



# C语言 程序设计与应用

万川梅 郑俏妍 李翠锦 张林 编著

 北京理工大学出版社  
BEIJING INSTITUTE OF TECHNOLOGY PRESS

# C 语言程序设计与应用

万川梅 郑俏妍 李翠锦 张 林 编 著



 **北京理工大学出版社**  
BEIJING INSTITUTE OF TECHNOLOGY PRESS

## 内 容 简 介

本书以满足学生对理论和实践相结合的知识需求为目的,服从创新教育和素质教育的教学理念,以 C 语言为程序设计与开发的基础,全书共有 10 个章节,分别为 C 程序设计初步、语言基础、简单判定性问题、循环结构与应用、模块化设计与应用、相同数据类型集合、指针与应用、构造数据类型、数据永久性存储、一个完整案例的综合设计与实现。

版权专有 侵权必究

---

### 图书在版编目 (CIP) 数据

C 语言程序设计与应用 / 万川梅等编著. —北京:北京理工大学出版社, 2018.6  
ISBN 978 - 7 - 5682 - 5800 - 5

I. ①C… II. ①万… III. ①C 语言 - 程序设计 IV. ①TP312.8

中国版本图书馆 CIP 数据核字 (2018) 第 138149 号

---

---

出版发行 / 北京理工大学出版社有限责任公司

社 址 / 北京市海淀区中关村南大街 5 号

邮 编 / 100081

电 话 / (010) 68914775 (总编室)

(010) 82562903 (教材售后服务热线)

(010) 68948351 (其他图书服务热线)

网 址 / <http://www.bitpress.com.cn>

经 销 / 全国各地新华书店

印 刷 / 北京金品印艺图文设计有限公司

开 本 / 787 毫米 × 1092 毫米 1/16

印 张 / 16.25

字 数 / 384 千字

版 次 / 2018 年 6 月第 1 版 2018 年 6 月第 1 次印刷

定 价 / 63.00 元

责任编辑 / 王玲玲

文案编辑 / 王玲玲

责任校对 / 周瑞红

责任印制 / 李志强

---

图书出现印装质量问题,请拨打售后服务热线,本社负责调换

# 前 言

C 语言程序设计是一门理论与实践紧密结合、以培养学生程序设计能力为目标的课程，它的任务是培养学生应用高级程序设计语言解决问题的能力基本能力。C 语言是学习编程的入门语言，它帮助学生从现有的思维模式转向计算机编程的思维模式。通过该门课程的学习，使学生了解高级程序语言的结构，掌握基本的计算机编程解决问题的思维方式，以及基本的程序语言的语法和程序设计的过程及方法。本书从提出问题、设计算法、选定数据、编程思想、代码实现、测试、调试及结果解析等过程，培养学生逻辑思维和抽象问题的能力，以及能够灵活使用程序设计语言来解决现实中的问题的能力。

## 本书目标

体现计算机基础教学是培养大学生综合素质和创新能力不可或缺的重要环节，是培养复合型创新人才的重要组成部分。计算机基础教学的内涵在快速提升和不断丰富，进一步推进计算机基础教学改革，适应计算机长期以来存在的“计算机知识工具”“计算机就是程序设计”和“计算机基础课程主要讲解软件工具的应用”等片面知识的认识。本书希望能在计算机基础教学过程中体现传授知识、培养能力、提高学生运用计算机思维解决现实的问题。

## 本书特点

本书结合“计算机思维能力培养目标”的教学模式，主要体现以下特点：

- (1) 强化程序语法、语言功能等内容，为程序设计概念做好铺垫；
- (2) 强化数据存储等内容，使学生更加了解计算机处理数据的过程；
- (3) 强化程序设计结构、编程思想，潜移默化地提高学生的计算机思维能力；
- (4) 强化数据的输入与输出，让学习内容更具有趣味性；
- (5) 引入真实案例和项目，让学生学以致用；
- (6) 加大课后习题的量，提供学生辅助教学资源。

## 本书内容

全书分为 10 章，其中第 1 章 C 程序设计初步，第 2 章语言基础，第 3 章简单判定性问题求解，第 4 章循环结构与应用，第 5 章模块化设计与应用，第 6 章相同数据类型集合，第 7 章指针与应用，第 8 章构造数据类型，第 9 章数据永久性存储，第 10 章一个完整案例综合设计与实现。

## 读者对象

无论是程序设计初学人员还是程序开发人员，本书都是一本难得的参考书，本书是高等院校计算机基础类教材，适合各种专业，如多媒体、软件开发、网络工程、通信工程、信息工程、物联网工程、数字媒体技术等专业本科生及教师使用，也适合广大科研和工程技术人员使用。

本书由万川梅整体设计，并完成第3、8、10章的编写，此外，郑俏妍完成第1、2章的编写，李翠锦完成第4、6章的编写，张林完成第5、9章的编写，向首超完成第7章编写。在编写过程中，参考了许多专家和学者的著作和论文，在此谨向他们表示衷心的感谢。

虽然我们希望能够为读者提供最好的教材和教学资源，但由于作者水平和经验有限，疏漏之处难免，不当之处请各位专家和读者赐教。

编 者

# 目 录

|                               |    |
|-------------------------------|----|
| 第 1 章 C 程序设计初步                | 1  |
| 1.1 C 程序的简史                   | 1  |
| 1.2 编程前的准备                    | 3  |
| 1.3 程序开发周期                    | 3  |
| 1.4 C 程序的构成                   | 4  |
| 1.4.1 简单的 C 程序实例              | 4  |
| 1.4.2 阅读 C 程序                 | 5  |
| 1.4.3 C 程序的结构                 | 6  |
| 1.5 C 语言的开发过程和开发环境            | 8  |
| 1.5.1 C 语言的开发过程               | 8  |
| 1.5.2 Turbo C 开发环境及其使用        | 8  |
| 1.5.3 Visual C++ 6.0 开发环境及其使用 | 13 |
| 1.6 本章小结                      | 18 |
| 1.7 习题                        | 18 |
| 第 2 章 语言基础                    | 20 |
| 2.1 注释、大括号、关键字、标识符            | 21 |
| 2.1.1 程序注释                    | 21 |
| 2.1.2 花括号的使用                  | 21 |
| 2.1.3 标识符与关键字                 | 22 |
| 2.2 数据类型、常量、变量                | 23 |
| 2.2.1 基本数据类型与数据的表示            | 23 |
| 2.2.2 常量和变量                   | 26 |
| 2.2.3 基本的输入/输出                | 33 |
| 2.2.4 赋值运算                    | 40 |
| 2.3 不同类型数据之间的转换               | 41 |
| 2.3.1 自动类型转换                  | 41 |
| 2.3.2 强制类型转换                  | 43 |

|                    |           |
|--------------------|-----------|
| 2.4 程序设计与案例实现      | 44        |
| 2.4.1 案例1:鸡兔同笼问题   | 44        |
| 2.4.2 案例2:学生成绩等级评定 | 45        |
| 2.5 本章小结           | 46        |
| 2.6 习题             | 46        |
| <b>第3章 简单判定性问题</b> | <b>48</b> |
| 3.1 判定性问题及条件描述     | 49        |
| 3.1.1 关系型判定条件      | 49        |
| 3.1.2 逻辑型判定条件      | 50        |
| 3.1.3 按位进行逻辑运算     | 53        |
| 3.2 if 条件语句        | 56        |
| 3.2.1 if 语句结构      | 56        |
| 3.2.2 if 语句的嵌套问题   | 63        |
| 3.2.3 条件运算符和条件表达式  | 64        |
| 3.3 switch 条件语句    | 65        |
| 3.4 案例实现           | 66        |
| 3.4.1 案例1:简易计算器    | 66        |
| 3.4.2 案例2:ATM取款机系统 | 69        |
| 3.5 本章小结           | 73        |
| 3.6 习题             | 74        |
| <b>第4章 循环结构与应用</b> | <b>75</b> |
| 4.1 for 循环语句       | 76        |
| 4.1.1 for 循环语句的结构  | 76        |
| 4.1.2 for 循环的应用    | 77        |
| 4.2 while 循环       | 79        |
| 4.3 do...while 循环  | 81        |
| 4.4 循环语句的常见问题      | 82        |
| 4.4.1 双重循环         | 82        |
| 4.4.2 无限循环         | 84        |
| 4.4.3 循环语句的选择      | 84        |
| 4.5 跳出循环语句         | 85        |
| 4.5.1 break 语句     | 85        |
| 4.5.2 continue 语句  | 87        |
| 4.5.3 goto 语句      | 88        |
| 4.6 案例实现           | 88        |
| 4.6.1 案例1:学生成绩管理系统 | 88        |

|                               |            |
|-------------------------------|------------|
| 4.6.2 案例2:简易计算器 .....         | 91         |
| 4.7 编码规范 .....                | 95         |
| 4.7.1 命名规范 .....              | 95         |
| 4.7.2 表达式书写 .....             | 95         |
| 4.7.3 语句排序 .....              | 96         |
| 4.8 本章小结 .....                | 97         |
| 4.9 习题 .....                  | 97         |
| <b>第5章 模块化设计与应用 .....</b>     | <b>99</b>  |
| 5.1 模块化程序设计方法 .....           | 99         |
| 5.1.1 模块化程序设计思想 .....         | 100        |
| 5.1.2 模块规划案例 .....            | 101        |
| 5.2 函数 .....                  | 101        |
| 5.2.1 函数的定义 .....             | 102        |
| 5.2.2 函数的一般调用 .....           | 105        |
| 5.2.3 函数的返回值 .....            | 108        |
| 5.2.4 函数的参数传递与返回值 .....       | 109        |
| 5.2.5 数组作为函数参数 .....          | 112        |
| 5.2.6 函数的嵌套调用 .....           | 116        |
| 5.2.7 函数的递归调用 .....           | 117        |
| 5.3 局部变量与全局变量 .....           | 118        |
| 5.3.1 局部变量 .....              | 119        |
| 5.3.2 全局变量 .....              | 119        |
| 5.3.3 全局变量、静态变量、局部变量的区别 ..... | 120        |
| 5.4 编译预处理 .....               | 121        |
| 5.4.1 宏定义#define .....        | 121        |
| 5.4.2 文件包含#include .....      | 123        |
| 5.4.3 条件编译 .....              | 123        |
| 5.5 本章小结 .....                | 125        |
| 5.6 习题 .....                  | 126        |
| <b>第6章 相同数据类型集合 .....</b>     | <b>129</b> |
| 6.1 数组与数组元素的概念 .....          | 129        |
| 6.2 一维数组 .....                | 130        |
| 6.2.1 一维数组的定义 .....           | 130        |
| 6.2.2 一维数组的初始化 .....          | 131        |
| 6.2.3 一维数组的引用 .....           | 132        |
| 6.2.4 一维数组的应用 .....           | 134        |

|                           |            |
|---------------------------|------------|
| 6.3 二维数组 .....            | 136        |
| 6.3.1 二维数组的定义 .....       | 136        |
| 6.3.2 二维数组的初始化 .....      | 138        |
| 6.3.3 二维数组元素的引用 .....     | 138        |
| 6.3.4 二维数组的应用 .....       | 139        |
| 6.4 使用字符数组处理字符串 .....     | 142        |
| 6.4.1 字符数组初始化 .....       | 142        |
| 6.4.2 字符数组的输入/输出 .....    | 144        |
| 6.4.3 字符串处理函数 .....       | 146        |
| 6.4.4 字符数组的应用 .....       | 150        |
| 6.5 typedef 定义类型 .....    | 152        |
| 6.6 案例实现 .....            | 153        |
| 6.6.1 案例1:课表查询系统 .....    | 153        |
| 6.6.2 案例2:竞赛选手评分系统 .....  | 155        |
| 6.7 本章小结 .....            | 157        |
| 6.8 习题 .....              | 157        |
| <b>第7章 指针与应用</b> .....    | <b>160</b> |
| 7.1 指针概述 .....            | 160        |
| 7.1.1 指针概念 .....          | 160        |
| 7.1.2 指针变量的定义 .....       | 161        |
| 7.1.3 指针的基本运算 .....       | 164        |
| 7.1.4 指针作为函数参数 .....      | 166        |
| 7.2 指针与数组 .....           | 167        |
| 7.2.1 指针与一维数组 .....       | 167        |
| 7.2.2 指针与二维数组 .....       | 169        |
| 7.2.3 指向字符串的指针变量 .....    | 172        |
| 7.2.4 指针数组 .....          | 172        |
| 7.2.5 多级指针 .....          | 174        |
| 7.3 指针与函数 .....           | 175        |
| 7.3.1 指针型函数 .....         | 175        |
| 7.3.2 用函数指针调用函数 .....     | 176        |
| 7.3.3 用指向函数的指针作函数参数 ..... | 177        |
| 7.3.4 带参数的 main 函数 .....  | 179        |
| 7.4 动态分配内存 .....          | 181        |
| 7.4.1 内存的动态分配 .....       | 181        |
| 7.4.2 动态内存分配函数 .....      | 182        |
| 7.4.3 void 指针类型 .....     | 183        |

|                                  |            |
|----------------------------------|------------|
| 7.5 指针综合案例 .....                 | 184        |
| 7.6 本章小结 .....                   | 187        |
| 7.7 习题 .....                     | 187        |
| <b>第8章 构造数据类型</b> .....          | <b>190</b> |
| 8.1 结构体 .....                    | 190        |
| 8.1.1 结构体的定义 .....               | 191        |
| 8.1.2 结构体变量的声明 .....             | 194        |
| 8.1.3 结构体变量的引用 .....             | 195        |
| 8.1.4 结构体变量的初始化 .....            | 196        |
| 8.1.5 结构体数组的应用 .....             | 197        |
| 8.1.6 结构体在函数中的应用 .....           | 199        |
| 8.2 共用体 .....                    | 200        |
| 8.2.1 共用体变量的定义 .....             | 200        |
| 8.2.2 共用体变量的赋值和引用 .....          | 201        |
| 8.3 枚举 .....                     | 202        |
| 8.3.1 枚举类型的定义 .....              | 202        |
| 8.3.2 枚举变量的基本操作 .....            | 203        |
| 8.4 自定义数据类型 .....                | 203        |
| 8.4.1 typedef 自定义数据类型 .....      | 204        |
| 8.4.2 typedef 与#define 的区别 ..... | 206        |
| 8.5 本章小结 .....                   | 207        |
| 8.6 习题 .....                     | 208        |
| <b>第9章 数据永久性存储</b> .....         | <b>211</b> |
| 9.1 文件概述 .....                   | 211        |
| 9.1.1 文件的概念 .....                | 211        |
| 9.1.2 文件指针 .....                 | 213        |
| 9.2 文件的基本操作 .....                | 214        |
| 9.2.1 文件的打开和关闭 .....             | 214        |
| 9.2.2 文件的读写 .....                | 216        |
| 9.2.3 字符串的读写 .....               | 217        |
| 9.2.4 数据块的读写 .....               | 219        |
| 9.2.5 格式的读写 .....                | 220        |
| 9.3 文件的定位 .....                  | 221        |
| 9.3.1 rewind 函数 .....            | 222        |
| 9.3.2 fseek 函数 .....             | 222        |
| 9.3.3 ftell 函数 .....             | 223        |

|                                     |            |
|-------------------------------------|------------|
| 9.4 文件状态检查函数 .....                  | 224        |
| 9.5 习题 .....                        | 224        |
| <b>第 10 章 一个完整案例的综合设计与实现 .....</b>  | <b>226</b> |
| 10.1 问题的提出 .....                    | 226        |
| 10.2 系统功能设计 .....                   | 226        |
| 10.2.1 系统模块设计 .....                 | 226        |
| 10.2.2 数据结构设计 .....                 | 227        |
| 10.3 程序流程图 .....                    | 227        |
| 10.4 源程序清单 .....                    | 228        |
| 10.5 程序测试 .....                     | 236        |
| 10.6 项目文档 .....                     | 237        |
| 10.6.1 需求分析文档 .....                 | 237        |
| 10.6.2 概要设计文档 .....                 | 238        |
| 10.6.3 详细设计文档 .....                 | 239        |
| <b>附录 A 常用字符与 ASCII 代码对照表 .....</b> | <b>242</b> |
| <b>附录 B 运算符的优先级与结合性 .....</b>       | <b>243</b> |
| <b>附录 C C 语言常用的库函数 .....</b>        | <b>245</b> |
| <b>参考文献 .....</b>                   | <b>250</b> |

## 第 1 章

---

# C 程序设计初步



### 学 习 目 标

- 了解 C 程序历史与发展
- 了解程序开发周期
- 掌握 C 程序的构成
- 掌握如何使用 Turbo C 2.0 开发 C 程序
- 掌握如何使用 Visual C++ 6.0 开发 C 程序

### 本章重点

- C 程序的构成
- Turbo C 2.0 工具的使用
- Visual C++ 6.0 工具的使用

### 本章难点

- C 程序的基本语法
- Turbo C 2.0 的安装与使用
- Visual C++ 6.0 的安装与使用

C 语言是国际上广泛流行的计算机高级程序设计语言，其从诞生就注定了会受到世界的关注。它是世界上最受欢迎的语言之一，它具有强大的功能，许多著名的软件都是用 C 程序编写的。学习好 C 语言，可以为以后的程序开发之路打下坚实的基础。在学习 C 语言之前，首先要了解 C 语言的发展历程、C 程序的构成、C 程序的开发环境等知识，才能更为深入地了解这门语言，并且增加对今后学习 C 语言的信心。

本章致力于使读者掌握 C 程序的构成和 Visual C++ 6.0 的集成开发环境中的各个部分的使用方法，并能编写一个简单的 C 语言程序。

## 1.1 C 程序的简史

人与人之间通过各种语言进行沟通，而用户和计算机的交流也需要用计算机和用户都能够理解的语言才可以。这种语言称为“计算机语言”。人们不能直接用自然语言来表达，因

为计算机并不能直接理解。因此，需要用某种特定的计算机语言表达出来，然后输入计算机。这种工作便是“计算机编程”或“程序设计”。用于编写计算机程序的语言称为程序设计语言。C语言就是一种计算机程序设计语言。它既有高级语言的特点，又具有低级汇编语言的特点。它可以作为系统设计语言，编写工作系统应用程序，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。

C语言的发展颇为有趣，它的原型是ALGOL60语言，也称A语言。ALGOL60是一种面向问题的高级语言，不适合编写系统程序。ALGOL60也就是算法语言60。它是程序设计语言由技艺转向科学的重要标志，其具有局部性、动态性、递归性和严谨性特点。

1963年，剑桥大学将ALGOL60语言发展为CPL（Combined Programming Language）语言。CPL语言在ALGOL60的基础上与硬件接近了一些，但规模仍然比较宏大，难以实现。

1967年，剑桥大学的马丁·理查兹（Martin Richards）对CPL语言进行了简化，于是产生了BCPL语言。BCPL语言是计算机软件人员在开发系统软件时，为记录语言而使用的一种结构化程序设计语言，它能够直接处理与机器本身数据类型相近的数据，具有与内容地址对应的指针处理方式。

C语言是在由UNIX的研制者丹尼斯·里奇（Dennis Ritchie）和肯·汤普森（Ken Thompson）于1970年研制的BCPL语言（简称B语言）的基础上发展和完善起来的。19世纪70年代初期，AT&T Bell实验室的程序员丹尼斯·里奇第一次把B语言改为C语言。

最初，C语言运行于AT&T的多用户、多任务的UNIX操作系统上。后来，丹尼斯·里奇用C语言改写了UNIX的编译程序，UNIX操作系统的开发者肯·汤普森又用C语言成功地改写了UNIX，从此开创了编程史上的新篇章，UNIX成为第一个不是用汇编语言编写的主流操作系统。

1983年，美国国家标准委员会（ANSI）对C语言进行了标准化。于1983年颁布了第一个C语言草案（83 ANSI C），后来于1987年又颁布了另一个C语言的标准草案（87 ANSI C），最新的C语言标准C99，于1999年颁布，并在2000年3月被ANSI采用。但是由于未得到主流编译器厂家的支持，C99并未得到广泛使用。

尽管C语言是在大型商业机构和学术界的研究实验室研发的，但是当开发者们为第一台个人计算机提供C编译系统之后，C语言就得以广泛传播，并为大多数程序员所接受。对MS-DOS操作系统来说，系统软件和实用程序都是用C语言编写的。Windows操作系统大部分也是用C语言编写的。

C语言是一种面向过程的语言，同时具有高级语言和汇编语言的特点。C语言可以广泛应用于不同的操作系统，如UNIX、MS-DOS、Microsoft Windows及Linux等。

在C语言的基础上发展起来的有支持多种程序设计风格的C++语言、网络上广泛使用JavaScript及微软的C#语言等。也就是说，学好C语言之后，再学习其他语言就会比较轻松。

**说明**

目前最流行的 C 语言有以下几种：

- (1) Microsoft C, 或称为 MS C;
- (2) Borland Turbo C, 或称为 Turbo C;
- (3) AT&T C。

## 1.2 编程前的准备

C 语言是一种编译性语言, 在编写代码前, 要确定开发环境。C 语言常用的集成开发环境有 Microsoft Visual C++6.0、Microsoft Visual C++.NET、Turbo C 及 Borland C++ Builder 等。

### 1. Microsoft Visual C++6.0

Microsoft Visual C++6.0 不仅是一个 C++ 编译器, 还是一个基于 Windows 操作系统的可视化集成开发环境。

### 2. Microsoft Visual C++.NET 或 Microsoft Visual C++2005

Microsoft Visual Studio.NET 是 Microsoft Visual 6.0 的后续版本, 是一套完整的开发工具集。在 .NET 平台下调用 Framework 的类库, 功能强大, 其中包含 Visual C++ 开发组件。

### 3. Turbo C

Turbo C 是美国 Borland 公司的产品, 目前最常用的版本是 Turbo C 2.0。

### 4. Borland C++ Builder

Borland C++ Builder 是由 Borland 公司继 Delphi 之后推出的一款高性能集成开发环境, 具有可视化的开发环境。

**说明**

不同版本的 C 语言编译系统, 所实现的语言功能和语法规则又略有差别。本书主要以 Microsoft Visual C++6.0 为 C 语言开发环境, 因为它功能完善, 操作简便, 界面友好, 适合初学者开发使用。

## 1.3 程序开发周期

程序开发周期指创建计算机程序的过程, 与一般问题求解过程 (理解问题、制订计划、实施计划、结果检验和回顾) 非常相似。当使用计算机程序来解决问题时, 这种过程呈现出以下形态:

### 1. 分析问题

确定已经得到了哪些信息, 需要得到哪些结果, 哪些信息是获取结果所必需的。通俗地说, 就是如何从已知数据推导出期望的结果。

### 2. 设计程序

这是程序开发过程的核心, 这个过程根据问题规模的大小, 可以花费某人几个小时的时间。

间，也可能需要一个程序员团队花费几个月时间来完成。

### 3. 程序编码

根据第2步所制订的设计方案，使用某种特定的程序设计语言编写程序的源码（source-code）。这一步的结果就是程序。

### 4. 程序测试

运行程序，检验其是否解决了给定的问题。

这种分析、设计、编码和测试构成了程序开发周期（program development cycle）的核心。这里使用周期（cycle）一词，是因为使用一般问题求解过程方法时，一旦后续步骤发现问题，经常需要返回到以前的步骤，并频繁往复。

## 1.4 C 程序的构成

### 1.4.1 简单的 C 程序实例

学习程序设计过程中，学习者将会遇到的最有意义的一件事，是尽管有许多不同的程序设计语言，但无论使用哪种，程序设计的基本概念是相通的。在学习了程序设计的概念和逻辑，以及掌握一门程序设计语言之后，再学习新的程序设计语言，相对会容易得多。但首先必须掌握程序设计的基本构成。

C 语言是学好其他编程语言的基础，下面通过一个简单的程序来看一看 C 程序是什么样子的。

【例 1-1】一个简单的 C 程序。

#### 程序清单

```
/* 一个简单的 C 语言程序 */
#include <stdio.h>
int main()
{
    printf("Hello,welcome to C world! \n"); /* 输出要显示的字符串 */
    return 0;                               /* 程序返回 0 */
}
```

运行结果如图 1-1 所示。



```
Hello,welcome to C world!
Press any key to continue
```

图 1-1 程序运行结果

代码解析：

在例 1-1 中，实现的功能只是显示一条信息：“Hello, welcome to C world!”，通过这个程序可以知道 C 程序的构成。虽然这个简单的小程序只有 7 行，却充分说明了 C 程序是由什么位置开始、在什么位置结束的。

## 1.4.2 阅读 C 程序

### 1. #include 指令

#include <stdio.h> 这个语句的功能是进行有关的预处理操作。其中字符“#”是预处理标志，用来对文本进行预处理操作，“include”是预处理指令，称为文件包含命令，它后面跟着一对尖括号，表示头文件在尖括号内读入，称之为头部文件或首文件。

### 2. 空行

C 语言是一个较灵活的语言，因此格式并不是固定不变的。也就是说，空格、空行、跳格并不会影响程序。合理、恰当地使用这些空格、空行，可以使编写出来的程序更加规范，对日后的阅读和整理有着重要的作用。



#### 说明

不是所有的空格都没有用，例如，在两个关键字之间用空格隔开（int main()），这种情况下如果将空格去掉，程序就不能通过编译。

### 3. main() 函数声明

int main() 的意思是声明 main() 函数为一个返回值为整型的函数。其中的 int 称为关键字，这个关键字代表的类型是整型。关于数据类型的内容将会在本书的第 2 章中进行讲解，在函数中这一部分则称为函数头部分。



#### 强调

在每一个程序中都会有一个 main 函数，main 函数是一个程序的入口部分。也就是说，程序都是从 main 函数头开始执行的，然后进入函数中，执行 main 函数中的内容。

### 4. 函数体

一个函数分为两部分：一部分是函数头，一部分是函数体。

```
{  
    printf("Hello,welcome to C world! \n"); /* 输出要显示的字符串 */  
    return 0;                               /* 程序返回 0 */  
}
```

以上的程序代码由两个大括号括起来就构成了函数体，函数体可以称为函数的语句块。在函数体中，“printf("Hello,welcome to C world!\n");”和“return 0;”这两条语句就是函数体重要的执行内容。

### 5. 执行语句

在函数体中，执行语句就是函数体中要执行的内容。代码“printf("Hello,welcome to C world!\n");”中的“printf”是产生格式化输出的函数，它的作用是向控制台输出文字或符号。括号中的内容称为函数的参数，在括号内可以看到输出的字符串“Hello, welcome to C world!”。执行语句中有“\n”，它叫做转义符，含义是把光标移动到下一行的行首，也就是回车换行。因为无法直接通过键盘输入换行的指令，所以需要使用转义符。

## 6. return 语句

“return 0;”这行语句使 main 函数终止运行，并向操作系统返回一个“0”整型常量。前面介绍 main 函数时，提过返回一个整型返回值，此时 0 就是要返回的整型。在此处可以将 return 理解成 main 函数的结束标志。

## 7. 代码的注释

```
printf("Hello,welcome to C world! \n"); /* 输出要显示的字符串 */
return 0;                               /* 程序返回 0 */
```

在以上两行代码后面，看到关于代码的文字描述。对于代码的解释描述称为代码的注释。代码注释的作用，就是对代码进行解释说明，以方便用户理解程序代码的含义和设计思想。它的语法格式为：

```
/* 注释内容 */
```



### 说明

虽然没有严格规定程序中一定要写注释，但是为程序代码写注释是一个良好的习惯，这将为以后查看代码带来方便，并且如果程序交给别人看，他人便可以快速地掌握程序思想及代码的作用。

## 1.4.3 C 程序的结构

### 1. C 程序的结构

一个 C 程序可以由若干个源程序文件组成，每个源文件可以由若干个函数和预处理命令，以及全局变量声明部分组成，每一个函数由函数首部和函数体组成。C 程序的结构如图 1-2 所示。

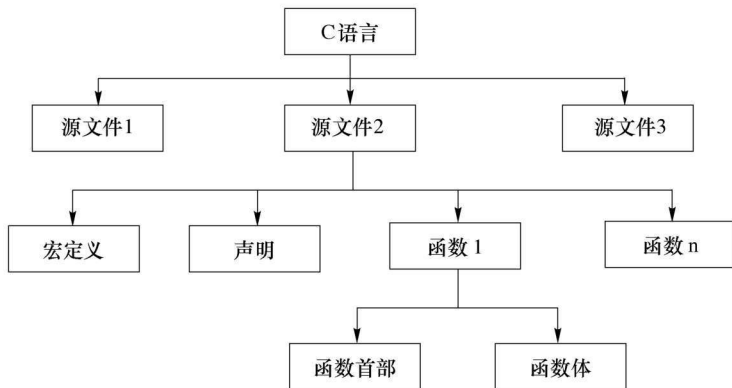


图 1-2 C 程序的结构

C 程序提供了丰富的函数集，称为标准函数库。标准函数库包括 15 个头文件，借助这些函数可以完成不同的功能。在例 1-1 中，实现的是显示“Hello, welcome to C world!”。如果要显示文字或一串字符，只需要调用 printf 函数即可。在编程过程中，要使用 printf()