Sams Teach Yourself C in 24 Hours

Previous | Table of Contents | Next

Hour 1 - Getting Started

A journey of a thousand miles is started by taking the first step.

—Chinese proverb

High thoughts must have high language.

—Aristophanes

Welcome to Teach Yourself C in 24 Hours. In this first lesson you'll learn the following:

- What C is
- Why you need to learn C
- The ANSI standard
- Hardware and software required in order to run the C program

What Is C?

C is a programming language. The C language was first developed in 1972 by Dennis Ritchie at AT&T Bell Labs. Ritchie called his newly developed language C simply because there was a B programming language already. (As a matter of fact, the B language led to the development of C.)

C is a high-level programming language. In fact, C is one of the most popular general-purpose programming languages.

In the computer world, the further a programming language is from the computer architecture, the higher the language's level. You can imagine that the lowest-level languages are machine languages that computers understand directly. The high-level programming languages, on the other hand, are closer to our human languages. (See Figure 1.1.)

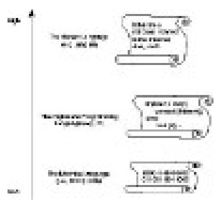
Figure 1.1. *The language spectrum.*

High-level programming languages, including C, have the following advantages:

- Readability: Programs are easy to read.
- Maintainability: Programs are easy to maintain.
- Portability: Programs are easy to port across different computer platforms.

The C language's readability and maintainability benefit directly from its relative closeness to human languages, especially English.

Each high-level language needs a compiler or an interpreter to translate instructions written in the high-level programming language into a machine language that a computer can understand and execute. Different machines may need different compilers or interpreters for the same programming language. For instance, I use Microsoft's C compiler to compile the C programs in this book for my personal computer (PC). If I need to run the C programs on a UNIX-based workstation, I have to use another type of C compiler to compile these programs. Therefore, the portability of programs written in C is realized by re-compiling the programs with different compilers for different machines. (See Figure 1.2.)

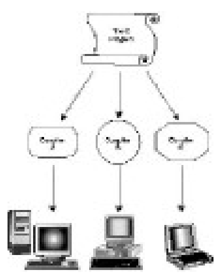
Figure 1.2. *Porting programs written in C into different types of computers.*

The Computer's Brain
You may know that the brain of a computer is the central processing unit (CPU). Some computers may have more than one CPU inside. A CPU has millions of transistors that make use of electronic switches. The electronic switches have only two states: off and on. (Symbolically, 0 and 1 are used to represent the two states.) Therefore, a computer can only understand instructions consisting of series of 0s and 1s. In other words, machine-readable instructions have to be in binary format.

However, a computer program written in a high-level language, such as C, Java, or Perl, is just a text file, consisting of English-like characters and words. We have to use some special programs, called compilers or interpreters, to translate such a program into a machine-readable code. That is, the text format of all instructions written in a high-level language has to be converted into the binary format. The code obtained after the translation is called binary code. Prior to the translation, a program in text format is called source code.

The smallest unit of the binary code is called a bit (from binary digit), which can have a value of 0 or 1. 8 bits make up one byte, and half a byte (4 bits) is one nibble.

In addition, the C language has other advantages. Programs written in C can be reused. You can save your C programs into a library file and invoke them in your next programming project simply by including the library file. More details on using libraries and invoking C library functions are covered in the rest of this book.

C is a relatively small programming language, which makes life easier for you. You don't have to remember many C keywords or commands before you start to write programs in C to solve problems in the real world.

For those who seek speed while still keeping the convenience and elegance of a high-level language, the C language is probably the best choice. In fact, C allows you to get control of computer hardware and peripherals. That's why the C language is sometimes called the lowest high-level programming language.

Many other high-level languages have been developed based on C. For instance, Perl is a popular programming language in World Wide Web (WWW) design across the Internet. Perl actually borrows a lot of features from C. If you understand C, learning Perl is a snap. Another example is the C++ language, which is simply an expanded version of C, although C++ makes object-oriented programming easier. Also, learning Java becomes much easier if you already know C.

NOTE

> There are two types of programming languages:compiled language and interpreted language.
>
> A compiler is needed to translate a program written in a compiled language into machine-understandable code (that is, binary code) before you can run the program on your machine. When the translation is done, the binary code can be saved into an application file. You can keep running the application file without the compiler unless the program (source code) is updated and you have to recompile it. The binary code or application file is also called executable code (or an executable file).
>
> On the other hand, a program written in an interpreted language can be run immediately after you finish writing it. But such a program always needs an interpreter to translate the high-level instructions into machine-understandable instructions (binary code) at runtime. You cannot run the program on a machine unless the right interpreter is available.
>
> You can think of the C language as a compiled language because most
> C language vendors make only C compilers to support programs written in C.

C and the ANSI Standard

For many years, the de facto standard for the C programming language was the K&R standard because of the book The C Programming Language, written by Brian Kernighan and Dennis Ritchie in 1978. However, there were many changes unofficially made to the C language that were not presented in the K&R standard.

Fearing that C might lose its portability, a group of compiler vendors and software developers petitioned the American National Standards Institute (ANSI) to build a standard for the C language in 1983. ANSI approved the application and formed the X3J11 Technical Committee to work on the C standard. By the end of 1989, the committee approved the ANSI standard for the C programming language.

The ANSI standard for C enhances the K&R standard and defines a group of commonly used C functions that are expected to be found in the ANSI C standard library. Now, all C compilers have the standard library, along with some other compiler-specific functions.

This book focuses on the C functions defined in the ANSI standard, which is supported by all compiler vendors. All programs in this book can be compiled by any compilers that support the ANSI standard. If you're interested in a specific compiler, you can learn the compiler-specific functions from the compiler's reference manual.

Assumptions About You

No previous programming experience is required for you to learn the C language from this book, although some knowledge of computers helps. Also, it's up to you to determine how quickly to go through the 24 hours of this book: You could sit up with a big pot of coffee and power through the book in a sitting or you could take an hour a day for 24 days.

Setting Up Your System

Basically, you need a computer and a C compiler in order to compile and run your own C programs or the C programs from this book. The recommended hardware and software are listed in the following sections.

Hardware

Any type of computer that has or can access a C compiler is fine. (The C compiler should support the ANSI standard.) More likely, you may have a PC on your desktop. A 286 PC with a 50MB hard drive and 1MB memory (RAM) is probably the minimum requirement to run a DOS-based C compiler. For a Windows-based C compiler, you must have a bigger hard drive and add more memory to your computer. Check your compiler vendor for more details.

Software

If you're using a UNIX-based workstation, you might already have a C compiler loaded

on your machine, or at least you might be able to access a C compiler on a server machine. Check with your system administrator to find out about a C compiler that supports the ANSI standard, and set up the right path to access it. On a UNIX-based machine, you should know how to use a text editor, such as vi and emacs, to write C programs.

If you have a PC, you need to install a C compiler and a text editor on it. Most C compilers come with an editor. You can also use a text editor that is already installed on your machine.

Borland International's Turbo C and Microsoft's Quick C used to be very popular in the C compiler market. These days, C compiler vendors like to bundle C and C++ compilers together. For instance, Borland International sells Borland C++ with a C compiler. The same thing is true for Microsoft's Visual C++. Both Borland C++ and Visual C++ support the ANSI standard. Both of them provide an integrated development environment (IDE), which you can use to edit your C programs, along with other fancy features for advanced Windows programming.

Appendix D, "Some Popular Commercial C/C++ Compilers," introduces several C compilers besides Borland C++ and Microsoft's Visual C++.

You can pick up any C compiler you like, as long as the compiler supports the ANSI standard. You shouldn't have any problems installing a C compiler on your machine if you read the manuals that come with the compiler and follow the installation instructions correctly. Most C/C++ compilers provide a quick tutorial that shows you how to install the compiler and set up a development environment on your computer.

TIP

> You can learn more about Borland C++ or Visual C++ from books such as Teach Yourself Borland C++ 5 in 21 Days (from Sams Publishing/Borland Press) and Teach Yourself Visual C++ 5 in 21 Days (also from Sams Publishing).

The Hardware and Software I Use for C Programming

I have a Pentium 100MHz PC with 32MB memory and with a 2.5GB hard drive. (32MB memory may be more than enough to run the C programs from this book, but I need a lot of memory space for Windows programming.) I have both Windows 95 and Windows NT on my machine.

Figure 1.3. *An example of the IDE from Visual C++ version 1.5.*

In this book, all C programs are developed with Microsoft Visual C++ version 1.5. (The latest version of Visual C++ is 5.0.) The reasons I chose Visual C++ 1.5 are simple: All C programs in this book are written in ANSI C and can be compiled into DOS-based applications; Visual C++ 1.5 has a good C compiler that supports the ANSI standard, as well as a simple but friendly enough IDE. Figure 1.3 shows an example of the IDE from Visual C++ 1.5. (The later version of Visual C++ has a fancy IDE with more features added. However, many of those features are not needed for running the C programs in this book.)

I set up my development environment in such a way that all C programs in this book can be compiled and made into DOS applications. Also, I test and run the applications made from the C programs at a DOS prompt provided by Windows 95.

Summary

In this first lesson you've learned the following:

- C is a general-purpose programming language.
- C is a high-level language that has the advantages of readability, maintainability, and portability.
- C is a very efficient language that allows you to get control of computer hardware and peripherals.
- C is a small language that you can learn easily in a relatively short time.
- Programs written in C can be reused.
- Programs written in C must be compiled and translated into machine-readable code before the computer can execute them.
- C provides many programming languages, such as Perl, C++, and Java, with basic concepts and useful features.
- The ANSI standard for C is the standard supported by all C compiler vendors to guarantee the portability of C.
- You can use any C compilers that support the ANSI standard and compile all C programs in this book.

In the next lesson you'll learn to write your first C program.

Q&A

Q What is the lowest-level language in the computer world?

A The computer's machine language is the lowest because the machine language, made up of 0s and 1s, is the only language that the computer can understand directly.

Q What are the advantages of high-level programming languages?

A Readability, maintainability, and portability are the main advantages of high-level programming languages.

Q What is C, anyway?

A C is a general-purpose programming language. It's a high-level language that has advantages such as readability, maintainability, and portability. Also, C allows you to get down to the hardware to increase the performance speed if needed. A C compiler is needed to translate programs written in C into machine-understandable code. The portability of C is realized by recompiling the C programs with different C compilers specified for different types of computers.

Q Can I learn C in a short time?

A Yes. C is a small programming language. There are not many C keywords or commands to remember. Also, it's very easy to read or write in C because C is a high-level programming language that is close to human languages, especially English. You can learn C in a relatively short time.

Workshop

To help solidify your understanding of this hour's lesson, you are encouraged to answer the quiz questions provided in the Workshop before you move to the next lesson. The answers and hints to the questions are given in Appendix E, "Answers to Quiz Questions and Exercises."

Quiz

1. What are the lowest-level and highest-level languages mentioned in this book?
2. Can a computer understand a program written in C? What do you need to translate a program written in C into the machine-understandable code (that is, binary code)?
3. If needed, can a C program be reused in another C program?
4. Why do we need the ANSI standard for the C language?

Previous | Table of Contents | Next