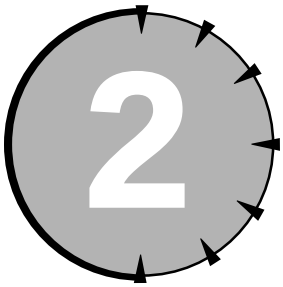
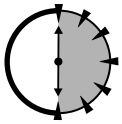
SESSION



Creating Your First C++ Program in Visual C++

Session Checklist

- ✔ Creating your first C++ program using Visual C++
- ✔ Building your C++ source statements into an executable program
- ✔ Executing your program
- ✔ Getting help when programming



30 Min. To Go

Chapter 1 looked at how you might program a human. This chapter and the next describe how to program a computer in C++. This chapter centers on writing programs in Visual C++, while the next concentrates on the public domain GNU C++, which is contained on the enclosed *C++ Weekend Crash Course* CD-ROM.

Don't get too worried about the designation Visual C++ or GNU C++. Both compilers represent true implementations of the C++ standard language. Either compiler can compile any of the programs in this book.

The program we are about to create converts a temperature entered by the user from degrees Celsius to degrees Fahrenheit.

Installing Visual C++

You need to install the Visual C++ package on your computer before you can write a Visual C++ program. The Visual C++ package is used to write your C++ programs and to convert them to .EXE programs that the computer can understand.

Visual C++ does not come with this book. You need to purchase Visual C++ separately, either as a part of the entire Visual Studio package or on its own. The very capable GNU C++ compiler is included.

Consult Appendix A if you need help installing Visual C++.

Creating Your First Program

A C++ program begins life as a text file containing the C++ instructions. I will lead you step-by-step through this first program.

Start the Visual C++ package. For Visual Studio 6.0, click Start followed by Programs and the Microsoft Visual Studio 6.0 menu options. From there, select Microsoft Visual C++ 6.0.

Visual C++ should start with two empty windows labeled Output and WorkSpace. If other windows appear or Output or WorkSpace is not empty, then someone has been using your Visual C++ on your machine. To close out whatever they were doing select File followed by Close Workspace.

Create an empty text file by clicking on the small New Text File icon at the left of the menu bar as shown in Figure 2-1.

Don't worry too much about indentation — it isn't critical whether a given line is indented two spaces or three spaces; case, however, is critical. C++ does not consider "Cheat" and "cheat" to be the same word.

(You can cheat and copy the Conversion.cpp file contained on the accompanying CD-ROM.)

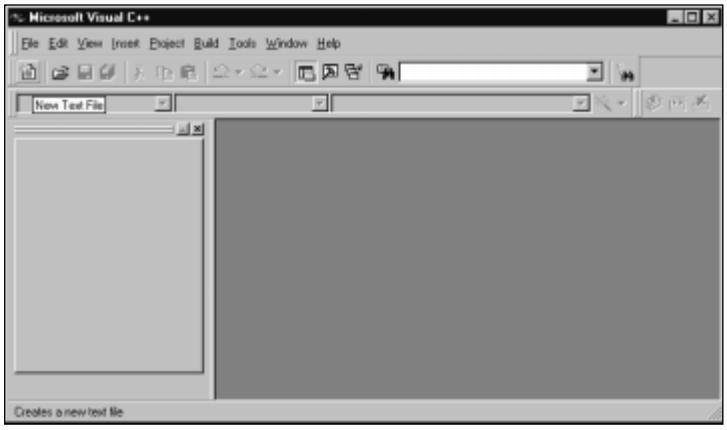


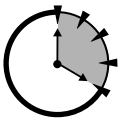
Figure 2-1
You begin writing your C++ program by using the New Text File button to create an empty text file.

Enter the following program exactly as written below.

```
//
//  Program to convert temperature from Celsius degree
//      units into Fahrenheit degree units:
//      Fahrenheit = Celsius  * (212 - 32)/100 + 32
//
#include <stdio.h>
#include <iostream.h>
int main(int nNumberofArgs, char* pszArgs[])
{
    // enter the temperature in Celsius
    int nCelsius;
    cout << "Enter the temperature in Celsius:";
    cin > nCelsius;

    // calculate conversion factor for Celsius
    // to Fahrenheit
    int nFactor;
    nFactor = 212 - 32;

    // use conversion factor to convert Celsius
    // into Fahrenheit values
```

```
    int nFahrenheit;
    nFahrenheit = nFactor * nCelsius/100 + 32;

    // output the results
    cout << "Fahrenheit value is:";
    cout << nFahrenheit;

    return 0;
}
```

Save the file under the name Conversion.cpp. The default directory is in one of the Visual Studio folders. I prefer to navigate to a folder that I created in a more convenient spot before saving the file.



20 Min. To Go

Building Your Program

We used a limited set of commands in Session 1 to instruct the human computer in changing the tire of a car. Although restricted, even these instructions were understandable to the average human (at least the average English-speaking human).

The Conversion.cpp program you just entered contains C++ statements, a language that doesn't look much like anything you would read in the morning paper. As cryptic and crude as these C++ commands might appear to be, the computer understands a language much more basic than even C++. The language your computer processor understands is known as *machine language*.

The C++ compiler converts your C++ program into the machine language of the microprocessor CPU in your PC. Programs you can execute from the Programs option of the Start menu, including Visual C++ itself, are nothing more than files consisting of these machine instructions.

It is possible to write a program directly in machine language, but it is much more difficult to do than it is to write the same program in C++.

The primary job of your Visual C++ package is to convert your C++ program into an executable file. The act of creating an executable .EXE is known as *building*. The build process is also known as *compiling* (there is a difference, but it is not relevant at this point). That part of the C++ package that performs the actual build process is known as the *compiler*.

To build your Conversion.cpp program, click the Build menu item under the Build menu option. (No, I was not stuttering.) Visual C++ responds by warning you that you have yet to create a Workspace, whatever that is. This is shown in Figure 2-2.

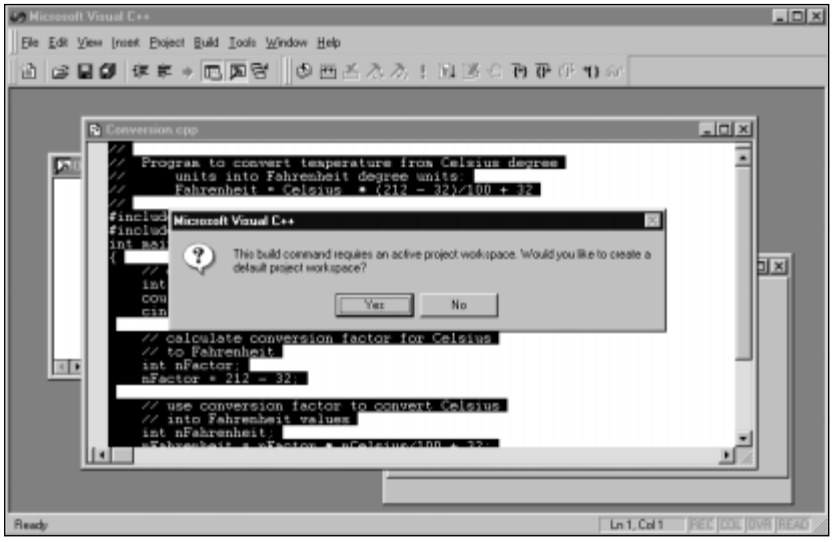


Figure 2-2
A workspace is required before Visual C++ can build your program.

Click Yes to create a Workspace file and to continue the build process.

The .cpp source code file is nothing more than a text file, similar to what you would build using Notepad. The Conversion.pdw Workspace that Visual C++ creates is a file in which Visual C++ can save special information about your program, information that will not fit anywhere in the Conversion.cpp file.

After a few minutes of frantic disk activity, Visual C++ responds with a pleasant bell ring sound, which indicates that the build process is complete. The output window should contain a message similar to that shown in Figure 2-3, indicating that Conversion.exe was created with 0 errors and 0 warnings.

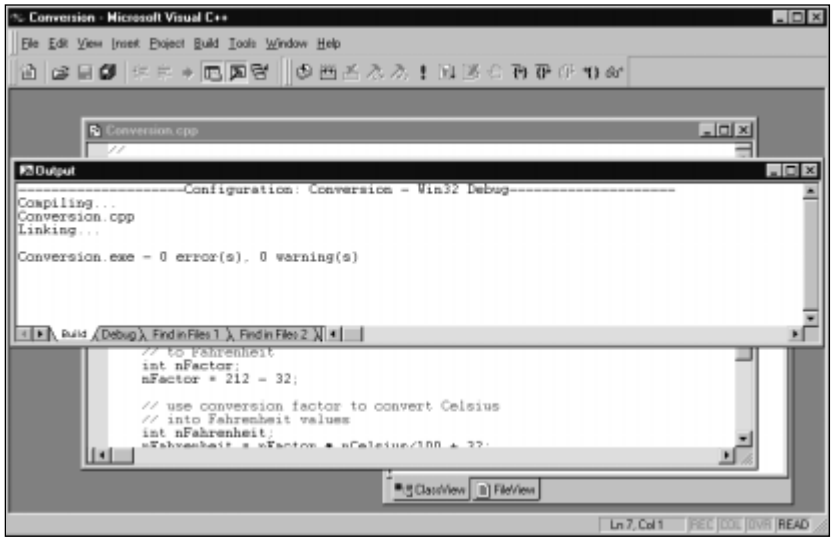


Figure 2-3
The "0 errors 0 warnings" message in the Output window indicates a successful build.

Visual C++ generates an unpleasant buzz if it detects anything wrong during the build process (at least, Microsoft thinks it's unpleasant — I've heard it so many times, it's starting to grow on me). In addition, the output window contains an explanation of what Visual C++ found wrong.

I removed a semicolon at the end of one of the lines in the program and recompiled just to demonstrate the error reporting process. The result is shown in Figure 2-4.

The error message displayed in Figure 2-4 is actually quite descriptive. It accurately describes the problem ("missing ; . . .") and the location (line 18 of the file Conversion.cpp). I replaced the semicolon and rebuilt the program to solve the problem.

Not all error messages are quite as clear as this one. Many times a single error can create a number of error messages. At first, these error messages seem confusing. Over time, however, you get a feel for what Visual C++ is "thinking" during the build process and what might be confusing it.

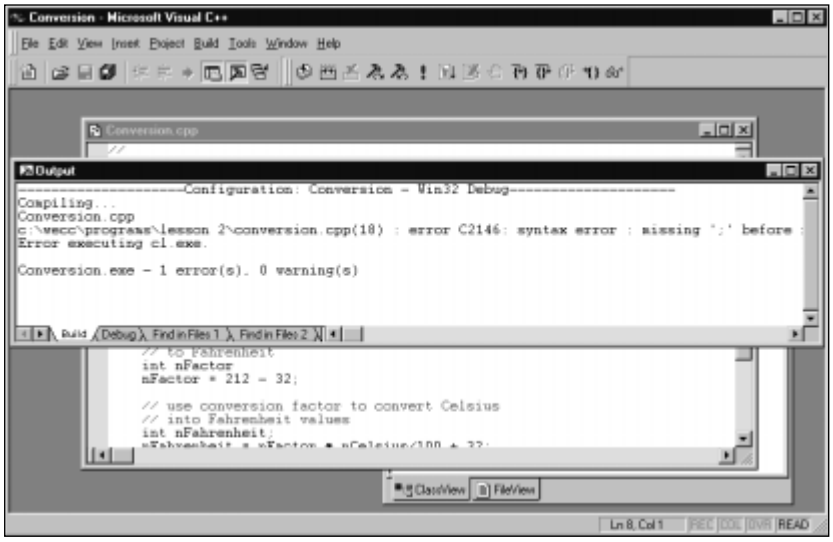


Figure 2-4
Visual C++ reports errors found during the build process in the Output window.

You will undoubtedly hear the unpleasant buzz of an error detected by Visual C++ before you eventually get Conversion.cpp entered correctly. Should you never get the code entered in a way that Visual C++ approves, you may copy the Conversion.cpp file from xxx\Session 5\Conversion.cpp of the enclosed CD-ROM.

C++ Error Messages

Why are all C++ packages, including Visual C++, so picky when it comes to C++ syntax? If Visual C++ can figure out that I left off a semicolon, why can't it just fix the problem and go on?

The answer is simple but profound. Visual C++ *thinks* that you left off a semicolon. I could have introduced any number of errors that Visual C++ might have misdiagnosed as a missing semicolon. Had the compiler simply "corrected" the problem by introducing a semicolon, Visual C++ would have masked the real problem.

Continued

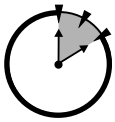
C++ Error Messages *Continued*

As you will see, finding an error buried in a program that builds without error is difficult and time consuming. It is far better to let the compiler find the error, if possible.

This lesson was hard in coming. Early in the days of computing, compilers tried to detect and correct any error that they could find. This sometimes reached ridiculous proportions.

My friends and I loved to torture one "friendly" compiler in particular by entering a program containing nothing but the existential question of the ages IF. (In retrospect, I guess my friends and I were nerdy.) Through a series of tortured gyrations, this particular compiler would eventually create an involved command line from this single word that would build successfully. I know that the compiler misunderstood my intent with the word IF because I didn't intend a single thing.

In my experience, almost every single time the compiler tried to "fix" my program, it was wrong. Although misguided, fixing the program was harmless if the compiler reported the error before fixing it. Compilers that corrected errors without reporting them did much more harm than good.



10 Min. To Go

Executing Your Program

You can execute the successfully built Conversion.exe by clicking Execute Conversion.exe item under the Build menu. Alternatively, you can press Ctrl+F5.

Avoid using the Go menu command or the equivalent F5 key for now.

Visual C++ opens a program window similar to that shown in Figure 2-5, requesting a temperature in degrees Celsius.

Enter a temperature, such as 100 degrees. After pressing Enter, the program responds with the equivalent measurement in degrees Fahrenheit as shown in Figure 2-6. The "Press any key to terminate" message jammed up against the temperature output is not aesthetically pleasant, but the converted temperature is unmistakable — we fix this blemish in Chapter 5.

Figure 2-5
The Conversion.exe program begins by requesting a temperature to convert.

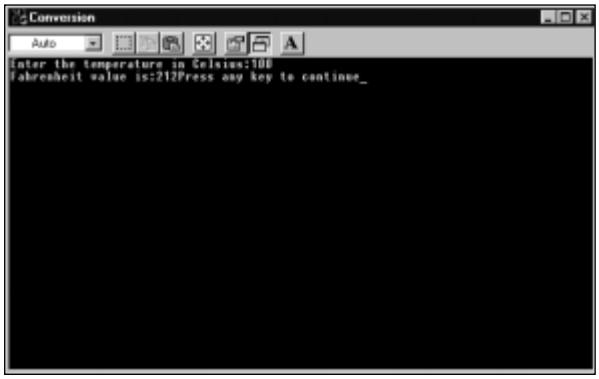


Figure 2-6
The Conversion.exe program waits for user input after the program terminates.

The "Press any key to terminate" prompt gives the user time to read the program output before closing the window after the program terminates. This message does not appear when using the Go command available through the F5 key.

Congratulations! You have just entered, built, and executed your first program.

Closing Points

There are two points worth noting before continuing. First, the form of the program output from Conversion.exe might surprise you. Second, Visual C++ offers a lot more help than just build error messages.

Program output

Windows programs have a very visually oriented, windows-based output. Conversion.exe is a 32-bit program that executes under Windows, but it is not a "Windows" program in the visual sense.

If you don't know what is meant by the phrase "32-bit program," don't worry about it.

As I pointed out in the introduction, this is not a book about writing Windows programs. The basic C++ programs that you write in this book have a command line interface executing within a DOS box.

Budding Windows programmers should not despair — you did not waste your money. Learning C++ is a prerequisite to writing Windows programs.

Visual C++ help

Visual C++ offers a help system that gives significant support to the C++ programmer. To see how this help works, double-click on the word `#include` until it is completely highlighted. Now press F1.

Visual C++ responds by opening the MSDN Library and displaying an entire page of information about `#include` as shown in Figure 2-7 (you probably don't understand all of what it's saying, but you soon will).

Done!

You can find the same information by selecting Index. . . under the Help menu. Enter `#include` in the index window which appears to reveal the same information.

As you grow as a C++ programmer, you will rely more and more on the MSDN Library help system.

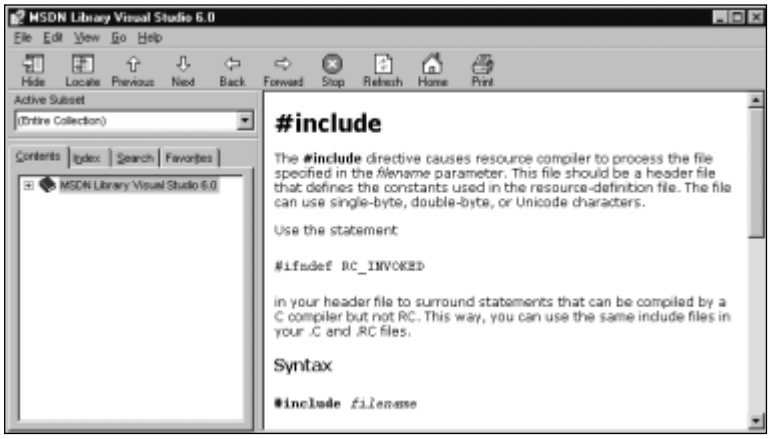


Figure 2-7
The F1 key provides significant help to the C++ programmer.

REVIEW

Visual C++ 6.0 has a user-friendly environment in which you can create and test your programs. You use the Visual C++ editor to enter your program's source code. Once entered, your C++ statements are converted into an executable file through a building process. Finally, you can execute your completed program from within Visual C++.

In the next chapter, we look at how we to create the same program using the GNU C++ compiler, which is found on the accompanying CD-ROM. If you are definitely committed to Visual C++, you may want to skip ahead to Session 4, which explains exactly how the program you just entered works.

QUIZ YOURSELF

1. What kind of file is a C++ source program? (That is, is it a Word file? An Excel spreadsheet file? A text file?) (See the first paragraph of Creating Your First Program.)
2. Does C++ care about indention? Does it care about case? (See Creating Your First Program.)

3. What does "building your program" mean? (See Building Your Program.)
4. Why does C++ generate error messages? Why can't it just try to make sense out of what I enter? (See the C++ Error Messages sidebar.)