Microsoft® QuickC®

Up and Running

Microsoft®

Microsoft® QuickC®

UP AND RUNNING

VERSION 2.0

MICROSOFT CORPORATION

Document No. 410840040-200-R00-1088
Part No. 04321
10 9 8 7 6 5 4 3 2 1

Table of Contents

Introduction

Welcome to Microsoft® QuickC® Version 2.0, a powerful and sophisticated yet easy-to-use integrated environment for writing programs in the C language.

In recent years, the popularity of C has grown tremendously. C programs often can be ported from one computer to another. C programs are fast. C source code is compact and concise. The language encourages, but does not enforce, modular and structured programming styles. For these and other reasons, many programmers and professional software developers prefer C to any other language.

Microsoft QuickC combines the power of C with an environment that makes C easy to learn and to use. You can write code, compile and link it, run the program, and debug it, all without leaving the QuickC environment.

Read This Manual First

This manual contains all the information you need to install and begin using QuickC on your computer. There are five chapters:

Unpacking QuickC Chapter 1 lists the system requirements (hardware that you provide) and the contents of this package (software and documentation that we provide).

Installing QuickC Before you can start using QuickC, you must install it. Although the installation process is quick and easy, you may want more information about libraries and memory models. Chapter 2 guides you through installation and provides answers to commonly asked questions.

Using QuickC Chapter 3 explains how to run QuickC and introduces the window and menu environment. Next, it presents a step-by-step example of a typical development cycle: writing, compiling, linking, running, and debugging a short program.

Getting Help The Microsoft QuickC Advisor (on-line help) provides important reference information at the click of a button or press of a key. Chapter 4 illustrates the many facets of QuickC's powerful on-line help system.

Where to Go from Here Once you have unpacked QuickC, installed it on your system, and compiled a sample program, you will probably want to investigate QuickC further. Chapter 5 provides some suggestions on what to do next.

Features New to Version 2.0

If you have used an earlier version of QuickC, you'll find that with Version 2.0 you can perform many new operations and many old operations more quickly:

- Invoke the on-line help system by pressing the F1 key. The expanded and improved help system gives you fast access to a unique electronic reference book. Press one key for instant keyword help (including prototypes, explanations, and examples), environment help, error help, and cross-references to related topics. There's even a HELPMAKE utility if you want to customize the help system.
- Call dozens of new graphics functions. The presentation graphics functions transform lists of data into charts and graphs. The font functions allow you to display text in a variety of typefaces and styles. The real-coordinate graphics functions automatically scale shapes to whatever size you prefer.
- Write programs that use any of the five memory models: small, medium, compact, large, and huge. (Previous versions of QuickC were limited to the medium model.)
- Include assembly-language routines in your C source code.
- Open the three new windows: local variables, programmer's notepad, and registers.
- Cut and paste text from any window, including the example programs from on-line help.

- Compile from within QC.EXE or use QCL.EXE to compile and link from the command line.
- Discover program bugs and logic errors more quickly with the improved QuickC debugger.
- Compile programs faster when you make a small change to existing code because QuickC compiles and links incrementally.
- Use QuickC to write applications that run under Microsoft Windows.
- Customize the QuickC editor's key commands with the MKKEY program. If you'd prefer to use your own word processor, just add it to the QuickC menu.
- Learn about these and other new features in the computer-based tutorial program called LEARN.COM and in *C for Yourself*.

86
Havoline
GOODYEAR
BBS
PORSCHE

CHAPTER 1

Unpacking QuickC®

1

You're probably eager to install QuickC immediately. But you should first take a few minutes to make sure your system meets minimum requirements and to determine that your package is complete.

System Requirements

QuickC requires the following minimum configuration:

- An IBM® Personal Computer or compatible running DOS Version 2.1 or later.
- One hard-disk drive and one floppy-disk drive *or* two floppy-disk drives. A hard-disk drive is strongly recommended.
- 448K (kilobytes) of available memory (512K is recommended for medium to large projects).

NOTE Microsoft documentation uses the term "DOS" to refer to both the Microsoft and the IBM Disk Operating Systems (MS-DOS® and PC-DOS).

The QuickC Package

Check your QuickC package to see if everything is there. If any pieces are missing, contact the retailer from whom you bought QuickC. In the package, you should find the following items:

Registration card: there are many advantages to being a registered owner of QuickC Version 2.0, including notification of future software releases and easy access to customer assistance. Please take the time to fill out and mail the registration card now.

Disks: ten 5 1/4 inch floppy disks or five 3 1/2 inch disks. The distribution disk labeled "Setup" contains a file named PACKING.LST that lists the location and description of all disk files in the Microsoft QuickC package.

Up and Running: the book you're reading now. It explains how to install and use QuickC.

C for Yourself: this book is written for programmers who know at least one language (such as BASIC or Pascal) but who don't know C. Part 1, "Learning C," is a tutorial that explains how to program in C. Many examples are included. Part 2, "Using C," examines further the library functions that perform input and output, the functions that create graphics, and new features such as real coordinate graphics, presentation graphics, fonts, and in-line assembly. The appendixes summarize the C language and QuickC library functions.

Microsoft QuickC Tool Kit: this book explains the individual tools and utilities that accompany QuickC. Beginners probably won't need to refer to this book while they're learning the fundamentals of C. Advanced C programmers should turn to this book for detailed information about compiling, linking, creating libraries, maintaining multiple-module programs, and more.

IMSA
CAMEL GT
Frank
lter Gerber
avolir

CHAPTER 2

Installing QuickC

2

This chapter tells you how to install QuickC on your system. The SETUP.EXE program on the Setup distribution disk performs the installation.

SETUP.EXE does two things. First, it copies several programs (the compiler, the linker, the library manager, the help system, and others) from the distribution disks to your hard disk (or floppy disks). Second, it creates one or more combined libraries. You can't program in C without a library.

This chapter also explains why it's necessary to build combined libraries and how the components fit together.

If you follow the instructions, when you finish the chapter you'll have a working version of QuickC on your system and you'll be ready to start programming in C.

A Quick Overview

There are four steps to installing QuickC:

1. Make backup copies of all distribution disks.
2. Read the first section of the README.DOC file for information on installing QuickC. If there are any corrections to this book, they're listed at the beginning of README.DOC.
3. Run SETUP.EXE, which is an interactive program. The questions you answer determine the QuickC environment that is installed. Default answers are listed in brackets. At the bottom of each screen is a brief synopsis of what each question means.

4. Adjust your system and environment variables. SETUP.EXE creates two files: NEW-VARS.BAT and NEW-CONF.SYS. Add the information from NEW-VARS.BAT to your AUTOEXEC.BAT file. If you'd prefer not to make the changes permanent, you can run NEW-VARS.BAT as a batch file. Change your CONFIG.SYS file if the `files` value and `buffers` value are currently smaller than the values in NEW-CONF.SYS. After modifying these files, reboot.

The process is not difficult and each screen provides helpful guidance. If you have enough DOS and programming experience to complete these four steps without further assistance, we encourage you to do so. (If you get stuck, return to this chapter.) You may then skip ahead to Chapter 3, "Using QuickC."

NOTE *If you make a mistake during the setup process, just run the program again. SETUP.EXE never erases files from the distribution disks.*

Running SETUP

Before you run SETUP, back up the distribution disks using the DOS COPY command or the DISKCOPY program. Then read the first part of the README.DOC file by loading it into a word processor or by using the TYPE command:

```
TYPE README.DOC | MORE
```

When you are ready to install QuickC, insert the Setup disk in drive A and then change to that drive (type `A:`). At the DOS command line, type

```
SETUP
```

NOTE *The following instructions assume that you plan to install QuickC on a system that has at least one floppy-disk drive and one hard-disk drive. If you don't own a hard drive, you must have at least two floppy drives. Please read the section "Installing on a Floppy-Disk System" at the end of this chapter if you aren't installing QuickC on a hard drive.*

Correcting Mistakes

Each setup screen ends by asking if you want to change any of the options. If you press the Y key, you can correct any of the answers you entered.

To exit the SETUP program at any time, press CTRL+C.

NOTE *Remember, the SETUP program doesn't erase any files from the distribution disks. If you make a mistake during the setup process, just run SETUP again.*

Understanding Libraries

Compared to other programming languages, C is very lean. It contains only a few dozen keywords and operators. To print something on the screen, for example, you call the **printf** function, which is *not* a keyword or an operator. It's not included in the C language proper. Strictly speaking, **printf** is a "library function."

When you link a program, the linker looks in the current library for any functions that were not defined in the main program. If your source file calls **printf**, for example, the linker looks in the library for the machine code that executes the function, adds it to the executable program, and resolves any references to **printf**.

A library, then, is just a set of commonly used functions that have been gathered into one place. The American National Standards Institute (ANSI) defines a great number of library functions (including **printf**). Microsoft QuickC adds even more. In C, the keywords and operators number in the dozens, while the library functions number in the hundreds. Incidentally, you can add your own heavily used functions to the library (or combine them into a separate library) if you wish. See *Microsoft QuickC Tool Kit* for more information on using the Library Manager (LIB.EXE).

One of the key jobs performed by SETUP.EXE is building at least one large "combined library." Individual smaller parts of the library are called "component libraries." The two component libraries you must have are the memory model library and the math package library. The two additional component libraries containing graphics functions are optional.

You'll ultimately use one of the combined libraries when you compile and link a program.

SETUP Options

SETUP has two options for shortcuts: /H and /L. You should *not* use these options the first time you run SETUP.

Use the /L option after you have already installed QuickC and you want to build additional combined libraries without going through the entire setup process.

Combined libraries are described above (see "Understanding Libraries"). If you wish to add more libraries in the future, use the /L switch:

```
SETUP /L
```

The /H option suppresses the SETUP help information sent to the screen. If you install QuickC again, you can bypass the help information by using the /H option:

```
SETUP /H
```

SETUP Stage One

Before SETUP can install QuickC, it needs some information about your system and the type of programming you intend to do. This first stage of SETUP is a series of questions split into three screens.

First Screen: The Libraries

The first screen asks these questions:

```
Source of compiler files [A:]:
Installing on a hard disk drive [Y]:
Math Options: Emulator [Y]: 8087 [N]:
Memory Models: Small [Y]: Medium [N]: Compact [N]: Large/Huge [N]:
Delete the component libraries when finished [Y]:
Include in combined libraries: GRAPHICS.LIB [N]: PGCHART.LIB [N]

Do you want to change any of the above options [Y]:
```

The questions appear on the screen one at a time. An explanation of each option appears at the bottom of the screen, unless you added the /H option when you ran SETUP. Answer the questions by typing in your responses and pressing ENTER.

Each question ends with a default answer inside square brackets (`[Y]`, for example). Press ENTER to accept the default. *If you are unsure of the proper reply for any of these questions, consider the default a good place to start.* If you later find you would have preferred to make another choice, you can always run SETUP again.

Each screen concludes by asking if you want to change any of your choices. When you're satisfied, press N (the default answer is Y, which returns you to the first question on the screen). If you start over, the default answers become the choices you made previously.

Which Disk Drive?

The first two questions on the first screen ask where you're starting from (the drive containing the distribution disks) and where you're going (destination disk):

```
Source of compiler files [A:]:
Installing on a hard disk drive [Y]:
```

Which Math Library?

Your answer to the third question determines which math component libraries will be included in the combined library:

```
Math Options: Emulator [Y]:    8087 [N]:
```

Some computers contain an 8087 or 80287 math coprocessor chip. Some don't.

If your machine is equipped with a coprocessor, linking your programs with the combined library tailored for the 8087 will speed up all floating-point calculations. However, these programs will run only on a machine that has an 8087 or 80287 chip.

Programs linked with the emulator library, on the other hand, will run on any computer, whether it has a coprocessor or not. The emulator library does check for the presence of a math coprocessor. If the computer has a coprocessor, it performs all floating-point math operations. If no coprocessor is installed, the emulator library "emulates" (imitates) the actions of a coprocessor.

If you're not sure which math library to include, choose the emulator library because it's the most flexible.

You may include both math libraries, but doing so doubles the number of combined libraries SETUP builds, thus doubling the time it takes to install QuickC. If you intend to install all possible library configurations, you'll need approximately 6 megabytes of available space on your hard disk. If you install just one library, you'll need about 2.7 megabytes of free space.

Which Memory Model?

The next question asks which memory models you plan to use:

```
Memory Models: Small [Y]:   Medium [N]:   Compact [N]:   Large [N]:
```

The small memory model is the default. If you're in a hurry to install QuickC, accept the default answer by pressing ENTER four times and bypass the explanation below. Should you discover that you need additional memory for your programs, you can run the SETUP program again. If you're curious about memory models, read on.

The 8086 processor and its relatives access memory in 64K blocks called "segments." To move outside of the current 64K segment requires additional machine instructions. Also, an address within the 64K segment can be specified with only two bytes. If you wish to access more than 64K of memory locations, their addresses must include additional bytes (which makes pointer variables longer and program execution slower).

C programs have two parts: the code (machine instructions) and the data (variables and constants). If you write programs that fit either the code or the data entirely within one 64K segment of memory, the program will execute faster and use less memory. However, if you need to use multiple memory segments for the code or the data, you may, although the program will run more slowly.

The various configurations of memory usage are called "memory models." QuickC supports five standard memory models. Table 2.1 illustrates the relationship between the five available memory models and the limits placed on their code and data segments.

Table 2.1 Memory Models

Memory Model	Code Segment Limit	Data Segment Limit
Small	64K	64K
Medium	None	64K
Compact	64K	None
Large	None	None
Huge	None	None

For a great many applications, the small memory model suffices. If you're working on a large database manager, you might want one of the models that provides unlimited data segments (compact, large, or huge). If you're writing a program with a great many functions, you might want to remove the limit on the code segment (medium, large, or huge).

The huge memory model uses the same library as the large model, so the SETUP program offers only four choices. The difference between large and huge is that the huge model allows individual arrays to exceed 64K, whereas the large model limits arrays to 64K.

Any or all of these memory models may be selected, but SETUP.EXE will build a separate combined library for each model. For example, if you choose the small memory model and the math emulator, the library file SLIBCE.LIB is created. If you choose the medium model and the 8087 math package, MLIBC7.LIB is created. The small memory model libraries start with the letter "S," medium with "M," compact with "C," and large with "L." Similarly, the emulator math package is shown by a final "E" in the file name, the 8087 package by a "7."

Given four memory models and two math packages, you can create a total of eight combined libraries. But the more combined libraries you create, the more space they'll take on your hard disk and the longer the installation process will take.

The first time you install QuickC, create only one or two combined libraries. If you find a need for other memory models or math packages, just rerun the SETUP program.

Include Graphics Libraries?

The graphics library GRAPHICS.LIB contains numerous functions for drawing lines, rectangles, circles, and other shapes. If you plan to write programs that use these functions, press Y to answer the first of these two questions:

```
Include in combined libraries: GRAPHICS.LIB [N]:  PGCHART.LIB [N]:
```

However, if you plan to write programs that use text output only, then you probably don't need GRAPHICS.LIB taking up room on your disk. Press N to omit the graphics library. (If you very rarely need graphics functions, you can omit the graphics library and, whenever you need the functions, explicitly link the graphics library GRAPHICS.LIB. See the *Microsoft QuickC Tool Kit* manual for more information about linking with a specific library.)

The PGCHART.LIB library contains presentation graphics functions for creating high-resolution graphs (line graphs, bar charts, column charts, scatter diagrams, and pie charts). Again, if you want to display such graphics in your programs, press Y to include this component library. If you'll use presentation graphics rarely or not at all, press N to omit this library.

Second Screen: Extra Files

The next screen asks the following questions about copying additional files to your hard disk:

```
Install Microsoft Mouse [Y]:
Copy documentation files [Y]:
Copy the DOS patch file [N]:
Copy sample C programs [N]:
Copy the QuickC tutorial files [N]:

Do you want to change any of the above options [Y]:
```

SETUP will copy these extra files to your hard disk if you want them.

As before, each of these questions is accompanied by an explanation at the bottom of your screen to help you decide if these files would be useful.

Third Screen: The Directories

The final set of questions asks for the names of directories in which to store the various files.

```
Directory for Executable files [C:\QC2\BIN]:
Directory for Libraries [C:\QC2\LIB]:
Directory for Include files [C:\QC2\INCLUDE]:
Directory for Sample files [C:\QC2\SAMPLES]:
Directory for Tutorial files [C:\QC2\TUTORIAL]:

Do you want to change any of the above options [Y]:
```

You don't have to choose any of the default options. You might decide you want your executable programs in C:\QC2\BIN and your source files in C:\C_CODE, for example. Type in the names of directories you want SETUP to use (including the drive name in the path). If the directories don't exist, SETUP asks if you want to create them.

Checking the Available Disk Space

Before moving on to stage two, the SETUP program checks your hard disk to see how much free space is available. If you don't have enough room to install QuickC (the amount needed varies according to the options you've chosen), an error message tells you how much space is required for the files you've requested.

If you attempt to install all possible libraries, you'll need approximately 6 megabytes. If you install just one combined library, you'll need roughly 2.7 megabytes.

If you don't have enough room on your hard disk, you have two choices. First, you can delete files from the hard disk until there's enough room for the libraries you want to create. Second, you can reduce the number of libraries you request (or you can choose not to copy the samples and documentation files), to cut down the amount of space you need. Either way, you'll have to run SETUP again.

SETUP Stage Two

Now that you've specified your system and programming needs, SETUP goes to work. This stage requires you to swap the distribution disks in and out of the drive. Insert the disks as SETUP asks for them and press ENTER. If you put in the wrong disk, SETUP will ask again for the proper disk.

At this point, all SETUP needs is the go-ahead from you. It starts building the combined libraries you requested, placing them in the directories you specified.

SETUP Stage Three

When SETUP finishes creating the combined libraries, it creates two files: NEW-VARS.BAT and NEW-CONF.SYS. To install these files permanently, you must now adjust the DOS environment and modify both your AUTOEXEC.BAT and CONFIG.SYS files.

NOTE *If you use other languages that have their own LINK.EXE program, you may not want to put the QuickC linker in your path. If this is the case, you can modify CONFIG.SYS and run NEW-VARS.BAT each time you use QuickC.*

Changing AUTOEXEC.BAT

The file NEW-VARS.BAT created by SETUP might look something like this:

```
PATH=C:\QC2\BIN;C:\DOS;C:\MYEXE;C:\WIN386;C:\WORD
set LIB=C:\QC2\LIB
set INCLUDE=C:\QC2\INCLUDE
```

PATH is a system variable that tells your computer where to find executable programs. The LIB and INCLUDE variables tell QuickC where to find the libraries and the include files.

To make the change permanent, load your current AUTOEXEC.BAT file into an editor or word processor (you'll find AUTOEXEC.BAT in the root directory), add the new path names, and save the modified file.

You can edit the AUTOEXEC.BAT file with the QuickC editor if you wish. Change to the directory containing the QC.EXE program (for example, type `CD \QC2\BIN`) and type `QC \AUTOEXEC.BAT` (the backslash indicates that the file is in the root directory). Then use the File menu Merge command to merge NEW-VARS.BAT into AUTOEXEC.BAT.

In most cases, it is best to change the AUTOEXEC.BAT file and reboot. However, if you use other languages and other compilers, you may wish to leave the AUTOEXEC.BAT file alone and run the NEW-VARS.BAT file before each QuickC session.

Modifying CONFIG.SYS

The NEW-CONF.SYS file might look like this:

```
files=20
buffers=10
```

You need to be sure that the `files` and `buffer` values are large enough to contain QuickC. Load your CONFIG.SYS file into an editor or word processor (again, it should be in the root directory), change the two lines that refer to files and buffers, and save the modified file. If your current CONFIG.SYS file has higher numbers (`files = 30`, for example), you can leave the higher value in effect. The numbers in NEW-CONF.SYS are minimums; you may safely use higher values.

NOTE *Merely changing the AUTOEXEC.BAT and CONFIG.SYS files does* not *affect the current DOS environment. To put the changes into effect, you must reboot your machine by powering off and then on or pressing CTRL+ALT+DEL*

After installing QuickC, changing the files, and rebooting, you can proceed to Chapter 3, "Using QuickC."

Installing on a Floppy-Disk System

The procedure for installing on a system with two floppy drives is similar to the hard-drive installation procedure.

NOTE *The explanations of libraries, memory models, and math emulators aren't repeated here. Read the appropriate sections above for more about these topics.*

Before you begin the SETUP program for a floppy-disk system, you'll need to have ready one blank formatted disk for each combined library you plan to build plus one extra. The extra disk is what SETUP calls a "scratch disk." SETUP writes intermediate files to this disk when it builds combined libraries.

The first screen looks like this:

```
Source of compiler files [A:]:
Installing on a hard-disk drive [Y]:
Math Options: Emulator [Y]: 8087 [N]:
Memory Models: Small [Y]: Medium [N]: Compact [N]: Large/Huge [N]:
Delete the component libraries when finished [Y]:
Include in combined libraries: GRAPHICS.LIB [N]: PGCHART.LIB [N]

Do you want to change any of the above options [Y]:
```

To install QuickC on a floppy system, you must avoid the default answer to the second question. Type N in response to the question about installing on a hard drive.

The second and third screens (described above) do not appear. The final question is this:

```
Drive to use to build combined libraries [B:]:

Do you want to change any of the above options [Y]:
```

The rest of the SETUP procedure involves swapping the disks that SETUP requests. When this is complete, you no longer need the scratch disk and you may delete any remaining files from it.

You must also change the AUTOEXEC.BAT and CONFIG.SYS files on your boot disk. See the section "SETUP Stage Three."

When you have finished installing the libraries, reboot your computer.

Using QuickC on a Floppy-Disk System

To compile and link on a system with two floppy drives, both drives (A and B) must be in your path. You can type the following line at the DOS prompt or add it to your AUTOEXEC.BAT file:

```
SET PATH = A:\; B:\
```

In addition, you should choose one drive as the current drive and place the source-code files on a working disk in that drive. If you use any **#include** directives, you should copy the include files to the source-code disk. For example, if a program called TEST.C contains the line

```
#include <stdio.h>
```

then both TEST.C and STDIO.H should be on the same disk.

You should use the second drive for the QC.EXE disk and other QuickC programs. For example, if A is the current drive, you should place the QC.EXE disk in drive B. To compile TEST.C (from the disk in drive A) using the QCL program (in drive B), type this:

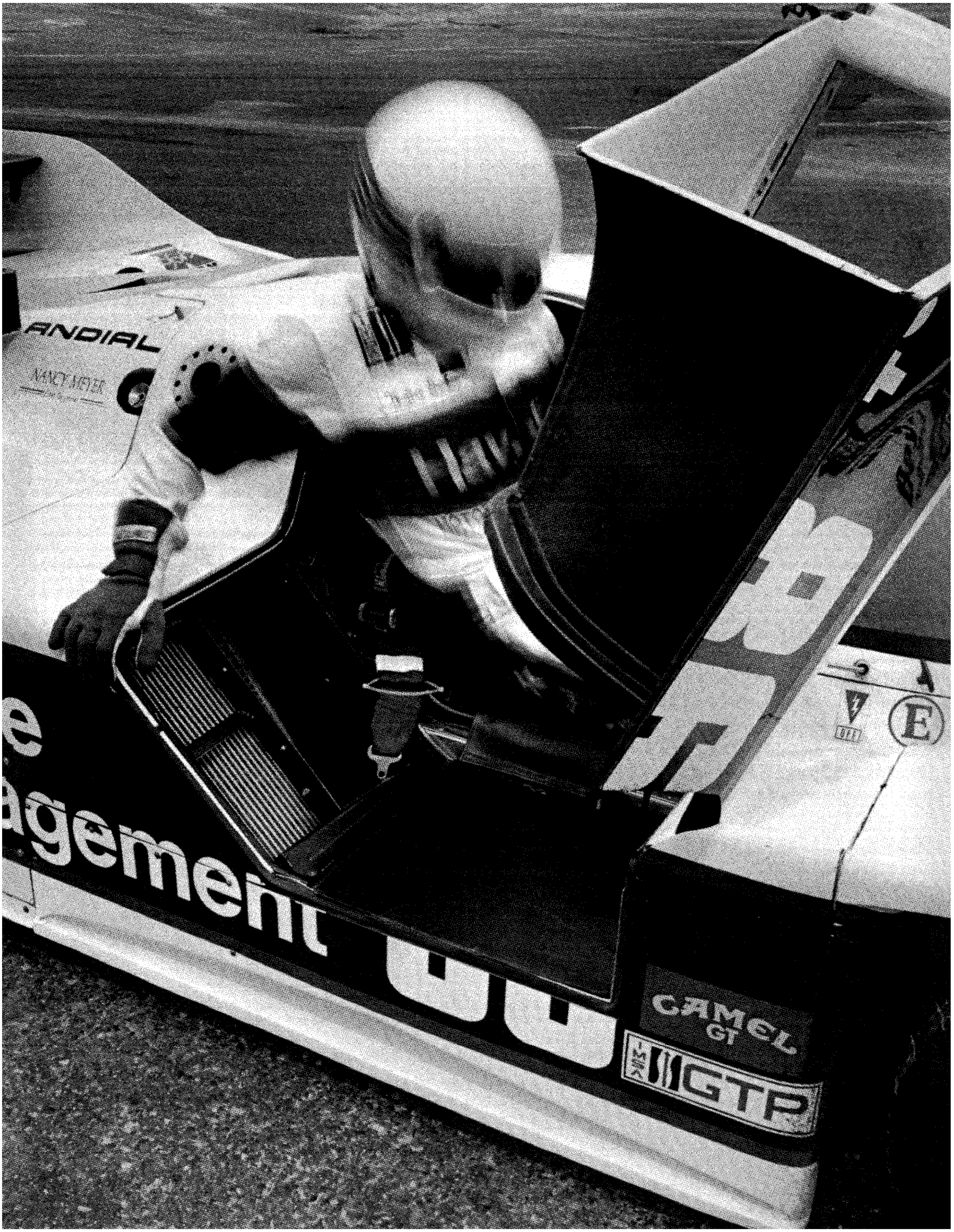
```
A:>  QCL   TEST.C
```

Note that the DOS prompt indicates A is the current working drive.

When you compile and link a program, you will be prompted to insert the appropriate disks in drive B. The .OBJ and .EXE files will be created on drive A.

Warning *You must have both drives in your path. The current working drive must contain the source code. You must never swap disks from this drive; always swap from the other drive.*

When you compile a program on a system with two floppy drives, QuickC prompts you to insert disks as it needs them. Be sure to wait for the prompt before you swap disks. For a complete list of disks and files, print out the file called PACKING.LST on disk 1.



ANDIAL
NANCY MEYER
e
agement
CAMEL GT
IMSA GTP

CHAPTER 3 Using QuickC

3

If you followed the instructions in Chapter 2, "Installing QuickC," you now have a working version of QuickC and are ready to write your first program. This chapter introduces the QuickC environment—a powerful tool to help you write and test programs.

In this chapter you'll work through a sample compiling and linking session. When you finish, you will have written, saved, built, and tested a working program.

If you'd prefer to experiment on your own or if you've previously used QuickC Version 1.0, you may skip this chapter (or skim through it). We strongly suggest, however, that if you do nothing else, review the next chapter, "Getting Help." In addition, we recommend that all QuickC users run the LEARN program, which teaches how to use the QuickC environment.

The QuickC Environment

QuickC is a window-based programming environment that integrates a text editor, a compiler, a linker, a debugger utility, a make utility, and an on-line help database. This chapter introduces and describes the following aspects of QuickC:

Windows and Menus The menu system allows you to quickly find the command or action you need. The first part of this chapter explains how to open and close windows and how to navigate the menus. It also defines certain terms used throughout the chapter.

Editor When you're writing source code, you'll spend a lot of time using the QuickC editor. If you know WordStar® commands, you'll know how to use the

QuickC editor. If you'd prefer to customize the editor, you can use the MKKEY utility, which is explained in the "Customizing the Editor" section.

Compiler/Linker You can compile, link, and test a program without ever leaving the editor. QuickC's integrated environment saves you hours of development time. This part of the chapter defines and illustrates the various compiling and linking options.

Debugger The debugger allows you to set breakpoints, to monitor the status of key variables, and to trace program execution line by line. This section of the chapter provides a brief overview of the debugger utility. The LEARN program includes a lesson that provides more details about the new, advanced debugging enhancements.

Using Windows and Menus

Even if you've never used windows and menus before, you'll find the QuickC programming environment easy to learn.

This section introduces the QuickC environment. You'll learn how to control windows and to choose commands from the menus.

Getting Started

To run QuickC, type

```
QC
```

at the DOS prompt. You'll immediately enter the QuickC editor.

If you enter a file name after the `QC` command, for example,

```
QC MYFILE
```

QuickC automatically adds the extension .C that marks a C source file. Typing the line above causes QuickC to load MYFILE.C.

If QuickC can't find the file you specified in the current directory, it asks if you want to create a new file.

If you do not enter a filename after the `QC` command, QuickC opens an empty file named UNTITLED.C, which you can rename later or save with another name.

Command-Line Options

Depending on your particular hardware, you may be required to include one of the following options on the command line after the `QC` command but before the file name, for example, `QC /b MYPROG`.

Option	Hardware
`/b`	For black-and-white systems, including Hercules® monochrome monitors, LCD screens, and black-and-white monitors.
`/g`	For AT-compatible systems that refresh the screen at a slower rate than the standard AT (including some COMPAQ® systems).
`/h`	For systems equipped with EGA-, VGA-, or MCGA-compatible graphics cards capable of displaying more than 25 lines of text.
`/nohi`	For systems that don't support high-intensity colors (including LCD monochrome monitors and some Amdek® color monitors).

Using the Mouse and Keyboard

You can enter all commands from the keyboard. If you own a Microsoft (or fully compatible) Mouse, you may choose to use either the keyboard or the mouse to enter commands. When this book explains a command, the two options are marked with icons of a key or a mouse as follows:

Press the ALT key.

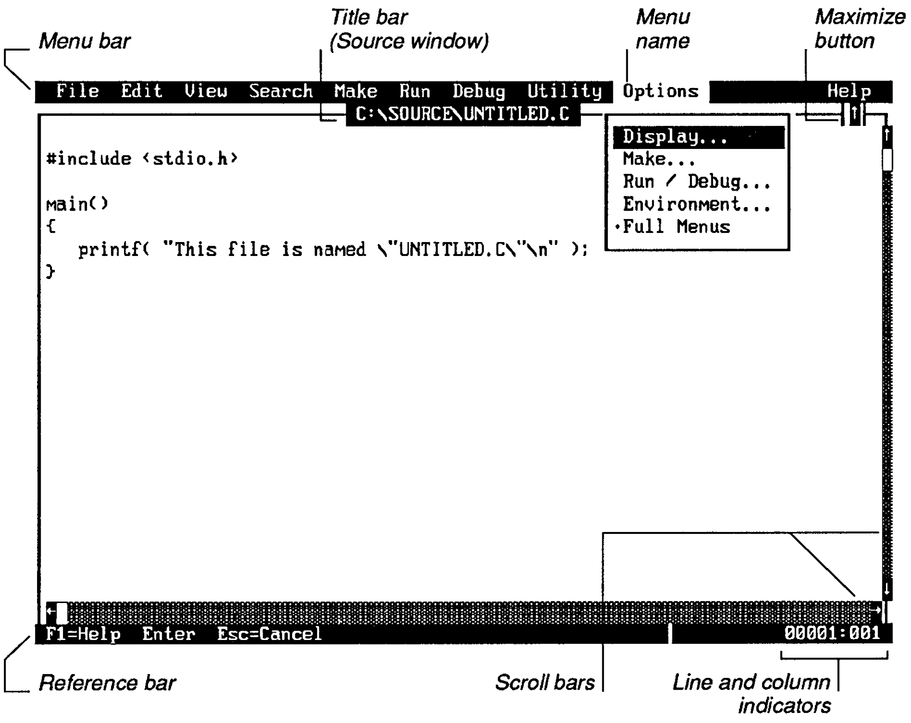
Click the File menu, then click Save.

***NOTE** Unless the right button is specifically mentioned, "clicking" means that you click the mouse's left button once.*

Windows

The system of windows and menus is simple to use and intuitive. Many programmers can learn how to use the QuickC environment without learning the terms that describe the various menus and buttons. In addition, if you're confused about windows or menus, you can almost always call up a Help window that explains how a menu works (see Chapter 4, "Getting Help"). However, if you want to read further in this chapter, you'll have to understand the terms that are used.

Figure 3.1 shows a typical QuickC screen, with labels that describe its parts. Some of the parts provide information only. For example, if the CAPS LOCK key is on, a letter C appears in the bottom right corner. The letter C is informational (it tells you the CAPS LOCK key is on). Other parts of a window perform actions triggered by a specific key or mouse action. For example, if you click the little box in the upper-left corner of a window, the window closes. The box is not informational; it's active.



```
#include <stdio.h>

main()
{
   printf( "This file is named \"UNTITLED.C\"\n" );
}
```

Figure 3.1 Parts of a Window

The parts of a window, their status, and their use are listed in Table 3.1 below.

Table 3.1 Parts of a Window

Name	Status	Use
Close button	Active	Closes the current window (the Source window cannot be closed). Appears in upper-left corner.
Menu bar	Active	Lists names of the available menus.
Title bar	Informational	Shows name of the window (the Source window title bar lists the file currently being edited).
Source window	Active	Contains source code for the program you're writing. Seven other windows are supported: Debug, Help, Locals, Registers, Notepad, Output, and Errors.
Maximize button	Active	Shrinks or enlarges the current window.
Reference bar	Informational/ Active	Lists shortcut keystrokes (keyboard users) and direct commands to QuickC (mouse users).
Scroll bars	Active	Indicate your position in the current file. If you click in the gray area on either side of the position marker, you move in that direction. If you click on the arrows, you move one line (or one character) in that direction. If you click and drag the position marker, you can move anywhere within the file.
Line/Column indicators	Informational	Show the current line and column of the text cursor. C means CAPS LOCK is on. N means NUM LOCK is on. R means the file is set to Read Only status. D means Debugging History is on. I means the debugging history includes user input.

Using the Menu Bar

To choose a command from a menu, you "pull down" or "open" the menu and choose the command you want:

1. Press the ALT key to activate the menu bar.
2. Press the highlighted character in the menu name (F = File, for example).
3. Press the highlighted character in the item name (in the File menu, S = Save, for example).

Or follow these steps:

1. Press the ALT key.
2. Use the right and left DIRECTION keys to move to the menu you want.
3. Use the up and down DIRECTION keys to highlight the command.
4. Press ENTER.

Or:

1. Open the menu by clicking the menu name.
2. Click the command.

The Menu Vanishes

If at any point you decide that you don't want to choose a command from a menu, you can make the menu disappear:

Press the ESC key.

Click somewhere on the screen outside of the menu.

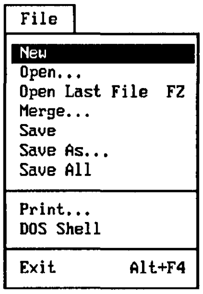
Shortcut Keys

In the menus below, you'll notice that certain menu items are followed by names of keys. These are the "shortcut keys" for heavily used commands. For example, the Run menu Restart command is followed by SHIFT+F5, which means that instead of opening the Run menu and choosing the Restart command, you can instead hold down the SHIFT key and press F5.

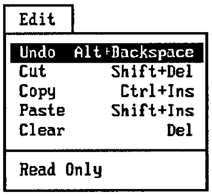
NOTE *The reference bar displays commonly used shortcut keys. In addition, the inside front cover of this book lists the important shortcut keys for easy reference.*

Menus

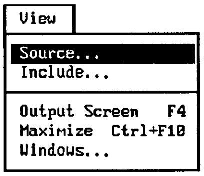
The menu bar contains ten menus, which you can pull down at any time. If you don't know what a menu does, invoke the on-line help system by highlighting the menu title (or highlighting a command within a menu) and pressing F1 or clicking the right mouse button. QuickC uses the following menus:



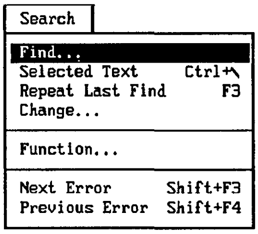
The File menu controls files, allowing you to clear the Source window (New), load an existing source file (Open), append a file to the source code in memory (Merge), save the current file (Save), rename the current file (Save As), print the source code (Print), temporarily exit to DOS (DOS Shell), or permanently exit QuickC (Exit).



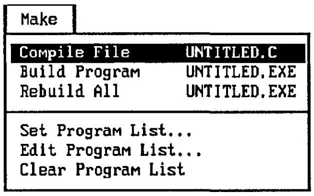
From the Edit menu you manipulate text by deleting, cutting, copying, and pasting lines of code. You can also set Read Only status to protect source files.



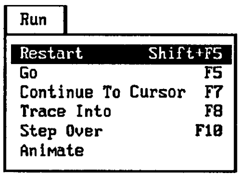
The View menu controls the visible screen. You can rapidly switch between multiple modules (Source), read through include files (Include), make visible the output screen (Output Screen), expand the Source window (Maximize), or open and close the various windows (Windows).



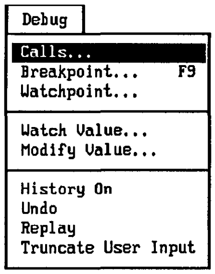
The Search menu invokes the commands that find or replace text or functions in source files. It also searches for the next source line that caused a compiler error.



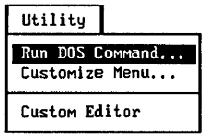
The Make menu allows you to compile (Compile) or to compile and link (Build) programs. From this menu, you also create or edit program lists that name the components of a multimodule program.



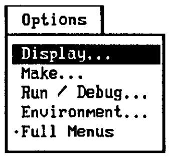
Once a program is compiled and residing in memory, use the Run menu to run it. You may run it from beginning to end, selectively run individual sections, or trace through the program line by line.



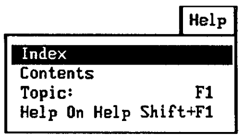
If your program contains logic errors, the Debug menu allows you to set breakpoints and watchpoints. In addition, you can keep track of variables and their changing values. (This feature means there's no need to sprinkle **printf** functions throughout your program just to watch variables as they change values.)



The Utility menu allows you to run DOS commands and programs from within the QuickC environment. If you find that you use certain programs often, you may want to add them to the Utility menu (Customize Menu). For example, you can add your favorite program editor to this menu if you wish.



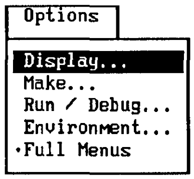
The items on the Options menu control the integrated environment. For example, the Options menu Display command allows you to change the colors used on the screen. The Options menu Make command controls flags that affect the actions of the compiler and linker.

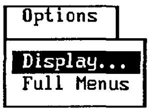


The Help menu is one route to the on-line help system. For more about this topic, see Chapter 4, "Getting Help."

Short Menus and Full Menus

If you open the Options menu, you'll see one of the two menus below:





When five commands are listed under Options and a dot appears beside Full Menus, it means full menus are turned on. When the Options menu holds only two commands and Full Menus has no dot, short menus are in effect.

To change from full menus to short menus or vice versa, choose the Full Menus command. The command is a toggle button; that is, it changes from on to off, or off to on, when pressed.

The short menus contain all the commands you need to write, compile, and run a C program. The short menus may seem easier to use, especially for beginners. If you prefer to see every possible option, you should enable Full Menus. The choice is yours.

NOTE *When you set preferences from the Options menu—full menus, memory models, compile options, and so on—your choices are saved in the current directory in a file called QC.INI. If you set an option, it stays set from one QuickC session to the next, or until you change it.*

Shaded Commands

When a command within a menu is shaded, it is unavailable. You can't use it.

For example, when you first run QuickC and haven't yet compiled or linked a program, you can open the Search menu and see that both Next Error and Previous Error are shaded. You haven't done anything to generate errors, so there are no errors to view.

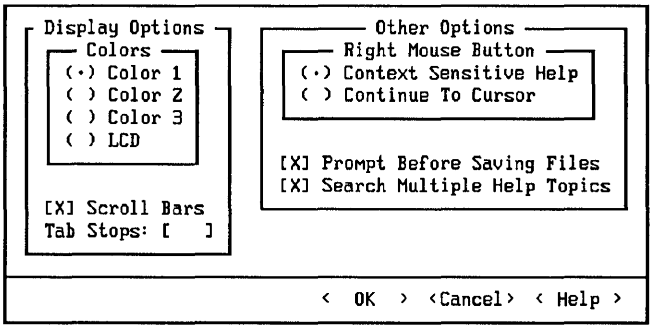
Ellipses...

When a command is followed by three periods (an ellipsis), it means QuickC needs more information before it executes the command. If a command is not followed by an ellipsis, choosing the command causes it to execute immediately.

For example, the File menu contains both the Save and Save As... commands. Choosing Save causes QuickC to save the current file with the current name (the file name appearing at the top of the Source window). Choosing Save As... causes a dialog box to appear (see the following section). Within that dialog box, you type the new name for your file.

Dialog Boxes

Very often, invoking a menu command causes a dialog box to appear. For example, Figure 3.2 shows the dialog box opened by the Options menu Display command.



```
Display Options                 Other Options
  Colors                          Right Mouse Button
  (•) Color 1                     (•) Context Sensitive Help
  ( ) Color 2                     ( ) Continue To Cursor
  ( ) Color 3
  ( ) LCD                       [X] Prompt Before Saving Files
                                [X] Search Multiple Help Topics
[X] Scroll Bars
Tab Stops: [   ]

                        <  OK  > <Cancel> < Help >
```

Figure 3.2 Dialog Box

NOTE *Dialog boxes usually offer a set of shortcut keys. Press* ALT *to see which keys are active within a dialog box.*

Dialog boxes can contain one or more of the items on the following list. Use the TAB key to move between the various items in a dialog box.

Option Buttons (•)	Option buttons offer a list of choices, of which you choose only one. Use the DIRECTION keys to move between the choices. In Figure 3.2, four option buttons allow you to pick the screen color you prefer. These are sometimes called "radio buttons," because they're similar to the buttons on a car radio: pushing in one button causes the others to pop out.
Check Box [X]	A check box is a yes/no switch. If the box is empty, the feature is turned off. If it contains a letter X, the feature is on. Press SPACEBAR to turn a check box on or off. Use the TAB key to move between check boxes.
Text Box [ ]	A text box contains text that you enter. In Figure 3.2, the setting for Tab Stops requires you to type in the number of spaces to be inserted when you press the TAB key.
Command Buttons < OK >	Command buttons enclosed in angle brackets pass commands to the dialog box. The < OK > button means you're satisfied with the choices you've made. The < Cancel > button allows you to exit the dialog box with no changes. If one of the command buttons is highlighted, pressing ENTER invokes that command.
List Boxes	Certain dialog boxes display the current disk directory inside a list box. If the number of files is too large for the list box, you may use the DIRECTION keys and PGUP/PGDN (keyboard) or click the scroll bar to move around the list (mouse).

Using the Editor

QuickC's program editor is an important part of the QuickC environment. This part of the chapter provides a brief overview of its many functions.

Moving Around in a Source File

Many of the keys within the editor act as you would expect. The PGDN and PGUP keys advance you forward or back within the source code. The HOME key moves the cursor to the beginning of the current line. The END key moves the cursor to the end of the current line. The DIRECTION keys move the cursor one character at a time.

NOTE For a complete list of editor commands, use on-line help. Open the Help menu, choose the Contents command, then ask for help about the Keyboard under the Using QuickC heading.

You can invoke many of the editor commands in two different ways. For example, to move one word to the right, you can press either CTRL+RIGHT or CTRL+F. The second choice is part of the WordStar-compatible command set. If you're familiar with WordStar commands, you already know how to use the QuickC editor. There is one difference you should know: you don't use separate commands to mark the beginning and end of a block of text.

Defining a Block

While you're working on a C program, you may want to delete a large block of text or copy it to another place in the program. To do this, you must define the block:

Move the cursor to the beginning of the block. Hold down the SHIFT key and move to the end of the block. Use the other editing keys (HOME, END, CTRL+RIGHT, and so on) while you're holding down SHIFT to extend the block.

Move the mouse cursor to the beginning of the block. While holding down the left button, move to the end of the block.

After defining a block, you can do several things: use the DELETE key to erase it or use SHIFT+DEL to copy it into the Paste buffer (then SHIFT+INS to insert the buffer into the source file at the current cursor location).

While a block is defined, anything you type will replace the defined text.

You may also press the TAB key to indent the entire block an additional tab setting or SHIFT+TAB to remove all tab settings.

Customizing the Editor

If you'd prefer to use another set of editing commands, QuickC comes with four "key" files and a utility for making your own key file. The four key files are QC.KEY, ME.KEY, BRIEF.KEY, and EPSILON.KEY. Refer to the Appendix for a complete list of the commands they control (for example, the QuickC and Microsoft editors use CTRL+E to move the cursor up, while Epsilon™ uses CTRL+P).

To change to a new key file, use the /k: option when you run QuickC. For example, to load the BRIEF.KEY file, enter this line:

```
QC /k:BRIEF.KEY
```

Your preference is automatically saved in the QC.INI file. In future editing sessions, you won't need to specify the key file.

Creating Your Own Key File

The MKKEY program allows you to make your own key file. You must use three options: -c, -i, and -o. The first (-c) specifies the type of conversion: ASCII to binary (ab) or binary to ASCII (ba). The two others specify the input file (-i) and the output file (-o).

To modify the default QC.KEY file, you first convert it to an editable ASCII file:

```
MKKEY -c ba -i QC.KEY -o MYEDITOR.TXT
```

You may use any text editor (including QuickC's) to edit the file named MYEDITOR.TXT, which lists the keystrokes that perform certain actions. For example, you press CTRL+G to delete a character. The line in MYEDITOR.TXT looks like this:

```
Del : CTRL+G
```

You could change that command to any other keystroke (CTRL+D, say), as long as the key isn't already assigned to another function. Elsewhere in the file, CTRL+D is assigned to CharRight, so you'd have to delete or change that line if you wanted to use CTRL+D for the Del function.

When you're satisfied with the new functions, you must convert the ASCII file to binary, so that it can be loaded into the QuickC editor:

```
MKKEY -c ab -i MYEDITOR.TXT -o MYEDITOR.KEY
```

Finally, to load the new key file, use the /k: option described above.

Using Another Editor

If you'd prefer to use your favorite word processor or text editor for writing programs, you may use the Utility menu Customize Menu command. This allows you to run any other program (including word processors) from within the QuickC environment. When you exit the program, you'll return to QuickC.

Compiling and Linking

Your ultimate goal in writing C programs is to create an executable program. To convert a C source file to a runnable program, you must compile and link it. This section introduces the commands that compile and link programs, a process called "building."

When QuickC builds a program, it performs two steps:

1. It compiles the .C source file into an object (.OBJ) file.
2. It links the object file with other object files or libraries to create an executable (.EXE) file.

Although you have the option of going through these two steps, it's generally easier and more convenient to build a program with a single command.

Building within the QuickC Environment

In this section, we'll illustrate how to compile and link a C program. First, type in this program:

```
/* HI.C: Prints hello and a name */

#include <stdio.h>

main()
{
   char name[80];

   printf( "Type your name, please.\n" );
   gets( name );
   printf( "Hello, %s\n", name );
}
```

The output of this simple program tells you to type your name. You enter any string of characters, and it says hello to you.

The program calls two library functions: **printf** and **gets**. For more information about what these functions do, use the on-line help system. Position the cursor on the function name and press F1. To close the Help window, press ESC. On-line help is explained in greater detail in the next chapter.

Compiling and Linking

You may notice that the reference bar at the bottom of the screen says `<Shift+F5=Restart>`. To build a program, open the Make menu and choose Build Program. Or use one of these shortcuts:

Press SHIFT+F5 to build the program.

Click the `<Shift+F5=Restart>` button on the reference bar.

A dialog box appears on the screen to show you how far the compiler and linker have progressed. The compiler or linker may halt if anything goes wrong. When the source file contains errors, the Errors window appears and the offending line is highlighted in the Source window.

Running the Program

When the program has been built, the reference bar displays several new items, including `<F5=Run>`. To run the program, open the Run menu and choose Go. Or use the shortcuts:

Press F5.

Click `<F5=Run>` on the reference bar.

Compiling, Linking, and Running

It's not necessary to press SHIFT+F5 (to build the program) before you press F5 (to run it). If you simply press F5, the QuickC editor knows if the source code in memory has changed. If it *has* changed, you'll be asked if you want to rebuild the program.

Viewing the Output Window

When you choose Go from the Run menu (or press F5), the output is automatically directed to the output screen. To see this output:

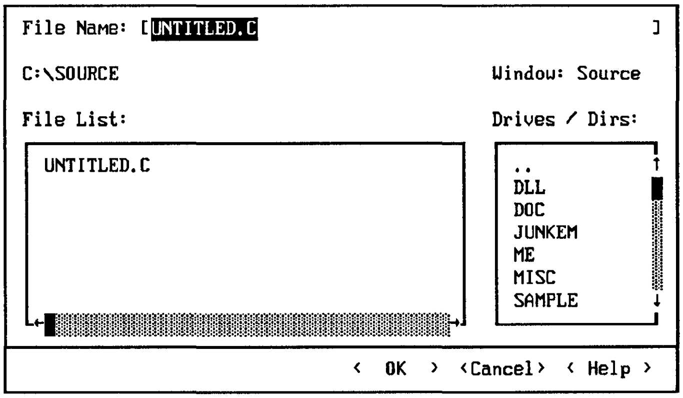
Press F4 to toggle between the two screens.

Open the View window and choose Output Screen. Click once to return to the Source window.

If you'd prefer to have both the Output and Source windows visible at the same time, choose the View menu Windows command and choose Output.

Saving the Program

To save the source file using the current name, go to the File menu and choose either Save or Save As. If you use Save, QuickC automatically saves the file under the name listed on the top line of the Source window. If you prefer to use another name, choose Save As. The following dialog box appears:



```
File Name: [UNTITLED.C                    ]

C:\SOURCE                       Window: Source

File List:                      Drives / Dirs:

 UNTITLED.C                       ..
                                  DLL
                                  DOC
                                  JUNKEM
                                  ME
                                  MISC
                                  SAMPLE

                    <  OK  > <Cancel> < Help >
```

Figure 3.3 Save As... Dialog Box

Type in the new name of the source file. If you don't add an extension, QuickC automatically adds .C (HI becomes HI.C, for example).

Building from Multiple Source Files

Now we'll make things a little more complicated by writing a multiple-module program.

First, edit the HI.C program, adding a few lines:

```
/* HIA.C: Illustrates external functions */

#include <stdio.h>

main()
{
   char name[80];

   welcome();
   outsider();
   printf( "Type your name, please.\n" );
   gets( name );
   printf( "Hello, %s\n", name );
}
welcome()
{
   printf( "Welcome to the program.\n" );
}
```

The `main` function calls two new functions: `welcome` and `outsider.` Unlike `printf` and `gets,` which are library functions, `welcome` and `outsider` are functions defined within the program. Note the definition of `welcome` below the `main` function.

Don't compile the program yet. Save it as HIA.C and then choose New from the File menu. Type in this second program:

```
/* HIB.C: Second program for HIA.C */

outsider()
{
   printf( "This line is from the HIB.C file.\n" );
}
```

Save it as HIB.C. Now there are two source files on your disk: HIA.C and HIB.C.

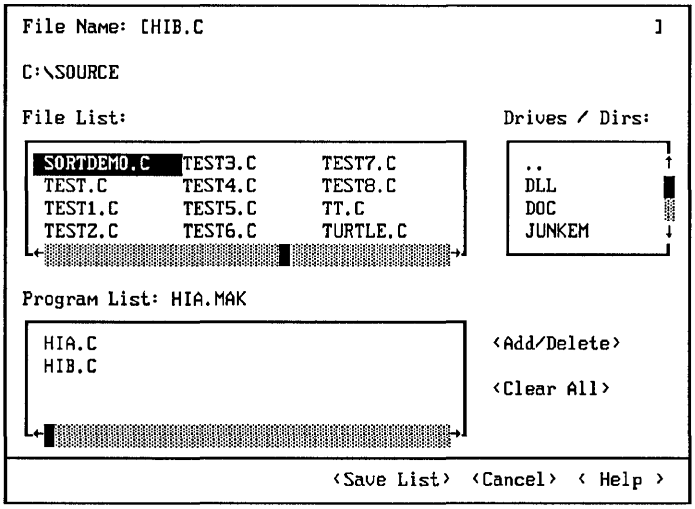
There are several reasons to split a file into multiple modules. If you write a function that's used in several programs, you can give it its own source file. If you write long programs, splitting the file up makes editing easier. In addition, the Build Program command runs faster when it compiles a program if you make changes to one file, but not others. Build doesn't spend time recompiling a source file that hasn't changed.

Creating a Program List

Since the example program now uses two source files, we must create a program list (also called a "make file") that tells QuickC which source files should be compiled.

Open the Make menu and choose Set Program List (you must have Full Menus enabled). Type the name HIA.MAK as the name of the file containing the program list (the MAK extension is used for make files), and press ENTER.

The following dialog box appears:



```
File Name: [HIB.C                                      ]

C:\SOURCE

File List:                                    Drives / Dirs:

 SORTDEMO.C  TEST3.C    TEST7.C                ..
 TEST.C      TEST4.C    TEST8.C                DLL
 TEST1.C     TEST5.C    TT.C                   DOC
 TEST2.C     TEST6.C    TURTLE.C               JUNKEM

Program List: HIA.MAK

 HIA.C                                        <Add/Delete>
 HIB.C
                                              <Clear All>

                            <Save List>  <Cancel>  < Help >
```

Figure 3.4 Creating a Program List

Now you must add both HIA.C and HIB.C to the list of programs.

Press TAB until the cursor is within the list box containing the directory of C source files (another list box lists the directory names). Use the DIRECTION keys to move to the HIA.C file. Press ENTER and the file name will appear in the program list below. Repeat this action to add HIB.C to the program list. (A second option is to type the file names in the text box at the top of the screen.) When you're finished, press TAB until the Save List command button is highlighted, and then press ENTER.

Click HIA.C once and click the Add/Delete button (or just double-click HIA.C). Do the same for HIB.C. When both programs appear in the program list, click the Save List command button to save the make file.

This program list tells QuickC that several source files are to be combined into one program.

NOTE *Within a program list, you can include source files (ending with the .C extension), object files (.OBJ), or libraries (.LIB). For example, if you didn't include graphics in the combined library when you ran the SETUP program, you can place GRAPHICS.LIB in the program list to gain access to graphics functions .*

Notice that the base name of the program list (HIA.MAK) matches the name of one of the source files (HIA.C). Because the names match, when you load HIA.C in the future, QuickC will ask if you wish to use the program list HIA.MAK. You don't have to give the source file and the make file the same names, but it's a good idea to do so.

The order that you list the files is inconsequential. It doesn't matter which file is first or second.

When you choose the Rebuild All command from the Make menu, every .C file in the program list is compiled into a .OBJ file. Then all of the .OBJ files are linked with .LIB files to create one .EXE file.

However, if you press SHIFT+F5 or choose the Build Program command, QuickC checks the time and date stamps on the source and object files. If the source code has not changed since the last time a Build Program command executed, there's no need to recompile the unchanged .OBJ files. Any files that have changed are re-compiled; the others are not. This means the Build command is often faster than the Rebuild All command when you have multiple source files.

Compiling and Linking from the Command Line

If you're new to the C language, you can skip this section. You don't need to know how to compile and link from the command line; you can do everything within the QuickC environment.

However, if you want to, you may exit the QuickC environment to build programs from the DOS prompt. This gives you slightly more control over the various options. In addition, if you're building a series of related executable programs, you can either put the compile and link commands in a batch file or create your own make file. Either method is faster than building programs individually.

The program that builds from the command line is called QCL.EXE (the C and L in the filename stand for "Compile" and "Link"). To build the HI.EXE program, type this:

```
QCL HI.C
```

You may include a variety of command-line options between QCL and the file name. For example,

```
QCL /AM HI.C
```

forces the linker to use the medium memory model (the default is the small model). If you try this example, you must have a medium memory model library installed. You will find a complete list of compiler and linker options in the *Microsoft QuickC Tool Kit* manual (or type `QCL /help`).

QCL can also build multiple-module programs. Since HIA.C and HIB.C are two parts of a multiple-module program, the following line does *not* build a program:

```
QCL HIA.C
```

The compiler works correctly, creating an .OBJ file, but when the linker looks for the `outsider` function (which is in HIB.C), it fails and returns the error `unresolved external.`

One solution is to specify both source files:

```
QCL HIA.C HIB.C
```

The example above builds a program called HIA.EXE, because HIA.C is listed first.

Another solution is to compile the two files and then link them yourself:

```
QCL /c HIA.C
QCL /c HIB.C
LINK HIA.OBJ HIB.OBJ
```

The /c option tells the QCL program to compile but not link. It must be entered as a lowercase character.

Since you created a make file called HIA.MAK, which contained the program list, you may also use the NMAKE program to build HIA.EXE. Simply pass it the name of the make file:

```
NMAKE /F HIA.MAK
```

For a complete list of compiler, linker, and NMAKE options, see *Microsoft QuickC Tool Kit.*

The Debugger

When you make obvious mistakes like misspelling a function name or forgetting to end a line with a semicolon, your code causes a compile-time error. The compiler (or linker) refuses to continue until you fix the mistake.

Other mistakes cause run-time errors. Attempting to divide by zero is one such example.

Still other mistakes are called logic errors. When a program includes a logic error, it may run, but it eventually acts unpredictably or yields incorrect results.

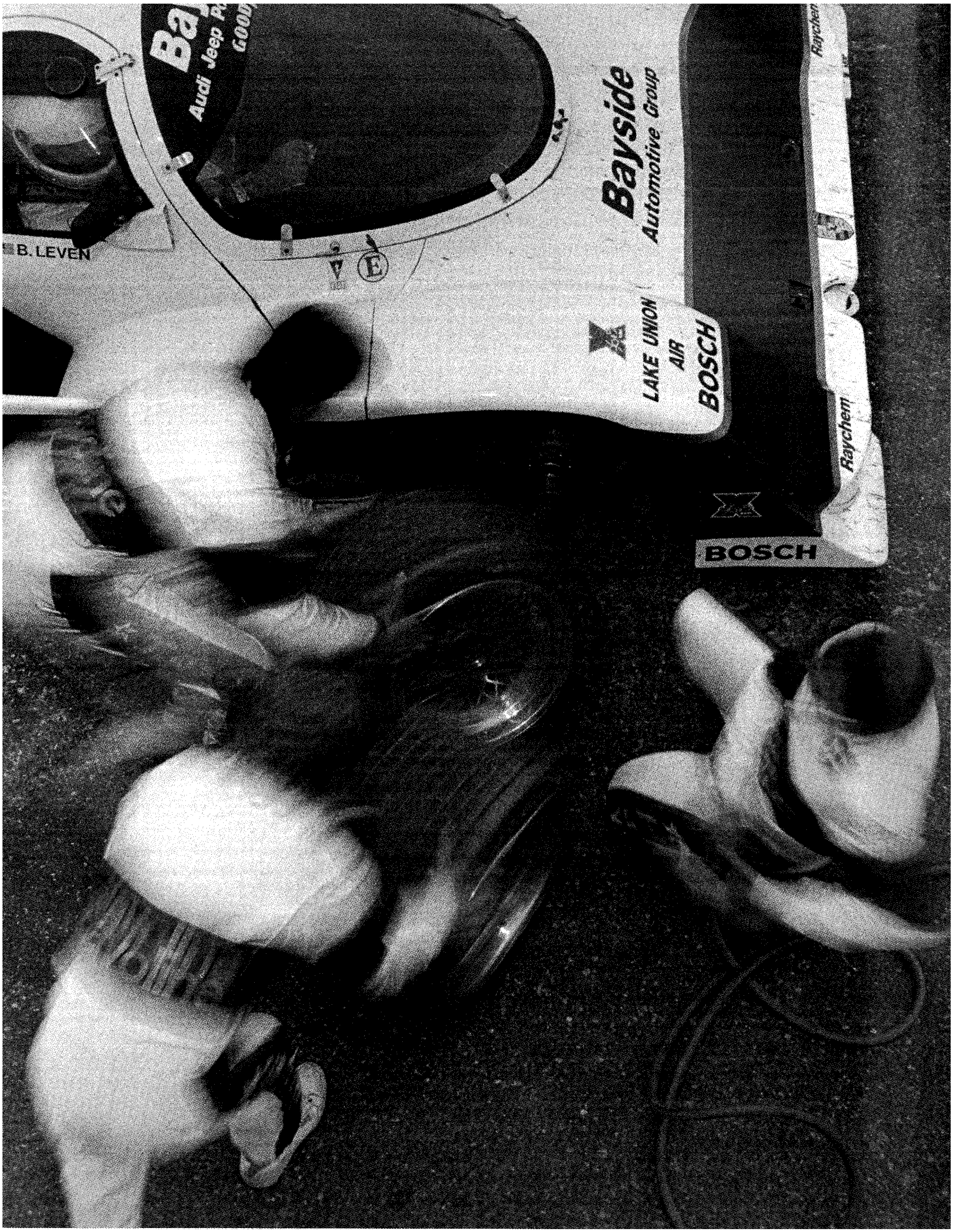
QuickC's built-in debugger helps you track down and correct logic errors. In the Run menu, you'll find Trace Into and Step Over, which execute the program in memory line by line. Trace Into follows functions when they're called; Step Over lets you execute a function without showing its inner workings.

You can use the Debug menu to set watchpoints and breakpoints and run a program up to the breakpoint.

The Watch Value command from the Debug menu is also useful. You enter one or more variable names and then monitor their values as you step through the program. There's no need to place **printf** statements at various points in a program just to monitor the value of a variable.

If you turn on Debugging History, the debugger records everything that happens during the session. Later, you can review the events and watch for the logic error.

The on-line tutorial includes a lesson explaining how to use the debugger. If you're interested in exploring this topic, run the LEARN program.


B. LEVEN
Audi Jeep
Bayside
Automotive Group
LAKE UNION
AIR
BOSCH
Raychem
Raychem
BOSCH

CHAPTER 4 Getting Help

4

The on-line help system provides instant information on all important C topics. You need not thumb through a large reference manual, trying to find a function prototype or return value. Just press a key and the information you need prints instantly on the screen. This chapter explains the many ways to use the QuickC Advisor:

Keyword Help On-line help recognizes C keywords, operators, library functions, and symbolic constants as topics for which it can provide definitions, prototypes, and examples.

Topic-Based Help If you don't know the name of a function or keyword, you can browse through the index of topics or look in the table of contents.

Environment Help If you're not sure what a menu does, you can ask for help about the menu in general or one of the menu commands in particular. In addition, whenever a dialog box appears and you want more information on the choices, press F1.

Error Help When the compiler returns an error message, QuickC highlights the incorrect source line. If you're not sure what's wrong, on-line help will provide more information about the error message.

Help on Help The help system can also provide additional information about using the help system itself.

Not only are the help screens a great learning tool for programmers who are new to the C language, they're also a fast and useful reference database for expert C programmers.

Keyword Help

You may ask for help about any keyword, operator, symbolic constant, or library function. This section explains how to open a help screen, using **printf** as an example.

First, run QuickC and type `printf`.

Here's how you ask for help about the **printf** function:

Position the cursor anywhere on `printf` (or on the space just after the word) and open the Help menu. Choose the `Topic: printf` command. Note that the word under the cursor is always inserted after `Topic` in the Help menu.

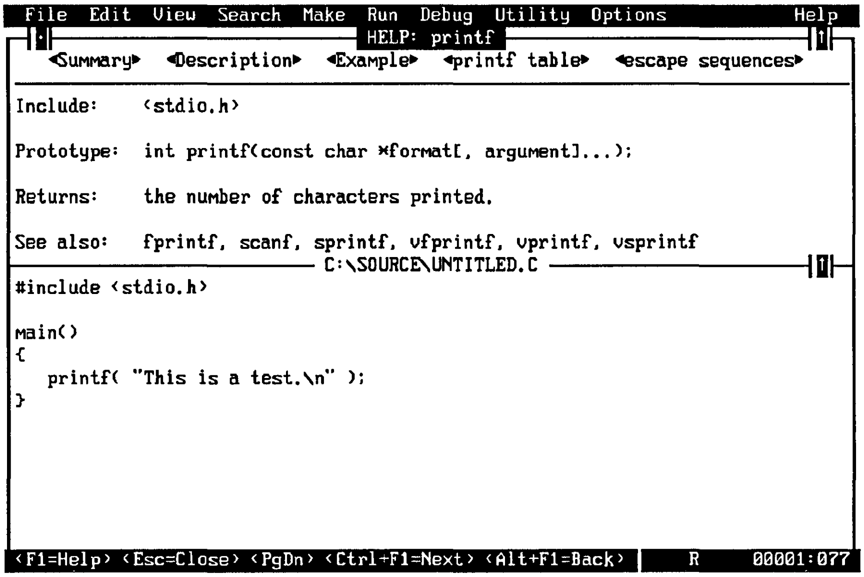
You may prefer to use the following shortcuts:

Position the cursor somewhere on `printf` and press the F1 key.

Move the mouse cursor to `printf` and click the right mouse button.

NOTE *If nothing happens when you press the right mouse button, open the Options menu, choose Display, and click Context Sensitive Help under the Right Mouse Button heading. Exit the dialog box and click the right mouse button on* `printf` *again.*

Whether you use the menu, the F1 key, or the right mouse button, the help screen that describes **printf** instantly appears:


File Edit View Search Make Run Debug Utility Options Help
HELP: printf
◄Summary► ◄Description► ◄Example► ◄printf table► ◄escape sequences►
C:\SOURCE\UNTITLED.C
<F1=Help> <Esc=Close> <PgDn> <Ctrl+F1=Next> <Alt+F1=Back>
R 00001:077

```
Include:    <stdio.h>

Prototype:  int printf(const char *format[, argument]...);

Returns:    the number of characters printed.

See also:   fprintf, scanf, sprintf, vfprintf, vprintf, vsprintf
```

```
#include <stdio.h>

main()
{
   printf( "This is a test.\n" );
}
```

Figure 4.1 Help on printf

Topics and Hyperlinks

All keywords, operators, and library functions are called "topics" within the help system. The example above activated the help screen for the **printf** topic.

Within the Help window describing **printf**, you'll see certain highlighted words and phrases at the top of the screen. These are called "hyperlinks." They link the current help screen to additional related topics.

The difference between a topic and a hyperlink is simple. You may ask for help on any topic at any time (pressing F1, clicking the right mouse button, or using the Help menu). *You may even ask for help from within the Help window.*

However, if you select a topic that doesn't exist—"elephant," for example—the help system looks through the list of available topics to see if it can offer help. If it can't, no help screen appears.

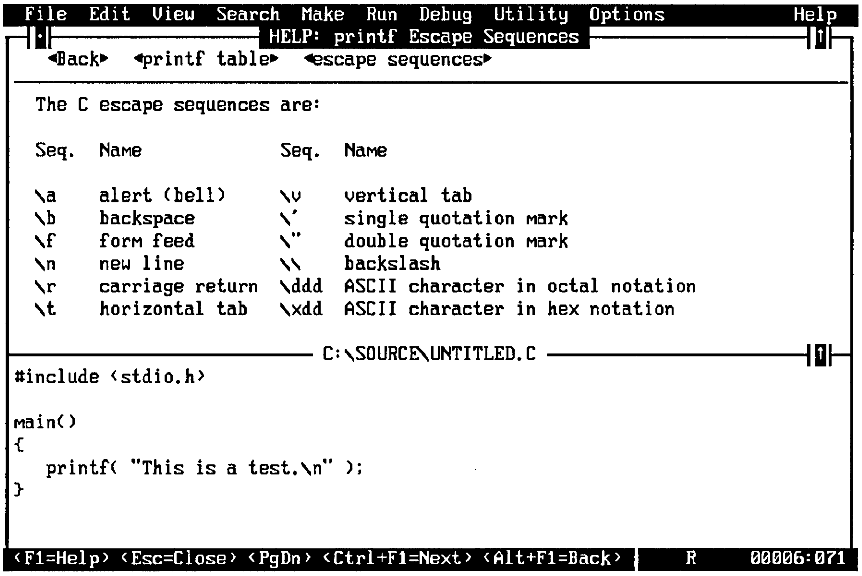
Hyperlinks, on the other hand, *always* lead to a related help screen. You activate hyperlinks the same way you ask for help about a topic: press F1 or click the right mouse button.

For example, here's how you activate the hyperlink labeled `escape sequences` from the **printf** Help window:

With the **printf** screen still visible, use the TAB key to position the cursor on the hyperlink. Press F1. (If the Source window is active, press F6 first to move to the Help window.)

Move the mouse cursor to the hyperlink and click the right button.

The help screen on escape sequences appears below:


File Edit View Search Make Run Debug Utility Options Help
HELP: printf Escape Sequences
◂Back▸ ◂printf table▸ ◂escape sequences▸

```
The C escape sequences are:

Seq.  Name             Seq.  Name

\a    alert (bell)     \v    vertical tab
\b    backspace        \'    single quotation mark
\f    form feed        \"    double quotation mark
\n    new line         \\    backslash
\r    carriage return  \ddd  ASCII character in octal notation
\t    horizontal tab   \xdd  ASCII character in hex notation
```

C:\SOURCE\UNTITLED.C

```
#include <stdio.h>

main()
{
   printf( "This is a test.\n" );
}
```

Figure 4.2 Escape Sequences

The newline character is described as `\n`. Suppose you want to add a new line to the end of the string in the program. There are two ways to add `\n` to the program:

1. Return to the Source window by pressing F6 (keyboard) or clicking once in the Source window (mouse). The Help window remains visible. Type the characters you wish to add to the source file.

2. Copy the items you want to use from the Help window. Paste them into the source file. This option is described in greater detail below.

Copy and Paste from Help

Any text that appears in the Help window can be copied quite easily to the Source window (or to the Notepad window).

This feature of the on-line help system is very useful. If you'd like to test an example program from the Help window, you can copy it to the Source window and compile it. If you need to copy a few **#include** directives, you can do it quickly.

Just follow the steps below.

1. Move the cursor to the beginning or the end of the text you want to select. Hold down the SHIFT key and move the cursor to the other end of the text. The text is now highlighted.
2. Execute the Copy command. Press ALT to activate the menus, E for Edit, C for Copy. Note that the menu lists the shortcut command (CTRL+INS), which may be substituted for ALT, E, C.
3. Press F6 to switch to the Source window. Position the cursor where you want to insert the text and execute the Paste command: ALT, E, P. Or use the SHIFT+INS shortcut. The text from the Help window is automatically inserted at the current cursor position.

Or:

1. Click and drag the cursor (hold down the left button while moving the mouse) to select the text you want to copy.
2. Choose Copy from the Edit menu (or press CTRL+INS).
3. Click once in the Source window to activate it, move the mouse cursor to the location where you want to insert the text, and click once. Then select Paste from the Edit menu. The text from the Help window is automatically inserted at the current cursor position in the program.

Viewing the Previous Help Screen

QuickC remembers the last 20 help screens you've accessed. Returning to one of the previous screens is easy. Hold down the ALT key and press F1 as many times as necessary to return to the screen you want to see. For example, if you want to see the third screen back, press and release ALT+F1 three times. The help screen you see is active; you can ask for help on any of its hyperlinks or topics.

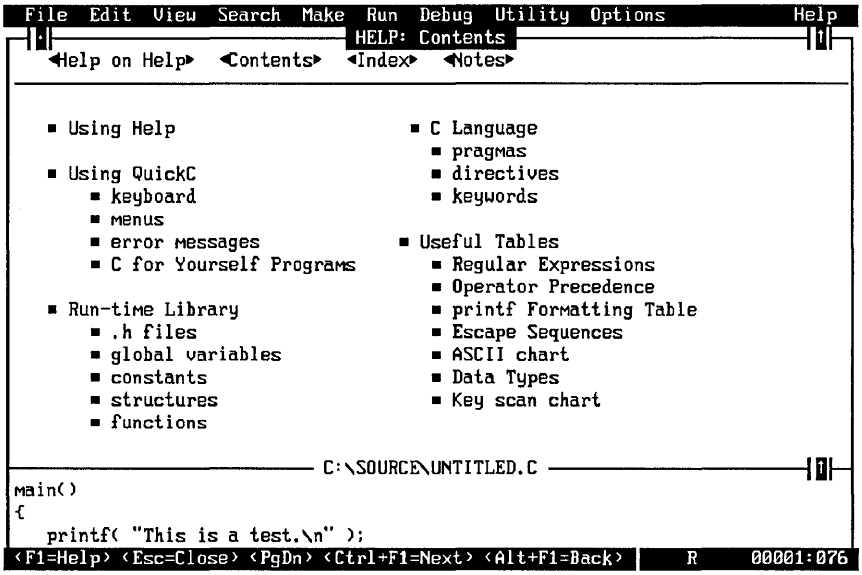
Topic-Based Help

The on-line help system also includes a table of contents for help topics, which comes in handy in those situations when you have only a general idea of what you need. To browse through the help system, go to the Help menu and select Contents.

Press ALT, H, C.

Click once on the Help menu. Choose Contents.

Below is the Contents screen that appears:



```
 File  Edit  View  Search  Make  Run  Debug  Utility  Options          Help
                              HELP: Contents
  ◄Help on Help►  ◄Contents►  ◄Index►  ◄Notes►

  ▪ Using Help                      ▪ C Language
                                      ▪ pragmas
  ▪ Using QuickC                      ▪ directives
      ▪ keyboard                      ▪ keywords
      ▪ menus
      ▪ error messages             ▪ Useful Tables
      ▪ C for Yourself Programs       ▪ Regular Expressions
                                      ▪ Operator Precedence
  ▪ Run-time Library                  ▪ printf Formatting Table
      ▪ .h files                      ▪ Escape Sequences
      ▪ global variables              ▪ ASCII chart
      ▪ constants                     ▪ Data Types
      ▪ structures                    ▪ Key scan chart
      ▪ functions

                    C:\SOURCE\UNTITLED.C
main()
{
   printf( "This is a test.\n" );
<F1=Help> <Esc=Close> <PgDn> <Ctrl+F1=Next> <Alt+F1=Back>    R    00001:076
```

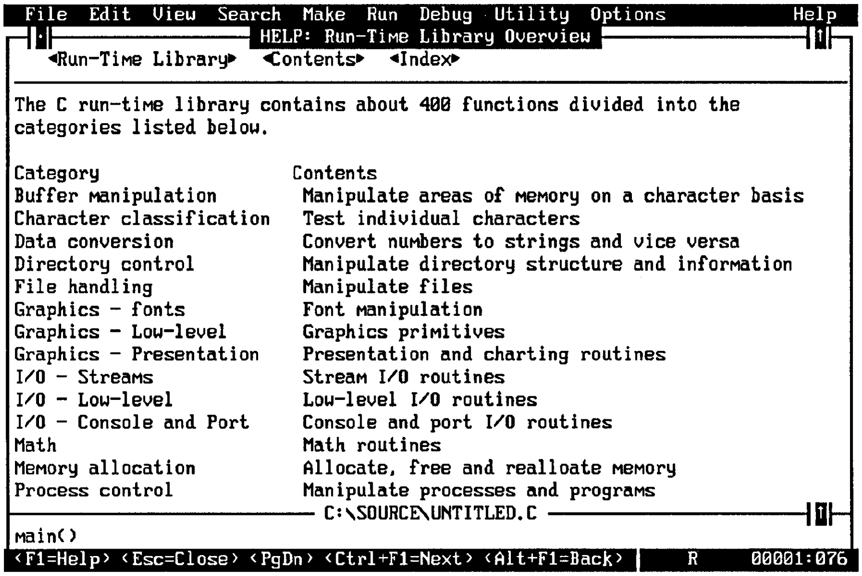
Figure 4.3 On-Line Help Contents Screen

Suppose you want more information about the run-time library.

Move the cursor to the Run-time Library topic and press F1.

Click the right button on the Run-time Library topic.

The following screen appears:


File Edit View Search Make Run Debug Utility Options Help
HELP: Run-Time Library Overview
◀Run-Time Library▶ ◀Contents▶ ◀Index▶

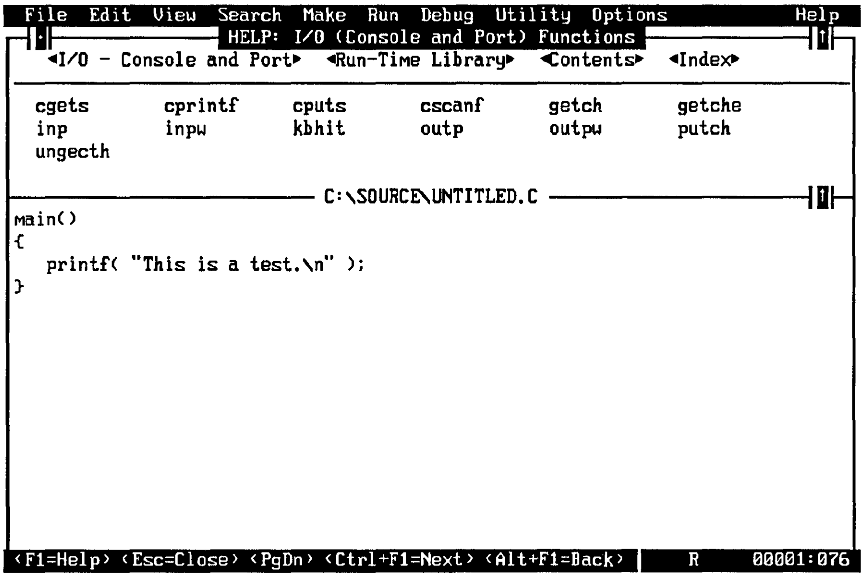
```
The C run-time library contains about 400 functions divided into the
categories listed below.

Category                  Contents
Buffer manipulation        Manipulate areas of memory on a character basis
Character classification   Test individual characters
Data conversion            Convert numbers to strings and vice versa
Directory control          Manipulate directory structure and information
File handling              Manipulate files
Graphics - fonts           Font manipulation
Graphics - Low-level       Graphics primitives
Graphics - Presentation    Presentation and charting routines
I/O - Streams              Stream I/O routines
I/O - Low-level            Low-level I/O routines
I/O - Console and Port     Console and port I/O routines
Math                       Math routines
Memory allocation          Allocate, free and realloate memory
Process control            Manipulate processes and programs
```

C:\SOURCE\UNTITLED.C
main()
<F1=Help> <Esc=Close> <PgDn> <Ctrl+F1=Next> <Alt+F1=Back> R 00001:076

Figure 4.4 Library Categories

Suppose that you're looking for a function that waits for a keypress. One of the topics is I/O–Console and Port, which sounds like the right area to explore.

When you choose I/O–Console and Port, the following screen appears:

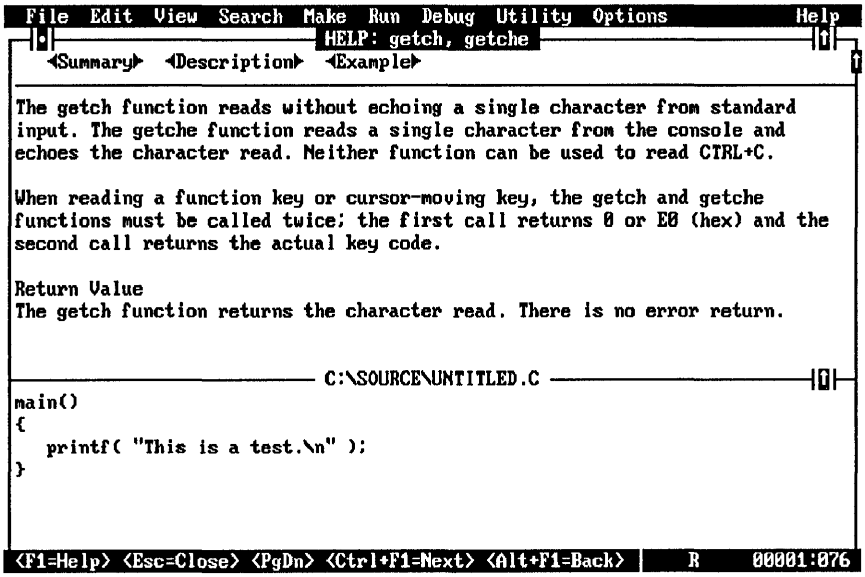


```
 File  Edit  View  Search  Make  Run  Debug  Utility  Options             Help
 HELP: I/O (Console and Port) Functions
   ◄I/O - Console and Port►  ◄Run-Time Library►  ◄Contents►  ◄Index►

   cgets       cprintf     cputs       cscanf      getch       getche
   inp         inpw        kbhit       outp        outpw       putch
   ungecth

                              C:\SOURCE\UNTITLED.C
main()
{
   printf( "This is a test.\n" );
}

<F1=Help> <Esc=Close> <PgDn> <Ctrl+F1=Next> <Alt+F1=Back>      R     00001:076
```

Figure 4.5 I/O Console and Port Functions

The function called **getch** sounds promising. Ask for help about the function by positioning the cursor on `getch` and pressing F1 (keyboard) or by clicking the right mouse button on `getch` (mouse).

To read about **getch**, tab over to `Description`. The following help screen appears:


File Edit View Search Make Run Debug Utility Options Help
HELP: getch, getche
◂Summary▸ ◂Description▸ ◂Example▸
C:\SOURCE\UNTITLED.C
<F1=Help> <Esc=Close> <PgDn> <Ctrl+F1=Next> <Alt+F1=Back> R 00001:076

```
The getch function reads without echoing a single character from standard
input. The getche function reads a single character from the console and
echoes the character read. Neither function can be used to read CTRL+C.

When reading a function key or cursor-moving key, the getch and getche
functions must be called twice; the first call returns 0 or E0 (hex) and the
second call returns the actual key code.

Return Value
The getch function returns the character read. There is no error return.
```

```
main()
{
   printf( "This is a test.\n" );
}
```

Figure 4.6 Description of getch

When you started, you didn't know the function name, but you tracked it down through the table of contents.

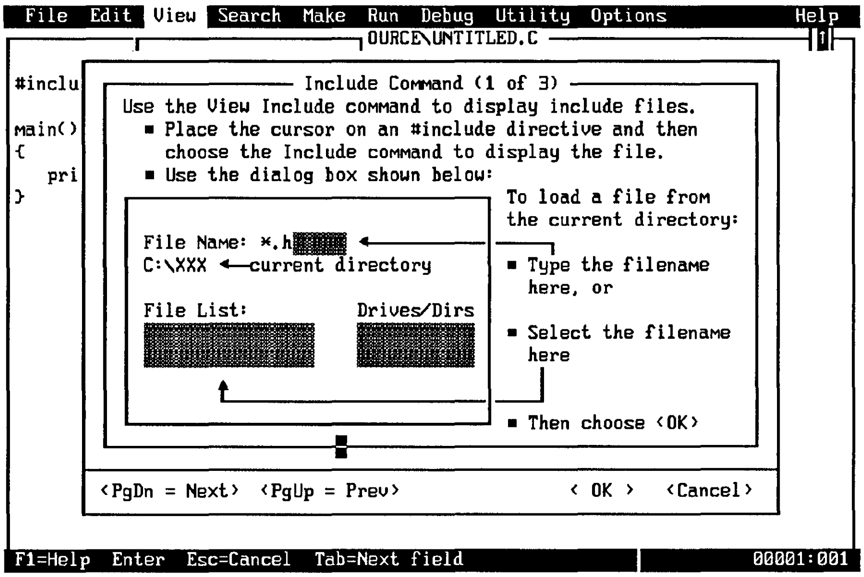
Environment Help

Open the View menu. (Be sure you have Full Menus turned on.) Suppose you notice that it has a command called Include..., but you are not sure what it does. Highlight—but don't choose—Include... :

Press ALT, then V (for the View menu), and use the DOWN direction key until the Include... command is highlighted. Press F1 for help.

Click once with the left mouse button to open the View menu. Use the DOWN key to highlight Include... and press F1.

The following screen appears:



```
 File  Edit  View  Search  Make  Run  Debug  Utility  Options        Help
                    OURCE\UNTITLED.C
#inclu
       ──────── Include Command (1 of 3) ────────
main()  Use the View Include command to display include files.
{         ■ Place the cursor on an #include directive and then
   pri      choose the Include command to display the file.
}         ■ Use the dialog box shown below:
                                                To load a file from
                                                the current directory:
           File Name: *.h       ←
           C:\XXX ←current directory            ■ Type the filename
                                                  here, or
           File List:           Drives/Dirs
                                                ■ Select the filename
                                                  here

                                                ■ Then choose <OK>

        <PgDn = Next>  <PgUp = Prev>              < OK >   <Cancel>
 F1=Help  Enter  Esc=Cancel  Tab=Next field                  00001:001
```

Figure 4.7 Help on the Include Command

Using the methods described above, you can ask for help about any of the menus or menu options.

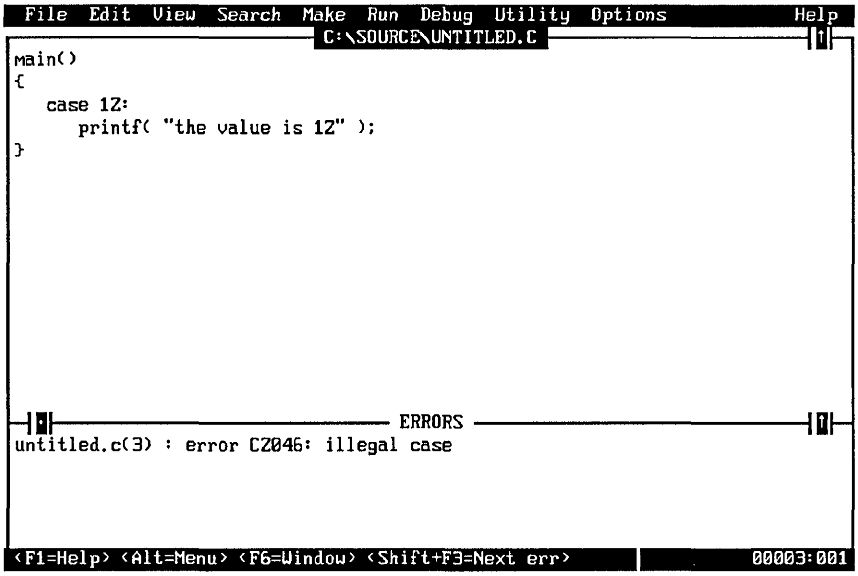
Error Help

The fourth type of help provides information about compiler and linker error messages.

For example, suppose you try to build this program:

```
main()
{
   case 12:
      printf( "The value is 12" );
}
```

The following Error window appears at the bottom of the screen:


File Edit View Search Make Run Debug Utility Options Help
C:\SOURCE\UNTITLED.C
ERRORS
<F1=Help> <Alt=Menu> <F6=Window> <Shift+F3=Next err>
00003:001

```
main()
{
   case 12:
      printf( "the value is 12" );
}
```

```
untitled.c(3) : error C2046: illegal case
```

Figure 4.8 Error Window

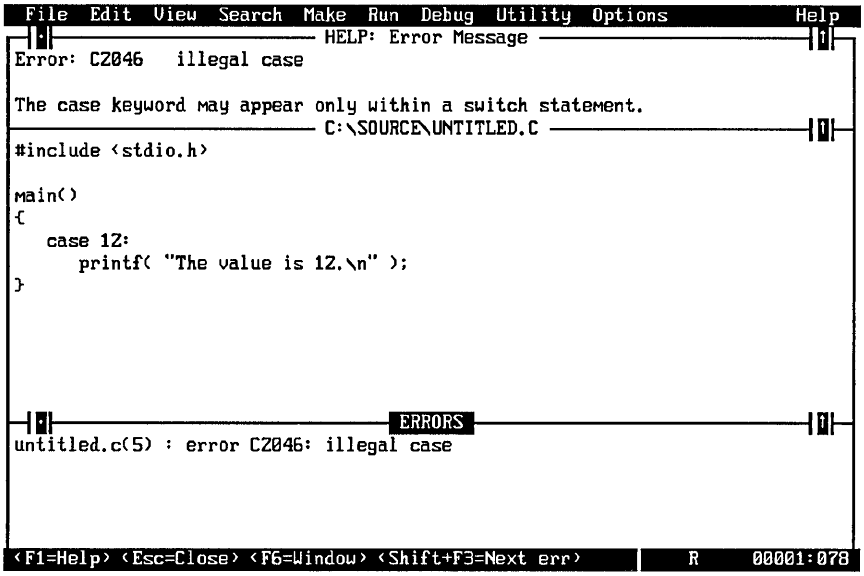
The error message says `C2046: illegal case`, which you might not understand.

Move to the Errors window by opening the View menu and choosing Windows, then choosing Errors. Or use the shortcut keystrokes:

Press F6 to change windows, then press F1 for help.

Position the mouse cursor anywhere on the error message and click the right mouse button.

The Help window pictured below opens:


File Edit View Search Make Run Debug Utility Options Help
HELP: Error Message
C:\SOURCE\UNTITLED.C
ERRORS
<F1=Help> <Esc=Close> <F6=Window> <Shift+F3=Next err>
R 00001:078

```
Error: CZ046   illegal case

The case keyword may appear only within a switch statement.
```

```
#include <stdio.h>

main()
{
   case 12:
      printf( "The value is 12.\n" );
}
```

```
untitled.c(5) : error CZ046: illegal case
```

Figure 4.9 Help on Errors

As you can see, the **case** keyword requires a **switch** statement. If you don't know what a **switch** statement is, ask for help about **switch**:

Move the cursor to the word `switch` and press F1.

Click the right button on the word `switch`.

Help on Help

For quick reference to the available help functions, open the Help menu and choose Help on Help. Or use the shortcut key SHIFT+F1.

Help on Help describes briefly how to use the help system. From within the Help on Help screen, you can use hyperlinks to see the Contents and Index. Those screens can guide you to other subjects, including ASCII tables, lists of C keywords, lists of pragmas, and much more.

Havoline
Havoline
MOTOR OIL
B. LEVEN

CHAPTER 5

Where to Go from Here

5

You've installed QuickC on your system and tried a sample editing session. What you do next depends on your level of experience. You can experiment, or you can further investigate some of the other books in this package.

Read README.DOC

The README.DOC file lists all known corrections and additions to the printed manuals. Before you continue, please read this file. You can view it from within on-line help or you can use the QuickC editor (type `QC README.DOC` to read the file).

For All Programmers

The QuickC environment has been designed with you, the programmer, in mind. You'll find writing, compiling, and debugging QuickC programs faster and easier than ever. To learn more about the editor, compiler, linker, and debugger, run the disk-based tutorial on the QuickC distribution disk labeled "Learning the Microsoft QuickC Integrated Environment."

Place your copy of the disk in your disk drive, type `LEARN`, and press ENTER. The LEARN program contains four lessons: How to Use This Tutorial, Getting Around in QuickC, Creating Programs in QuickC, and Debugging in QuickC.

For Programmers New to the C Language

We wrote the book *C for Yourself* for people who already know how to program (in BASIC, Pascal, or some other language) but have never used C. Part 1 covers everything from functions to flow control and from data types to pointers. If you've never programmed in C or if you need a refresher course, start with *C for Yourself.*

The introduction in *C for Yourself* lists several additional books that newcomers to C may find helpful.

Another good source of information about how C works is the on-line help system. Prototypes, explanations, and examples for any function are immediately available at the press of a key. You may ask for help about a specific topic or browse through the table of contents.

For C Programmers New to QuickC

Appendix A of *C for Yourself* is a guide to the QuickC implementation of C. Both the proposed ANSI standard and the original Kernighan & Ritchie standard are supported.

Appendix B of *C for Yourself* summarizes the most useful QuickC library functions, listing the header files to include, values to pass, and values returned.

For specific details on the compiling, linking, library, and other support programs, see the *Microsoft QuickC Tool Kit* manual.

For QuickC Programmers New to Version 2.0

Advanced C topics and features unique to QuickC are covered in *C for Yourself.* Part 2 includes detailed information about the new functions such as real coordinate graphics, presentation graphics, fonts, and in-line assembly. If you've previously used Version 1.0 of QuickC, we recommend that you read through Part 2 of *C for Yourself.* In addition, Appendix B summarizes commonly used functions in the QuickC run-time library.

Additional Tools

QuickC offers more than just an integrated editor, compiler, and linker. Additional utilities include the QCL program (for compiling from the DOS command line), the LINK program (for linking object modules), the NMAKE program (for automating the compile/link process and maintaining programs), and the LIB program (for creating your own libraries).

The *Microsoft QuickC Tool Kit* manual explains how to use these advanced tools to best advantage. Part 1 is a tutorial that explains step-by-step how these programs work. Part 2 is a complete and exhaustive reference guide that summarizes the many options.

Appendix

Editor Functions

This appendix has two parts. The first part, Table A.1, lists the keystrokes that invoke commands within the QuickC Editor and three other editors. The second part, Table A.2, lists the editor functions alphabetically and defines them.

Each editor has its own .KEY file. Chapter 3, "Using QuickC," explains how to use the /k: option to load one of the four .KEY files supplied with QuickC. It also contains directions for using MKKEY.EXE to customize the commands within a .KEY file. Functions are not assigned default values. If you omit Backspace from your customized .KEY file, you will be unable to backspace.

Note that in Table A.1 below, the plus sign indicates that both keys should be held down. For example, CTRL+H means that you hold down the CTRL key while you press H. When keys are pressed separately, they are separated with a comma. For example, ESC, D means that you press (and release) ESC and then press the D key.

The CAPS LOCK key affects individual keystrokes, but not CTRL or ALT sequences. Holding down the SHIFT key defines a block of text.

Table A.1 Editor Keystrokes

Function	QuickC	Microsoft	BRIEF®	Epsilon
Backspace	CTRL+H	CTRL+H	CTRL+H	—
BegLine	CTRL+Q, CTRL+S CTRL+Q, S	—	—	—
BegPgm	CTRL+HOME CTRL+Q, CTRL+R CTRL+Q, R	ALT+A, PGUP	CTRL+PGUP	CTRL+HOME ESC, <
Cancel	ESC	ESC	ESC	F12
Cancel2	F12	F12	F12	ESC
Change	CTRL+Q, CTRL+A CTRL+Q, A	CTRL+L CTRL+\	ALT+T	ESC, %
CharLeft	CTRL+S LEFT	CTRL+S	LEFT	CTRL+B LEFT
CharRight	CTRL+D RIGHT	CTRL+D	RIGHT	CTRL+F RIGHT

Table A.1 *(continued)*

Function	QuickC	Microsoft	BRIEF	Epsilon
Del	DEL CTRL+G	DEL CTRL+G	DEL	DEL CTRL+D
DelWord	CTRL+T			ALT+D ESC, D
DoEsc	ESC	—	—	—
DoQuote Character	CTRL+P	CTRL+P	ALT+Q	CTRL+Q
DoTab	TAB ALT+TAB	TAB	TAB	TAB ALT+TAB
EndLine	END CTRL+Q, CTRL+D CTRL+Q, D	END	END	END CTRL+E
EndPgm	CTRL+END CTRL+Q, CTRL+C CTRL+Q, C	ALT+A, PGDN	CTRL+PGDN	CTRL+END ESC, >
EndScn	CTRL+Q, CTRL+X CTRL+Q, X	ALT+A, DOWN	CTRL+DOWN	—
EraseEol	CTRL+Q, CTRL+Y CTRL+Q, Y	ALT+A, CTRL+Y	ALT+K	CTRL+K
Find	CTRL+Q, CTRL+F CTRL+Q, F	—	ALT+S	CTRL+S
GotoBookmark0	CTRL+Q, 0	—	—	—
GotoBookmark1	CTRL+Q, 1	—	—	—
GotoBookmark2	CTRL+Q, 2	—	—	—
GotoBookmark3	CTRL+Q, 3	—	—	—
HomeLine	HOME	HOME	HOME	HOME CTRL+A
HomeScn	CTRL+Q, CTRL+E CTRL+Q, E	ALT+A, UP	CTRL+HOME	—
KillLine	CTRL+Y	CTRL+Y	ALT+D	—
LineDown	DOWN CTRL+X	DOWN CTRL+X	DOWN	DOWN CTRL+N
LineUp	UP CTRL+E	UP CTRL+E	UP	UP CTRL+P

Table A.1 *(continued)*

Function	QuickC	Microsoft	BRIEF	Epsilon
MatchBrace	CTRL+]	—	—	CTRL+]
Menu	ALT	F 11	ALT	ALT
Menu2	F 11	ALT+M	F 11	F 11
NewLine	CTRL+M	CTRL+M	CTRL+M	CTRL+M
NextLine	CTRL+J	—	—	CTRL+J
PageDown	PGDN CTRL+C	PGDN CTRL+C	PGDN	PGDN CTRL+V
PageLeft	CTRL+PGUP	—	—	CTRL+PGUP
PageRight	CTRL+PGDN	—	—	CTRL+PGDN
PageUp	PGUP CTRL+R	PGUP CTRL+R	PGUP	PGUP ALT+V ESC, V
ResetState	CTRL+K, CTRL+U CTRL+Q, CTRL+U CTRL+U	—	—	—
ScrollDown	CTRL+Z CTRL+DOWN	CTRL+Z	CTRL+D	CTRL+Z CTRL+DOWN
ScrollUp	CTRL+UP CTRL+W	CTRL+W	CTRL+U	CTRL+UP ESC, Z F4
SearchNext	CTRL+L	—	—	—
SetBookMark0	CTRL+K, 0	—	ALT+M	—
SetBookMark1	CTRL+K, 1	—	—	—
SetBookMark2	CTRL+K, 2	—	—	—
SetBookMark3	CTRL+K, 3	—	—	—
SplitLine	CTRL+N	CTRL+N	—	CTRL+O
ToggleInsertMode	INS CTRL+V	INS CTRL+V	ALT+I	INS
Undo	CTRL+Q, CTRL+L CTRL+Q, L	ALT+H	ALT+U	—
WordLeft	CTRL+LEFT CTRL+A	CTRL+LEFT CTRL+A	CTRL+LEFT	CTRL+LEFT ALT+B ESC, B
WordRight	CTRL+RIGHT CTRL+F	CTRL+RIGHT CTRL+F	CTRL+RIGHT	CTRL+RIGHT ALT+F ESC, F

Table A.2 Function Definitions

Function	Description
Backspace	Moves the cursor left and erases the character in that position.
Beep	Causes the computer to beep, usually when an unassigned key is pressed.
BegLine	Moves the cursor to the beginning of the line (column 1), ignoring any indentation.
BegPgm	Moves the cursor to the beginning of the program (line 1, column 1).
Cancel	Closes Help windows and cancels dialog boxes.
Cancel2	Cancels dialog boxes.
Change	Searches for selected text and changes it to something else.
CharLeft	Moves the cursor one character left.
CharRight	Moves the cursor one character right.
Del	Deletes the character under the cursor.
DelWord	Deletes the word from the current cursor position to the next white-space character.
DoEsc	Cancels the current selected text.
DoQuoteCharacter	Inserts the next typed character into the text (except carriage returns, linefeeds, and nulls). This allows you to insert characters such as CTRL+C.
DoTab	Indents the entire defined block by one tab setting. A shifted DoTab removes the tabs from a block.
EndLine	Moves the cursor to the last nonspace character in the current line.
EndPgm	Moves the cursor to column 1 of the last line in the current file.
EndScn	Moves the cursor to the bottom line of the current screen. The current column is maintained.
EraseEol	Erases all characters from the cursor position to the end of the line. The characters are placed in the insert buffer.
Find	Searches from the current cursor position to the end of the file for defined text.
GotoBookmark0	Moves the cursor to bookmark 0.
GotoBookmark1	Moves the cursor to bookmark 1.
GotoBookmark2	Moves the cursor to bookmark 2.
GotoBookmark3	Moves the cursor to bookmark 3.

Table A.2 *(continued)*

Function	Description
HomeLine	Moves the cursor to the first nonspace character on the line.
HomeScn	Moves the cursor to line 1 of the current screen, maintaining the current column position.
IgnoreChar	Does nothing. In addition, the editor doesn't beep as it would for undefined keys.
KillLine	Erases everything on the current line and places the line in the insert buffer.
LineDown	Moves the cursor to the next line.
LineUp	Moves the cursor to the previous line.
MatchBrace	Finds matching (parentheses), [square brackets], or {curly braces}. The editor searches for opening or closing braces, depending on which type the cursor is resting on.
Menu	Activates the menu bar.
Menu2	Activates the menu bar.
NewLine	Splits the current line into two lines at the cursor position. The new line is automatically indented to match the line above.
PageDown	Moves forward one screen.
PageLeft	Moves left one screen.
PageRight	Moves one screen to the right (each line on a screen may contain up to 255 characters).
PageUp	Moves back one screen.
ResetState	Cancels prefix tables.
ScrollDown	Scrolls down one line at a time, maintaining the current cursor position.
ScrollUp	Scrolls the text up one line at a time, maintaining the current cursor position.
SearchNext	Repeats the Find command.
SetBookMark0	Sets bookmark 0. You may later return to this position with GotoBookMark0.
SetBookMark1	Sets bookmark 1.
SetBookMark2	Sets bookmark 2.
SetBookMark3	Sets bookmark 3.
SplitLine	Splits a line in two (like NewLine), but leaves the cursor at the end of the first line instead of the beginning of the second.

Table A.2 *(continued)*

Function	Description
ToggleInsertMode	Toggles between insert mode and overtype mode.
Undo	Cancels any commands that might have been performed on the current line. Undo works only when the cursor remains on the line being edited.
WordLeft	Moves the cursor to the previous word.
WordRight	Moves the cursor to the next word.

Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717

Microsoft®

1088 Part No. 04321