Exercise II: Experiencing Syntax Errors in Visual C++

As with any compiler, if the programmer instructs Visual C++ to compile a source file which includes syntax errors, the file will not be compiled to an object file, but rather a (sometimes long) list of those syntax errors will be presented.

Launch Visual C++ on your computer. As you did for Exercise I, go to the **FILE** menu and select **New**. On the Projects tab of the **New** dialog box, ***single click*** on **Win32 Console Application**, specify the location for this project and then give the project the name **exercise2**. Click on the **OK** button.

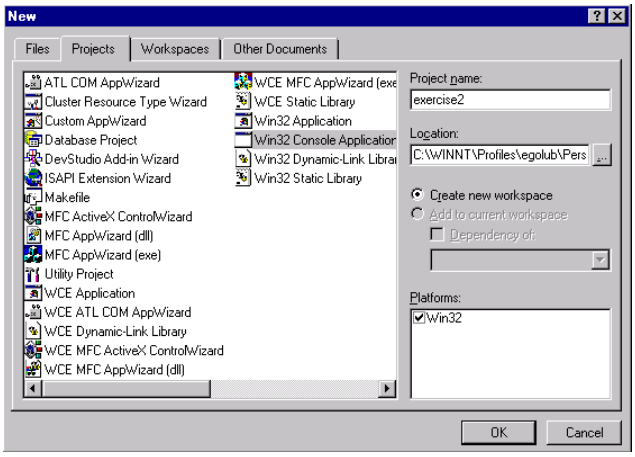

Figure II.1

For this project, when presented with the list of applications to create, select **An empty project**. This will create a project workspace, but will not create any files within the workspace.

For this exercise, we will create our own source code file named exercise2.cpp that will contain a main() function. To accomplish this, go to the **PROJECT** menu and move your mouse to the **Add To Project** sub-menu as shown in Figure II.2. From the **Add To Project** sub-menu, select **New…**. The **New** dialog box will be displayed, and the **Files** tab will be active (see Figure II.3).

Single click on **C++ Source File** from the presented list. In the text entry box labeled **File name:** enter the name exercise2.cpp and click on the **OK** button. A file named exercise2.cpp has now been created in your project's directory, added to the project and opened in the editor panel in Visual C++.

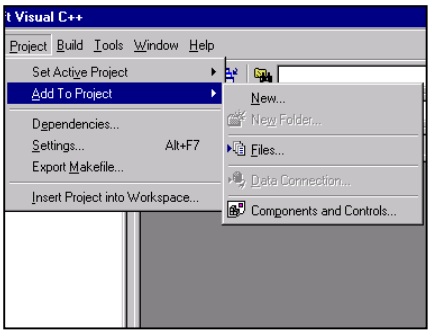

Figure II.2

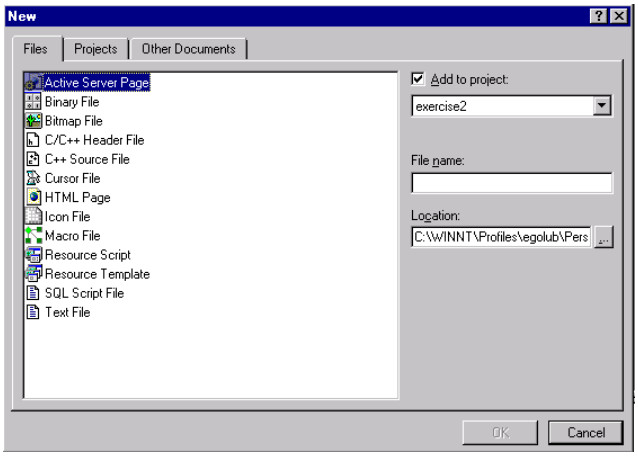

Figure II.3

If you go to the FileView for this project, and expand the tree, you will see that exercise2.cpp is listed under Source Files.

Our last step before beginning our programming is to tell Visual C++ that we want to use only the standard components of C++ for this program. Go to the **PROJECTS** menu and select **Settings**. Single click on the name of the project at the top of the tree that is presented. Click on the **C/C++** tab. From the pop-up menu labeled **Category**, select **Customize**. You should now have a dialog box that appears similar to Figure II.4.

Single click in the checkbox next to **Disable language extensions** and then ***single click*** on the **OK** button. This instructs Visual C++ not to use the Microsoft extensions to C++. For the most part, this option will not affect us. However, one noticeable difference is that if you do not disable the language extensions, then when you declare a loop control variable inside a for loop, it does not go out of scope when the loop ends. In the program we are about to create, this would cause the compiler to give an error saying that the variable ***i*** was multiply defined.

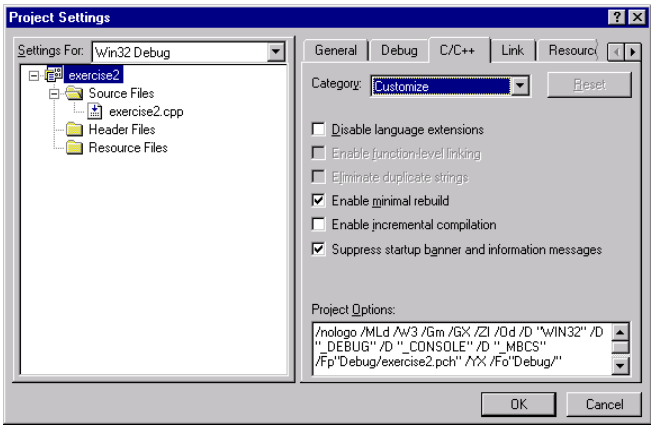

Figure II.4

Now that the environment is prepared, we can continue creating our project. Go to the FileView and ***double click*** on exercise2.cpp to confirm that it is in the editor. In the editor, very carefully enter <u>exactly</u> the following code:

```
#include <iostream.h>

int
main() {
int num;

        cout << "How many times should we do this?";
        cin >> num;

        for (int i=0; i<num; i++)
                cout << "Hello " << i << endl;

        for (int i=0; i<num; i++)
                cout << "Goodbye  " << i << endl;

        return 0;
}
```

Figure II.5

Use any of the techniques mentioned in Exercise I to compile your project. Remember that Visual C++ will automatically save any files that are part of the project that have been modified. If you typed in everything correctly, it should have compiled.

This program expects a single positive integer as input. This should be input from the keyboard. We will learn how to redirect input from files in Exercise IV. Run the program using one of the techniques shown in Exercise I.

Let us now introduce a syntax error to our program. Delete the semicolon at the end of the line **cin >> num;** in the program. Now, compile the project once again. This time,. It should not compile, but rather indicate that there is a syntax error in your code. This error message will be displayed in the panel across the bottom of your Visual C++ window similar to Figure II.6.

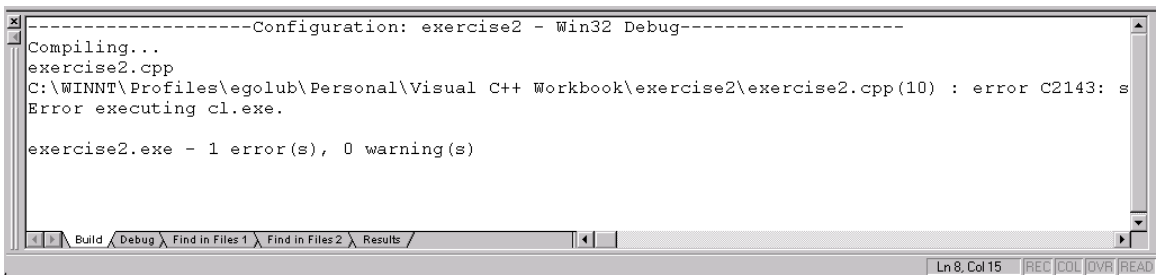

Figure II.6

If you find the panel is too narrow, you can extend it by using the mouse to click and drag the top boundary higher on the screen.

Within this panel, each syntax error will be displayed along with the name of the file in which the error occurred and the line within that file on which the error was detected. As with other compilers, this is a starting point but the actual error may appear earlier in your code. A nice feature which Visual C++ provides is that if you double click on an error, Visual C++ will open the file in which the error was detected, bring it to the foreground and position the cursor on the line within that file where the error was detected.

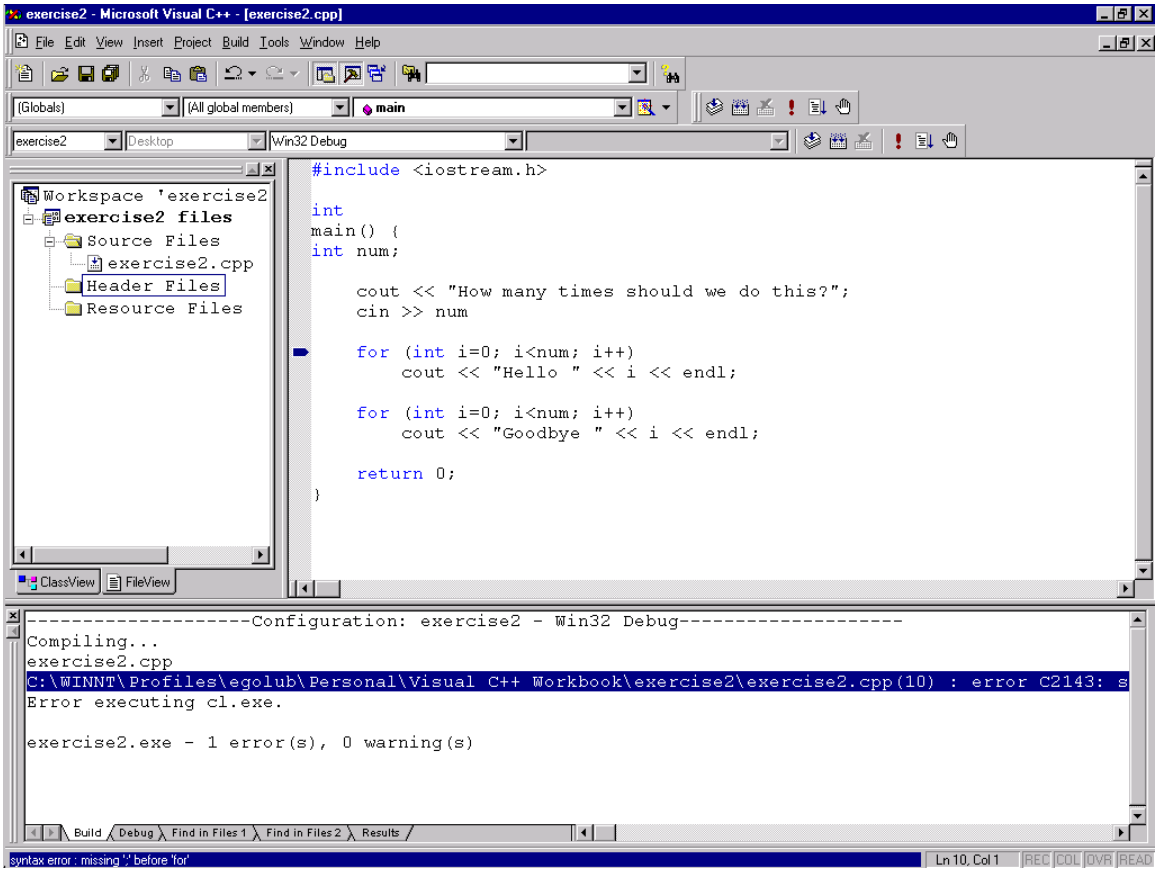

Figure II.7

In the case of this error, the cursor was positioned at the beginning of the line after the missing semicolon. This *is* where the error was actually detected.

Now go correct the error and compile the project again. This time, it should compile successfully and create an executable once again.

Our next change will be to replace one of the **for** loops with a **while** loop. Go into your program and replace the lines shown in Figure II.8 with those shown in Figure II.9.

```
for (int i=0; i<num; i++)
        cout << "Hello " << i << endl;
```

Figure II.8

```
i=0;
while (i<num){
        cout << "Hello " << i << endl;
        i++;
}
```

Figure II.8

Now compile the project once again. Again, the project will not compile, but rather will indicate the syntax errors that have been detected. In this case, the variable *i* was not defined. The solution is to declare *i* as an **int**. Do so now and compile the project.

The final error we will introduce in this exercise is to "forget" to include iostream.h in the file. Go to the top of your exercise2.cpp file and delete the line that includes iostream.h. Now compile the project once again. This time, the compiler indicates that there are many errors in your code (there should be 7 errors and 1 warning). All of these were caused by the omission of that header file. One note about warnings: while a project can compile and have an executable created even if warnings are generated, it is widely felt that all warnings should also be eliminated in a program.

When confronted with a long list of error messages, it is often a good idea to look at the first error in the list, correct it, and then compile the project again. Many times, the compiler will become "confused" by the first error, and falsely report other parts of your code as being incorrect. This is a characteristic of all compilers that attempt to go past the first error that they encounter.

Congratulations! You have now compiled and executed your second exercise.

To leave the Visual C++ environment, go to the **FILE** menu and select **Exit**.