[美] 高永强 编著

全C编程

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书把 C 和 C++ 融为一体,从实用角度介绍程序设计和编程技术。它从教学和自学两方面的需求出发,利用 300 多个完整的程序例子,由浅入深,一步一步系统地讨论和介绍了 C 和 C++ 中基本和常用的语句、运算以及操作。由于本书打破了传统的人为设立在 C 和 C++ 之间的"界限",而本着"不管是 C 还是 C++,哪个好用好学就用哪个"的思想,因此读者可以利用贯穿于书中的有效的学习方法,在较短的时间内同时掌握 C 和 C++ 的基本和主要的编程概念和技术。

除以综合方式介绍程序设计和编程方法外,本书还具有如下几个特点:一是利用一章的篇幅详细介绍了怎样利用排错工具 Debugger 来学习程序和语言;二是在本书的后四章中着重讨论了面向对象的编程技术(Object-Oriented Programming);三是在几乎每章最后,都包括了解决实际问题的实例,综合性地介绍了如何利用在该章讨论过的概念、语句、运算以及操作来解决实际问题。在每章后安排的"边做边学练习"中,还进一步要求读者理解和掌握在本章程序例子中介绍的概念和编程技术。

本书是为初学语言和编程的读者编写的。它适合于大学、大专、中专以及培训中心作为教科书使用,可以作为自学课本,也可以作为专业人员的参考书。

书　名:全 C 编程
作　者:[美] 高永强　编著
出版者:清华大学出版社(北京清华大学学研大厦,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者:清华大学印刷厂
发行者:新华书店总店北京发行所
开　本:787×1092 1/16　**印张**:39.75　**字数**:915 千字
版　次:2002 年 6 月第 1 版　2002 年 6 月第 1 次印刷
书　号:ISBN 7-302-05054-6/TP・2950
印　数:0001～4000
定　价:59.00 元

目　　录

第1章　全C编程

本章目的

通过本章的学习,你可以了解:

- 什么是和为什么要学习全C编程
- 什么是程序设计的8个步骤以及每个步骤的内容
- 什么是编译器以及它的作用
- 用什么标准来衡量一个程序
- 怎样分析问题,如何建立解决问题的模式,以及完成算法的基本步骤
- 怎样用程序设计的8个步骤来设计一个简单程序

1.1　什么是全C编程

因为当前微机上的大多数C编译软件都包括C++编译器,因此全C编程可定义为一种综合利用C和C++的优越性和功能,不把它们分隔开来的编程方法。事实上,在实际应用中,许多编程人员正是这样做的。无论是C还是C++,它们的数据类型、表达式、语法和基本语句,例如赋值语句、循环语句或分支语句,都是相同的,在本质上没有把它们分开的必要。从另一方面讲,许多C++的优越功能若单纯从C语言的规范来讲是非法的,但若从全C编程的角度来介绍,读者就容易理解和掌握这些功能。

总之,全C编程以面向实践并强调解决实际问题为编程人员的首要任务来介绍语言,而不拘泥于编写的程序是属于C还是C++。

1.2　C和C++的历史

C编程语言的最初版本称为K&R C语言,是由两位AT&T贝尔实验室的编程人员布莱恩·科尔尼汉(Brian Kerninghan)和丹尼斯·瑞塔奇(Dennins Ritchie)于1972年共同开发的。K&R C语言是由另一个源语言"基本综合编程语言"(Basic Combined Programming Language)或称为BCPL的语言演进而来的。BCPL在编程界被人们简称为B,是由一位贝尔实验室的操作系统设计员肯·汤姆森(Ken Thompson)在1967年开发的。C语言在最初只是用于编写操作系统。但因为它简单易用、灵活,并具有很强功能,因此不久就被许多编程和应用领域所采用。C后来被开发成许多版本,例如ANSI(American National Standards

Institute) C, Unix C 以及 Turbo C 等。C 可说是当前最流行的编程语言之一。

C 语言与一些其他早期流行的语言一样,是一种面向过程的语言(Procedure-Oriented Language, POL)。这意味着这种语言必须先对数据进行定义,然后应用一系列步骤对数据进行运算处理。因而,在面向过程的语言中,数据和对数据的运算操作是分开的。计算机学者和编程人员发现,如果能把数据和对数据的操作合为一体,这种处理结构就会使程序更易理解,并能具有重复使用性,且易于编写。这种结构就是现在被称为"对象"(Object)或"类"(Class)的结构。编写具有"类"这种功能的程序被称为面向对象的编程(Object-Oriented Programming, OOP)。设计这种有"类"的程序的过程被称为面向对象的设计(Object-Orientde Design, OOD)。

20 世纪 80 年代初,在 AT&T 工作的贝加尼·斯床斯塔普(Bjarne Stroustup)对 C 进行了扩展,并使之具有了某些面向对象编程的特性。这个新的 C 版本起初被称作"具有类的 C 语言"。后来,这个版本不断修改、补充和论证,才成为了今天的 C++。C++ 实际上是 C 的扩展,任何标准的 C 程序都是 C++ 的合法程序,所有 C 的系统子程序都是 C++ 系统子程序的一部分。当前流行的 C 编译软件,如 Unix C, Borland Turbo C 或微软的 C 中,都包括C++ 编译器。

1.3 为什么要学习全 C 编程

一个初学编程的人应当先学 C 还是先学 C++? 这个问题在计算机语言教学中引起了广泛的兴嗓和争论。主张先学 C 的人认为,C++ 对初学者来说太难了,因为面向对象编程不是以传统的方式编写程序,所以应当先学 C,然后再学 C++。主张先学 C++ 的人则反驳道,如果先学 C 再学 C++,这是浪费时间。还有一些人的看法是,没有必要让初学者先用旧的并且是难的方式学习 C 语言中的输出/输入操作,然后再在 C++ 中学习容易的并且是简单的同类操作,因为这类操作在 C++ 中已做了改进。另外,这些人还指出,一些在 C 中是非法而在 C++ 中为合法的操作和指令,如果把它们分隔开来介绍,只能对初学者造成概念上的混乱。

全 C 编程这种方式吸取了以上这些论点中的长处,对初学编程的人来说是一种有效并且省时的学习方法。因为这种方法可以使初学者同时学习 C 和 C++ 而不用花费两倍的时间。全 C 编程避免了那些先学习 C 好还是先学习 C++ 好的争论,并避免了这些方法中各自存在的缺点。本书在介绍全 C 编程这一方法时,力求在编排上由浅入深,重在练习和实践,并强调分析问题和解决问题的能力。对书中介绍的每一个概念、操作和指令,都通过许多例子来加以解释。这些例子都是完整的程序,并附在本书所带的软盘中,读者可以调出运行,以便加深理解。另外,全 C 编程不讨论某条指令、某个函数,或者是某个系统子程序究竟是属于 C 还是属于 C++,这样就避免了没有必要的限制,使编程人员有更大的灵活性。

1.4　一个全 C 编程的例子

清单 1.1 是一个简单的全 C 程序。这个程序把信息“你好！全 C 编程。”显示到屏幕上。

清单 1.1　一个简单的全 C 编程例子

```
1: //这个程序的文件名为 ALL_C.CPP,在本书所带软盘中。
2: /* 这个程序将把"你好！ 全 C 编程。"的信息显示到屏幕上。 */
3:
4: #include<iostream.h>
5:
6: void main()
7: {
8:     cout<<"你好！ 全 C 编程。"<<endl;
9: }
```

首先来看程序的第 4 行：

```
#include<iostream.h>
```

其中，#include< >是一个编译指令，iostream.h 是一个系统文件，或称头文件。这个头文件是专门用来定义输入/输出操作的。在 C/C++ 中，头文件一般是针对在程序中进行的某种类型的操作，由语言开发者设计的接口子程序。这些接口子程序又称作库子程序，或系统子程序。在 C/C++ 中规定，这些系统子程序必须由编译指令 #include< > 调用，而且必须放在一人程序的开头。在这个例子中，由于有如下一行输出语句：

```
cout<<"你好！ 全编程。"<<endl;
```

所以开始必须用 #include< > 来调用管理输入/输人操作的头文件。一般来说，对每一个具有输入、输出操作的全 C 程序，都必须在程序的开始包括这一用于调用输入/输出头头文件的编译指令。这个问题将在第三章详细讨论。

接着再看下一句(第 6 行)：

```
void main()
```

综定义一个主程序，表示下面的程序行都属于这个主程序，并由一对花括号({})来表示这些程序行的范围。关键字 void 表示这个主程序不送回任何返回值。因为在 C/C++ 中，任何一个完整的程序都必须有一个主程序块，所以 main 表示它所代表的程序是主程序。

由花括号括起来的程序行都称作语句。在清单 1.1 的例子中，只有一条语句：

```
cout<<"你好！ 全 C 编程。"<<endl;
```

因为这一行程序执行输出操作，因此也称为输出语句。cout 称作输出，在 iostream.h 中解

释为标准输出设备,一般指屏幕。<< 是运算符,称作插入运算符。把插入运算符(<<)放置在输出 cout 的后边,将形成一个标准的输出操作。在这个例子中,该语句把信息(或字符串)

```
你好! 全 C 编程
```

输出到计算机屏幕上。引号所包括的内容称为常量字符串。这种类型的输出称为常量字符串输出。endl 是一个操作算子。它的功能和回车键一样,表示开始一个新行。在第三章中将更详细地讨论这些内容。

现在回到这个例子的头两行:

```
//这个程序的文件名为 ALL C.CPP,在本书所带软盘中。
/* 这个程序将把"你好! 全 C 编程。"的信息显示到屏幕上。 */
```

这两行称作程序注释行,或简单称作注释行。注释行可以有两种形式,分别由双斜线//或一对/* */组成。由双斜线开始的程序行表示双斜线后面的内容为程序注释;由/* */组成的程序行表示用这对符号包括的所有内容为程序注释,其中可以包括多行内容。在以后的章节里将进一步讨论注释行的各种用法。编译器在编译程序时,将跳过所有注释行,不对它们做任何处理。程序注释可以是任何文字内容,一般用来对程序、语句以及一些重要的操作进行必要的解释,以增加程序的可读性。所以,虽然程序注释行可有可无,但对人们了解和看懂程序来说是必要的。

现在来编译和运行清单 1.1 中的程序例子。如果计算机已经进入了 C/C++ 的编辑屏幕,那么读者可以从键盘键入这个简单的程序,或者从本书所带的软盘中打开相应文件(ALL_C.CPP)。如果不熟悉 C 的编辑屏幕,或不知道怎样进入 C/C++ 编译器,可请教教师或有关的计算机室人员。

假设已经键入或打开了这个程序,下一步就是编译并运行这个程序。由于这个例子已经被笔者测试过且正确无误,所以可以用最简单的方法来编译和运行它,即:**从菜单或图标上选择 Run(或按下 Alt + F9 键)**。这时,编译器将对程序进行编译,把源程序翻译成机器码,然后再对机器码进行连接,使之生成一个可执行程序,最后运行这个程序。屏幕上最终应该显示这样一行输出结果:

```
你好! 全 C 编程。
```

注意,如果程序有任何错误的话,编译器将停止对程序的编译或运行,并将有错误的语句的行号以及错误的类型显示在屏幕上。在这种情况下,必须修改这些错误,保存好修改后的文件,然后对程序再次编译运行。

下一节介绍在一般情况下怎样编译、连接并运行一个程序,讨论程序错误的种类,以及如何查找和修改这些错误。在第 4 章将详细介绍程序的纠错和调试方法。

1.5 全 C 编程的步骤

从实践的角度来看,编写程序的目的就是要解决实际中所存在的问题。全 C 编程与其他编程一样,需要编程人员遵守一定的步骤。这些步骤可以归纳为如下八步:

第一步：分析问题

第二步：建立解决问题的模式

第三步：把解决问题的模式转化为算法

第四步：把算法转化为语言程序

第五步：对程序进行编译

第六步：运行程序及查错

第七步：对程序进行测试

第八步：编写程序的文档及说明书

下面对这八个步骤逐一进行讨论。在下一节中将用实例来对这些步骤的具体应用进行说明。

第一步：分析问题。在大多数情况下，一个给定的问题以及对这个问题的描述往往不很具体和精确，这就需要编程人员对问题进行分析和研究，以便对问题有确切的描述，并找出解决问题的方法。在分析问题的过程中，应把注意力集中在如下几个方面：

- 要解决的问题是什么？
- 哪些数据已经给定？
- 哪些数据没有给出？
- 哪些数据需要在计算中产生？
- 要用哪些计算、逻辑或数学模式来产生这些数据？
- 要输出哪些数据来满足解决问题的需要？

基于以上分析的结果，就可以对问题有一个规范化的确切描述。这些规范说明应当包括如下几个方面：

- 对输入数据的描述。
- 对计算公式、逻辑以及数据模式的描述。
- 对输出数据的描述。

在第二步中将应用这些规范化的描述的结果来建立一个解决问题的模式。

第二步：建立解决问题的模式。从程序设计的角度看，大多数问题都包括三个基本模块：输入模块、处理模块以及输出模块。如图 1.1 所示，在第一步的分析问题中对输入数据的描述可以进一步精炼成为一个输入数据表；同样，在分析问题中对计算分式、逻辑以及数学模式的描述，可以进一步精炼成为一个处理模式表；而对输出数据的描述则可以进一步精炼成为具体的输出数据表。

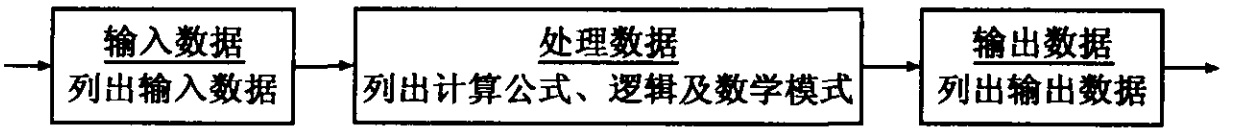

图 1.1　解决问题的模式

第三步：把解决问题的模式转化为算法。在建立了解决问题的模式的基础上，把这个模式转化为解决问题的算法就比较容易了。算法是一系列确切描述的方法或步骤，这些方法或步骤在一段有限的时间内解决给定的问题。算法的形式多种多样，没有固定的模式。一般习惯上以提纲的形式书写，必要时加入一些控制流程符、分支符、循环符等，以对相应的处理及操作加以说明。算法中的每一行都表示一个或若干个解决问题的运算或操作。下一节中将用实例对算法加以解释。

第四步：把算法转化为语言程序。编写程序实际上是把算法转化为一个具体语言程序的过程。例如，在全 C 编程中，编程人员必须遵守 C 或 C++ 的语法、词法以及书写格式的规定，把解决问题的算法编写为具体的程序。这样的程序称为源程序。源程序可以被编译器编译成为机器码，即可执行文件。这些操作将在以后的章节里详细加以论述。

第五步：对程序进行编译。编译器实际上是一个程序包，它的功能是把源程序翻译成机器码，并将机器码连接为可以由计算机运行的可执行文件。在全 C 编程中，C++ 编译器把 C 或 C++ 源文件翻译成为具体的计算机机器语言，例如 IBM 微机及其兼容机的英特尔(Intel)机器码、苹果机的摩托罗拉机器码。这些机器码称作目标文件。目标文件必须经过编译器中的连接程序连接成为可执行文件，才可以由计算机执行。连接程序实际上把机器码以及源程序中调用的库程序连接起来，以形成一个可执行文件。

编译器的另一个功能是对源程序中的语法以及词法错误进行检查。如果源程序中有这类错误，编译器将显示错误所在的行号以及可能的错误类型，并停止建立目标文件。在这种情况下，程序员必须对源程序进行检查和修改，然后对源程序重新进行编译，直到没有任何这类错误为止。

第六步：运行程序及查错。编译器产生了一个可执行文件，并不等于说程序中一点错误都没有了。除去语法及词法错误之外，源程序中还可能存在运行错误。运行错误也称为语义错误、含义错误或者逻辑错误。例如，程序不能产生应有的计算结果；程序运行中的死循环；除法运算中的除数为零；以及在程序运行过程中出现的其他错误。运行错误在英文中被称为“Bug”，意思也就是故障错误。排除运行错误的过程被称为“Debug”，即为查错排除故障的意思。用来排除运行错误的软件被称为“Debugger”。在以后的章节里，我们会用实例对各种排除运行错误的方法进行详细论述。如果程序在运行中出现了运行错误，程序员就必须对源程序进行检查，找出错误所在，并修改这些错误，然后对修改后的源程序进行编译和连接，以产生一个新的可执行文件。接着，再运行这个可执行文件，查看是否仍有错误，如此反复，直到程序产生满意的输出结果为止。

第七步：对程序进行测试。对程序进行测试的目的是增加程序的可靠性，以保证程序对各种输入数据都能产生正确的输出结果。对程序测试的另一个目的是为了保证程序能够对各种错误的输出产生必要的警告信息，必要时停止程序的运行。在程序的测试中可以应用许多方法和技巧，例如，设计各种测试数据对程序进行测试；在程序中可能出现运行错误的地方加入各种查错子程序，专门用来处理各类运行错误，并输出必要的警告信息或中断程序的运行；用查错软件对程序进行逐步或分段的检测，并向一些重要的变量赋