

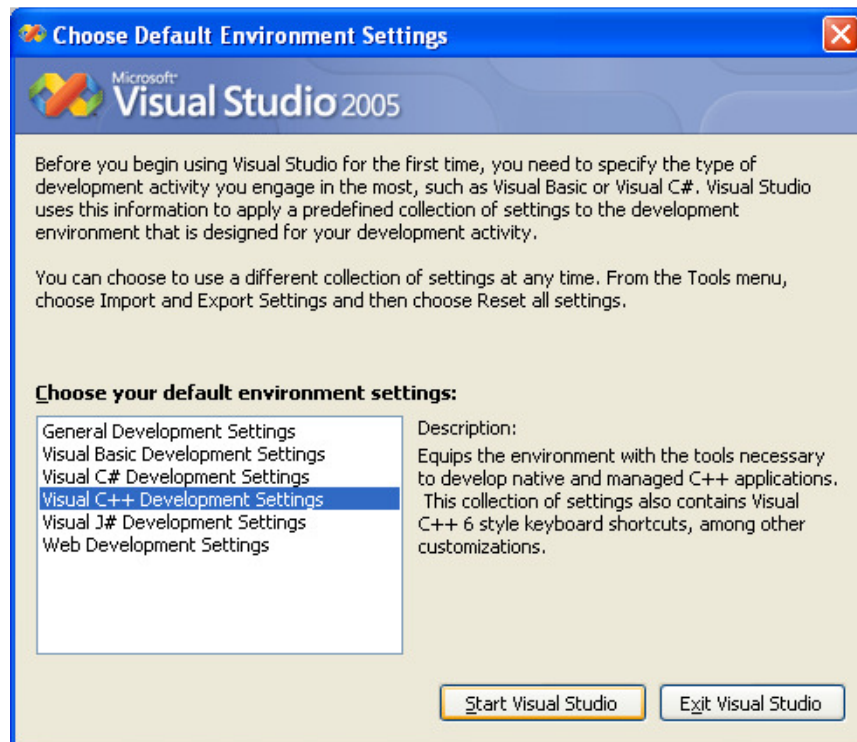
OBJECTIVE

The goal of this lab is to become familiar with the Microsoft Visual Studio .NET 2005 IDE (Integrated Development Environment), to compile and run a C++ program, and to observe several types of common compile-time errors. The IDE provides a broad set of development tools for completing, testing, and refining your program.

STEP 1 – Create a Project

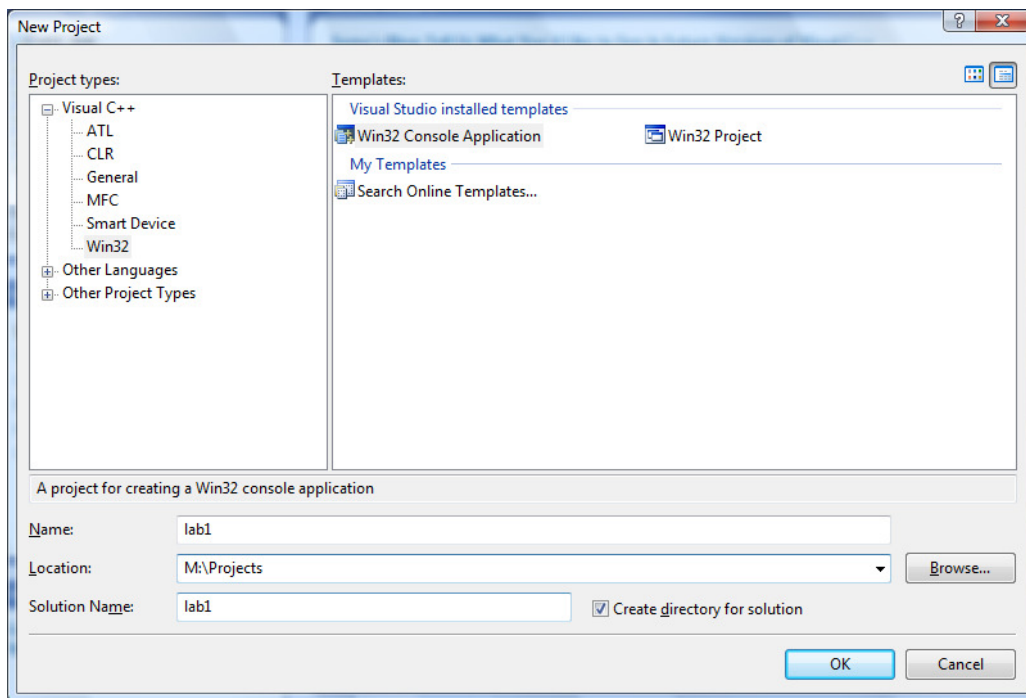
In this step you will create a project that contains a single C++ source file.

1. After you have logged into Windows XP, start up Microsoft Visual Studio .NET 2005. It's located under Start → All Programs → Microsoft Visual Studio 2005. Once it is loaded, you may want to maximize it, forcing it to cover the entire screen.
2. If this is the first time you have used Visual Studio you will see the dialog box below asking you to set the default environment settings. Select "Visual C++ Development Settings" and click the Start Visual Studio button.

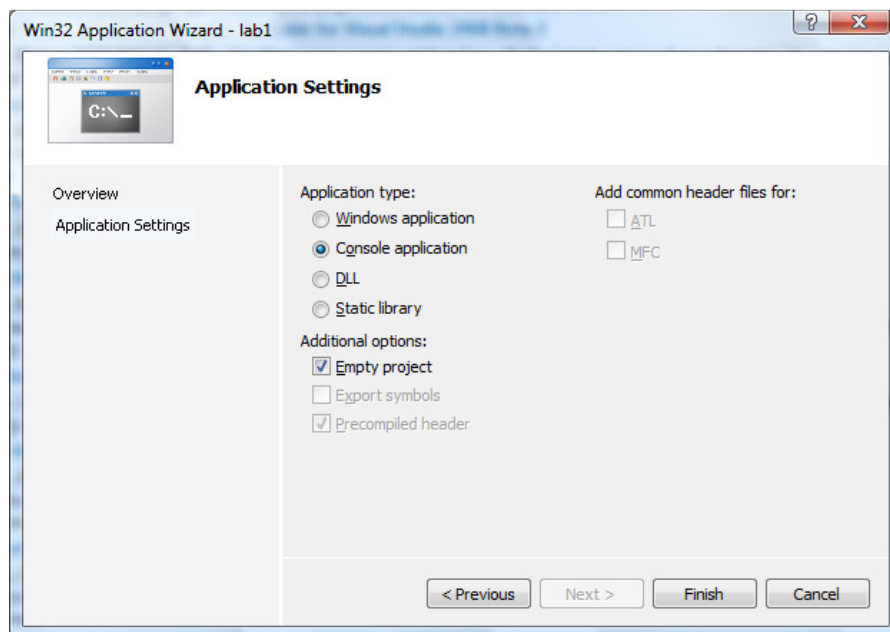


3. The first thing you will see is the "Start Page". This page allows you to set some profile information, open a new project or select a previously open one, and browse on-line information about .NET. We will not need to use this page.
4. Before you can write a program, you must create a new project. Create a new project by selecting from the menu File → New → Project... The "New Project" dialog box will be displayed.
 - A. Under "Project Types:" select **Visual C++** and then **Win32**.
 - B. Under "Visual Studio Installed Templates:" select the **Win32 Console Application**.

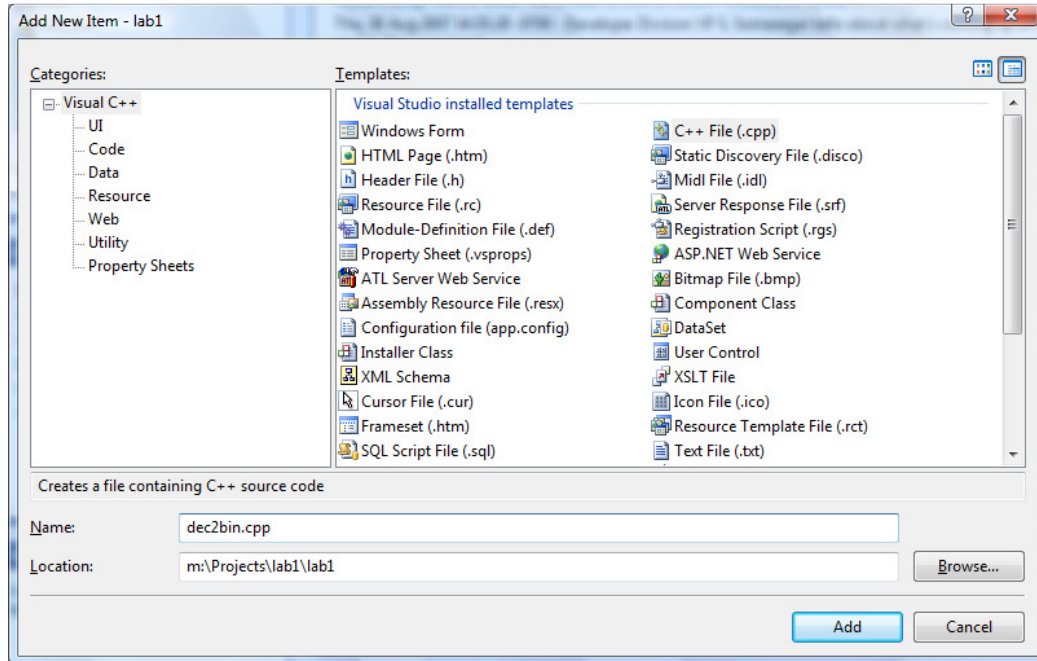
- C. Type “lab1” for the Name of your project. Then change the Location to M:\Projects. This will save your work to the networked M drive which is accessible from any Harding computer.



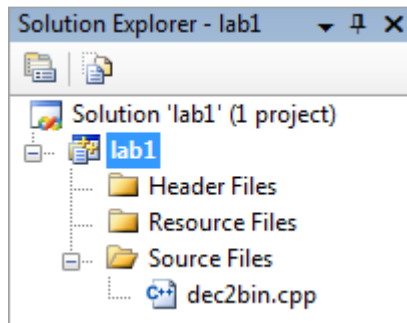
- D. Press the “OK” button. This will close the dialog box and display an application wizard. Press Next and then check the “Empty project” checkbox. Then click the Finish button. A “solution” named lab1 will be created along with the project lab1. A solution can contain one or more projects. In this case, the lab1 solution contains the lab1 project. Usually the project and solution names will be the same.



5. You will now create a C++ program file that will be added to your project.
- From the menu select Project → Add New Item... The “Add New Item” dialog box will pop up with a variety of file types that can be created.
 - Select “Code” on the left and the “C++ File (.cpp)” option on the right.
 - Give the file a name by entering “dec2bin.cpp” in the Name field.
 - Click the Add button. This will create the file, displaying an empty window entitled “dec2bin.cpp”.



- Now if you click the “+” beside the Source Files folder in the Solution Explorer, you will see the dec2bin.cpp file has been added to your project. It should look something like this:



STEP 2 - Compile the Program

Now that you have an open project with a .cpp file in it, you are ready to enter and compile a C++ program that converts decimal numbers to binary as in the attached flow chart.

- At this point you should now have a Lab1 project open containing the file dec2bin.cpp. Attached to this lab is a printed copy of the dec2bin.cpp program which you should **now type into the dec2bin.cpp window**. Type the program *exactly* as shown (use your own name though), comments and all. Note: Do not type in the numbers at the beginning of each line; they are for reference only.
- Once you have entered the entire program, notice the asterisk (*) next to the dec2bin.cpp in the title bar (blue area) of the program window. The asterisk indicates that the file has not yet been saved. Type Control-S to save

your file. Once the file is saved, the asterisk will disappear. Note that even if you do not save your program, Visual C++ will automatically save it for you when you compile your program (in the next step).

3. From the “Build” menu, select the “Build Solution” option. This will do two things: 1) compile the program and 2) build the executable if there were no compiling errors. At the bottom of the VS.NET IDE, you will see the compiler messages as it attempts to compile and build the application. If you typed in the program correctly, you will see the following message shortly:

```
1>----- Build started: Project: lab1, Configuration: Debug Win32 -----
1>Compiling...
1>dec2bin.cpp
1>Linking...
1>Embedding manifest...
1>Build log was saved at "file://m:\Projects\lab1\lab1\Debug\BuildLog.htm"
1>lab1 - 0 error(s), 0 warning(s)
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

4. If the dialog box says there are one or more **errors**, then you have mistyped something. Scrolling back up the compiler message window a little will reveal the errors encountered when compiling the program. Double-clicking on the error message will move the cursor in the dec2bin.cpp window close to where the error occurred. Syntax errors almost always occur at or above the line indicated by the compiler. Sometimes one error will kick-off one or more additional errors. It's always best to fix the first error and recompile to see if the rest go away. If you have difficulty correcting the error(s), seek assistance from the instructor so that you may continue with the lab.

STEP 3 - Run the Program

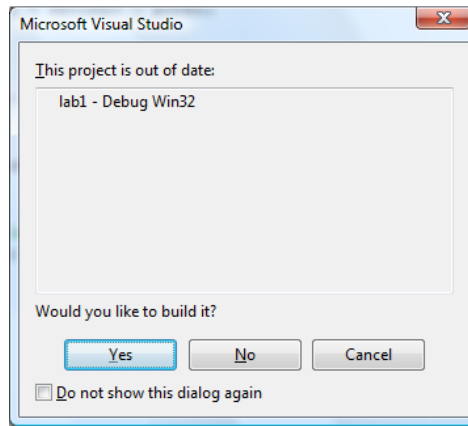
In this step you will run your program and make some observations about it.

1. Once you have a program that compiles without errors, run your program by pressing Control-F5. Your program will then run in a black window.
2. When the program asks you a yes/no question, you must respond with a lowercase “y” character for the program to treat your answer as a “yes.” Everything else, including “Y”, is treated as a “no” response.
3. Use the program to convert the decimal value 12345 to its binary representation. Write the binary number in the space below.

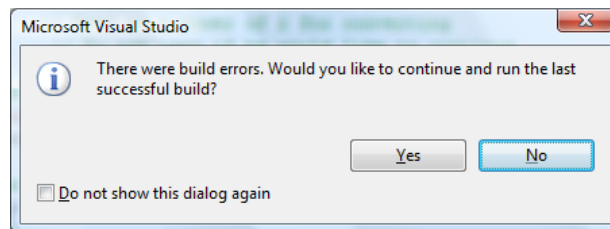


4. Enter some smaller numbers to verify that your program is working.
5. When you are finished converting numbers, answer ‘n’ to the question that asks if you would like to convert another number. The window will say, “Press any key to continue.” Press a key to close the window.

Note: Occasionally when you make changes to your program and attempt to run it before building it, the following dialog box will appear:



Just press the Yes button, and the project will be re-compiled before running. If there were errors in your program, you will see the following dialog box:



Just click No and try to fix the error(s).

STEP 4 - Syntax Errors

In this step you will introduce some errors into the program in a controlled manner to observe the messages reported by the compiler. This will assist you in “debugging” your programs when you encounter these error messages in the future.

1. Remove the '#' character from the beginning of line 10. The line number is displayed bottom of the Visual Studio window (e.g. “Ln 10 Col 1”). That is, change line 10 from

```
#include <iostream> to include <iostream>
```

Now rebuild the program. This “typo” will generate quite a few errors? Record the first error message you see in the dialog box at the bottom of the VS IDE. Then restore line 10 to its original value before proceeding.



2. Now comment out line 10 and recompile. That is, change line 10 so that it now appears

```
//#include <iostream>
```

Notice that this generates a ton of errors. Look for a line that begins “1>m:\projects\lab1\lab1\dec2bin.cpp(21)”. This error corresponds to line 21. Record the error message for line 21 below.



Since iostream is the library which defines cout, cin, and endl, commenting-out the header file hides the definition of these from the compiler. Be sure to restore line 10 to its original state before proceeding.

3. Delete the semicolon character on line 16 and observe the results after recompiling. What line number does the compiler say the error is on?



Sometimes the compiler errors are on the line(s) previous to the actual line the compiler is complaining about. Put the semicolon back before proceeding.

4. Change the “<<” operator at line 21 to “>>” and observe the errors generated after recompiling. Put line 21 back and then try reversing the operator at line 22 and note the error generated. In both cases, almost the exact same error messages are generated. Unfortunately they are very cryptic and don't give any hint as to what exactly is wrong. Just remember to check the direction of your arrows if you see these types of errors on input or output statements.
5. Change the variable “answer” at line 22 to “nswer” and recompile. Record the generated error messages, then change it back.



6. Delete the ‘}’ character at line 62 and recompile. Record the error message generated below.



Missing a closing curly-brace is a very common although difficult to catch error if you are not careful.

STEP 5 – Finishing Up

When you are through programming, close your solution by selecting “Close Solution” from the “File” menu. If you are asked to save changes to your project, click “Yes”. Once all the windows have closed, you may exit the application.


If you want to run your project again later, follow these steps:

1. Select File → Open → Project/Solution...
2. Select the Lab1 workspace file from M:\Projects in the Open dialog box and click the “Open” button.

STEP 6 – Run Another Program

Now that you have written your first C++ program, type in and run the program from chapter 2, pg. 37 (Listing 2.3 - ComputeArea2.cpp) of your text book. This program determines the area of a sphere. You will need to repeat the steps of this lab for typing in and running this program. These steps are summarized below:

1. Start Visual Studio .NET 2005.
2. Create a new project (Visual C++ Projects, Win32 Console Application) on your M drive with a unique name. Check the “Empty project” checkbox from the application wizard.
3. Add a C++ file to the project.
4. Type in the program.
5. Compile (or build) the program and recompile until it is free of syntax errors.
6. Run the program.
7. Once you are through, close the solution and close Visual Studio .NET.

Turn in this lab, making sure you have completed all the steps and have written answers everywhere you see the  symbol. Also turn in a printout of your ComputeArea2.cpp program.