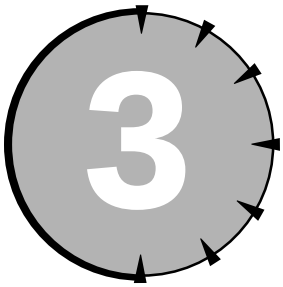
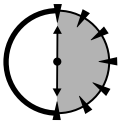
SESSION



Creating Your First C++ Program in GNU C++

Session Checklist

- ✔ Installing GNU C++ from the enclosed CD-ROM
- ✔ Creating your first C++ program using GNU C++
- ✔ Building your C++ source statements into an executable program
- ✔ Executing your program



30 Min. To Go

Chapter 2 went through the steps to write, build, and execute a Visual C++ program. Many readers of *C++ Weekend Crash Course* will not have access to Visual C++. For those readers, this book includes the capable public domain GNU C++ on the enclosed CD-ROM.

"GNU" is pronounced "guh-new." GNU stands for the circular definition "GNU is Not Unix." This joke goes way back to the early days of C++ — just accept it as is. GNU is a series of tools built by the Free Software Foundation. These tools are available to the public with some restrictions but without cost.

This chapter goes through the steps necessary to turn the same Conversion.cpp program demonstrated in Chapter 2 into an executable program using GNU C++.

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

Installing GNU C++

The CD-ROM that accompanies this book includes the most recent version of the GNU C++ environment at the time of this writing. The installation instructions are included in Appendix C; this session provides instructions for downloading and installing GNU C++ from the Web.

The GNU environment is maintained by a number of dedicated volunteer programmers. If you prefer you may download the most recent version of GNU C++ from the web.

The GNU development environment is an extremely large package. GNU includes a number of utilities and programming languages besides C++. GNU C++ alone supports a number of computer processors and operating systems. Fortunately you don't need to download all of GNU in order to develop C++ programs. The GNU environment is broken up into a number of ZIP files. The "ZIP picker" utility, provided by Delorie Software on their Web site, calculates the ZIP files that you need to download based on the answers to a set of simple questions.

To install GNU C++ from the Web:

1. Surf the web page `http://www.delorie.com/djgpp/zip-picker.html`.
2. The site shows you the questionnaire reproduced below. Answer the zip-picker's questions as shown in bold to install a minimum configuration:

   FTP Site

   Select a suitable FTP site: **Pick one for me**

   Basic Functionality

   Pick one of the following: **Build and run programs with DJGPP**

   Which operating system will you be using? **<your version of Windows>**

   Do you want to be able to read the on-line documentation? : **No**

   Which programming languages will you be using? **Click C++**

   Integrated Development Environments and Tools

   Which IDE(s) would you like? **Click on RHIDE. Leave the emacs options unchecked.**

   Would you like `gdb`, the text-mode GNU debugger? **No**

Extra Stuff

Please check off each extra thing that you want. **Don't check anything in this list.**

3. Next, click on "Tell me what files I need". The ZIP picker responds with some simple installation instructions plus a list of the zip files that you will need. The listing below shows the files necessary to implement the minimal installation described here — the file names that you receive will differ to reflect the current version number.

```
Read the file README.1ST before you do anything else with
DJGPP! It has important installation and usage instructions.

v2/djdev202.zip      DJGPP Basic Development Kit   1.4 mb
v2/faq211b.zip       Frequently Asked Questions    551 kb

v2apps/rhide14b.zip  RHIDE                         1.6 mb

v2gnu/bnu281b.zip    Basic assembler, linker       1.8 mb
v2gnu/gcc2952b.zip   Basic GCC compiler            1.7 mb
v2gnu/gpp2952b.zip   C++ compiler                  1.6 mb
v2gnu/lgpp295b.zip   C++ libraries                 484 kb
v2gnu/mak377b.zip    Make (processes makefiles)    242 kb

Total bytes to download: 9,812,732
```

4. Create a folder named \DJGPP.
5. Download each of the ZIP files listed by the ZIP picker in the DJGPP directory by clicking on the name of the file.
6. Unzip the files into the DJGPP folder itself.
7. Add the following commands to AUTOEXEC.BAT:

```
set PATH=C:\DJGPP\BIN;%PATH%
set DJGPP=C:\DJGPP\DJGPP.ENV
```

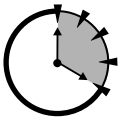
   Note: the above command lines assume your DJGPP folder is directly under C:\. If you've placed your DJGPP folder somewhere else, substitute that path in the commands above.

8. Reboot to complete the installation.

The \BIN folder includes the actual GNU tool executables. The DJGPP.ENV file sets a series of options to describe the Windows GNU C++ "environment."

Before you begin using GNU C++, check the DJGPP.ENV file to make sure that Long File Name support is enabled. Disabling Long File Name support is the most common GNU C++ installation error.

Open the DJGPP.ENV file using a text file editor such as Microsoft Notebook. Don't worry if you see one long string of text punctuated by little black boxes—Unix uses a different newline character than Windows. Look for the phrase "LFN=y" or "LFN=Y" (the case is not important). If you find "LFN=n" instead (or if you don't find "LFN" at all), change the "n" to a "y". Save the file. (Make sure that you save the file as an ASCII text file and not in some other format such as a Word .DOC file.)



20 Min. To Go

Creating Your First Program

The heart of the GNU C++ package is a utility known as `rhide`. At its core `rhide` is just an editor that links to the remaining pieces of GNU C++ into an integrated Visual C++–like package.

Entering the C++ code

Open an MS-DOS window by clicking on the MS-DOS icon under the Programs menu. Create a directory where you would like your program created. I created the directory c:\wecc\programs\lesson3. In this directory, enter the command `rhide` at the MS-DOS prompt.

The rhide Interface

The `rhide` interface is fundamentally different in appearance than that of a Windows-oriented program. Windows programs "paint" their output to the screen, which gives Windows programs a more refined appearance.

By comparison, the `rhide` interface is based on characters. `rhide` uses a number of blocking characters available in the PC arsenal to simulate a Windows interface, which gives `rhide` a less elegant appearance. For example, `rhide` does not support resizing the window away from the 80×25 character display which is the standard for MS-DOS programs.

For those of you old enough to remember, the `rhide` interface looks virtually identical to the interface of the now defunct Borland suite of programming tools.

Nevertheless the `rhide` interface is functional and provides convenient access to the remaining GNU C++ tools.

Create an empty file by entering New under the File menu. Enter the following program exactly as written.

Don't worry too much about indentation or spacing — it isn't critical whether a given line is indented two spaces or three spaces, or whether there is one or two spaces between two words.

You can cheat and copy the Conversion.cpp file that is found on the enclosed CD-ROM.

```
//
//  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;
}
```

Once you have completed, your `rhide` window should appear like that shown in Figure 3-1.

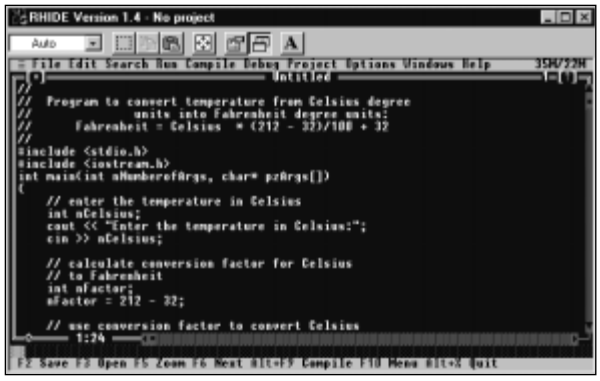


Figure 3-1
rhide provides a character-based user interface for creating GNU C++ programs.

Select Save As. . . under the File menu as shown in Figure 3-2 to save the file under the name Conversion.cpp.

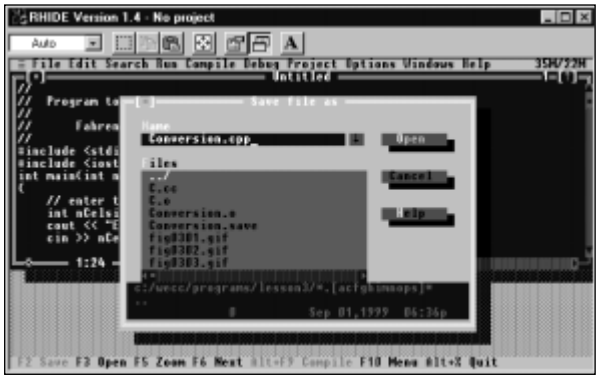


Figure 3-2
The Save As. . . command enables the user to create C++ source files.

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 to the machine language of the microprocessor CPU in your PC. Programs that 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 to write the same program in C++.

The primary job of your GNU C++ package is to convert your C++ program to an executable machine instruction file.

The act of creating an executable .EXE is known as *building*. The build process is also known as *compiling* or *making* in the Unix world (there is a difference between the three, but the differences are 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 Compile and then click Make or press F9. `rhide` opens a small window at the bottom of the current window to display the progress of the build process. If all goes well, the message "Creating Conversion.exe" followed by "no errors" appears as shown in Figure 3-3.

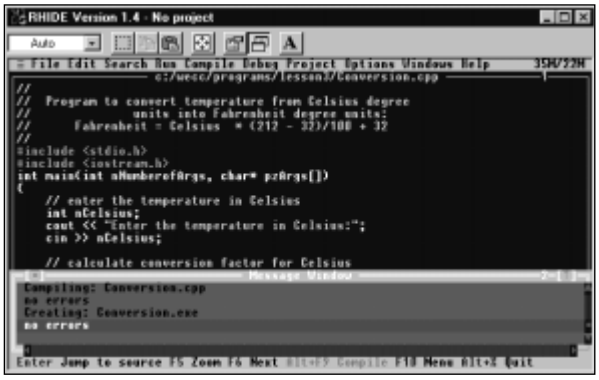


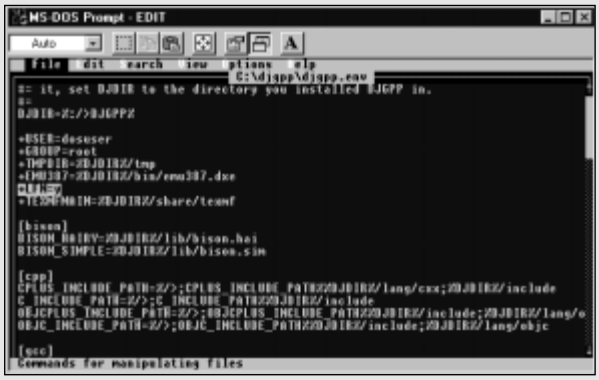
Figure 3-3
rhide displays the "no errors" message if the build process is successful.

GNU C++ Installation Errors

A number of common errors might occur during installation to spoil your "out-of-the-box" programming experience with inexplicable errors.

The message "Bad command or file name" means that MS-DOS can't find gcc.exe, the GNU C++ compiler. Either you did not install GNU C++ properly or your path does not include c:\djgpp\bin where gcc.exe resides. Try reinstalling GNU C++ and make sure that the command `SET PATH=c:\djgpp\bin;%PATH%` is in your autoexec.bat file. After reinstalling GNU C++, reboot your computer.

The message "gcc.exe: Conversion.cpp: No such file or directory (ENOENT)" indicates that gcc does not know that you are using long filenames (as opposed to old MS-DOS 8.3 filenames). To correct this problem, edit the file c:\djgpp\djgpp.env. Set the LFN property to Y as shown in this figure.



GNU C++ generates an error message if it finds an error in your C++ program. To demonstrate the error reporting process, I removed a semicolon at the end of one of the lines in the program and recompiled. The result is shown in Figure 3-4 (line 4 should read `nFactor = 212 - 32;` with the semicolon)

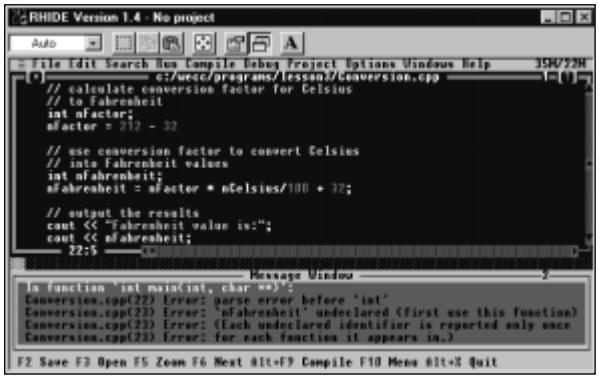


Figure 3-4
GNU C++ reports errors found during the build process in the rhide output window.

The error message that is reported in Figure 3-4 is a little imposing; however, it is reasonably descriptive, if you take one line at a time.

The first line indicates that it detected the problem while it was analyzing code contained within `main()`, that is, code found between the open and closed braces following the keyword `main()`.

The second line indicates that it couldn't understand how `int` on line 22 fit into the scheme of things. Of course, `int` doesn't fit, but without the semicolon, GNU C++ thought that line 18 and 22 were one statement. The remaining errors stem from not being able to understand line 22.

To fix the problem, I first analyzed line 22 (notice the line 22:5 at the lower left of the code window — the cursor is on column 5 of line 22). Because line 22 seems to be in order, I look back up to line 18 and notice that a semicolon is missing. I add the semicolon and recompile. This time, GNU C++ completes without complaint.

C++ Error Messages

Why are all C++ packages so fussy when it comes to C++ syntax? Visual C++ was able to determine without a doubt that I had removed a semicolon in the previous example. However, if a C++ compiler can figure out that I left off a semicolon, then why doesn't it just fix the problem and go on?

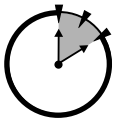
The answer is simple but profound. Visual C++ *thinks* that you left off a semicolon. I could have introduced any number of errors that Visual C++ might have misdiagnosed as a missing semicolon. Had the compiler simply "corrected" the problem by introducing a semicolon, Visual C++ would have masked the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

As you will see, finding an error buried in a program that builds without error is difficult and time consuming. It is far superior 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 of these "friendly" compilers 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 time that the compiler tried to "fix" my program, the compiler was wrong. Although misguided, this is harmless if the compiler reports the error before fixing it. Compilers that correct errors without reporting them do much more harm than good.



10 Min. To Go

Executing Your Program

To execute the Conversion program, click Run and Run again or enter Ctrl+F9 as shown in Figure 3-5.

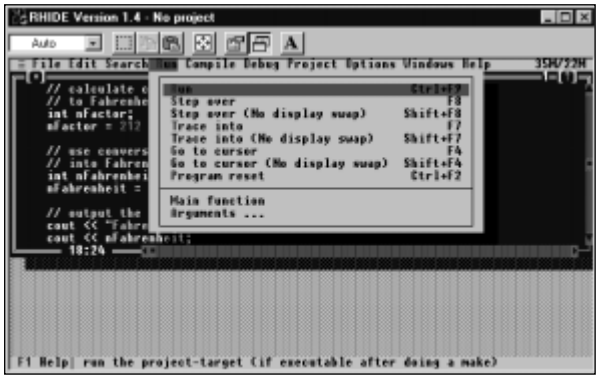


Figure 3-5
rhide opens a window in which it executes the current program.

Immediately a window opens in which the program requests a temperature in Celsius as shown in Figure 3-6.

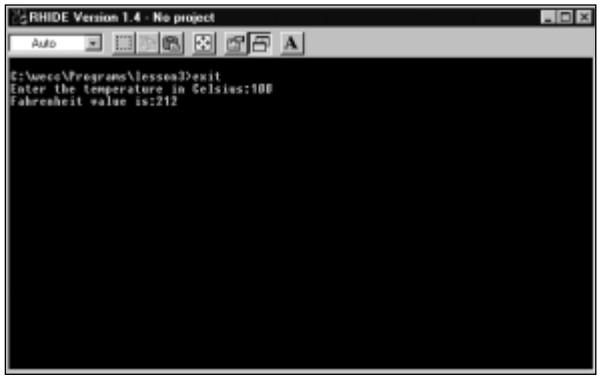


Figure 3-6
The user screen displays the calculated temperature in Fahrenheit degrees.

Enter a known temperature such as 100 degrees. After pressing the Enter key, the program returns with the equivalent temperature of 212 degrees Fahrenheit. However, because `rhide` closes the window as soon as the program terminates, you are not given a chance to see the output before the window closes. `rhide` opens an alert box with the message that the program terminated with an error code of zero. Despite the name "error code," a zero means no error occurred.

To see the output from the now terminated program, click the User Screen menu item of the Windows menu or enter Alt+F5. This window displays the current MS-DOS window. In this window, you see the last 25 lines of output of the program including the calculated Fahrenheit temperature as shown in Figure 3-6.

Congratulations! You have just entered, built, and executed your first program using GNU C++.

Closing Points

There are two points to emphasize. First, GNU C++ is not intended for developing Windows programs. In theory, you could write a Windows application using GNU C++, but it wouldn't be easy without the help provided by the Visual C++ libraries. Second, GNU C++ provides a type of help that can be very helpful.

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 C++ programs that you write in this book have a command-line interface executing within an MS-DOS box.

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

Done!

GNU C++ help

GNU C++ provides a help system through the `rhide` user interface. Place your cursor on a construct that you don't understand and press F1; a window pops up like that shown in Figure 3-7. Alternatively, click Index under Help to display a list of help topics. Click the topic of interest to display help.

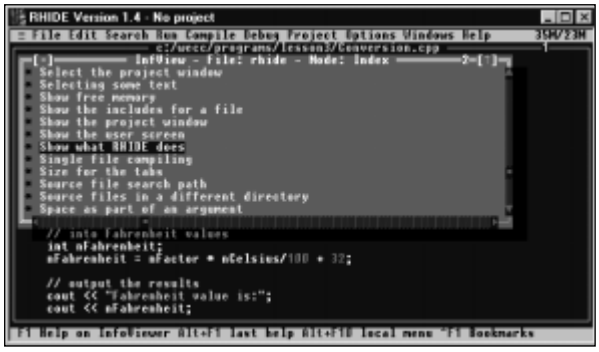


Figure 3-7
rhide supports F1- and Index-type help.

The help provided by GNU C++ is not nearly as comprehensive as that provided by Visual C++. For example, place the cursor on the `int` statement and press F1. What appears is a window describing the editor, not exactly what I was looking for. The help provided by GNU C++ tends to center on library functions and compiler options. Fortunately, once you have mastered the C++ language itself, GNU C++ help is satisfactory for most applications.

REVIEW

GNU C++ provides a user-friendly environment in which you can create and test your programs in the form of the `rhide` utility. You can use `rhide` in much the same way that you would use Visual C++. You use the `rhide` editor to enter the code and the builder to convert the source code into machine code. Finally, `rhide` provides the capability to execute the final program from within the same environment.

The next chapter goes over the C++ program step by step.

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 is entered? (See the "C++ Error Messages" sidebar.)