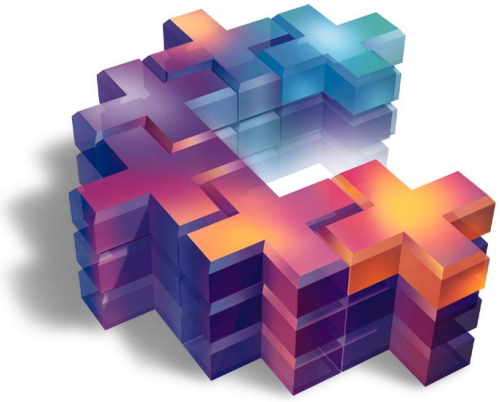


Microsoft® Visual C++® 6.0

Introduction

Key features and lab exercises to familiarize new users to the Visual environment

January 1999

CONTENTS

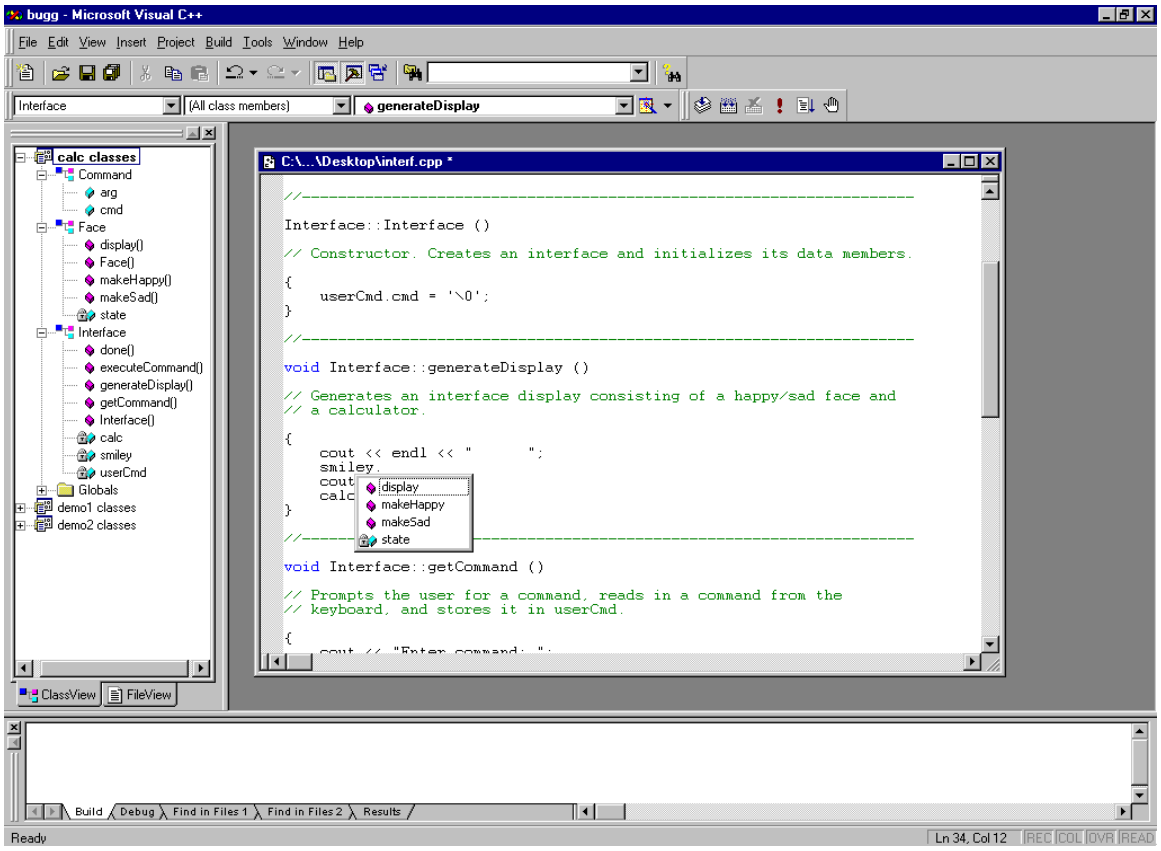
Key Features

Statement Completion Options

To turn on Auto List Members, Auto Type Info, and Auto Parameter Info simply go to the Tools | Options. On the page with the **Editor** tab, you will find a checkbox for these three items plus one for Code Comments. Check the boxes so that an X is visible within the box to turn these options on.

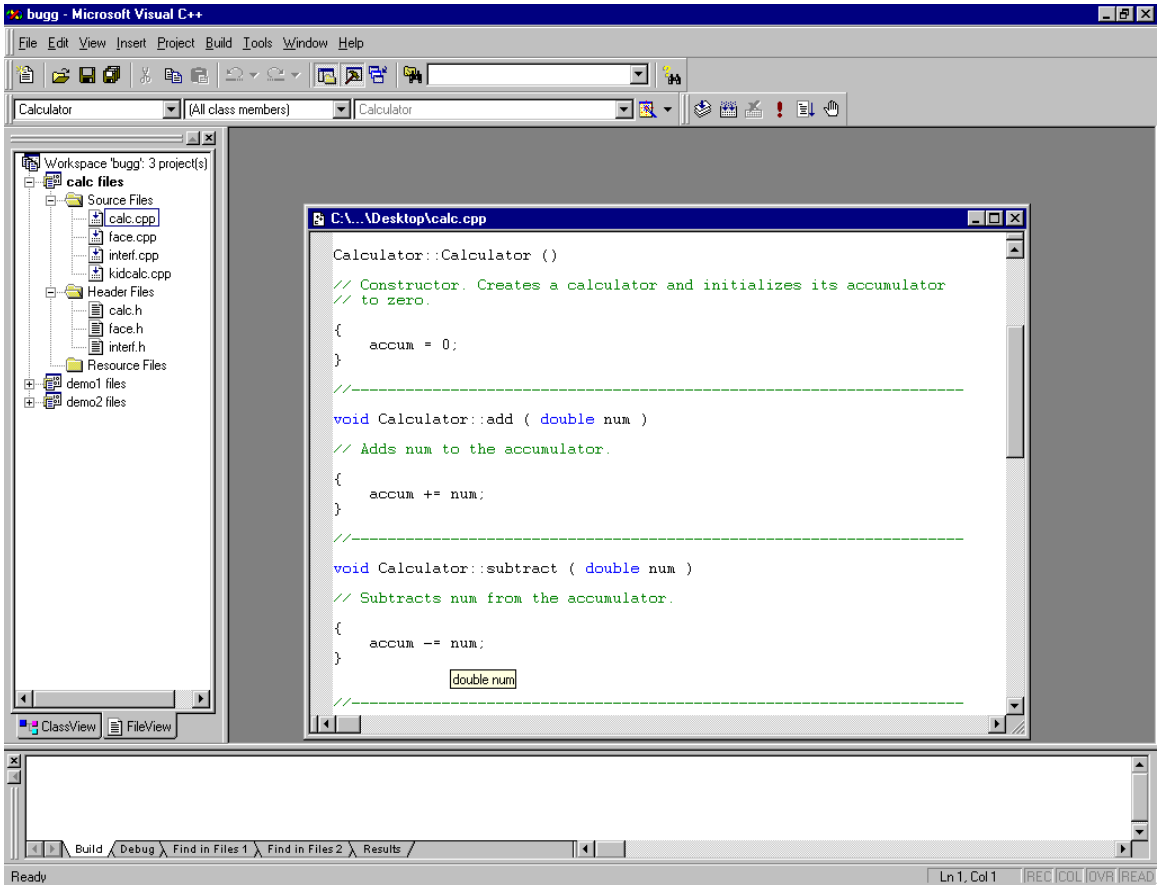
Auto List Members

When selected, the editor displays a list of valid member functions and variables when you type "." or "->" after a variable name. When cleared, you can still invoke the member list with a key combination. The default key combination is CTRL-ALT-T but you can assign different keys by choosing customize from the Tools menu, selecting the keyboard tab and then selecting the List Members entry under the Edit category for the Text editor.



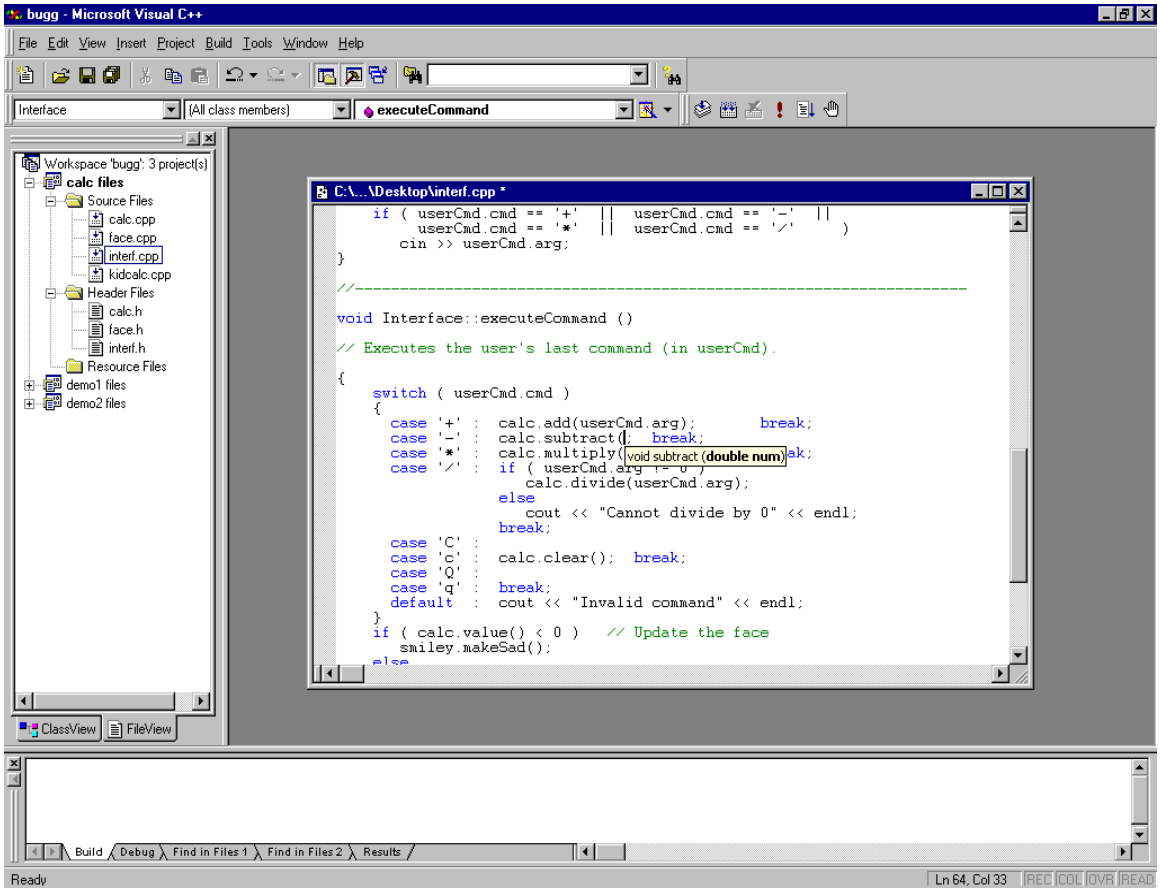
Auto Type Info

When selected, this feature enables you to view type information for an identifier by simply hovering the mouse over the identifier. When cleared, you can still invoke the type information with a key combination. The default key combination is CTRL+T but you can again assign different keys under the Type Info entry under the Edit category for the Text editor.



Auto Parameter Info

When selected, the editor displays the complete prototype for a function, including any required parameters, after you type the opening parenthesis "(". When cleared, you can still invoke the parameter information with a key combination. The default combination is CTRL+SHIFT+SPACE.



Searching Techniques

Find

The first one is **Find** within the current file. Typing CTRL-F will bring up a dialog that provides a space for you to specify the search text or the regular expression to match. Type the text or expression in the "Find What" box. Use the drop-down list to select previous search strings. Use the right-arrow button to the right of the dropdown list to display a list of regular search expressions. When you select an expression in this list, the expression is added as search text in the Find What box. If you use regular expressions, be sure the Regular Expression check box is selected. There is also an option to black a bookmark on all instances of the word or expression.

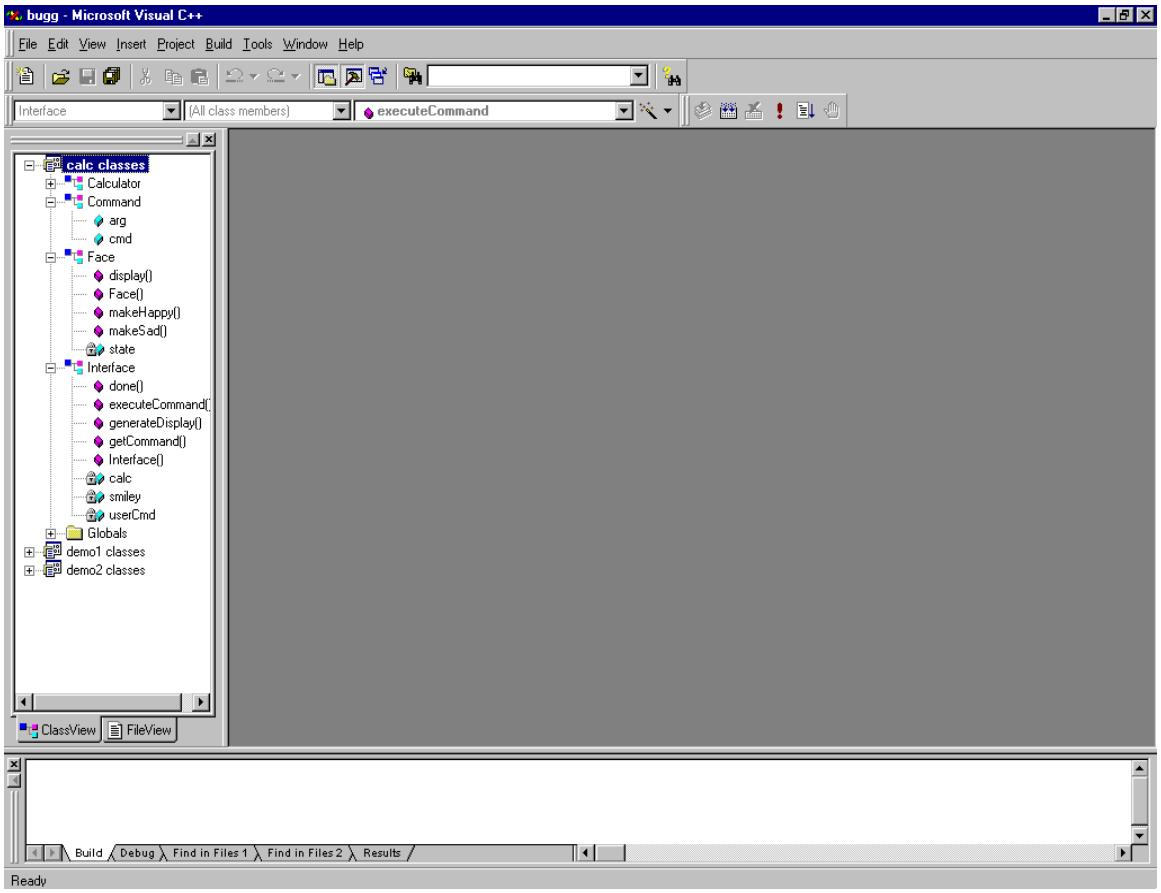
Find In Files

The second option is **Find In Files**. Use this command to display a list of all occurrences of a character string in the VC++ files that you specify. You can use the command on a single file or an entire project.

Navigation

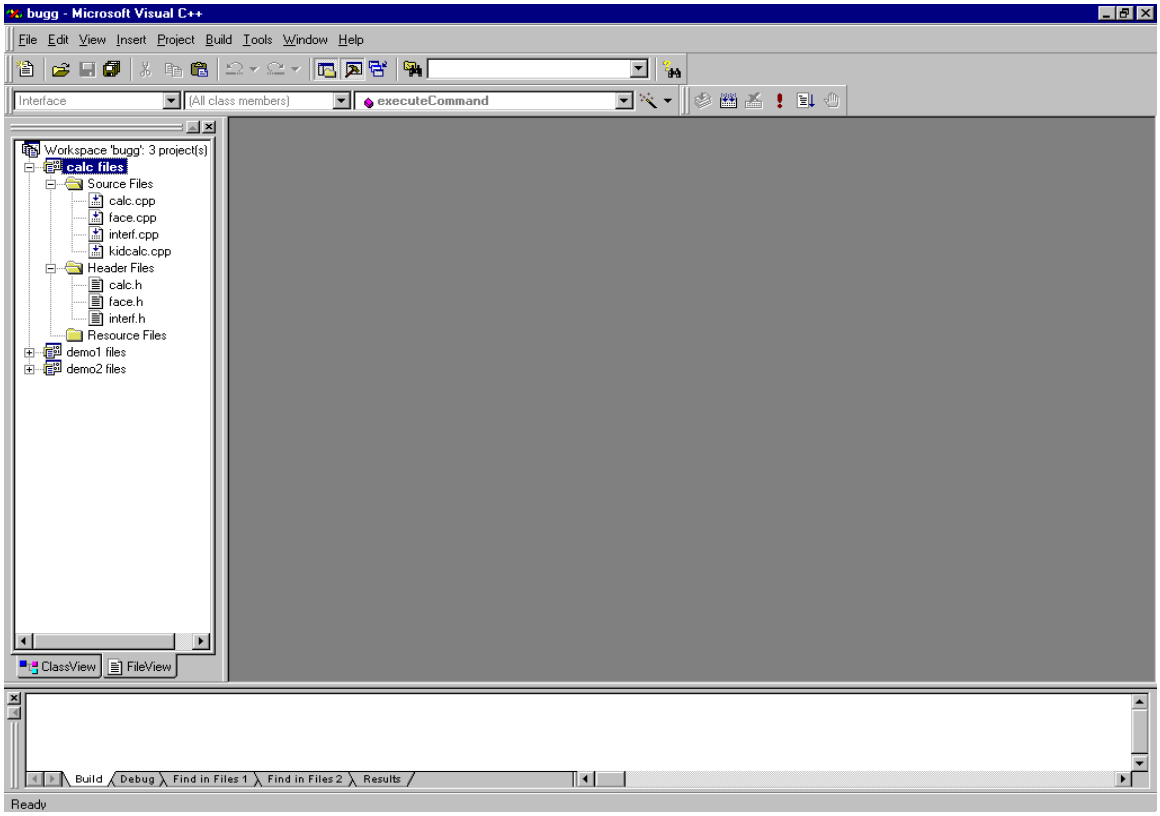
ClassView

Another great feature of VC++ 6.0 is the **ClassView**. The ClassView Tab of the Project Workspace window displays the classes and their members for all projects on the workspace. The folder name shown in bold type represents the default project configuration. Expanding a project folder displays the classes included in the project. Expanding a class displays the members in that class. The ClassView shortcut menu changes dynamically depending on whether the focus is on a project, class or class member. When focus is on a project, you can use the ClassView shortcut menu to add a new class. When focus is on a class, you can add members to the class.



FileView

The **FileView** pane of the Project Workspace shows the relationships among the projects and files included in the project workspace. The relationships in FileView are logical relationships, not physical relationships, and do not reflect the organization of files on your hard disk. The icons used also give you information about the files. FileView shows the relationships of the source files and the dependent files used to build all project configurations. The active project in the workspace is indicated in FileView with bold type. The active project is the project that will be built when you use the commands Build or Rebuild All. You can select a different active project by using the Set Active Project command on the Project menu. The Active configuration determines which set of build options is used when you build the active project. You can select a different active configuration by using the Set Active Configuration command on the Build menu.

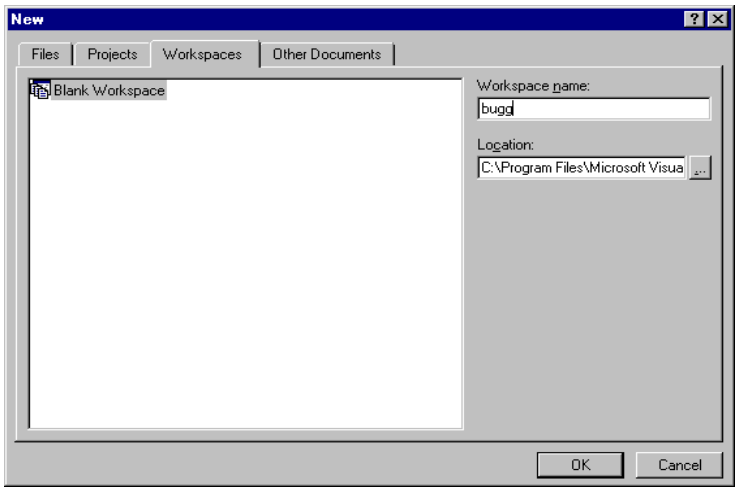


Lab Exercises

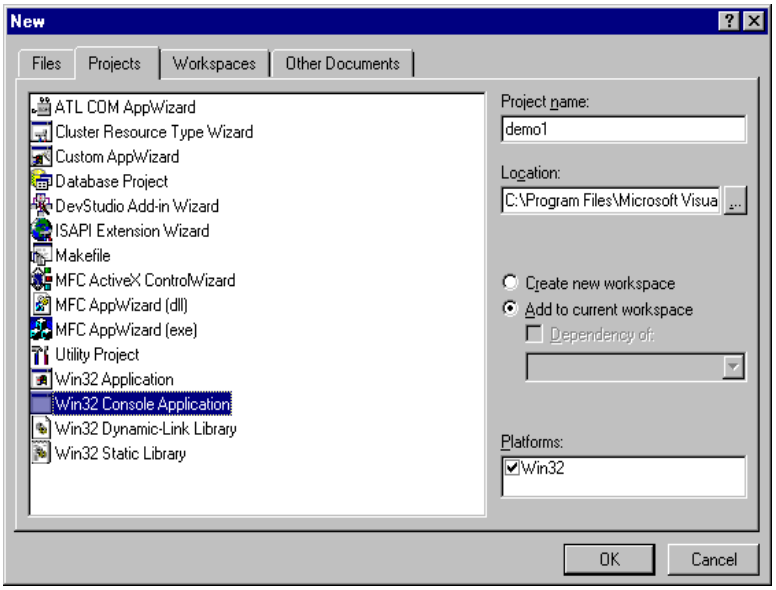
One Source File

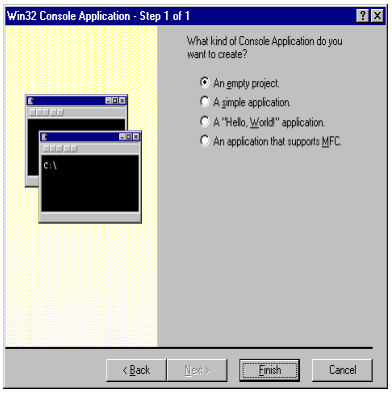
1. From the Windows desktop go to Start | Programs | Visual Studio | Visual C++

2. A new Workspace must be created in order to start our project. A Workspace is just the name of the folder or directory where you will keep your projects. To create a new Workspace go to File | New. Then click on the Workspace tab. Enter the name of your new Workspace in the Workspace Name dialog box. For our example we will call our Workspace, "bugg". Take note that the Location dialog box is the folder where the workspace is created.



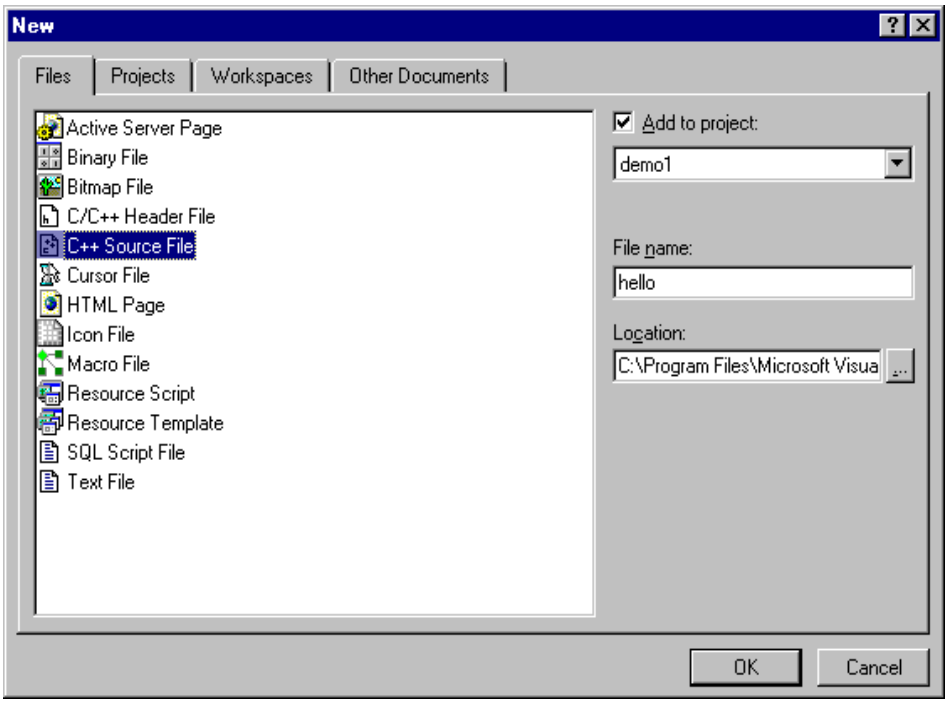
3. Now, you will need to create a new Project. A Project is simply the name given to the folder that will contain the files in your program. This includes source files, headers, etc... Again, go to File | New. This time click on the Project tab and check the box labeled Win32 Console Application. Enter the name of the Project in the Project Name dialog box, for our example we will use "demo1" as our name. Make sure that the Add to Current Workspace box is checked, this will add Project to the "bugg" Workspace. Click OK and you will be





prompted to select the type of project you want to create. We want to create our source file and then add it to the project so select Empty Project then click Finish. Visual Studio informs you that it has created the skeleton for your program. Take note of the path of the Workspace and Project at the bottom of the window.

4. Next, we must create the source code file for our program. Go to File | New. This time click on the File tab and check the C++ Source File box. Now we must choose a name for our source code file, we will use "hello" as our name. Make sure that the Add to Project box is checked and that the name of the Project is "demo1" and then click OK.

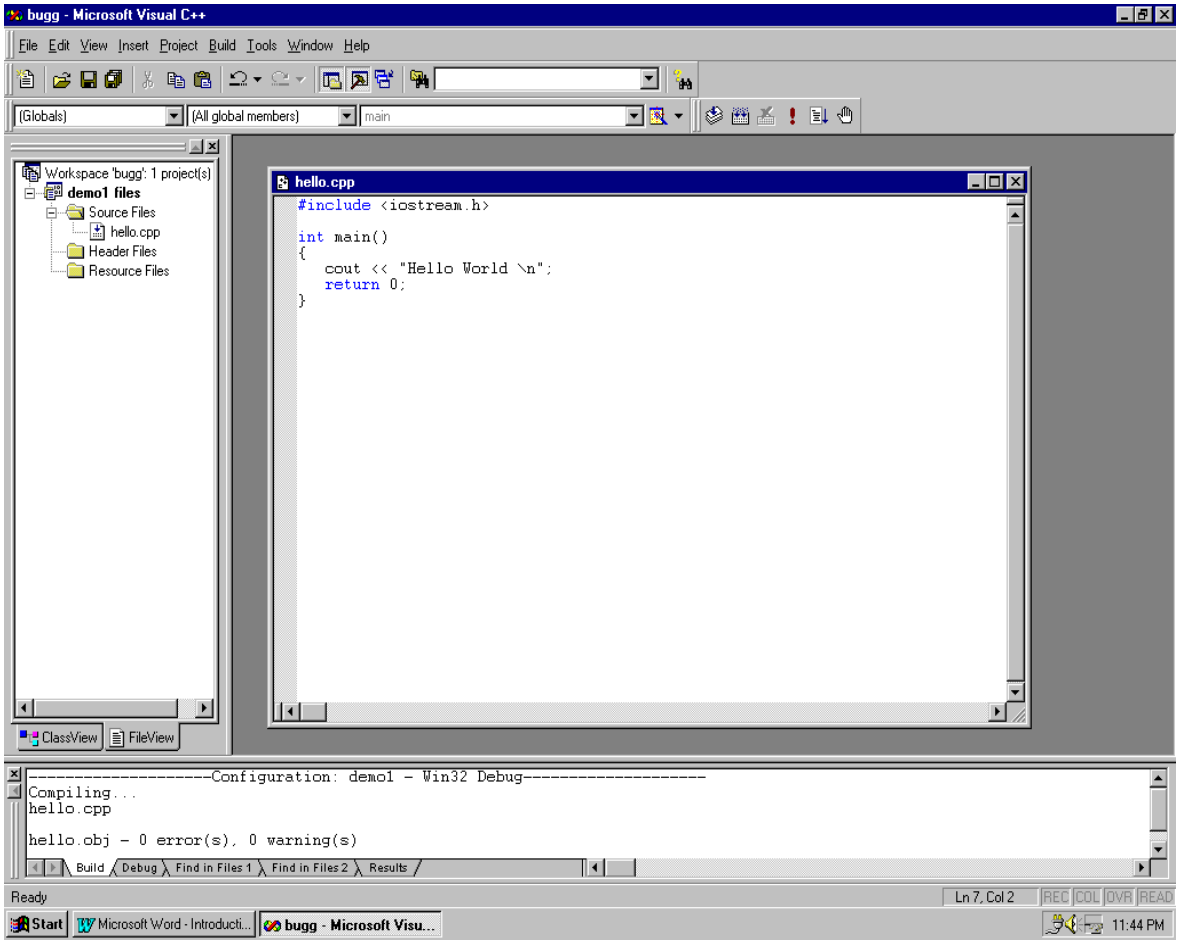


5. Once you have created the source file, a window will open allowing you to type your C++ code. Take note of the FileView window, “hello.cpp” will be listed in the source files folder. We will now type the following C++ code:

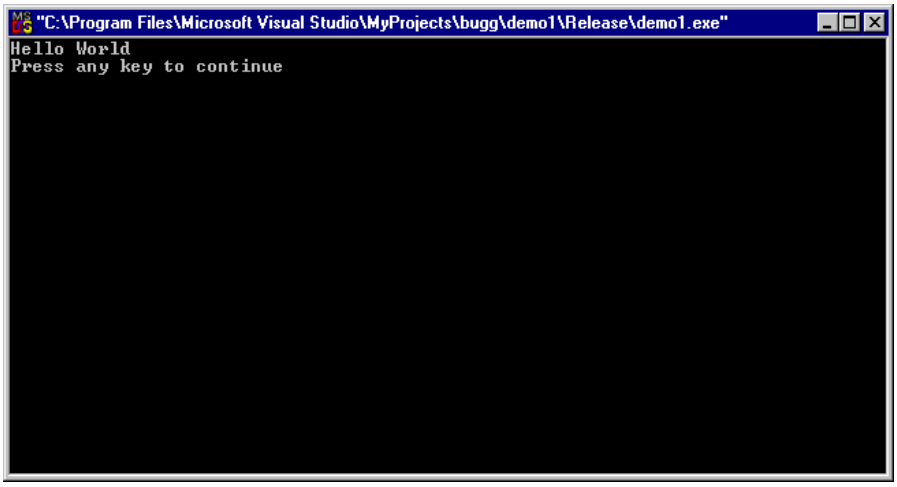
```
#include <iostream.h>

    int main()
    {
            cout << "Hello World \n";
            return 0;
    }
```

6. Now we must compile our code. Go to Build | Compile. This will compile the code, notice that warnings and errors will appear in the bottom window. No errors occurred during compilation so now we will build the executable file and run our program.



7. To build the executable file go to Build | Demo 1. Now the executable has been made and it is time to run the program. Go to Build | Execute Demo1.exe. This will run the program and the output window will open showing you the output.



Two Source Files

1. Now we are going to create a program with multiple files. More specifically, we are going to create one source code file and a header file. We still have our Workspace, "bugg", open with the "demo1" Project. In order to start our new program we need to create a new Project in our Workspace, so we repeat Step 3 from above, except we will name this Project, "demo2". Notice in the FileView window that there are now two projects in our Workspace.We must create our first source file now. Go to File | New and click on the File tab. Highlight the C++ source file. We will call our source file "main". Also, make sure that the Add to Project Box is checked and that you are adding the file to "demo2".

2. When the main source file window opens type the following code:

```
#include "functions.h"

        int main()
        {
                Display();
                return 0;
        }
```

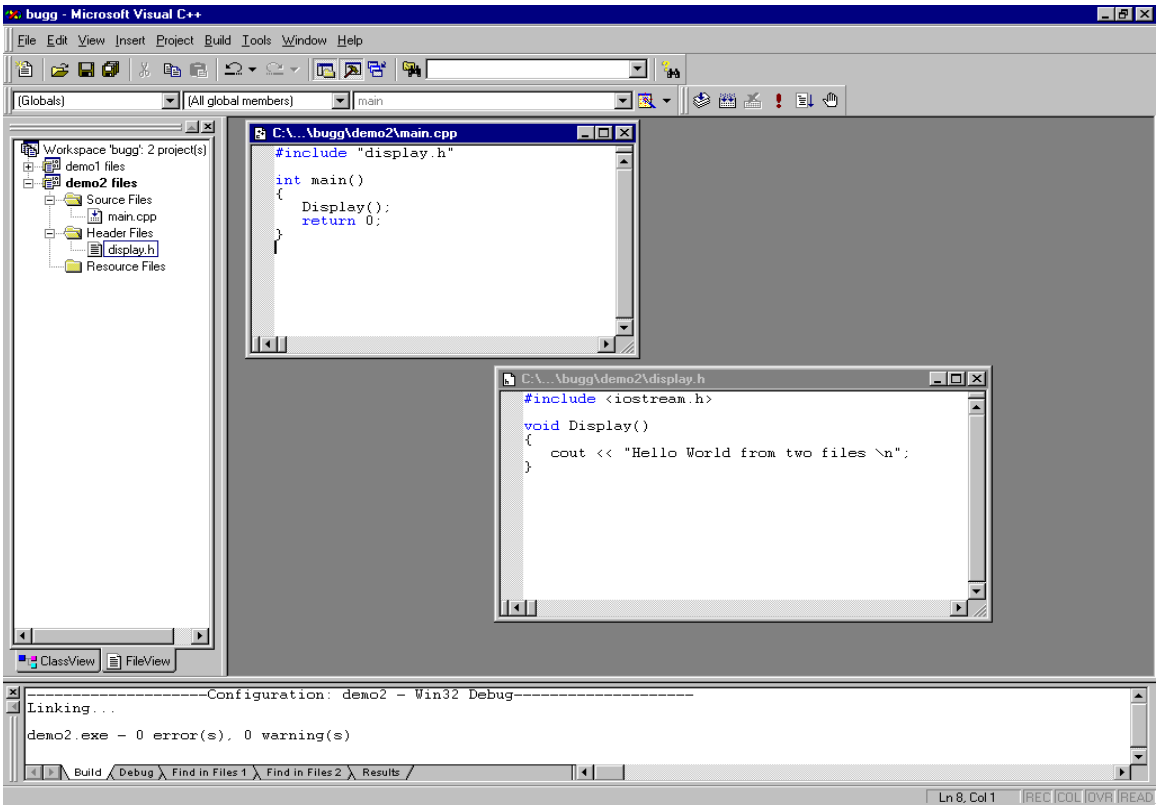
3. Next we need to create the header file for our Display function. Go to File | New and click on the File tab. Highlight C/C++ Header File in the file type list and again make sure the Add to Project box is checked and that the file is being added to "demo2". We will name our header file "display".

4. When the header file window opens type the following code:

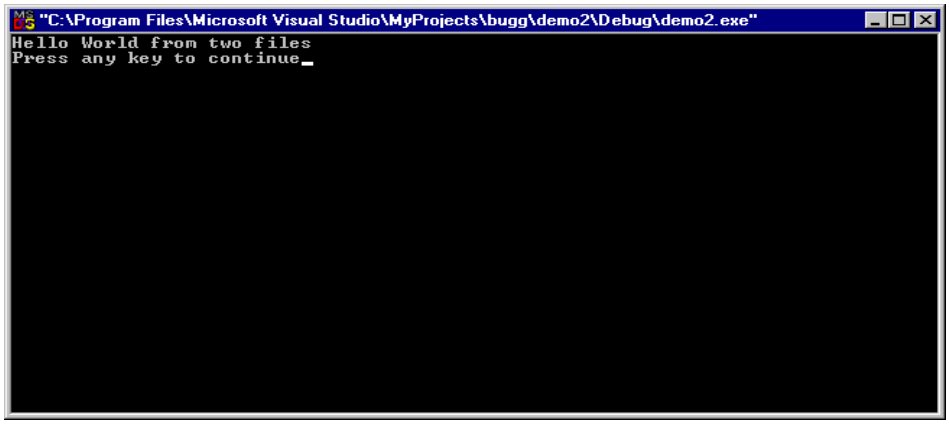
```
#include <iostream.h>

        void Display()
        {
                cout << "Hello World from two files \n";
        }
```

5. The coding of our program is now complete so we will compile and build our application. Make sure that "main.cpp" window is the active window and go to Build | Compile main.cpp. Now we want to create the executable file so go to Build | Build demo2.exe.



6. The last step is to run the executable file and verify that the output is correct. Go to Build | Execute demo2.exe. the output window will then appear with your program's output.



Basic Debugging

Starting & Stopping Debugger

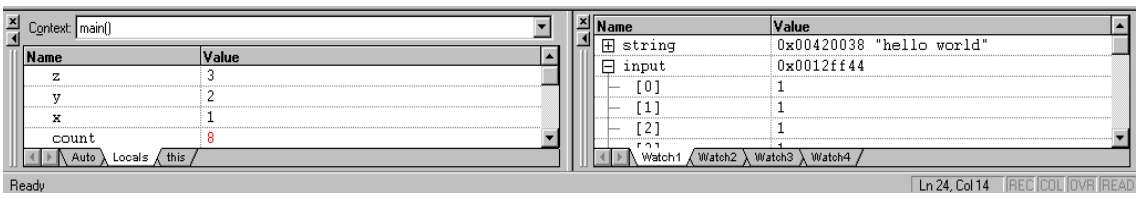
To begin stepping at the first line of your code, click on Start Debug | Step Into (F11). To begin stepping at some other line set a breakpoint at the point where you want to begin, then click on Start Debug | Go (F5). Breakpoints are inserted and removed by placing the keyboard cursor anywhere in the statement, and clicking on the icon that looks like a hand, by pressing F9, or by right clicking the mouse and selecting Insert/Remove Breakpoint. Large circles to the left of a statement denote a breakpoint. To exit debug click on Debug | Stop Debugging (Shift+F5) or click on the icon that looks like source code with an X through it on the debugging mini tool bar. To restart debugging, click on Debug | Restart (Ctrl+Shift+F5).

Stepping

There are three modes of stepping: Step Into (F11), Step Over (F10), or Step Out (Shift F11). These modes allow you to step into a function, step over a function, or step out of a function. Most of the time you will want to Step Into your code and Step Over library functions. If you go into a library function by mistake, you can Step Out. The yellow arrow on the left of your code shows you what statement the debugger is currently on.

Watching Variables

At the bottom of the screen there are two windows that allow you to view the contents of variables. The left window, the Variable Window, automatically displays variables related to the statements being executed when the Auto tab is selected and displays local variables when the Locals tab is selected. The right window, the Watch Window, allows you to enter variables that you want to watch during program execution. When an array or string appears in either window a + will appear, clicking on the + expands the window to show all of values of the elements in the array or the string.



Changing Value of Variables

You can change the value in variables by double clicking on the value given in the lower windows and entering new values. The color of the variable value will change to red to indicate that the value has changed.

Getting Help

F1 Help

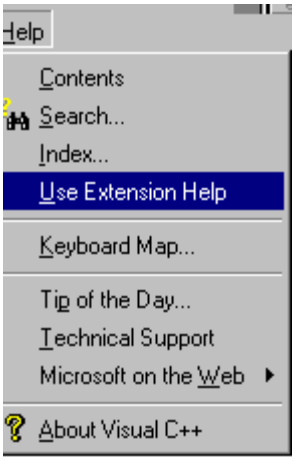
The framework of VC++ implements F1 help for windows, dialog boxes, message boxes, menus and toolbar buttons. If the cursor is over a window, dialog box, or message box when the user presses the F1 key, the framework opens Windows Help for that window. If a menu item is highlighted, the framework opens Windows Help for that menu item. And if a toolbar button has been pressed (but the mouse not released yet), the framework opens Windows help for that toolbar button when the user presses F1.

Shift + F1 Help

If the user presses SHIFT +F1 or clicks on the Help toolbar button at anytime that the application is active, the framework puts the application into Help mode and changes the cursor to a Help cursor. The next thing the user clicks determines what help context the framework opens in Windows Help.

MSDN Help

To access MSDN from within Visual C++, the Help menu's Use Extension help command must be unchecked. Choosing the Contents, Search, or Index command from within Visual C++ Help menu causes the environment to run MSDN.



More Information

World Wide Web

http://www.microsoft.com/visualc

http://mspress.microsoft.com

http://msdn.microsoft.com

Books

Microsoft Visual Studio Core Reference Set, Microsoft Press
ISBN 1-57231-884-8

Microsoft Visual C++ Reference Library, Microsoft Press
ISBN 1-57231-865-1

Acknowledgements

Microsoft Research

Meghan Bellock, Student Consultant – Notre Dame University

Microsoft Visual C++ Programmer's Guide, Beck Zaratian