

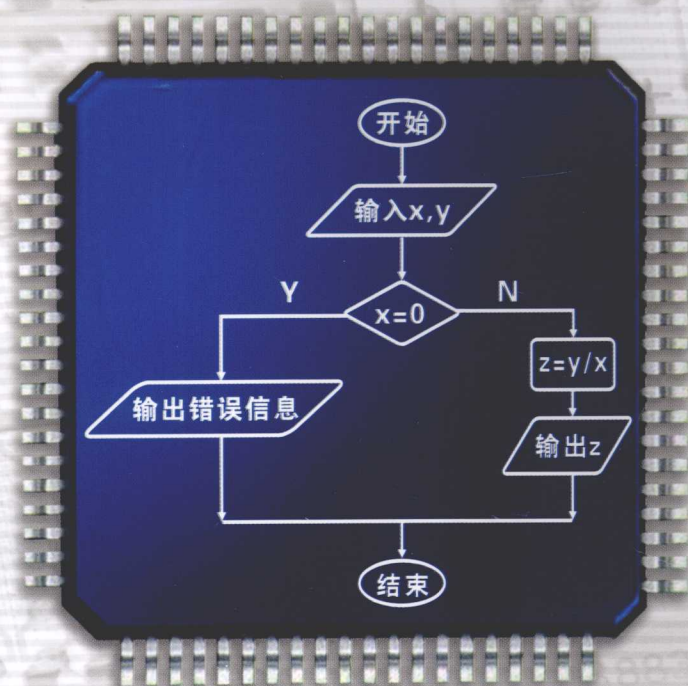


普通高等教育计算机类专业“十三五”规划教材

# C语言程序设计

## (第2版)

许大炜 陆丽娜 缪相林 毕鹏 丁凰 赵彩 编



西安交通大学出版社  
XI'AN JIAOTONG UNIVERSITY PRESS



普通高等教育计算机类专业“十三五”规划教材

内容简介

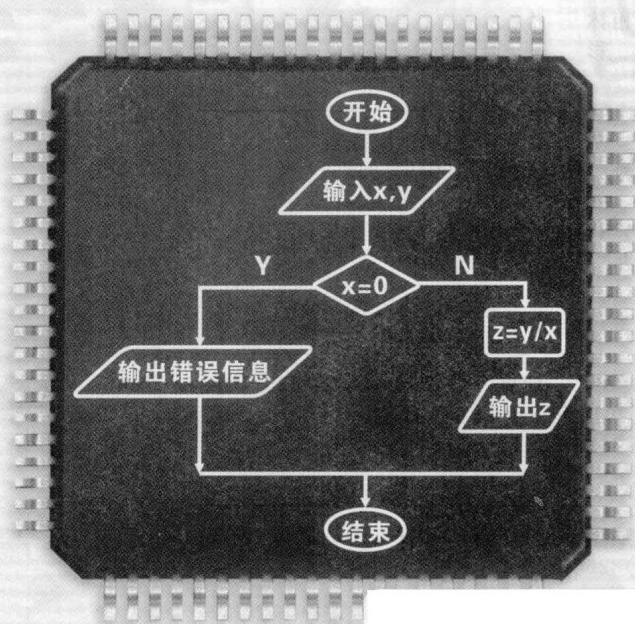
本书是普通高等教育计算机类专业“十三五”规划教材，是根据教育部颁布的《计算机组成原理》课程教学大纲，结合编者多年从事该课程的教学和科研工作的经验编写而成的。本书可作为高等院校计算机专业及相关专业的教材，也可供从事计算机工作的工程技术人员参考。

本书共分8章。第1章介绍计算机的发展概况、分类及应用；第2章介绍运算方法和运算指令；第3章介绍中央处理器(CPU)的组成、指令系统、指令格式及指令的执行过程；第4章介绍总线系统的组成、总线协议及总线接口；第5章介绍指令流水线的组成、流水线的性能及流水线的冒险；第6章介绍Cache的组成、Cache的访问及Cache的替换算法；第7章介绍多处理器的组成、多处理器的访问及多处理器的同步；第8章介绍并行处理的组成、并行处理的访问及并行处理的同步。

# C语言程序设计

## (第2版)

许大炜 陆丽娜 缪相林 毕鹏 丁凰 赵彩 编



西安交通大学出版社  
XI'AN JIAOTONG UNIVERSITY PRESS

## 内容简介

本教材针对应用型人才培养目标,从学生思维方式、理解能力及后续课程中的应用诸方面因素出发编写。全书分为九章,主要内容包括:C 语言程序设计基础,数据类型、运算符及表达式,数据的输入输出,程序控制结构,数组、函数、指针、结构体与共用体和文件操作等。

本教材在结构上突出了以程序设计为中心,以语言知识为工具的思想,并介绍它们在程序设计中的应用;在内容上注重知识的完整性,适合初学者的需要;在写法上追求循序渐进,通俗易懂。本教材配有教学参考书《C 语言程序设计——实验指导·课程设计·习题解答》(第 2 版),以方便读者深入学习和上机操作。

本教材既可以作为高等学校本科及专科学生 C 语言程序设计的教材,又可以作为自学者的参考用书,同时也可供各类考试人员复习参考。

### 图书在版编目(CIP)数据

C 语言程序设计/许大伟等编. —2 版. —西安:  
西安交通大学出版社, 2015. 8  
ISBN 978-7-5605-7650-3

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

中国版本图书馆 CIP 数据核字(2015)第 162742 号

书 名 C 语言程序设计(第 2 版)  
编 者 许大伟 陆丽娜 缪相林  
毕 鹏 丁 凰 赵 彩  
责任编辑 屈晓燕

出版发行 西安交通大学出版社  
(西安市兴庆南路 10 号 邮政编码 710049)  
网 址 <http://www.xjtpress.com>  
电 话 (029)82668357 82667874(发行中心)  
(029)82668315(总编办)  
传 真 (029)82668280  
印 刷 陕西宝石兰印务有限责任公司

开 本 787mm×1092mm 1/16 印张 18.125 字数 435 千字  
版次印次 2015 年 8 月第 2 版 2015 年 8 月第 1 次印刷  
书 号 ISBN 978-7-5605-7650-3/TP·676  
定 价 38.00 元

读者购书、书店添货、如发现印装质量问题,请与本社发行中心联系、调换。  
订购热线:(029)82665248 (029)82665249  
投稿热线:(029)82664954  
读者信箱:jdgy@yahoo.cn

版权所有 侵权必究

# 前 言

要真正地掌握软件开发的艺术,首先至少要掌握一种程序设计语言,在众多的程序设计语言中,C语言以其灵活性和实用性,受到了广大计算机应用人员的喜爱。近年来,即使出现了C++、Java、C#等语言,也没有动摇C语言的基础地位。C语言几乎具备现代程序设计语言的所有语言成分,一旦掌握了C语言,就可以较为轻松地学习其他任何一种程序设计语言。

程序设计既是一种技术,也是一项工程。作为一本程序设计教材,不仅要介绍关于C语言的基本语法知识,还要强调程序设计思想方法的培养,并且着眼于应用现代软件工程的思想进行程序开发能力的训练。如何解决好这三个方面的衔接,将它们有机地结合起来,是当前程序设计教材需要解决的一个重要问题,也是一个难点问题。本书第二版在第一版的基础上,根据用户的反馈及在教学过程中得到的经验,我们重写了部分章节,并对其他内容进行了大量的修改和补充,以力图体现以下几点:

1. 讲解C语言最基本、最常用的内容,重点主要放在语言本身的难点和程序设计思想、技巧方面;各章节之间密切结合并且以阶梯式前进,使读者在学习的过程中能够循序渐进。

2. 通过第9章的综合举例提供一个完整的软件系统开发的过程,将每一章的知识点融入其中,使读者学以致用,最终能够开发出一个学生信息管理系统,并且能够得到设计一个软件的实践经验。

3. 本书中使用结构化的流程图表示方法有助于理解和掌握模块化的程序设计思想和控制结构。

4. 每一章的小结、技术提示和编程经验主要总结了一些易错的概念和知识点。介绍一些软件开发的编程经验,使读者在学习的过程中就能领悟到高质量的C语言编程知识。

5. 本书新加入了部分计算机等级考试的经典试题,使学生学完以后,不仅能够掌握C语言的理论知识,具备一定的实践能力,同时又能够为参加计算机等级考试打下基础。

本书由西安交通大学城市学院许大炜、陆丽娜、缪相林、毕鹏、丁凰和赵彩老师编写。感谢西安交通大学出版社编辑多次组织我们讨论如何编写适合应用型人才的教材,给了我们许多的启发。感谢西安交通大学杨麦顺老师对本书提出的很好的建议。感谢西安交通大学城市学院领导对我们的关心、支持和帮助。

欢迎使用本书进入美妙的C语言世界。C语言博大精深,本书只能从实用易懂的角度去描述它,希望能给读者抛砖引玉,使读者尽可能地达到一种专业的编程境界。

由于笔者的水平和编程经验有限,加之时间比较紧促,本书尚有很多不足之处,希望能够得到专家和读者的指正。

编者

于西安交通大学城市学院

2015年7月

# 目 录

|                            |      |
|----------------------------|------|
| 第1章 C语言程序设计基础 .....        | (1)  |
| 1.1 程序设计与程序设计语言 .....      | (1)  |
| 1.1.1 程序与程序设计 .....        | (1)  |
| 1.1.2 程序设计语言 .....         | (3)  |
| 1.2 C语言简介 .....            | (4)  |
| 1.2.1 C语言出现的历史背景 .....     | (4)  |
| 1.2.2 C语言的基本特点 .....       | (5)  |
| 1.2.3 C语言的标准化 .....        | (6)  |
| 1.3 C语言程序设计简介 .....        | (6)  |
| 1.3.1 简单C语言程序的构成与格式 .....  | (6)  |
| 1.3.2 C语言程序的结构 .....       | (8)  |
| 1.3.3 良好的编程风格 .....        | (9)  |
| 1.4 运行C语言程序的步骤 .....       | (10) |
| 1.4.1 C语言程序的编辑、编译与运行 ..... | (10) |
| 1.4.2 使用语言编程注意要点 .....     | (11) |
| 1.5 小结 .....               | (12) |
| 1.6 技术提示 .....             | (12) |
| 1.7 编程经验 .....             | (13) |
| 习题 .....                   | (13) |
| 第2章 数据类型、运算符及表达式 .....     | (15) |
| 2.1 C语言基本字符、标识符和关键字 .....  | (15) |
| 2.1.1 C语言字符集 .....         | (15) |
| 2.1.2 标识符 .....            | (15) |
| 2.1.3 关键字 .....            | (16) |
| 2.2 常量与变量 .....            | (16) |
| 2.2.1 常量和符号常量 .....        | (17) |
| 2.2.2 变量 .....             | (17) |
| 2.3 数据类型与数据表示 .....        | (18) |
| 2.3.1 整型数据 .....           | (19) |
| 2.3.2 实型数据 .....           | (22) |

|                        |      |
|------------------------|------|
| 2.3.3 字符型数据            | (24) |
| 2.3.4 字符串常量            | (27) |
| 2.4 运算符及表达式            | (28) |
| 2.4.1 C语言运算符概述         | (28) |
| 2.4.2 算术运算符及其表达式       | (29) |
| 2.4.3 关系运算符及其表达式       | (30) |
| 2.4.4 逻辑运算符及其表达式       | (31) |
| 2.4.5 赋值运算符及表达式        | (33) |
| 2.4.6 自增1,自减1运算符       | (35) |
| 2.4.7 逗号运算符及其表达式       | (36) |
| 2.4.8 条件运算符及其表达式       | (37) |
| 2.4.9 位运算符及其表达式        | (38) |
| 2.4.10 运算中数据类型的自动和强制转换 | (40) |
| 2.5 小结                 | (41) |
| 习题                     | (42) |
| <b>第3章 数据的输入和输出</b>    | (45) |
| 3.1 字符数据的输出和输入         | (45) |
| 3.1.1 字符输出             | (45) |
| 3.1.2 字符输入             | (46) |
| 3.2 格式输出和输入            | (47) |
| 3.2.1 格式输出             | (47) |
| 3.2.2 格式输入             | (52) |
| 3.3 小结                 | (56) |
| 3.4 技术提示               | (56) |
| 3.5 编程经验               | (56) |
| 习题                     | (57) |
| <b>第4章 程序控制结构</b>      | (59) |
| 4.1 算法的基本概念            | (59) |
| 4.1.1 算法的概念与特征         | (59) |
| 4.1.2 算法的描述方法          | (60) |
| 4.1.3 算法应用举例           | (63) |
| 4.2 顺序结构               | (64) |
| 4.3 选择结构               | (67) |
| 4.3.1 if 语句            | (67) |
| 4.3.2 switch 语句        | (78) |

|                     |              |
|---------------------|--------------|
| 4.4 循环结构            | (80)         |
| 4.4.1 while 语句      | (80)         |
| 4.4.2 do-while 语句   | (82)         |
| 4.4.3 for 语句        | (85)         |
| 4.4.4 goto 语句       | (88)         |
| 4.4.5 循环的跳转和嵌套      | (88)         |
| 4.5 小结              | (92)         |
| 4.6 技术提示            | (92)         |
| 4.7 编程经验            | (92)         |
| 习题                  | (93)         |
| <b>第5章 数 组</b>      | <b>(100)</b> |
| 5.1 一维数组            | (100)        |
| 5.1.1 数组的基本概念       | (100)        |
| 5.1.2 一维数组的定义       | (101)        |
| 5.1.3 一维数组的引用和初始化   | (102)        |
| 5.2 二维数组            | (109)        |
| 5.2.1 二维数组的定义       | (109)        |
| 5.2.2 二维数组的引用和初始化   | (110)        |
| 5.3 字符数组和字符串        | (116)        |
| 5.3.1 字符数组的定义       | (117)        |
| 5.3.2 字符数组的引用和初始化   | (117)        |
| 5.3.3 字符串的定义        | (119)        |
| 5.3.4 字符串与字符数组的输入输出 | (120)        |
| 5.3.5 字符串的处理函数      | (121)        |
| 5.4 小结              | (126)        |
| 5.5 技术提示            | (126)        |
| 5.6 编程经验            | (126)        |
| 习题                  | (127)        |
| <b>第6章 函 数</b>      | <b>(133)</b> |
| 6.1 函数概述            | (133)        |
| 6.2 函数的定义和调用        | (136)        |
| 6.2.1 函数的定义         | (136)        |
| 6.2.2 函数的参数和返回值     | (138)        |
| 6.2.3 函数的声明         | (141)        |
| 6.2.4 函数的调用         | (143)        |

|            |                    |       |
|------------|--------------------|-------|
| 6.2.5      | 数组作为函数参数 .....     | (144) |
| 6.2.6      | 函数的嵌套和递归调用 .....   | (147) |
| 6.3        | 变量的作用域 .....       | (151) |
| 6.3.1      | 变量的作用域 .....       | (152) |
| 6.3.2      | 局部变量及其作用域 .....    | (152) |
| 6.2.3      | 全局变量及其作用域 .....    | (153) |
| 6.4        | 变量的存储类别及生命周期 ..... | (155) |
| 6.5        | 外部函数和内部函数 .....    | (160) |
| 6.5.1      | 外部函数 .....         | (160) |
| 6.5.2      | 内部函数 .....         | (161) |
| 6.6        | 编译预处理 .....        | (161) |
| 6.6.1      | 文件包含 .....         | (161) |
| 6.6.2      | 不带参宏定义 .....       | (163) |
| 6.6.3      | 带参的宏定义 .....       | (165) |
| 6.7        | 小结 .....           | (166) |
| 6.8        | 技术提示 .....         | (166) |
| 6.9        | 编程经验 .....         | (167) |
|            | 习题 .....           | (167) |
| <b>第7章</b> | <b>指 针</b> .....   | (175) |
| 7.1        | 指针和指针变量 .....      | (175) |
| 7.1.1      | 地址和指针的概念 .....     | (175) |
| 7.1.2      | 指针变量的定义和初始化 .....  | (177) |
| 7.1.3      | 指针变量的引用和运算 .....   | (179) |
| 7.2        | 指针和数组 .....        | (182) |
| 7.2.1      | 指针和一维数组 .....      | (182) |
| 7.2.2      | 指针和二维数组 .....      | (186) |
| 7.2.3      | 指针数组 .....         | (189) |
| 7.3        | 指针与字符串 .....       | (190) |
| 7.4        | 指针与函数 .....        | (192) |
| 7.4.1      | 指针变量作为函数参数 .....   | (192) |
| 7.4.2      | 指向函数的指针变量 .....    | (195) |
| 7.4.3      | 返回指针值的函数 .....     | (197) |
| 7.5        | 指向指针的指针 .....      | (199) |
| 7.6        | 指针与内存动态的分配 .....   | (201) |
| 7.7        | 小结 .....           | (204) |
| 7.8        | 技术提示 .....         | (205) |

|                            |       |
|----------------------------|-------|
| 7.9 编程经验 .....             | (205) |
| 习题 .....                   | (206) |
| <b>第 8 章 结构体与共用体</b> ..... | (211) |
| 8.1 结构体 .....              | (211) |
| 8.1.1 结构体的定义 .....         | (211) |
| 8.1.2 结构体变量的定义和初始化 .....   | (214) |
| 8.1.3 typedef 的使用方法 .....  | (218) |
| 8.1.4 结构体数组 .....          | (219) |
| 8.1.5 指向结构体的指针 .....       | (222) |
| 8.2 共用体 .....              | (223) |
| 8.2.1 共用体的定义 .....         | (223) |
| 8.2.2 共用体变量的定义和初始化 .....   | (224) |
| 8.3 枚举类型 .....             | (226) |
| 8.4 小结 .....               | (228) |
| 8.5 技术提示 .....             | (229) |
| 8.6 编程经验 .....             | (229) |
| 习题 .....                   | (230) |
| <b>第 9 章 文件操作</b> .....    | (235) |
| 9.1 文件概述 .....             | (235) |
| 9.1.1 文件 .....             | (235) |
| 9.1.2 文件的分类 .....          | (235) |
| 9.1.3 文件指针 .....           | (236) |
| 9.1.4 文件系统 .....           | (237) |
| 9.2 文件的打开和关闭 .....         | (238) |
| 9.2.1 文件的打开 .....          | (238) |
| 9.2.2 文件的关闭 .....          | (240) |
| 9.3 文件的读写 .....            | (240) |
| 9.3.1 字符输入输出函数 .....       | (240) |
| 9.3.2 文件字符串输入输出函数 .....    | (242) |
| 9.3.3 数据块输入输出函数 .....      | (243) |
| 9.3.4 格式化输入输出函数 .....      | (247) |
| 9.3.5 字输入输出函数 .....        | (248) |
| 9.4 文件的定位 .....            | (250) |
| 9.5 文件的检错 .....            | (251) |
| 9.6 C 语言库文件 .....          | (253) |

|                             |       |
|-----------------------------|-------|
| 9.7 综合举例 .....              | (254) |
| 9.8 小结 .....                | (263) |
| 9.9 技术提示 .....              | (263) |
| 9.10 编程经验 .....             | (263) |
| 习题 .....                    | (264) |
| 附录一 C99 标准和 C11 标准新特性 ..... | (268) |
| 附录二 头文件 .....               | (273) |
| 附录三 ASCII 表 .....           | (275) |
| 附录四 C 运算符和优先级 .....         | (277) |
| 参考文献 .....                  | (279) |

# 第 1 章 C 语言程序设计基础

什么是程序？什么是程序设计？什么是程序设计语言？本章首先介绍编程的基本概念，建立起对程序、程序设计、程序设计语言的基本认识。而后将简单介绍 C 程序设计语言，并通过简单实例介绍 C 语言的一些基本概念。

## 1.1 程序设计与程序设计语言

### 1.1.1 程序与程序设计

#### 1. 输入、处理和输出

计算机已应用到订票、商店、饭店和其他各种地方。在每个地方的使用都遵循类似的惯例。由人将一些数据输入计算机中，并由计算机生成结果，结果可显示在屏幕上，也可以打印在纸上。

让我们将航班订票系统整个动作系列分成一个个阶段。在第一阶段，要输入航班需求信息（包括目的地、起始日期和时间、舱的级别），这一阶段叫输入阶段。在最后一个阶段中，在屏幕上显示座位存在与否的信息，这一阶段叫做输出阶段。在上述两个阶段中间阶段，计算机处理由操作计算机的业务员输入的数值，并生成座位存在状态的输出，这一阶段叫处理阶段。因此，由计算机完成的活动的周期称为输入—处理—输出周期，或 I-P-O 周期。

#### 2. 程序和编程语言

(1) 程序。计算机是如何知道应该遵循哪些步骤来满足订票需求呢？它是如何计算出你在本商店的购物金额呢？它是如何生成你在学校的情况报告单呢？计算机是一台“什么都知道”的机器吗？不是的。看一看计算机是如何运作的，它被设计为具有接收输入、处理和产生输出功能的键盘、鼠标、显示器、打印机、中央处理单元和内存储器等组件来完成这些功能。然后，还必须给它一些列指令，指令中说明下列内容：

① 需提供的输入的类别。例如，起程日期、时间、舱位等级、目的地。

② 期望输出的类别。例如，座位存在与否的状态。

③ 需要进行的处理。例如，接收数值、检验座位存在与否、显示结果。

完成一个特定工作的一系列指令叫程序。因此，对于你要计算机完成的每项工作，都需要一个单独的程序，任何现实生活中的问题都包含许多小的工作。因此，要解决现实生活中的问题，你需要将许多小程序组合在一起，形成应用程序。例如，处理员工工资应用程序，其中需要完成生成付款核查和相关报表的任务，由单独的程序完成这些任务。

(2) 程序设计。计算机是一种通用的计算机器，加上一个或一组程序后，它就会变为处理

某个专门问题、完成某种特殊工作的专用机器。它还可以运行不同的程序,一台计算机可以在不同时候处理不同问题,甚至同时处理多个不同的问题。人们描述(编制)计算机程序的工作被称为程序设计或者编程,这种工作的产品就是程序。由于计算机的本质特征,从计算机诞生之初就有了程序设计工作。

程序设计一般包含以下几个部分:

①确定数据结构。根据任务提出的要求制定的输入数据和输出结果,确定存放设计的数据结构。

②确定算法。针对存放数据的数据结构来确定问题、完成任务的步骤。

③编码。根据确定的数据结构和算法,使用选定的计算机语言编写程序代码,输入到计算机中并保存到磁盘上。

④调试程序。检查编码中的错误。可以输入各种数据对程序进行测试,检查出程序中的错误,进行改进。

(3)编程语言。一般我们使用计算机高级语言来编写应用程序。高级语言有许多种,如 C、C++、Java 等,每种语言都有其优势和长处,你需要在决定使用哪种编程语言之前,必须对你的应用问题作出估计。

所有语言都有一个词汇表,其中给出了这种语言中有特定含义的单词列表。各种语言也都有它们的语法规则,其中说明了组词成句的规则。正是这些规则确保了不管用哪一种特定的语言讲述什么内容,所有懂这种语言的人的解释总是一样的。编程语言的词汇表,称为该语言的一些列关键字;编程语言也有语法,称为该语言的句法。

(4)编译器。计算机是否可以直接识别用高级语言编写的程序呢?不,它们无法做到这一点。它需要一个翻译,将用编程语言写的指令转换成计算机能够识别的机器语言。编译器是一种特殊的程序,它处理用一种特定的编程语言写的程序,并将它转换成机器语言。编译器也遵循 I-P-O 周期,它接收编程语言作为输入。然后,处理这些指令,将它们转换为机器语言。这些机器语言指令可以在计算机上执行,一次一句。这个转换过程称为编译。对于每种编程语言,都有不同的编译器,例如,编译用 C 语言写的程序需要 C 编译器,编译 Java 程序需要 Java 编译器。

### 3. 解决问题的技术

程序是解决一个特定问题的一系列指令。在正式开始写程序之前,我们先要拟定解决问题的过程。算法(algorithm)是解决问题所需的一系列步骤。描述算法常用的有两种方法:伪代码和流程图。

(1)伪代码。伪代码是用通俗易懂的语言表达的算法。在程序开发过程中,伪代码作为一个起始步骤。它为程序员提供了用一种特定的语言编写指令的详细模板。

伪代码不仅详细,而且可读性强。在伪代码阶段纠错的代价要低于以后的阶段。一旦伪代码被接受,就可以用编程语言的词汇和语法对其重写。

(2)流程图。流程图是算法的图形表示形式,它包含一系列符号。每个符号表示算法中描述的一种特定活动。其中涉及的有:接收输入、处理输出、显示输出、处理过程中将包含作出判断以及表示这些所有活动的符号等(见表 4-1 流程图符号)。

### 1.1.2 程序设计语言

要说明在一个程序的运行中需要做些什么,就需要仔细给出这一程序性活动的每一步细节过程,需要描述程序运行中的各种动作及其执行的顺序。为做到这些,就需要一种意义清晰、人用起来比较方便、计算机也能处理的描述方式。也就是说,需要有描述程序的合适语言。可供人们编写程序用的语言就是程序设计语言,这是一类人们自己设计的语言。程序设计语言常被称为编程语言,也常常简称为程序语言或语言。

程序语言的一个突出特点就在于不仅人能懂得和掌握它,能用它描述所需的计算过程,而且计算机也可以“识别”它,并且可以按程序语言所给出的计算过程去运行,完成人所需要的计算工作。程序设计语言是人描述计算的工具,也是人与计算机进行交流信息的媒介。通过用程序语言写程序,人能指挥计算机完成各种特定工作,完成各种计算。

计算机语言经历了机器语言、符号语言和高级语言三个发展阶段。

(1) 机器语言。计算机诞生之初,人们只能直接用二进制形式的机器语言写程序。对于人的使用而言,二进制的机器语言很不方便,用它书写程序非常困难,不但工作效率极低,程序的正确性也难以保证,发现有错误也很难辨认和改正。下面是一台假设计算机上的指令系列:

|                        |                         |
|------------------------|-------------------------|
| 0000000010000000001000 | 将单元 1000 的数据装入寄存器 0     |
| 000000001000100001010  | 将单元 1010 的数据装入寄存器 1     |
| 000001010000000000001  | 将寄存器 1 的数据乘到寄存器 0 原有数据上 |
| 0000000010001000011100 | 将单元 1100 的数据装入寄存器 1     |
| 000001000000000000001  | 将寄存器 1 的数据加到寄存器 0 原有数据上 |
| 0000001000000000011110 | 将寄存器 0 的数据存入单元 1110     |

这里想描述的是计算算术表达式  $a \times b + c$  (这里的符号  $a$ 、 $b$ 、 $c$  分别代表地址为 1000、1010 和 1100 的存储单元),而后将结果保存到单元 1110 的计算过程(程序)。对于一个复杂程序若用二进制机器指令来书写,十分困难且难于理解。

(2) 符号语言。也称汇编语言。为缓解使用机器语言的问题,人们发展了用符号形式表示,使用相对容易些的汇编语言。用汇编语言写的程序需要用专门软件(汇编系统)加工,翻译成二进制机器指令后才能在计算机上使用。

下面是用某种假想的汇编语言写出的程序,它完成与上面程序同样的工作:

|          |                         |
|----------|-------------------------|
| load 0 a | 将单元 a 的数据装入寄存器 0        |
| load 1 b | 将单元 b 的数据装入寄存器 1        |
| mult 0 1 | 将寄存器 1 的数据乘到寄存器 0 原有数据上 |
| load 1 c | 将单元 c 的数据装入寄存器 1        |
| add 0 1  | 将寄存器 1 的数据加到寄存器 0 原有数据上 |
| save 0 d | 将寄存器 0 的数据存入单元 d        |

汇编语言的每条指令对应于一条机器语言指令,但采用了助记的符号名,存储单元也用符号形式的名字表示。这样,每条指令的意义都更容易理解和把握了。但是,汇编语言的程序仍然完全没有结构,仅仅是许多这样的指令堆积形成的长长序列,是一团散沙。因此,复杂程序作为整体仍然难以理解。

(3) 高级语言。为克服低级语言(机器语言与汇编语言)的缺点,1954 年诞生了第一个高

级程序语言 FORTRAN,它采用接近于人们习惯使用的自然语言,用类似数学表达式的形式描述数据计算。语言中提供了有类型的变量,还提供了一些流程控制机制,如循环和子程序等。这些高级机制使编程者可以摆脱许多具体细节,方便了复杂程序的书写,写出的程序也更容易阅读,有错误也更容易辨认和改正。FORTRAN 语言诞生后受到广泛欢迎。

高级程序语言更接近人所习惯的描述形式,更容易被接受,也使更多的人能加入程序设计活动中。用高级语言书写程序的效率更高,这使人们开发出更多应用系统,反过来又大大推动了计算机应用的发展。应用的发展又推动了计算机工业的大发展。可以说,高级程序设计语言的诞生和发展,对于计算机发展到今天起了极其重要的作用。从 FORTRAN 语言诞生至今,人们已提出的语言超过千种,其中大部分只是实验性的,只有少数语言得到了广泛使用,如 C、C++、PASCAL、Ada、Java、LISP、Smalltalk、PROLOG 等,这些语言都曾在程序语言或计算机的发展历史上起过(有些仍在起着)极其重要的作用。随着时代发展,今天绝大部分程序都是用高级语言写的。人们也已习惯于用程序设计语言特指各种高级程序语言了。在使用高级语言(例如 C 语言)描述前面同样的程序片断只需一行代码:

```
d = a * b + c;
```

这表示要求计算机求出等于符号右边的表达式,而后将计算结果存入由 d 代表的存储单元中。这种表示方式与人们所熟悉的数学形式直接对应,因此更容易阅读和理解。高级语言程序中完全采用符号形式,使人可以摆脱难用的二进制形式和具体计算机的细节。此外,高级语言中还提供了许多高级的程序结构,供编写程序者组织复杂的程序。

计算机能否理解用这些高级语言编写的指令呢?不,它无法做到这一点。如何使计算机执行用高级编程语言写的程序指令呢?这时就需要一个翻译,将用编程语言写的指令翻译成机器指令。编译器就是这样一种特别的程序,对每一种语言都有不同的编译器。编译器有如下两种方式。

① 编译方式。人们首先针对具体语言(例如 C 语言)开发出一个翻译软件(程序),其功能是将采用该种高级语言书写的程序翻译为所用计算机的机器语言的等价程序。这样,用这种高级语言写出程序后,只要将它送给翻译程序,就能得到与之对应的机器语言程序。此后,只要命令计算机执行这个机器语言程序,计算机就能完成我们所需要的工作了。

② 解释方式。人们首先针对具体高级语言开发一个解释软件,其功能是一条一条地读入高级语言的程序,并能一步步地按照程序要求工作,完成程序所描述的计算。有了这种解释软件,只要直接将写好的程序送给运行着这个软件的计算机,就可以完成该程序所描述的工作了。

## 1.2 C 语言简介

### 1.2.1 C 语言出现的历史背景

#### 1. C 语言的出现

C 语言是一种计算机程序设计语言。它既具有高级语言的特点,又具有汇编语言的特点。它是由美国贝尔研究所的 D. M. Ritchie 于 1972 年推出。1978 年后,C 语言已先后被移植到大、中、小及微型机上。它可以作为系统设计语言,编写系统,也可以作为应用程序设计语言,编写不依赖计算机硬件的应用程序。它的应用范围广泛,具备很强的数据处理能力,不仅仅是

在软件开发上,而且各类科研都需要用到C语言,适于编写系统软件、三维和二维图形和动画软件。具体应用如单片机以及嵌入式系统开发。

在使用最多的微机上,也有许多性能良好的商品C语言系统可用。包括Borland公司早期的Turbo C和后续Borland C/C++系列产品;Microsoft(微软)公司的Microsoft C和后续Visual C/C++系列产品。还有其他C/C++语言系统产品,使用较广的有Watcom C/C++和Symantic C/C++等。此外还有许多廉价的和免费的C语言系统。各种工作站系统大都采用UNIX和Linux,C语言是它们的标准系统开发语言。各种大型计算机上也有自己的C语言系统。

## 2. C语言的发展过程

C语言的发展经历如下:

ALGOL 60 → CPL → BCPL → B → C → 标准C → ANSI C → ISO C

(1)ALGOL 60语言:它是一种面向问题的高级语言。ALGOL 60离硬件较远,不适合编写系统程序。

(2)CPL(combined programming language,组合编程语言):CPL是一种在ALGOL 60基础上更接近硬件的一种语言。CPL规模大,实现困难。

(3)BCPL(basic combined programming language,基本的组合编程语言):BCPL是对CPL进行简化后的一种语言。

(4)B语言:B语言是对BCPL进一步简化所得的一种很简单接近硬件的语言。B语言取BCPL语言的第一个字母。B语言精练、接近硬件,但过于简单,数据无类型。B语言诞生后,UNIX开始用B语言改写。

(5)C语言:它是在B语言基础上增加数据类型而设计出的一种语言。C语言取BCPL的第二个字母。C语言诞生后,UNIX很快用C语言改写,并被移植到其他计算机系统。

从C语言的发展历史可以看出,C语言是一种既具有一般高级语言特性,又具有低级语言特性的程序设计语言。C语言从一开始就是用于编写大型、复杂系统软件的,当然它也可以用来编写一般的应用程序。也就是说,C语言是程序员的语言!

### 1.2.2 C语言的基本特点

C语言之所以能被世界计算机界广泛接受是由于其自身的特点,主要有如下几点。

(1)C是中级语言。它把高级语言的基本结构和语句与低级语言的实用性结合起来。C语言可以像汇编语言一样对位、字节和地址进行操作,而这三者是计算机最基本的工作单元。

(2)C是结构式语言。结构式语言的显著特点是代码及数据的分隔化,即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰,便于使用、维护以及调试。C语言是以函数形式提供给用户的,这些函数可方便地调用,并具有多种循环、条件语句控制程序流向,从而使程序完全结构化。

(3)C语言功能齐全。具有各种各样的数据类型,并引入了指针概念,可使程序效率更高。而且计算功能、逻辑判断功能也比较强大,可以实现决策目的的游戏。

(4)C语言适用范围大。适合于多种操作系统,如Windows、DOS、UNIX等等;也适用于多种机型。

C语言对编写需要硬件进行操作的场合,明显优于其他高级语言,有一些大型应用软件也

是用 C 语言编写的。

C 语言的工作得到世界计算机界的广泛赞许。一方面,C 语言在程序设计语言研究领域具有一定价值,由它引出了不少后继语言,还有许多新语言从 C 语言中汲取营养,吸收了它的不少优点;另一方面,C 语言对计算机工业和应用的发展也起了很重要的推动作用。正是由于这些情况,C 语言的设计者获得世界计算机科学技术界的最高奖——图灵奖。

### 1.2.3 C 语言的标准化

在设计 C 语言时,设计者主要把它作为汇编语言的替代品,作为自己写操作系统的工具,因此更多强调的是灵活性和方便性。语言的规定很不严格,可以用许多不规矩的方式写程序,因此也留下了许多不安全因素。使用这样的语言,就要求程序员自己注意可能出现的问题,程序的正确性主要靠人来保证,语言的处理系统(编译程序)不能提供多少帮助。随着应用范围的扩大,使用 C 语言的人越来越多,C 语言在这方面的缺点日益突出起来。由此造成的后果是,人们用 C 语言开发的复杂程序里常带有隐藏很深的错误,难以发现和改正。

随着应用发展,人们更强烈地希望 C 语言能成为一种更安全可靠、不依赖于具体计算机和操作系统(如 UNIX)的标准程序设计语言。美国国家标准局(ANSI)在上世纪 80 年代建立了专门小组研究 C 语言标准化问题,在 1988 年颁布 ANSI C 标准。这个标准被国际标准化组织和各国标准化机构接受,同样也被采纳为中国国家标准。此后,1999 年通过了 ISO/IEC 9899:1999 标准(一般称为 C99)。这一新标准对 ANSI C 做了一些小修订和扩充。

## 1.3 C 语言程序设计简介

### 1.3.1 简单 C 语言程序的构成与格式

用 C 语言写出的程序简称为 C 语言程序。下面先请读者阅读几个简单的 C 语言程序例子,以初步了解用 C 语言写出的程序是什么样子。

**【例 1-1】** 要求在屏幕上输出一行信息。

```
#include <stdio.h>           // 编译预处理命令
void main()                  // 定义主函数
{                             // 函数开始的标志
    printf("This is a C program.\n"); // 输出指定一行信息
}                             // 函数结束标志
```

本程序运行的结果是:

This is a C program.

程序说明:

①上面这个简单程序可分为两个基本部分:第一行是个特殊行,include<stdio.h>说明程序用到 C 语言系统提供的标准功能(参考标准库文件 stdio.h)。其他几行是程序的基本部分。

②main 是主函数名,void 是函数类型。每个 C 语言程序都必须有一个 main 函数,它是每一个 C 语言程序的执行起始点(入口点)。

主函数 main 的函数体是用一对花括号“{}”括起来的。“{}”是函数开始和结束的标志,不

可省略。main 函数中的所有操作(或语句)都在这一对花括号之间。也就是说,main 函数的所有操作都在 main 函数体中。

③“printf”语句是 C 语言的库函数,功能是用于程序的输出(显示在屏幕上),本例用于将一个字符串“This is a C program.\n”的内容输出。即在屏幕上显示:“This is a C program.”。“\n”表示输出后换行。

④注意:函数体中每条语句都要用“;”号结束。

⑤在使用库函数中输入输出函数时,必须提供有关此函数的信息“#include <stdio.h>”。

**【例 1-2】** 用 C 语言实现求和问题。

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int a,b,sum;    // 定义变量 a、b、sum
```

```
    a = 123;b = 456;    // 以下 3 行为 C 语句
```

```
    sum = a + b;
```

```
    printf("sum = %d\n",sum);    // %d 为格式控制
```

```
}
```

本程序运行的结果是:

sum = 579

程序说明:

①同样此程序也必须包含一个 main 函数作为程序执行的起点。在“{”之间为 main 函数的函数体,main 函数所有操作均在 main 函数体中。

②注释可以用“//”和“/\*”和“\*/”来标识。从“//”开始到换行符结束的为单行注释;用/\*和\*/括起来的部分为段注释,注释只是为了改善程序的可读性,在编译、运行时不起作用。所以可以用汉字或英文字符表示,可以出现在一行中的最右侧,也可以单独成为一行。注释允许占用多行,只是需要注意“/\*”与“\*/”配对使用,一般不要嵌套注释。

③int a,b,sum;是定义三个具有整数类型的变量 a、b、sum。C 语言的变量必须先声明再使用。

④a=123;b=456;是两条赋值语句。将整数 123 赋给整型变量 a,将整数 456 赋给整型变量 b。a、b 两个变量的值分别为 123 和 456。注意这是两条赋值语句,每条语句均用“;”结束。也可以将两条语句写成两行,即:

```
a = 123;
```

```
b = 456;
```

由此可见 C 语言程序的书写可以相当随意,但是为了保证容易阅读要遵循一定的规范。

⑤sum=a+b;是将 a、b 两个变量的内容相加,然后将其结果赋值给整型变量 sum。此时 sum 的内容为 579。

⑥printf("sum=%d\n",sum);是调用库函数输出 sum 的结果。“%d”为格式控制,表示 sum 的值是以十进制整数形式输出。

**【例 1-3】** 输入两个数,求两个整数中较大者。

```
#include <stdio.h>
```