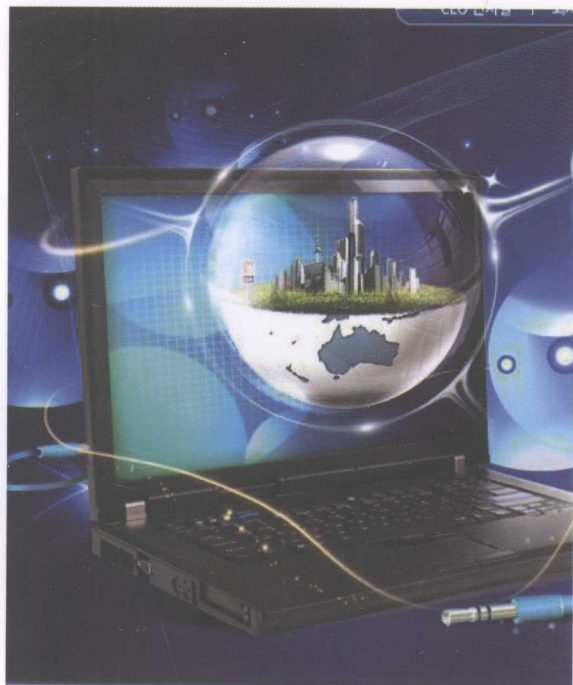


# C语言程序设计

- ◆ C语言概述
- ◆ 数据类型、运算符与表达式
- ◆ 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
- ◆ 指针
- ◆ 结构体与共用体
- ◆ 文件



杜 红 邓绍金 主 编  
王圆妹 伍 鹏 副主编

清华大学出版社

014004557

TP312C-43  
849

高等学校计算机应用规划教材

# C 语言程序设计

主 编 杜 红 邓绍金

副主编 王圆妹 伍 鹏



清华大学出版社

北 京



北航

C1691682

TP312C-43  
849

## 内 容 简 介

本书是一本关于C语言程序设计基础及应用的教程,共由10章内容构成,全面系统地介绍了C语言程序设计的基本概念及实现方式,包括C语言概述、数据类型、运算符与表达式、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、指针、结构体与共用体以及文件等内容。

本书在传统教材的编写模式基础上,体现了模块化编程、结构化程序设计的特点。以实例为导引,将知识点全面概括在应用实例中;对每一个实例实现方式及程序代码均给出注释,方便阅读、理解;配有大量课后习题,方便检测和巩固学习成果,并做到理论与实践相结合,突出应用;引入一些实例与工程应用紧密结合;每一章首列出本章要掌握的内容,后面有小结以及测试习题,便于自学和自测。此外,本书的配套教程《C语言习题集与实验指导》(ISBN: 978-7-302-32449-2)在习题和实践环节上对本书进行了补充和指导。

本书适用于高等学校各专业程序设计基础教学,特别适合应用型本科计算机及非计算机相关专业的学生使用,同时也是计算机等级考试备考的一本实用辅导书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

### 图书在版编目(CIP)数据

C语言程序设计 / 杜红, 邓绍金 主编. —北京: 清华大学出版社, 2013.9

(高等学校计算机应用规划教材)

ISBN 978-7-302-33413-2

I. ①C… II. ①杜… ②邓… 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(2013)第180802号

责任编辑: 王 定 程 琪

装帧设计: 牛艳敏

责任校对: 邱晓玉

责任印制: 何 芊

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦A座 邮 编: 100084

社总机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, [c-service@tup.tsinghua.edu.cn](mailto:c-service@tup.tsinghua.edu.cn)

质量反馈: 010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

课 件 下 载: <http://www.tup.com.cn>, 010-62794504

印 刷 者: 北京世知印务有限公司

装 订 者: 三河市漂源装订厂

经 销: 全国新华书店

开 本: 185mm×260mm

印 张: 17.75

字 数: 410千字

版 次: 2013年9月第1版

印 次: 2013年9月第1次印刷

印 数: 1~4000

定 价: 32.00元

产品编号: 051908-01

## 前 言

从计算机诞生到今天,人们学习、工作、生活、娱乐的方方面面都已经离不开计算机和计算机技术了。随着信息技术的进步,计算机技术得到了快速的发展。在计算机技术的各个组成部分中,计算机软件设计占据着重要的地位,它为人们更好地利用计算机解决问题提供了重要途径。

C 语言作为一门最通用的语言,是其他语言的基础。C 语言从诞生到现在,已经成为最重要和最流行的编程语言之一,是进一步学习“面向对象程序设计”、“数据结构”、“算法设计与分析”等课程的先导课程之一。

C 语言具有高级语言的强大功能,又有很多直接操作计算机硬件的功能(这些都是汇编语言的功能),因此,C 语言通常又被称为中级语言。学习和掌握 C 语言,既可以增进对于计算机底层工作机制的了解,又可以为进一步学习其他高级语言打下坚实的基础。

C 语言具有表达能力强、功能丰富、目标程序质量高、可移植性好、使用灵活等特点。因此既具有高级语言的优点,又具有低级语言的某些特性,特别适合于编写系统软件和嵌入式软件。C 语言的上述特点使得我国大部分高等院校都把 C 语言作为计算机和非计算机专业的第一门程序设计语言课程。全国计算机等级考试、全国计算机应用技术证书考试也都把 C 语言列入考试范围。

本书立足于应用型本科教育,以培养应用型人才为主要目标,在介绍 C 语言的基本概念和知识的同时,重点突出应用和实践,全面培养学生计算机程序设计的能力。本教材面向程序设计初学者,在编写教材过程中,作者力求用读者容易理解的体系和叙述方法,深入浅出、循序渐进地帮助读者更好地掌握 C 语言程序设计的基本内容和方法,以总结的形式,对各章内容进行归纳,以便于学生对重要知识点内容的掌握。书中精心选编了大量例题及习题,供老师在教学中根据需要进行选择,方便学生通过多读程序例子和多动手上机编程,以达到开阔思路和提高程序设计能力之目的。

本书全面翔实地介绍了 C 语言的基本概念和程序设计方法。第 1 章介绍了 C 程序的结构、特点和上机步骤;第 2 章介绍了 C 语言的基本数据类型、常量与变量,运算符、表达式和常用库函数的概念和应用;第 3 章、第 4 章、第 5 章介绍了三种结构化程序设计方法;第 6 章介绍了一维数组、二维数组的应用,以及字符数组和字符串处理的方法;第 7 章综述了 C 语言的函数和模块化程序设计的方法,介绍了变量的作用域、变量的存储类型等概念;第 8 章介绍了指针及其应用;第 9 章介绍了结构体、共用体的概念和应用以及位和位段的操作;第 10 章介绍了文件的概念、文本文件和二进制文件的读写方法及其应用。

本书第 1 章、第 6 章和第 10 章由邓绍金编写,第 2 章、第 8 章由伍鹏编写,第 3 章、第 4 章和第 5 章由王圆妹编写,第 7 章、第 9 章和附录由杜红编写。全书由杜红负责统稿。

在本书编写过程中,长江大学教务处、电子信息学院的领导和老师也给予了很大的支持并提出了许多宝贵意见,使本书更具实用性。在此,对他们表示衷心感谢!也谢谢汤有禄同学的帮助。

由于时间紧迫,加之编者水平有限,书中疏漏之处在所难免,敬请读者批评指正。

编 者

2013 年 7 月

## 目

## 录

|                                  |    |
|----------------------------------|----|
| 第 1 章 C 语言概述 .....               | 1  |
| 1.1 C 语言的发展简史和特点 .....           | 1  |
| 1.1.1 C 语言的起源与发展 .....           | 1  |
| 1.1.2 C 语言的特点 .....              | 2  |
| 1.2 C 语言的程序结构与书写规则 .....         | 3  |
| 1.2.1 C 语言程序的构成 .....            | 3  |
| 1.2.2 C 语言程序的结构特点及<br>书写规则 ..... | 5  |
| 1.3 C 语言的语句及关键字 .....            | 6  |
| 1.3.1 C 语言的语句 .....              | 6  |
| 1.3.2 关键字 .....                  | 7  |
| 1.4 VC++ 6.0 集成开发环境 .....        | 8  |
| 1.4.1 C 程序一般运行步骤 .....           | 8  |
| 1.4.2 Visual C++ 集成开发环境 .....    | 8  |
| 1.5 小结 .....                     | 16 |
| 1.6 习题 .....                     | 17 |
| 第 2 章 数据类型、运算符与表达式 .....         | 18 |
| 2.1 C 语言的数据类型 .....              | 18 |
| 2.2 常量和变量 .....                  | 19 |
| 2.2.1 标识符 .....                  | 19 |
| 2.2.2 常量 .....                   | 20 |
| 2.2.3 变量 .....                   | 20 |
| 2.3 整型数据 .....                   | 21 |
| 2.3.1 整型常量 .....                 | 21 |
| 2.3.2 整型变量 .....                 | 22 |
| 2.4 实型数据 .....                   | 22 |
| 2.4.1 实型常量 .....                 | 22 |
| 2.4.2 实型变量 .....                 | 23 |
| 2.5 字符型数据 .....                  | 23 |
| 2.5.1 字符常量 .....                 | 23 |
| 2.5.2 字符串常量 .....                | 24 |

|                              |    |
|------------------------------|----|
| 2.5.3 字符型变量 .....            | 25 |
| 2.6 算术运算符与算术表达式 .....        | 26 |
| 2.6.1 算术运算符 .....            | 26 |
| 2.6.2 算术表达式 .....            | 27 |
| 2.6.3 数据类型转换 .....           | 28 |
| 2.7 赋值运算符与赋值表达式 .....        | 30 |
| 2.7.1 赋值运算符 .....            | 30 |
| 2.7.2 赋值表达式 .....            | 31 |
| 2.8 C 语言特有的运算符与<br>表达式 ..... | 32 |
| 2.8.1 自增(++)和自减(--)运算符 ..... | 32 |
| 2.8.2 逗号(,)运算符及其表达式 .....    | 33 |
| 2.8.3 条件运算符(?:)及其表达式 .....   | 33 |
| 2.9 小结 .....                 | 34 |
| 2.10 习题 .....                | 35 |
| 第 3 章 顺序结构程序设计 .....         | 36 |
| 3.1 数据的输入和输出 .....           | 36 |
| 3.1.1 格式输出函数 .....           | 37 |
| 3.1.2 格式输入函数 .....           | 45 |
| 3.1.3 字符输入、输出函数 .....        | 49 |
| 3.2 顺序结构程序设计 .....           | 51 |
| 3.3 小结 .....                 | 54 |
| 3.4 习题 .....                 | 55 |
| 第 4 章 选择结构程序设计 .....         | 58 |
| 4.1 关系运算符和关系表达式 .....        | 58 |
| 4.1.1 关系运算符及其优先次序 .....      | 58 |
| 4.1.2 关系表达式 .....            | 59 |
| 4.2 逻辑运算符和逻辑表达式 .....        | 60 |
| 4.2.1 逻辑运算符 .....            | 61 |
| 4.2.2 逻辑表达式 .....            | 62 |
| 4.2.3 逻辑表达式的应用 .....         | 62 |

|                                 |     |                         |     |
|---------------------------------|-----|-------------------------|-----|
| 4.3 if 语句和条件表达式 .....           | 63  | 6.3.5 字符串的输入/输出 .....   | 125 |
| 4.3.1 if 语句的三种形式 .....          | 63  | 6.3.6 字符串处理函数 .....     | 126 |
| 4.3.2 if 语句的嵌套 .....            | 67  | 6.4 数组应用实例 .....        | 130 |
| 4.3.3 条件表达式 .....               | 68  | 6.5 小结 .....            | 134 |
| 4.4 switch 语句 .....             | 70  | 6.6 习题 .....            | 136 |
| 4.5 程序举例 .....                  | 72  | 第 7 章 函数 .....          | 138 |
| 4.6 小结 .....                    | 77  | 7.1 函数概述 .....          | 138 |
| 4.7 习题 .....                    | 79  | 7.2 函数的定义 .....         | 139 |
| 第 5 章 循环结构程序设计 .....            | 82  | 7.2.1 函数的定义形式 .....     | 139 |
| 5.1 while 语句 .....              | 82  | 7.2.2 用户函数的分类 .....     | 140 |
| 5.2 do-while 语句 .....           | 85  | 7.3 函数的调用 .....         | 143 |
| 5.3 for 语句 .....                | 87  | 7.3.1 函数的调用形式 .....     | 143 |
| 5.4 break 语句和 continue 语句 ..... | 91  | 7.3.2 函数调用时的参数传递 .....  | 144 |
| 5.4.1 break 语句 .....            | 91  | 7.3.3 函数的返回值 .....      | 145 |
| 5.4.2 continue 语句 .....         | 94  | 7.3.4 函数的声明 .....       | 145 |
| 5.5 循环的嵌套 .....                 | 96  | 7.4 函数的递归调用 .....       | 146 |
| 5.6 程序举例 .....                  | 97  | 7.4.1 递归调用 .....        | 146 |
| 5.7 小结 .....                    | 103 | 7.4.2 递归调用的优缺点 .....    | 149 |
| 5.8 习题 .....                    | 105 | 7.5 数组作为函数的参数传递 .....   | 149 |
| 第 6 章 数组 .....                  | 108 | 7.5.1 数组元素作为函数参数 .....  | 149 |
| 6.1 一维数组 .....                  | 108 | 7.5.2 数组名作为函数参数 .....   | 150 |
| 6.1.1 一维数组的定义 .....             | 108 | 7.5.3 多维数组名作为函数参数 ..... | 154 |
| 6.1.2 一维数组的初始化 .....            | 109 | 7.6 变量的作用域和生存期 .....    | 155 |
| 6.1.3 一维数组元素的引用 .....           | 110 | 7.6.1 作用域 .....         | 155 |
| 6.1.4 一维数组的边界 .....             | 111 | 7.6.2 生存期 .....         | 158 |
| 6.1.5 一维数组的应用 .....             | 112 | 7.6.3 变量的作用域和生存期 .....  | 160 |
| 6.2 二维数组 .....                  | 115 | 总结 .....                | 160 |
| 6.2.1 二维数组的定义 .....             | 116 | 7.7 小结 .....            | 160 |
| 6.2.2 二维数组的初始化 .....            | 116 | 7.8 习题 .....            | 162 |
| 6.2.3 二维数组元素的引用 .....           | 117 | 第 8 章 指针 .....          | 165 |
| 6.2.4 二维数组的应用 .....             | 118 | 8.1 指针和地址 .....         | 165 |
| 6.3 字符数组 .....                  | 121 | 8.1.1 变量的地址 .....       | 165 |
| 6.3.1 字符数组的定义 .....             | 121 | 8.1.2 指针变量 .....        | 166 |
| 6.3.2 字符数组的初始化 .....            | 122 | 8.2 指针变量的定义、初始化及 .....  | 167 |
| 6.3.3 字符数组元素的引用 .....           | 123 | 使用 .....                | 167 |
| 6.3.4 字符串的初始化 .....             | 124 | 8.2.1 指针变量的定义及初始化 ..... | 167 |

|                                |     |                                      |     |
|--------------------------------|-----|--------------------------------------|-----|
| 8.2.2 指针变量的使用 .....            | 170 | 9.6.1 链表的概念 .....                    | 218 |
| 8.3 指针的运算 .....                | 172 | 9.6.2 单链表的定义 .....                   | 218 |
| 8.3.1 指针的算术运算 .....            | 172 | 9.6.3 单链表的基本操作 .....                 | 219 |
| 8.3.2 指针的关系运算 .....            | 176 | 9.6.4 单链表的应用 .....                   | 222 |
| 8.3.3 指针的赋值运算 .....            | 177 | 9.7 枚举类型 .....                       | 225 |
| 8.4 指针与数组及字符串 .....            | 179 | 9.7.1 枚举类型的定义 .....                  | 225 |
| 8.4.1 指针与数组 .....              | 179 | 9.7.2 枚举类型变量的赋值和<br>使用 .....         | 226 |
| 8.4.2 字符指针与字符串 .....           | 183 | 9.8 类型重定义 typedef .....              | 227 |
| 8.5 指针数组和多级指针 .....            | 184 | 9.8.1 类型重定义形式 .....                  | 228 |
| 8.5.1 指针数组 .....               | 184 | 9.8.2 类型重定义应用 .....                  | 228 |
| 8.5.2 多级指针 .....               | 188 | 9.8.3 类型重定义举例 .....                  | 229 |
| 8.6 小结 .....                   | 192 | 9.9 位操作 .....                        | 231 |
| 8.7 习题 .....                   | 194 | 9.9.1 按位与运算(&) .....                 | 232 |
| 第9章 结构体与共用体 .....              | 196 | 9.9.2 按位或运算( ) .....                 | 232 |
| 9.1 结构体类型 .....                | 196 | 9.9.3 按位异或运算(^) .....                | 232 |
| 9.1.1 结构体的定义 .....             | 196 | 9.9.4 按位取反运算(~) .....                | 233 |
| 9.1.2 结构体的嵌套定义 .....           | 197 | 9.9.5 左移运算(<<) .....                 | 233 |
| 9.1.3 结构体变量的定义 .....           | 198 | 9.9.6 右移运算(>>) .....                 | 233 |
| 9.1.4 结构体成员的引用和<br>变量的使用 ..... | 200 | 9.9.7 位运算复合赋值运算符 .....               | 234 |
| 9.1.5 结构体初始化 .....             | 201 | 9.10 位段 .....                        | 235 |
| 9.2 结构体数组 .....                | 204 | 9.10.1 位段的定义 .....                   | 235 |
| 9.2.1 结构体数组的定义和<br>初始化 .....   | 204 | 9.10.2 位段的应用 .....                   | 235 |
| 9.2.2 结构体数组元素的引用 .....         | 206 | 9.11 小结 .....                        | 237 |
| 9.3 结构体指针 .....                | 207 | 9.12 习题 .....                        | 238 |
| 9.3.1 结构体指针的定义和访问 .....        | 207 | 第10章 文件 .....                        | 240 |
| 9.3.2 指向结构体数组的指针 .....         | 209 | 10.1 文件概述 .....                      | 240 |
| 9.3.3 结构体指针作为函数参数 .....        | 211 | 10.2 文件类型指针 .....                    | 243 |
| 9.4 共用体(联合体)类型 .....           | 212 | 10.3 文件的打开与关闭 .....                  | 243 |
| 9.4.1 共用体的定义 .....             | 212 | 10.3.1 文件打开函数<br>(fopen 函数) .....    | 244 |
| 9.4.2 共用体变量的定义和<br>存储分配 .....  | 213 | 10.3.2 文件关闭函数<br>(fclose 函数) .....   | 245 |
| 9.4.3 共用体成员的访问 .....           | 215 | 10.4 文件的读写 .....                     | 246 |
| 9.5 动态存储分配 .....               | 217 | 10.4.1 字符读写函数 fgetc 和<br>fputc ..... | 246 |
| 9.6 链表 .....                   | 218 |                                      |     |

|      |                             |
|------|-----------------------------|
| 815  | 10.4.2 字符串读写函数 fgetc 和      |
| 815  | fputc ..... 249             |
| 815  | 10.4.3 数据块读写函数 fread 和      |
| 555  | fwrite ..... 251            |
| 555  | 10.4.4 格式化读写函数 fscanf 和     |
| 555  | fprintf ..... 253           |
| 10.5 | 文件的定位 ..... 255             |
| 555  | 10.5.1 文件定位函数 ..... 255     |
| 555  | 10.5.2 文件的随机读写 ..... 256    |
| 855  | 10.5.3 文件位置确定 ..... 257     |
| 855  | 10.6 文件检测函数 ..... 258       |
| 555  | 10.6.1 文件结束检测函数 ..... 258   |
| 555  | 10.6.2 读写文件出错检测函数 ..... 259 |

|      |                          |
|------|--------------------------|
| 1071 | 10.6.3 文件出错标志和文件         |
| 175  | 结束标志清除函数 ..... 259       |
| 555  | 10.7 文件输入/输出小结 ..... 259 |
| 175  | 10.8 小结 ..... 260        |
| 175  | 10.9 习题 ..... 262        |
| 附录   | ..... 263                |

|      |                      |
|------|----------------------|
| 附录 A | ISO/ANSI C90(C99)    |
|      | 标准关键字 ..... 263      |
| 附录 B | 标准 ASCII 码与扩展        |
|      | ASCII 码对照表 ..... 263 |
| 附录 C | 运算符和结合性列表 ..... 266  |
| 附录 D | 系统库和常用库函数 ..... 268  |

# 第1章 C语言概述

C语言是国际上流行的计算机高级语言，它是一种强大的专业化编程语言，既可用于编写系统软件，也可用于编写应用软件。自从C语言诞生以来，其强大的功能和各方面的优点逐渐为人们认识，它的应用可以适应不同的操作系统，从20世纪70年代的UNIX系统，到20世纪80年代的Windows系统、Linux系统，也在各类大、中、小和微型计算机上得到了广泛的使用，现在已然成为当代最优秀的程序设计语言之一。

C语言也是一种理想的结构化语言，具有描述能力强的特点，是相关专业进行计算机语言学习中比较理想的示范语言。掌握了C语言后，再学其他的语言就比较容易，通过简单的变通，就能够适应其他语言规范，进而完成其他环境下的应用程序设计。

本章通过介绍C语言的发展及特点、C程序总体结构与书写规则、C语言的语句与关键字的概念，以及VC++运行环境下的基本操作，从一个比较低的层次，让读者尽快掌握C程序设计理念。

## 本章应掌握的内容

- C语言的产生及特点
- C程序的结构特征
- C语言的语句规范
- C语言的关键字
- C程序的运行环境及运行步骤

## 1.1 C语言的发展简史和特点

### 1.1.1 C语言的起源与发展

在C语言诞生之前，汇编语言是编写系统软件的主要语言。由于汇编语言程序是依赖于计算机硬件来实现，其可读性和可移植性很差，其他的高级语言又难以实现对计算机硬件的直接操作(汇编语言的特点)，于是人们期望有一种兼有汇编语言和高级语言特性的新语言出现。

C语言最早的原型是ALGOL60，也称为“A语言”。1963年，剑桥大学将其发展成为CPL语言(Combined Programming Language)。1967年，剑桥大学的Martin Richards对CPL语言进行了简化，产生了BCPL语言。有趣的是，1970年，美国AT&T贝尔实验室的Ken Thompson将BCPL进行了修改，将其取名为“B语言”，并用B语言写了第一个UNIX

系统。1973 年, AT&T 贝尔实验室的 Dennis Ritchie 在 BCPL 和 B 语言的基础上设计出了一种新的语言, 将其取名为“C 语言”。随后不久, UNIX 的内核(Kernel)和应用程序全部用 C 语言改写, 从此, C 语言成为 UNIX 环境下使用最广泛的主流编程语言。在 C 语言的发展过程中, 最具代表的是 1978 年, Brian W.Kernighan 和 Dennis M.Ritchie 出版了名著 *The C Programming Language*, 从而使 C 语言成为目前世界上流行最广泛的高级程序设计语言。1988 年, 随着微型计算机的日益普及, 出现了许多 C 语言版本。由于没有统一的标准, 使得这些 C 语言之间出现了一些不一致的地方。为了改变这种情况, 美国国家标准研究所(ANSI)为 C 语言制定了一套 ANSI 标准, 成为现行的 C 语言标准。1989 年, 草案被 ANSI 正式通过成为美国国家标准, 被称为 C89 标准。1990 年, 国际标准化组织(International Organization for Standardization, ISO)批准 ANSI C 成为国际标准, 被称为 C90 标准。之后, ISO 在 1995 年对 C90 进行技术补充, 推出 C95 标准。1999 年, ANSI 和 ISO 又通过了最新版本的 C 语言标准和技术勘误文档, 该标准被称为 C99。

### 1.1.2 C 语言的特点

C 语言作为一种计算机程序设计语言, 它既有高级语言的特点, 又具有汇编语言的特征。它可以应用在操作系统层面、应用程序设计层面及需要对硬件进行操作的环境, 因此, 它的应用范围广泛, 主要特点如下。

(1) 语言简洁、紧凑, 使用灵活、方便。C99 标准规定, C 语言一共只有 37 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, 从而满足编程人员如汇编语言一样对计算机硬件操作的需要。

(2) 运算符丰富。C 语言的运算符包含的范围很广泛, 共有 34 个运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理, 从而使 C 的运算类型极其丰富, 表达式类型多样化, 通过灵活使用各种运算符, 可以实现在其他高级语言中难以实现的运算。

(3) 数据类型丰富。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, 并能通过现有数据类型来构建各种复杂的数据类型。

(4) 模块化设计、结构化编程。模块化设计的显著特点是以功能模块为单位进行程序设计, 各个模块相对独立、功能单一、结构清晰、接口简单。C 语言提供了各种多样的系统库函数给用户, 用户还可以根据需要自定义函数, 这种模块化程序设计可使程序易于调试、使用以及维护。任何程序都可由顺序、选择、循环三种基本控制结构来构造。

(5) C 语言的语法限制不太严格、程序设计自由度大。一般的高级语言语法检查比较严格, 能够检查出几乎所有的语法错误, 而 C 语言通过放宽语法检查, 允许程序编写者有较大的自由度。

(6) C 语言允许直接访问物理地址, 可以直接对硬件进行操作。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, 还能够像汇编语言一样对位、字节和地址进行操作, 运用这一特性, 能够较好地编写应用软件和系统软件。

(7) C 语言程序生成代码质量高, 程序执行效率高。由于 C 语言合理地运用指针数据

类型,并具有汇编语言的部分功能,因而,C语言程序生成的目标代码执行效率会很高,仅比汇编程序生成的目标代码效率低10%~20%。

(8) C语言适用范围大,可移植性好。C语言能够适合于多种操作系统,如DOS、UNIX,也能够适用于多种机型,因而它具有适用范围大、可移植性好的特点。

事物都是两面的,C语言在表达方面的自由会增加程序的风险,指针的运用可能会使程序的错误难以追踪,C语言的简洁性与其丰富的运算符相结合,使其可能会编写出极难理解的代码,降低了程序代码的可读性。

## 1.2 C语言的程序结构与书写规则

### 1.2.1 C语言程序的构成

在介绍C程序的结构之前,先来看几个C程序的例子。

为了帮助程序员阅读程序,理解程序的功能,在程序中添加必要的注释,其中“/\*...\*/”为块注释,“//”为行注释,注释的内容可以任意,但对编译和运行不起任何作用。

【例1.1】由main()函数构成C语言程序,在屏幕上输出一行信息。

```
/*功能: 输出文本信息*/
#include<stdio.h>           //编译预处理指令
void main()                 //定义主函数
{                             //main 函数开始标志
    printf("This is a C program.\n"); //输出指定的一行信息
}                             //main 函数结束标志
```

【运行结果】

```
This is a C program.
press any key to continue
```

【说明】第1行为注释。第2行为编译预处理指令,由#include包含扩展名为.h的头文件,由一对“<>”作为引导。第3行main是主函数的函数名,它是整个程序执行的入口,void表示本函数没有返回值。第4行和第6行表示本函数函数体的开始与结束,由一对“{ }”作为定界符。第5行为函数调用语句,printf函数是一个由系统定义的标准函数,其功能是把要输出的内容送到显示器去显示,“\n”表示回车换行,语句由分号结束。输出结果中的press any key to continue为VC++系统提示,表示程序运行结束,按任意键返回系统环境。

【提示】由于press any key to continue是系统对程序运行结束的一种提示,在本书的后面内容中,省略该行内容。

【例 1.2】计算两数之和，并在屏幕上显示其结果。

```
/*功能：求两数的和*/
#include<stdio.h>           //编译预处理指令
int main()                  //定义主函数
{
    int a,b,sum;            //定义变量的数据类型
    a=10;b=20;              //给定义的变量赋值
    sum=a+b;                //计算变量的和
    printf("sum is %d\n",sum); //输出计算结果
    return 0;
}
```

【运行结果】

sum is 30

【说明】该程序的作用是求两个整数之和，然后在屏幕上显示其结果。第 5 行是变量的声明部分，定义了 3 个整型变量 a、b 和 sum。第 6 行是两个赋值语句，分别给 a 和 b 赋值为 10 和 20。第 7 行也是一个赋值语句，计算 sum 的值。第 8 行为函数调用语句，调用系统函数 printf 以指定的格式在屏幕输出 sum 变量的值。

【例 1.3】由主函数和其他函数构成的 C 语言程序，计算两数之和。

```
/*功能：用函数实现两数的和*/
#include<stdio.h>
int main()
{ int add(int x, int y);    //函数的声明
  int a,b,sum;              //定义变量
  a=10;b=20;                //给定义的变量赋值
  sum=add(a,b);             //调用 add 函数，计算变量 a,b 的和
  printf("sum is %d\n",sum); //输出计算结果
  return 0;
}

int add(int x, int y)       //定义一个求两个整数和的函数
{ int z;                    //定义一个整型变量
  z=x+y;                    //计算两变量的和
  return z;                  //返回函数的计算值
}
```

【运行结果】

sum is 30

【说明】本程序包含两个函数的定义：main 函数和 add 函数。add 函数的作用是将 x 和 y 相加，并将计算结果赋给变量 z，通过 return 语句将 z 的值返回给主调函数 main。变量和函数都要“先定义，后使用”，在 main 函数中要调用 add 函数，而 add 函数的位置在

main 函数之后,为了使编译系统能够正确识别和调用 add 函数,必须在调用 add 函数之前对 add 函数进行声明。程序的第 7 行调用了 add 函数,在调用时将 a 和 b 的值(实际参数)分别传送给 add 函数中的参数 x 和 y(称为形式参数),经过用户自定义的 add 函数调用,实现了两个数的求和。第 8 行是输出变量 sum 的值。

### 1.2.2 C 语言程序的结构特点及书写规则

综合上述实例,C 语言程序的结构特点如下。

(1) C 程序由函数构成。一个 C 源程序有且仅有一个 main 函数,还可以包含若干个其他函数。对于 C 源程序,由于函数是 C 程序的基本单位,被调用的函数可以是系统提供的库函数,也可以是用户根据需要自己定义的函数,用于实现特定的功能。程序的全部工作都是由各个函数分别完成的,程序的设计就是模块化的结构。

(2) 一个函数由两部分组成:

- 函数的首部,即函数的第 1 行,包括函数类型、函数名、函数参数(形式参数)。
- 函数体,即函数首部下面的花括号内的部分。如果一个函数内有多层花括号,则最外层的一对花括号为函数体的范围。

函数体一般包括以下两部分。

- 声明部分。在这部分中定义所用到的变量和对所调用函数的声明。
- 执行部分。由若干行语句组成,是完成具体功能的代码。

在某些情况下可以没有声明部分,甚至可以既无声明部分也无执行部分。

```
void dump()  
{  
}  

```

这是一个空函数,什么也不做,但在理论上是合法的。

(3) 一个 C 程序总是从 main 函数开始执行,不论 main 函数在程序中什么位置。并且,一个源程序不论由多少个文件组成,有且仅有一个 main 函数(即主函数)。

(4) C 程序书写格式自由,一行内可以写几条语句,一条语句也可以分写在多行上,C 程序没有行号。标识符、关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符,也可不再加空格来间隔。

(5) 源程序中可以有预处理命令(#include 命令仅是其中的一种),预处理命令通常应放在源文件的最前面。

(6) 每个语句和数据声明的最后必须有一个分号。分号是 C 语句的必要组成部分。但预处理命令、函数头和花括号“}”之后不能加分号。

(7) C 语言本身没有输入/输出语句。输入和输出的操作是由库函数所定义的标准函数来实现的。

(8) 可以用 /\*...\*/ 对 C 程序中的任何部分做块注释。用“//”对一行中右边的内容进行注释。一个好的源程序都应当有必要的注释,以增加程序的可读性。

注意:

为了避免遗漏必须配对使用的符号,如注释符号、函数体的起止标识符(大括号)、圆括号等,在输入时,可连续输入这些起止标识符,然后再在其中进行插入来完成内容的编辑。在起止标识符嵌套以及相距较远时,这样做更有必要,可以保证层次结构的完整性。

## 1.3 C 语言的语句及关键字

### 1.3.1 C 语言的语句

C 语言程序从程序结构来说,可分为三种基本结构,即顺序结构、选择结构、循环结构,这三种基本结构可以组成所有的各种复杂程序。

C 语言程序从程序的执行来说,是由各种语句组成的,按其功能或构成的不同分为以下五类:表达式语句、函数调用语句、控制语句、复合语句、空语句。

#### 1. 表达式语句

表达式语句由表达式加上分号“;”组成。其一般形式为:

表达式;

执行表达式语句就是计算表达式的值。

例如: `x=y+z;` `a=520;`等。

#### 2. 函数调用语句

函数调用语句由函数调用加一个分号“;”构成。其一般形式为:

函数名(实际参数表);

执行函数调用语句就是执行调用函数内部的代码,即执行函数体。在执行时,先把实际参数与函数定义中的形式参数相结合,然后执行被调函数体中的语句,执行相应的功能。

例如:

```
printf("%d,%d,%d",a,b,c); //调用系统库函数 printf 函数
```

#### 3. 控制语句

控制语句用于控制程序的执行顺序,实现程序的各种复杂结构,以完成程序的特定功能。C 语言有九种控制语句,可分成以下三类。

(1) 选择结构判断语句。if 语句、switch 语句。

(2) 循环结构控制语句。while 语句、do-while 语句、for 语句、break 语句、continue 语句。

(3) 转向语句。`return` 语句、`goto` 语句(`goto` 语句尽量少用, 其语句不利于结构化程序设计, 用它后会使得程序流程无规律、可读性变差)。

#### 4. 复合语句

把多条或一条语句用括号“{}”括起来组成一个语句, 称为复合语句。在程序中应把复合语句看成是单条语句, 而不是多条语句。

例如:

```
{  
    x=y+z;  
    a=b+c;  
    printf("%d%d", x, a);  
}
```

是一条复合语句。复合语句内的各条语句都必须以分号“;”结尾。此外, 在括号“}”后不能加分号。复合语句具有如下性质:

- (1) 在语法上和单一语句相同, 单一语句可以出现的地方, 也可以使用复合语句。
- (2) 复合语句可以嵌套, 复合语句也可出现在复合语句中。

#### 5. 空语句

只有分号“;”构成的语句称为空语句, 空语句什么也不执行。在程序中空语句可用来作空循环体。

例如, `while(getchar()!='#');` 语句的功能是: 只要从键盘输入的字符不是“#”字符就重新输入。这里的循环体为空语句。

### 1.3.2 关键字

C 语言关键字就是被 C 语言本身使用的, 不能作其他用途使用的字。关键字不能用作变量名、函数名。在 C99 标准中, C 语言的关键字共有 37 个, 根据关键字的作用, 可分为数据类型关键字、流程控制关键字、其他类关键字[详见附录 A: ISO/ANSI C90(C99)标准关键字], 具体划分如下。

#### 1. 数据类型关键字

- (1) 基本数据类型: `void`、`char`、`int`、`float`、`double`。
- (2) 类型修饰关键字: `short`、`long`、`signed`、`unsigned`。
- (3) 复杂类型关键字: `struct`、`union`、`enum`、`typedef`、`sizeof`。
- (4) 存储级别关键字: `auto`、`static`、`register`、`extern`、`const`、`volatile`。

#### 2. 流程控制关键字

- (1) 跳转结构: `return`、`continue`、`break`、`goto`。
- (2) 分支结构: `if`、`else`、`switch`、`case`、`default`、`for`、`do`、`while`。

### 3. 其他类关键字

restrict、inline、\_Complex、\_Imaginary、\_Bool。

## 1.4 VC++ 6.0 集成开发环境

### 1.4.1 C 程序一般运行步骤

为了使计算机能按照人的思想进行工作，必须根据解决问题的要求及步骤，编写出相应的程序代码，即一组计算机能识别和执行的指令，每一条指令使计算机执行特定的操作。用高级语言编写的程序称为“源程序”。对于计算机而言，它只能识别和执行由 0 和 1 组成的二进制的指令，而不能识别和执行用高级语言写的指令。为了使计算机能执行高级语言源程序，必须先用一种“编译程序”的软件，把源程序翻译成二进制形式的“目标程序”，然后再将该目标程序与系统的函数库以及其他目标程序连接起来，形成可执行的目标程序。

编写 C 的源程序，然后上机运行，需要经过四个步骤：(1)编辑源程序，生成文本字符文件。(2)对源程序进行编译，形成目标文件。(3)连接，与库函数和其他目标程序连接，形成执行文件。(4)运行目标程序，得到运行结果。在这四个步骤中，如果任何一步有错，均可返到上一步或回到第一步重新操作，其流程如图 1-1 所示。其中实线表示操作流程，虚线表示文件的输入/输出。在上述流程图中，编辑后得到一个源程序文件为 f.c，经过编译得到目标程序文件 f.obj，再将目标程序与系统提供的库函数等连接，得到可执行的目标程序 f.exe，最后得到程序执行结果。

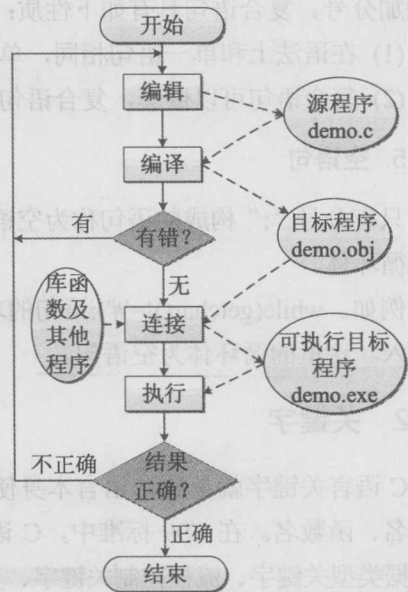


图 1-1 C 程序运行流程图

### 1.4.2 Visual C++集成开发环境

Microsoft Visual C++(简称 Visual C++、MSVC、VC++或 VC)是微软公司开发的一个集成开发环境(IDE)，可提供 C 语言、C++语言等应用程序的开发。它具有程序框架自动生成、灵活方便的类管理、代码编写和界面设计集成交互操作、可开发多种程序等优点，可以将简单的 C 程序开发与高端的面向对象应用程序开发进行有机结合，这些特征明显缩短程式