



工业和信息化普通高等教育“十二五”规划教材立项项目

21 世纪高等学校计算机规划教材

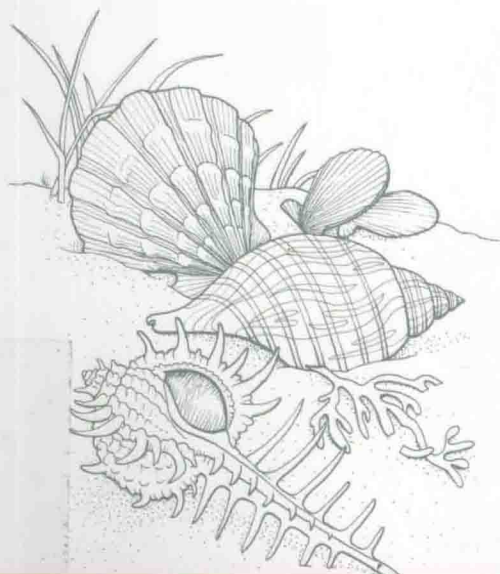
21st Century University Planned Textbooks of Computer Science

本教材获“陕西省高等教育教学改革研究重点项目(13BZ54)”资助

# 大学C语言 设计教程

The C Programming Language

张峰 王伯槐 刘红霞 编著



高校系列



人民邮电出版社  
POSTS & TELECOM PRESS



工业和信息化普通高等教育“十二五”规划教材立项项目

21 世纪高等学校计算机规划教材

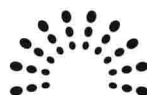
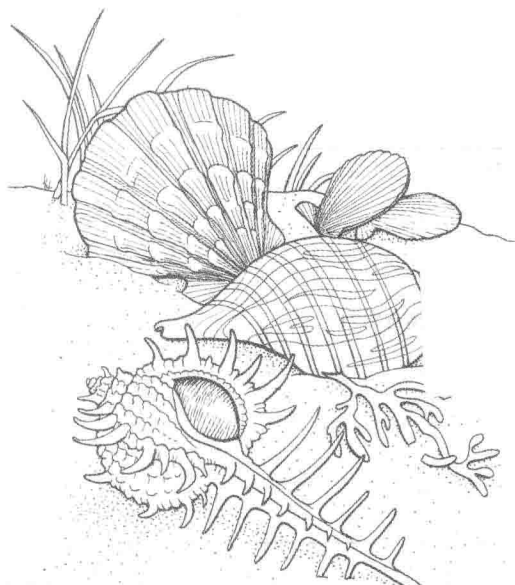
21st Century University Planned Textbooks of Computer Science

本教材获“陕西省高等教育教学改革研究重点项目(13BZ54)”资助

# 大学C语言 设计教程

The C Programming Language

张峰 王伯槐 刘红霞 编著



高校系列

人民邮电出版社

北京

## 图书在版编目 (C I P) 数据

大学C语言设计教程 / 张峰, 王伯槐, 刘红霞编著

— 北京: 人民邮电出版社, 2014. 7

21世纪高等学校计算机规划教材

ISBN 978-7-115-35330-6

I. ①大… II. ①张… ②王… ③刘… 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(2014)第118229号

## 内 容 提 要

本书从初学者学习 C 语言程序设计的角度出发, 根据作者多年的程序设计经验, 按照科学的学习规律, 由浅入深、循序渐进地向读者讲述了 C 语言开发环境的搭建、C 语言基础语法知识, 主要内容包括基本数据类型和运算符、控制结构、函数、数组、指针、结构与共用、文件等。每章都配有大量的程序设计案例、问题集锦与解答以及多种类型的练习题, 有助于读者复习、巩固所学知识, 培养读者的实际编程能力。

本书内容丰富、逻辑性强、文字流畅、通俗易懂, 可以作为各类高等院校、高职院校计算机专业及理工科非计算机专业学生学习“计算机程序设计”课程的教材, 也可作为广大计算机爱好者学习 C 程序设计语言的参考书。

---

◆ 编 著 张 峰 王伯槐 刘红霞

责任编辑 邹文波

执行编辑 吴 婷

责任印制 彭志环 焦志炜

◆ 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>

北京艺辉印刷有限公司印刷

◆ 开本: 787×1092 1/16

印张: 15.5

2014 年 7 月第 1 版

字数: 403 千字

2014 年 7 月北京第 1 次印刷

---

定价: 39.00 元

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

反盗版热线: (010) 81055315

近年来,功能强大的C语言成为最流行的程序设计语言,学习C语言的人不计其数。然而,在笔者工作和教学过程中,发现大多数人学过C语言后,并没有发挥C语言和计算机的强大功能,他们通常只使用C语言非常简单的功能,为什么有些人学过C语言后,与简直没学过一样呢?其中一个原因就是,C语言程序设计具有较强的实践性,但目前大部分C语言教程都是注重基本概念知识的介绍,忽视了对学生编程激情的激发,书中没有上机指导、问题解答、自我检测与项目实践,使得学生的实践太少。在C语言的教学过程中,预期教学目标与最终效果有着明显的差距,教师感觉难教,学生感觉难学、难理解,学会了也不会编程。在编程领域里,没有所谓的“武林秘籍”,再好的教材也一定要配合大量上机实验指导,并在课后做大量习题进行自我检测与项目实践,所以本书配备了另一本上机实践类教材作为进一步提高学生实践编程能力的“抓手”。

C语言不仅为计算机专业人员所使用和喜爱,近年来,许多计算机应用人员也开始喜爱和使用C语言。实践证明,它是一种很好的程序设计语言。随着经济全球化、社会信息化时代的到来,当代大学生不但要会使用计算机获取专业领域知识,还要会使用计算机进行编程,解决专业领域中的具体问题。C语言是当前流行的操作系统Windows、Linux、UNIX上的一种系统开发语言,同时又是计算各专业问题的有效语言,因此,C语言已成为各高校计算机专业和非计算机专业学生必学的一门语言。在非计算机专业等级考试中,C语言已替代了Pascal和Fortran语言,因此,学好C语言的重要性已不言而喻。

从本质上看,前面提到的问题都可以通过上机实践、案例开发来加强。因而,学习C语言最好的方法是结合计算机知识在实践中不断学习。这正是C语言程序设计的秘笈所在,本书正是基于这种考虑而编写的。

## 本书的组织结构

全书共8章。第1~4章详细介绍了C语言的发展过程,C语言开发环境的搭建,C程序的结构、数据类型、运算符、表达式、程序控制结构和函数等。第5~6章详细介绍了数组和指针的应用,第7~8章详细介绍了结构体、共用体和文件等。

本书编写分工如下:张峰编写第1章、第2章、第3章、第4章,刘红霞编写第5章、第6章,王伯槐编写第7章、第8章,全书由张峰统稿。张永恒、艾晓燕教授在百忙中抽出时间对本书进行了审阅,在此表示感谢。

## 本书的特点和优势

本书以通俗的语言深入浅出地带领读者全面学习C语言。清晰的结构、丰富的图例、简单易懂却又一针见血点破本质的示例代码使得读者能轻松迈进C语言的大门。本书考虑了初学者的认知特点,在讲解的过程中从基本的环境配置讲起,告诉你如何搭建开发环境。细致讲解每一个例子,让你学会程序开发的基本步骤。本书强调动手实践,每个知识点都有相应的代码段,让你在操作中领会编程的真谛。注释极为详尽,方便读者阅读代码,理解其中的含义。

本书配备全程上机实验指导教材,案例驱动。与同类图书相比,本书在内容设计上注重结合最新编程技术进行讲解,并且内容涉及C语言的各个应用方面,使自学者能够全面掌握C编程中

涉及的各种技术。本书并不是单纯从知识角度来讲解 C 语言，而是从实践和解决开发问题的角度来介绍 C 语言，所以本书中包含了大量上机实验指导、问题集锦与解答、自我检测与实践，这些案例既能让读者巩固每章的知识，又能让读者学以致用。

本书的作者都是一线的 C 语言应用软件的教学和开发人员，注重实践，对 C 语言各种应用技术有深刻的理解。理论和实际开发的着眼点完全不同，前者注重理论和技术的高度，后者注重符合实际的应用需求。在本书中，一切内容的设计、整理和编写都完全针对实际开发人员的需求。

本书为读者提供了大量的图片和代码注释，并且尽量做到通俗易懂，循序渐进，从开发者的角度去理解和把握各种概念和技术。精心设计的上机实验内容引导读者实践和提高，大量常见问题解答让读者少走弯路，同时提供了大量的习题方便读者练习和提高。

### 本书读者对象

- C 语言初学者、C 语言程序员、想自学 C 语言编程的在校学生
- 高等院校、高职院校计算机专业及理工科非计算机专业学生
- 大中专院校或培训机构需要 C 语言教材的学生

### 注意事项

本书在编写过程中得到了榆林学院教材出版基金的资助以及信息工程学院领导的大力协助，在此表示衷心的感谢。参加本书编写工作的还有张永恒、艾晓燕、刘黎、李赵兴、安强强、卢磊、杨飞、王海荣、董建刚、吴疆、康亚明、曹课兴、吴敏宁、屈晓渊、张惠、张烨、李瑞华、张珏、刘峰、刘汉烨、郭红波、张雅琼、王小平、郭小平、杜博、蔺跟荣、尚佩妮、李少恒、杜承宁、麻少华、边瑞锋、王乃生、周亚婷、董培英、马二平、冯妮娅、杨深刚、张治友、张记、宋向阳、白兆华等。

由于时间仓促，作者水平有限，书中的错误在所难免，真诚希望广大读者提出宝贵意见。如果读者在阅读本书过程中遇到疑难问题，可以把问题发到邮箱 [tfnew21@sina.com](mailto:tfnew21@sina.com) 或博客 <http://blog.csdn.net/tfnew21>，共同交流探讨。

本教材获得“陕西省高等教育教学改革研究重点项目（13BZ54）”资助。

编 者

2014 年 1 月

# 目 录

## 第1章 C语言概述.....1

|                                   |  |
|-----------------------------------|--|
| 1.1 C语言的发展过程.....1                |  |
| 1.1.1 什么是语言.....1                 |  |
| 1.1.2 什么是机器语言.....1               |  |
| 1.1.3 什么是汇编语言.....2               |  |
| 1.1.4 什么是高级语言.....2               |  |
| 1.1.5 C语言的发展过程.....2              |  |
| 1.1.6 理工科学生为什么要学C语言.....3         |  |
| 1.2 C语言的特点.....3                  |  |
| 1.3 C程序的结构及书写格式.....4             |  |
| 1.3.1 C程序的结构.....4                |  |
| 1.3.2 C程序的书写格式.....6              |  |
| 1.4 C语言开发环境的搭建.....6              |  |
| 1.4.1 C程序可执行文件的生成过程.....6         |  |
| 1.4.2 Visual C++ 6.0 上机操作过程.....7 |  |
| 1.4.3 程序调试.....12                 |  |
| 习 题.....15                        |  |

## 第2章 数据类型、运算符和表达式.....17

|                      |  |
|----------------------|--|
| 2.1 基本数据类型.....17    |  |
| 2.1.1 整型数据.....17    |  |
| 2.1.2 整数的正负.....17   |  |
| 2.1.3 实数类型.....18    |  |
| 2.1.4 字符类型.....18    |  |
| 2.2 常 量.....19       |  |
| 2.2.1 整型常量.....19    |  |
| 2.2.2 实型常量.....19    |  |
| 2.2.3 字符常量.....20    |  |
| 2.2.4 字符串常量.....21   |  |
| 2.3 变 量.....22       |  |
| 2.3.1 变量的定义.....22   |  |
| 2.3.2 标识符和关键字.....23 |  |
| 2.4 运算符和表达式.....24   |  |

|                             |  |
|-----------------------------|--|
| 2.4.1 操作数、运算符和表达式.....25    |  |
| 2.4.2 运算符优先级.....25         |  |
| 2.4.3 赋值运算.....26           |  |
| 2.4.4 算术运算.....26           |  |
| 2.4.5 自增和自减.....27          |  |
| 2.4.6 关系运算.....28           |  |
| 2.4.7 逻辑运算符.....29          |  |
| 2.4.8 位运算.....30            |  |
| 2.4.9 ?运算.....31            |  |
| 2.4.10 复合赋值运算.....32        |  |
| 2.4.11 逗号运算.....32          |  |
| 2.4.12 类型转换与类型转换规则.....33   |  |
| 2.5 输入/输出函数.....35          |  |
| 2.5.1 向屏幕输出函数 printf.....35 |  |
| 2.5.2 格式化输入函数 scanf.....37  |  |
| 2.5.3 字符输入/输出函数.....39      |  |
| 2.6 本程序案例.....40            |  |
| 2.7 问题集锦与解答.....41          |  |
| 习 题.....43                  |  |

## 第3章 程序控制结构.....46

|                            |  |
|----------------------------|--|
| 3.1 程序的灵魂——算法.....46       |  |
| 3.1.1 算法描述.....46          |  |
| 3.1.2 用流程图表示算法.....48      |  |
| 3.1.3 算法的结构化描述.....50      |  |
| 3.1.4 用N-S图描述算法.....51     |  |
| 3.1.5 用PAD图描述算法.....52     |  |
| 3.1.6 复合语句.....52          |  |
| 3.2 条件控制语句.....53          |  |
| 3.2.1 if语句.....53          |  |
| 3.2.2 if嵌套语句.....58        |  |
| 3.2.3 if语句综合实例.....58      |  |
| 3.3 开关语句——switch结构.....60  |  |
| 3.3.1 switch语句的一般形式.....60 |  |
| 3.3.2 switch语句的执行过程.....60 |  |

|                                    |           |                                |            |
|------------------------------------|-----------|--------------------------------|------------|
| 3.3.3 switch 语句应用案例 .....          | 61        | 4.5.3 条件编译 .....               | 106        |
| 3.3.4 default 语句 .....             | 63        | 4.6 问题集锦与解答 .....              | 108        |
| 3.4 循环结构 .....                     | 64        | 习 题 .....                      | 109        |
| 3.4.1 for 语句 .....                 | 64        | <b>第 5 章 数 组 .....</b>         | <b>114</b> |
| 3.4.2 for 语句程序案例 .....             | 66        | 5.1 一维数组 .....                 | 114        |
| 3.4.3 while 语句 .....               | 68        | 5.1.1 一维数组的定义 .....            | 114        |
| 3.4.4 while 语句程序案例 .....           | 68        | 5.1.2 一维数组的引用 .....            | 115        |
| 3.4.5 do~while 语句 .....            | 70        | 5.1.3 一维数组的初始化 .....           | 117        |
| 3.4.6 do~while 语句程序案例 .....        | 71        | 5.1.4 一维数组程序举例 .....           | 117        |
| 3.4.7 三种循环语句的比较 .....              | 72        | 5.2 二维数组 .....                 | 120        |
| 3.4.8 循环嵌套 .....                   | 72        | 5.2.1 二维数组的定义 .....            | 120        |
| 3.5 break、continue 及 goto 语句 ..... | 74        | 5.2.2 二维数组的引用 .....            | 121        |
| 3.5.1 break 语句 .....               | 74        | 5.2.3 二维数组的初始化 .....           | 121        |
| 3.5.2 continue 语句 .....            | 75        | 5.2.4 二维数组程序举例 .....           | 122        |
| 3.5.3 随心所欲 goto 语句 .....           | 76        | 5.3 字符数组 .....                 | 123        |
| 3.6 问题集锦与解答 .....                  | 77        | 5.3.1 字符数组的定义 .....            | 123        |
| 习 题 .....                          | 78        | 5.3.2 字符数组的初始化 .....           | 124        |
| <b>第 4 章 函 数 .....</b>             | <b>85</b> | 5.3.3 字符数组的输入/输出 .....         | 125        |
| 4.1 函数概述 .....                     | 85        | 5.3.4 字符串常用处理函数 .....          | 126        |
| 4.2 函数说明与返回值 .....                 | 86        | 5.4 函数与数组 .....                | 128        |
| 4.2.1 函数定义的一般形式 .....              | 86        | 5.4.1 数组元素作为函数的实参 .....        | 128        |
| 4.2.2 函数的返回类型说明 .....              | 86        | 5.4.2 数组名作函数参数 .....           | 128        |
| 4.2.3 形式参数与实际参数 .....              | 87        | 5.5 综合实例 .....                 | 130        |
| 4.2.4 函数参数数据传递 .....               | 88        | 习 题 .....                      | 132        |
| 4.2.5 函数原型声明 .....                 | 89        | <b>第 6 章 指 针 .....</b>         | <b>137</b> |
| 4.3 递归函数 .....                     | 91        | 6.1 指针与变量 .....                | 137        |
| 4.3.1 递归流程 .....                   | 91        | 6.1.1 指针的基本概念 .....            | 137        |
| 4.3.2 递归程序应用案例 .....               | 92        | 6.1.2 指针变量的定义与引用 .....         | 138        |
| 4.4 函数的作用域规则 .....                 | 95        | 6.1.3 指针的运算 .....              | 140        |
| 4.4.1 变量的存储类别 .....                | 95        | 6.2 指针与数组 .....                | 141        |
| 4.4.2 生存期 .....                    | 95        | 6.2.1 指针数组 .....               | 141        |
| 4.4.3 作用域与可见域 .....                | 96        | 6.2.2 指向数组元素的指针 .....          | 144        |
| 4.4.4 自动型 (auto) .....             | 96        | 6.2.3 通过指针引用数组元素 .....         | 144        |
| 4.4.5 寄存器型 (register) .....        | 98        | 6.2.4 指向多维数组的指针和指针<br>变量 ..... | 151        |
| 4.4.6 外部型 (extern) .....           | 98        | 6.3 指针与字符串 .....               | 156        |
| 4.4.7 静态型 (static) .....           | 100       | 6.3.1 使指针变量指向“字符串”的方法<br>..... | 156        |
| 4.5 编译预处理 .....                    | 102       |                                |            |
| 4.5.1 文件包含 .....                   | 102       |                                |            |
| 4.5.2 宏定义 .....                    | 103       |                                |            |

|                             |            |                                              |            |
|-----------------------------|------------|----------------------------------------------|------------|
| 6.3.2 指向字符串常量的指针变量的使用.....  | 157        | 8.1.2 文件的分类.....                             | 209        |
| 6.3.3 使用字符指针变量与字符数组的区别..... | 158        | 8.1.3 文件类型指针.....                            | 210        |
| 6.4 指针与函数.....              | 159        | 8.2 文件的打开与关闭.....                            | 211        |
| 6.4.1 指针变量作函数参数.....        | 159        | 8.2.1 文件的打开.....                             | 211        |
| 6.4.2 返回指针值的函数.....         | 163        | 8.2.2 文件的关闭.....                             | 212        |
| 6.4.3 函数指针.....             | 165        | 8.3 文件的读写.....                               | 213        |
| 6.5 指向指针的指针.....            | 167        | 8.3.1 字符读写函数 fgetc( )和 fputc( )<br>.....     | 213        |
| 6.6 问题集锦与解答.....            | 169        | 8.3.2 字符串读写函数 fgets( )和 fputs( )<br>.....    | 215        |
| 习 题.....                    | 169        | 8.3.3 格式化读写函数 fscanf( )和 fprintf( )<br>..... | 217        |
| <b>第 7 章 自定义数据类型</b> .....  | <b>175</b> | 8.3.4 数据块读写函数 fread( )和 fwrite( )<br>.....   | 218        |
| 7.1 结构体的定义和引用.....          | 175        | 8.4 文件的定位.....                               | 221        |
| 7.1.1 结构体的定义、初始化与使用.....    | 175        | 8.5 文件检测.....                                | 223        |
| 7.1.2 结构体类型数组.....          | 179        | 习 题.....                                     | 224        |
| 7.1.3 结构体类型数据的指针.....       | 181        | <b>附录 1 常用字符与 ASCII 值对照表</b><br>.....        | <b>228</b> |
| 7.1.4 结构体的嵌套.....           | 185        | <b>附录 2 C 语言保留字一览表</b> .....                 | <b>229</b> |
| 7.1.5 用指针处理链表.....          | 186        | <b>附录 3 运算符的优先级及其结合性</b><br>.....            | <b>230</b> |
| 7.2 共用体类型.....              | 193        | <b>附录 4 常用 C 库函数</b> .....                   | <b>231</b> |
| 7.2.1 共用体类型的定义.....         | 193        | <b>参考文献</b> .....                            | <b>238</b> |
| 7.2.2 共用体类型变量的引用.....       | 195        |                                              |            |
| 7.3 枚举类型.....               | 196        |                                              |            |
| 7.4 自定义类型.....              | 198        |                                              |            |
| 习 题.....                    | 200        |                                              |            |
| <b>第 8 章 文 件</b> .....      | <b>209</b> |                                              |            |
| 8.1 文件概述.....               | 209        |                                              |            |
| 8.1.1 什么是文件.....            | 209        |                                              |            |

# 第 1 章

## C 语言概述

### 1.1 C 语言的发展过程

自从 1946 年世界上第一台电子计算机 ENIAC 问世以来, 计算机科学及其应用的发展十分迅猛, 计算机被广泛地应用于人类生产、生活的各个领域, 推动了社会的进步与发展。特别是随着互联网日益深入千家万户, 传统的信息收集、传输及交换方式正被革命性地改变, 我们已经难以摆脱对计算机的依赖, 计算机已将人类带入了一个新的时代——信息时代。

计算机语言是人与计算机进行交互的工具, 是用户进行计算机软件开发、编写计算机程序的工具。计算机程序设计语言的发展, 经历了从机器语言、汇编语言到高级语言的 3 个阶段。

#### 1.1.1 什么是语言

好比人类的语言, 如汉语、英语、日语等, 是用来交流沟通的。有一方在说, 有另一方在听, 必须有两方参与, 这是语言最重要的功能。语言就是用来表达意思传递信息。说的一方传递信息, 听的一方接受信息。说的一方下达指令, 听的一方遵从命令做事情。没有语言, 双方就无法交流沟通。但语言有独特的语法规则。交流双方都了解明白并遵守这些规则, 一个只会说汉语的中国人, 和一个只会说英语的英国人, 如果戴上面具, 只通过嘴巴发出声音互相交流, 结果一定是鸡同鸭讲, 信息完全传递不出去。为什么? 因为互相不知道对方的语法规则, 当然听不懂了。为什么要戴面具? 为什么只能通过嘴巴? 因为人类的一些面部表情, 身体动作, 这些却是相通的, 不通过声音, 而通过肢体语言, 也能多少表达出一些意思。

#### 1.1.2 什么是机器语言

电子计算机所使用的是由“0”和“1”组成的二进制数, 二进制是计算机的语言的基础。计算机是一个忠实的仆人, 时刻等候着主人的命令。如何才能使计算机听话呢? 当然是用计算机听得懂的语言去命令它了。计算机的大脑或者说心脏就是 CPU, 它控制着整个计算机的运作。每种 CPU 都有自己的指令系统。这个指令系统就是该 CPU 的机器语言。计算机指令采用二进制(0、1)表示, 也就是说, 计算机能识别的指令代码只能是二进制形式。这种语言就是机器语言, 或称为低级语言。如 PC 中两个数进行加法的指令为: 0100110 11100001。由于机器语言采用的是二进制序列表示指令, 十分难记; 另外, 采用机器语言编写的计算机程序具有不可移植性, 即在一台计算机上执行的程序, 要想在另一台计算机上执行, 必须另编程序, 造成了重复工作。但由

于使用的是针对特定型号计算机的语言，故而运算效率是所有语言中最高的。机器语言是第一代计算机语言。

### 1.1.3 什么是汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进，计算机科学家将机器语言的每一条指令采用助记符表示，即机器语言的符号法，称为汇编语言。如上面 PC 的加法指令用符号表示为：ADD AH, BL。这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了，这种程序设计语言是第二代计算机语言。采用汇编语言编写的计算机程序必须经过翻译过程，变为机器语言后，计算机才能识别运行，这种翻译程序称为汇编程序，对应的过程称为汇编过程。用汇编语言编写的计算机程序仍与体系结构有关，具有不可移植性。但采用机器语言和汇编语言编写的计算机程序具有运算效率高的特点。汇编语言同样十分依赖于机器硬件，移植性不好，但效率仍十分高，针对计算机特定硬件而编制的汇编语言程序，能准确发挥计算机硬件的功能和特长，程序精练而质量高，所以至今仍是一种常用而强有力的软件开发工具。



程序的可移植性指的是程序不加修改或只需少许修改即可在不同的机器上运行，实现程序可移植性的一种方法就是采用与硬件无关的程序设计语言。C 语言是目前最常用的与硬件无关的程序设计语言之一。

### 1.1.4 什么是高级语言

从最初与计算机交流的痛苦经历中，人们意识到，应该设计一种这样的语言，这种语言接近于数学语言或人的自然语言，同时又不依赖于计算机硬件，编出的程序能在所有机器上通用。经过努力，1954 年，第一个完全脱离机器硬件的高级语言——Fortran 问世，高级语言是一种更接近于自然的数学形式语言，如两个数的加法可写为  $z = x + y$ 。采用高级语言编写的计算机程序与机器类型无关，具有可移植性、易学易记等特点，采用高级语言编写的程序称为源程序。但高级语言编写的计算机程序，计算机不能直接识别，必须经过翻译过程将其译为机器语言后，计算机才能识别运算。其翻译过程分为两种，一种是边翻译，边运行，翻译一句，执行一句，这种过程称为解释过程，对应的语言称为解释语言。每次执行程序时，都必须经过相同的翻译过程，如早期的 BASIC 语言和 FoxBase 等。采用解释语言编写的计算机程序不能离开其解释环境。另一种是编译语言，它是将整个源程序全部翻译成机器语言指令后，计算机才能运行，这样的翻译过程称为编译过程，对应的翻译程序称为编译程序。源程序经编译后生成的机器语言程序称为目标程序，计算机不能直接运行目标程序，还必须经过连接过程，才能变为可执行文件，对应连接过程的程序称为连接程序，这样生成的可执行文件具有永逸性，即经过一次编译、连接后，生成的可执行文件以后不需要再进行编译连接过程，可以脱离语言环境，在同类型的计算机上仍可运行，如 Fortran 语言、Pascal 语言、Lisp 语言、Ada 语言和 C 语言等。

### 1.1.5 C 语言的发展过程

C 语言是在 20 世纪 70 年代初问世的。1978 年由美国电报电话公司(AT&T)贝尔实验室正式发表了 C 语言。20 世纪 70 年代初，编写计算机系统软件时使用了一种符号法的自展组合语言 BCPL，BCPL 进一步发展为一种系统软件描述语言 B 语言。20 世纪 80 年代初，美国贝尔实验室软件开发人员 Dennis M. Ritchie 将 B 语言发展成为 C 语言。C 语言继承了 B 语言的特点，成为编

写系统软件的重要工具语言。最初 C 语言有各种不同的标准,1983 年美国标准协会制定了 C 语言标准草案,称为 83ANSI C,1989 年正式修订后成为大家公认的标准,称为 89ANSI C。该标准中规定了 C 语言的关键字为 28 个,1999 年在原 89ANSI C 基础上增加了新的面向对象特性,并增加了 4 个关键字,该标准即为现在的 99ANSI C。

不同的编译器开发商在遵照 C 语言标准的基础上,对标准 C 新增了一些特性,如增加了图形图像处理能力,或在标准 C 的基础上,增加了特定的库函数,编译器的实现方式不同,这样市面上出现了 Borland 公司的 Turbo C,Microsoft 公司的 Microsoft C 等不同的编译器,都可实现对 C 语言程序的编辑、编译、连接和运行。Microsoft C 增加面向对象特性后,发展为 Microsoft C++ 和可视化编程的 Microsoft Visual C++。

C 的名字从何而来?



C 源自 Ken Thompson 的实验性语言 B,而 B 由 Martin Richards 的 BCPL(Basic Combined Programming Language)得到灵感,而 BCPL 是 CPL(Combined Programming Language 或 Cambridge Programming Language)的简化版。有一段时间,人们猜测 C 的后继者会命名为 P(BCPL 的第三个字母)而不是 D,当然,如今最先进的后裔语言是 C++。

### 1.1.6 理工科学生为什么要学 C 语言

为什么要学习 C 语言?这是每个学生应该问的问题。对于理工科的大学生而言,掌握一门高级语言及其基本的编程技能是必需的。大学学习,除了掌握本专业系统的基础知识外,科学精神的培养,思维方法的锻炼,严谨踏实的科研作风养成以及分析问题、解决问题的能力训练,都是日后工作的基础。学习计算机语言,正是一种十分有益的训练方式,而语言本身又是与计算机进行交互的有力的工具。一台计算机是由硬件系统和软件系统两大部分构成的,硬件是物质基础,而软件可以说是计算机的灵魂,没有软件,计算机是一台“裸机”,是什么也不能干的,有了软件,才能灵动起来,成为一台真正的“电脑”。所有的软件,都是用计算机语言编写的。而 C 语言的应用极其广泛,从网站后台到底层操作系统,从多媒体应用到大型网络游戏,均可使用 C 语言来开发。比如嵌入式系统、底层驱动、网络通信和机器人控制系统等。

当掌握了 C 语言后,再去学习其他面向过程的语言,最多一个月就能学会。因为万变不离其宗,只是语法上有些许更改,而思想却没有更改。

#### 扩展阅读:

TIOBE 近日公布了 2013 年 10 月份的编程语言排行榜,C 语言战胜上月排名第一的 Java 语言,夺回了第一位的宝座。C 和 Java 这两门语言所占的份额相差不大,也会经常来回波动,相信随着 Android 平台的普及,Java 份额的下降也不会维持多久,不过未来的几个月内 C 还是会保持领先的位置。目前 C 语言主要用于游戏的图像引擎、数控、嵌入式、汽车的嵌入式系统、电梯、冰箱、洗衣机等领域。

## 1.2 C 语言的特点

C 语言简洁、紧凑,使用方便、灵活,与其他高级语言或中级语言相比,具有如下特点。

(1) C 语言有丰富的运算符。C 语言除提供了其他高级语言提供的算术运算、关系运算、逻辑运算、下标运算和赋值运算等运算符外,还提供了位运算、地址运算、成员运算等运算符,这些运算符有助于程序员编写出高效的系统软件。

(2) C 语言有丰富的数据类型。C 语言包括整数型、字符型、实数型、空类型等基本数据类型和数组、指针、结构、共用、枚举、位结构等构造数据类型,还允许用户自定义新的数据类型。

(3) 虽然从严格的学术观点上看,C 语言是块结构(block-structured)语言,但是它还是常被称为结构化语言。C 语言提供了 3 种基本程序设计结构,即顺序结构、选择结构和循环结构。采用 3 种基本结构反复嵌套可实现任何复杂的运算。

(4) C 语言是模块化语言。C 程序由函数组成,这些函数可以是系统提供的库函数,也可以是用户自定义的函数,程序员可以利用函数构造计算机程序。

(5) 任何一个 C 程序有且仅有一个称之为主函数的 main 函数。程序执行从主函数开始,其他函数通过主函数直接或间接调用才能执行,主函数执行结束时,标志程序执行结束。

(6) C 语言有丰富的预处理功能。预处理有利于提高程序的可读性、可移植性、正确性和书写程序的高效性。

(7) C 语言是面向过程的语言,其函数采用面向过程的思想进行设计。

(8) C 程序具有可移植性。与汇编语言相比,用 C 语言写的程序可移植性好。不同的程序员可以在不同的平台上设计实现某一大型软件中的子功能,然后在另一平台上进行组装,构成大型软件。

## 1.3 C 程序的结构及书写格式

### 1.3.1 C 程序的结构

任何一种程序设计语言都具有特定的语法规则和规定的表达方法。一个程序只有严格按照语言规定的语法和表达方式编写,才能保证编写的程序在计算机中能正确地执行,同时也便于阅读和理解。在介绍 C 程序的基本结构与特征前,我们先看如下两个 C 程序的例子。

**例 1.1** 向控制台输出信息“Hello, World.”。

```
#include <stdio.h> /*预处理命令: 包含有标准输入输出库函数的头文件 stdio.h*/
main() /*主函数*/
{
    printf("Hello,World.\n");
}
```

在上面的代码中,main 是主函数的函数名,表示这是一个主函数。每一个 C 源程序都必须有,且只能有一个主函数(main 函数)。函数调用语句 printf 的功能是把要输出的内容送到显示器去显示。printf 函数是一个由系统定义的标准函数,可在程序中直接调用。

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:

```
Hello,World.
```



注意

在上面的代码中, #include 是 C 语言预处理器指示符, #和 include 之间可以有多个空格, #也不一定要顶格, 但是一定是第一个非空白字符。#include 的作用, 是告诉编译器在编译前, 要做些预先处理: 将后面<>中的文件内容包含到当前文件内。所谓包含, 是指将<>中列出的文件的内容复制到当前文件里。

### 例 1.2 输入两个数 a、b, 并输出其总和。

解题思路: 输入两个数 a、b, 调用自定义函数 add(a,b), 求出其和并赋给 a, 然后输出总和 a。

```
#include "stdafx.h"
main()/*主函数*/
{
    void show(); /*函数声明*/
    int add(int a,int b);
    int a=5; /*指定 a 为整数, 初始值为 5 */
    show (); /*调用函数 show , 无返回*/
    a = add(a,2); /*调用函数 add , 结果返回给 a */
    printf("程序运行结果为: %d\n",a);
}
void show() /*定义函数 show , void 指定该函数不返回结果*/
{
    printf("add a+b. \n");
}
int add(int a,int b) /*定义函数 add , int 指定该函数返回一个整数*/
{
    return (a+b) ; /*返回 a+b 的值 */
}
```

本程序的执行过程是:

- 程序从 main()处开始。
- 变量 a 代表一个整数, 并且初始值为 5。
- 执行程序(函数) show(), 屏幕上显示 add a+b, \n 为转义字符, 代表换行的意思。
- 执行程序(函数) add(a,2), 并将结果赋予 a, 此时, a 的值为 7。
- 屏幕上显示“程序运行结果为:7”。

从上面程序例子, 可以看出 C 程序的基本结构。

C 程序为函数模块结构, 所有的 C 程序都是由一个或多个函数构成的, 其中必须只能有一个主函数 main()。程序从主函数开始执行, 当执行到调用函数的语句时, 程序将控制转移到调用函数中执行, 执行结束后, 再返回主函数中继续运行, 直至程序执行结束。C 程序的函数分为由编译系统提供的标准函数(如 printf、scanf 等)和由用户自己定义的函数。函数的基本形式是:

[返回值类型] 函数名([形参说明表])

```
{
    变量定义部分;
    语句执行部分;
}
```

其中, C 函数中要用到的变量必须先定义, 然后才能使用, 因此变量定义在执行语句前。

## 1.3.2 C 程序的书写格式



在例 1.2 出现最多的是/\*和\*/包裹起来的中文语句，这些是注释。注释是用来帮助程序员阅读源代码和理解源代码的。编译器在编译源代码的时候，在目标代码生成以前，会把注释剔除掉然后再进行编译，当然编译器是不会修改源文件的，这一切是在内存中完成。由于对于注释部分忽略不处理，就如同没有这些字符一样，所以注释不会增加编译后的程序的可执行代码长度，对程序运行不起任何作用。

所谓书写格式即代码风格，是编写代码时对代码的排版布局，如何命名代码中的变量名称、函数名称等。一个优秀的程序员，除了代码质量高，错误少之外，另一个非常重要的标准就是在代码风格方面。从书写清晰，便于阅读、理解、维护的角度出发，在书写程序时应遵循以下规则。

(1) C 程序采用块注释方法，块注释书写方法为：

```
/* 注释部分 */
```

注释部分只为了提高程序的可读性，不参与程序的编译和运行。但在书写格式上要注意：“/”与“\*”之间或“\*”与“/”之间不能有空格。C 程序在 Visual C++ 6.0 编程环境下，也可采用 C++ 的注释方法，即若要对该行进行注释，只需在该行前面加上两个正斜杠符“//”即可。

(2) C 语言一般采用小写字母作为标识符。而 BASIC 语言中，一般采用大写字母作为标识符。

(3) C 语言是区分大小写的。如：COUNT、count 和 Count 表示的是 3 个不同的标识符。

(4) C 程序书写格式灵活，一个语句可连续写在多行上，一行也可以写多个语句。如例 1.2 中的 max 函数可以写成如下形式：

```
int add(int a,int b){ return (a+b);}
```

(5) 为了使书写的程序结构清晰、层次分明，建议采用“缩进对齐”的格式编辑 C 语言源程序，即同一结构层次的语句应左对齐，而结构下的语句相对于结构本身而言向右缩进。

C 程序书写格式灵活，这对程序员书写程序没有什么约束，如标识符可以采用小写字母，也可以采用大写字母表示，程序可以采用缩进对齐的格式书写，也可以不采用缩进对齐的格式书写，但我们建议初学者养成良好的程序书写规范，以便于交流和调试。

## 1.4 C 语言开发环境的搭建

### 1.4.1 C 程序可执行文件的生成过程

C 语言程序可执行文件的生成过程如下。

(1) 利用编辑器生成文本文件，该文本文件又称为 C/C++ 源程序，其扩展名为.c++或.c。编辑器可以是 C 系统提供的，也可以是其他文本编辑器，如：Notepad、Edit、Edlin 等。

(2) 采用 C 编译器将源程序编译为二进制的机器目标文件，生成的目标文件扩展名为.obj。

(3) 采用 C 连接程序将目标文件与库文件连接，生成可执行文件，可执行文件扩展名为.exe。

上述三步过程如图 1.1 所示。



图 1.1 C 程序可执行文件的生成过程

## 1.4.2 Visual C++ 6.0 上机操作过程

Visual C++ 6.0 开发环境是一个基于 Windows 操作系统并包含 C 语言子集的可视化集成开发环境 (Integrated Development Environment, IDE)，该开发环境提供了优秀的代码编辑功能，同时提供了编译连接程序。在该开发环境里，输入完源代码，可立即编译运行，并且可以参照代码进行调试，这些操作都可以通过单击菜单选项或工具栏按钮来完成，使用方便、快捷。在 Visual C++ 6.0 开发环境下，C 程序按工程 (project) 进行组织，每个工程可包括一个或多个 C/CPP 源文件，但只能有一个 main 函数。下面以例 1.1 为示例 (例 1.1 源文件命名为 ch1\_test1.c) 介绍在 Visual C++ 6.0 IDE 中建立工程并进行 C 程序调试的主要操作步骤。



VC++ 并不是一门语言，而是开发环境。一般来说，现在进行软件开发，都是在集成的编程环境中进行。当然如果愿意，也可以单独编辑源代码，然后用命令行编译并连接程序。由于 Visual C++ 6.0 的汉化版本很多，菜单项的汉化名称不尽相同 (如主菜单项“Build”，有的版本翻译成“组建”，有的版本则翻译成“编译”，而其下拉菜单项中第二个子菜单项名也叫“Build”，有的版本翻译成“生成”，有的翻译成“构建”)，所以下面在介绍相应菜单项名称时，用圆括号附上其英文菜单项名。

### 1. 启动 Visual C++ 6.0 IDE

可以从桌面上或“开始”按钮中的“程序”项中启动 Visual C++ 6.0 IDE，如图 1.2 所示。集成开发环境分为标题区、菜单区、工具栏区、工作区、程序编辑区、调试信息区等，如图 1.3 所示。

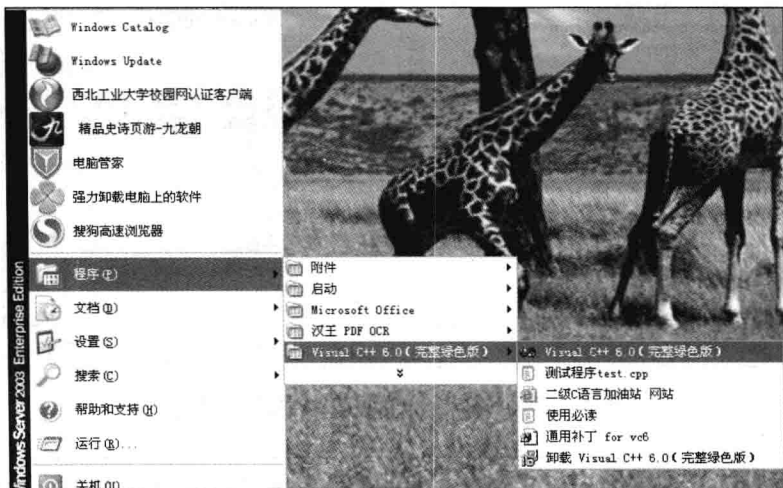


图 1.2 从开始按钮启动 Visual C++ 6.0 IDE

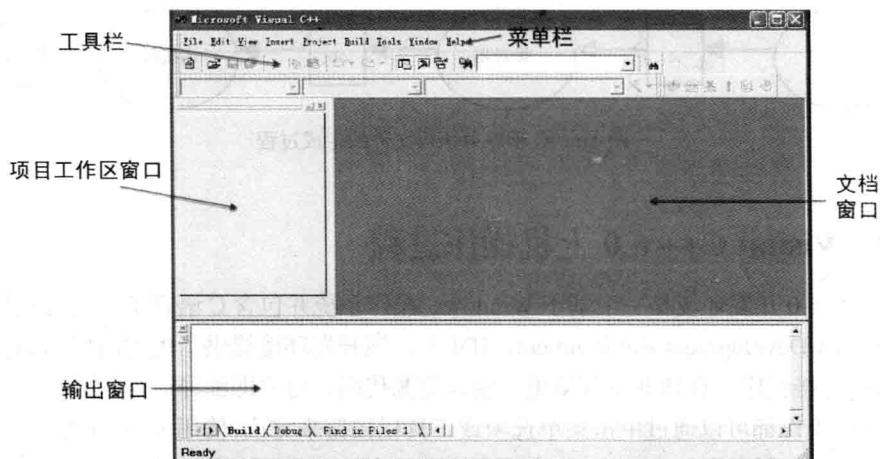


图 1.3 Visual C++ 6.0 IDE 界面

## 2. 工程 (Project) 的创建

从 Visual C++ 6.0 IDE “文件 (File)” 菜单上选择 “新建 (new)” 菜单项，此时将弹出新建对话框，如图 1.4 所示。该对话框有分别用于创建新的“文件”、“工程”、“工作区”和“其他文档”等 4 个选项标签。先选择“工程”标签项，然后选中“Win32 Console Application”项，在“工程”文本框中输入欲建工程名称，如 ch1\_test1 (Visual C++ 6.0 IDE 自动将用户输入的工程名作为文件夹名)；然后在“位置”文本框中输入欲保存该工程的路径，或是通过单击其右边的 [...] 按钮，在弹出的“选择目录”对话框中选择保存路径。单击“确定”弹出如图 1.5 所示的界面，在图 1.5 中选择“一个空工程 (An empty project)”后单击“完成”按钮。然后在“新建工程信息 (New Project Information)”对话框中单击“确定”按钮即可。

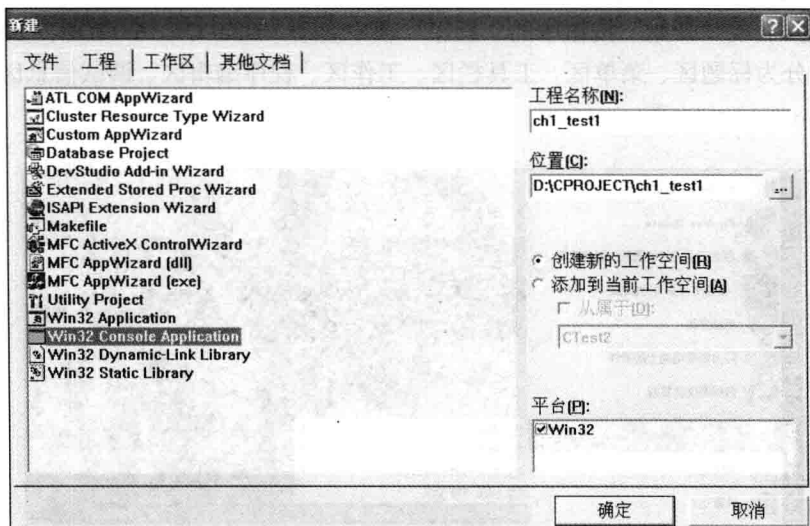


图 1.4 Visual C++ 6.0 IDE 的“新建”对话框



在输入工程名之前，【确定】按钮是灰色的表示不可以使用。请读者记住工程的路径，后面将会在这个路径下面查看生成的文件。

### 3. 在工程 (Project) 中添加并编辑源程序

从 Visual C++ 6.0 IDE “工程 (Project)” 菜单上选择“添加到工程 (Add to project)”菜单项, 然后单击“新建 (new)”下拉菜单项, 弹出界面如图 1.6 所示, 选择文件类型为 C++ Source File, 输入源文件名 (如 ch1\_test1.c, 注意加上扩展名.c, 若不加则默认扩展名为.cpp), 选择保存源文件位置, 单击“确定”按钮后将生成一个新的空文件 ch1\_test1.c, 并弹出源文件编辑窗口 (如图 1.7 所示), 在编辑窗口中输入程序代码并修改, 完成后可保存源文件。程序员也可按这种方式向工程中增加其他源文件。

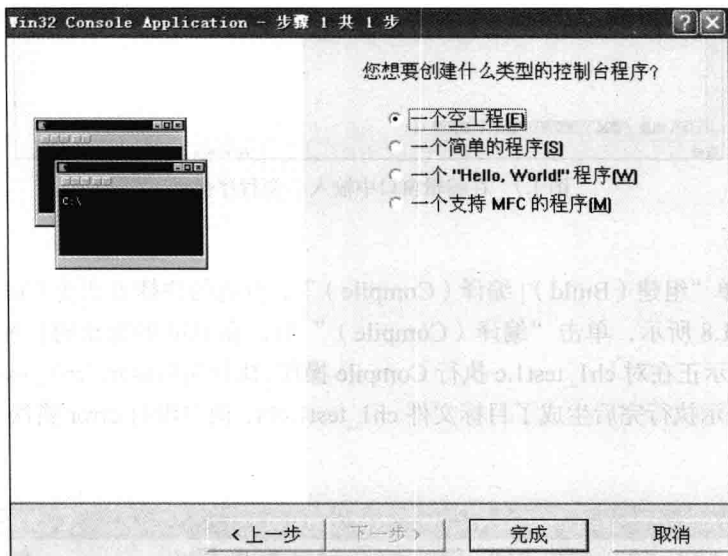


图 1.5 Visual C++ 6.0 IDE 创建控制台项目类型选择



注意

如果有多个源文件, 可以在该对话框中通过“添加到工程 (Add to project)”按钮继续添加新的文件到工程列表中, 也可以在以后需要的时候, 通过菜单命令添加新文件。

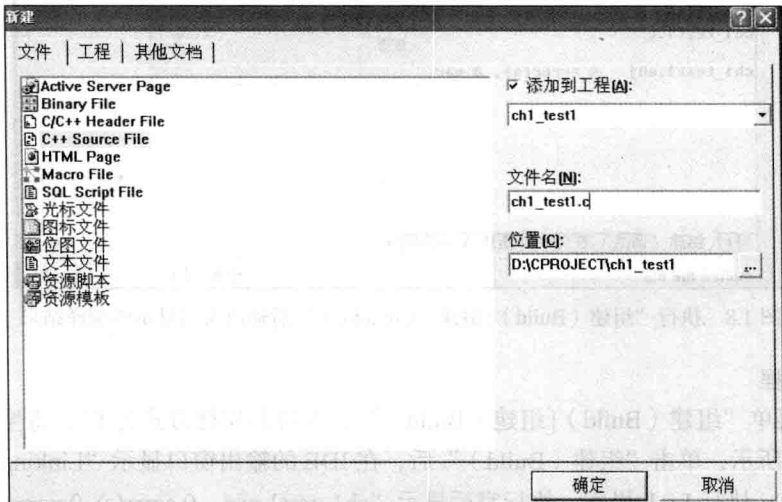


图 1.6 在工程中添加文件的对话框