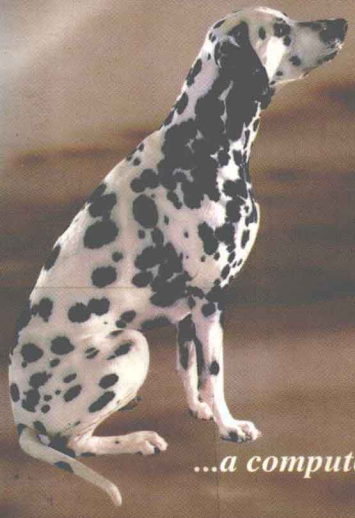
JAMSA'S C/C++ PROGRAMMER'S BIBLE

The Ultimate Guide to C/C++ Programming

By Kris Jamsa, Ph.D.
and Lars Klander



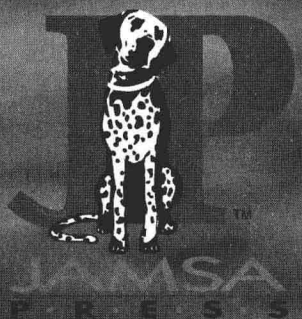
...a computer user's best friend®

Companion CD-ROM includes
Borland Turbo C++ Lite—
Everything you need to
create C++ programs!

JAMSA'S C/C++ PROGRAMMER'S BIBLE

The Ultimate Guide to C/C++ Programming

By Kris Jamsa, Ph.D.
and Lars Klander



Published by
Jamsa Press
2975 S. Rainbow Blvd., Suite I
Las Vegas, NV 89102
U.S.A.

http://www.jamsapress.com

For information about the translation or distribution of any Jamsa Press book, please write to Jamsa Press at the address listed above.

Jamsa's C/C++ Programmer's Bible

Printed in the United States of America.
987

ISBN 1-884133-25-8

Publisher
Debbie Jamsa

Content Manager
Todd Peterson

Composition
Eugene Marks
James Rehrauer
Nelson Yee

Proofers
Rosemary Pasco
Jeanne K. Smith

Technical Advisor
Phil Schmauder

Cover Photograph
O'Gara/Bissell

Illustrators
Eugene Marks
James Rehrauer
Nelson Yee

Indexer
John Bianchi

Director of Publishing Operations
Janet Lawrie

Cover Design
Marianne Helm
James Rehrauer

Copy Editors
Ann Edwards
Dorothy Oppenheimer
Renée Wesberry

Technical Editors
C.J. Bockmann
LingYan Tang

Jamsa Press is an imprint of Gulf Publishing Company:

Gulf Publishing Company
Book Division
P.O. Box 2608
Houston, TX 77252-2608
U.S.A.

http://www.gulfpub.com

Table of Contents

GETTING STARTED WITH C

MACROS AND CONSTANTS

UNDERSTANDING STRINGS

FUNCTIONS

KEYBOARD OPERATIONS

MATH

FILES, DIRECTORIES, AND DISKS

ARRAYS, POINTERS, AND STRUCTURES

DOS AND BIOS SERVICES

MEMORY MANAGEMENT

DATE AND TIME

REDIRECTING I/O AND PROCESSING COMMAND-LINES

PROGRAMMING TOOLS

ADVANCED C

GETTING STARTED WITH C++

OBJECTS

COMMON CLASS FUNCTIONS

I/O WITH C++

INHERITANCE AND POLYMORPHISM

GENERIC FUNCTIONS AND TEMPLATES

EXCEPTION HANDLING AND TYPE PORTABILITY

CREATING SAMPLE REUSABLE CLASSES

STANDARD TEMPLATE LIBRARY

GETTING STARTED WITH WINDOWS PROGRAMMING

MESSAGES AND MENUS

DIALOG BOXES

WINDOWS MEMORY MANAGEMENT

PROCESSES AND THREADS

THE GRAPHICS DEVICE INTERFACE

XIV

BITMAPS, METAFILES, AND ICONS

WINDOWS I/O

AN INTRODUCTION TO PROGRAMMING 1

Computer programs, also known as *software*, are made up of a series of instructions that the computer executes. When you create a program, you must specify the instructions that the computer must execute to perform the desired operations. The process of defining the instructions the computer is to execute is known as *programming*. When you create a program, you store the instructions in an ASCII file whose name usually contains the extension C for a C program and CPP for a C++ program. For example, if you create a C program that performs payroll operations, you might name the file containing the program instructions *payroll.c*. When you create programs, you specify the desired instructions using a *programming language*. C and C++ are only two of many programming languages. Many programmers use programming languages such as BASIC, Pascal, and FORTRAN. Different programming languages provide unique features and have their own strengths (and weaknesses). In any case, programming languages exist to let us define the instructions that we want the computer to execute.

The instructions a computer executes are actually sets of 1's and 0's (binary digits) that represent electronic signals that occur inside the computer. To program the earliest computers (in the 1940s and 1950s), programmers had to understand how the computer interpreted different combinations of 1's and 0's because the programmers wrote all their programs using binary digits. As programs became larger, it became very impractical to make programmers work in terms of the computer's 1's and 0's. Instead, researchers created programming languages that let people express the computer instructions in a form more meaningful to humans. After programmers placed their instructions in a file (called a *source file*), a second program (called a *compiler),* converted the programming language instructions into the 1's and 0's (known as *machine code*) the computer understood. The files on your disk with the EXE and COM extensions contain the machine code (1's and 0's) the computer will execute. Figure 1 illustrates the process of compiling a source code file into an executable program.

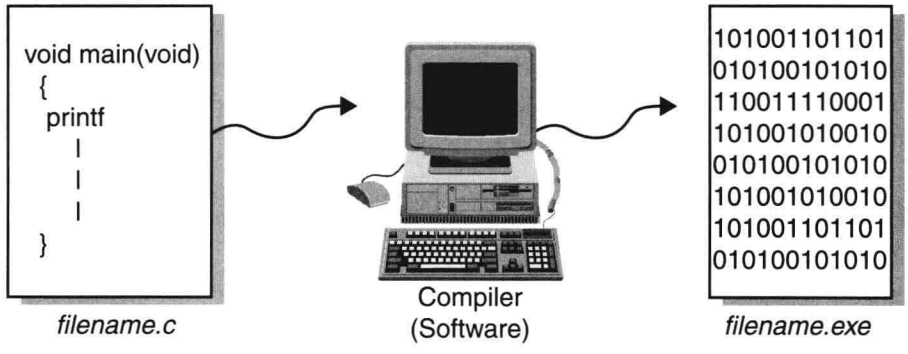


Figure 1 *A compiler converts source code instructions into machine code.*

After you create a source code file, you run a compiler to convert the instructions into a format the computer can execute. If you are using, for example, Borland's *Turbo C++ Lite*™ (included on the companion CD-ROM that accompanies this book), you will use the Compile menu Compile to OBJ option to invoke the compiler (that is, instruct it to compile the source file). The following Tips walk you through the steps required to create and compile a C program.

CREATING AN ASCII SOURCE FILE 2

When you create a program, you must place the program statements that you want the computer to execute in a file called a *source file*. If you are not using *Turbo C++ Lite* or a full-featured compiler and editor, you should create your program files using an ASCII editor, such as the EDIT program that DOS provides. You should not create programs using a word processor (such as Microsoft *Word*® or Corel's *WordPerfect*®). As you know, word processors let you format documents by aligning margins, italicizing and underlining text, and so on. To perform these operations, word processors embed special characters within your documents. Although these characters are meaningful to your

word processor, they will confuse the compiler that converts your source file to machine code, and this confusion will cause errors. When you create your source file, make sure you assign a meaningful name that accurately describes the program's function to the file. For example, you might name the source code for a billing program *billing.c*, and the source file for a game program *football.c*.

If, on the other hand, you are using a compiler that includes a built-in editor, you should create your programs within that editor. For example, if you are using *Turbo C++ Lite*, you will use the File menu New option to create a new program file. To create your first program within *Turbo C++ Lite*, perform the following steps:

1. Select the File menu New option. *Turbo C++ Lite* will create the *noname00.cpp* file.
2. Enter the following code into the *noname00.cpp* window:

```
#include <stdio.h>
void main(void)
 {
   printf ("Jamsa\'s C/C++ Programmer\'s Bible!");
 }
```

3. Select the File menu Save As option. *Turbo C++ Lite* will display the Save File As dialog box.
4. Within the Save File As dialog box, enter the name *first.c* and press ENTER. *Turbo C++ Lite* will save the *first.c* program file.

Although the *first.c* program contains six lines, only the *printf* statement actually performs any work. When you execute this program, *printf* will display the message *Jamsa's C/C++ Programmer's Bible!* on your screen. Every programming language (just like languages such as English, French, and German) has a set of rules, called *syntax rules*, which you must follow when you use the language. When you create C programs, you must obey the syntax rules of the C programming languages. Examples of syntax rules include the parentheses that follow the name *main* and the semicolon at the end of the *printf* instruction. When you type in your program, you must be very careful that you do not omit any of these elements. Double-check your typing to ensure that you have successfully typed in the C program instructions exactly as they appear earlier. If the instructions are correct, save the contents of the file to your disk. In the next Tip you will learn how to compile your source file and to convert your C programming instructions to the machine language that your computer can understand and execute.

3 COMPILING YOUR C PROGRAM

In the previous Tip you created the C source file, *first.c*, which contains the *printf* statement that will display the message *Jamsa's C/C++ Programmer's Bible!* on your screen when you execute the program. A source file contains instructions in a format you can understand (or at least you will be able to understand after you learn C). An executable program, on the other hand, contains instructions expressed as 1's and 0's that the computer understands. The process of converting your C source file to machine code is known as *compiling*. Depending on the C compiler you are using, the command you must perform to compile your source file will differ. Assuming you are using Borland's *Turbo C++ Lite*, you can compile the program (*first.c*) that you created in Tip 2, using the following command sequence:

1. Select the Compile menu Build All option. *Turbo C++ Lite* will display the Compiling dialog box.
2. If the compiler successfully completes the compilation, it will prompt you to *Press any key*. If the C compiler does not create the file *first.exe*, but instead displays error messages on your screen, you have probably violated a C syntax rule, as the next Tip discusses.
3. If you successfully typed in the C statements as shown in Tip 2, the C compiler will create an executable file named *first.exe*. To execute the *first.exe* program, you can either select the Run menu Run option or press the CTRL+F9 keyboard shortcut.