



普通高等院校
计算机优质平台课系列教材

C语言程序设计

C YUYAN
CHENGXU SHEJI

主 编 胡泉 谢芳
副主编 闫朝晖 谢家荣

华中科技大学出版社
<http://www.hustp.com>



普通高等院校
计算机优质平台课系列教材

C 语言程序设计

C YUYAN

· CHENGXU SHEJI

主 编 胡泉 谢芳
副主编 闫朝晖 谢家荣

华中科技大学出版社

中国·武汉

图书在版编目(CIP)数据

C 语言程序设计/胡 泉 谢 芳 主编. —武汉:华中科技大学出版社,2009 年 4 月
ISBN 978-7-5609-5206-2

I. C… II. ①胡… ②谢… III. C 语言-程序设计-高等学校-教材 IV. TP312

中国版本图书馆 CIP 数据核字(2009)第 037981 号

C 语言程序设计

胡 泉 谢 芳 主编

策划编辑:徐晓琦

封面设计:刘 卉

责任编辑:徐晓琦

责任监印:周治超

责任校对:张 琳

出版发行:华中科技大学出版社(中国·武汉)

武昌喻家山 邮编:430074 电话:(027)87557437

录 排:武汉众心图文激光照排中心

印 刷:华中科技大学印刷厂

开本:787mm×1092mm 1/16

印张:17

字数:410 000

版次:2009 年 4 月第 1 版

印次:2009 年 4 月第 1 次印刷

定价:29.80 元

ISBN 978-7-5609-5206-2/TP·677

(本书若有印装质量问题,请向出版社发行部调换)

内 容 简 介

本书介绍了计算机高级语言 C 语言的基础知识和语法规则,以及应用 C 语言设计各种实用程序的方法。其主要内容包括:第 1 章程序设计基础知识,第 2 章 C 语言程序设计,第 3 章数据类型,第 4 章基本算法,第 5 章顺序结构,第 6 章选择结构,第 7 章循环结构,第 8 章存储类型与数组,第 9 章函数文件与编译预处理,第 10 章指针基础,第 11 章指针与函数,第 12 章结构体与联合体,第 13 章文件。全书概念清楚、内容丰富,并且每章配有习题,便于教学和自学。

本书可以作为高等院校计算机专业本、专科学生初学计算机高级语言程序设计的教材,或其他各专业计算机公共课的教材,也可作为计算机等级考试二级 C 语言的考前培训教材。

前 言

C 语言是一种通用的高级程序设计语言,它以丰富的数据类型、完善的功能特征和快速的编译环境等特点吸引着众多的计算机工作者,同时也深受广大计算机语言学习者的喜爱。

目前,绝大多数高等院校的计算机专业都将 C 语言作为培养学生程序设计能力的首选程序语言,许多院校的非计算机专业也将 C 语言程序设计作为公共平台课程开设,而在竞争激烈的人才市场,许多 IT 企业招收计算机软件人员时也会将 C 语言程序设计作为笔试或面试的重要考核内容。

鉴于 C 语言的基础性和普遍应用性,通常它都会作为第一门程序设计课程安排在第一学期或者第二学期开设。由于在高级语言程序设计课程中给定了具体的语言(多数为 C 语言),不少教师不自觉地将教授“程序设计”变成了教授“一种高级语言的程序编写”,从而过多地强调语言的语法和语义。经验表明,这种做法容易对学生产生误导,使学生片面地认为学计算机就是编程序,导致一开始就扎进程序设计的语言细节中去,而忽视了数据结构、算法等问题。更重要的是,由于没能使学生在头脑中形成如何利用计算机进行“计算”的概念,学生往往在需要用程序求解问题时,“满脑袋都是语句,却不知道应该写哪个语句”;当需要到机器上实际操作时,却只会照着课本“敲”例题,“一分一毫”也不敢错。造成这些现象的主要原因就是把“程序设计”课讲授成了“一种高级语言的程序编写”课。

所以在教学过程中,应把重点放在程序设计实践以及培养学生分析问题和解决问题的能力上。首先使学生养成“程序设计”的习惯,初步掌握程序设计的基本方法和思想(如抽象的思维方法和自顶向下的设计思想);在有了这些基础以后,再引导他们用具体的高级语言进行“程序编写”。而对于学习者而言,具备了真正的“程序设计”基础,不仅能为今后顺利地更换程序设计语言提供支撑,而且也在学习后续的数据结构与算法课程、理解典型系统软件的设计、掌握有关的设计方法打下良好的基础。为此,我们以极大的热情编写了这本教材,书中在程序设计的思想上力求多一点分析、多一点启发、多一点引导、多一点设计思路上的提示。

本书内容丰富,概念清楚,每章配有练习题,便于教学和自学。全书共分为 13 章,从程序设计和算法的基础知识出发,循序渐进地介绍 C 语言程序设计的规则、方法和相关技术。由于各种原因,书中错误在所难免,敬请读者批评指正。

编 者
2008 年 12 月

目 录

第 1 章 程序设计基础知识	(1)
1.1 计算机的工作原理	(1)
1.2 程序语言与程序设计	(2)
习题	(4)
第 2 章 C 语言程序设计概论	(5)
2.1 C 语言的发展及其特点	(5)
2.2 C 语言的程序结构	(7)
2.3 C 程序的运行环境	(10)
习题	(12)
第 3 章 数据类型、运算符与表达式	(14)
3.1 C 语言的数据类型	(14)
3.2 常量与变量	(15)
3.3 C 语言的基本数据类型	(18)
3.4 运算符与表达式	(22)
3.5 简单程序设计与常见错误列举	(26)
习题	(28)
第 4 章 基本算法	(31)
4.1 程序与算法	(31)
4.2 算法的特征	(31)
4.3 算法的表示算法	(32)
4.4 程序设计的基本算法	(35)
习题	(38)
第 5 章 顺序结构	(39)
5.1 C 语言的程序语句	(39)
5.2 数据的输入和输出	(41)
5.3 顺序结构程序设计	(47)
习题	(48)

第 6 章 选择结构	(50)
6.1 关系运算和逻辑运算	(50)
6.2 选择结构程序设计	(55)
6.3 选择结构语句	(56)
6.4 常见错误列举	(67)
习题	(68)
第 7 章 循环结构	(71)
7.1 while 型循环	(71)
7.2 do-while 型循环	(73)
7.3 for 型循环	(75)
7.4 循环嵌套	(78)
7.5 break 和 continue 语句	(81)
7.6 goto 语句和标号语句	(86)
7.7 各种循环语句的比较	(88)
习题	(89)
第 8 章 存储类型与数组	(92)
8.1 变量的存储类型与作用范围	(92)
8.2 一维数组	(96)
8.3 二维数组与多维数组	(102)
8.4 字符串与字符数组	(110)
8.5 常见错误举例	(118)
习题	(120)
第 9 章 函数与编译预处理	(123)
9.1 C 语言函数	(123)
9.2 函数的定义与调用	(126)
9.3 函数间数据的传递方式	(133)
9.4 函数的嵌套调用和递归调用	(141)
9.5 函数的存储类型	(146)
9.6 编译预处理	(147)
习题	(155)
第 10 章 指针	(158)
10.1 指针的基本概念	(158)
10.2 变量的指针和指向变量的指针变量	(159)
10.3 指针与数组	(169)

10.4	指针和字符串	(180)
10.5	指针数组	(185)
习题		(190)
第 11 章	指针与函数	(193)
11.1	指针参数	(193)
11.2	指针函数	(195)
11.3	函数指针	(197)
11.4	多级指针	(201)
11.5	动态内存管理	(205)
11.6	命令行参数	(208)
习题		(209)
第 12 章	结构体与联合体	(213)
12.1	结构体	(213)
12.2	结构型指针与函数	(220)
12.3	线性链表	(227)
12.4	联合体	(232)
12.5	枚举类型变量的定义和引用	(235)
12.6	typedef 的使用	(239)
习题		(240)
第 13 章	文件	(244)
13.1	文件的基本概念与分类	(244)
13.2	文件结构体与文件指针	(246)
13.3	文件的打开、读/写与关闭	(247)
13.4	文件的定位	(257)
13.5	非缓冲文件系统文件的基本操作	(259)
习题		(262)
参考文献		(264)

第 1 章

程序设计基础知识

随着科学技术的迅猛发展,计算机技术日新月异,计算机程序设计语言也层出不穷。那么,什么是程序语言?什么是程序设计?应该学哪一种程序语言?如何进行程序设计?这些都是程序设计初学者首先遇到的问题,也是程序设计的基本问题和共性问题。

本章提要

- 程序设计的基本知识
- 计算机的工作原理
- 系统开发的相关知识
- 程序设计的关键要素

1.1 计算机的工作原理

1.1.1 计算机的指令系统

计算机的存储器由千千万万的电子线路单元组成,每个单元有两个稳定的工作状态(例如,二极管或三极管的截止和导通,磁性元件的消磁和充磁等),分别以 0 和 1 表示,因此计算机中的信息是以二进制形式存储的。人们要计算机处理信息,就要给计算机指定一些最基本的操作,并用 0 和 1 表示这些操作,这就构成一条一条的指令。一台计算机所能执行的各种不同类型指令的总和称为指令系统(Instruction Set)。不同型号的计算机,其指令系统也不相同。

一条指令由操作码(Opcode)和操作数(Oprand)两部分构成,例如,有这样一条指令:

11000110

操作码

00000110

操作数

指令中,操作码 11000110 表示加法操作,操作数是 00000110。这条指令的功能是把操作数 00000110 与计算机累加器中的数相加,相加的和仍放在累加器中。例如,先在累加器中放一个数 00000101,则计算机执行这条指令的过程如图 1-1 所示。这条指令用十六进制表示为 C6 06。

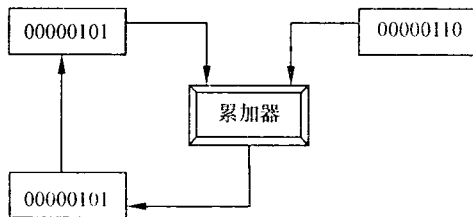


图 1-1 指令执行过程示意图

1.1.2 计算机的求解过程

计算机的求解是由操作人员事先告诉它“求解”的方法和过程,再由机器一步一步地去执行的。如果输入的求解过程是正确的,则能输出正确的结果;否则,计算机无法得到正确结果,甚至会陷入死循环等状态。

下面以求解算式 $7+8$ 作为例子,将计算机的解题步骤归纳如下。

- (1) 把二进制数 7 和 8 送到计算机的内存中存放起来,例如,46 号地址存放 7,写为 (46)←7,47 号地址存放 8,写为 (47)←8;
- (2) 把数 7 取出来,送到累加器;
- (3) 把数 8 取出来,与累加器中的数相加,结果放在累加器中;
- (4) 把累加器中的结果送回到内存的 48 号地址存放起来,即 (48)←15;
- (5) 把结果输出到打印机或显示器上;
- (6) 结束操作。

上述解题步骤的集合称为程序,其中,第 1 步是数据的输入,第 5 步是数据的输出,第 2~4 步是计算机内部的处理。由机器指令构成的有序集合,称为机器语言程序。计算机的工作就是按规定顺序执行这些程序。人们要使用计算机进行运算,就要为它编制程序,称为程序设计。

1.1.3 存储程序原理

通过指令和程序的概念就可以了解计算机的工作原理。现代计算机的工作原理基于冯·诺依曼提出的存储程序的思想,即首先把程序和数据通过输入设备送入内存,然后从第一条指令开始,一条一条地执行程序。计算机每执行一条指令,都需要经过三个步骤:第一步,把指令从内存中取出并送往译码器,称为取指;第二步,译码器把指令分解成操作码和操作数,并产生相应的控制信号送往各电器部件;第三步,各部件执行相应的操作。这些过程都是由电子线路来控制的,从而实现计算机自动连续的工作。

1.2 程序语言与程序设计

1.2.1 计算机语言

给计算机编写程序必须使用计算机语言。多年来,计算机语言已经从机器语言进化到自然语言,计算机语言的发展史如图 1-2 所示。

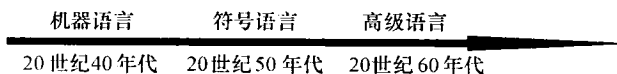


图 1-2 计算机语言的进化

在计算机历史的早期,可用的语言仅有机器语言,每种类型的计算机都有自己的机器语言,它由 0 和 1 构成的二进制串组成。计算机硬件可以直接识别机器语言,不需要编译器来翻译。

如果程序员使用机器语言进行工作,那么对于不同的类型的计算机,就要编制不同的程

程序代码。20世纪50年代早期,数学家 Admiral Grace Hopper 提出了使用符号语言和将符号语言程序转换为机器语言程序的计算机编程思想。这些早期的程序语言使用符号和助忆符来表示机器语言指令,称为符号语言。由于计算机不能直接理解符号语言,所以需要使用汇编器将符号代码翻译为机器语言,因此符号语言又称为汇编语言。

虽然使用符号语言可以大大提高编程效率,但它仍然需要程序了解运行的硬件环境。使用符号语言时,每一条机器指令都必须单独编码。提高程序员的效率和将焦点从计算机转移到求解问题的期望导致了高级程序语言的诞生。

高级程序语言可以移植到不同类型的计算机,它允许程序员专注于应用问题而不是计算机本身错综复杂的性质,使程序员从汇编语言的细节中解放出来。然而,高级语言和符号语言有其共同之处,即它们在运行时必须转换为机器语言,转换的过程称为编译。

1.2.2 程序设计

所谓程序,就是用计算机语言对所要解决的问题中的数据以及处理问题的方法和步骤所作的完整而准确的描述,实现这个描述的过程称为程序设计。对数据的描述就是指明数据结构;对处理方法和步骤的描述就是确定算法。因而,数据结构和算法是程序设计过程中不可或缺的两个方面。

对于程序设计的初学者来说,首先要学会设计一个正确的程序。一个正确的程序通常包括两个含义:一是书写正确,二是结果正确。书写正确是指程序在语法上正确,符合程序语言的规则;而结果正确通常是指对应于正确的输入程序能产生所期望的输出,且符合使用者对程序功能的要求。编制出正确的程序仅仅是程序设计的最低要求,对于优秀的程序员,除了程序的正确性以外,更要注重程序的高效性。

进行程序设计主要包括下面四个步骤。

(1) 分析问题,建立数学模型。使用计算机解决具体问题时,首先要对问题进行充分的分析,确定问题是什么,解决问题的步骤又是什么;针对所要解决的问题,找出已知的数据和条件,确定所需的输入对象、处理对象及输出对象;然后将解题过程归纳为一系列的数学表达式,建立各种量之间的关系,即建立起解决问题的数学模型。

(2) 确定数据结构和算法。根据建立的数学模型,为指定的输入数据和预期的输出结果确定存放数据的数据结构,并针对所建立的数学模型和确定的数据结构,选择合适的算法加以实现。

(3) 编制程序。用具体的程序语言把问题的解决方案描述出来,也就是编写出程序代码。

(4) 调试程序。用实际的输入数据对程序进行测试,并分析所得的运行结果,再进行程序调试,直到获得预期的结果为止。

1.2.3 程序设计的关键要素

不论使用何种语言进行程序设计,必须注意以下几个方面的问题。

(1) 确认对需求的理解正确无误,并且在开始编码之前要多花时间设计求解方案。

(2) 熟悉系统的编译器,包括编译器命令和键盘快捷键,这样可以高效地创建和编辑程序。

(3) 阅读编译器的错误信息,熟悉错误类型对学习 C 语言有很大的帮助。

(4) 记得对每次修改的源文件都进行保存和编译。

(5) 使用不同的数据集合多次运行程序以保证其确实完成了所期望的任务。

【本章小结】 本章主要介绍了程序设计的基础知识,包括计算机的工作原理、计算机程序设计的方法、计算机语言的发展史以及程序设计的主要步骤。通过本章的学习,读者对程序设计语言的地位和重要性应该会有初步的了解。

习 题

1. 简述计算机存储程序原理。
2. 简述计算机语言的发展史。
3. 计算机程序设计的基本步骤有哪些?
4. 程序设计的关键要素有哪些?

第 2 章

C 语言程序设计概论

C 语言是目前世界上最流行、使用最广泛的计算机高级语言之一。在设计操作系统等系统软件和需要对硬件进行操作时,C 语言明显优于其他计算机语言;而许多大型应用软件也都是用 C 语言编写的。

本章提要

- C 语言的发展和特点
- C 语言的结构
- C 语言的上机步骤与环境
- C 语言的数据输入和输出

2.1 C 语言的发展及其特点

2.1.1 C 语言的发展

C 语言的起源可以追溯到 1960 年出现的 ALGOL 60 语言。1963 年,英国的剑桥大学在 ALGOL 60 语言的基础上推出了 CPL(Combined Programming Language)语言,但是 CPL 语言复杂而难以实现。1967 年,剑桥大学的 Martin Richards 对 CPL 语言进行了简化,推出了 BCPL 语言。1970 年,美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础,对其作了进一步的简化,设计出了很简单且接近硬件的 B 语言,并用 B 语言写了第一个 UNIX 操作系统。1972 年,美国的 Dennis M. Ritchie 在 B 语言的基础上设计出了 C 语言,并首次在 UNIX 操作系统的 DEC PDP-11 计算机上使用。为了推广 UNIX 操作系统,1977 年,Dennis M. Ritchie 发表了不依赖于具体操作系统的可移植的 C 语言编译程序。1978 年,Brian W. Kernighan 和 Dennis M. Ritchie 出版了《The C Programming Language》一书,从而使 C 语言成为世界上最流行的高级程序设计语言之一。

随着计算机技术的日益普及,C 语言出现了许多版本。由于没有统一的标准,这些版本之间出现了许多不一致的地方。为了改变这种状况,1983 年,美国国家标准化协会(ANSI)为 C 语言制定了第一个 ANSI 标准,称为 ANSI C。1987 年,ANSI 又公布了新的 C 语言标准,称为 87 ANSI C,这个标准于 1989 年被国际标准化组织(ISO)采用,所以又称为 ANSI/ISO Standard C(即 C89)。目前国际上流行的 C 编译系统都是以这个标准为基础的。

C 语言已经成为全球程序员所使用的公共编程语言,并由此产生了当前的两个主流程

序设计语言 C++ 和 Java——它们都是建立在 C 语言的语法和基本结构基础上的。由于现在世界上的许多软件都是在 C 语言及其衍生的各种语言的基础上开发而成的,而 C 语言也是学习 C++ 和 Java 语言的基础语言,所以每一个大学生都应该努力学会用 C 语言编写程序。

2.1.2 C 语言的主要特点

一种语言之所以能存在和发展,是由于其有着不同于或者优于其他语言的特点。C 语言的主要特点可归纳为以下七点。

(1) 语言简洁、紧凑,使用方便、灵活。C 语言一共只有 32 个关键字,9 种控制语句,程序书写形式自由,主要用小写字母表示,压缩了一切不必要的成分。

(2) 运算符丰富。C 语言共有 34 种运算符,涉及的范围十分广泛。C 语言把括号、赋值、强制类型转换等都作为运算符来处理,从而可以实现在其他高级语言中难以实现的运算。

(3) 数据结构丰富,具有现代化语言的各种数据结构。C 语言的数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型,等等,能用来实现各种复杂数据结构(如链表、树、堆栈等)的运算。尤其是指针类型数据,使用起来比其他语言(如 PASCAL)更为灵活、多样。

(4) 结构化。C 语言是一种结构化的程序设计语言,它层次清晰,便于按模块化方式组织程序,易于调试和维护。例如,它具有 if-else 语句、while 语句、do-while 语句、switch 语句、for 语句等结构化的控制语句;还可用函数作为程序的模块单位,从而方便地实现程序的模块化。

(5) 可对硬件进行操作。C 语言允许直接访问物理地址,能进行位(bit)操作,能实现汇编语言的大部分功能,可直接对硬件进行操作。因此,C 语言既具有高级语言的功能,又具有低级语言的许多功能,可用来编写系统软件。C 语言的这种双重性质使它既是成功的系统描述语言又是通用的程序设计语言。有人把 C 语言称为“高级语言中的低级语言”或“中级语言”,就是因为它兼有高级和低级语言的特点。

(6) 生成的目标代码质量高,程序执行效率高。C 语言生成的目标代码效率一般只比汇编程序低 10%~20%。

(7) 可移植性好。与汇编语言相比,C 语言程序具有更好的移植性,基本上不用修改就能用于各种型号的计算机和操作系统。

虽然 C 语言具有以上优点,但它也不是完美无缺的,其缺点主要表现在:变量类型的转换比较随便、不够安全,例如 C 语言程序中字符型数据与整型量可以通用;C 语言编译系统对程序的语法检查不太严格,例如对数组的下标越界不作检查;另外,C 语言中运算符的优先级太多,容易混淆,程序的正确性要由程序编制者自己来判断保证。

目前,在微机上广泛使用的 C 语言编译系统有 Turbo C、Borland C++、Microsoft Visual C++ 等。虽然它们的基本部分都相同,但还是有一些细微的差异。本书采用 Turbo C 作为上机编程调试环境。

2.2 C语言的程序结构

2.2.1 基本的C语言程序结构

例 2.1 C语言实现输出字符串“Welcome to C world!”。

```
#include <stdio.h>
void main( )                      /* 主函数 */
{
    printf("Welcome to C world!\n"); /* 调用标准函数,显示引号中的内容 */
}
```

这是一个最简单的C程序,其执行结果是在屏幕上显示一行信息:

Welcome to C world!

程序第一行将头文件 `stdio.h` 嵌入到本程序中,以便程序调用系统提供的输入、输出函数;第二行到第五行是主函数(main 函数),第三行的左花括号表示函数体开始,第五行的右花括号表示函数体结束;函数体中有一个输出函数(`printf` 函数),它是系统定义的标准函数,可在程序中直接调用,其功能是把要输出的内容送到显示器显示,其后双引号中的内容就是要输出的字符串,`\n` 则表示换行。

例 2.2 求两个数的和。

```
#include <stdio.h>
void show(int x,int y);          /* (自定义)函数原型声明 */
void main( )                     /* main 函数定义 */
{
    int a,b;
    a=1; b=2;
    show(a,b);
}
void show(x,y)                   /* (自定义)函数定义 */
int x,y;
{
    int sum;
    sum=x+y;
    printf("a add b is %d \n",sum);
}
```

本程序的功能是计算两个数 `a`、`b` 的和,并输出结果。程序由主函数和 `show` 函数组成,且从主函数中调用 `show` 函数,`show` 函数的主要功能是计算并显示 `a`、`b` 两数的和。由于 `show` 函数是一个用户自定义函数,因此在主函数调用它之前要进行函数原型声明。具体的函数使用方法将在后面章节详细讲到。

以上程序的输出结果为

a add b is 3

由上面两个例子可以看出C语言程序的结构有如下特点。

- (1) 一个C语言程序可以由一个或多个源文件组成。
- (2) 每个源文件可由一个或多个函数组成。

(3) 一个程序不论由多少个文件组成,都有一个且只能有一个 main 函数,即主函数。任何一个完整的 C 语言程序都必须有这个主函数,不管这个主函数出现在程序中的什么位置,C 语言程序的执行都从主函数开始。

(4) 程序中可以有预处理命令(include 命令仅为其中的一种),预处理命令通常放在源文件或程序的最前面。

(5) 每一个语句都必须以分号结尾,但预处理命令、函数头和花括号之后不能加分号。

(6) 标识符、关键字之间必须加至少一个空格以示间隔。若已有明显的间隔符,也可不再加空格来间隔。

(7) 程序中可以用“/ * * /”作注释,其格式为:/ * 注释内容 * /。应当注意的是,在“/”和“*”之间不允许插入空格或其他字符。注释的内容可以用英文、汉字或汉语拼音书写,通常是关于程序的功能、算法或者某些注意事项等方面的说明;注释语句的位置可以放在程序的开头,也可以放在某一行的后面,或者在两行之间单独占用一行。在程序中插入注释的目的是帮助阅读和理解程序,增强程序的可读性。注释在编译时不产生目标代码,仅作为源程序的一部分留在源程序清单中。

2.2.2 C 语言的书写规则

为了养成良好的编程习惯和编程风格,在编写 C 语言程序时应力求遵循以下规则。

(1) 一般情况下,一个说明或一条语句占一行。

(2) 用{ }括起来的部分通常表示程序的某一层结构,所以{ }一般与该层结构语句的第一个字母对齐,并单独占一行。

(3) 低一层次的语句可比高一层次的语句或说明缩进若干格后书写,以便程序层次看起来更加清晰,增加程序的可读性。

2.2.3 C 语言的字符集

字符是组成语言最基本的元素。C 语言的字符集由字母、数字、空格、标点和特殊字符组成。在字符常量、字符串常量和注释中还可以使用汉字或其他可表示的图形符号。

(1) 字母:小写字母 a~z 共 26 个,大写字母 A~Z 共 26 个。

(2) 数字:0~9 共 10 个。

(3) 空白符:空格符、制表符、换行符等统称为空白符。空白符只在字符常量和字符串常量中起作用,在其他地方出现时只起间隔作用,编译程序将忽略它们。因此,在程序中是否使用空白符,对程序的编译不产生影响,但在程序中适当的地方使用空白符可以增加程序的清晰性和可读性。

(4) 标点符号:如'、"、;、:、。

(5) 特殊符号:如\、_、\$、#。

2.2.4 C 语言的词汇

在 C 语言中使用的词汇可分为六类:标识符、关键字、运算符、分隔符、常量和注释符。

(1) 标识符。

在程序中使用的变量名、函数名称为标识符。C 语言规定,标识符只能是字母(A~Z,

a~z)、数字(0~9)、下划线(_)组成的字符串,并且第一个字符必须是字母或下划线。一组合法及非法的标识符如表 2-1 所示。

表 2-1 标识符的命名举例

合法标识符	非法标识符
x	3s(以数字开头)
_3x	s * T(出现非法字符*)
BOOK_1	-3x(以减号开头)
sum5	bowy-1(出现非法字符-)

在使用标识符时必须注意以下几点。

① 标准 C 不限制标识符的长度,但它受各种版本的 C 语言编译系统的限制,同时也受机器种类的限制。例如,在某版本 C 语言中规定标识符的前 8 位有效,则当两个标识符前 8 位相同时会被认为是同一个标识符,此时 sumx123456 和 sumx12349876 将是同一个标识符。

② 在标识符中大小写是有区别的。例如,BOOK 和 book 是两个不同的标识符。

③ 标识符虽然可由程序员随意定义,但标识符是用于标识某个量的符号,因此其命名应尽量有对应的意义,以便程序的阅读理解,做到“顾名思义”。

(2) 关键字。

关键字是由 C 语言规定的具有特定意义的字符串,通常也称为保留字。用户定义的标识符不应与关键字相同。C 语言的关键字分为以下几类。

① 类型说明符 用于定义、说明变量、函数或其他数据结构的类型,如 int、double 等。

② 语句定义符 用于表示一个语句的功能,如 if else 就是条件语句的语句定义符。

③ 预处理命令字 用于表示一个预处理命令,如前面各例中用到的 include。

(3) 运算符。

C 语言中含有相当丰富的运算符。运算符与变量、函数一起组成表达式,表示各种运算功能。运算符由一个或多个字符组成。具体的运算符将在第 3 章中详细介绍。

(4) 分隔符。

在 C 语言中采用的分隔符有逗号和空格两种。逗号主要用在类型说明和函数参数表中分隔各个变量;空格多用于语句各单词之间作为间隔符。在关键字、标识符之间必须要有一个以上的空格符作间隔,否则将会出现语法错误。例如,把 int a 写成 inta,C 编译器就会把 inta 当成一个标识符处理,其结果必然出错。

(5) 常量。

C 语言中使用的常量可分为数字常量、字符常量、字符串常量、符号常量、转义字符等多种。在第 3 章中将专门给予介绍。

(6) 注释符。

C 语言的注释符是以“/ * ”开头并以“ * / ”结尾的串。在“/ * ”和“ * / ”之间的即为注释的内容。程序编译时,不对注释作任何处理。注释可出现在程序中的任何位置,它用来向用户提示或解释程序的意义。