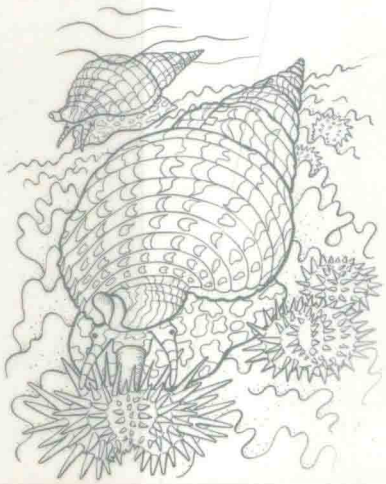
工业和信息化普通高等教育“十三五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计与应用（第2版）

Programming and Application of the C Language (2nd Edition)

张小东 郑宏珍 主编
初佃辉 主审

- 全面的基础圈点，轻松构筑程序框架
- 精要的解析方法，平滑实现基础转换
- 生动的案例分析，提高探索创新技能



高校系列

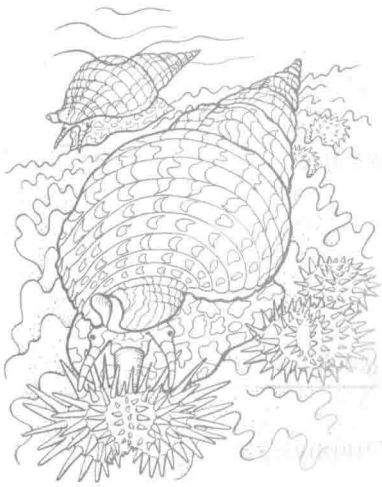
中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

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

C语言程序设计与应用（第2版）

Programming and Application of the C Language (2nd Edition)

张小东 郑宏珍 主编
初佃辉 主审



高校系列

人民邮电出版社
北京

图书在版编目（CIP）数据

C语言程序设计与应用 / 张小东，郑宏珍主编. -- 2版（修订本）. -- 北京 : 人民邮电出版社，2017.9（2018.8重印）
21世纪高等学校计算机规划教材
ISBN 978-7-115-46862-8

Ⅰ. ①C… Ⅱ. ①张… ②郑… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2017)第219059号

内 容 提 要

本书以 C 语言的基本知识为基础，以教育部考试中心公布的全国计算机等级考试大纲（二级 C 语言）为依据，并结合基本的工程实践编写而成。全书共分 9 章，包括：简单 C 程序设计、选择控制结构与应用、循环结构与应用、模块化设计与应用、数组及其应用、深入模块化设计与应用、构造型数据类型与应用、综合设计与应用和数据永久性存储等内容。

本书注重可读性和实用性，从计算机软件工程的角度展开讲解、探索和论述。每章开头都有关键词和难点提示，每章结尾安排本章小结，从知识层面和方法层面对本章进行总结。本书从日常生活和实际工程中所遇到的问题出发，运用多种方法对典型例题进行求解，强化对知识点、算法、编程方法与技巧的把握。同时本书还融入了程序测试、程序调试、软件的健壮性、代码风格、结构化与模块化程序设计方法等软件工程方面的知识。

本书可作为高等学校 C 语言程序设计课程的教材，也可作为全国计算机等级考试参考书及 C 语言自学教材。

◆ 主　　编　张小东　郑宏珍
主　　审　初佃辉
责任编辑　张　斌
责任印制　陈　犇

◆ 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
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：18.5　　2017 年 9 月第 2 版
字数：524 千字　　2018 年 8 月河北第 4 次印刷

定价：49.80 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：京东工商广登字 20170147 号

本书编审人员

主　审　初佃辉

主　编　张小东　郑宏珍

副主编　孟凡超　张　华　李春山　周学全　喻小光　张维刚

参　编　郭长勇　闫健恩　张　华　向　曦　马　帅　刘艺姝

张壹帆　崔　杨　倪　烨　过友辉　衣景龙　张天昊

张博凯　杨　帆　刘　娟

前　言

目前市面上有关C语言方面的图书，内容大都按传统思路组织：从C语言的历史讲起，然后是数据类型、运算符、表达式、常量、变量、控制结构、数组、指针、函数、结构体、共用体及文件等。内容涵盖C语言中最基本的知识点：由单词到句子，再程序段等，逐渐复杂化，分步进行讲解。表面上看，这样符合人类认知的规律，把握问题的关键与语言的主体，能够使读者很快掌握C语言。然而，经过本书编者多年的教学之后，发现这样并没有达到预期的实践效果，多数学生学习时并不是很懂，虽然能考个不错的分数，但是学习结束后却无法上手进行工程实践，甚至不能解决很简单的命题。显然，这与我国的传统应试教育模式有很大的关系，但就事论事地讲，这与现在使用的教材也存在着一定的关系。其实，这种传统的学习方式与长久以来人类学习和掌握语言的方法大相径庭。回想父母教幼儿说话的过程，从没有人尝试过先教汉字的偏旁部首或英文的abc，然后再教孩子单词、词法、语法，进而组成语句……如果这样，孩子恐怕一辈子也学不会说话！教幼儿说话时，先是从很具体、很简单又能跟实物相对应的单词开始，如爸爸、妈妈、苹果等，充分发挥孩子对事物的好奇心和感知力，循序渐进去认知世界，进而运用丰富的语言信息表达自己的思想。相对而言，虽然一般的C语言教材中开始讲解的数据类型、运算符、表达式、常量、变量等知识内容很简单，但几乎没有任何应用与它们相对应，把那么多抽象的知识点简单地堆到一起，然后让已经进入理解性记忆的读者去学习，恐怕难于接受。而后又将很多难点集中到一起，如数组、指针、函数，读者就更加难以理解这门语言了。这种目的性、应用性不强的组织方式，诱使读者进入被动的学习过程中，于是在他们的脑海里形成了一个个的信息孤岛，以至于无法完全理解语言的奥妙，更谈不上使用C语言去解决一些实际的命题了。

本着学以致用的原则，本书将从第1章开始就鼓励读者“张嘴说话”——学习编写简单应用程序，把构成C语言的最基本的知识点揉进去，像自然语言的“自然”学习方法一样（如妈妈教幼儿学说话般），不做单纯的知识点堆积，而是将知识点与实例相结合，具体而实用，使读者更容易接受。随后，把传统组织方式当中前几章的那些“简单的”知识点分散到用到它们的各个章节中，在充分体现它们的应用价值和使用技巧的同时，也分散了难点，避免了机械记忆。本书以应用为主线，体现出“用到了才讲，讲是为了更好地用”的编书理念，注重培养读者的工程实践能力，从问题的规范描述开始，依次进行分析问题、模型建立、程序设计描述、程序实现，直到最后的程序测试、结果分析或程序解读。从严格的科学研究与工程应用的角度出发，进行C语言的学习与研究，并将这种思想渗透到每一个实例中！

本书在提供丰富而有趣的经典实例的同时，还精心设计了两个相对完整的应用：计算器与学生成绩档案管理。计算器属算法研究四大类问题之一的计算类问题，包括很多经典的数值计算方法，如应用泰勒（Taylor）公式求解三角函数等；学生成绩档案管理属非数值运算处理的问题，而计算机处理的绝大部分数据是非数值信息，因此本例在实际应用中具有一定的代表性。学生成绩档案管理系统从最简单的单个学生成绩分类开始，到用多维数组存储学生基本信息与成绩信息，利用冒泡排序与选择排序按不同科目、不同成绩进行排序，再到更有聚合力的组织方式——结构体，以及“新”的数据管理方式——链表，最终能够将这些数据永久性地存储到

文件中，完全贯彻实用、实践和工程应用的目的。通过这两个实例的学习，可以让读者对 C 语言程序设计有一个更全面的认知，能够综合运用所学知识解决较为实际的问题。

与第 1 版相比，第 2 版继承了第 1 版的优点：以工程实践能力为主要目标，注重 C 语言设计思维的培养与训练，同时，去除了第 1 版中的一些缺点，如知识点过于分散、两个实例在后继章节的编写中过于冗长，难以理解等。首先，微调了某些知识点，如把指针从第 2 章调整到第 5 章，与数组进行对比讲解，且在其后的每一章都有相关的比较和训练，更利于循序渐近地学习。出于同样的学习方法和目的，将第 2 章的位运算符调整到第 7 章的位段中进行对比讲解。其次，进一步强化设计思维的培养和训练，多数例题都会分为问题分析、程序设计描述、程序实现、结果展示及程序解读几个部分来进行讲解，脉络更加清晰，更容易使读者养成良好的程序设计习惯。最后，将两个大程序设计分解为灵活小巧的程序段，既能合成一个较为全面的应用系统，又可以作为学习某几个知识点的小程序而独立运行。总之，通过几年的应用实践，这本书取得了一定进步，期待它能够更好地帮助读者学习 C 语言程序设计。

全书由张小东、郑宏珍担任主编，第 1、2、4 章由张小东编写，第 3 章由孟凡超编写，第 5 章由喻小光编写，第 6 章由李春山编写，第 7 章由张华编写，第 8 章由周学全编写，第 9 章由张维刚编写。张小东、郑宏珍负责全书的统稿。

在本书的编写过程中，初佃辉教授在百忙之中审阅了全部初稿，对本书提出了很多宝贵意见。在书稿的录入、校对及例题和课后习题的调试过程中，郭长勇、闫健恩、张华、向曦、马帅、刘艺姝、张壹帆、崔杨、倪烨、过友辉、衣景龙、张天昊、张博凯、杨帆、刘娟等同志做了大量的工作，在此向他们表示衷心的感谢。

因编者水平有限，书中错误在所难免，恳请读者批评指正。欢迎读者给我们发送电子邮件，对本书提出宝贵意见。作者 E-mail 地址为 z_xiaodong7134@163.com，zhua547@163.com。

目 录

第 1 章 简单 C 程序设计 ……1
1.1 C 程序的构成 ……1
1.2 C 语言的入门知识 ……3
1.2.1 C 语言的常见标识符号 ……3
1.2.2 基本数据类型 ……5
1.2.3 格式化输出输入函数 ……7
1.2.4 C 语言的书写规则 ……8
1.3 简单 C 程序的扩展 ……8
1.3.1 基本功能设计 ……8
1.3.2 程序设计描述的方法 ……8
1.3.3 程序实现及常见错误分析 ……10
1.3.4 浅谈编程风格 ……11
1.4 本章小结 ……13
练习与思考 1 ……14
第 2 章 选择控制结构与应用 ……16
2.1 选择结构的基本运算符 ……16
2.1.1 关系运算符及表达式 ……16
2.1.2 逻辑运算符及表达式 ……17
2.2 if-else 选择结构 ……18
2.2.1 if 结构 ……18
2.2.2 if 语句的嵌套 ……24
2.2.3 表达式在 if 结构中使用的拓展 ……26
2.3 switch 选择结构 ……27
2.3.1 基本定义及应用 ……27
2.3.2 if-else-if 语句与 switch 语句 ……29
2.4 应用实例 ……31
2.4.1 计算器 ……31
2.4.2 学生成绩管理 ……33
2.5 本章小结 ……35
练习与思考 2 ……36
第 3 章 循环结构与应用 ……38
3.1 概述 ……39
3.2 循环控制结构 ……39
3.2.1 for 循环 ……39
3.2.2 while 循环 ……45
3.2.3 do while 循环 ……47
3.2.4 goto 循环 ……49
3.3 循环控制结构的设计 ……50
3.3.1 循环的嵌套 ……50
3.3.2 循环的控制 ……52
3.3.3 循环语句的选择 ……58
3.4 应用实例 ……58
3.4.1 计算器 ……58
3.4.2 学生成绩档案管理系统 ……62
3.5 本章小结 ……65
练习与思考 3 ……66
第 4 章 模块化设计与应用 ……68
4.1 模块化程序设计方法 ……69
4.1.1 模块化程序设计思想 ……69
4.1.2 模块规划实例 ……69
4.2 函数 ……71
4.2.1 函数的定义 ……72
4.2.2 函数的调用 ……73
4.2.3 函数设计实例 ……74
4.2.4 函数调用的执行过程 ……78
4.3 预处理 ……79
4.3.1 文件包含 ……79
4.3.2 宏定义 ……80
4.4 应用实例 ……85
4.4.1 计算器 ……85
4.4.2 学生成绩档案管理 ……87
4.5 本章小结 ……90
练习与思考 4 ……90
第 5 章 数组及其应用 ……93
5.1 数组与数组元素的概念 ……93
5.2 一维数组 ……95
5.2.1 一维数组的定义 ……95
5.2.2 一维数组的初始化 ……97
5.2.3 一维数组的使用 ……98
5.2.4 一维数组综合应用 ……99
5.3 二维及多维存储 ……100
5.3.1 二维数组的定义 ……101
5.3.2 二维数组的初始化 ……102
5.3.3 二维数组元素的使用 ……103
5.3.4 多维数组的初始化和引用 ……105
5.3.5 数组程序综合应用 ……106
5.4 字符类型数组及字符串 ……108
5.5 字符串处理函数 ……111

5.6 指针变量、字符串指针变量与字符串……114
5.6.1 指针变量……114
5.6.2 字符串指针变量……118
5.6.3 字符串数组和字符串指针……119
5.7 综合应用实例……119
5.8 本章小结……129
练习与思考 5……129
第 6 章 深入模块化设计与应用…132
6.1 算法的基本概念……132
6.1.1 概念……133
6.1.2 引例……134
6.2 简单的排序算法……136
6.2.1 冒泡排序算法……137
6.2.2 选择排序算法……141
6.3 嵌套与递归设计与应用……145
6.3.1 函数的嵌套调用……145
6.3.2 函数的递归调用……147
6.4 模块间的批量数据传递……152
6.4.1 指针作为函数参数……152
6.4.2 一维数组作为函数参数……153
6.4.3 二维数组作为函数参数……154
6.5 模块化设计中程序代码的访问……155
6.6 应用实例……156
6.6.1 计算器……157
6.6.2 学生成绩管理……159
6.7 本章小结……164
练习与思考 6……165
第 7 章 构造型数据类型与应用…169
7.1 结构体……170
7.1.1 结构体类型的定义……170
7.1.2 结构体变量……171
7.1.3 结构体数组……174
7.1.4 结构体指针……177
7.1.5 结构体与函数……181
7.2 位运算与位段……182
7.2.1 位运算……182
7.2.2 位段……184
7.3 共用体……186
7.3.1 共用体类型定义……186
7.3.2 共用体变量定义……186
7.3.3 共用体变量的赋值和引用……187
7.4 枚举……189
7.5 自定义类型……189
7.6 应用实例……190
7.7 本章小结……200
练习与思考 7……201
第 8 章 综合设计与应用…… 203
8.1 变量的作用域与存储类别……203
8.1.1 变量的作用域……203
8.1.2 变量的存储类别……207
8.2 指针与数组……210
8.2.1 一维数组与指针……210
8.2.2 多维数组与指针……212
8.2.3 指针数组……215
8.3 函数 main()中的参数……216
8.4 指针型函数……219
8.5 链表……221
8.5.1 链表的概念……221
8.5.2 链表的基本操作……222
8.5.3 带头结点链表简介……232
8.6 本章小结……233
练习与思考 8……234
第 9 章 数据永久性存储…… 238
9.1 数据的永久性存储……238
9.2 文件组织方式……239
9.3 文件操作……241
9.3.1 标准输入输出头文件 stdio.h……241
9.3.2 文件打开与关闭……243
9.3.3 文件读/写函数……246
9.3.4 文件定位函数……254
9.4 综合应用实例……256
9.5 本章小结……258
练习与思考 9……259
附录 C 语言参考资料…… 262
附 1 C 语言发展史及版本历程……262
附 2 C 语言编辑软件简介……263
附 3 C 语言关键字……268
附 4 标识符的命名方法……269
附 5 ASCII 表……270
附 6 VC++各数据类型所占字节数和取值范围……273
附 7 C 语言运算符及优先级……273
附 8 格式化输入/输出控制……274
附 9 程序流程图……278
附 10 ANSI C 常用标准库函数……282
参考文献…… 288

第 1 章 简单 C 程序设计

内容提示

关键词

- C 语言的基本构成
- 注释、分隔符、标识符、关键字
- 数据类型、变量、运算符
- 流程图

难点

- 绘制流程图
- 运算符的优先级与结合性
- 格式化输出函数 printf 和格式化输入函数 scanf

随着互联网的崛起，计算机对人们的生活影响越来越大，正在逐步地改变着人们的生活方式，因此学会使用计算机将成为我们谋生的重要手段之一。软件赋予计算机以“生命”，离开软件的计算机相当于一堆废铜烂铁，难以使用。然而，使用软件和设计编写软件的概念并不相同，软件的设计与编写能够让我们更好地理解、掌握和控制计算机，使之成为我们研究相关专业领域的重要工具。在众多程序设计语言中，C 语言无疑是一颗璀璨的明星！

C 语言从诞生至今，已经走过了 40 多年的辉煌历程，其以紧凑的代码、高效的运行、强大的功能和灵活的设计与使用而闻名于世，受到众多编程人员的青睐。下面就让我们步入 C 语言的殿堂，揭开它“古老”而神秘的面纱，开启我们的编程之旅。

1.1 C 程序的构成

“说”是学好语言的最佳方法之一，C 语言也不例外。下面就让我们先“说”出第一个程序，跟奇妙的 C 语言打个招呼，了解一下它的基本构成。程序清单 1-1 包含了 C 语言程序的一些基本特征。仔细阅读程序中的代码，凭借对英文单词的理解及后续运行的结果，尝试猜测程序中每一行所起的作用，总结出该程序的基本框架，然后与后续解读进行对比。

程序清单 1-1　dream.c

```
/* 一个简单的 C 程序实例 */
/* purpose: I have a dream
      author : Xiao Zhang
      created: 2017/01/01 16:46:08
```

```
   */
#include <stdio.h>
int main(void)
{
    int nNumber;
    nNumber = 1;
    printf("Hello C language!\n");
    printf("I have a dream that one day I will solve the Goldbach's conjecture"
           "problem and prove that %d + %d = %d by using C!\n,"
           "nNumber, nNumber, 2);"
    return1;
}
```

使用 C 语言编辑软件（参见附录 2）输入程序清单 1-1 中的内容，保存成以.c 结尾的文件，如 dream.c，再进行编译、链接和运行后，该程序的输出结果如下所示。

```
Hello C language!
I have a dream that one day I will solve the Goldbach's conjecture problem and prove
that 1 + 1 = 2 by using C!
```

结果并不奇特，大部分内容都在程序清单中出现过。但是，程序清单中的“\n”和“%d”却消失得无影无踪，取而代之的却是换行效果和整数数值，这是怎么回事呢？

现在对上述代码作出解释。在“/*”和“*/”之间的内容，被称为注释。它不是程序代码，编译时，系统会自动删除它们，不会出现在目标程序中。读者可试着把这段内容删除，再编译运行一下，看看结果有什么变化。没有变化！那为什么还要加上所谓的注释呢？注释的最大作用就是方便阅读。也就是说，程序不光是让计算机执行任务，还有一个很重要的功能，就是让程序员之间能够通过阅读代码进行程序设计方面的交流。注释在这方面起着举足轻重的作用。书写注释是程序员一个非常好的习惯，读者应该从开始学习编程时就养成这个习惯。有些程序员为了设计与调试方便，甚至在每一行重要的代码旁边都加注释。对于行注释，C 语言还提供了另一种更为方便的方法——用“//”，这种注释只在一行内有效。例如：

```
nNumber = 1;//对变量 nNumber 进行初始化
```

#include <stdio.h>被称为预编译指令，它的含义是把一个名为 stdio.h 的文件引入到本段程序中。在编程时，我们通常不是从零开始，而是站在“巨人”的肩膀上。也就是说，在程序设计中，可能会用到其他工程师编写的一段段功能完善且相对独立的代码，这一段段代码被封装成一个个函数，分类归放到不同的文件中，需要时，可将它们“组装”进自己的程序中。方法是先通过 include 指令把这些文件包含进来，然后按名称调用这些函数，如 printf()。

int main(){……}被称为主函数。当程序被执行时，首先找到一个名叫 main 的函数，从它的“{”开始执行，到“}”执行完毕。所以说，主函数既是程序执行的“入口”，也是程序执行的“出口”。需要注意的是，在一个可运行的软件中，无论有多少个文件、代码有多么庞大，有且只能有一个主函数。

int nNumber 是变量定义语句，nNumber=1 是给变量 nNumber 赋值的语句。

两条 printf()语句非常相似：①它们都被称为输出语句，功能是向显示器输出相应内容；②都用一个“\n”在语句输出结束后换到下一行。但也略有不同：第二个 printf()多了三个“%d”，它们被统称为格式控制符。细心的读者可能会发现前两个“%d”被 nNumber 中的值所取代，第三个“%d”被 2 取代，没错！它们就是这个功能！之所以称它们为格式控制符，是因为不同类型的变量或常量用的是不同的格式。“%d”只适合输出整数，不能输出含有小数部分的数。若想输出带有小数的数，需另外一种格式控制。最后的“return 1”为 main()函数的返回值语句。

依据上述对代码的解释，可以得到 C 语言的基本组织结构，如图 1-1 所示。

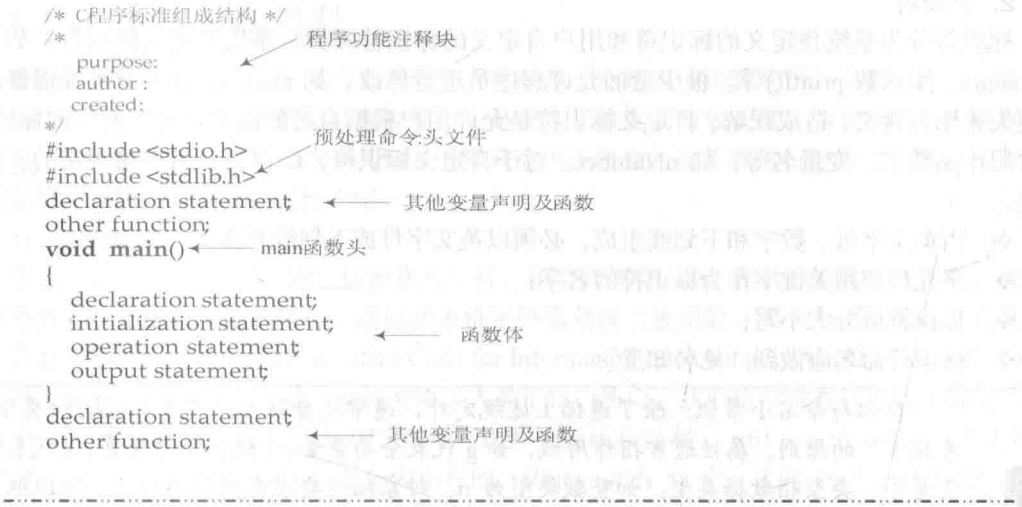


图 1-1　C 程序结构

细心的读者可能会发现这个 C 程序结构比程序清单 1-1 多总结出了“其他函数及代码段”。严格地说，对于功能比较多的程序，把所有代码全部写在 main()函数中是不合理的。在程序设计中，通常会把能够执行一定功能且相对独立的代码写到其他函数（函数名不能与 main 相重复）里，然后在主函数中调用这些函数。这些知识将在第 4 章中详细介绍。至于代码段，除变量定义可能被执行外，非函数形式的代码段是不能被执行的。注意前面对 main()函数的解释，没有名字是无法在 main()中被调用的。

套用这个结构模板，可以编写很多小程序。现在，读者就可以做一些简单的替换，比如修改变量名称，替换掉 printf()中的文字等。

在做完上述解释和总结后，可能读者心中仍然会有一些问题：为什么会有变量类型？它的作用是什么？定义变量有什么规定？能不能让计算机自动执行 nNumber+nNumber 的操作而不是让程序员心算？……接下来，就让我们一起学习上述代码所涉及的知识点，回答读者心中的疑问，然后利用这些知识，对上述程序进行扩充，编出一个功能稍微强大一点儿的程序。

1.2　C 语言的入门知识

每一门语言都有独特的组织方式，如词素、词素分隔、语句、段落等，C 语言也不例外——尽管它是一种面向计算机的语言。下面介绍一些简单的 C 语言知识。

1.2.1　C 语言的常见标识符号

程序清单 1-1 中有很多带一定含义的标识符号，如 int、nNumber 等，这些符号分别代表不同的含义。C 语言的常用符号主要有以下 5 类。

1. 关键字

关键字，又称为保留字，是 C 语言构成语句的基本词汇。它们由 C 语言预先定义，具有固定的含义，如 int。程序员只能按预定含义使用它们。C 语言共有 32 个关键字，下面的章节中会陆续介绍。关键字的汇总参见附录 3。

2. 标识符

标识符分为系统预定义的标识符和用户自定义的标识符两类。系统预定义标识符，如主函数 main()、库函数 printf()等。极少量的允许程序员进行修改，如 main()；大部分不提倡修改，以免失去原有含义，造成误解。自定义标识符是允许用户根据自己的需要定义并使用的标识符，通常用作函数名、变量名等，如 nNumber。对于自定义标识符，C 语言是有严格规定的，如下所述。

- ✧ 由英文字母、数字和下划线组成，必须以英文字母或下划线开头；
- ✧ 不允许使用关键字作为标识符的名字；
- ✧ 标识符区分大小写；
- ✧ 标识符命名应做到“见名知意”。

标识符命名小常识：除了遵循上述规定外，通常还会融入“变量名=属性+类型+对象描述”的原则。属性通常指作用域，如 g 代表全局变量，l 代表局部变量，s 代表静态变量等。类型指数据类型，如整数类型为 n。对象描述要求有明确的含义，可以取对象名字全称或一部分，如数字为 Number 或 Num。那么，程序清单 1-1 中 nNumber 的含义为：局限于 main 函数内的整数类型的数字变量。这样定义容易理解，并且便于程序员之间进行交流。更多内容详见附录 4。

3. 分隔符

像写文章要有标点符号一样，写程序也必须有分隔符，否则，程序不但难于阅读理解，还会出错。在 C 程序中，空格、回车/换行、制表符（键盘上的 Tab 键）、逗号、分号等，在各自不同的场合起着分隔的作用，如下面语句所示。

```
int l_nA,l_nB,l_nC;  //定义了三个整数类型的变量 l_nA、l_nB、l_nC
```

在这个例子中，空格作为表示整数类型的关键字 int 与三个自定义变量之间的分隔，逗号作为三个变量之间的分隔，分号作为语句之间的分隔。

4. 运算符

运算符是代表 C 语言运算规则的符号。C 语言有非常丰富的运算符，可将它们分为算术运算符、关系运算符、逻辑运算符等，详见附录 7。前面已经见过一个运算符，即赋值运算符“=”。现在再推荐几个非常熟悉的数学符号，如表 1-1 所示。但要注意，不是所有的数学符号都能直接在 C 语言中应用。学好运算符，只需把握住三点即可：一是运算符的优先级，二是运算符的结合性，三是操作数个数。三者共同决定了含有多个运算符的表达式的运算次序。

表 1-1　　运算符

优先级	运算符	分类	操作数个数	结合性
3	*、/、%（求余）	算术运算符	2	自左向右
4	+、−	算术运算符	2	自左向右
14	=	赋值运算符	2	自右向左

由此可知，这六个运算符与数学中的含义及运算规则基本相同。由运算符、常量与变量可以组成 C 语言的表达式，如 a=c+2*3，其中，a、c 为变量，而 2、3 为常量。下一小节将介绍这两个概念。

5. 其他符号

除了上述符号外，C 语言中还有一些特定含义的符号。如“{}”常用于标识函数体或一个语句块，“/**/”及“//”是程序注释所需的界定符等。

1.2.2　基本数据类型

数据类型是计算机语言发展的一大进步，它规范了计算机的运算规则，提高了计算速度，节省了内存空间。它的主要贡献包括两方面：一是规定了数据的有效长度，即数据的取值范围；二是规定了数据的运算规则，如求余（%）运算只适用于整数，而不能进行浮点型运算。接下来对 C 语言的基本数据类型分别进行介绍。

1. 字符类型

形如“a”“b”“c”……的数据被称为字符，用字符类型 char 表示。然而，计算机只能存储二进制数据，无法直接存入字符，因而必须将字符编制成二进制数。英文采用的编码方式是美国标准信息交换代码（American Standard Code for Information Interchange，ASCII），如字符“a”的二进制编码为 01100001。但是二进制不便于人类记忆，所以，我们也常常看到它的十进制、八进制或十六进制的表达，如“a”为 97、0141（0 开头表示八进制）、x61（x 开头表示十六进制）。在 C 语言中，英文字母的表示是区分大小写的，因此一共有 52 个（大写 26 个、小写 26 个）英文字母，加上数字型字符及一些特殊字符（如“*”和“$”等），一共不超过 128 个，所以用一个字节（一个 8 位二进制数）即可全部表示。

把字符放在一对单引号里的做法，适用于多数可打印字符，但某些控制字符（如回车符、换行符等）无法通过键盘输入将其放到字符串里。因此，C 语言还引入了另外一种特殊形式的字符常量——转义字符。它是以反斜线“\”开头的字符序列，使用时同样要放在一对单引号里。它有特定的含义，用于描述特定的控制字符，如程序清单 1-1 中出现的“\n”表示换行，更多的转义字符的表达见附录 5。

定义字符变量的关键字是 char，声明字符变量的方式为

```
字符类型关键字 变量名;
```

例如：

```
char cFirst;
```

字符类型与整数类型：因为计算机存储字符的方式是二进制，它是一种数字，这就使得字符类型的数据具有一定的整数特征。实际上，C 语言中是允许字符类型作为只有一个字节长度的整数类型进行运算的，所有适用于整数类型的运算都适用于字符类型。我们知道，整数是有正负的，那么在 C 语言里，字符也有正负。

2. 整数类型

整数类型与数学中的定义基本相同，只是计算机不能表达无穷大（小），所以，整数类型是有长度限制的。按照长短的不同，依次为字符型（char）、短整型（short int）、整型（int）、长整型（long int）。另外，整数类型在存储和运算时，分有符号（signed）和无符号（unsigned）。这两个关键字可被用作类型修饰符。整数类型的家族可汇总为表 1-2。

表 1-2　整数类型家族

修饰符	类型	组合结果	中文名称	简写	字节数	取值范围	举例
signed	char	signed char	有符号字符型	char	1	−128～127	signed char cFirst; char cSec;
	short	signed short	有符号短整型	short	2	-2^{15}～$2^{15}-1$	signed short sThr; short sFour;
	int	signed int	有符号整型	int/signed	4	-2^{31}～$2^{31}-1$	signed nFive; int nSix;
	long	signed long	有符号长整型	long	4	-2^{31}～$2^{31}-1$	signed long lSeven; long lEig;

续表

修饰符	类型	组合结果	中文名称	简写	字节数	取值范围	举例
unsigned	char	unsigned char	无符号字符型	无	1	0～255	unsigned char cFirst;
	short	unsigned short	无符号短整型	无	2	$0\sim2^{16}-1$	unsigned short sThr;
	int	unsigned int	无符号整型	unsigned	4	$0\sim2^{32}-1$	unsigned nFive;
	long	unsigned long	无符号长整型	无	4	$0\sim2^{32}-1$	unsigned long lSeven;

由表 1-2 容易看出几个特点：

✧ 有符号的数据类型可省掉类型修饰符 signed。

✧ int 与 long 长度相同，这是 C 语言设计时的规划预留。在后来的发展中，long 的长度视操作系统和编译器的不同而有所区别，读者可用 sizeof（long）对其进行测试。

✧ 有符号与无符号取值范围不同。

有符号的数据类型，最高位用于表示符号，0 表示正数，1 表示负数。无符号的全部表示值。以 short 型为例来看一下整数在内存中的存储形式。

最高位→ 0 111 1111 1111 1111　　十进制数为 32 767，有符号，最高位为符号位

最高位→ 1 111 1111 1111 1111　　十进制数为 65 535，无符号，最高位也是值，运算方式为 $1\times2^0+1\times2^1+1\times2^2+1\times2^3+1\times2^4+1\times2^5+1\times2^6+1\times2^7+1\times2^8+1\times2^9+1\times2^{10}+1\times2^{11}+1\times2^{12}+1\times2^{13}+1\times2^{14}+1\times2^{15}=65\,535$。

数值在计算机中均是以补码形式存储的。正数的补码与原码相同，而负数的补码=原码除符号位外全部取反+1。以 short 型-1 为例，其转换过程如下。

符号位→1 000 0000 0000 0001　　原码

符号位→1 111 1111 1111 1110　　反码

符号位→1 111 1111 1111 1111　　补码

补码：使用补码有三方面的好处：一是可以将减法运算变为加法运算，即加负数；二是可避免出现+0 与-0 的错误表达，即确保数字 0 表达的唯一性；三是如果两个数相加后产生溢出（结果超出类型所约束的范围），结果仍然是正确的。

定义整数类型变量的关键字和例子如表 1-2 所示，声明整数类型变量的方式为

```
[修饰符] 整数类型关键字 变量名;
```

3. 浮点类型

计算机中的实数类型被称为浮点类型，又称实型。顾名思义，这种数据类型的小数点是“浮动”的。何为浮动呢？举个最简单的例子：假设有一个字节的数据，如 0100 1111，若把小数点放到最右端 0100 1111.，则表示整数；若把小数点放到最左端.0100 1111，则表示一个纯小数。当小数点从右往左移动时，数的取值范围不断缩小，而精度不断提高。由此可见，精度要求不同的数据，小数点位置是不同的，称之为“浮动”。浮点型小数点的设置，对于初学者来说比较复杂，此处不做详解。

在 C 语言中，按不同的精度要求将浮点类型的数据分为单精度（float）和双精度（double）两种。float 型占 4 个字节，取值范围为 $-3.4\times10^{38}\sim3.4\times10^{38}$，有效位数是 7～8 位。double 型占 8 个字节，取值范围为 $-1.7\times10^{308}\sim1.7\times10^{308}$，有效位数是 15 位。有效位数与小数点的设置位置有关。

定义浮点类型变量的关键字为 float 和 double，声明浮点类型变量的方式为

```
浮点类型关键字 变量名;
```

例如：

```
float fNum;  double dNum;
```

依据上面的讲解，可以总结出变量的一般定义方式：

```
[修饰符]数据类型 变量名 1,变量名 2,…, 变量名 n;
```

上面反复提到了变量，那么，究竟什么是变量？有没有常量呢？

变量就是在程序运行时，可以根据需要不断被改变的量。它必遵循先定义后使用的原则，例如：

```
char cFirst='a', cSec='z';//定义字符并进行初始化，即赋值
float fNum=1.0f;
cFirst='c';
……
cFirst='x';
fNum=3.0f;
```

cFirst、fNum 在程序执行过程中，其值均发生了改变，称之为变量。

在上述代码段中，赋值符号“=”右边的被称为常量，如“a”“c”“x”。更准确地说，“a”“c”“x”叫字符常量，1.0f，3.0f 被称为单精度浮点型常量。当然，还有双精度浮点型常量、整型常量以及后面将要学习的枚举常量、符号常量等。

常量标识：由于整数类型和浮点类型有不同子类，C 语言表示时也做了相应的区分。如长整型会在常量值后面加 L 或 l，如-129l，1288L 等。无符号整型常量后面加 U 或 u，如 25u，44U，但是不能有-15u。无符号长整型则为两者的结合，如 25lu 等。单精度浮点型在常量值后跟 f 或 F，如 23.69f 等。双精度则什么也不用跟，如 25.88 等。

1.2.3　格式化输出输入函数

为了使程序员能够快速有效开发出程序，C 语言提供了丰富的库函数，其中就包括格式化输出/输入函数，即 printf()函数和 scanf()函数。

1. 格式化输出函数

printf()函数的作用是按指定格式向标准化输出设备（通常指屏幕）输出数据，其调用形式可以简单表示为

```
printf("<格式化字符串>", <参量表>);
```

格式化字符串是放在两个双引号之间，以%开头的合法的格式字符，如"%c"表示输出一个字符，"%d"表示输出有符号整型，"%f"表示输出单精度浮点型。"%"后的 c、d、f 称为格式控制符，在其前面还可以加上格式修饰符，如"%lf"表示输出双精度浮点型数据，"%ld"表示输出长整型数据。每个格式符必须对应参量表中的一个变量，而且必须与参量表里的变量类型相符，否则得不到正确的输出结果。例如：

```
char cValue='c';
int nNum=5;
float fNum=6.8f;
double dNum=7.55;
printf("字符:%c;整数:%d;单精度浮点型:%f;双精度浮点型:%lf \n", cValue, nNum,fNum,dNum);
```

由上述代码段可知，不受格式控制的字符串将原样输出，如%前面的字符串“字符”“整数”等和非格式控制符与修饰符“;”等。

2. 格式化输入函数

scanf()函数的作用是按指定格式从标准化输入设备（通常指键盘）读入数据，其调用形式可以简单表示为

```
scanf("<格式化字符串>", <参量表>);
```

scanf()函数的要求与 printf()函数相似，输入字符使用"%c"，输入有符号的整型数据使用"%d"，输入单精度浮点型数据使用"%f"等。不过，参量表中的变量前面需要加上一个符号"&"。"&"被称为取地址运算符，运算级别为 2，它的含义为把由键盘输入的数据存入参量表中指定地址的内存

中，并以回车作为输入结束。例如：

```
int nNum1,nNum2;
float fNum1, fNum2;
scanf("%d",&nNum1);//&nNum 表示取出 nNum 在内存中的地址，把从键盘输入的数据存入该地址
scanf("%f",&fNum1);
scanf("%d%f",&nNum2,&fNum2)
```

初学者注意，这里没有在“%”前面加任何字符，也没有在最后一个 scanf 中的"%d"与"%f"之间加空格或其他字符，请大家也不要加。关于这两个函数更为详尽的解释参见附录 8。

1.2.4 C 语言的书写规则

（1）C 语句都是以分号作为结束标志的，分号与特定内容结合可以形成不同的句式，如表达式加分号成为表达式语句：nNumber=1；函数调用加分号形成函数调用语句：printf("Hello C language");。

（2）C 程序的书写格式比较自由，既可以在一行上写多条语句，也可以把一条语句写在多行上。但是为提高程序的可读性和可测试性，建议读者一行只写一条语句。

（3）为了更加明确地表明一段代码的独立功能，可以对 C 语言代码划分段落。“段”是以“{}”进行划分的，每个段里可包含 0 条或多条语句。函数是“段”最常见的表达形式，“{}”也可以独立使用，如本书第 8 章中的例子。

（4）为了对程序进行必要的说明，可以添加注释，它有两种方式：/**/和//。

1.3 简单 C 程序的扩展

尽管程序清单 1-1 所展示的代码非常简单，但其所涉及的知识点对于初学者来说并不少。随后，我们对这些知识点进行了较为详细的讲解及适当的扩展。从程序清单 1-1 到讲解 C 语言入门知识的 1.2 节，显示出一条非常重要的信息：在没有特殊语法控制的前提下，C 语言的语句是按顺序执行的。顺序控制结构是 C 语言三种基本控制结构之一，其他两种结构——选择控制和循环控制将分别在第 2 章和第 3 章中陆续介绍。

下面将运用这些知识点及顺序控制结构，对程序清单 1-1 中的代码进行完善，把它扩展为一个可以进行代数运算的计算器。

1.3.1 基本功能设计

对于编程者来说，最重要的其实并不是语言学习，而是程序设计，它的本质是一种计算思维的培养。所以，先不要急于编写程序，在此之前应该先想好需要做什么，并用适当的方式表达出来，这就是所谓的程序设计。就目前我们所学习的知识来说，可以做以下三方面的功能规划。

- ✧ 字符定义及输出。
- ✧ 整数的加、减、乘、除、取余。
- ✧ 实数/浮点数的加、减、乘、除。

下面的例子中将只给出整数的加减、取余及浮点数的乘除，其余功能留给读者自行实现。

1.3.2 程序设计描述的方法

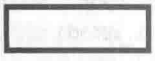
程序描述的方法不仅仅局限于代码编写。实际上，编写代码就好像建造一幢大厦过程中的现场施工阶段，在此之前，通常还要进行市场调研、建筑物的设计（绘制蓝图）等很多工作。这里

把程序描述看作“蓝图绘制”，程序描述的过程就是程序设计的过程。最终要用比较专业的程序设计图纸来表达程序设计思想，它有别于类似“现场施工”的代码编写。程序设计描述是计算思维培养的重要内容之一。

程序设计描述的方法有很多种，如自然语言、流程图（Flow Chart）、盒图（N-S Chart）、问题分析图（Problem Analysis Diagram，PAD）、伪代码及程序设计语言。无论用哪一种方法表达程序设计思想，所使用的描述符号都应该是确定的、唯一的，不应该引发理解上的二义性（在相同上下文中的同一符号表达出两种含义）。本书使用流程图作为程序设计的描述方法。

流程图的全称为程序流程图，是一种传统的程序设计表示法，是人们对解决问题的方法、思路或算法的一种描述。它利用图形化的符号框来表示各种不同性质的操作，并用流程线将这些操作连接起来。在程序的设计（编码之前）阶段，画流程图，可以帮助我们理清程序的设计思路。流程图包含一套标准框图符号。绘制时，通常按照从上到下、从左到右的顺序来画。除判断框和改进的 for 循环外，其他程序框图只有一个进入点和一个退出点。本小节先引入 4 种图形符号来完成第 1 章的程序设计工作，如表 1-3 所示。

表 1-3　　流程图的基本符号

图形	名称	意义
	起止框	程序开始或结束
	输入输出框	数据的输入输出
	处理框	对数据进行处理
	流程线	表示程序执行的顺序

用上述流程图组件表达 1.3.1 小节中所述功能，如图 1-2 所示。

图 1-2　计算流程图