Exercise IV: Input and Output File Redirection

The output of a program will often be more that can fit in a single window without scrolling off screen. Additionally, our input files are often quite large. It is often useful to be able to redirect input from a text file and output to a text file. In this exercise, we will do both of these.

Launch Visual C++ on your computer. Create a new, empty, **Win32 Console Application** named **exercise4**. Go to the project settings and disable the language extensions as shown in Exercise II. Go to your Windows environment and copy the files class1.h and class1.cpp from the exercise3 directory into the exercise4 directory. Return to the Visual C++ environment and add these files to exericse4. Create a new C++ source code file named exercise4.cpp and enter the following code:

```
#include <iostream.h>
#include "class1.h"

int
main() {
int num;

     cout << "Enter a number: ";
     cin >> num;
     for (int i=0; i<num; i++)
          for (int j=0; j<num; j++)
               cout << class1(i,j) << endl;

     return 0;

}
```

Figure IV.1

Compile the project and create an executable file.

Run the program and enter the integer **4** when prompted by the program. Your console should appear similar to the following:

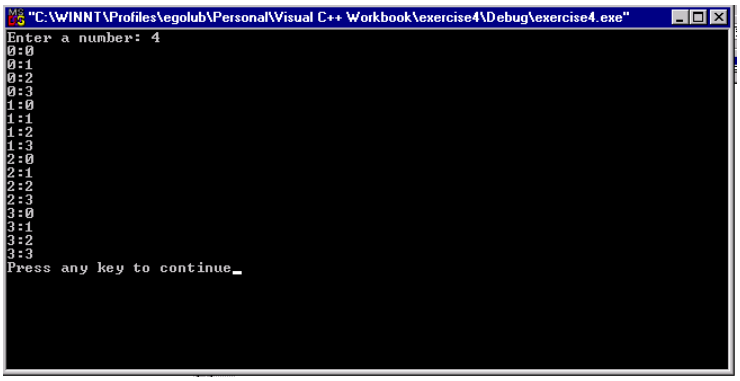

Figure IV.2

Press some key to continue.

Now, run the program again, but this time, enter **16** as the number when prompted. Your console should now appear similar to the following:

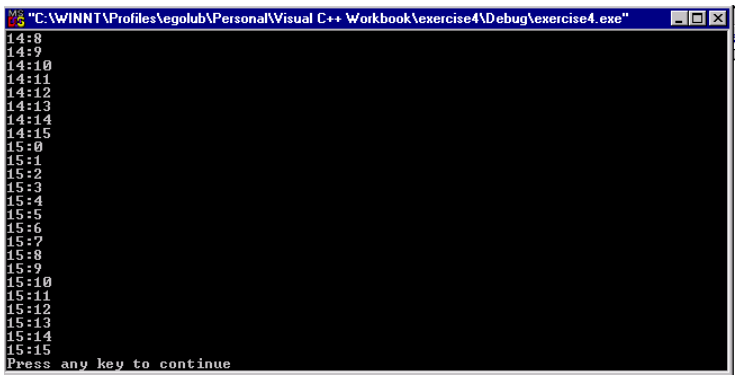

Figure IV.3

The output of the program scrolled off the screen, so we were unable to view it. This is one time when redirecting the output to a file can be useful. Another place where output file redirection is useful is when you are given an output that your program needs to match. If you redirect the output to a file, you can then use a utility such as Windiff (see Appendix B) to compare your output to the given one.

One question to address is "Where should we put the input and output files?" Ideally, the place to put them would be the project directory. However, in order for file redirection to work consistently while working on your computer (possibly in mutliple applications at the same time) you need to specify the full path to these files. For this reason, I recommend creating a directory such as **C:\IOfiles** to use for this purpose. In this exercise, I will assume that you have created such a directory, and that you will place all input files there as well as look there for your output files.

To specify input and/or output redirection, go to the **PROJECT** menu and select **Settings…**. You should see a dialog box similar to the one in Figure IV.4. Since it is rare that one would use redirection in a release of a product, but rather would use this when testing one, you will need to go to the **Debug** tab (as shown in Figure IV.5) to specify redirection.

You specify the redirection instructions in the **Program arguments:** text entry box in the same way you would if running a program from the command line; using < and > with file names. For our current program, we would like to redirect the output to a file called **e4.out** in **C:\IOfiles**. To accomplish this, type **> C:\IOfiles\e4.out** in the text entry box as shown in Figure IV.6.

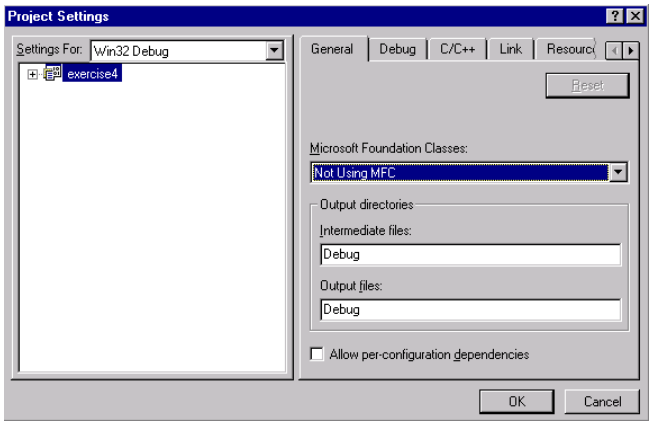

Figure IV.4

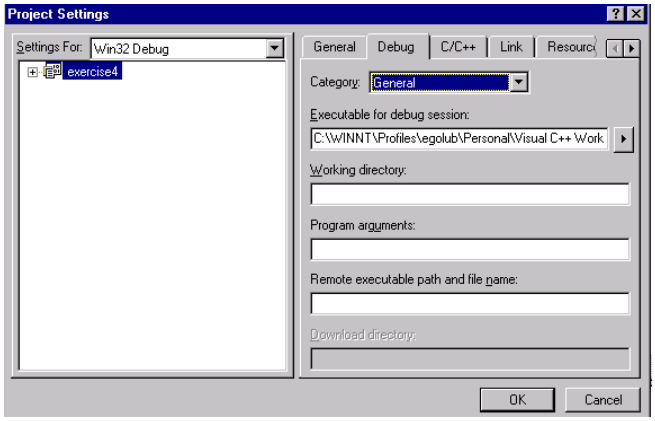

Figure IV.5

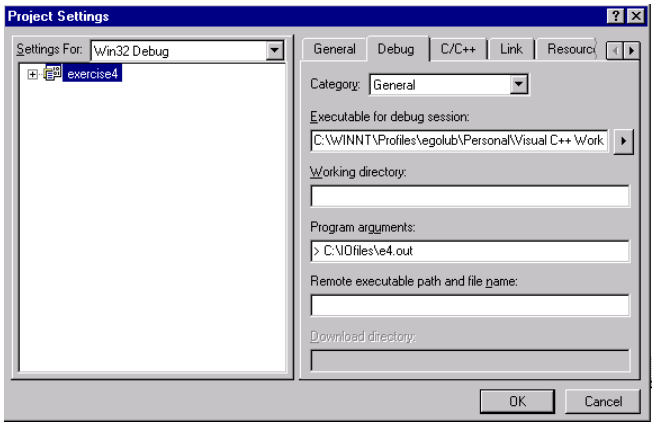

Figure IV.6

Now, when run your program again, all messages will be redirected to the output file. This means that you will not see the prompt to enter an integer nor will you see the prompt to press any key to continue. Run the program now. Don't forget that you do need to enter an integer (use **16**) and then after waiting until you think the program is done press a key to continue.

To view the output, you need to view the file **C:\IOfiles\e4.out**. This can be done through Visual C++ easily by adding that file to the project. Go through the steps to add a file to the project, but this time when you are supposed to select the file to add, go to the **File name:** text entry box, and explicitly enter the name of the file, then click the **OK** button.

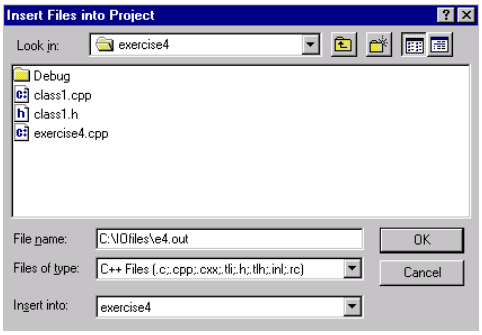

Figure IV.7

If you now look at the FileView for the project, e4.out has been added to the bottom of the list. Since the suffix is not known as a Visual C++ component, it is not placed into any of the folders such as Source Code or Header Files. If you double click on e4.out, the file will be opened in the editor. You may now scroll through the file to view any part of the program's output.

Up until now, all the files that were part of the project had been exclusively updated by us in the editor while working on the project. However, if you view e4.out, and then run the program again, the program (which is running as an independent process) will be modifying the file as well. We will want to assure that the file in the editor is updated each time the file is modified by the program. Visual C++ will see that the file has been modified based on file dates. We can specify whether Visual C++ automatically refreshes the file or asks us whether we want it to refresh the file by going to the **TOOLS** menu and selecting **Options**. On the **Editor** tab, there is a checkbox labeled Automatic reload of externally modified files.

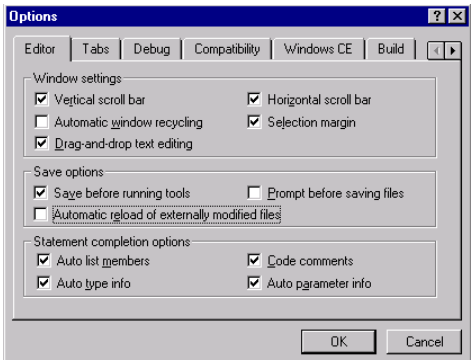

Figure IV.8

If you check that box, the modified files will automatically reload. If you leave the box unchecked, then if a file is modified, Visual C++ will ask if you want to refresh it with a dialog similar to the following:

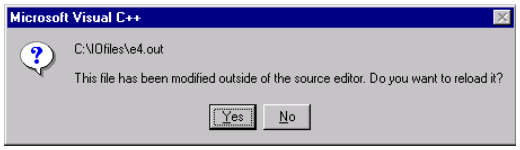

Figure IV.9

In many ways, it is more convenient to have the files automatically refreshed, so for the exercises in this workbook, we ask that you check that box now.

Now that you have checked that box, run the program again, using an input value other than **16** so that you can observe that the e4.out file is being refreshed.

In addition to redirecting the output of a program to a file, it is often desirable to redirect the standard input to a program from a text file. This can be done using the same basic technique as output redirection. Go to the same text entry box in which you typed the > C:\IOfiles\e4.out directive. If you type in < C:\IOfiles\e4.in before that, then the contents of C:\IOfiles\e4.in will be used for the standard input to the program. Do that now.

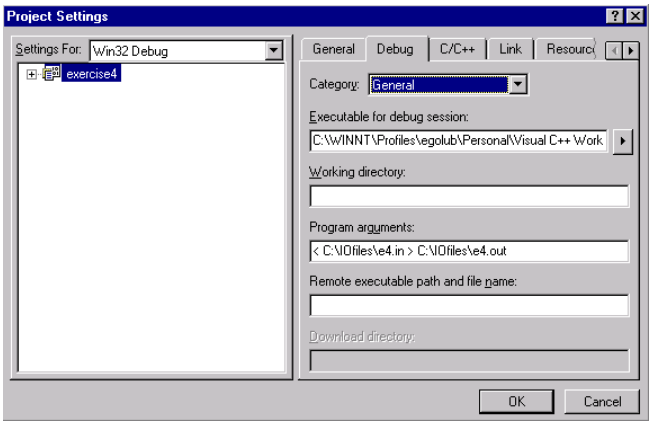

Figure IV.10

As with e4.out, we will want to be able to edit e4.in within the IDE. However, since the file has not been created yet, we can't use the technique of adding an existing file to the project. Instead, to begin the process you use the option of adding a new file to the project, but rather than selecting C++ Source File or C/C++ Header File, single click on **Text File**. Next, go to the **Location:** text entry box, delete what is already there and enter **C:\IOFiles**. Finally, enter the name **e4.in** in the **File name:** text entry box.

At this point, the file e4.in has been created in the C:\IOFiles directory, has been added to the project and has been opened in the editor. Enter an integer (such as **4** or **16**) to be used as input to your program in e4.in. When you run the program, the contents of this

file will be used as the standard input for the program. However, it is important to note that after the program has terminated, it will still expect the user to press a key to continue. This will not be taken from the redirected input file. The result is that when you run the program, the console window will appear totally blank and sit there waiting unless you press a key (such as the space bar).

Now, run the program a few times with different inputs.

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

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