# Appendix B

# Executing Turbo C++

## B.1 Introduction

All programs in this book were developed and run under Turbo C++ compiler Version 3.0, in an MS-DOS environment on an IBM PC compatible computer. We shall discuss briefly, in this Appendix, the creation and execution of C++ programs under Turbo C++ system.

## B.2 Creation and Execution of Programs

Executing a computer program written in any high-level language involves several steps, as listed below:

1. Develop the program (source code).
2. Select a suitable file name under which you would like to store the program.
3. Create the program in the computer and save it under the filename you have decided. This file is known as *source code file*.
4. Compile the source code. The file containing the translated code is called *object code file*. If there are any errors, debug them and compile the program again.
5. Link the object code with other library code that are required for execution. The resulting code is called the *executable code*. If there are errors in linking, correct them compile the program again.
6. Run the executable code and obtain the results, if there are no errors.
7. Debug the program, if errors are found in the output.
8. Go to Step 4 and repeat the process again.

These steps are illustrated in Fig. B.1. The exact steps depend upon the program environment and the compiler used. But, they will resemble the steps described above.

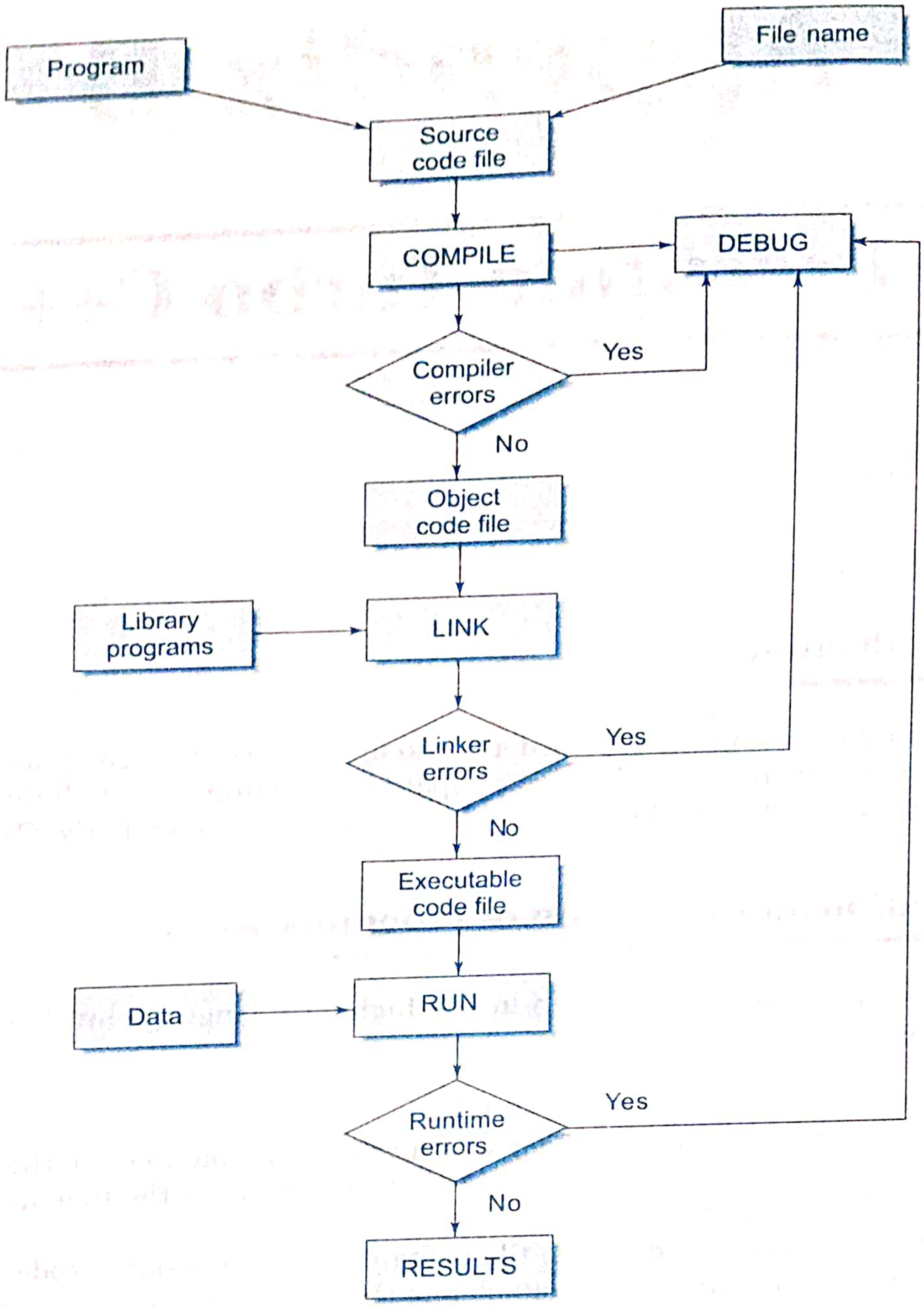


**Fig. B.1** ⇔ *Program development and execution*

Turbo C++ and Borland C++ are the two most popular C++ compilers. They provide ideal platforms for learning and developing C++ programs. In general, both Turbo C++ and Borland C++ work the same way, except some additional features supported by Borland C++ which are outside the scope our discussions. Therefore, whatever we discuss here about Turbo C++ applies to Borland C++ as well.

## B.3 Turbo C++

Turbo C++ provides a powerful environment called ***Integrated Development Environment* (IDE)** for creating and executing a program. The IDE is completely menu-driven and allows the user to create, edit, compile and run programs using what are known as ***dialogue boxes*.** These operations are controlled by single keystrokes and easy-to-use menus.

We first use the editor to create the source code file, then compile, link and finally run it. Turbo C++ provides error messages, in case errors are detected. We have to correct the errors and compile the program again.

## B.4 IDE Screen

It is important to be familiar with the details of the IDE screen that will be extensively used in the program development and execution. When we invoke the Turbo C++, the IDE screen will be displayed as shown in Fig. B.2. As seen from the figure, this screen contains four parts:

- Main menu (top line)
- Editor window
- Message window
- Status line (bottom line)

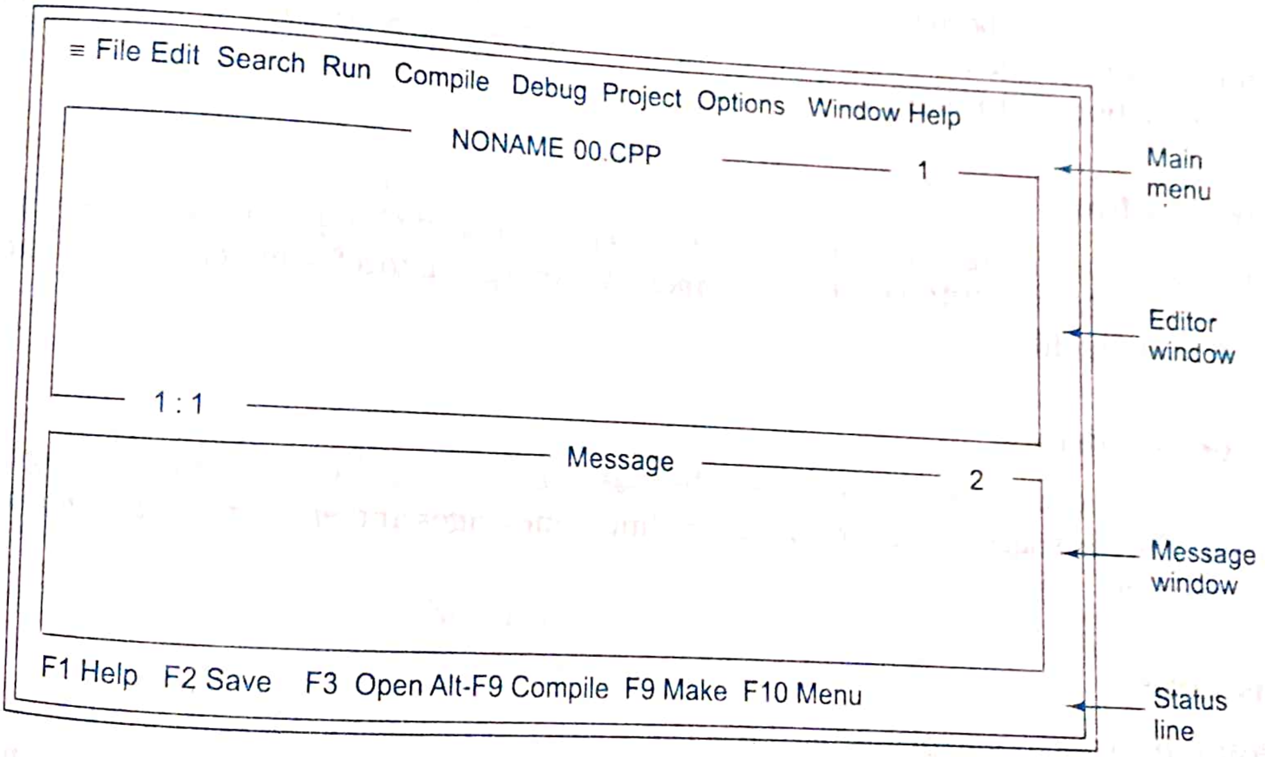

≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME 00.CPP
1
1 : 1
Message
2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
Main menu
Editor window
Message window
Status line

**Fig. B.2** ⇔ *IDE opening screen*

### Main Menu

The *main menu* lists a number of items that are required for the program development and execution. They are summarized in Table B.1.

**Table B.1** *Main menu items*

| *Item* | *Options* |
|---|---|
| – | Displays the version number, clears or restores the screen, and execute various utility programmes supplied with Turbo C++ |
| File | Loads and saves files, handless directories invokes DOS, and exists Turbo C+ |
| Edit | Performs various editing functions |
| Search | Performs various text searches and replacements |
| Run | Complies, links and runs the program currently loaded in the environment |
| Compile | Compiles the program currently in the environment |
| Debug | Sets various debugger options, including setting break points |
| Projects | Manages multifile projects |
| Options | Sets various compiler, linker, and environmental options |
| Window | Controls the way various windows are displayed |
| Help | Activates the context–sensitive Help system |

The *main menu* can be activated by pressing the F10 key. When we select an item on the main menu, a *pull-down menu*, containing various options, is displayed. This allows us to select an action that relates to the main menu item.

### Editor Window

The *editor window* is the place for creating the source code of C++ programs. This window is named NONAME00.CPP. This is the temporary name given to a file which can be changed while we save the file.

### Message Window

The other window on the screen is called the *message window* where various messages are displayed. The messages may be compiler and linker messages and error messages generated by the compiler.

### Status Line

The *status line* which is displayed at the bottom of the screen gives the status of the current activity on the screen. For example, when we are working with FILE option of main menu, the status line displays the following:

F1 Help | Locate and open a file

## B.5 Invoking Turbo C++

Assuming that you have installed the Turbo C++ compiler correctly, go to the directory in which you want to work. Then enter TC at the DOS system prompt:

```
C:>TC
```

and press RETURN. This will place you into the IDE screen as shown in Fig. B.2. Now, you are ready to create your program.

## B.6 Creating Source Code File

Once you are in the IDE screen, it is simple to create and save a program. The F10 key will take you to main menu and then move the cursor to *File*. This will display the file dialogue window containing various options for file operations as shown in Fig. B.3. The options include, among others, opening an existing file, creating a new file and saving the new file.

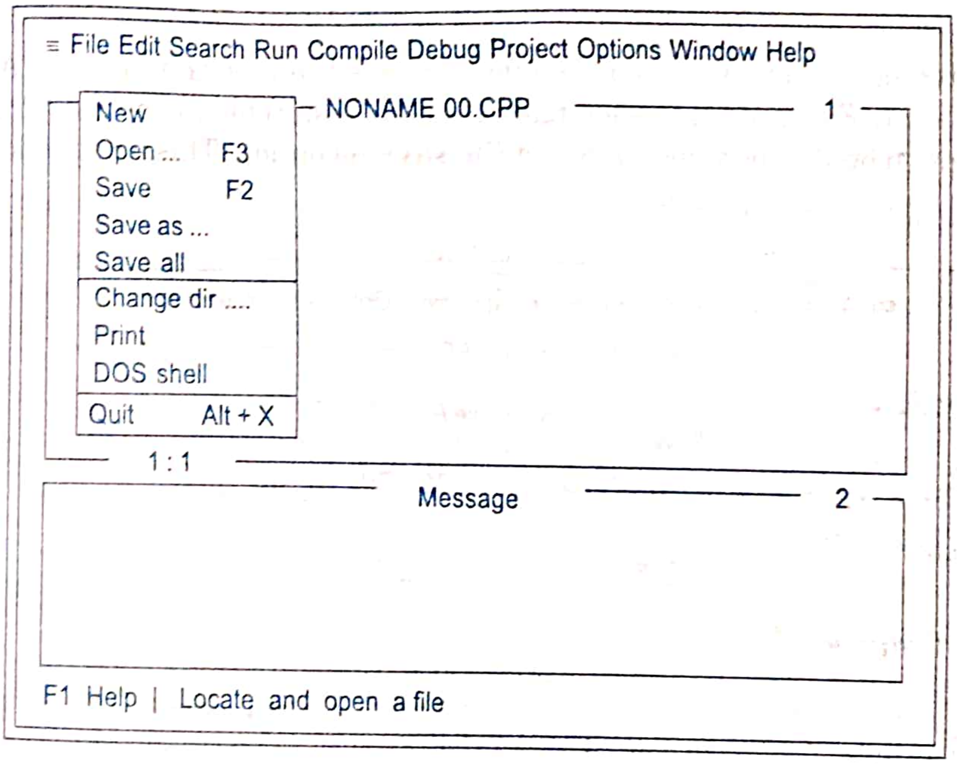
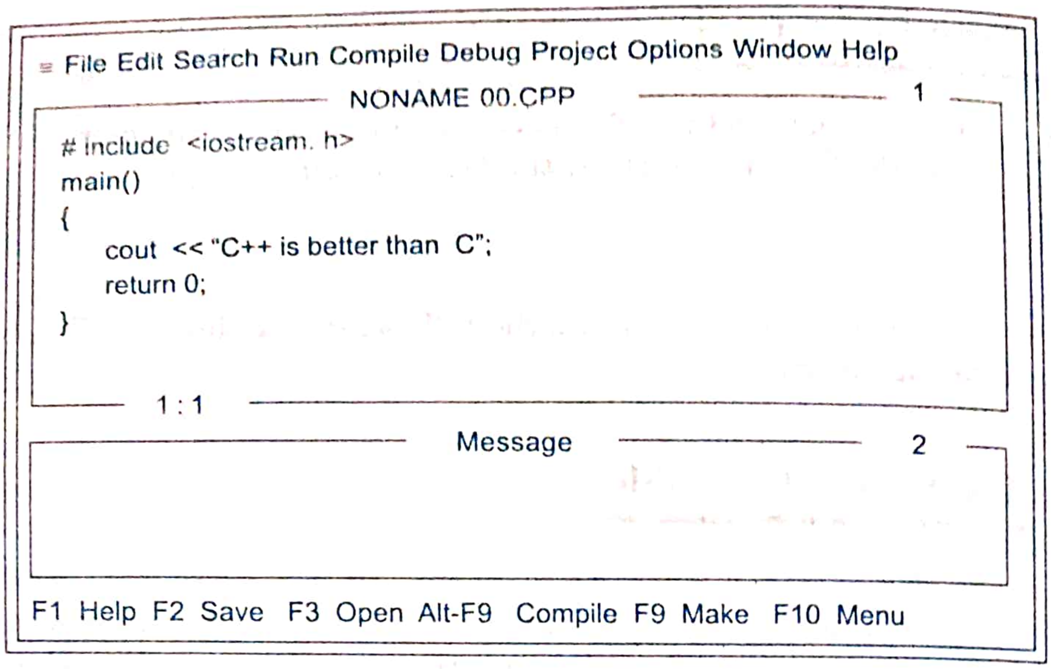


**Fig. B.3** ⇔ *File dialogue window*

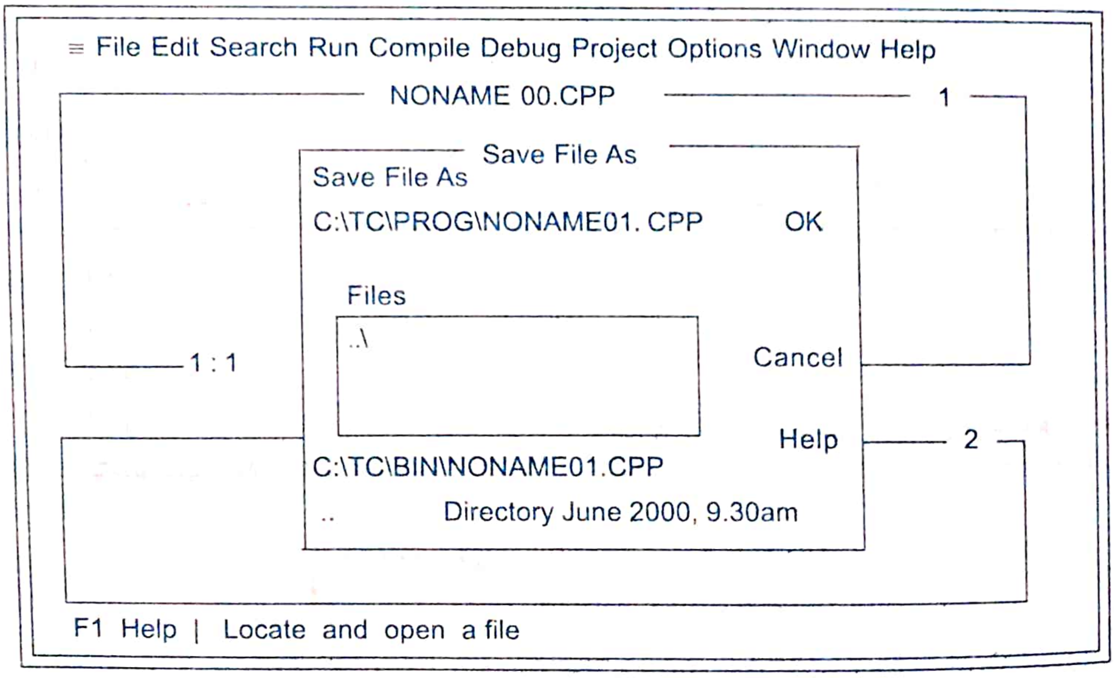
Since you want to create a new file, move the cursor to **New** option. This opens up a blank window called *editing window* and places the cursor inside this window. Now the system is ready to receive the program statements as shown in Fig. B.4.


≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME 00.CPP
1
1 : 1
Message
2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
# include  <iostream. h>
main()
{
    cout  << "C++ is better than  C";
    return 0;
}
```

Fig. B.4 ⇔ *Editor screen with statements*

Once the typing is completed, you are ready to save the program in a file. At this time, you must go to *the File dialogue* menu again to select a suitable file option. Press F10 and select **File** option on the main menu. Select the **save as** option. This brings the save editor file window as shown in Fig. B.5.


≡ File Edit Search Run Compile Debug Project Options Window Help
NONAME 00.CPP
1
Save File As
OK
Files
1 : 1
Cancel
Help
2
F1 Help | Locate and open a file

Save File As

C:\TC\PROG\NONAME01. CPP

..\

C:\TC\BIN\NONAME01.CPP

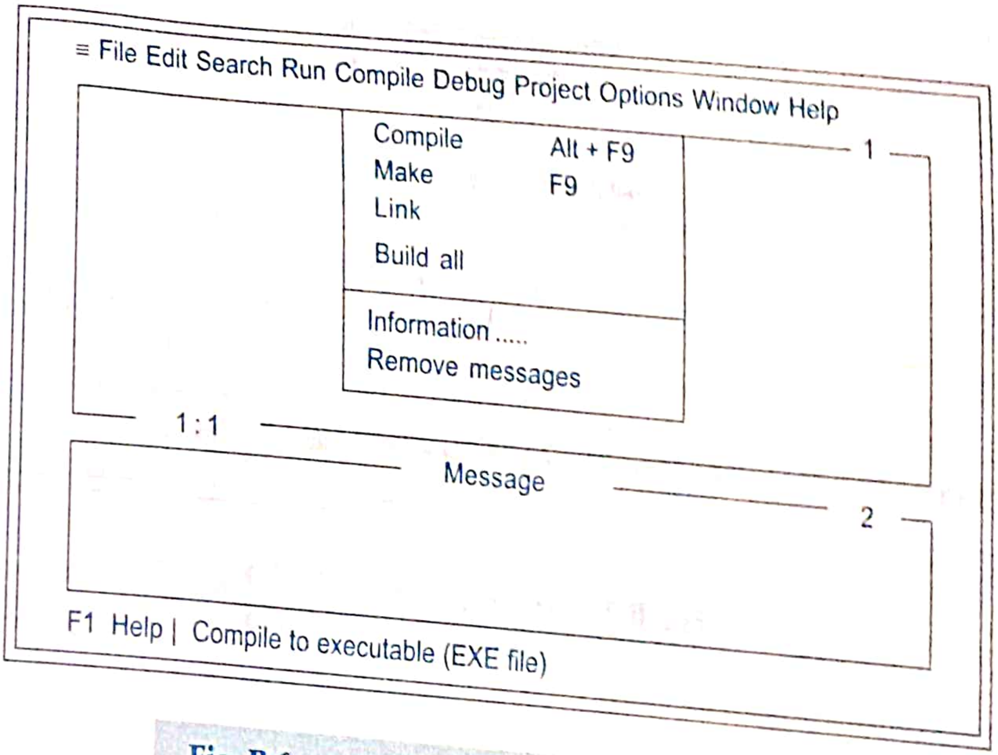
..        Directory June 2000, 9.30am

Fig. B.5 ⇔ *Save editor file window*

Now, you may change the file name NONAMEOO.CPP (shown in the editor file window) to the name you have selected. Make sure that your name has the extension **.CPP** to indicate to the compiler that your program is a C++ one and not C. Let's Assume that you have selected **test.cpp** as name. Press RETURN key and the program is saved in the file **test.cpp**.

## B.7 Compiling the Program

When you select the **compile** option on the main menu, the *compile dialogue* window is displayed as shown in Fig. B.6. The *compile* to OBJ option allows you to compile the current file in the editor to an object file. In the present case, **test.obj** file is created, if there are no errors in your program. If there are any errors, appropriate error messages are displayed in the message window.



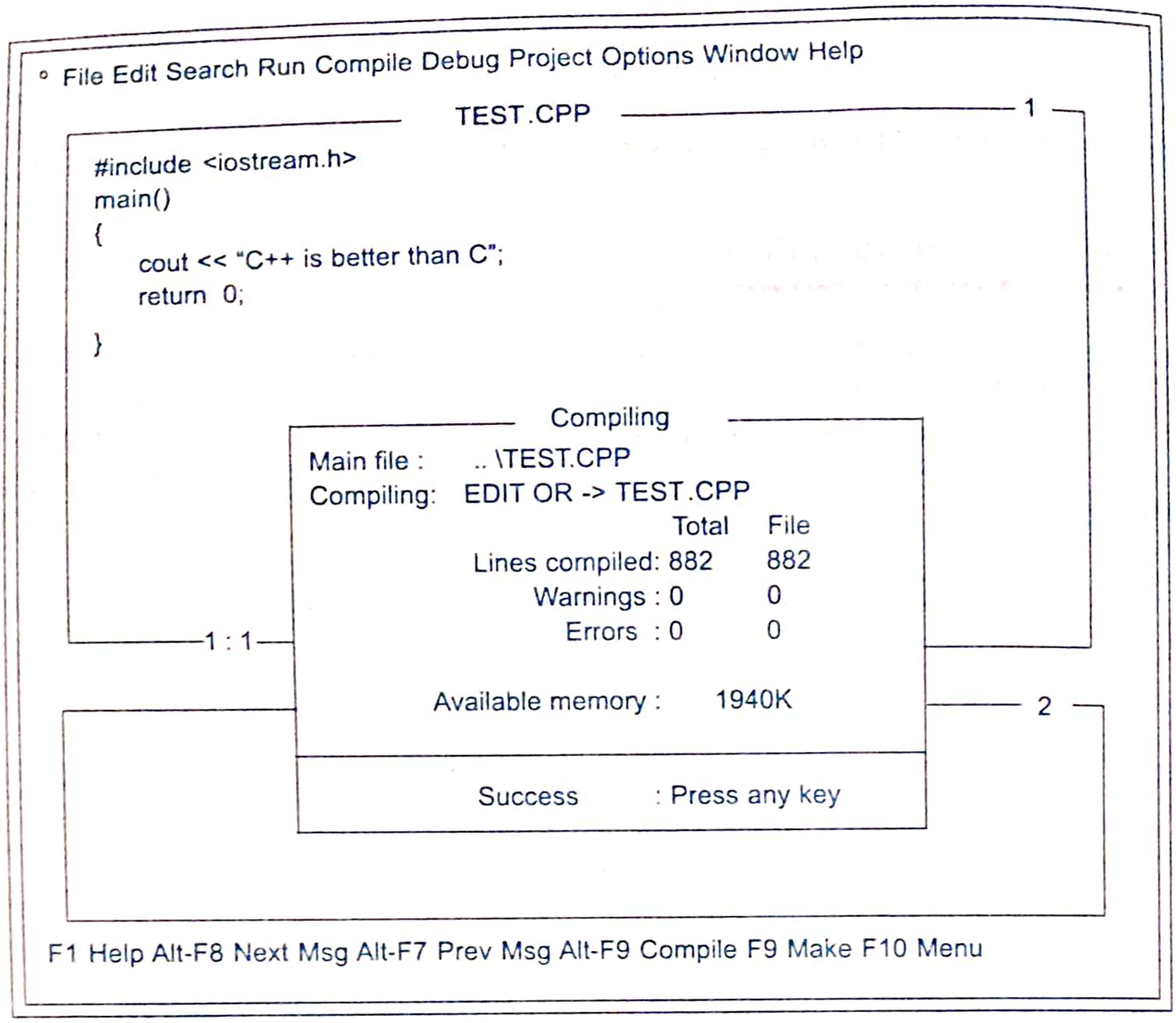
F1 Help | Compile to executable (EXE file)

Fig. B.6 ⇔ *Compile dialogue menu*

During compilation, a window called *compilation window* will appear on the screen as shown in Fig. B.7. If there are no errors during compilation, this window will display **"Success: Press any key"** message. The entries for warning and errors will be 0.

## B.8 Linking

To link the object file **test.obj** with the library functions, select the **LINK EXE** file option from the compile menu (See Fig. B.6). The **test.obj** will be linked and a third file named **test.exe** is created.



File Edit Search Run Compile Debug Project Options Window Help
TEST.CPP
1
1 : 1
2
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
#include <iostream.h>
main()
{
     cout << "C++ is better than C";
     return  0;
}
```

```
                         Compiling
Main file :      .. \TEST.CPP
Compiling:    EDIT OR -> TEST.CPP
                                   Total      File
              Lines compiled: 882        882
                  Warnings : 0           0
                    Errors  : 0          0

          Available memory :      1940K

              Success        : Press any key
```

Fig. B.7 ⇔ *Compilation window*

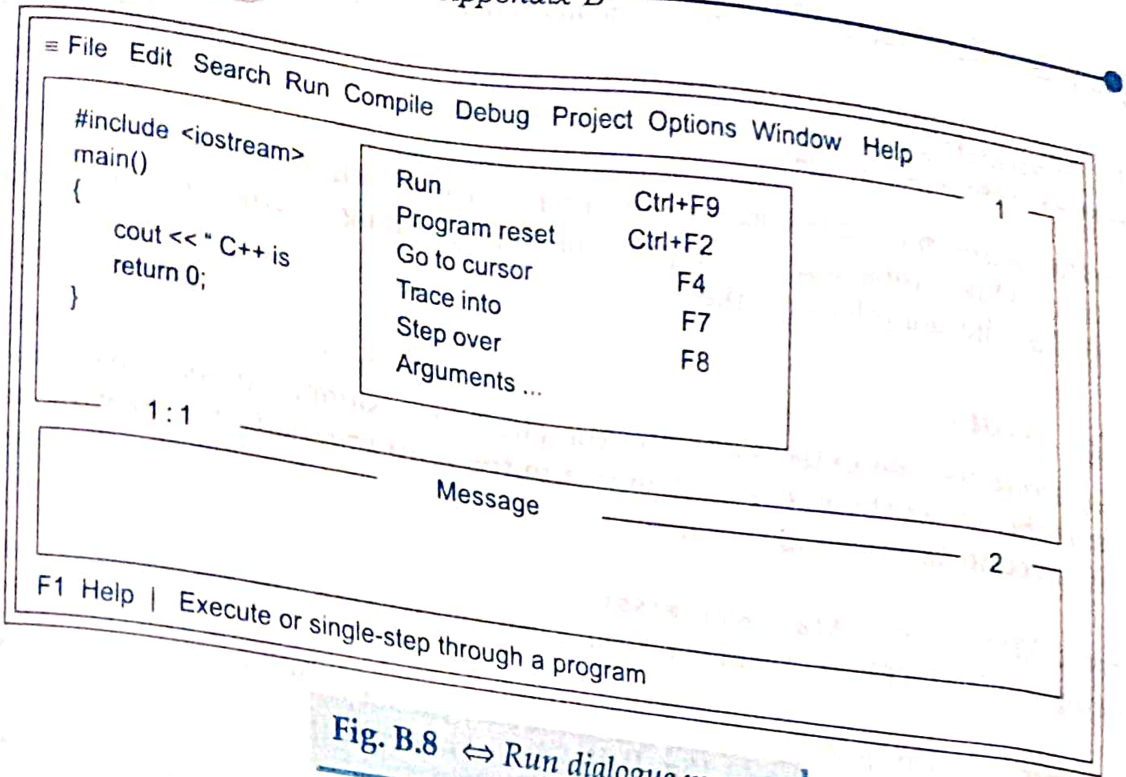
## B.9 Running the Program

You have reached successfully the final stage of your excitement. Now, select the **Run** from the main menu and again **Run** from the run *dialogue window* (See Fig. B.8). You will see the screen flicker briefly. Surprisingly, no output is displayed. Where has the output gone? It has gone to a place known as *user screen*.

In order to see the user screen, select **window** from the main menu and then select *user screen* from the window dialogue menu (See Fig.B.9). The IDE screen will disappear and the user screen is displayed containing output of the program **test.cpp** as follows:

```
C > TC
```

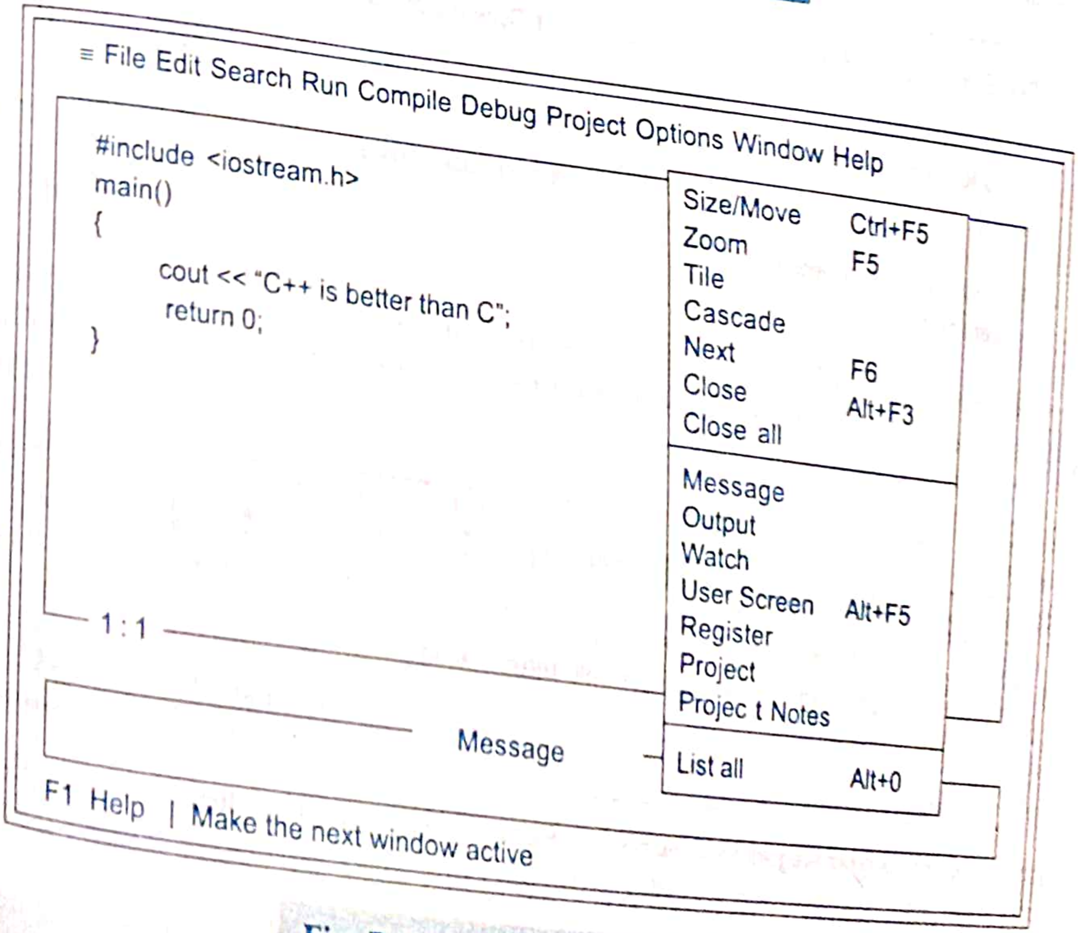
Note that, at this point, you are outside the IDE. To return to IDE, press RETURN key.



```
#include <iostream>
main()
{
    cout << " C++ is
    return 0;
}
```

F1 Help | Execute or single-step through a program

Fig. B.8 ⇔ *Run dialogue menu*



```
#include <iostream.h>
main()
{
        cout << "C++ is better than C";
        return 0;
}
```

F1 Help | Make the next window active

Fig. B.9 ⇔ *Window dialogue menu*

## B.10 Managing Errors

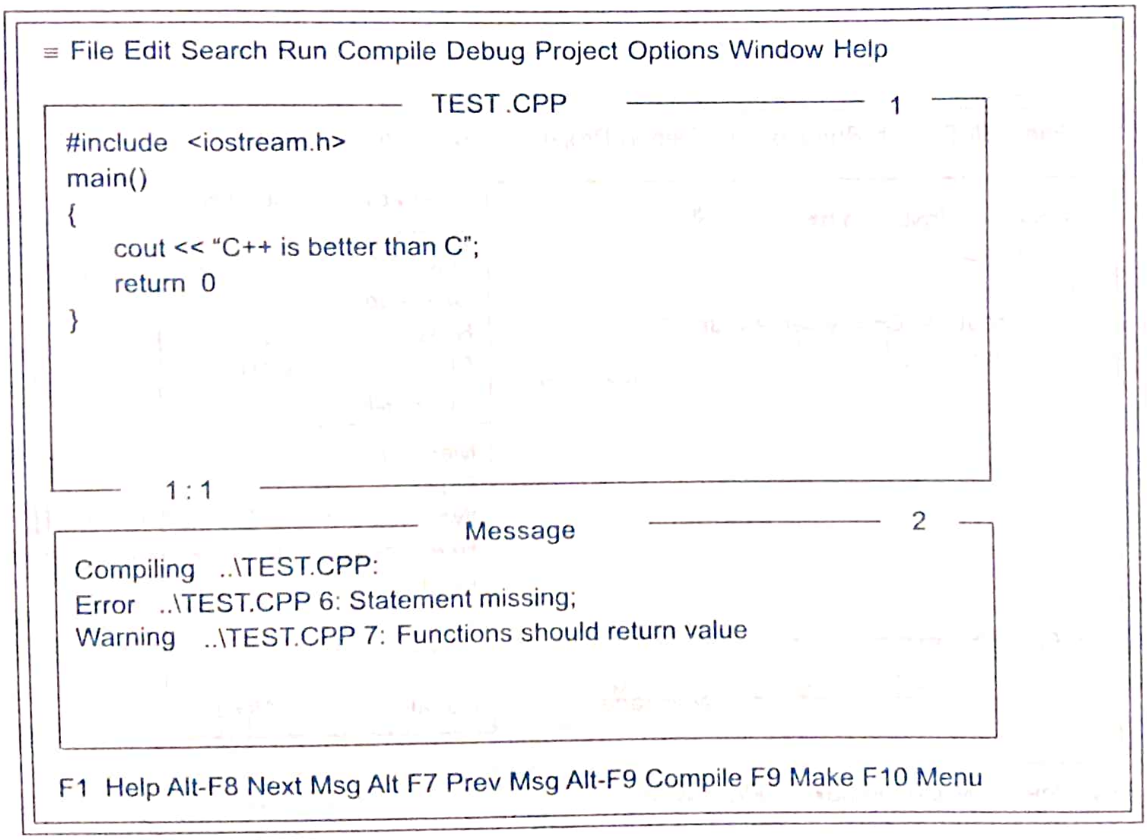
It is rare that a program runs successfully the first time itself. It is common to make some syntax errors while preparing the program or during typing. Fortunately, all such errors are detected by the compiler or linker.

### Compiler Errors

All syntax errors will be detected by the compiler. For example, if you have missed the semicolon at the end of the **return** statement in **test.cpp** program, the following message will be displayed in the message window.

```
Error...\TEST.CPP 6 Statement missing;
Warning...\TEST.CPP 7: Function should return a value
```

The number 6 is the possible line in the program where the error has occurred. The screen now will look like the one in Fig. B.10.


≡ File Edit Search Run Compile Debug Project Options Window Help
TEST .CPP
1
1 : 1
Message
2
F1 Help Alt-F8 Next Msg Alt F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
#include <iostream.h>
main()
{
    cout << "C++ is better than C";
    return 0
}
```

```
Compiling ..\TEST.CPP:
Error ..\TEST.CPP 6: Statement missing;
Warning ..\TEST.CPP 7: Functions should return value
```

**Fig. B.10** ⇔ *Display of error message*

Press ENTER key to go to **Edit** window that contains your program. Correct the errors and then compile and run the program again. Hopefully, you will obtain the desired results.

### Linker Errors

It is also possible to have errors during the linking process. For instance, you may not have included the file *iostream.h*. The program will compile correctly, but will fail to link. It will display an error message in the *linking window*. Press any key to see the message in the message window.

### Run-time Errors

Remember compiling and linking successfully do not always guaranty the correct results. Sometimes, the results may be wrong due logical errors or due to errors such as stack overflow. System might display the errors such as *null pointer assignment*. You must consult the manual for the meaning of such errors and modify the program accordingly.

## B.11 Handling an Existing File

After saving your file to disk, your file has become a part of the list of files stored in the disk. How do we retrieve such files and execute the programs written to them? You can do this in two ways:

1. Under DOS prompt
2. Under IDE

Under DOS prompt, you can invoke as follows:

```
C > TC  TEST.CPP
```

Remember to type the complete and correct name of the file with **.cpp** extension. This command first brings Turbo C++ IDE and then loads **edit window** containing the file **test.cpp**.

If you are working under IDE, then select **open** option from the *file menu*. This will prompt you for a file name and then loads the file as you respond with the correct file name. Now you can edit the program, compile it and execute it as before.

## B.12 Some Shortcuts

It is possible to combine the two steps of compiling and linking into one. This can be achieved by selecting **Make EXE file** from the compile dialogue window.

We can shorten the process by combining the execution step as well with the above step. In this case, we must select **Run** option from the run dialogue window. This causes the program to be compiled, linked and executed.

Many common operations can be activated directly without going through the main menu, again and again. Turbo C++ supports what are known as *hot keys* to provide these shortcuts. A list of hot keys and their functions are given Table B.2. We can use them whenever necessary.

| *Hot Key* | *Meaning* |
|---|---|
| F1 | Activates the online Help system |
| F2 | Saves the file currently being edited |
| F3 | Loads a file |
| F4 | Executives the program unit the cursor is reached |
| F5 | Zooms the active window |
| F6 | Switches between windows |
| F7 | Traces program; skips function calls |
| F8 | Traces program; skips function calls |
| F9 | Compiles and links programs |
| F10 | Activates the main menu |
| ALT-O | Lists open windows |
| ALT-n | Activates window n (n must be 1 through 9) |
| ALT-F1 | Shows the previous help screen |
| ALT-F3 | Deletes the active window |
| ALT-F4 | Opens an Inspector window |
| ALT-F5 | Opens an Inspector window |
| ALT-F7 | Previous error |
| ALT-F8 | Next error |
| ALT-F9 | Compiles file to .OBJ |
| ALT-SPACEBAR | Activates the main menu |
| ALT-C | Activates the Compile menu |
| ALT-D | Activates the Debug menu |
| ALT-E | Activates the Edit menu |
| ALT-F | Activates the File menu |
| ALT-H | Activates the Help menu |
| ALT-O | Activates the Options menu |
| ALT-P | Activates the Project menu |
| ALT-R | Activates the Run menu |
| ALT-S | Activates the Run menu |
| ALT-W | Activates the Window menu |

*(Contd)*

| | |
|---|---|
| ALT-X | Quits Turbo C++ |
| CTRL-F1 | Requests help about the item the cursor is on |
| CTRL-F2 | Resets the program |
| CTRL-F3 | Shows the function call stack |
| CTRL-F4 | Evaluates an expression |
| CTRL-F5 | Changes the size or location of the active window |
| CTRL-F7 | Sets a watch expression (debugging) |
| CTRL-F8 | Sets or clears a break point |
| CTRL-F9 | Executes the current program |