1 Getting Started with Borland™ C++Builder™ Compiler

Objectives

- To be able to install and configure the Borland C++Builder Compiler.
- To be able to use a text editor to create C/C++ programs.
- To be able to use the Borland C++Builder Compiler to compile C/C++ source files.
- To be able to execute programs in MS-DOS.
- To be able to compile and execute programs with multiple source files.

Outline

1.1 Introduction
1.2 Installation
1.3 Configuration
1.4 Creating a C/C++ Program
1.5 Compiling and Executing a C/C++ Program
1.6 Compiling Programs with Multiple Source Files

1.1 Introduction

Welcome to the Borland™ C++Builder™ Compiler. In this appendix you will learn how to install the software included on the CD accompanying *C How to Program: Third Edition* and how to create, compile and execute C and C++ programs using the C++ compiler from Borland—*C++Builder Compiler*. When you complete this appendix, you will be able to use the C++Builder Compiler to begin building applications.

The C++Builder Compiler compiles both C and C++ programs. C files are usually have a **`.c`** file extension and C++ files typically have a **`.cpp`** file extension. The C++Builder Compiler uses the file extension to determine whether the file it is compiling is a C or C++ program. This appendix assumes that you are compiling a C program. However, the same steps apply to C++ programs.

Before proceeding with this appendix, you should be familiar with the topics in Chapter 1, "Computing Concepts," Chapter 2, "Introduction to C Programming," and Chapter 3, "Structured Programming Development," of *C How to Program: Third Edition*. A few of the examples in this chapter make reference to *functions*. For these examples, you should be familiar with the material through Section 5.6 of Chapter 5, "Functions," in *C How to Program: Third Edition.* We hope you enjoy learning about the C++Builder Compiler.

1.2 Installation

The Borland C++Builder Compiler is included on the CD accompanying *C How to Program: Third Edition*. To install this compiler perform the following steps:

1. Go to the **`CBuilder5\CommandLineTools\`** directory on your CD drive and double click **`FreeCommandLineTools.exe`**. [*Note:* The C++Builder Compiler is also available for free download at **`www.borland.com/bcppbuilder/freecompiler/`**.] This displays the **Borland C++ Compiler 5.5 - Welcome dialog**.
2. Click **Next >** to load the licensing information.
3. After you have read the licensing agreement, click **I Agree** to load the **Installation Folder** dialog.
4. Choose a directory into which you will install the compiler. It is recommended that you use the default **Installation Folder** (Fig. 1.1), **`c:\Borland\bcc55`** (the rest of this appendix assumes that the default installation folder is used).
5. Click **Finish** to complete the installation.

©2000 Deitel & Associates, Inc. and
Prentice Hall. All rights reserved.

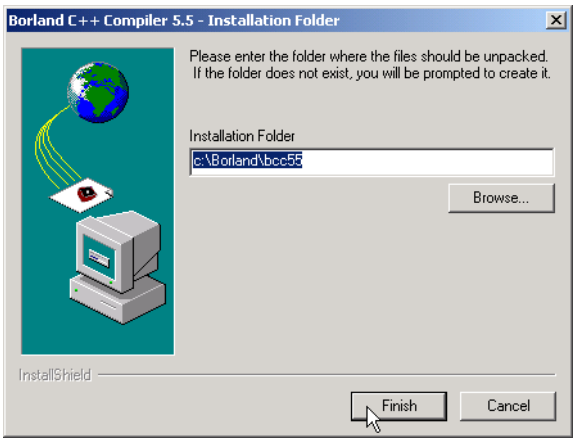

Fig. 1.1 **Installation Folder** dialog.

1.3 Configuration

The compiler must be configured after it is installed. To specify the location of the libraries for both the compiler and the linker, two **.cfg** files must be created. These files (**bcc32.cfg** and **ilink32.cfg**) are available for free download from the Deitel & Associates, Inc. Web site.

Download and save these files into the directory **c:\Borland\bcc55\Bin** (assuming the default installation folder was used) to complete the configuration.

If you are unable to download the files, you need to create the **.cfg** files. To do so, open a text editor such as Microsoft Notepad. To load Notepad click the **Start** button and select **Programs** followed by **Accessories** and **Notepad**. Type into the editor the following lines:

```
-I"c:\Borland\bcc55\include"
-L"c:\Borland\bcc55\lib"
```

Then, click **Save** from the **File** menu and save the file in the **c:\Borland\bcc55\Bin** directory and name it **bcc32.cfg**. Some text editors, including Notepad, will automatically append a **.txt** extension to the end of all files they create. To avoid this problem, place the file name in double quotes. The text in the **File name** box should read **"bcc32.cfg"** (Fig. 1.2).

©2000 Deitel & Associates, Inc.—All rights reserved.
©2000 Prentice Hall—All rights reserved.

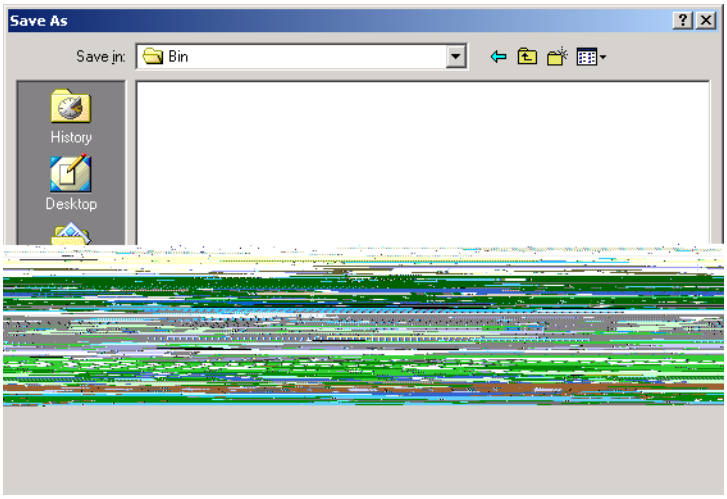

Fig. 1.2 Saving a file as a **.cfg**.

Open a new document in your text editor and type:

```
-L"c:\Borland\bcc55\lib"
```

Then save the file in the **c:\Borland\bcc55\Bin** directory and name it **ilink32.cfg** using the same process outlined for the previous file.

The C++Builder Compiler is now configured. If you are still having problems with installation or configuration read the file **c:\Borland\bcc55\readme.txt** or consult **www.borland.com\techpubs\bccpbuilder** for more information.

1.4 Creating a C/C++ Program

Before creating a C/C++ program, create a directory to store your files. We created a directory named **c:\cFiles**, you of course can choose a different name.

You are now ready to create a program. Open a text editor and type in a program, such as the following: [*Note*: We have included line numbers to improve readability of our example, however they are not part of the program and should not be included.]

```
1  #include <stdio.h>
2
3  int main()
4  {
5     printf( "Welcome to C!\n" );
6     return 0;
7  }
```

©2000 Deitel & Associates, Inc. and
Prentice Hall. All rights reserved.

(i.e., the grammatical rules) of C/C++. For example, if the quotes are eliminated from around the string **"Hello World!\n"** (an illegal change) then upon compilation the syntax errors shown in Fig. 1.5 are generated.

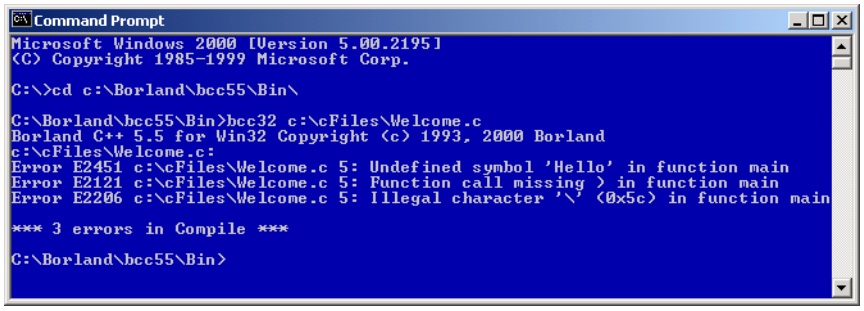

Fig. 1.5 Unsuccessful compilation.

An error message consists of an error code (i.e., **E2451**), the file in which the error occurred (i.e., **c:\cFiles\Welcome.c**), the line number (i.e., **5**) and a brief description of the error (i.e., **Undefined symbol 'Hello' in function main()**). If you received a syntax error reopen your program in the text editor, go to the line specified by the error message and correct it. Then, save and recompile the program.

If you received a message which was a warning rather than an error, then your compile was successful but there may be a problem with your code which will prevent it from working as you intended. It is recommended that you resolve all warnings before executing your program.

C/C++ compilers often list more errors than actually occur in the program. For example, a compiler may locate a syntax error in your program (e.g., a missing semicolon). That error may cause the compiler to report other errors in the program when, in fact, there may not be any other errors.

Testing and Debugging Tip 1.1

When a syntax error on a particular line is reported by the compiler, check that line for the syntax error. If the error is not on that line, check preceding lines of code for the cause of the syntax error.

Testing and Debugging Tip 1.2

After fixing one error, recompile your program. You may observe that the number of overall errors perceived by the compiler is significantly reduced.

Once the program compiles without errors, you can execute it by typing the file name, without a file extension. Type **Welcome** and your program will execute (Fig. 1.6). Whenever you are finished with the C++Builder Compiler, type **exit**, to close the **Command Prompt**.

To create additional C/C++ files, repeat the steps in the previous two sections.

©2000 Deitel & Associates, Inc. and
Prentice Hall. All rights reserved.

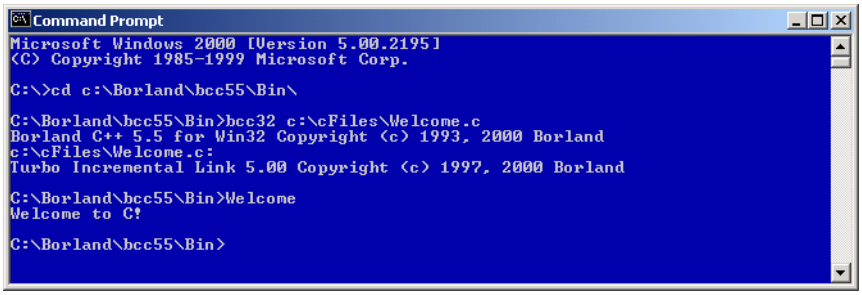

Fig. 1.6 Execution of **Welcome.c**.

1.6 Compiling Programs with Multiple Source Files

More complex programs often consist of multiple C or C++ source files. We introduce this concept, called *multiple source files*, in Chapter 14 of *C How to Program: Third Edition*. This section explains how to compile a program with multiple source files using the Borland C++Builder Compiler.

Compiling a program, which has two or more source files, requires listing them both on the command line. Using the methods described in Section 1.4 of this appendix, create the following two C program files named **Welcome2.c** and **Hello.c**, respectively.

```
1  /* Welcome2.c */
2
3  #include <stdio.h>
4
5  void myFunction( void );
6
7  int main()
8  {
9     printf( "Welcome to C!\n" );
10    myFunction();
11    return 0;
12 }
```

©2000 Deitel & Associates, Inc.—All rights reserved.
©2000 Prentice Hall—All rights reserved.

```
1  /* Hello.c */
2
3  #include <stdio.h>
4
5  void myFunction( void )
6  {
7     printf( "Hello from Hello.c!\n" );
8  }
```

To compile the program, using these two source files, type:

```
bcc32 c:\cFiles\Welcome2.c c:\cFiles\Hello.c
```

The executable file which is created has the same name as the first file listed in compilation. In our example, the executable is named **`Welcome2.exe`**. Therefore, to execute your program (Fig. 1.7), type **`Welcome2`**.

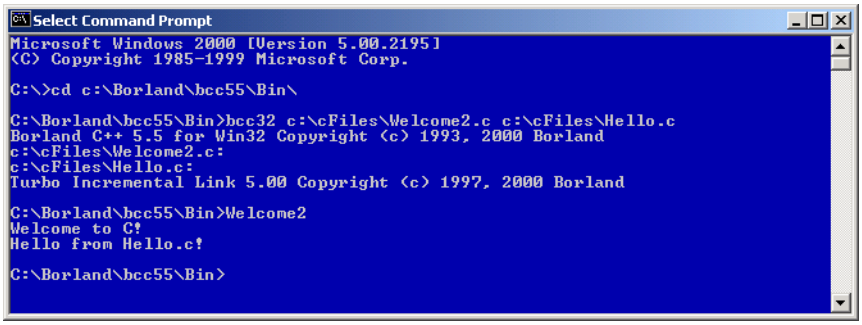

Fig. 1.7 Compiling and executing a program with two source files.

©2000 Deitel & Associates, Inc. and
Prentice Hall. All rights reserved.