Exercise IX: Templates

In this exercise we will see one way in which a template class can be used within Visual C++. The technique that will be shown is required if you are using a separate header and implementation file for your template class. If you fully implement the template class within the header file, then the steps taken in this exercise are not required.

There is a zip file named **e9.zip** at the anonymous FTP server **ftp.cs.umd.edu** in the **/pub/egolub/VC.workbook** directory. Downloaded this file and extract the files which it contains. Unzip those files to a temporary directory on your machine.

Launch Visual C++ on your computer. Create a new, empty, **Win32 Console Application** named **exercise9**. Go to the project settings and disable the language extensions as shown in Exercise II. Go to your Windows environment and copy the files that you extracted from e9.zip into the exercise9 directory. Return to the Visual C++ environment and add the following files to the project: **main.cpp**, **Rational.h**, **Rational.cpp**, **Set.h**, **SetTypes.cpp**. Notice that you ***do not*** add **Set.cpp** to the project. It is very important that you ***do not*** add that file to this project. At this point, if you go to FileView and fully expand that tree, your Visual C++ window should appear similar to Figure IX.1.

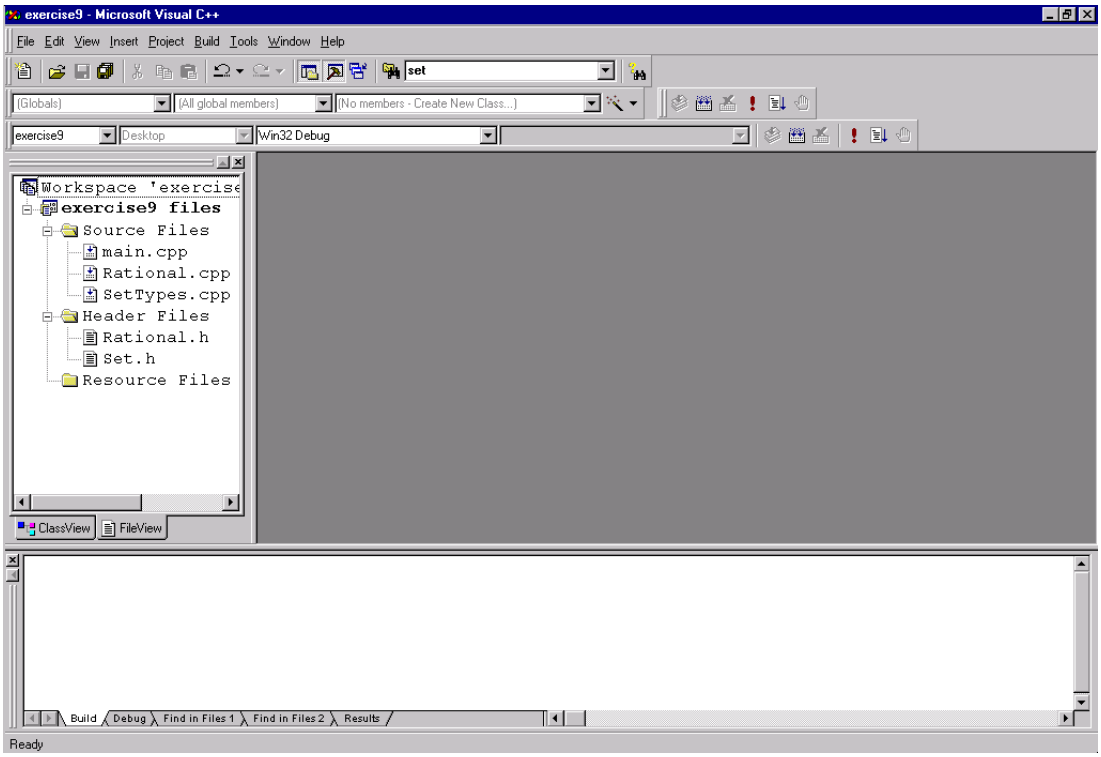

Figure IX.1

Double click on **SetTypes.cpp** in the FileView panel to view that file in the editor. This file is a **wrapper file** that will enable Visual C++ to create all of the required versions of the **Set** template class even though it is compiled independent of the files that use it. If the **Set** template class had been fully defined within the **Set.h** header file, then each module that used a **Set**, and therefore included **Set.h** would have included the entire template class, and Visual C++ would have been able to create the required version at that time. However, since we have implemented the **Set** template class in **Set.cpp**, we need some way of instructing the compiler to generate the appropriate versions of the class. The use of the **SetTypes.cpp** wrapper is one such technique.

Now, compile the project. Notice that a new folder has been added to the FileView tree named **External Dependencies**. If you expand this folder, you will see that **Set.cpp** has been added to the project by Visual C++ when it detected that it was included by **SetTypes.cpp** but was not a member of the project yet. You are required to create the project in this manner so that when you make changes to **Set.cpp**, that file is recompiled appropriately.

If your own projects require template classes, you should either fully implement them within the header files or use the wrapper technique showed in this exercise.

Congratulations! You have now completed your templates exercise.

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