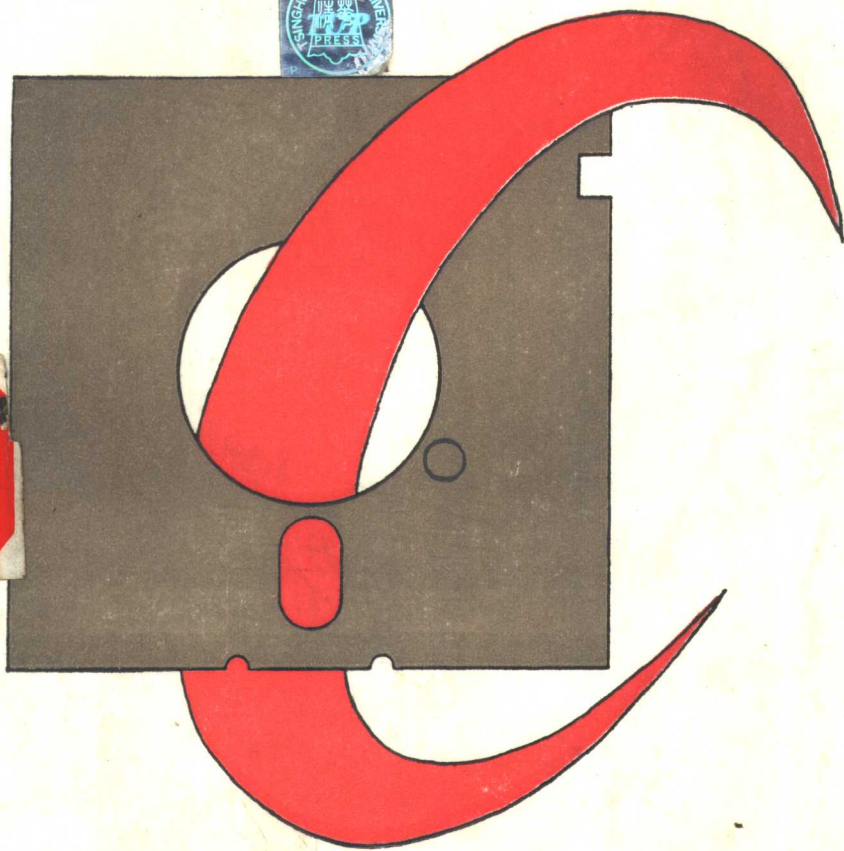


C 语言程序设计

实用指南

王继贤 编

清华大学出版社



(京)新登字 158 号

内 容 提 要

本书全面讲述了 C 语言的基本内容,重点讲述了数据和运算、语句、数组、指针、结构与联合、函数、变量的性质及预处理等。

本书可供计算机软硬件开发人员、工程技术、科研管理人员学习,也可以供大专院校师生以及各种培训班和研究生班学员参考。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标志,无标志者不得销售。

图书在版编目(CIP)数据

C 语言程序设计实用指南/王继贤编. —北京:清华大学出版社, 1994

ISBN 7-302-01482-5

I. C… II. 王… III. C 语言-程序设计-手册 IV. TP312C

中国版本图书馆 CIP 数据核字(94)第 01911 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:通县宏飞印刷厂

发行者:新华书店总店 北京科技发行所

开 本: 850×1168 1/32 印张:7.125 字数:183 千字

版 次:1994 年 8 月第 1 版 1995 年 4 月第 2 次印刷

书 号:ISBN 7-302-01482-5/TP·589

印 数:10001—25000

定 价:6.80 元

前 言

C 语言已作为一种主要的程序设计语言,广泛地流行于国外,而国内尚处于普及阶段。我们总结了这几年应用 C 语言的实际工作经验,编写了本书,为读者提供了一本有关 C 语言的实用性读物。

全书共分十一章。在一、二章讲述概论及程序设计的基础上,通过第三章到第十章,全面而详尽地讲解了数据和运算、语句、数组、指针、函数、变量的性质及预处理等主要内容。第十一章介绍了综合应用。

本书的主要特点是:

1. 理论联系实际,实用价值高。由于它是我们实践经验的总结,根据自己的实践而又用程序员常用的词语编写而成,所以容易理解,书中列举了 112 个例子,对应于每一个概念和定义,使本书不仅可读性强,而且富有实际参考价值。例如:第三章讲了逻辑运算符的规则之后,就用例 3-12,例 3-13 两个例子,进一步讲解它在实际的程序设计中,是如何应用的。第九章用了 12 个例子说明各种变量的性质及其应用。

2. 通俗易懂,概念清晰。本书在理论、概念的阐述中,注意将定义讲明确,利用插图使内容更加生动和形象,在讲述各种语句时,尽可能用逻辑框图。例如第四章就用了 17 张图。这在同类书籍中还是不多的。

3. 简明扼要,重点突出。通过本书极其简炼的叙述,使读者对 C 语言的程序设计从理论到实践有一个全面的了解。

由于编者水平有限,时间短促,一些问题的阐述,不当之处敬
希广大读者不吝指正。

作 者

1993 年 10 月

目 录

前 言	(I)
第一章 概论	(1)
§ 1.1 C 语言的发展史	(1)
§ 1.2 C 语言的特点	(3)
§ 1.3 C 语言的符号	(5)
§ 1.4 C 语言的程序结构	(6)
§ 1.5 C 语言的格式	(12)
思考题及练习	(12)
第二章 程序设计	(14)
§ 2.1 程序设计概念	(14)
§ 2.2 提高程序设计的质量和水平	(15)
§ 2.3 程序的编辑、编译和执行	(16)
思考题及练习	(24)
第三章 数据和运算	(25)
§ 3.1 标识符	(25)
§ 3.2 常量	(27)
§ 3.3 变量	(32)
§ 3.4 运算符	(34)
§ 3.5 优先级	(46)
§ 3.6 数据类型转换	(48)
思考题及练习	(50)
第四章 语句	(51)
§ 4.1 语句分类及程序结构类型	(51)
§ 4.2 类型说明语句	(54)

• ■ •

§ 4.3	赋值语句	(57)
§ 4.4	表达式语句	(58)
§ 4.5	空语句	(58)
§ 4.6	返回语句	(59)
§ 4.7	输入语句	(59)
§ 4.8	输出语句	(61)
§ 4.9	复合语句	(71)
§ 4.10	分支结构	(71)
§ 4.11	循环结构	(88)
§ 4.12	转移语句	(97)
	思考题及练习	(99)
第五章	数组	(101)
§ 5.1	数组的概念	(101)
§ 5.2	数组的定义	(102)
§ 5.3	数组在内存中的存储方法	(103)
§ 5.4	数组的初始化	(106)
§ 5.5	数组的应用	(108)
	思考题及练习	(110)
第六章	指针	(111)
§ 6.1	指针的概念	(111)
§ 6.2	指针的定义及初始化	(112)
§ 6.3	指针的运算	(113)
§ 6.4	指针与数组	(115)
§ 6.5	指针和函数	(117)
§ 6.6	字符指针与字符串	(117)
§ 6.7	指针数组与多级指针	(119)
	思考题及练习	(122)

第七章 结构、联合及其它类型	(123)
§ 7.1 结构的概念	(123)
§ 7.2 结构的定义和说明	(123)
§ 7.3 结构的使用	(126)
§ 7.4 结构数组与结构指针	(128)
§ 7.5 结构的初始化	(130)
§ 7.6 结构中的结构	(131)
§ 7.7 联合	(134)
§ 7.8 枚举类型	(139)
§ 7.9 类型定义	(141)
思考题及练习	(143)
第八章 函数	(144)
§ 8.1 函数的概念及定义	(144)
§ 8.2 函数调用及说明	(146)
§ 8.3 函数之间通信——参数传递数据	(150)
§ 8.4 函数之间通信——全局变量传递数据	(152)
§ 8.5 递归函数	(153)
§ 8.6 函数和数组	(155)
§ 8.7 库函数	(159)
思考题及练习	(161)
第九章 变量的性质及存储类型	(162)
§ 9.1 自动变量	(162)
§ 9.2 外部变量	(166)
§ 9.3 静态变量	(170)
§ 9.4 寄存器变量	(175)
§ 9.5 初始化	(176)
§ 9.6 小结	(181)
思考题及练习	(183)

第十章 预处理	(184)
§ 10.1 预处理的概念.....	(184)
§ 10.2 宏定义.....	(185)
§ 10.3 文件包含.....	(190)
§ 10.4 条件编译.....	(192)
§ 10.5 利用预处理功能提高软件的质量.....	(194)
思考题及练习.....	(197)
第十一章 综合应用	(198)
§ 11.1 国外的例子.....	(198)
§ 11.2 国内的例子.....	(204)
思考题及练习.....	(210)
附录 A 常用的 UNIX 命令	(211)
附录 B 关键字	(213)
附录 C ASCII 码表	(214)
参考文献	(217)

第一章 概 论

本章主要介绍 C 语言的发展历史、特点、符号以及程序结构和程序格式等问题。

§ 1.1 C 语言的发展史

计算机技术从 40 年代中期问世以来,在短短的几十年时间里,其发展速度是十分惊人的。软硬件更新的周期都很短,每一个创新,都刻划着时代的特征,推动着计算机技术的进步。C 语言的出现以及相继问世的 UNIX 操作系统,专家们一致认为是具有划时代意义的。

C 语言的发展有一个不断演变、扩充及改进的过程,如图 1-1 所示。C 语言的前身是复合程序设计语言 CPL^[1,2,3] (Combined Programming Language),它是仿照 ALGOL 60 而设计出来的语言,由于 CPL 庞大而又复杂,不便于使用,推广受到了很大的限制。于是 Martin Richard 改进设计了一个 BCPL 语言 (BASIC CPL),但 BCPL 在实践中仍不十分方便,1970 年 Ken Thompson 再一次对 BCPL 加以改进,提出了一种 B 语言。接着 D. M. Ritchie 在 B 语言的基础上,再作了若干充实,改进,于 1972 至 1973 年间,提出了一种 C 语言。

在 C 语言发展的每一个阶段,都有它的优点和不足,科学家们总是在吸收前人研究成果的基础上,补充新的功能,改进它的不足。例如 B 语言就缺乏丰富的数据类型,而 D. M. Ritchie 不仅改进了这一点,而且还增加了语言的控制能力,对整个 B 语言进行了

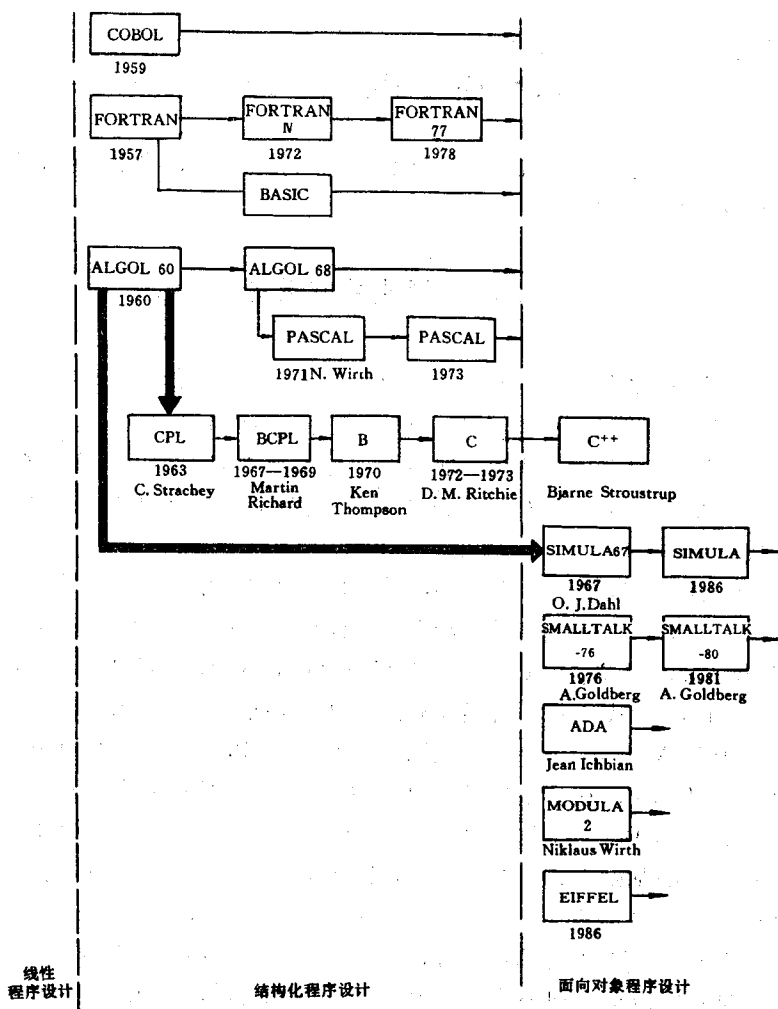


图 1-1 程序设计语言及程序设计的发展

总体设计和清理,从而形成了一个较为理想的编程工具——C 语言。

Ken Thompson 和 D. M. Ritchie 于 1973 年利用新的版本 C 语言,重新设计了 UNIX,进而扩大了 UNIX 的功能。所以 C 语言的发展和它在编写 UNIX 操作系统中的应用,是紧密相关的。UNIX 的发展之所以有今天这样的扩展面,正是因为有了 C 语言这种好的编程语言。

§ 1.2 C 语言的特点

C 语言在它短短的十多年发展中,能够成为当今计算机界公认的一种通用的、好的程序设计语言,究竟有什么特点呢?

1. C 语言是一种结构化的程序设计语言。如图 1-1 所示,程序设计大致可以划分为三个阶段:线性程序设计、结构化程序设计和面向对象的程序设计^[3,5]。

所谓线性程序设计,是直接用机器语言和符号(机器代码)来编写程序,这些程序仅能完成简单的任务,而且是线性的、顺序的系列。

结构化程序设计就完全不同了,它能针对各种复杂的大型应用问题进行设计。按照系统论的观点,一个大型的复杂问题(系统),可以分成若干小系统,每个小系统又可分成若干进程,或者需要解决的问题(C 语言将这些进程称为一个函数)。当程序运行时,把这些不同的函数(或者叫做模块)组装起来就可以了。所以函数是 C 语言中结构化程序设计的基本单元。

结构化程序设计技术,逻辑思维清晰、容易编制,也易于理解和维护。今天几乎所有的计算机语言都支持这种技术,C 语言以其结构简洁、逻辑清晰而深受广大程序设计人员的喜爱。

2. C 语言独立于计算机,对机器的依赖性很小。今天无论微

型机(PC)机,工作站(WS),大、中、小型机,乃至巨型机,都配置有C语言编译系统,这是C语言能够获得广泛应用的又一个重要原因。

C语言既有面向结构化程序设计的许多高级语言所具有的功能,也有面向硬件,具有汇编语言的某些功能。例如:它可以直接处理字符、位加工,地址及指针运算等,这些通常需要由汇编语言来完成。这一重要特征可以使C语言大大提高运算效率^[3,5]。C语言在硬件、软件中的地位示于图1-2。图中椭圆的大小,标志着该语言的地位及作用范围的相对大小。

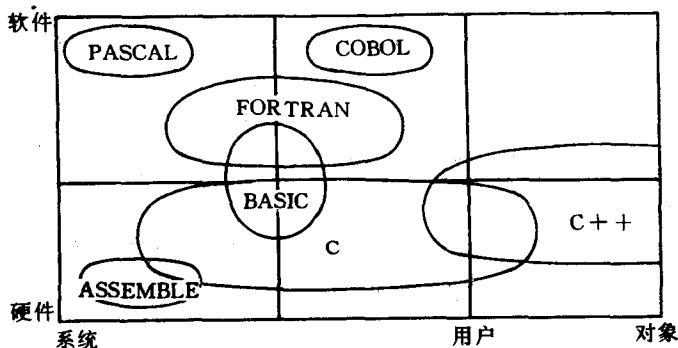


图 1-2 C语言在硬件、软件中的地位

3. C语言具有丰富的运算功能。C语言除了具有一般高级语言所具有的四则运算:加(+)、减(-)、乘($\times$)、除(/);逻辑运算:逻辑与(&&)、逻辑或(||)、逻辑非(!),位运算(&,|,>>,<<)以及关系运算而外,还有其它高级语言所没有的单项运算 $x++$ 和 $Y--$ 以及各种复合运算($+=$, $-=$, $*=$, $/=$),指针运算,条件运算等等。

4. C语言具有丰富的数据类型。一种语言一方面要对数据有较强的处理能力,即具有丰富的运算功能,另一方面也要对处理对

象——数据,有较强的描述(定义)能力。C语言除了同其它高级语言一样,具有整型、实型、字符型、数组型等而外,更有其它高级语言所没有的类型,如结构型、联合型、指针型等。

5. C语言预处理程序的功能,独树一帜,别具一格,是其它高级语言所没有的。C语言利用它独有的`#define`, `#include`, 及`#if`, `#else`等语句,就可以进行字符串,参数的各种宏定义,文件包含,条件编译等各种预处理,从而大大提高了软件开发的效率,提高了软件开发的质量。

6. C语言是面向对象程序设计的基础。C语言中几乎所有的功能和特点,都已被C++吸引,而C++经过改进和扩充,已发展成为面向对象的程序设计语言^[5]。面向对象的程序设计是软件设计技术今后发展的方向(参看图1-1)。所以我们学习掌握了C语言,也就为跨入程序设计的第三阶段,为今后的发展,提前做好了准备。正是由于C语言具有这种独有的特点,才使它具有广泛的应用领域和强大的生命力。

§ 1.3 C语言的符号

任何一种语言都有它自己的一些规定的语言符号,利用这些符号就可以定义语法,并组成不同的语句。

C语言的基本符号有4类103种:

1. 字母:英文大小写字母各26个,共52个。

2. 阿拉伯数字:0—9,共10个。

3. 特殊字符:

(1) 算术运算符:加(+),减(-),乘(*),除(/),取模(%);

(2) 自增自减运算符:自增加1(++),自减减1(--);

(3) 关系运算符:小于(<),小于等于(<=),大于(>),大于等于(>=),不等于(!=),全等比较(==);

- (4) 逻辑运算符: 逻辑与(&&), 逻辑或(||), 逻辑非(!);
 - (5) 位逻辑运算符: 与(&), 或(|), 加(^), 求反(~);
 - (6) 位移运算符: 左移(<<), 右移(>>);
 - (7) 赋值运算符: =;
 - (8) 组合运算符: 加等(+=), 减等(-=), 乘等(*=), 除等(/=), 取模等(%=), 左移等(<<=), 右移等(>>=); 与等(&=), 或等(|=), 按位加等(^=);
 - (9) 逗号运算符: , ;
 - (10) 条件运算符: ? ;
 - (11) 地址运算符: 取地址(&), 取内容(*);
 - (12) 其它运算符: 不同 C 语言版本, 各有不同。如数组下标定界[], 结构、联合体取成员, 函数定义及引用(), 结构指针取成员→, 等等。
4. 下线符: _。

§ 1.4 C 语言的程序结构

下面通过几个简单的例子来说明 C 语言的程序结构及其最基本的组成部分。

例 1-1 要在一个终端上显示一句话:

C 语言是非常重要的。即: C language is very important!

要用 C 语言来实现这个任务, 应该这样编写:

```
main()
{
    printf("C language is very important! \n");
}
```

这是一个非常简单的 C 语言程序, 如果将它键入计算机, 并经过编译、执行, 就可以在终端上看到该程序的输出结果是:

C language is very important!

现在我们根据 C 语言的一些规定,来对例 1-1 作些解释。

程序的第一行:main()表示该程序的名字为 main。main 是 C 语言编译系统用的专有名字,它表示该程序是从这里开始。按照惯例,C 语言将程序称为函数,所以本书也采用函数一词。在 FORTRAN, BASIC 等语言中有主程序、子程序的说法,而在 C 语言中,为便于理解以及叙述的方便,我们将 main() 称为主函数,其他函数,则称为子函数。如果有几个函数时,C 语言编译系统就从带有 main 这个名字的主函数开始执行。当然例 1-1 只有一个函数,它就是 main 函数。main 后边括号中,可以有参数,也可以不写参数(用隐含值),但这对圆括号()必须要有,不能省略。

{ } 是一对花括号,它相当于 ALGOL, PASCAL 语言中的 begin 和 end。在 { } 中可以包含许多语句,组成函数体。本例中,只有一个语句。

printf() 是一个函数,是用 C 语言编写的标准的库函数。目前,在 UNIX 系统中,大多数计算机都装有这种标准的库函数,以下简称 C 库。printf 的功能是:在输出设备或终端荧光屏上打印或显示一些信息以及需要输出的内容(将在以后各章节中介绍具体实例)。在这里是调用这一标准库函数输出一句话:“C 语言是非常重要的!”

双引号“ ”中所包含的字符串“C language is very important!”,就是 printf 这一函数的参数,也就是我们要输出的信息。在“ ”中的最后两个符号 \n,表示换行。它表示在 \n 后面的信息均要输出到下一行去。在这里 \n 后面已无信息可传输,读者将在以后有关的实例中看到, \n 后面就有换行的输出。

在 printf(...) 后面是一个分号;这是 C 语言中语句的结束符。它是不可缺少的。

如果我们还要显示或者打印输出一句话:

Programming in c is very useful!
应该如何编写呢? 请看下面的例题。

例 1-2 要求输出更多信息的程序。

```
main()  
{  
    printf("C language is very important! \n");  
    printf("Programming in c is very useful! \n");  
}
```

这一程序的输出结果是:

```
C language is very important!  
Programming in C is very useful!
```

例 1-3 求三个整型数 a,b,c 之和。

```
main()  
{  
    int a, b, c, sum;  
    a=2;  
    b=4;  
    c=6;  
    sum=a+b+c;  
    printf("The sum of a+b+c is % d\n", sum);  
}
```

在这个例题中,函数体由 6 个语句组成。其中:int a, b, c, sum 是说明语句,它说明 a, b, c, sum 四个变量为整型数。a=2, b=4, c=6, 是赋值语句,其中“=”号为赋值运算符,表示将 2 赋给变量 a, 4 赋给 b, 6 赋给 c, sum=a+b+c 也是一个赋值语句,表示将 a+b+c 之和赋给 sum。最后一行是输出语句,它将要输出: The sum of a+b+c is, 然后按 % d 指示符的规定(%d 规定输出十进制数)输出后面紧跟着的变量 sum 的值。这个例子的输出结果是:

The sum of $a+b+c$ is 12

在C语言中,常常可以用注释语句来提高程序的可读性。例1-3可以如例1-4那样来写。

例 1-4 C语言注释语句的使用。

```
/* calculate adds of three integer values
 * and displays the results: sum
 */
main()
{
    /* Declare variables */
    int a, b, c, sum;
    a=2;
    b=4;
    c=6;
    sum=a+b+c; /* compute the result */

    /* Display the results */
    printf("The sum of a+b+c is %d\n", sum);
}

/* * * * * *
 * output: The sum of a+b+c is 12
 * * * * *
 */
```

C语言的注释语句由/*两个字符开始,由*/两个字符结束。在/*...*/之间的任何字符,都将被C编译系统忽略。注释语句可以写在C语言程序的任何地方。程序的头部,尾部,或中间,甚至在某一行中,例如在 $sum=a+b+c$;之后也写了一个注释行:
/* Compute the Result */。