

21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

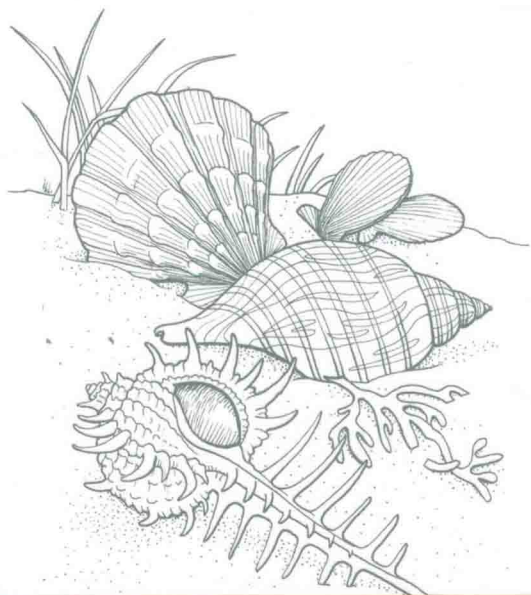
# C语言 程序设计 (第2版)

The C Language Programming (2nd Edition)

韩增红 佟继红 主编

王冬梅 许盟 副主编

- 循序渐进，融会贯通
- 案例丰富，实用性强
- 详解应用系统开发实例



高校系列



人民邮电出版社  
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材

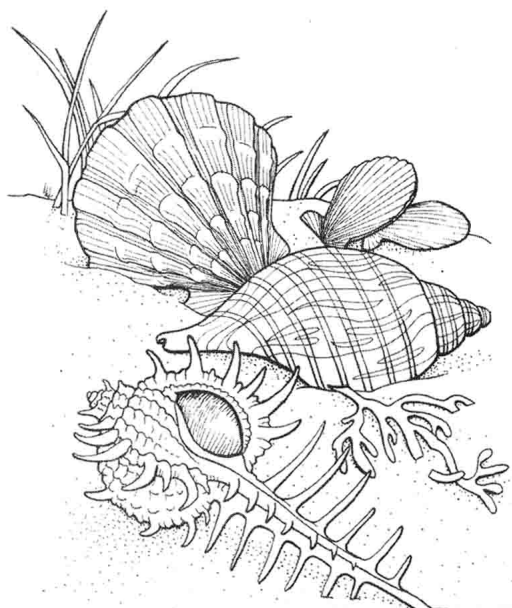
21st Century University Planned Textbooks of Computer Science

# C语言 程序设计 (第2版)

The C Language Programming (2nd Edition)

韩增红 佟继红 主编

王冬梅 许盟 副主编



高校系列

人民邮电出版社

北京

## 图书在版编目 (CIP) 数据

C语言程序设计 / 韩增红, 佟继红主编. -- 2版. --  
北京: 人民邮电出版社, 2015.2  
21世纪高等学校计算机规划教材. 高校系列  
ISBN 978-7-115-38280-1

I. ①C… II. ①韩… ②佟… 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021715号

## 内 容 提 要

本书以 C 语言程序设计的基本原理为出发点, 以程序设计及应用为主线, 讲解由浅入深、循序渐进、突出重点。本书的特点是概念准确、内容合理、案例丰富、实用性强。

全书共 12 章, 内容包括概述、数据类型和输入/输出、运算符和表达式、顺序结构程序设计、选择结构程序设计、循环结构程序设计、数组、函数、构造数据类型、指针、文件、综合应用及附录 I、附录 II 和附录 III。每一章都附有适量的习题, 读者可通过习题巩固已学的知识。书中全部程序均上机调试通过。

本书是面向没有程序设计基础的读者编写的入门教材, 适用于本科、专科及各类成人教育的 C 语言程序设计课程, 可作为计算机培训和计算机等级考试的教材, 也可作为广大程序开发人员和计算机爱好者学习 C 语言程序设计的参考用书。为配合本书的学习, 本书配有配套的《C 语言程序设计上机指导与习题(第2版)》, 供学习者参考。

---

◆ 主 编 韩增红 佟继红  
副 主 编 王冬梅 许 盟  
责任编辑 武恩玉  
责任印制 沈 蓉 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号  
邮编 100164 电子邮件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京圣夫亚美印刷有限公司印刷

◆ 开本: 787×1092 1/16  
印张: 18  
字数: 475 千字

2015 年 2 月第 2 版  
2015 年 2 月北京第 1 次印刷

---

定价: 39.80 元

读者服务热线: (010) 81055256 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

广告经营许可证: 京崇工商广字第 0021 号

# 前 言

---

---

---

---

---

---

---

---

C 语言是目前广泛使用的一种计算机程序设计语言。它因其功能丰富、表达能力强、使用灵活方便、应用面广、目标程序效率高、可移植性好,同时既具有高级语言的优点,又具有低级语言的许多特点,而成为当今软件开发领域中广泛应用的一种语言。C 语言既可用于编写系统软件,也可用来编写应用软件,是国际公认的最重要的几种通用程序设计语言之一,也是国内外大学介绍计算机程序设计方法的首选语言。

本书以 C 语言程序设计的基本原理为出发点,以程序设计为主线,以实际应用为目标,讲解由浅入深、循序渐进、突出重点,内容安排合理、案例丰富、实用性强。

全书共 12 章和 3 个附录。第 1 章为概述,简单介绍了程序设计的基础知识、C 语言的特点、程序结构和上机步骤及常用集成开发环境;第 2 章为数据类型和输入/输出,介绍了数据在计算机内的表示、C 语言的数据类型、常量和变量以及数据的输入和输出;第 3 章为运算符和表达式,介绍了常用运算符和表达式;第 4 章至第 6 章详细介绍了 C 语言的结构化程序设计方法,包括顺序结构程序设计、选择结构程序设计和循环结构程序设计;第 7 章为数组,介绍了各类数组的定义和使用方法;第 8 章为函数,详细介绍了 C 语言程序的结构、函数的定义及使用,并简单介绍了程序编译预处理;第 9 章为构造数据类型,介绍了结构体、共用体、枚举类型的定义及使用;第 10 章为指针,深入浅出地介绍了指针的概念和应用;第 11 章为文件,介绍了文件的概念和对文件的各种操作;第 12 章为综合应用,从结构化程序设计方法学角度出发,阐述了 C 语言开发应用程序的一般步骤和方法。各章后都附有适量的习题,对于检验读者的学习情况和巩固已学的知识都有裨益。书中全部实例和习题均已上机调试通过。

本书由韩增红、佟继红主编,王冬梅、许盟任副主编,参加本书编写的还有段立平、肖丽君、张泽梁、范银平、曹健、毕馨文等。韩增红编写第 1 章、第 8 章,段立平编写第 2 章、第 11 章,王冬梅编写第 3 章,张泽梁编写第 4 章、第 12 章,曹健编写第 5 章,许盟编写第 6 章,肖丽君编写第 7 章,范银平、毕馨文共同编写第 9 章,佟继红编写第 10 章。

感谢读者选用本书,同时欢迎您对本书的不足之处提出宝贵意见和建议。

编者

2014 年 11 月

---

---

---

---

## 第1章 概 述 .....1

|                                  |    |
|----------------------------------|----|
| 1.1 程序设计基础.....                  | 1  |
| 1.1.1 程序与程序设计语言 .....            | 1  |
| 1.1.2 程序设计方法 .....               | 3  |
| 1.1.3 程序设计的基本过程 .....            | 4  |
| 1.2 C语言及其特点.....                 | 7  |
| 1.2.1 C语言的发展过程.....              | 7  |
| 1.2.2 C语言的特点.....                | 8  |
| 1.3 简单的C语言程序.....                | 9  |
| 1.4 函数.....                      | 11 |
| 1.5 C语言程序的调试.....                | 14 |
| 1.5.1 调试步骤 .....                 | 14 |
| 1.5.2 常用的C语言集成开发环境.....          | 15 |
| 1.5.3 Visual C++ 6.0 集成开发环境..... | 16 |
| 本章小结.....                        | 21 |
| 习 题.....                         | 22 |

## 第2章 数据类型和输入/输出.....23

|                           |    |
|---------------------------|----|
| 2.1 数据在计算机内部的表示.....      | 23 |
| 2.1.1 常用的进位制 .....        | 23 |
| 2.1.2 数值与字符在计算机中的表示 ..... | 24 |
| 2.2 字符集和保留字.....          | 25 |
| 2.2.1 基本符号集 .....         | 25 |
| 2.2.2 标识符.....            | 26 |
| 2.3 C语言的数据类型.....         | 27 |
| 2.4 常量.....               | 28 |
| 2.4.1 数值常量 .....          | 28 |
| 2.4.2 字符常量和字符串常量 .....    | 29 |
| 2.4.3 符号常量 .....          | 31 |
| 2.5 变量.....               | 32 |
| 2.5.1 变量的定义和变量的存储 .....   | 33 |
| 2.5.2 变量的初始化 .....        | 35 |
| 2.6 数据的输入和输出 .....        | 36 |
| 2.6.1 格式输入/输出函数 .....     | 36 |

|                      |    |
|----------------------|----|
| 2.6.2 字符输入/输出函数..... | 41 |
| 本章小结.....            | 43 |
| 习 题.....             | 43 |

## 第3章 运算符和表达式 .....45

|                         |    |
|-------------------------|----|
| 3.1 运算符和表达式简介 .....     | 45 |
| 3.1.1 运算符.....          | 45 |
| 3.1.2 表达式.....          | 46 |
| 3.2 常用运算符和表达式 .....     | 47 |
| 3.2.1 算术运算符和算术表达式.....  | 47 |
| 3.2.2 赋值运算符和赋值表达式.....  | 49 |
| 3.2.3 逗号运算符和逗号表达式.....  | 51 |
| 3.2.4 关系运算和逻辑运算.....    | 51 |
| 3.2.5 条件表达式.....        | 54 |
| 3.2.6 位运算表达式.....       | 55 |
| 3.2.7 求字节数表达式.....      | 59 |
| 3.3 运算符的结合性和优先级 .....   | 60 |
| 3.4 混合运算中的数据类型的转换 ..... | 60 |
| 3.4.1 自动类型转换.....       | 61 |
| 3.4.2 强制类型转换.....       | 61 |
| 本章小结.....               | 62 |
| 习 题.....                | 62 |

## 第4章 顺序结构程序设计.....65

|                        |    |
|------------------------|----|
| 4.1 结构化程序设计的3种结构 ..... | 65 |
| 4.2 顺序结构 .....         | 66 |
| 4.2.1 赋值语句和空语句.....    | 66 |
| 4.2.2 复合语句.....        | 67 |
| 4.3 顺序结构程序设计举例.....    | 68 |
| 本章小结 .....             | 71 |
| 习 题.....               | 71 |

## 第5章 选择结构程序设计.....72

|                  |    |
|------------------|----|
| 5.1 if语句.....    | 72 |
| 5.1.1 if形式 ..... | 72 |

|                             |            |                           |            |
|-----------------------------|------------|---------------------------|------------|
| 5.1.2 if-else 形式 .....      | 73         | 7.4.4 字符串处理函数 .....       | 124        |
| 5.1.3 if-else-if 形式 .....   | 74         | 7.4.5 字符数组的应用举例 .....     | 126        |
| 5.1.4 if 语句的嵌套 .....        | 77         | 本章小结 .....                | 128        |
| 5.2 switch 语句 .....         | 80         | 习 题 .....                 | 128        |
| 5.3 选择结构程序设计举例 .....        | 84         | <b>第 8 章 函 数</b> .....    | <b>129</b> |
| 本章小结 .....                  | 86         | 8.1 函数概述 .....            | 129        |
| 习 题 .....                   | 87         | 8.2 函数的定义及函数返回值 .....     | 132        |
| <b>第 6 章 循环结构程序设计</b> ..... | <b>90</b>  | 8.2.1 函数的定义 .....         | 132        |
| 6.1 while 语句 .....          | 90         | 8.2.2 函数的返回值 .....        | 134        |
| 6.2 do-while 语句 .....       | 92         | 8.3 函数的参数 .....           | 136        |
| 6.3 for 语句 .....            | 93         | 8.3.1 有参函数的一般形式 .....     | 136        |
| 6.4 3 种循环语句的区别 .....        | 96         | 8.3.2 形式参数与实际参数 .....     | 136        |
| 6.5 循环结构的嵌套 .....           | 97         | 8.3.3 数组作为函数的参数 .....     | 138        |
| 6.5.1 定义 .....              | 97         | 8.4 函数的调用 .....           | 145        |
| 6.5.2 嵌套结构的规则 .....         | 97         | 8.4.1 函数原型 .....          | 145        |
| 6.6 转向语句 .....              | 99         | 8.4.2 函数调用的一般形式 .....     | 148        |
| 6.6.1 break 语句 .....        | 99         | 8.4.3 函数的嵌套调用 .....       | 149        |
| 6.6.2 continue 语句 .....     | 100        | 8.4.4 函数的递归调用 .....       | 151        |
| 6.6.3 goto 语句 .....         | 101        | 8.5 变量的作用域和存储类型 .....     | 155        |
| 6.7 循环结构程序设计举例 .....        | 101        | 8.5.1 变量的作用域 .....        | 155        |
| 本章小结 .....                  | 104        | 8.5.2 变量的存储类型 .....       | 156        |
| 习 题 .....                   | 104        | 8.6 程序编译预处理 .....         | 159        |
| <b>第 7 章 数 组</b> .....      | <b>105</b> | 8.6.1 宏定义 .....           | 160        |
| 7.1 数组和数组元素 .....           | 105        | 8.6.2 文件包含 .....          | 162        |
| 7.2 一维数组 .....              | 106        | 8.6.3 条件编译 .....          | 163        |
| 7.2.1 一维数组的定义 .....         | 106        | 本章小结 .....                | 165        |
| 7.2.2 一维数组的引用 .....         | 107        | 习 题 .....                 | 166        |
| 7.2.3 一维数组的初始化 .....        | 108        | <b>第 9 章 构造数据类型</b> ..... | <b>167</b> |
| 7.2.4 一维数组的应用举例 .....       | 109        | 9.1 结构体类型 .....           | 167        |
| 7.3 二维数组 .....              | 113        | 9.1.1 结构体类型定义 .....       | 167        |
| 7.3.1 二维数组的定义 .....         | 113        | 9.1.2 结构体变量的说明及使用 .....   | 169        |
| 7.3.2 二维数组的引用 .....         | 115        | 9.1.3 结构体变量的初始化 .....     | 171        |
| 7.3.3 二维数组的初始化 .....        | 116        | 9.1.4 结构体数组 .....         | 172        |
| 7.3.4 二维数组的应用举例 .....       | 118        | 9.1.5 结构体和函数 .....        | 173        |
| 7.4 字符数组与字符串 .....          | 119        | 9.2 共用体类型 .....           | 175        |
| 7.4.1 字符数组的定义和引用 .....      | 120        | 9.2.1 共用体类型的定义 .....      | 175        |
| 7.4.2 字符数组的初始化 .....        | 120        | 9.2.2 共用体变量的说明及使用 .....   | 176        |
| 7.4.3 字符串的输入和输出 .....       | 121        | 9.3 位字段类型 .....           | 177        |

|                                   |            |                                                         |            |
|-----------------------------------|------------|---------------------------------------------------------|------------|
| 9.3.1 位字段类型的定义 .....              | 177        | 11.1.1 文件的概念 .....                                      | 220        |
| 9.3.2 位字段变量的说明及其使用 .....          | 178        | 11.1.2 文件类型结构及文件指针 .....                                | 221        |
| 9.4 枚举类型 .....                    | 180        | 11.2 文件的打开与关闭 .....                                     | 222        |
| 9.4.1 枚举类型的定义 .....               | 180        | 11.2.1 文件的打开 .....                                      | 222        |
| 9.4.2 枚举类型变量的说明及其使用 .....         | 180        | 11.2.2 文件的关闭 .....                                      | 224        |
| 9.5 用 typedef 定义类型 .....          | 182        | 11.3 文件的读写操作 .....                                      | 224        |
| 本章小结 .....                        | 182        | 11.3.1 字符的读写 .....                                      | 224        |
| 习 题 .....                         | 183        | 11.3.2 字符串的读写 .....                                     | 228        |
| <b>第 10 章 指 针 .....</b>           | <b>184</b> | 11.3.3 数据块的读写 .....                                     | 229        |
| 10.1 指针和地址的概念 .....               | 184        | 11.3.4 格式读写 .....                                       | 231        |
| 10.2 变量的指针与指针变量 .....             | 185        | 11.4 位置指针与文件的定位 .....                                   | 232        |
| 10.2.1 指针变量的定义及使用 .....           | 185        | 11.5 文件状态的检测 .....                                      | 234        |
| 10.2.2 指针变量的初始化 .....             | 187        | 11.6 文件程序设计举例 .....                                     | 234        |
| 10.2.3 指针运算 .....                 | 187        | 本章小结 .....                                              | 237        |
| 10.3 指针与数组 .....                  | 190        | 习 题 .....                                               | 237        |
| 10.3.1 指向数组元素的指针 .....            | 190        | <b>第 12 章 综合应用 .....</b>                                | <b>238</b> |
| 10.3.2 字符指针与字符数组 .....            | 193        | 12.1 应用系统的设计方法 .....                                    | 238        |
| 10.3.3 多级指针及指针数组 .....            | 196        | 12.1.1 结构化程序设计方法概述 .....                                | 238        |
| 10.3.4 指针与多维数组 .....              | 198        | 12.1.2 结构化程序设计方法举例 .....                                | 239        |
| 10.4 指针与函数 .....                  | 200        | 12.2 应用系统的设计举例 .....                                    | 241        |
| 10.4.1 函数参数为指针 .....              | 200        | 12.2.1 小学算术运算模拟测试系统 .....                               | 241        |
| 10.4.2 函数的返回值为指针 .....            | 206        | 12.2.2 学生成绩管理系统 .....                                   | 246        |
| 10.4.3 指向函数的指针 .....              | 208        | 本章小结 .....                                              | 262        |
| 10.4.4 命令行参数 .....                | 210        | 习 题 .....                                               | 262        |
| 10.5 指针与结构体 .....                 | 211        | <b>附录 I ASCII 码表 .....</b>                              | <b>263</b> |
| 10.5.1 结构体指针与指向结构体数组<br>的指针 ..... | 211        | <b>附录 II C 语言常用库函数 .....</b>                            | <b>264</b> |
| 10.5.2 结构体指针与函数 .....             | 215        | <b>附录 III Visual C++ 6.0 编译 C 语言<br/>程序常见错误信息 .....</b> | <b>277</b> |
| 本章小结 .....                        | 217        | <b>参考文献 .....</b>                                       | <b>282</b> |
| 习 题 .....                         | 218        |                                                         |            |
| <b>第 11 章 文 件 .....</b>           | <b>220</b> |                                                         |            |
| 11.1 文件概述 .....                   | 220        |                                                         |            |

# 第 1 章

## 概 述

计算机语言是人与计算机之间交流信息的工具，由计算机能够识别的语句组成，它使用一整套带有严格规定的符号体系来描述计算机语言的词法、语法、语义、语用。词法负责从构成源程序的字符串中，识别出一个个具有独立意义的最小语法单位（单词）；语法涉及语言的构成规律，确定程序的结构形式；语义说明语句代表的含义及该语句的执行过程；语用指出语句的实际用途。

C 语言是一种通用的程序设计语言，它具有丰富的运算符和表达式，以及先进的控制结构和数据结构。C 语言既具有高级语言简单易学、可移植性好的特点，又具有汇编语言生成代码质量高的优点。因此，C 语言具有较强的生命力和广泛的应用前景。

本章从程序设计基础知识入手，对 C 语言做了概括性介绍，让读者了解一个 C 语言程序的基本框架和它的书写格式，使读者能够学会编写简单的 C 程序，并能够进行编辑、编译、连接、调试运行等上机操作。

## 1.1 程序设计基础

在介绍 C 语言程序设计之前，我们先来了解一些有关程序设计的基础知识。

### 1.1.1 程序与程序设计语言

#### 1. 程序

程序就是一系列遵循一定规则和思想并能正确完成指定工作的代码（也称为指令序列）。简单地说，程序主要用于描述完成某项功能所涉及的对象和动作规则。通常，一个计算机程序主要描述两部分的内容：一是描述问题的每个对象及它们之间的关系，即数据结构的内容；二是描述对这些对象进行处理的动作、这些动作的先后顺序以及它们所作用的对象，要遵守一定的规则，即求解某个问题的算法。

因此，对程序的描述，也可以用经典的公式来表示：

**程序=数据结构+算法**

一个设计合理的数据结构往往可以简化算法，而且一个好的程序应该具有可靠性、易读性、可维护性等良好特点。

计算机程序有以下共同的性质：

（1）目的性。程序有明确的目的，运行时能完成赋予它的功能。

- (2) 分步性。程序为完成其复杂的功能, 由一系列计算机可执行的步骤组成。
- (3) 有序性。程序的执行步骤是有序的, 不可随意改变程序步骤的执行顺序。
- (4) 有限性。程序是有限的指令序列, 程序所包含的步骤是有限的。
- (5) 操作性。有意义的程序总是对某些对象进行操作, 使其改变状态, 完成其功能。

## 2. 程序设计语言

我们平时在使用计算机为我们工作时, 计算机所做的工作实际上是由人们事先编好的程序来控制的, 编写程序的工具就是程序设计语言。

程序设计语言 (Programming Language) 是一组用来定义计算机程序的语法规则。它是一种被标准化的交流技巧, 用来向计算机发出指令。一种计算机语言能够让程序员准确地定义计算机所需的数据, 并精确地定义在不同情况下所应当采取的行动。

程序设计语言有许多种, 按照程序设计语言发展的过程, 大概分为三类: 机器语言、汇编语言和高级语言。

### (1) 机器语言

机器语言是由 0 和 1 二进制代码按一定规则组成的、能被机器直接理解和执行的指令集合。它的突出优点是具有最快的运行速度和最少的存储开销。它的缺点是机器指令不容易记忆, 可读性差, 编程工作量大, 容易出错, 通用性差, 且随着不同的计算机系统而改变。现在已经没有人用机器语言直接编程了。

下面这三行就是某计算机的机器指令, 功能是计算  $A=15+10$  的值:

```
10110000 00001111
00101100 00001010
11110100
```

首先, 把 15 放入累加器 A 中, 然后让 10 与累加器 A 中的值相加, 结果仍放入 A 中, 最后结束, 停机。

### (2) 汇编语言

为了克服机器语言的缺点, 人们用英文助记符描述机器指令, 如用 ADD 表示加, SUB 表示减等, 这种采用指令助记符表示的语言就是汇编语言, 又称符号语言。例如, 上述  $A=15+10$  的计算可用下面的汇编语言程序实现:

```
MOV A, 15      : 把 15 放入累加器 A 中
ADD A, 10      : 10 与累加器 A 中的值相加, 结果仍放入 A 中
HLT           : 结束, 停机
```

这 3 条汇编指令与前面 3 条机器指令的作用是一一对应的。两者相比, 后者的清晰度明显有了改善。可见, 汇编语言一定程度上克服了机器语言难读、难改的缺点, 同时保持了其编程质量高、占存储空间少、执行速度快的优点。因而在程序设计中, 对实时性要求较高的地方, 如过程控制等, 仍经常采用汇编语言。但汇编语言面向机器, 使用汇编语言编程不仅需要直接安排存储, 规定寄存器和运算器的动作次序, 还必须知道计算机对数据约定的表示 (定点、浮点、双精度) 等。这对大多数人员来说, 都不是一件简单的事情。此外, 该语言还是依赖于机器, 不同的计算机在指令长度、寻址方式、寄存器数目、指令表示等方面都不一样, 这样使得汇编程序不仅通用性较差, 而且可读性也差。这促使了高级语言的出现。

用汇编语言编写的程序, 必须翻译成计算机所能识别的机器语言后, 才能被计算机执行。

### (3) 高级语言

高级语言是由表达各种意义的英文单词和数学公式,按照一定的语法规则来编写程序的语言。例如,用“+”表示加法、用“-”表示减法等。

高级语言是更接近于自然语言或数学语言的程序设计语言,便于学习和记忆。例如,上述计算  $A=15+10$  的 Basic 语言程序如下:

```
A=15+10      '15 与 10 相加的结果放入 A 中
PRINT A      '输出 A
END          '程序结束
```

可见,高级语言编写的程序非常直观明了。

高级语言彻底摆脱了依赖于机器硬件的指令系统,是面向应用的计算机语言。它不再依赖于具体的计算机,用高级语言编写的程序可以在不同的计算机上使用,程序具有可移植性。编程时,用户不再考虑计算机的内部结构和硬件环境,可以集中精力考虑解题的算法和数据结构,因此编程效率大大提高。

高级语言不能直接在计算机上运行,因为计算机只能识别机器语言程序,高级语言编写的源程序必须经过另外的语言处理程序翻译成机器语言程序后才能被机器接受。因此,高级语言程序的执行速度通常比不上机器语言。

翻译程序分为两种:一种是解释系统;另一种是编译系统。解释系统是对高级语言编写的程序翻译一句执行一句;而编译系统是先将高级语言编写的程序文件全部翻译成机器语言,等生成可执行文件以后再执行。高级语言几乎在每一种机器上都有自己的翻译程序。C 语言的翻译程序属于编译系统。

高级语言可分为 3 类:面向过程的语言、面向问题的语言和面向对象的语言。

① 面向过程的语言致力于用计算机能够理解的逻辑来描述需要解决的问题和解决问题的具体方法、步骤。在程序中,不仅要告诉计算机“做什么”,还要告诉计算机“如何做”,即在程序中要详细描述用什么动作加工什么数据,即解题的过程和细节。面向过程的语言有 Fortran、Basic、Pascal、C 等。

② 面向问题的语言又称非过程化的语言或称第四代语言(4GLS)。用面向问题的语言解题时,不必关心问题的求解算法和求解的过程,只需指出问题是要计算机做什么和数据的输入和输出形式,就能得到所需结果。面向问题的语言是采用快速原型法开发应用软件的强大工具,能够快速构造应用系统,从而大大地提高软件开发效率。它与数据库的关系非常密切,能够对大型数据库进行高效处理。目前应用最广泛的面向问题的语言有 SQL 等。

③ 面向对象的语言是将客观事物看作具有属性和行为的对象,通过抽象找出同一类对象的共同属性和行为,形成类。通过类的继承与多态可以很方便地实现代码重用,这大大地提高了程序的复用能力和程序开发效率。面向对象语言已是程序语言的主要研究方向之一。面向对象的语言有 C++、Java、Visual Basic 等。

## 1.1.2 程序设计方法

简单地说,程序设计就是用计算机语言编写程序的过程。学习计算机语言的目的就是利用该语言工具设计出可供计算机运行的程序。通常,程序设计是很讲究方法的,程序设计方法是影响程序设计成败以及程序设计质量的重要因素之一。C 语言主要采用结构化程序设计方法。

结构化程序设计是荷兰学者狄克斯特拉 (E.W.dijkstra) 在 1969 年提出的一种程序设计方法, 它规定了一套如何进行程序设计的准则, 采用了自顶向下、逐步求精的分析设计方法、分而治之的分割技术和模块化的组织结构, 使得设计的程序具有合理的结构、易读、易调试和容易保证其正确性。

结构化程序设计方法采用:

- (1) 自顶向下;
- (2) 逐步细化;
- (3) 模块化设计;
- (4) 结构化编码。

结构化程序设计以模块化设计为中心, 将待开发的软件系统划分为若干个相互独立的模块, 这样使完成每一个模块的工作变得单纯而明确, 为设计一些较大的软件打下了良好的基础。

由于模块相互独立, 因此在设计其中一个模块时, 不会受到其他模块的牵连, 因而可将原来较为复杂的问题简化为一系列简单模块的设计。模块的独立性还为扩充已有的系统、建立新系统带来了不少的方便, 我们可以充分利用现有的模块做积木式的扩展。

按照结构化程序设计的观点, 任何模块都可以通过 3 种基本程序结构: 顺序结构、选择结构和循环结构组合来实现。3 种基本结构应具有如下良好的特性:

- (1) 只有一个入口, 即每个模块与外部联系只有单一的入口;
- (2) 只有一个出口, 即每个模块与外部联系只有单一的出口;
- (3) 无死语句, 即不存在永远都执行不到的语句;
- (4) 无死循环, 即不存在永远都执行不完的循环。

每个模块根据其功能先编写程序框架, 逐步深入, 直到精确地编写出每一个程序结构, 精确地编写每一条语句。

完成编程后, 应该检查每条语句、每个程序结构的逻辑及每个模块的功能是否正确, 直到检查整个程序是否达到问题的要求, 通过编辑、编译、连接运行、调试检验程序是否达到精度要求。

程序设计的首要目标是在程序正确的前提下, 提高程序的可读性、易维护性和可移植性。可读性是指使用良好的书写风格和易懂的语句编写程序; 易维护性是指当业务需求发生变化时, 不需要太多的开销就可以扩展和增加程序的功能; 可移植性是指编写的程序在各种计算机和操作系统上都能运行, 并且运行结果相同。

### 1.1.3 程序设计的基本过程

程序设计是人们根据要解决的实际问题, 提出相应的需求, 在此基础上设计数据结构和算法, 然后再编写相应的源程序代码并测试该代码运行的正确性, 通过反复调试直到能够得到正确的运行结果为止, 最后整理设计文档的全过程。较小规模的程序设计由一个程序员完成, 大型程序设计是由多个程序员分工、共同协作完成的, 因此必须经过多种测试, 并详细整理设计文档。一般来讲, 这个过程应当按图 1-1 所示的步骤进行。

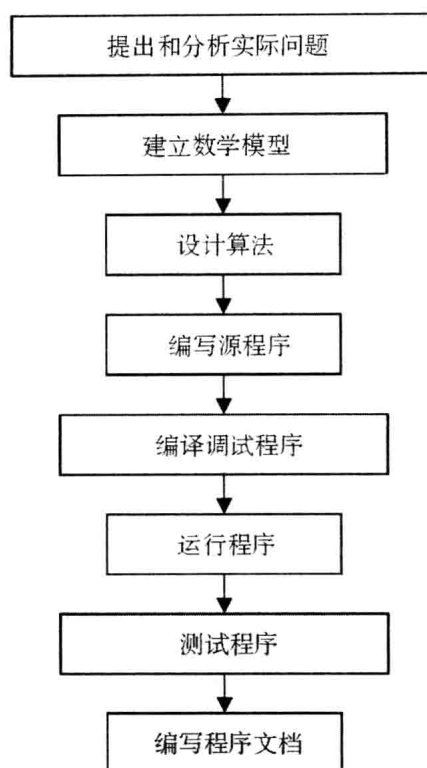


图 1-1 程序设计的基本过程

### 1. 提出和分析实际问题

提出实际需求的用户往往不具备太多的计算机知识，而程序设计人员可能又不具备用户的专业知识，因此，在程序开发初期首先必须由用户和程序设计人员一起对实际问题进行分析，充分理解用户的要求，明确哪些要求可以实现，哪些要求不能实现或需经一定的处理才能实现，确定开发的总目标，提出开发的任务和要求，从中获得必要的输入数据，明确问题要求做什么，需要什么样的数据输出。

按照分析问题的要求，弄清问题的性质，发现问题的特点，确定解决问题的目标这些步骤，能够使我们采取有效的方法解决问题。

### 2. 建立数学模型

要用计算机解决实际问题，首先要用理想模型模拟实际问题。理想模型是从实体中抽象出来并能用数学表达式精确定义的实体。建立数学模型的过程就是把错综复杂的实际问题进行简化抽象，用数学公式来描述实际问题的运动和变化过程。例如，用一些数学方程来描述人造卫星的飞行轨迹等。

在建立数学模型的过程中，分析问题的要求，确定解决问题的目标，明确问题的输入数据和输出信息，理解问题的约束限制条件，是选择制定数学模型、建立数学模型的关键步骤，是解决问题的关键所在。

### 3. 设计算法

一般来说，从实际问题抽象出数学模型通常是有关领域的专业工作者的任务，计算机工作人员只起辅助作用。程序设计人员的工作，最关键的一步就是设计算法。算法是指为解决某一特定问题而进行一步一步有限的操作过程，是一组规则的集合，可以用流程图来表示算法。编写程序

代码之前,设计好算法,画出流程图,往往会起到事半功倍的效果。

算法可以分为两大类:数值计算算法和非数值计算算法。数值计算算法的目的是求数值解,其特点是少量的输入、输出,复杂的运算,如求高次方程的根、求函数的定积分等。非数值计算算法目的是对数据的处理,其特点是大量的输入、输出,简单的运算,例如,对数据的排序、查找等算法。

一个好的算法应当具有以下特性。

(1) 有穷性。一个算法应包含有限个操作步骤。也就是说,在执行若干个操作步骤之后,算法将结束,并且每一步都在合理的时间范围内完成。

(2) 确定性。算法中每一条指令必须有确切的含义,不能有二义性,并且对于相同的输入必然有相同的执行结果。

(3) 可行性。一个算法是可行的,即算法中指定的操作,都可以通过已实现的基本运算执行有限次来实现。

(4) 输入。一个算法一般有零个或多个输入。在计算机上实现的算法,通常是用来处理数据对象的,在大多数情况下这些数据是需要通过输入来得到的。

(5) 输出。一个算法一般有一个或多个输出,算法的目的是求“解”,这些“解”只有通过输出才能得到。

描述算法的常用工具是流程图,也称程序框图,流程图是算法的图形描述,它用一组图形符号来表示各种操作,它往往比程序更直观,能较清晰地表达各种操作之间的逻辑关系,容易阅读和理解。常用的流程图符号如图 1-2 所示。



图 1-2 常用流程图符号

#### 4. 编写源程序

如果算法正确,将它转换为任何一种高级语言程序并不困难,这一步通常称为编码。

现在的程序设计语言一般都是一个集成开发环境,自带编辑器,方便编辑程序。编写好的程序代码通过编辑器输入计算机内,利用编辑器可对输入的程序代码进行复制、删除、移动等编辑操作,然后以文件(源程序)形式保存。

#### 5. 编译调试程序

源程序必须通过编译程序将源程序翻译成目标程序,这期间编译器对源程序进行语法结构检查,找出在程序编制过程中存在的语法错误,加以修改。这是一个重复进行的过程,需要反复调试程序。

#### 6. 运行程序

将编译源程序生成的目标程序和程序中所需的系统中固有的目标程序模块(如调用的标准函数、执行的输入/输出操作的模块)连接后生成可执行文件。运行该程序,检查程序输出结果是否正确,如发现错误,检查程序的逻辑是否正确,主要解决算法设计错误。

#### 7. 测试程序

测试程序是对照问题的要求,通过让程序试运行一组数据,分析输出的结果,检查是否达到

问题要求的功能和精度要求，输出信息的格式是否符合要求。这组测试数据应是以任何程序都是有错误的前提精心设计出来的，称为测试用例。

测试有黑盒测试和白盒测试两种方法。对于不同的测试方法有不同的测试用例。

(1) 黑盒测试也称为功能测试或数据驱动测试。它把程序看成一个黑盒子，完全不考虑程序的内部结构和处理过程，只对程序的接口进行测试，即检查程序是否能适当地接受输入数据并产生正确的输出信息。实际上目前有些软件开发商推出的软件  $\beta$  版即为测试版，免费提供给用户使用，从使用角度找出软件的问题，这属于黑盒测试方法。

(2) 白盒测试是把程序看成一个透明的白盒子，也就是完全了解程序的内部结构和处理过程。这种方法按照程序内部的逻辑来测试，检验程序中的每条通路是否正确工作。因此白盒测试又称为结构测试或逻辑驱动测试。白盒测试一般由计算机专业人员进行。

例如，对于求解一元二次方程根： $ax^2+bx+c=0$ ，测试用例可为：

$a=0, b=0, c=0, b^2-4ac \geq 0, b^2-4ac < 0$

等各种特殊情况时对应输入  $a$ 、 $b$ 、 $c$  的值，观察程序运行的结果，属于白盒测试法；若输入任何  $a$ 、 $b$ 、 $c$  的值，包括非法的数据，观察程序运行的结果，属于黑盒测试法。

## 8. 编写程序文档

在程序准确无误后，要认真编写程序文档。其包括设计要求、设计思路、设计过程、使用的算法、数据结构、输出信息及格式等，在源程序中要用注释语句加上必要的说明。如果没有程序文档，编制的程序过一段时间后自己也看不懂，更不要说给别人看，对此要引以为戒，要学会从学习程序设计开始就养成良好的习惯。

# 1.2 C 语言及其特点

C 语言是一种得到广泛重视并普遍应用的计算机程序设计语言，也是国际公认的最重要的几种通用程序设计语言之一，它既可用来编写系统软件，也可用来编写应用软件。

## 1.2.1 C 语言的发展过程

C 语言的发展是一个充实和完善的过程。

C 语言是由贝尔实验室的丹尼斯·里奇 (Dennis Ritchie) 和布莱恩·柯林汉 (Brian Kernighan) 根据 B 语言开发出来的，而 B 语言又是由一种早期的编程语言 BCPL (Basic Combined Programming Language) 发展演变而来的。BCPL 的根源可以追溯到 1960 年的 ALGOL 60 (Algol Programming Language)。ALGOL 60 是一种面向问题的高级语言，离硬件较远。1963 年，英国剑桥大学推出 CPL (Combined Programming Language)，CPL 修改了 ALGOL 60，使其能够直接作较低层次的操作。1967 年，英国剑桥大学的 Martin Richards 对 CPL 做了改进，推出了 BCPL 语言。1970 年，美国贝尔实验室的 Ken Thompson 以 BCPL 为基础，又做了进一步简化，设计出了很简单的而且很接近硬件的 B 语言 (取 BCPL 的第一个字母)，并用 B 语言写了第一个 UNIX 操作系统，在 PDP-7 上实现。1971 年，在 PDP-11/20 上实现了 B 语言，并写了 UNIX 操作系统。但 B 语言过于简单，功能有限。1972 年至 1973 年间，Dennis Ritchie 在 B 语言的基础上增加了数据类型 (Datatype) 和结构 (Structure) 设计出了 C 语言 (取 BCPL 的第二个字母)。C 语言既保持了 BCPL 和 B 语言的优点 (精练，接近硬件)，又克服了它们的缺点

(过于简单,数据无类型等)。

最初的 C 语言是为描述和实现 UNIX 操作系统提供的一种工具语言。但 C 语言并没有被束缚在任何特定的硬件或操作系统上,它具有良好的可移植性。1977 年出现了不依赖于具体机器的 C 语言编译文本——《可移植 C 语言编译程序》,用该程序编写的 UNIX 系统迅速在各种机器上实现,UNIX 系统支持的 C 语言也被移植到相应的计算机上。C 语言和 UNIX 系统在发展过程中相辅相成,得到了广泛应用,使它先后被移植到各种大、中、小、微型计算机上。

以 1978 年发表的第七版本 UNIX 系统中的 C 语言编译程序为基础,B.W.Kernighan 和 D.M.Ritchie 合著了《The C Programming Language》。这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础,被称为标准 C 语言。1983 年,美国国家标准化协会(ANSI)根据 C 语言问世以来的各种版本对 C 语言的发展和扩充制定了新的标准,称为 ANSI C。1990 年,C 语言成为国际标准化组织(ISO)通过的标准语言。

## 1.2.2 C 语言的特点

一种语言之所以能够存在和发展,并具有生命力,总是有不同于其他语言的特点。C 语言也是如此,它的特点是多方面的,人们从不同的角度可总结出众多的特点,但若全面考虑,可归纳为以下几点。

### 1. C 语言是比较低级的语言

有人把 C 语言称为高级语言中的低级语言,也有人称它是中级语言。它具有许多通常只有像汇编语言才具备的功能,如位操作、直接访问物理地址等,这使 C 语言在进行系统程序设计时显得非常有效,而过去系统软件通常只能用汇编语言编写。事实上,C 语言的许多应用场合是汇编语言的传统领地,现在用 C 语言代替汇编语言,使程序员得以减轻负担,提高效率,而且写出的程序具有更好的可移植性。

C 语言简洁紧凑,灵活方便。C 语言只有 32 个关键字,9 种控制语句,程序书写自由,主要用小写字母表示。它把高级语言的基本结构和语句与低级语言的实用性相结合。

### 2. C 语言是结构化的程序设计语言

C 语言的主要结构成分是函数,函数允许一个程序中的各任务分别定义和编码,使程序模块化。C 语言还提供了多种结构化的控制语句,如用于循环的 for、while、do-while 语句,用于判定的 if-else、switch 语句等,十分便于采用自顶向下、逐步细化的结构化程序设计技术。因此,用 C 语言编制的程序容易理解、便于维护。

### 3. C 语言具有丰富的运算能力

在 C 语言中,除了一般高级语言使用的算术运算及逻辑运算功能外,还具有独特的以二进制位(bit)为单位的位与、位或、位非以及移位操作等运算。并且 C 语言具有如 a++、b-等单项运算和+=、-=等复合运算功能。

### 4. C 语言数据类型丰富,具有现代化语言的各种数据类型

C 语言的基本数据类型有整型(int)、浮点型(float)、字符型(char)。在此基础上,按层次可产生各种构造类型,如数组、指针、结构体、共用体等。同时还提供了用户自定义数据类型。用这些数据类型可以实现复杂的数据结构,如栈、链表、树等。因此,C 语言具有较强的数据处理能力。

### 5. C 语言具有预处理能力

在 C 语言中提供了#include 和#define 两个预处理命令来实现对外部文件的包含以及对字符

串的宏定义。同时还具有`#if ~ #else` 等条件编译预处理语句。这些功能的使用提高了软件开发的工作效率，并为程序的组织和编译提供了便利。

### 6. C 语言可移植性好

目前，C 语言在许多机器上实现大部分是由 C 语言编译移植得到的。C 编译程序的可移植性，也就使 C 语言程序便于移植。

C 语言的优点很多，但也有一些不足。例如，语法限制不太严格、类型检验太弱、不同类型数据转换比较随便，这就要求程序员对程序设计的方法和技巧更熟练，以保证程序的正确性。

总之，C 语言已成为国内外广泛使用的一种编程语言，它不仅是面向过程的程序设计语言中功能最强、效率最高的语言，更是面向对象程序设计语言 C++、Java 和 C#的基础，并且非常适合用于程序设计语言课程的教学工作中。

## 1.3 简单的 C 语言程序

学习 C 语言的关键是要把握程序设计的思想，反复实践，培养自己独立编写程序的能力。下面通过几个简单的 C 程序实例介绍 C 语言程序的基本结构和编写方法，使读者对 C 语言程序有一个初步的认识。

**例 1.1** 编写 C 语言程序，实现在屏幕上显示字符串 “This is a C program.”。

具体程序如下：

```
/* ex1_1.c 在屏幕上输出字符串 */
#include "stdio.h"
void main(void)
{
    printf("This is a C program.\n");
}
```

运行结果：

This is a C program.

说明：

(1) 这是一个完整的 C 语言程序。C 语言程序结构一般由注释、编译预处理和程序主体组成。

(2) `/* ..... */`部分为注释语句。注释是程序员为读者作的说明，不影响程序的执行。注释一般分为两种：序言注释和注解性注释。前者用于程序开头，说明程序或文件的名称、用途、编写时间、编写人员以及输入/输出说明等；后者用于对程序中难于理解的部分加以解释说明。

注释使程序变得清晰，能帮助读者阅读和理解程序，提高程序的可读性。给程序加注释是一个良好的编程习惯。C 语言注释部分由“`/*`”开始，至“`*/`”结束，应括在`/* ..... */`之间，`/`和`*`之间不允许留有空格。注释部分允许出现在程序中的任何位置上。注释可占多行，但不允许在`/* ..... */`中间又出现`/* ..... */`注释，即不允许嵌套。

(3) 其中以“`#`”开头的语句是预处理命令。这些命令是在编译系统翻译代码之前需要由预处理程序处理的语句。本例中的`#include`称为文件包含预处理命令。

`#include "stdio.h"`的作用是在编译之前请求预处理程序将文件 `stdio.h` 的内容增加（包含）到

程序 ex1\_1.c 中来, 以作为程序的一部分。文件 `stdio.h` 是系统定义的一个头文件, 它为输入和输出提供支持, 在本程序中的 `printf("This is a C program.\n");` 语句的执行需要 `stdio.h` 的支持, 没有它, 程序将不能通过编译系统的翻译。

(4) `void main(void)` 表明以下是一个 C 程序的主函数, 每一个 C 程序都必须有一个 `main()` 函数, `main()` 是程序的入口。`main` 前面的 `void` 表示该 `main()` 函数没有返回值。用 `{}` 括起来的部分是函数体。描述一个函数所执行算法的过程称为函数定义。例如, 此程序中的 `main()` 函数头和函数体构成了一个完整的函数定义。

(5) `printf()` 是 C 语言提供的标准输出库函数, 它的功能是将一对双引号中的内容 (称为字符串常量) 输出到标准输出设备显示器上。“`\n`” 是换行符, 即在输出字符串 “This is a C program.” 后回车换行。`printf()` 后的分号是语句的结束符, C 语言程序的每一个语句必须以分号 “`;`” 结束。

**例 1.2** 编写程序, 从键盘输入两个实数, 计算并输出这两个实数平方之和的平方根。

具体程序如下:

```
/* ex1_2.c 计算两个实数平方之和的平方根 */
#include "stdio.h"
# include "math.h"
void main(void)
{
    double x,y,s;                                /* 定义 x, y 和 s 3 个双精度实型变量 */
    printf("Input two real numbers: ");          /* 输出一行提示信息 */
    scanf("%lf%lf",&x,&y);                        /* 用标准输入函数输入两个实数, 赋值给 x 和 y */

    s=sqrt(x*x+y*y);                              /* 计算  $\sqrt{x^2 + y^2}$ , 并将计算结果赋给变量 s */
    printf("s= %lf\n",s);                         /* 将 s 变量的值输出到屏幕上 */
}
```

运行输入:

Input two real numbers: 3.0 4.0 <回车>

运行结果:

s=5.000000

说明:

(1) 当运行该程序时, 首先, 屏幕上显示一条提示信息:

Input two real numbers:

此时用户从键盘输入两个数。如果此时用户输入 3.0 和 4.0, 即

Input two real numbers: 3.0 4.0 <回车>

带下划线部分是由用户输入的, 输入 <回车> 以示输入结束。此时屏幕显示运行结果 (s=5.000000)。

(2) 该程序从 `main()` 开始运行。`double x,y,s;` 是数据类型说明语句, 把 `x`、`y` 和 `s` 定义为双精度实型 (`double`) 变量, `double` 是类型说明符。值得注意的是, 所有 C 语言程序中的变量, 必须在声明其数据类型之后才能使用, 以便给变量分配相应的内存空间, 用来存放变量的值。

(3) `printf("Input two real numbers: ");` 输出语句的作用是在屏幕的当前光标位置处显示输入提示, 提醒用户需要从键盘输入两个实数。

(4) `scanf("%lf%lf",&x,&y);` 是输入语句, 其中 `scanf()` 是 C 语言提供的标准输入库函数, 它