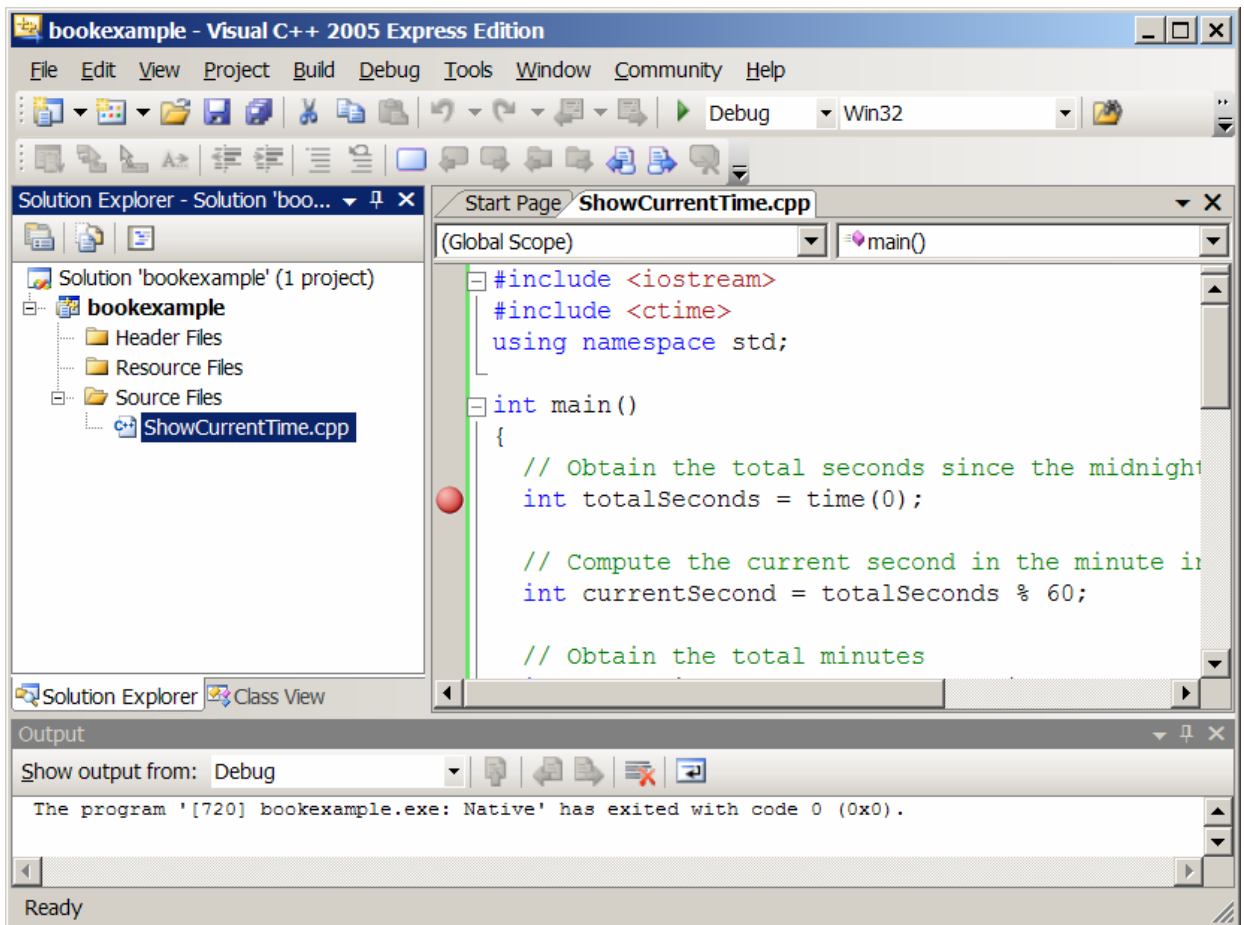


Figure 13

A breakpoint is set in ShowCurrentTime.cpp.

7.2 Starting the Debugger

To start debugging, set a break point at the first line in the main function and choose *Debug, Start* (or F5). If the program compiles without problems, debugging starts. You will see several small windows appearing at the bottom, as shown in Figure 14.

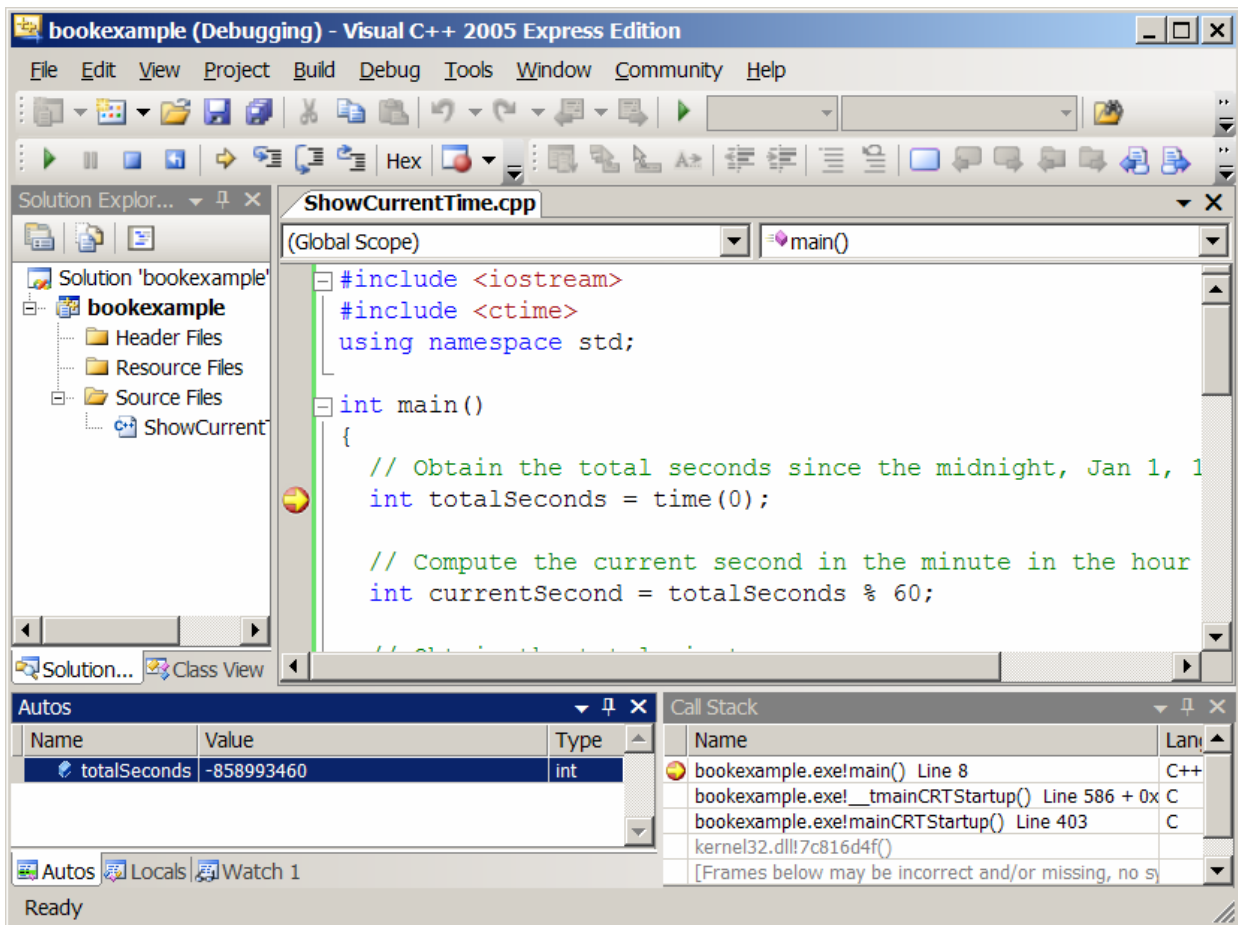


Figure 14

The debugging windows appear in the IDE.

The window on the left is called the *variable window* with three tabs at the bottom (*Autos*, *Locals*, and *Watch*). (If these windows do not appear, choose *Debug, Windows* to open these windows.)

- The Autos tab displays variables *and expressions* from the current line of code, and the preceding line of code. See Figure 15.
- The Locals tab (see Figure 16) displays all local variables in the current block of code. If you are inside of a function, the locals tab will display the function parameters and locally defined variables.
- The Watch tab (see Figure 17) display a variable state until you explicitly remove it from the window. You can add a variable to the watch window by right clicking a variable and selecting "Add Watch".

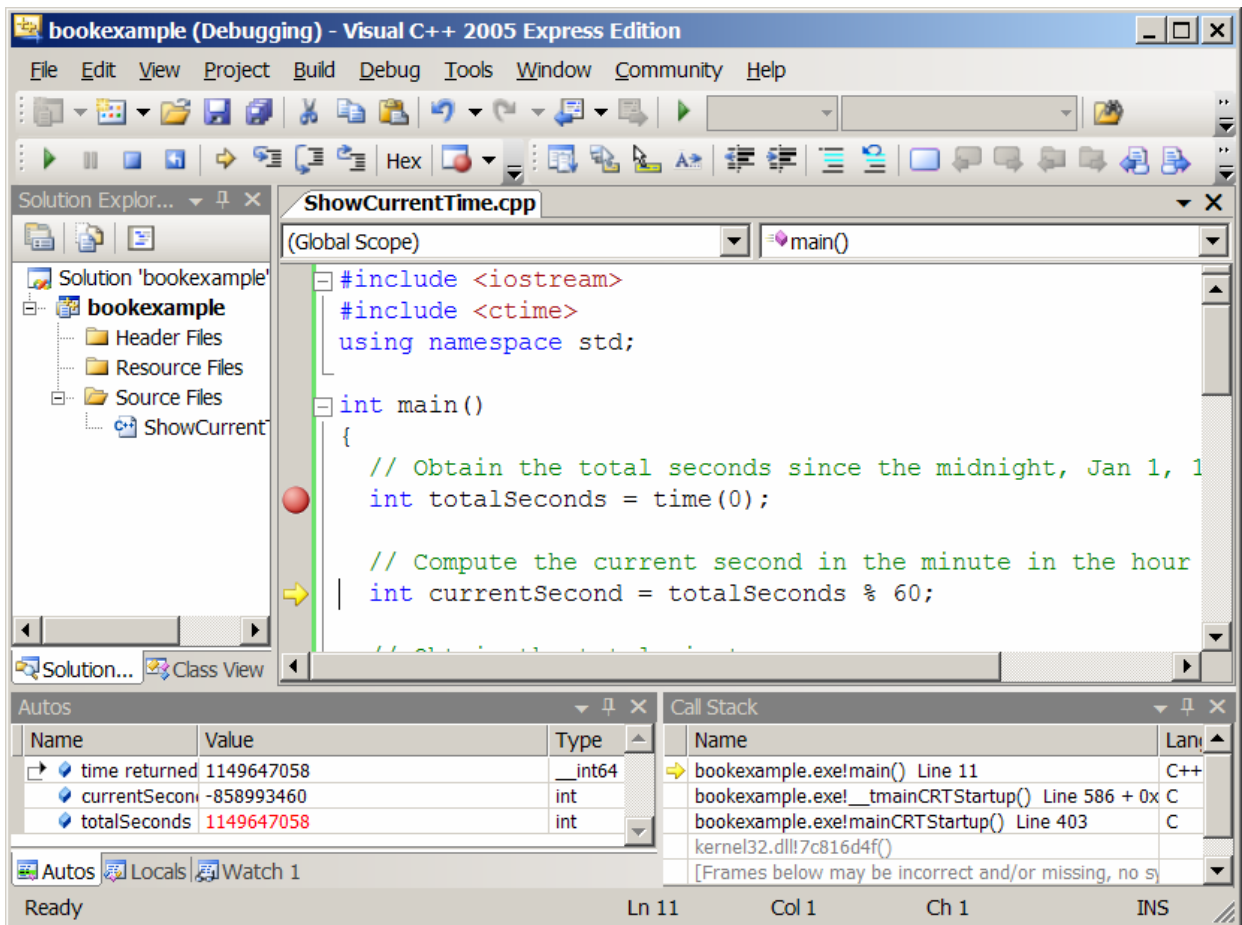


Figure 15

The Autos window displays the variable in the current line.

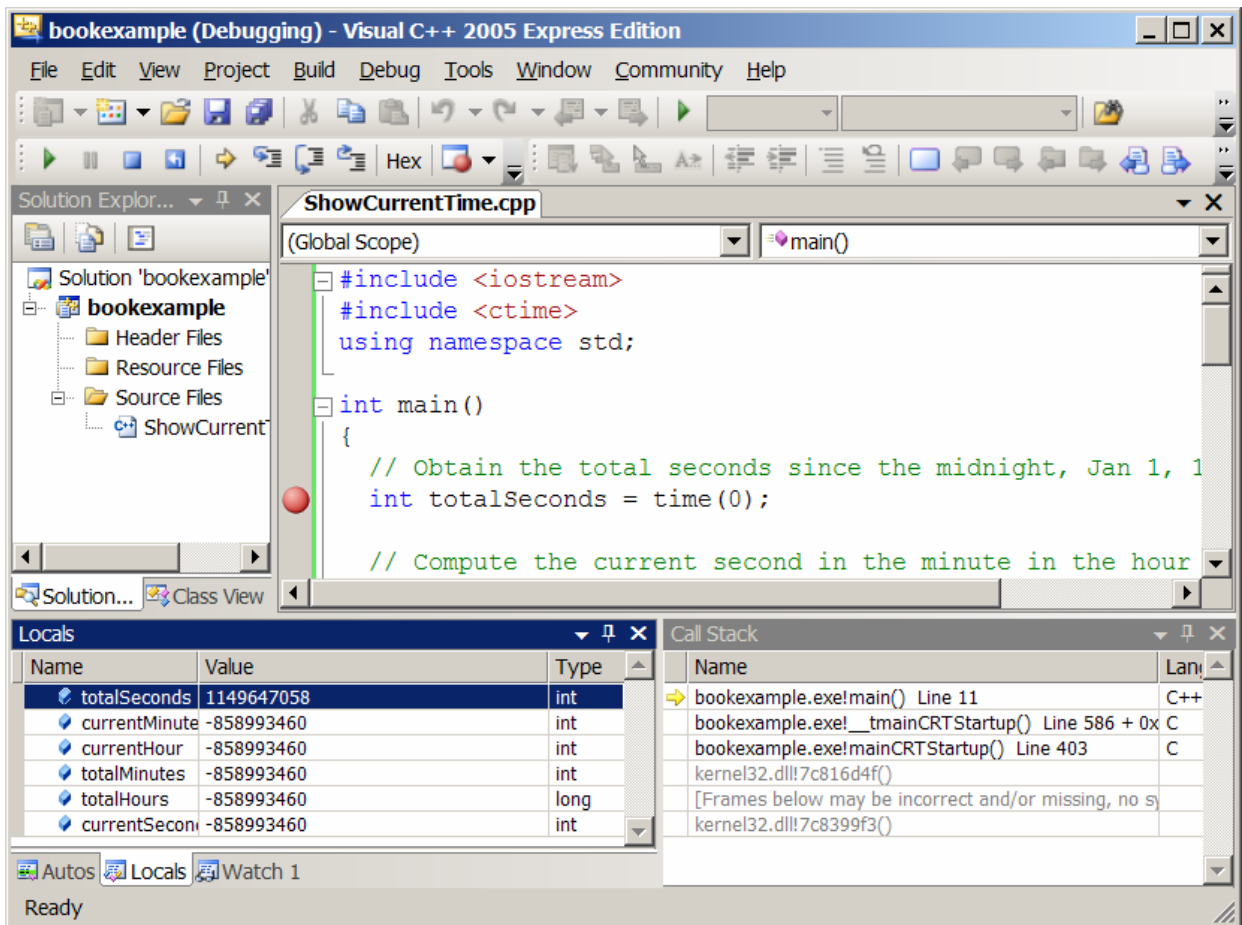


Figure 16

The Locals window displays all variables in the block.

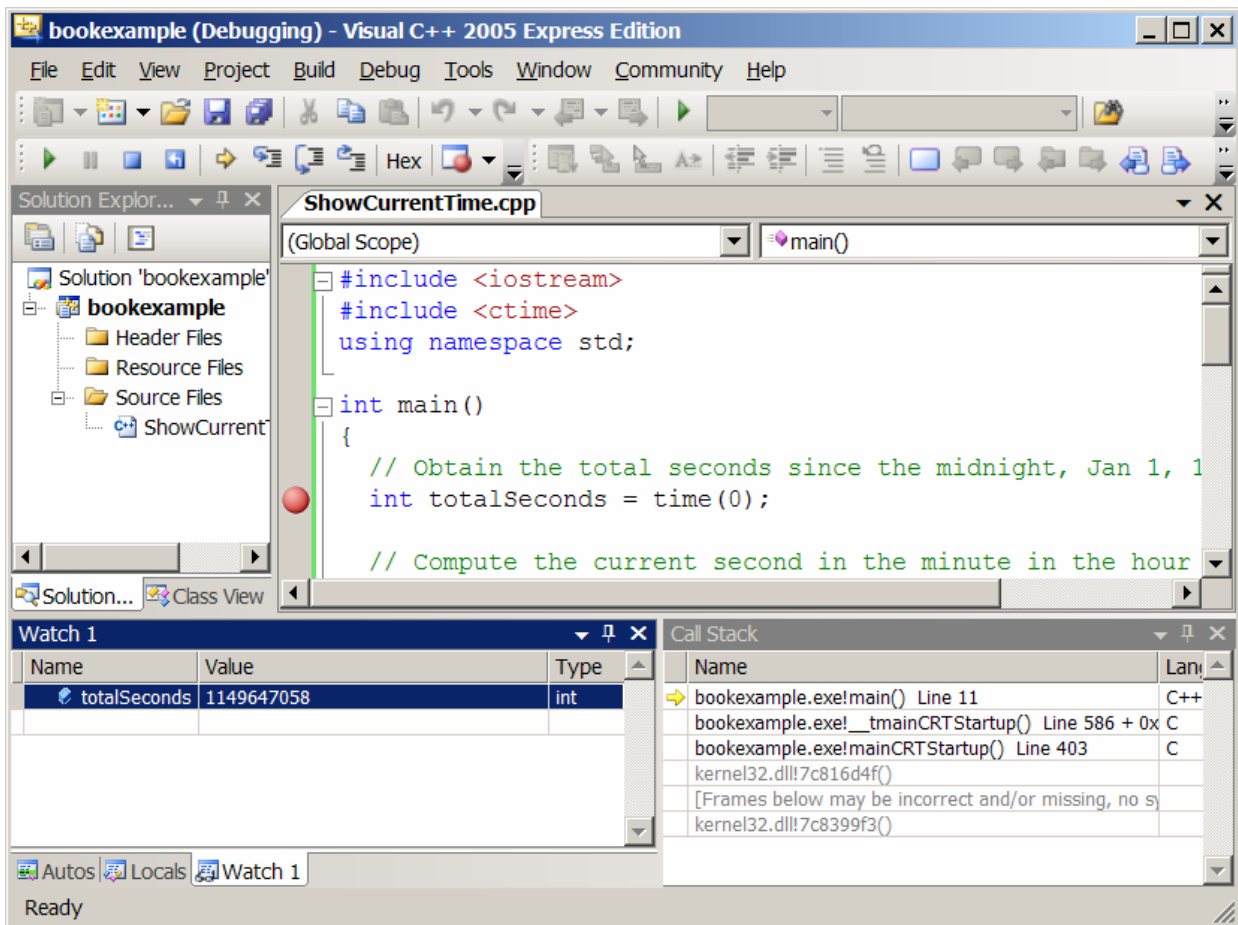


Figure 17

The Watches window displays the variables you want to watch for.

7.3 Controlling Program Execution

The program pauses at a line called the *current execution point*. This line is highlighted and has a yellow arrow to the left. The execution point marks the next line of source code to be executed by the debugger.

When the program pauses at the execution point, you can issue debugging commands to control the execution of the program. You also can inspect or modify the values of variables in the program.

When VC++ is in the debugging mode, the *Debug* menu contains the debugging commands (see Figure 18). Most of the commands also appear in the toolbar under the message pane. The toolbar contains additional commands that are not in the *Run* menu. Here are the commands for controlling program execution:

- **Step Over** executes a single statement. If the statement contains a call to a function, the entire function is executed without stepping through it.
- **Step Into** executes a single statement or steps into a function.
- **Step Out** executes all the statements in the current function and returns to its caller.

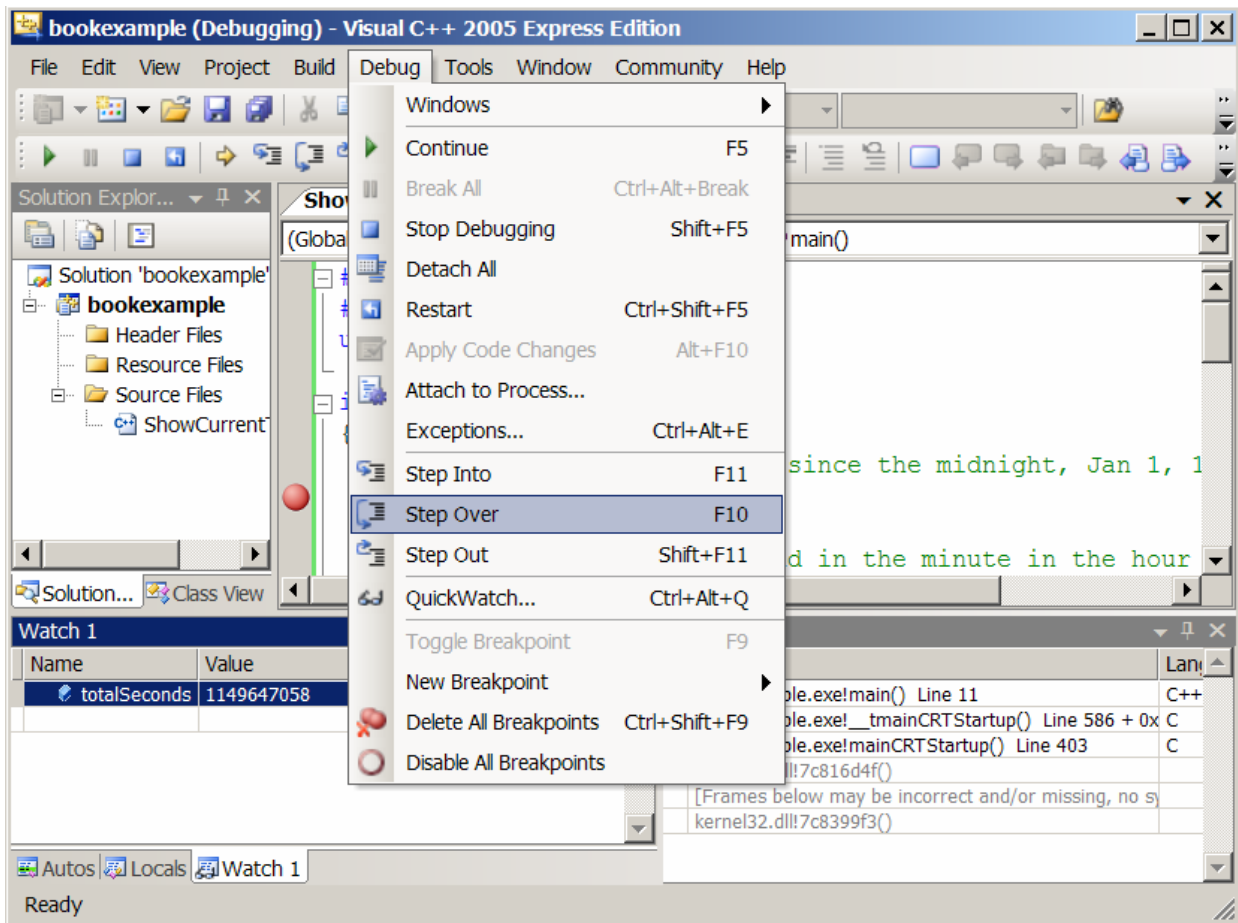


Figure 18

The debugging commands appear under the Debug menu.

NOTE: The debugger is an indispensable, powerful tool that boosts your programming productivity. It may take you some time to become familiar with it, but your investment will pay off in the long run.

8 Including Header File from Different Directories

If you need to include a header file from a directory that is different from the program directory, you need to add the directory in the project properties. For example, suppose `Test.cpp` needs to include `Temp.h` and `Temp.h` is in `c:\teacher`. Follow the steps below to add `c:\teacher` in the project property dialog box:

1. Right-click on `Test.cpp` in the Solution explorer to display a context dialog box, as shown in Figure 19.
2. Choose *Properties* to display Project Properties dialog box, as shown in Figure 20.
3. Enter `c:\teacher` in the *Additional Include Directories* field.

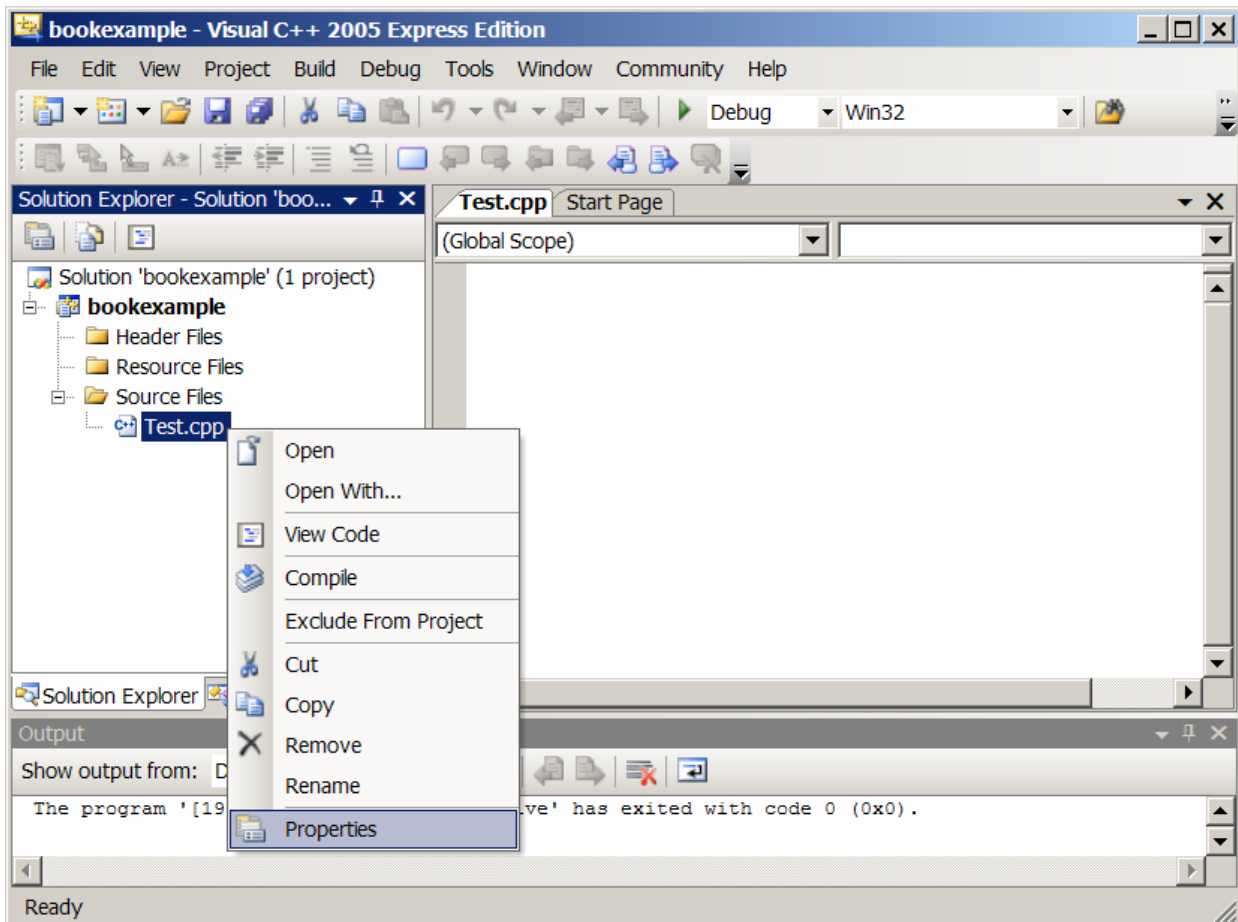


Figure 19

You can customize project properties in VC++.

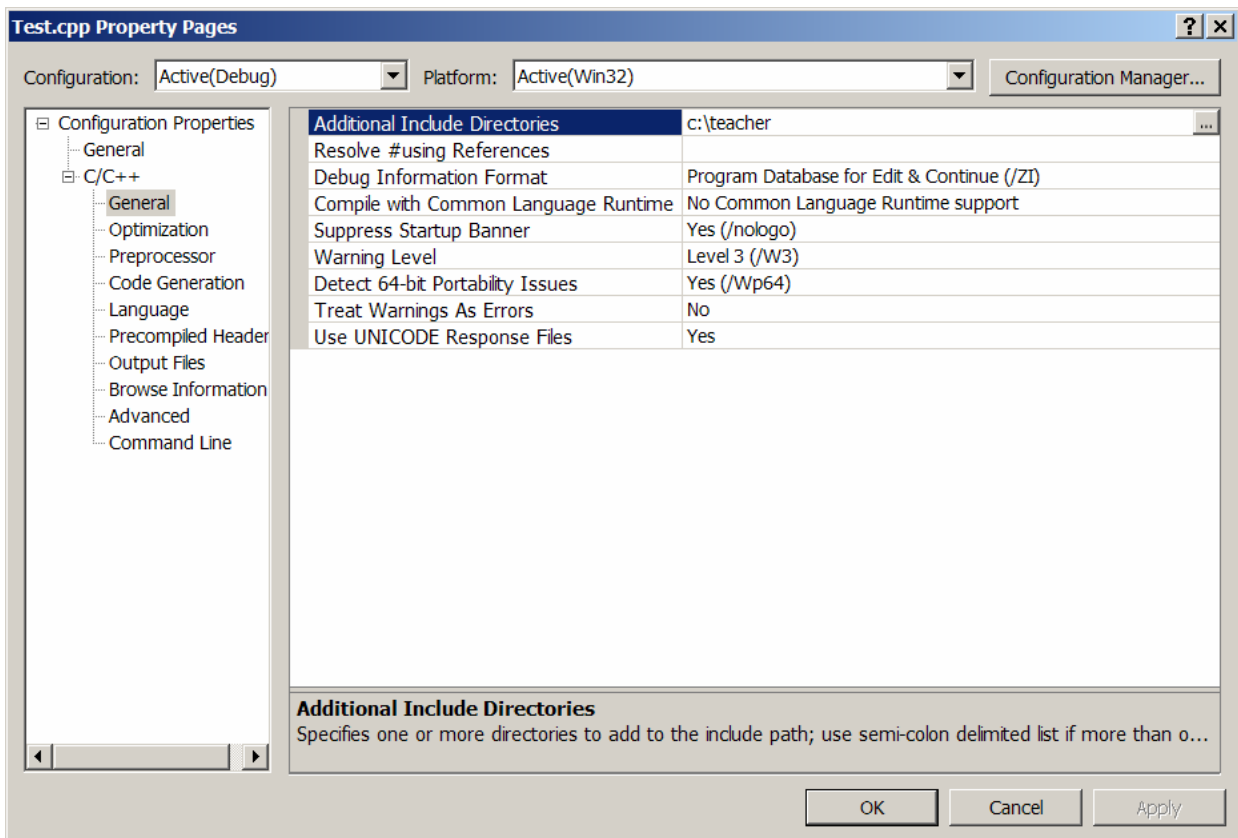


Figure 20

C:\teacher is added in the path.