Appendix M: Introduction to Microsoft Visual C++ 2010 Express Edition

This book may be ordered from Addison-Wesley in a value pack that includes Microsoft Visual C++ 2010 Express Edition. Visual C++ 2010 Express Edition is a powerful but lightweight C++ development environment that is ideal for learning C++. For more information on the value pack, contact your Addison-Wesley representative or send an email to computing@pearson.com.

This appendix serves as a quick reference for performing the following operations using Microsoft Visual C++ 2010 Express Edition

- Starting a new project and entering code
- Saving a project to disk
- Compiling and executing a project
- Closing a project
- Opening an existing project
- Creating a multi-file project
- Removing files from a project
- Adding example source files to a project
- Determining where data files created by a program are located

Starting a New Project

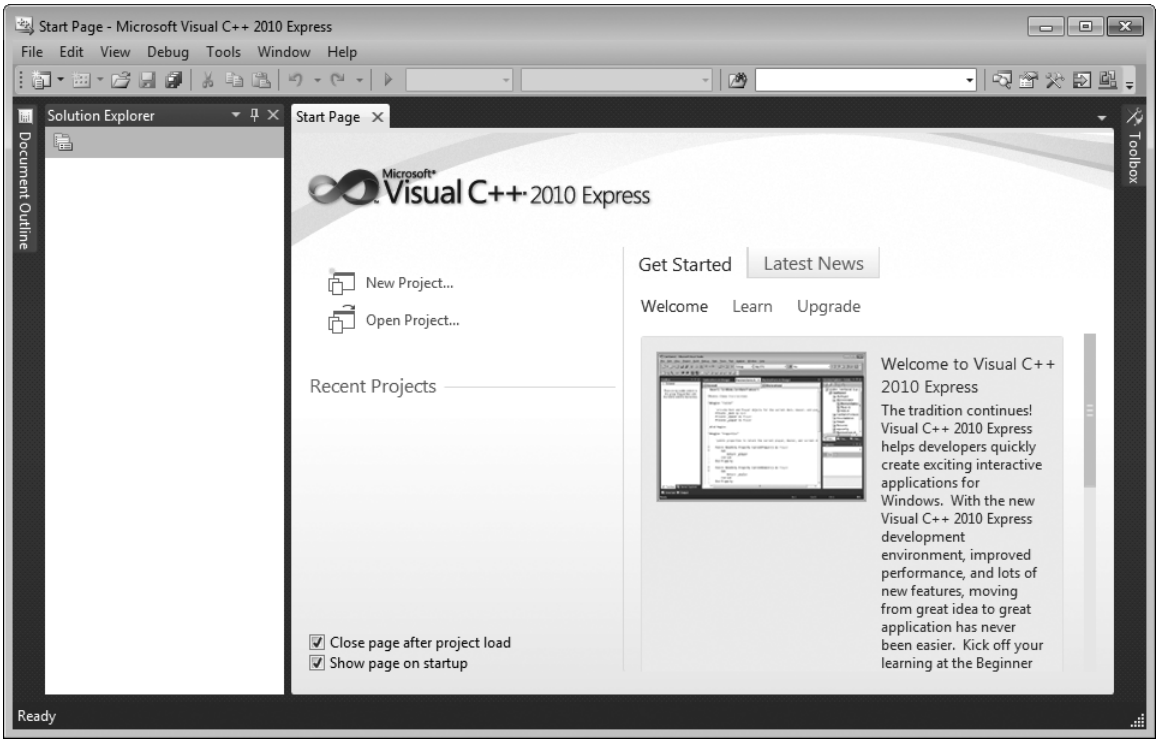
The first step in creating a program with Visual C++ 2010 is to start a project. A *project* is a group of one or more files that make up a software application. (Even if your program consists of no more than a single source code file, it still must belong to a project.)

To start a project:

1. Visual C++ 2010 Express Edition appears similar to Figure M-1 when it first starts up. The screen shown in the figure is the Start Page. The Start Page typically displays information about new software releases, upcoming conferences and training events, links to recent articles, etc.

   To start a new project, click **File** on the menu bar, then **New**, then click **Project**.

Figure M-1



2. The New Project dialog box appears as shown in Figure M-2. This dialog box allows you to select the type of project you are creating, select a project template, and enter a name for the project.

   The type of project that you will normally create while learning C++ is a Win32 Console Application. Under Installed Templates (the left pane) select **Win32**. Under the adjacent pane (the right pane), select **Win32 Console Application**.

3. Each project must have a name. When your instructor gives you a programming assignment, you will probably want to give the project a name that identifies the assignment, such as Lab6, or Assignment7. Enter the name of the project in the Name text box, and then click the **Ok** button to continue. (Do not enter an extension for the filename.)

NOTE: When you create a project, Visual C++ creates a folder where all the project files are normally stored. The folder is called the *project folder* and it has the same name as the project. The Location text box lets you specify a location on your system where the project folder will be created. You will probably want to use the default location, but if not, click the **Browse...** button to select a different one.

Once you have selected the project type and template, and entered a name for the project, the dialog box should appear similar to Figure M-3. In the figure, the name Lab6 is given to the project.

Figure M-2

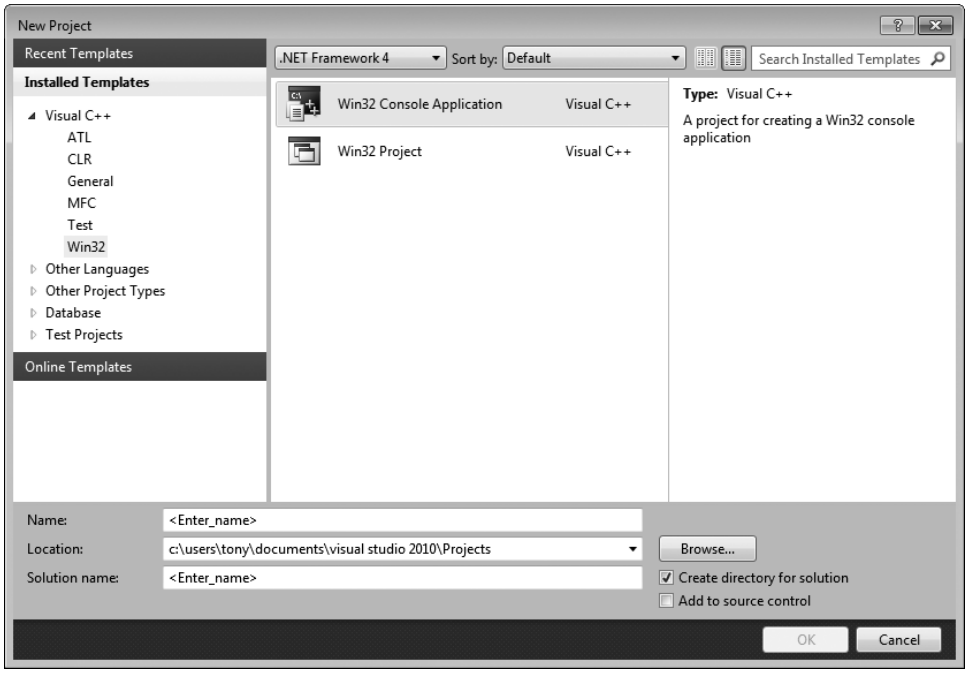
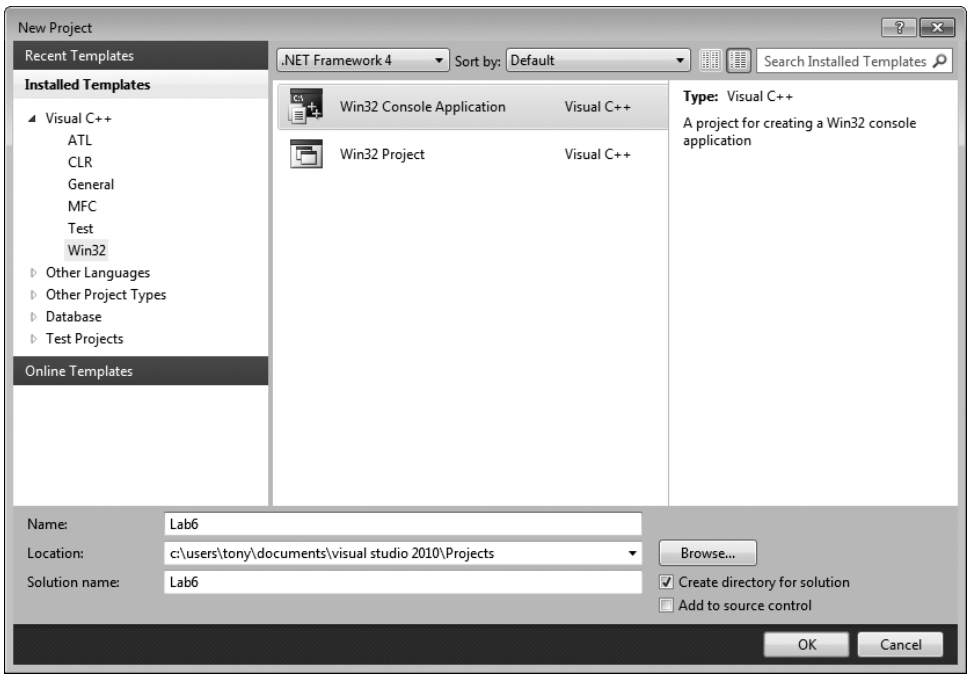


Figure M-3



4. The Win32 Application Wizard starts and you should see the dialog box shown in Figure M-4. Click the **Application Settings** options. The dialog box should now appear as shown in Figure M-5. Make sure that **Console Application** and **Empty Project** are both selected, as shown in the figure. Click the **Finish** button to create the project.

Figure M-4

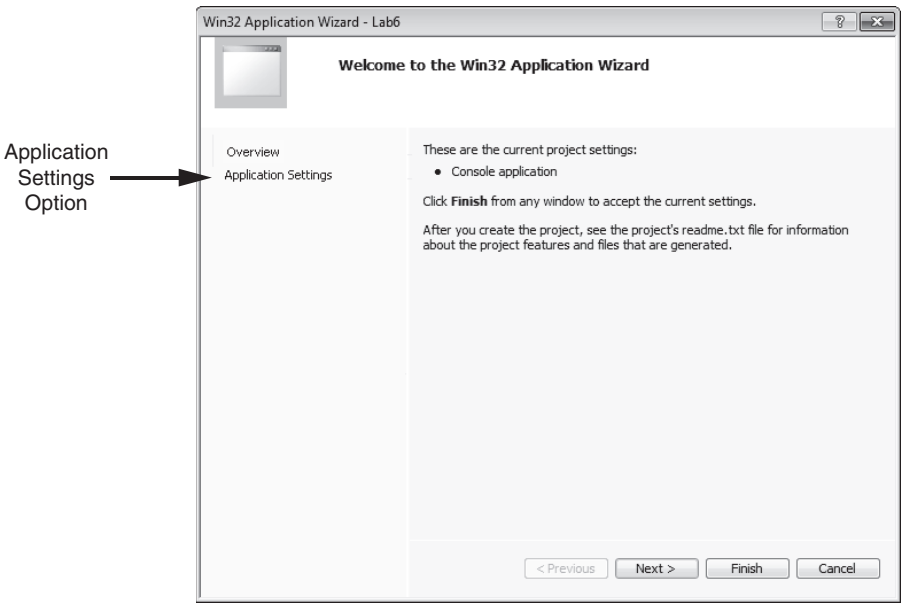
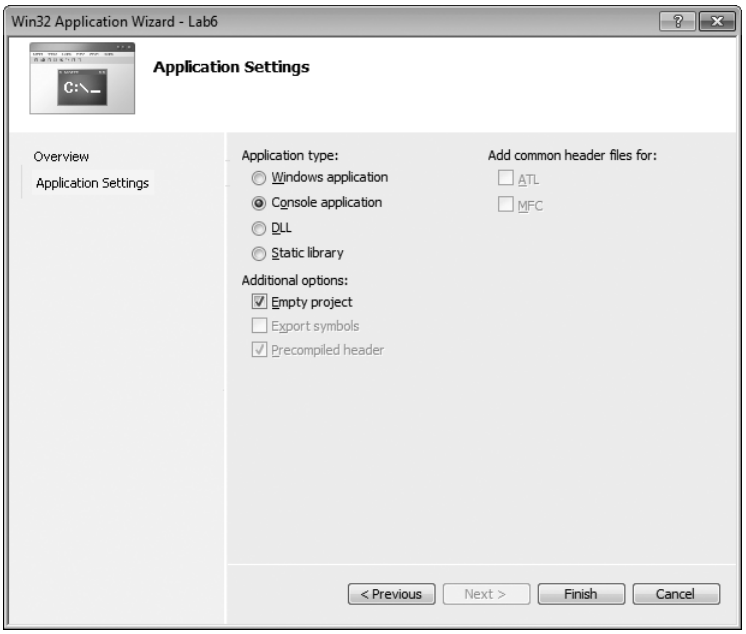
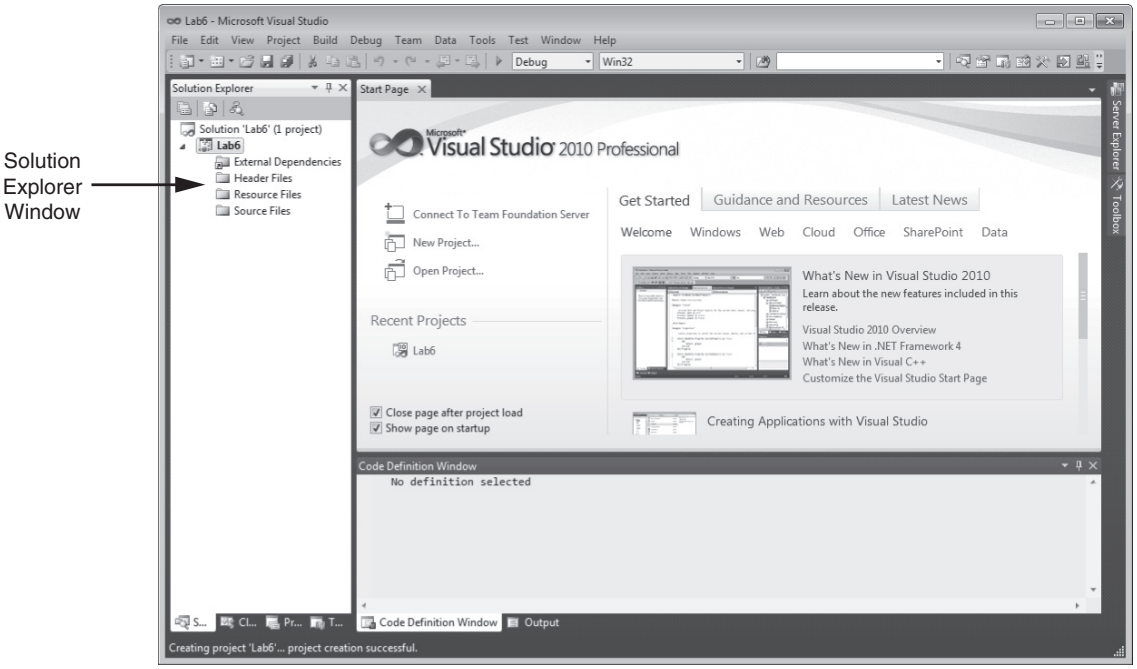


Figure M-5



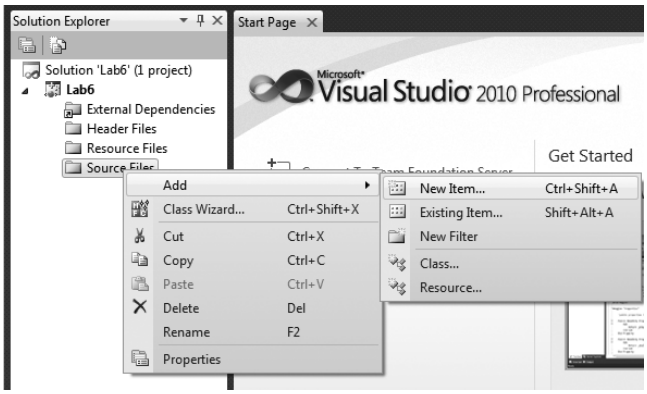
5. You should be back at the main screen, as shown in Figure M-6. Now you will notice that items appear in the left pane, which is called the Solution Explorer window. In particular, you will notice folders labeled **External Dependencies, Header Files, Resource Files,** and **Source Files.** These folders are used to organize the files that you add to the project.

Figure M-6



6. Now you need to insert a new C++ source file into the project. In the Solution Explorer window, *right-click* the Source Files folder. On the menu that pops up select **Add**, then select **New Item...** as shown in Figure M-7.

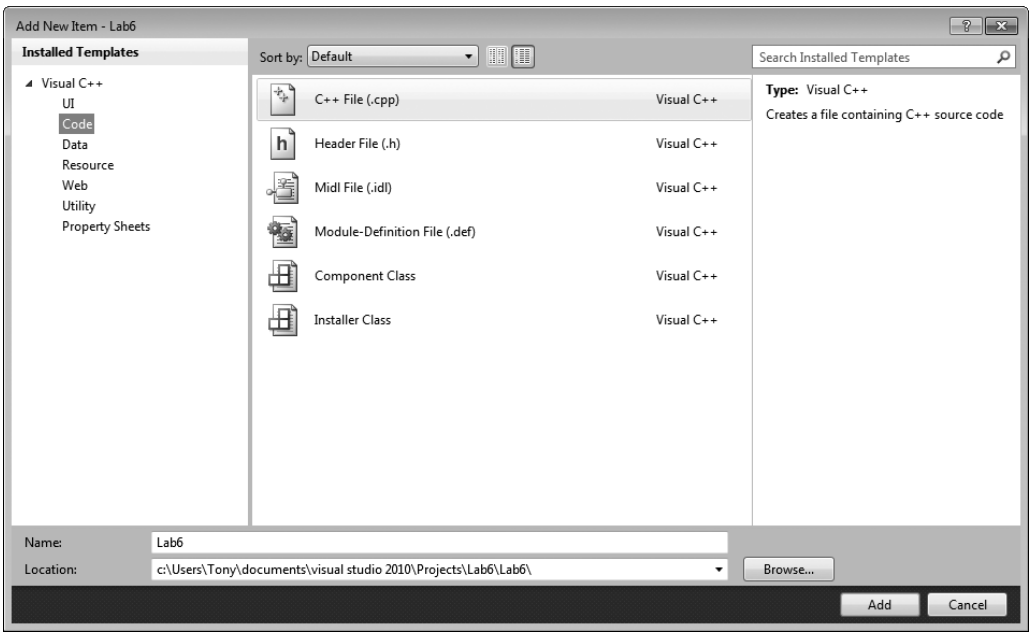
Figure M-7



7. You should now see the Add New Item dialog box shown in Figure M-8. In the Installed Templates pane select **Code**, and in the adjacent pane select **C++ File (.cpp)**. In the Name box enter the name of the source file. (This can be the same as the project name, if you wish.) Figure M-8 shows an example of the dialog box with this step completed. Click the **Add** button to create the source file and add it to the project.

NOTE: Notice in the figure that Lab6 was entered as the source file's name. Visual C++ automatically adds the `.cpp` extension. As a result, a source file named `Lab6.cpp` will be created in the project folder.

Figure M-8



8. You have now created an empty source file and added it to the project. The C++ editor should now appear, as shown in Figure M-9. Note that above the text editor is a tab showing the name of the file that is currently open. There is also a tab for the Visual C++ Start Page. You can click the tabs to switch your view between the file and the Start Page.

   Also notice that in the Solution Explorer window an entry for the source file you just created now appears under the Source Files folder.

   The C++ editor is where you enter the code for your program. Figure M-10 shows the screen after a program has been entered into the editor.

Figure M-9

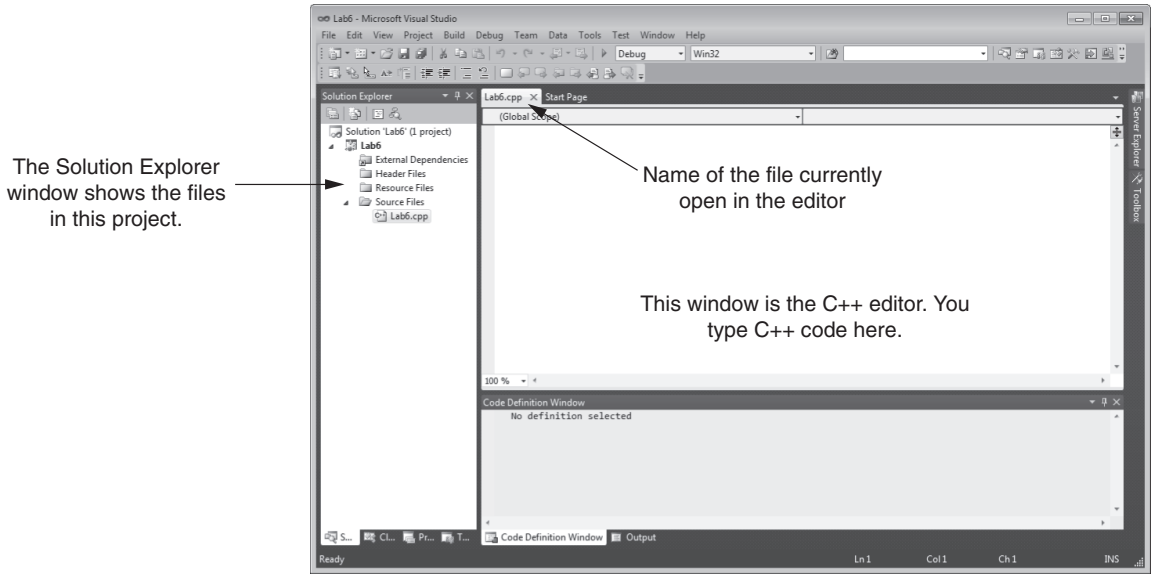
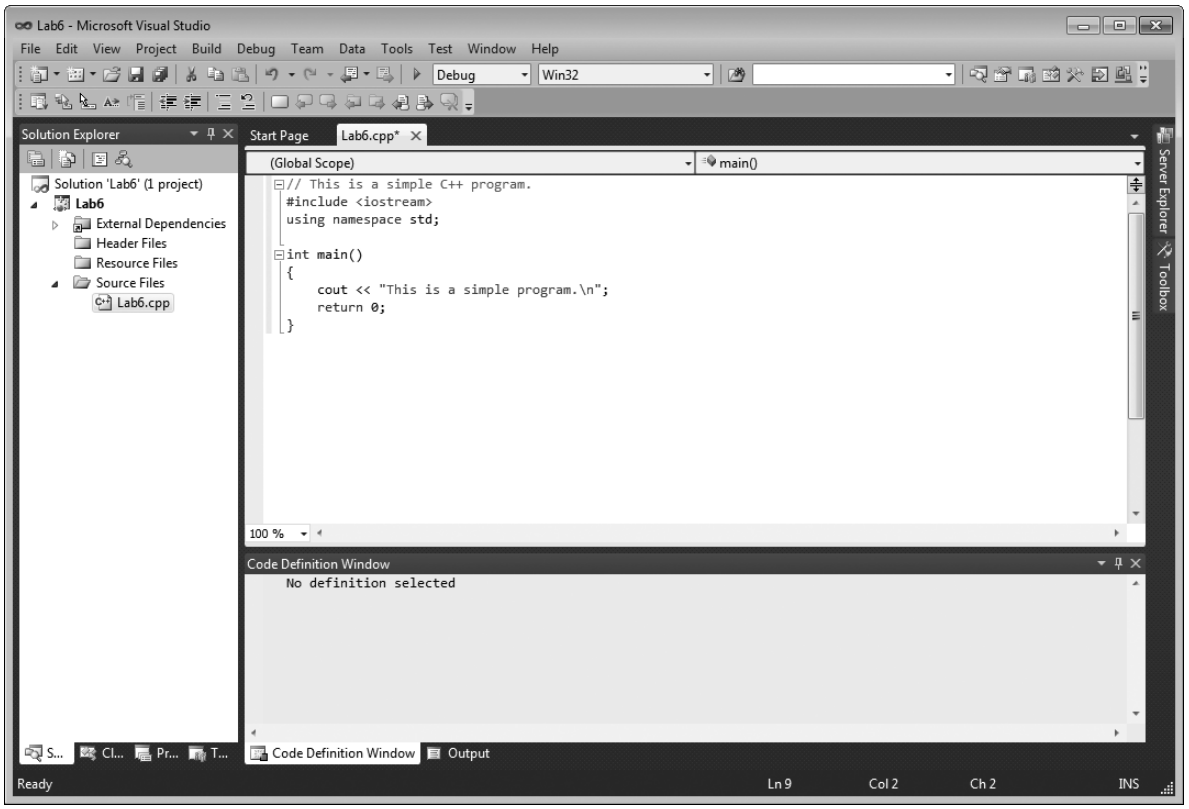


Figure M-10



Saving Your Project to Disk

It is important to periodically save your work. To do so, perform any one of the following:

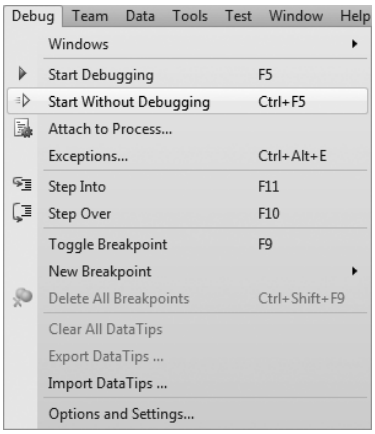
- Click **File** on the menu bar, then click **Save All** on the File menu.
- Click the Save All button ( ) on the tool bar that appears just below the menu bar.
- Press **Ctrl+Shift+S**

Compiling and Executing

Once you have entered a program's source code, you may compile and execute it by any of the following methods:

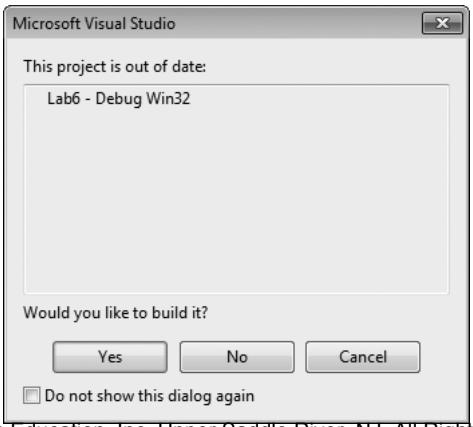
- Pressing **Ctrl+F5**
- Click **Debug** on the menu bar and select **Start without Debugging** as shown in Figure M-11.

Figure M-11



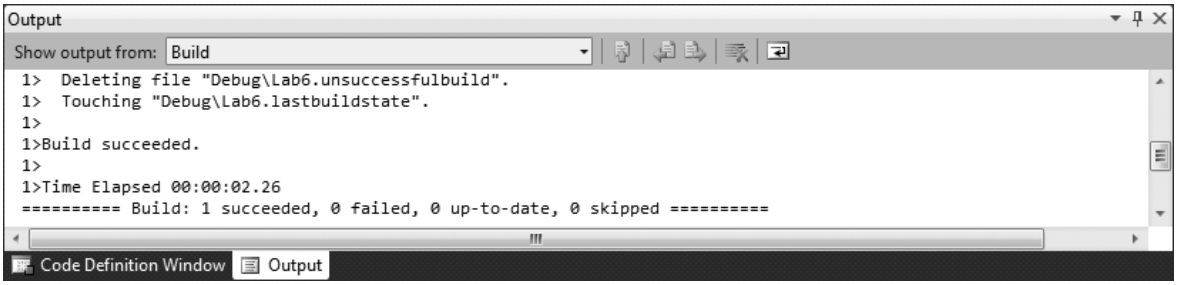
After performing one of these steps, you may see the dialog box in Figure M-12. If you see this dialog box click the **Yes** button.

Figure M-12



The Output window shown in Figure M-13 appears at the bottom of the screen while the program is being compiled. This window shows various status messages as the program is compiled. Error messages are also displayed in this same area of the screen.

Figure M-13



If There Are No Errors

If the program compiles successfully, it will begin to execute. The program's output will appear in a console window, such as that shown in Figure M-14.

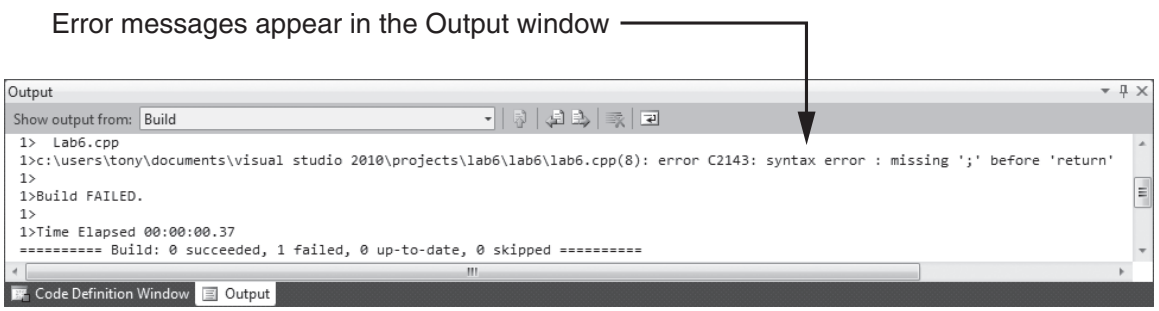
Figure M-14

If There Are Errors

If a program has syntax errors you will see the dialog box in Figure M-15 when you compile the program. Click the **No** button. To see a list of error messages, look in the Output window at the bottom of the screen as shown in Figure M-16. It's common to have several error messages displayed. If this is the case, scroll up in the window until you find the first error message.

Figure M-15

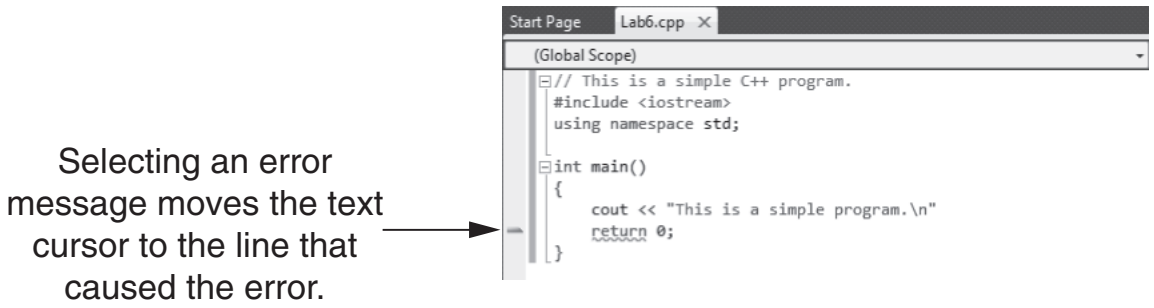
Figure M-16



When you find the first error message, double-click it. This selects the error message and causes it to appear highlighted as shown in Figure M-17. When you select an error message the editor will move the text cursor to the line of code where the error was encountered, and a small marker will appear in the left margin next to the line. (Syntax errors will also be shown with a wavy red underline.) For example, look at the program in Figure M-18. The `cout` statement is missing a semicolon. By double-clicking the error message, the text cursor is positioned on the line with the error. (Actually, in this case the cursor appears on the line after the statement with the missing semicolon. The compiler did not detect that the semicolon was missing until it encountered the beginning of the next line. Finding errors is not always straightforward.)

Figure M-17

Figure M-18



Closing a Project

To close the project that you currently have open, click **File** on the menu bar, and then click **Close Solution**.

Opening an Existing Project

To open an existing project, click File on the menu bar, then select Open, then select Project/Solution. Use the resulting dialog box to browse to the folder containing your project. In that folder you should see a solution file. This file has the same name that you gave the project, and the `.sln` extension. Double-click this file to open the project.

NOTE: If the source file does not open in the editor, double-click the name of the file in the Solution Explorer window.

Creating a Multi-File Project

Many of your programs will consist of more than one file. For example, suppose your instructor has asked you to write a simple class named `Pet`. An instance of the class will hold the name of your pet. You decide to write the following specification and implementation files.

Contents of `Pet.h`

```
1  // Pet class specification file
2  #ifndef PET_H
3  #define PET_H
4  #include <string>
5  using namespace std;
6
7  class Pet
8  {
9  private:
10    string name;   // To hold the pet's name
11
12 public:
13    // Constructor
14    Pet(string);
15
16    // Accessor function for name
17    string getName() const;
18 };
19
20 #endif
```

Contents of `Pet.cpp`

```
1  // Pet class implementation file
2  #include <string>
3  #include "Pet.h"
4  using namespace std;
5
6  // Constructor
7  Pet::Pet(string str)
8  {
9     name = str;
10 }
11
12 // getName member function
13 string Pet::getName() const
14 {
15    return name;
16 }
```

To demonstrate the class, you decide to write the code in Program M-1.

Program M-1

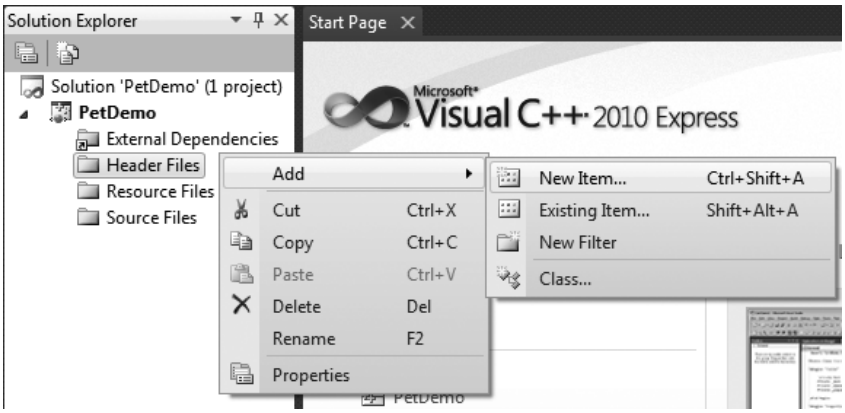
```
1  #include <iostream>
2  #include "Pet.h"
3  using namespace std;
4
5  int main()
6  {
7     // Create an instance of the Pet class.
8     Pet myPet("Fido");
9
10    // Display my pet's name.
11    cout << "My pet's name is " << myPet.getName() << endl;
12    return 0;
13 }
```

The following steps show you how to create all three files in the same project.

1. Start Visual C++ 2010 Express Edition. To start a new project, click **File** on the menu bar, then click **New**, then click **Project**.
2. The New Project dialog box appears. Under Installed Templates (the left pane) select **Win32**. In the adjacent pane (the right pane), select **Win32 Console Application**.
3. We will name this project `PetDemo`. Enter **`PetDemo`** in the Name text box. Click the **Ok** button to continue.
4. You should see the Win32 Application Wizard dialog box. Click the **Application Settings** options. In the next dialog box make sure that **Console Application** and **Empty Project** are both selected. Click the **Finish** button to create the project.
5. You should be back at the main screen. Now you will notice that folders labeled **External Dependencies**, **Header Files**, **Resource Files**, and **Source Files** appear in the Solution Explorer window. These folders are used to organize the various files that you will add to the project.
6. The first file that we will add to the project is the `Pet.h` file. Because this is a header file (ending with `.h`) we will add it to the Header Files folder in the Solution Explorer.

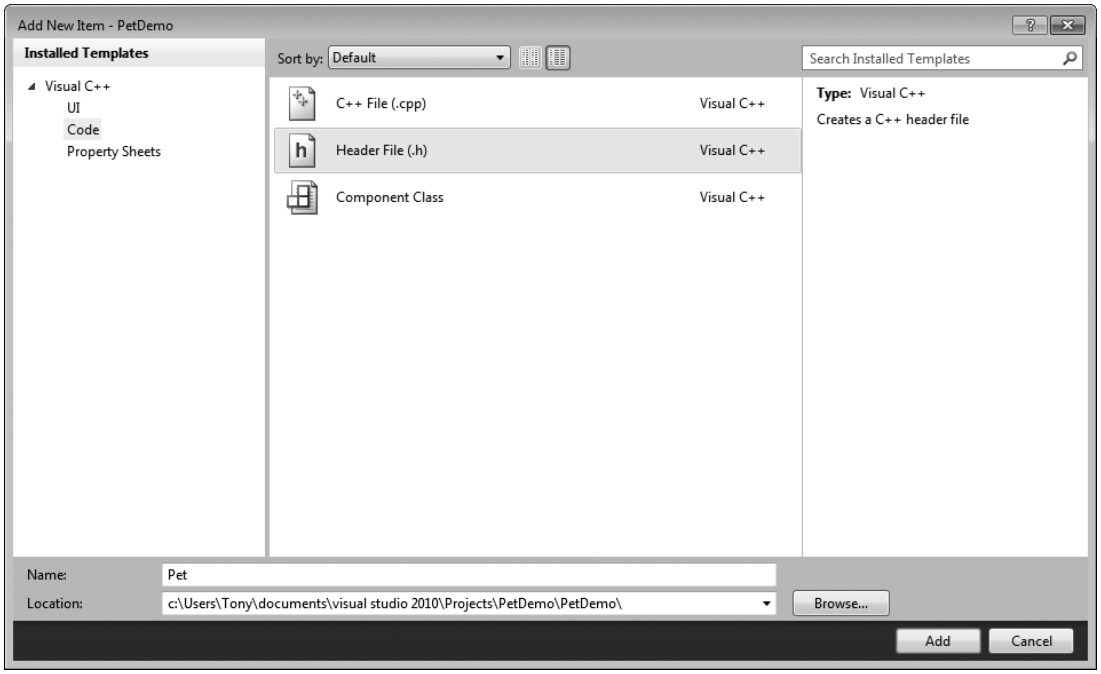
In the Solution Explorer window, *right-click* the Header Files folder. On the menu that pops up select **Add**, then select **New Item...** as shown in Figure M-19.

Figure M-19



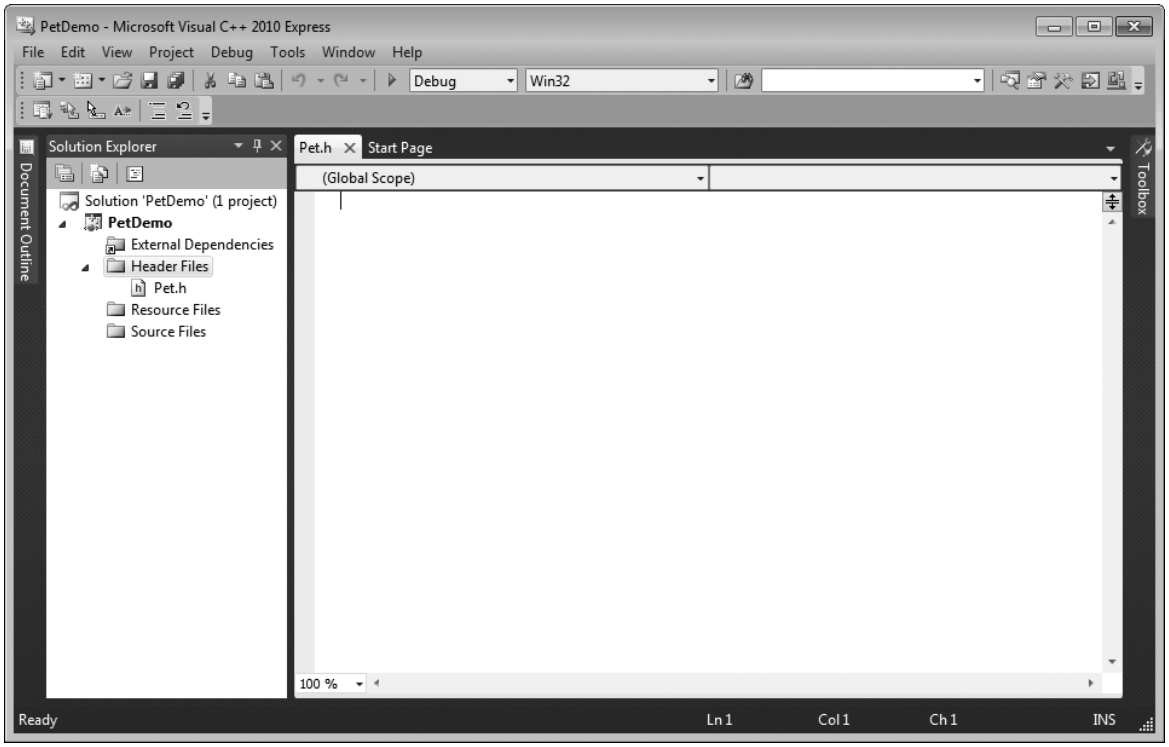
7. You should now see the Add New Item dialog box shown in Figure M-20. In the Installed Templates pane select **Code**, and in the adjacent pane select **Header File (.h)**. In the Name box enter the name of the header file. Because Visual C++ automatically adds the .h extension, you only need to enter **`Pet`**. Click the **Add** button to create the `Pet.h` header file and add it to the project.

Figure M-20



8. You have now created an empty header file named `Pet.h` and added it to the project. The C++ editor should now appear, as shown in Figure M-21. Note that above the text editor is a tab showing the name of the file that is currently open. Also notice that in the Solution Explorer window an entry for the `Pet.h` file now appears under the Header Files folder.

Figure M-21



9. Type the code for the `Pet.h` file into the editor. Figure M-22 shows the screen with the code entered. Be sure to save the file.
10. The next file that we will add to the project is the `Pet.cpp` file. Because this is a regular source code file (ending with `.cpp`) we will add it to the Source Files folder in the Solution Explorer.

    In the Solution Explorer window, *right-click* the Source Files folder. On the menu that pops up select **Add**, then select **New Item...** You should now see the Add New Item dialog box shown in Figure M-23. In the Installed Templates pane select **Code**, and in the adjacent pane select **C++ File (.cpp)**. In the Name box enter the name of the source file. Since Visual C++ automatically adds the `.cpp` extension, you only need to enter **`Pet`**. Click the **Add** button to create the `Pet.cpp` source file and add it to the project.

Figure M-22

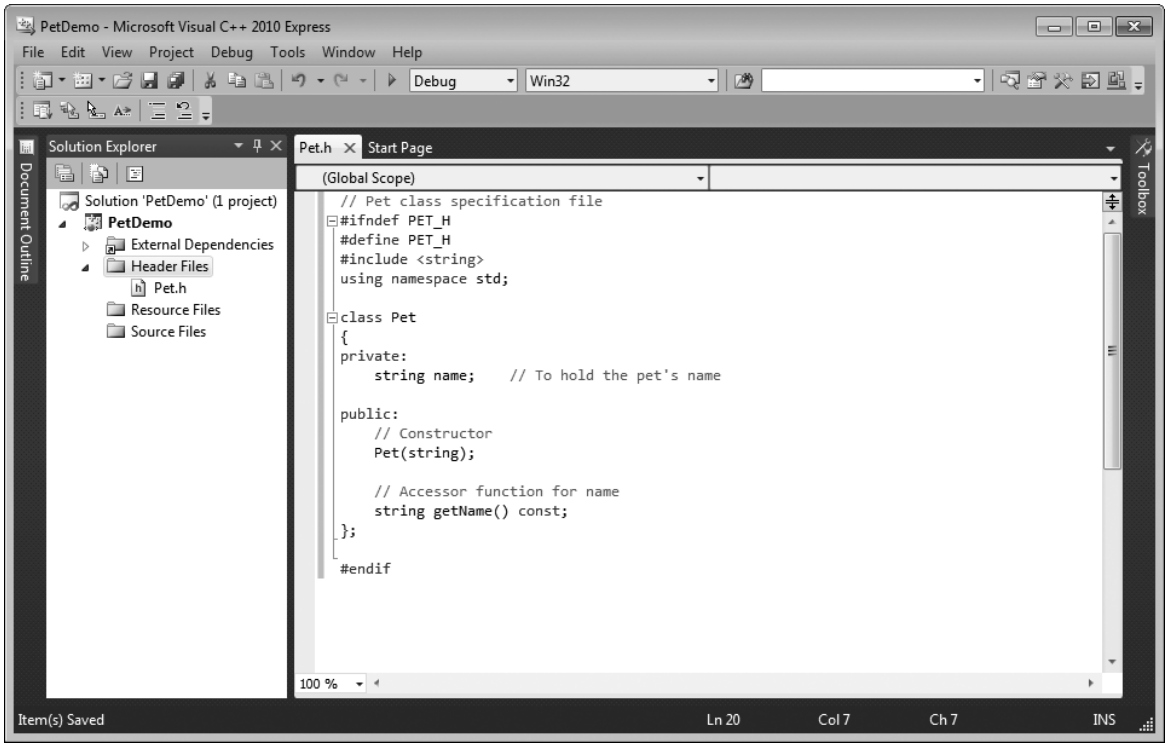
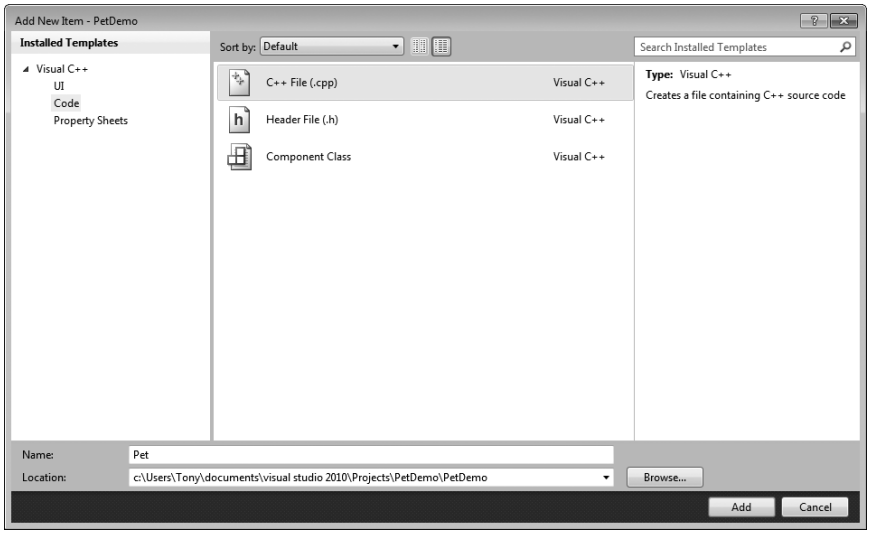
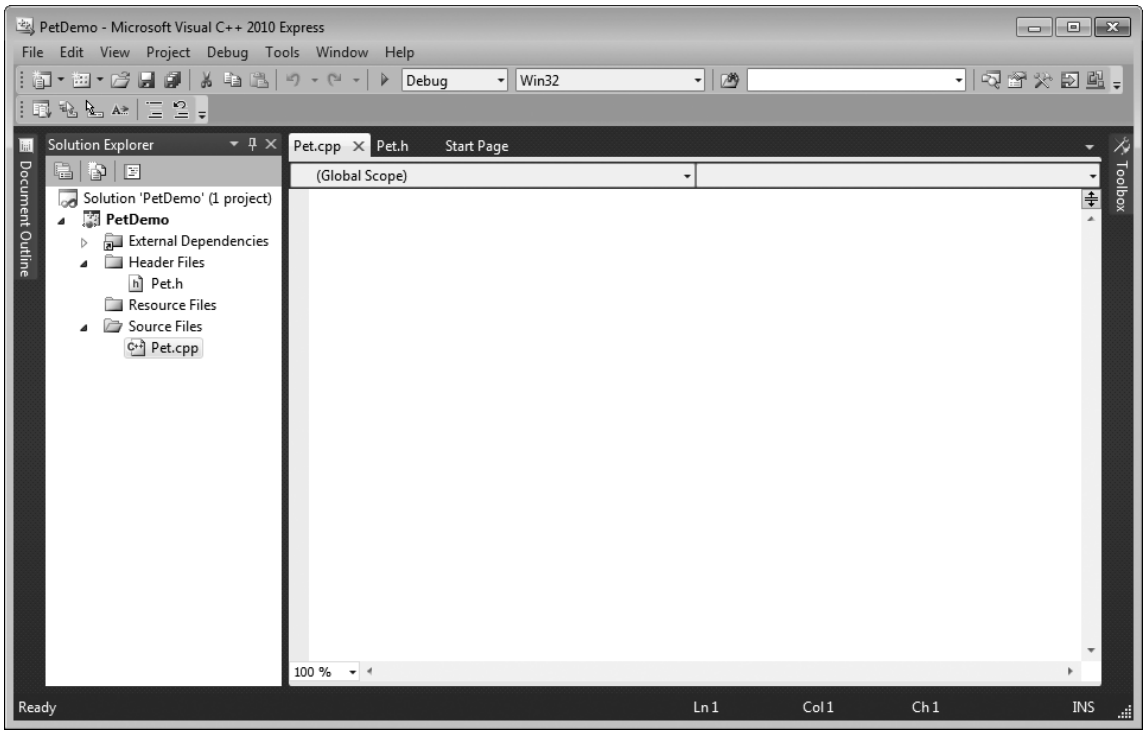


Figure M-23



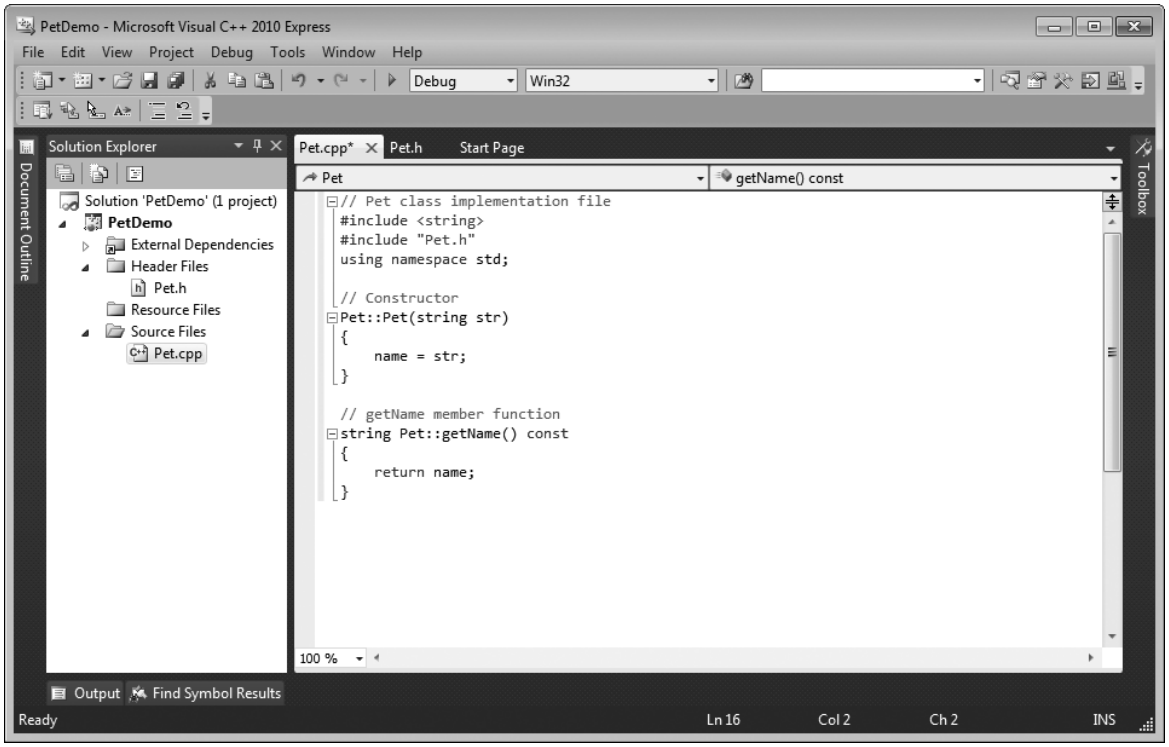
11. You have now created an empty source code file named `Pet.cpp` and added it to the project. The C++ editor should now appear, as shown in Figure M-24. Note that three tabs now appear above the text editor: one for `Pet.cpp`, one for `Pet.h`, and one for the Start Page. You can click any of these tabs to switch between the two files and the Start Page. Also notice that in the Solution Explorer window an entry for the `Pet.cpp` source file now appears under the Source Files folder.

Figure M-24



12. Type the code for the `Pet.cpp` file into the editor. Figure M-25 shows the screen with the code entered. Be sure to save the file.

Figure M-25



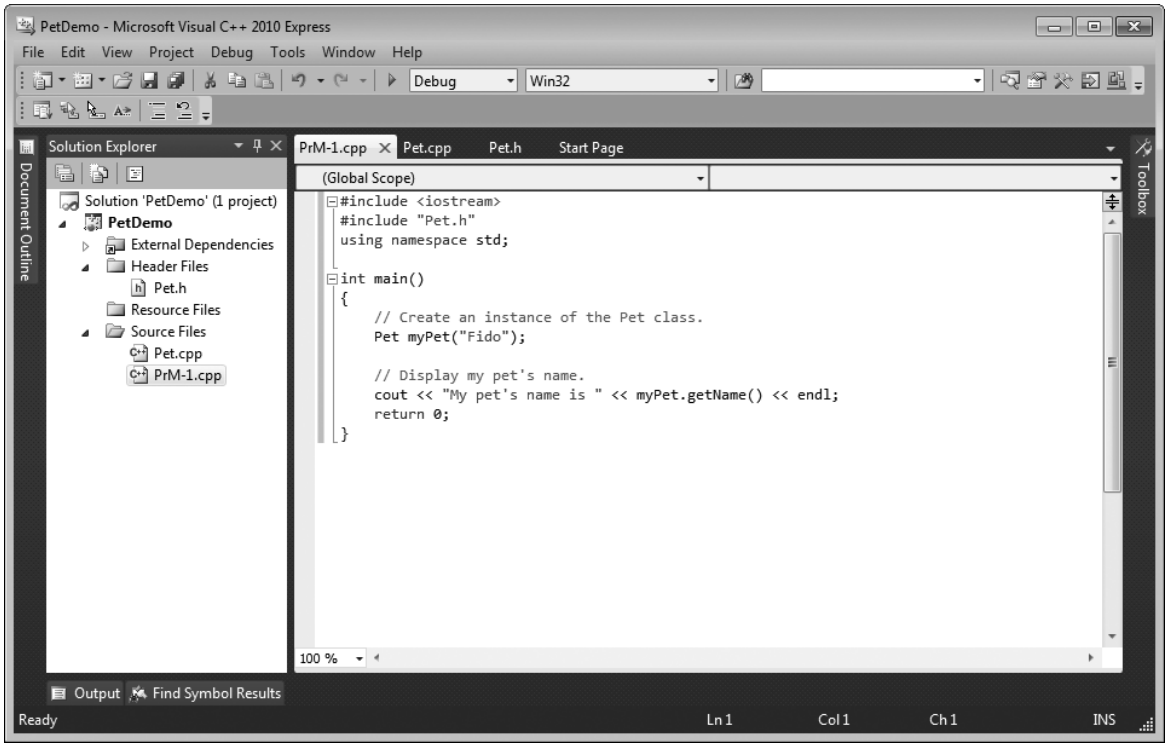
13. The next file that we will add to the project is the code shown in Program M-1. Because this is a regular source code file (ending with `.cpp`) we will add it to the Source Files folder in the Solution Explorer.

    In the Solution Explorer window, *right-click* the Source Files folder. On the menu that pops up select **Add**, then select **New Item...** You should now see the Add New Item dialog box. In the Installed Templates pane select **Code**, and in the adjacent pane select **C++ File (.cpp)**. In the Name box enter **`PrM-1`**, which is the name we will give the source file. Click the **Add** button to create the `PrM-1.cpp` source file and add it to the project.

    You have now created an empty source code file named `PrM-1.cpp` and added it to the project. Note that four tabs now appear above the text editor: one for `PrM-1.cpp`, one for `Pet.cpp`, one for `Pet.h`, and one for the Start Page. Also notice that in the Solution Explorer window an entry for the `PrM-1.cpp` source file now appears under the Source Files folder.

    Type the code for the `PrM-1.cpp` file into the editor. Figure M-26 shows the screen with the code entered. Be sure to save the file.

Figure M-26



14. All three source code files have been added to the project. Press **Ctrl+F5** to compile and execute the program.

Adding Example Source Files to a Project

If you wish to compile and execute one of the example source files that come with this book, you will have to create a project and add the file to the project. If the example source file is part of a multi-file program, you will have to add all of the files that are part of the program to the project. The following steps lead you through the process. (This tutorial assumes you have downloaded the Appendix M example source code folder from the book's companion Web site. If you have not, do so before performing these steps.)

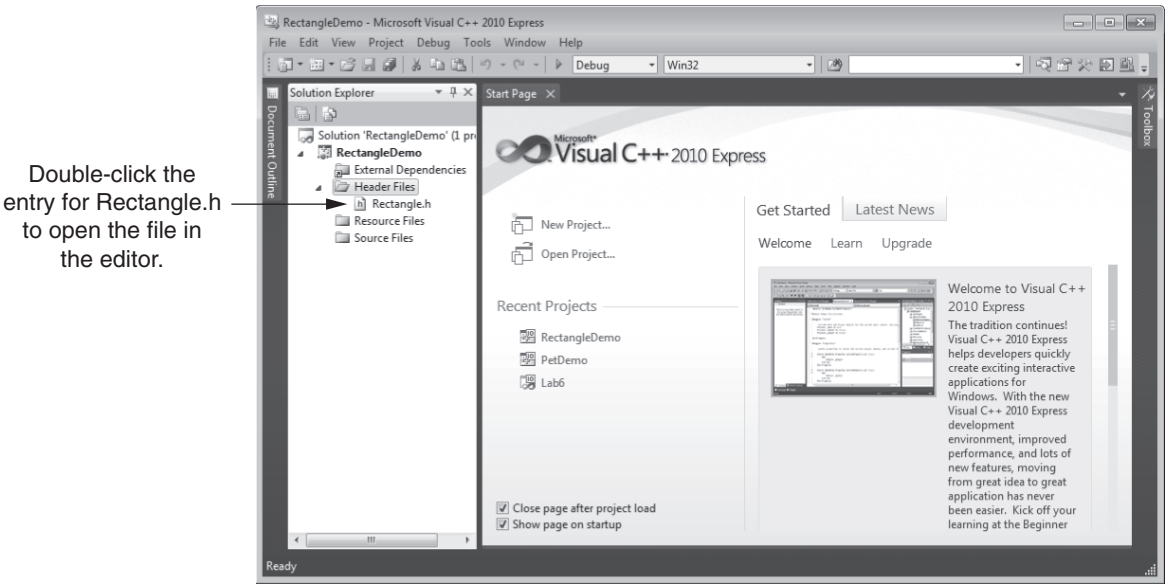
1. Start Visual C++ 2010 Express Edition. To start a new project, click **File** on the menu bar, then click **New**, then click **Project**.
2. The New Project dialog box appears. Under Installed Templates (the left pane) select **Win32**. In the adjacent pane (the right pane), select **Win32 Console Application**.
3. We will name this project `RectangleDemo`. Enter **`RectangleDemo`** in the Name text box. Click the **Ok** button to continue.

4. You should see the Win32 Application Wizard dialog box. Click the **Application Settings** options. In the next dialog box make sure that **Console Application** and **Empty Project** are both selected. Click the **Finish** button to create the project.
5. You should be back at the main screen. Now you will notice that folders labeled **External Dependencies, Header Files, Resource Files**, and **Source Files** appear in the Solution Explorer window. These folders are used to organize the various files that you will add to the project.
6. The first file that we will add to the project is the `Rectangle.h` file. This is the specification file for the `Rectangle` class. Because this is a header file (ending with `.h`) we will add it to the Header Files folder in the Solution Explorer.

   In the Solution Explorer window, *right-click* the Header Files folder. On the menu that pops up select **Add**, then select **Existing Item...**

   You should now see the Add Existing Item dialog box. Browse to the Appendix M example source code folder and select the file `Rectangle.h` and click the **Add** button. You will see an entry for the `Rectangle.h` file appear in the Solution Explorer, as shown in Figure M-27. If the file does not appear in the editor, you can double-click its entry in the Solution Explorer to open it.

Figure M-27



7. The next file that we will add to the project is the `Rectangle.cpp` file. This is the implementation file for the `Rectangle` class. Because this is a regular source code file (ending with `.cpp`) we will add it to the Source Files folder in the Solution Explorer.

   In the Solution Explorer window, *right-click* the Source Files folder. On the menu that pops up select **Add**, then select **Existing Item...**

You should once again see the Add Existing Item dialog box. Browse to the Appendix M example source code folder and select the file `Rectangle.cpp` and click the **Add** button. You will see an entry for the `Rectangle.cpp` file appear in the Solution Explorer. If the file does not appear in the editor, you can double-click its entry in the Solution Explorer to open it.

8. The next file that we will add to the project is the `PrM-2.cpp` file. This is a simple program that tests the `Rectangle` class. Because this is a regular source code file (ending with `.cpp`) we will add it to the Source Files folder in the Solution Explorer.

   In the Solution Explorer window, *right-click* the Source Files folder. On the menu that pops up select **Add**, then select **Existing Item...**

   You should once again see the Add Existing Item dialog box. Browse to the Appendix M example source code folder and select the file `PrM-2.cpp` and click the **Add** button. You will see an entry for the `PrM-2.cpp` file appear in the Solution Explorer. If the file does not appear in the editor, you can double-click its entry in the Solution Explorer to open it.

9. All three source code files have been added to the project. Press Ctrl+F5 to compile and execute the program.

Removing a File from a Project

To remove a file from a project, perform one of the following:

- Select the file's name in the Solution Explorer window and press the **Delete** key.
- Right click the file's name in the Solution Explorer window and then select **Exclude From Project**.

These operations remove the file from the project, but do not delete it from the disk.

Where Files Are Created

When you create a new project, Visual C++ creates a project folder with the same name as the project. This is where all of the files associated with the project are normally stored. You can specify the location of the folder when you create the project. (See step 3 of the Starting a New Project section.)

For example, suppose we've created a project named Lab5, and its project folder is at `C:\MyProjects\Lab5`. All of the C++ source files that you have created for this project will be stored in that folder. In addition, Visual C++ will create another folder named `Debug`, under the `C:\MyProjects\Lab5` folder. The `Debug` folder will contain the project's executable file and its object file(s).

It's important to know the location of the project folder when writing code that opens files for input or output. When running a program from the Visual C++ IDE, the project folder is the default location for all data files. For example, suppose our Lab5 project creates a file with the following code:

```
ofstream outputFile("names.txt");
```

Because we have not specified a path with the file name, the file will be created in the `C:\MyProjects\Lab5` project folder. Similarly, if our program attempts to open an

existing file for reading, it will look in the project folder by default. For example, suppose our Lab5 program attempts to open a file with the following statement:

```
ifstream inputFile("names.txt");
```

Because we have not specified a path, the program will look for the file in the project folder, `C:\MyProjects\Lab5.`

If we want to open a file in a location other than the project folder, we must provide a path as well as a file name. For example, the following statement opens a file named `Data.txt` in the `C:\Data` folder.

```
ofstream outputFile("C:\\Data\\Data.txt");
```

Notice that we've used two backslashes in the path where you normally use only one. Remember, in a literal string, the compiler interprets the backslash character as the beginning of an escape sequence. In order to represent a backslash as a character in a literal string, you must use two backslashes.

Here's another example:

```
ifstream inputFile("Z:\\names.txt");
```

This statement attempts to open the `names.txt` file in the root folder of drive Z.