Chapter 2. Basics of Program Writing

Programs start as a set of instructions written by a human being. Before they can be used by the computer, they must undergo several transformations. In this chapter, we'll learn how to enter a program, transform it into something the machine can use, and run it. Detailed steps are provided for the most popular UNIX and DOS/Windows compilers.

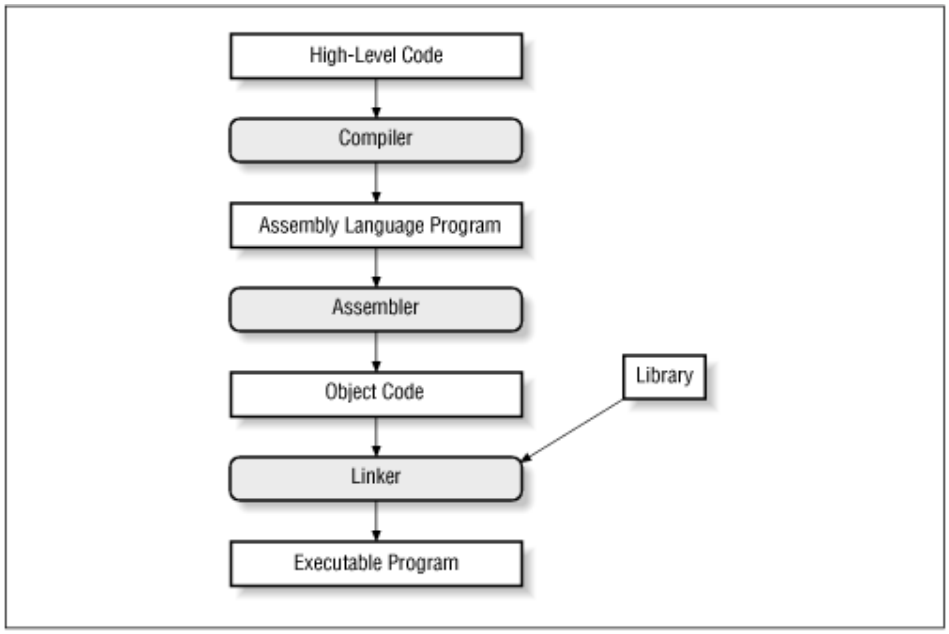
2.1 Programs from Conception to Execution

C programs are written in a high-level language using letters, numbers, and the other symbols you find on a computer keyboard. Computers actually execute a very low-level language called *machine code* (a series of numbers). So, before a program level can be used, it must undergo several transformations.

Programs start out as an idea in a programmer's head. He uses a text editor to write his thoughts into a file called a *source file*, containing *source code.* This file is transformed by the *compiler* into an *object file*. Next, a program called the *linker* takes the object file, combines it with predefined routines from a *standard library,* and produces an *executable program* (a set of machine-language instructions). In the following sections, we'll see how these various forms of the program work together to produce the final program.

Figure 2-1 shows the steps that must be taken to transform a program written in a high-level language into a executable program.

Figure 2-1. Transformation of a high-level language into a program



2.1.1 Wrappers

Fortunately you don't have to run the compiler, assembler, and linker individually. Most C compilers use "wrapper" programs that determine which tools need to be run and then run them.

Some programming systems go even further and provide the developer with an Integrated Development Environment (IDE). The IDE contains an editor, compiler, linker, project manager, debugger, and more in one convenient package. Both Borland and Microsoft provide IDEs with their compilers.

2.2 Creating a Real Program

Before we can actually start creating our own programs, we need to know how to use t he basic programming tools. In this section, we will take you step by step through the process of entering, compiling, and running a simple program.

We will describe how to use two different types of compilers. The first type is the standalone or command-line compiler. This type of compiler is operated in a batch mode from the command line. In other words, you type in a command, and the compiler turns your source code into an executable program.

The other type of compiler is contained in an IDE. The IDE contains an editor, compiler, project manager, and debugger in one package.

Most UNIX systems use command-line compilers. There are a few IDE compilers available for UNIX, but they are rare. On the other hand, almost every compiler for MS-DOS/Windows contains an IDE. For the command-line die-hards, these compilers do contain a command-line compiler as well.

2.3 Creating a Program Using a Command-Line Compiler

In this section, we'll go through the step-by-step process needed to create a program using a command-line compiler. Instructions are provided for a generic UNIX compiler (cc), the Free Software Foundation's gcc compiler, Turbo C++, Borland C++, and Microsoft Visual C++.[1]

[1] Turbo C++, Borland C++, and Microsoft Visual C++ are all C++ compilers that can also compile C code.

However, if you are using a Borland or Microsoft compiler, you might want to skip ahead to the section on using the IDE.

2.3.1 Step 1. Create a Place for Your Program

You can more easily manage things if you create a separate directory for each program that you're working on. In this case, we'll create a directory called hello to hold our hello program.

On UNIX type:

```
% mkdir hello
% cd hello
```

On MS-DOS type:

```
C:> MKDIR HELLO
C:> CD HELLO
```

2.3.2 Step 2. Create the Program

A program starts out as a text file. Example 2-1 shows our program in source form.

Example 2-1. *hello/hello.c*

```
[File: hello/hello.c]
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return (0);
}
```

Use your favorite text editor to enter the program. Your file should be named *hello.c*.

MS-DOS/Windows users should *not* use a word processor such as MS-Word or WordPerfect to write their programs. Word processors add formatting codes to files, which confuse the compiler. You must use a text editor such as the MS-DOS "EDIT" program that is capable of editing ASCII files.

2.3.3 Step 3. Run the Compiler

The compiler takes the source file you've just made and converts it into an executable program. Each compiler has a different command line. The commands for the most popular compilers are listed below.

2.3.3.1 UNIX cc compiler (generic UNIX)

Most UNIX-based compilers follow the same generic standard. The C compiler is named cc, and to compile our hello program we need the following command:

```
% cc -g -ohello hello.c
```

The `-g` option enables debugging. (The compiler adds extra information to the program to make the program easier to debug.) The switch `-ohello` tells the compiler that the program is to be called `hello`, and the final `hello.c` is the name of the source file. See your compiler manual for details on all the possible options.

There are several different C compilers for UNIX, so your command line may be slightly different.

2.3.3.2 Free Software Foundation's gcc compiler

The Free Software Foundation, the GNU people, publish a number of high-quality programs. (See the Glossary entry for information on how to get their software.) Among their offerings is a C compiler called `gcc`.

To compile a program using the `gcc` compiler use the following command line:

```
% gcc -g -Wall -ohello hello.c
```

The additional switch `-Wall` turns on the warnings.

The GNU compiler contains several extensions to the basic C language. If you want to turn these features *off,* use the following command line:

```
% gcc -g -Wall -ansi -pedantic -ohello hello.c
```

The switch `-ansi` turns off features of GNU C that are incompatible with ANSI C. The `-pedantic` switch causes the compiler to issue a warning for any non-ANSI feature it encounters.

2.3.3.3 Borland's Turbo C++ under MS-DOS

Borland International makes a low-cost MS-DOS C++ compiler called Turbo C++. This compiler will compile both C and C++ code. We will describe only how to compile C code. Turbo C++ is ideal for learning. The command line for Turbo C++ is:

```
C:> tcc -ml -v -N -w -ehello hello.c
```

The `-ml` tells Turbo C++ to use the large-memory model. (The PC has a large number of different memory models. Only expert PC programmers need to know the difference between the various models. For now, just use the large model until you know more.)

The `-v` switch tells Turbo C++ to put debugging information in the program. Warnings are turned on by `-w`; stack checking is turned on by `-N`. Finally `-ehello` tells Turbo C++ to create a program named `HELLO` with `hello.c` being the name of the source file. See the Turbo C++ reference manual for a complete list of options.

Windows Programming

You may wonder why we describe MS-DOS programming when Windows is widely used. We do so because programming in Windows is much more complex than programming in MS-DOS.

For example, to print the message "Hello World" in MS-DOS, you merely print the message.

In Windows, you must create a window, create a function to handle the messages from that window, select a font, select a place to put the font, and output the message.

You must learn to walk before you can run. Therefore, we limit you to the MS-DOS or Easy-Win (Simplified Windows) programs in this book.

2.3.3.4 Borland C++ under MS-DOS and Windows

In addition to Turbo C++, Borland International also makes a full-featured, professional compiler for MS-DOS/Windows called Borland C++. Its command line is:

```
C:> bcc -ml -v -N -P -w -ehello hello.c
```

The command-line options are the same for both Turbo C++ and Borland C++.

2.3.3.5 Microsoft Visual C++

Microsoft Visual C++ is another C++/C compiler for MS-DOS/Windows. To compile, use the following command line:

```
C:> cl /AL /Zi /W1 hello.c
```

The `/AL` option tells the program to use the large memory model. Debugging is turned on with the `/Zi` option and warnings with the `/W1` option.

2.3.4 Step 4. Execute the Program

To run the program (on UNIX or MS-DOS/Windows) type:

```
% hello
```

and the message:

```
Hello World
```

will appear on the screen.

2.4 Creating a Program Using an Integrated Development Environment

Integrated Development Environments (IDEs) provide a one-stop shop for programming. They take a compiler, editor, and debugger and wrap them into one neat package for the program.

2.4.1 Step 1. Create a Place for Your Program

You can more easily manage things if you create a separate directory for each program that you're working on. In this case, we'll create a directory called HELLO to hold our hello program.

On MS-DOS type:

```
C:> MKDIR HELLO
C:> CD HELLO
```

2.4.2 Step 2. Enter, Compile, and Run Your Program Using the IDE

Each IDE is a little different, so we've included separate instructions for each one.

2.4.2.1 Turbo C++

1. Start the Turbo C++ IDE with the command:

   ```
   C:> TC
   ```

2. Select the Window|Close All menu item to clear the desktop of any old windows. We'll want to start clean. The screen should look like Figure 2-2.

Figure 2-2. Clean desktop

3. Select the *Options*|*Compiler*|*Code Generation* menu item to pull up the Code Generation dialog as seen in Figure 2-3. Change the memory model to Large.

Figure 2-3. Code Generation dialog

4. Select the *Options*|*Compiler*|*Entry/Exit* menu item and turn on "Test stack overflow" as seen in Figure 2-4.

Figure 2-4. Entry/Exit Code Generation dialog

5. Select the *Options*|*Compiler*|*Messages*|*Display* menu item to bring up the Compiler Messages dialog as seen in Figure 2-5. Select All to display all the warning messages.

Figure 2-5. Compiler Messages dialog

6. Select the *Options*|*Save* menu item to save all the options we've used so far.
7. Select the *Project*|*Open* menu item to select a project file. In this case, our project file is called *HELLO.PRJ*. The screen should look like Figure 2-6 when you're done.

Figure 2-6. Open Project File dialog

8. Press the INSERT key to add a file to the project. The file we want to add is *HELLO.C* as seen in Figure 2-7.

Figure 2-7. Add to Project List dialog

9. Press ESC to get out of the add-file cycle.
10. Press UP-ARROW to go up one line. The line with *HELLO.C* should now be highlighted as seen in Figure 2-8.

Figure 2-8. Hello project

11. Press ENTER to edit this file.
12. Enter Example 2-2.

Example 2-2. *hello/hello.c*

```
[File: hello/hello.c]
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return (0);
}
```

The results should look like Figure 2-9.

Figure 2-9. Finished project

13. Select the *Run*|*Run* menu item to execute the program.
14. After the program runs, control returns to the IDE. This control change means that you can't see what your program output. To see the results of the program you must switch to the user screen by selecting the *Window*|*User* menu item.

    To return to the IDE, press any key. Figure 2-10 shows the output of the program.

Figure 2-10. User screen

15. When you are finished, you can save your program by selecting the *File*|*Save* menu item.
16. To exit the IDE, select the *File*|*Quit* menu item.

2.4.2.2 Borland C++

1. Create a directory called \HELLO to hold the files for our Hello World program. You can create a directory using the Windows' File Manager program or by typing the following command at the MS-DOS prompt:

   ```
   C:> mkdir \HELLO
   ```

2. From Windows, double-click on the "Borland C++" icon to start the IDE. Select the *Window*|*Close* all menu item to clean out any old junk. The program begins execution and displays a blank workspace as seen in Figure 2-11.

Figure 2-11. Borland C++ initial screen

3. Select the *Project*|*New Project* menu item to create a project for our program. Fill in the "Project Path and Name:" blank with **c:\hello\hello.ide**. For the *Target Type*, select *EasyWin(.exe)*. The *Target Model* is set to Large. The results are shown in Figure 2-12.

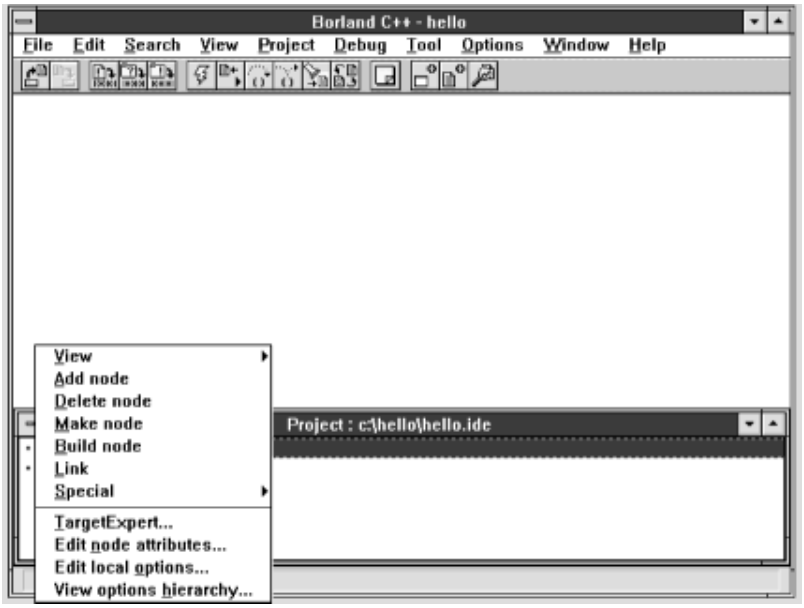
Figure 2-12. New Target dialog

4. Click on the *Advanced* button to bring up the Advanced Options dialog. Clear the *.rc* and *.def* items and set the *.c Node* items as shown in Figure 2-13.
5. Click on *OK* to return to the New Target dialog. Click on *OK* again to return to the main window.

Figure 2-13. Advanced Options dialog

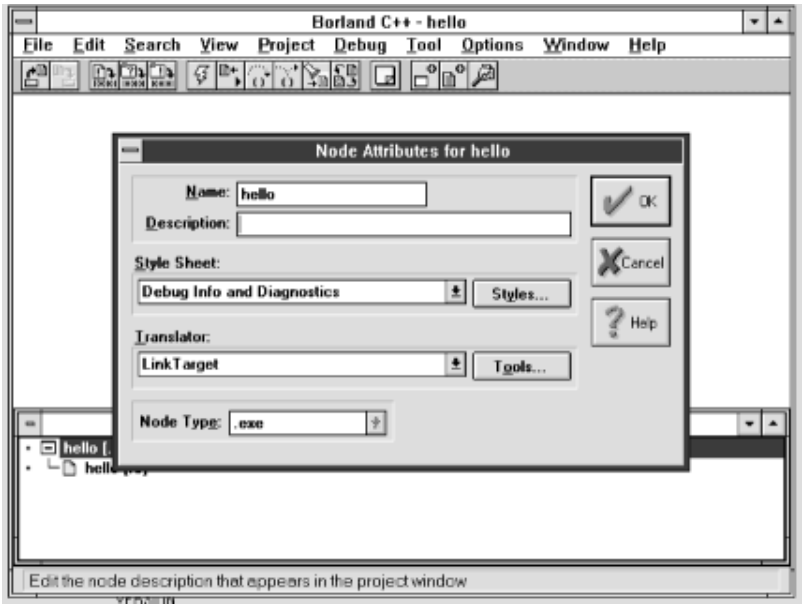
6. Press ALT-F10 to bring up the node submenu shown in Figure 2-14.

Figure 2-14. Target Options submenu



7. Select the *Edit node attributes* menu item to bring up the dialog shown in Figure 2-15. In the *Style Sheet* blank, select the item *Debug Info and Diagnostics*. Click on *OK* to return to the main window.

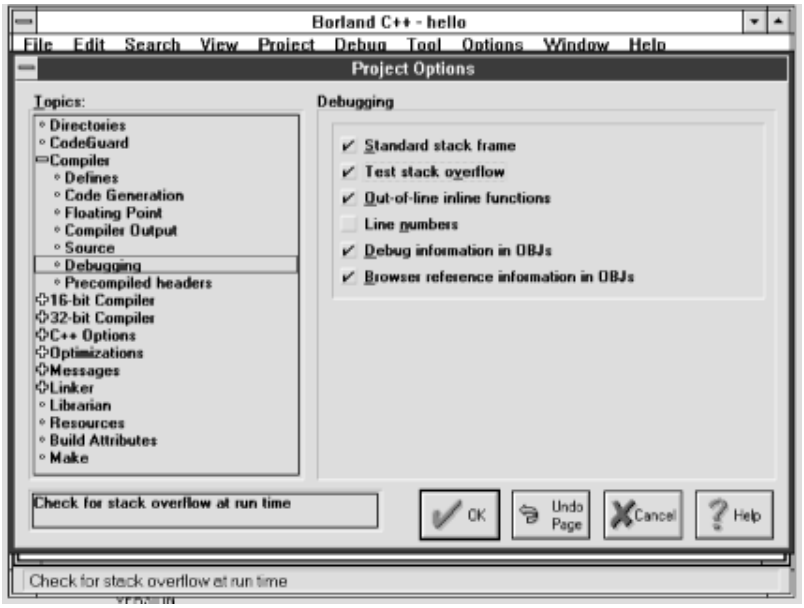
Figure 2-15. Node Attributes dialog



8. Go to the Project Options dialog by selecting the *Options*|*Project Options* menu item. Go down to the Compiler item and click on the + to expand the options.

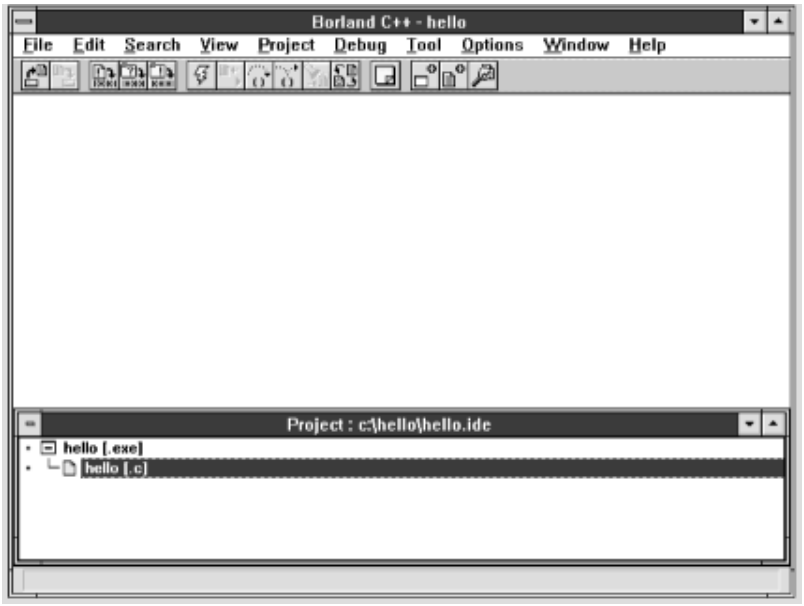
   Turn on the *Test stack overflow* option as seen in Figure 2-16. Click on OK to save these options.

Figure 2-16. Project Options dialog



9. Click on OK to return to the main window. Press DOWN-ARROW to select the *hello[.C]* item in the project as seen in Figure 2-17.

Figure 2-17. Hello project



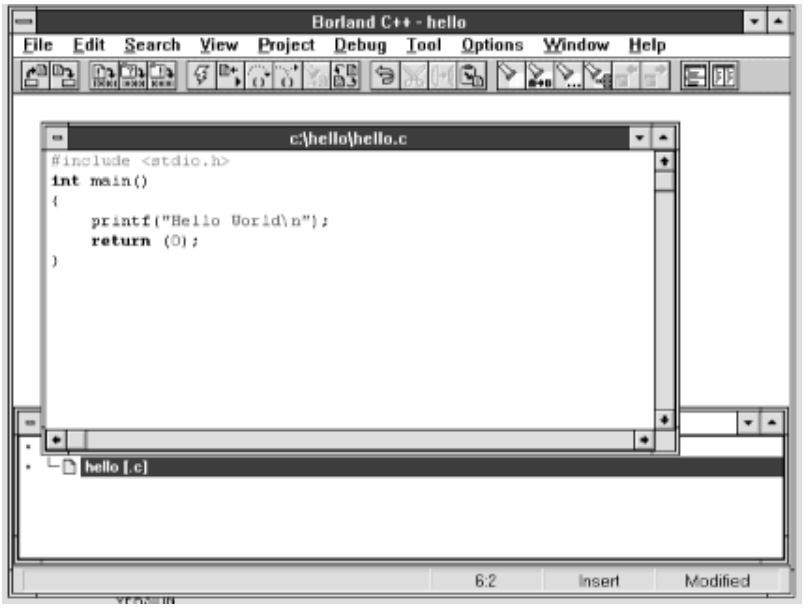
10. Press RETURN to start editing the file *hello.c*. Type in Example 2-3.

Example 2-3. *hello/hello.c*

```
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return (0);
}
```

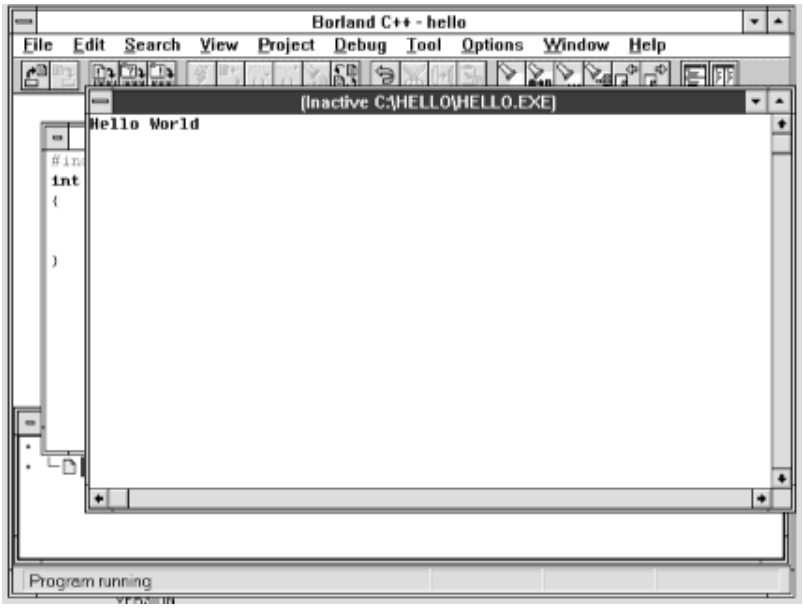
When you finish, your screen will look like Figure 2-18.

Figure 2-18. Hello World program



11. Compile and run the program by selecting the *Debug|Run* menu item. The program will run and display "Hello World" in a window as seen in Figure 2-19.

Figure 2-19. Hello World program after execution



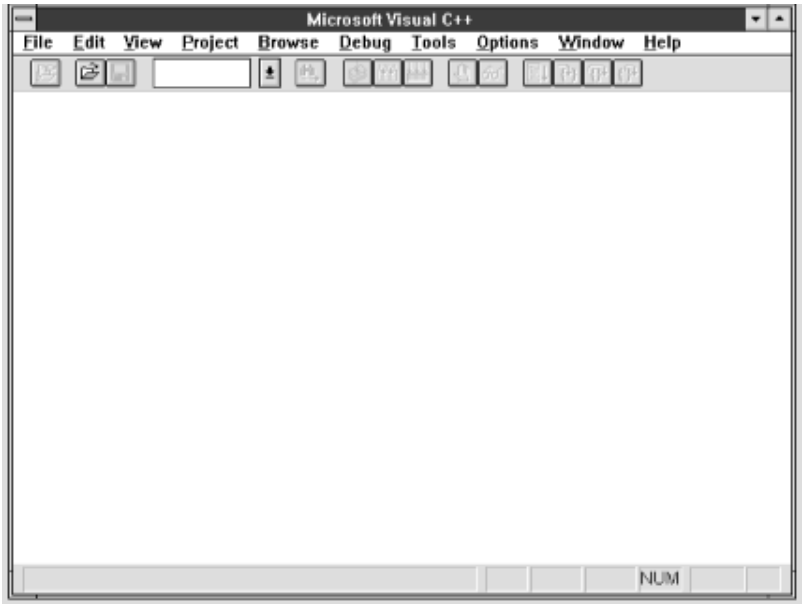
2.4.2.3 Microsoft Visual C++

1. Create a directory called *\HELLO* to hold the files for our *Hello World* program. You can create a directory using the Windows' *File Manager* program or by typing the following command at the MS-DOS prompt:

   ```
   C:> mkdir \HELLO
   ```

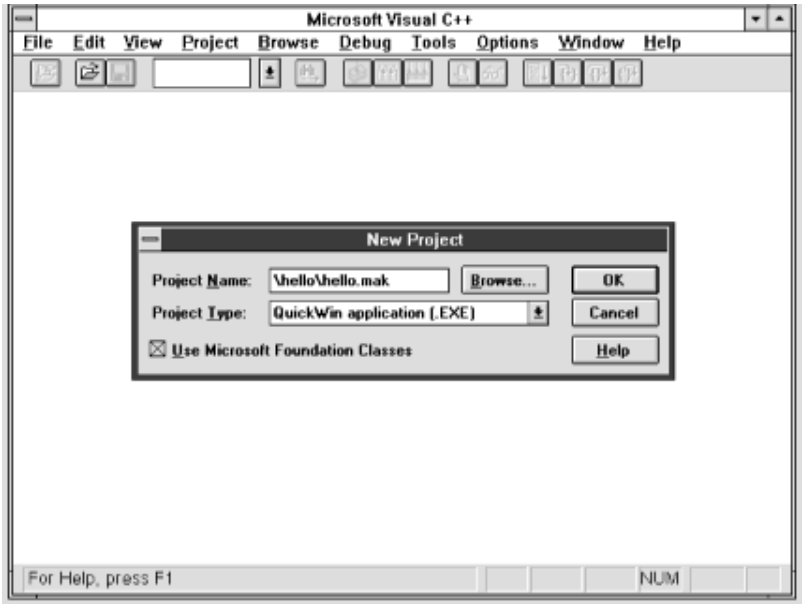
2. From Windows, double-click on the Microsoft Visual C++ to start the IDE. Clear out any old junk by selecting the *Window*|*Close All* menu item. A blank workspace will be displayed as seen in Figure 2-20.

Figure 2-20. Microsoft Visual C++ initial screen



3. Click on the *Project*|*New* menu item to bring up the New Project dialog as shown in Figure 2-21.

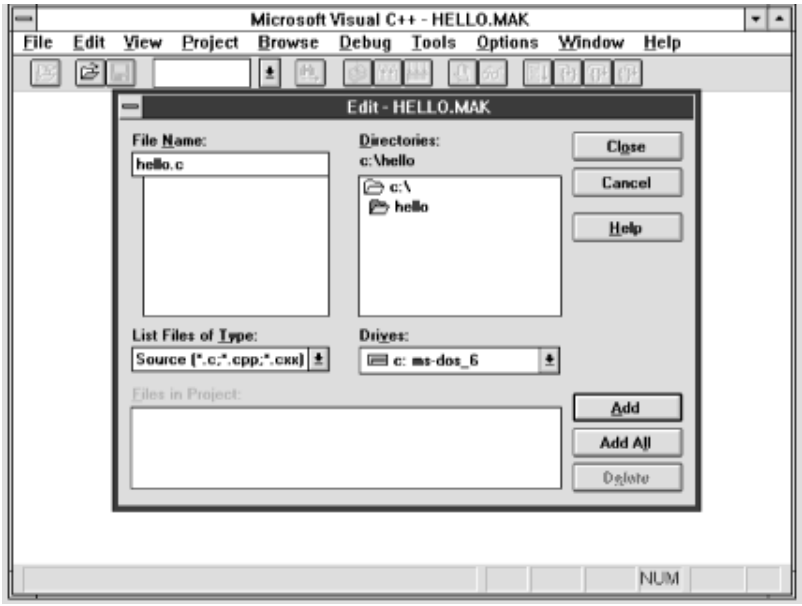
Figure 2-21. New Project dialog



Fill in the Project Name blank with "\hello\hello.mak". Change the *Project Type* to *QuickWin application (.EXE)*.

4. Visual C++ goes to the Edit dialog to allow you to name the source files in this project (see Figure 2-22). In this case, we have only file *hello.c*. Click on *Add* to put this in the project and Close to tell Visual C++ that there are no more files in the program.

Figure 2-22. Edit Project dialog



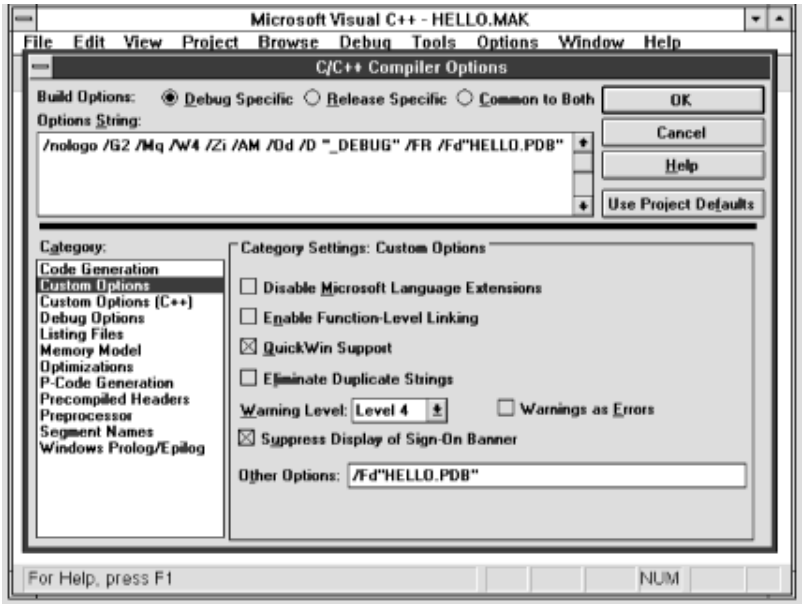
5. Select the *Options*|*Project Options* menu item to bring up the Project Options dialog as seen in Figure 2-23.

Figure 2-23. Project Options dialog

Click on the *Compiler* button to change the compiler options.

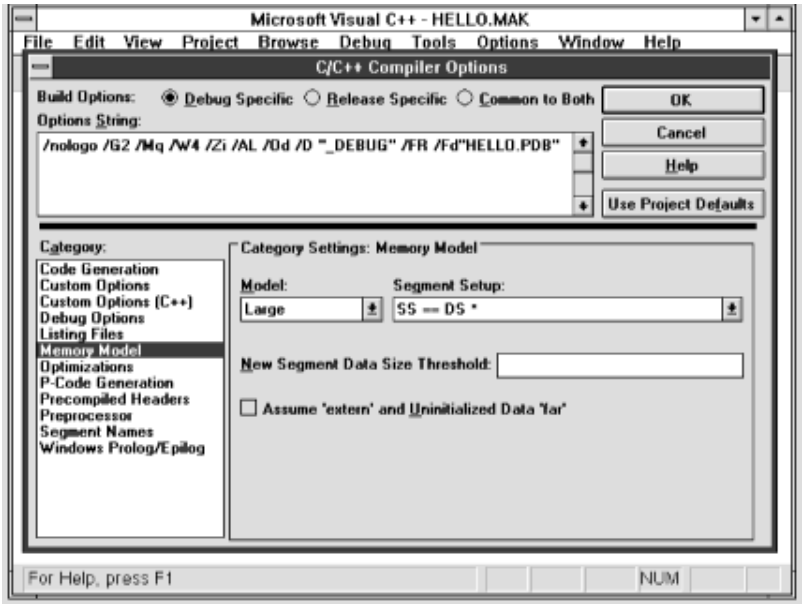
6. Go down to the *Custom Options* menu item under *Category* and change the *Warning Level* to 4 as seen in Figure 2-24.

Figure 2-24. C/C++ Compiler Options dialog



7. Select the *Memory Model* category and change the *Model* to *Large* (see Figure 2-25).

Figure 2-25. Memory Model options



8. Close the dialog by clicking on the *OK* button. You return to the Project Options dialog. Click on *OK* to dismiss this dialog as well.
9. Select the *File|New* menu item to start a new program file. Type in Example 2-4.

Example 2-4. *hello/hello.c*

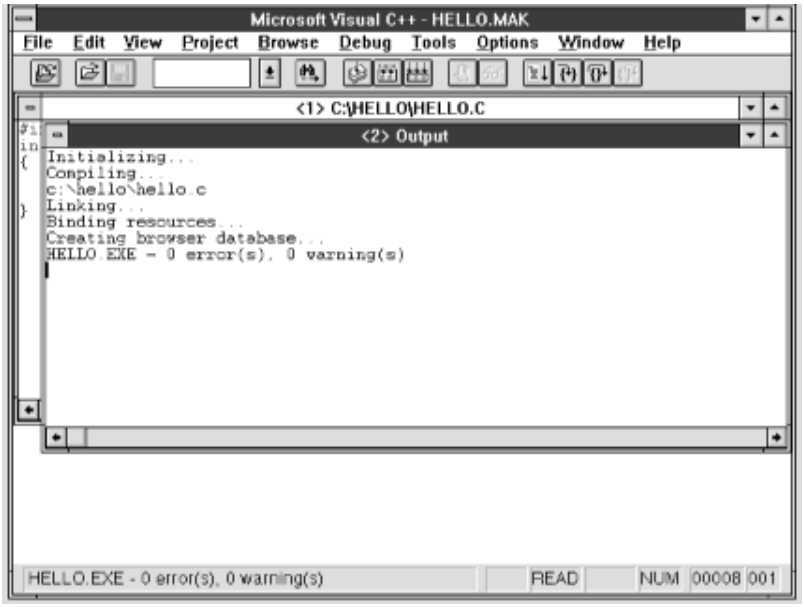
```
[File: hello/hello.c]
#include <stdio.h>
int main()
{
    printf("Hello World\n");
    return (0);
}
```

Your results should look Figure 2-26.

Figure 2-26. Microsoft Visual C++ with Hello World entered

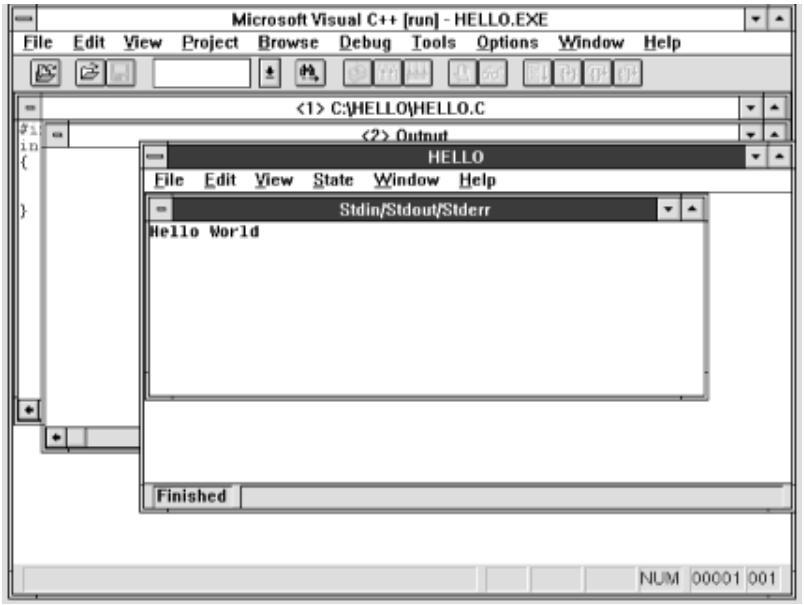
10. Use the *File|Save* As menu item to save the file under the name *hello.c*.
11. Use the *Project|Build* menu item to compile the program. The compiler will output messages as it builds. When the compiler is finished, your screen should look like Figure 2-27.

Figure 2-27. Microsoft Visual C++ project build screen



12. The program cannow be started with the*Debug*|*Go* menu item. The results appear in Figure 2-28.

Figure 2-28. Hello World results



2.5 Getting Help on UNIX

Most UNIX systems have an online documentation system called the manpages. These manpages can be read with the `man` command. (UNIX uses `man` as an abbreviation for manual.) To get information about a particular subject, use the following command:

man *subject*

For example, to find out about the classes defined in the`printf` function, you would type:

man printf

The command also has a keyword search mode:

man -k *keyword*

To determine the name of manpage with the word "output" in its title, use the command:

man -k output

2.6 Getting Help in an Integrated Development Environment

IDEs such as Turbo C++, Borland C++, and Microsoft C++ have a Help menu item. This item activates a hypertext-based help system.

2.7 IDE Cookbooks

This section contains a brief summary of the commands used to enter, compile, and execute a simple program using the three IDEs described in this chapter.

2.7.1 Turbo C++

1.	Window|Close All	Clean out any old junk.
2.	Options|Compiler|Code Generation	For simple program, use large memory model.

	Memory Model = Large	
3.	Options|Compiler|Entry/Exit Test stack overflow = On	Turn on test for a common programming error.
4.	Options|Compiler|Messages|Display Display warnings = All	Tell compiler that you want all diagnostics that it can give you.
5.	Options|Save	Save options.
6.	Project|Open Project file = ***program***.PRJ	Create a new project.
7.	Insert Add file ***program***.c	Add program file to project.
8.	ESC	Get out of "add-file" cycle.
9.	UP-ARROW	Move to ***program***.c line.
10.	RETURN	Edit program file.
11.	Type in the program	Enter text of program.
12.	Run|Run	Execute program.
13.	Window|User	Display results of the program.
14.	File|Save	Save the program.
15.	File|Quit	Exit Turbo C++ IDE.

2.7.2 Borland C++

1.	Window|Close All	Clean out any old junk.
2.	Project|New Project Project Path and Name = c:\ ***program******program***.ide Target Type = EasyWin(.exe) Target Model = Large	Create new project.
3.	Click on Advanced button Set .c Node Clear .rc and .def	Setup a simple C program.
4.	Click on OK	Return to New Target window.

5.	Click on OK	Return to main window.
6.	ALT-F10	Select node submenu.
7.	Edit|Node Attributes Style Sheet = Debug Info and Diagnostics	Turn on debugging.
8.	Click on OK button	Return to main menu.
9.	Options|Project Options Click on + under Compiler Test stack overflow = On	Turn on valuable run-time test.
10.	Click on OK button	Save options.
11.	Click on OK button	Return to main window.
12.	DOWN-ARROW	Move to *program* [.c] line.
13.	RETURN	Edit program file.
14.	Type in the program	Enter text of program.
15.	Debug|Run	Run program.

2.7.3 Microsoft Visual C++

1.	Window|Close All	Clean out any old junk.
2.	Project|New Project Name = ***program******program***.mak Project Type = QuickWin application (.EXE)	Start project. Set up project. Click on OK button. Go to Edit dialog.
3.	File name = *program* .c	Enter program name.
4.	Click on Add button	Add program to project.
5.	Click on Close button	Tell Visual C++ that there are no more files.
6.	Options|Project Options	Get to Project Options dialog.
7.	Click on Compiler button	Go to C|C++ Compiler Options dialog.
8.	Select Custom Options category Warning Level = 4	Turn on all warnings.
9.	Select the Memory Model category	For simple program, use large-memory model.

	Memory Model = Large	
10.	Click on OK button	Return to Project Options dialog.
11.	Click on OK button	Return to main window.
12.	File|New	Open program file.
13.	Type in the program	Edit program file.
14.	File|Save As — File name = *program*.c	Save file.
15.	Project|Build	Compile program.
16.	Debug|Go	Execute program.

These instructions are for version 4.0 of Microsoft Visual C++. Microsoft frequently changes the user interface from version to version, so these instructions may require some slight modification.

2.8 Programming Exercises

Exercise 2-1: On your computer, type in the hello program and execute it.

Exercise 2-2: Take several programming examples from any source, enter them into the computer, and run them.