责任编辑 魏 彬
封面设计 庞鸿设计

计算机类人才培养系列教材

计算机二级Visual FoxPro考点精要
▶ C语言程序设计教程（第2版）
C语言程序设计上机指导与习题（第2版）
数据库原理与SQL Server应用
网页设计与制作（第二版）
大学计算机文化基础（第2版）
计算机应用基础（Windows 10 + Office 2016）（第2版）
计算机应用基础实训教程（Windows 10 + Office 2016）（第2版）
计算机应用基础任务式教程（Windows 10 + Office 2016）

ISBN 978-7-5770-1303-9
9 787577 013039 >
定价：49.80元

辽宁省“十四五”职业教育规划教材
C语言程序设计教程（第2版）
主编◎周 新 王习特 付先平
电子科技大学出版社

辽宁省“十四五”职业教育规划教材

C语言程序设计教程（第2版）

主编◎周 新 王习特 付先平

电子科技大学出版社
University of Electronic Science and Technology of China Press

辽宁省“十四五”职业教育规划教材

C语言程序设计教程（第2版）

主编◎周　新　王习特　付先平

电子科技大学出版社
University of Electronic Science and Technology of China Press
·成都·

图书在版编目（CIP）数据

C语言程序设计教程 / 周新，王习特，付先平主编.
2版. -- 成都 : 成都电子科大出版社，2024. 11.
ISBN 978-7-5770-1303-9

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 20241WX626 号

C语言程序设计教程（第2版）
C YUYAN CHENGXU SHEJI JIAOCHENG（DI-ER BAN）
周　新　王习特　付先平　主编

策划编辑　魏　彬　张　鹏
责任编辑　魏　彬
责任校对　刘　凡
责任印制　梁　硕

出版发行　电子科技大学出版社
　　成都市一环路东一段159号电子信息产业大厦九楼　邮编 610051
主　　页　www.uestcp.com.cn
服务电话　028-83203399
邮购电话　028-83201495

印　　刷　北京荣玉印刷有限公司
成品尺寸　185mm × 260mm
印　　张　19
字　　数　462千字
版　　次　2024 年 11 月第 2 版
印　　次　2024 年 11 月第 1 次印刷
书　　号　ISBN 978-7-5770-1303-9
定　　价　49.80元

编写委员会

主　编：周　新　王习特　付先平

副主编：张　瑾　李　楠　李　鹏

赵恩宇　冶　红　朱　斌

前　言

本书是根据教育部高等学校大学计算机课程教学指导委员会编制的《新时代大学计算机基础课程教学基本要求》及《全国计算机等级考试二级C语言程序设计考试大纲（2024年版）》，专为高等院校学生编写的教材。

本书分为核心篇和拓展篇，核心篇包括第1章至第10章，拓展篇包括模块1和模块2。第1章概述计算机程序设计语言，重点介绍C语言；第2章介绍构成C语言程序的基本语法要素：数据类型、运算符和表达式；第3～5章分别介绍程序设计的三种基本结构，即顺序结构、选择结构和循环结构；第6章介绍数组这一数据类型，使用数组综合运用前5章的知识完成更复杂的任务；第7章详细介绍函数的定义、函数的调用、函数的参数传递、函数返回值等相关知识；第8章介绍一种特殊的、专门用于存放地址的变量——指针；第9章学习结构体、共用体、位域与枚举等构造数据类型；第10章介绍如何将内存中的数据持久化存储到文件中；模块1介绍扩展C语言的面向对象程序设计语言C++的相关基础知识；模块2介绍Visual C++编程基础知识，包括Windows界面编程和基于对话框的应用程序实现。

本书为贯彻落实党的二十大精神，在各个章节结合具体内容加入思政元素，让思政融入教材，让思政走进课堂。第1章通过概述C语言程序设计，让学生了解C语言的发展脉络，引导学生将“青春小我”融入“时代大我”，以“青春之我”践行“强国有我”，激发学生爱国热情，拓宽学生的国际视野，培养家国情怀。第2章通过讲解数据类型的选择，引导学生提高合理利用资源的意识，培养可持续发展理念。第3章通过讲解C语言的顺序结构，培养学生树立踏实进取、一丝不苟的奋发精神。第4章借助选择结构的知识讲解，引导学生关注社会，树立正确的价值观，做出公正的选择和决策，成为具有社会责任感的人才。第5章通过循环结构的知识讲解，引导学生不断迭代优化自身技能，养成坚持不懈、精益求精的品质。第6章通过数组的知识讲解，培养学生在团队协作中共同奋斗的责任感和奉献精神。第7章通过介绍函数相关知识，培养学生的创新意识。第8章通过指针的学习，培养学生树立正确的信息安全意识。第9章通过结构体、共用体、枚举等复杂数据类型的学习，培养学生的团队协作精神。第10章通过文件操作教学，培养学生独立思考和解决实际问题的能力。模块1和模块2引导学生找准自己的定位，不断提升自己各方面的素养，充分发挥自己的主观能动性，为中华民族的和谐、振兴和富强而不懈奋斗。

本书建议课堂教学36～54学时，上机实践24～30学时。本书内容丰富、概念清晰、注重实践、图文并茂、简洁易懂，适合作为高等院校C语言程序设计课程教材，也可作为培训或自学教材。

本书由周新、王习特、付先平担任主编，张瑾、李楠、李鹏、赵恩宇、冶红、朱斌担任副主编。具体分工如下：第4、5章由周新编写；第6、8章由王习特编写；第1、2、3章由付先平、张瑾编写；第7章由李楠编写；第9章由李鹏编写；第10章由赵恩宇编写；模块1、2由冶红、朱斌编写。

本书是《C语言程序设计上机指导与习题（第2版）》（主编：王习特、周新、付先平）的配套教材。为了加强实践学习、提高学生的动手能力，编者将本书中的实验指导部分独立出来，并增补了大量实验内容，放入《C语言程序设计上机指导与习题（第2版）》，旨在通过大量实验加深学生对理论知识的理解，培养学生的编程能力，提高学生使用计算机解决实际问题的能力。因编者水平有限，书中难免有疏漏之处，恳请读者批评指正。

本书配有教学大纲、源代码等丰富的立体化教学资源，有需要者可发邮件至2393867076@qq.com领取。此外，本书还配有丰富的文档资料，学习者可以通过扫描书中的二维码进行学习。

目　录

核心篇

核心篇

在数字时代，C语言作为一种基础编程语言，不仅是操作系统和嵌入式系统的核心，还是学习其他编程语言和从事IT行业工作的重要基础，是量子通信、人工智能、区块链、元宇宙等新技术构建和发展的基石，是新时代发展新质生产力不可缺少的力量。不断拓展的新业态、新技术、新要素、新范式，带来了生产方式的根本性转变。在学习C语言的同时，要拓宽国际视野、培养家国情怀、树立正确的价值观，为中华民族的和谐、振兴、富强而奋斗。

第 1 章 C 语言程序设计概述

知识目标

1. 理解计算机程序设计语言的概念、分类、发展历程和特点。
2. 掌握 C 语言程序的运行原理和基本结构，熟悉 C 语言程序的常见设计风格。
3. 熟悉常见的算法描述工具，如自然语言、流程图、N-S 图和伪代码等。

技能目标

1. 能够独立设计、编写简单的 C 语言程序。
2. 能够利用 Visual C++ 2010 开发环境调试、编译、链接、执行 C 语言程序。
3. 能够使用合适的算法描述方法，清晰地表达程序的逻辑结构和执行流程。

素质目标

1. 培养对程序设计语言多样性和发展趋势的认知，拓展思维广度和深度。
2. 提高代码规范意识，培养良好的职业素养。
3. 培养分析问题和解决问题的能力，锻炼逻辑思维和创新思维。

本章概述

计算机是当代科技发展的引擎，而程序设计语言是推动其不断前行的灵魂。C语言作为坚实的基石，见证了科技的跨越式发展。C语言作为一种高级程序设计语言，不仅展现了其出色的性能，更代表了人类对技术的不懈追求。本章通过讲解计算机程序设计语言的相关概念，使得学生能够掌握理论知识和相关技能。通过深入了解C语言程序的基本结构、运行过程及程序设计算法，学生能够更好地理解信息科技的伦理、社会责任与可持续发展的重要性，将“青春小我”融入“时代大我”，让“青春之我”与“强国有我”同频共振，为科技创新贡献更多智慧。

1.1 计算机程序设计语言概述

计算机系统由硬件系统和软件系统两大部分组成。硬件系统是指由构成计算机的各种机械部件和电子元件组成的设备和装置，是组成计算机系统的物质基础。软件系统是控制、管理计算机各硬件设备工作的所有程序文件和数据文件的总称，是计算机系统的头脑和灵魂。没有软件的计算机是一台“裸机”，什么也不能干，计算机的工作过程就是执行各种程序的过程。程序是指为完成特定功能而编写的指令的集合，这些指令依据既定的逻辑控制计算机的运行。控制计算机运行的所有程序都是用计算机语言编写的。

随着计算机技术的迅速发展，计算机程序设计语言经历了从机器语言到汇编语言再到高级语言的发展历程。程序设计方法也伴随着计算机硬件技术的提高而不断发展，可分为三个阶段，即面向计算机的程序设计、面向过程的程序设计和面向对象的程序设计。

高级语言是一种面向人类的编程语言，它使用更接近自然语言和数学表达式的语法来编写程序。使用高级语言编写的程序与计算机的硬件结构及指令系统无关，具有更强的表达能力，可方便地表示数据的运算和程序的控制结构，能更好地描述各种算法，而且易于学习掌握。近几十年来，为解决不同的实际问题出现了几百种高级语言，目前较流行的有Java、C、C++、Python、C#、Visual Basic .NET、JavaScript、PHP、Perl、Ruby等。

C语言是1972年由美国的Dennis Ritchie（丹尼斯·里奇）设计发明的，它由早期的编程语言BCPL（Basic Combined Programming Language）发展演变而来。早期的C语言主要用于UNIX系统。由于C语言的强大功能和各方面的优点，其逐渐为人们所认识，到了20世纪80年代，C语言开始进入其他操作系统，并很快在各类大、中、小和微型计算机上得到了广泛的应用，成为当代非常优秀的程序设计语言之一。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，美国国家标准学会（ANSI）于1989年为C语言制定了一套ANSI标准，成为现行C语言标准的基础。

C语言是一种面向过程的结构化程序设计语言，它层次清晰，便于模块化组织程序，易于调试和维护。C语言的表现能力和处理能力极强，它不仅具有丰富的运算符和数据类型，便于实现各类复杂的数据结构，还可以直接访问内存的物理地址，进行位（bit）一级的操作，因此，C语言集高级语言和低级语言的功能于一体，既可用于系统软件的开发，也适合应用软件的开发。此外，C语言还具有语言简练、源程序短、效率高、可移植性强等优点。

面向过程的程序设计必须按照算法的实现过程逐条编写程序，通知计算机一步一步怎么做。20世纪80年代后，面向对象的程序设计概念日益普及。所谓面向对象，是通过类和对象把程序涉及的数据结构和对它施行的操作有机地组织成模块，对数据及数据处理细节进行最大限度的封装，从而使开发出来的软件易于重用、修改、测试、维护和扩充，C++就是在C语言的基础上增加了面向对象的思想发展而来的。为了满足网页开发的需要，1994年又出现了Java语言，Java语言脱胎于C++，不仅支持面向对象的程序设计思想，且

具有与软硬件平台无关的特点，适合进行网络开发。2000年，微软推出的Microsoft Visual Studio .NET是一个具有公共语言子集的开发平台，实现了多种语言及其类库的无缝集成，C#是专为这一平台推出的全新语言。C#也派生于C和C++，具有语法简洁、面向对象、与Web紧密结合、卓越的安全性能、灵活性和兼容性俱佳等特点，成为.NET平台一流的网络编程工具。

从面向过程的C到面向对象的C++，再到Windows平台上可视化的C++开发工具Visual C++，最后到网络平台上面向对象的Java和C#，各种适应新的编程环境的程序设计语言层出不穷，但很多语言都是基于C语言发展而来的，而且开发系统程序（如操作系统和嵌入式系统）和底层应用程序（如接口程序、通信和自动控制等）时的首选语言仍然非C语言莫属。C语言在程序设计语言中可谓“常青树”，其在全球编程语言排名中的坚实地位反映了C语言的重要性。学习C语言既是学习编程技能的起点，更是培养逻辑思维能力和处理实际问题能力的基石。

1.2 C语言程序的结构特点、运行及设计风格

本节主要讲解C语言程序的结构特点，并通过Visual C++ 2010 Express集成开发环境介绍C语言程序的运行方法。

1.2.1 C语言程序的结构特点

下面通过几个简单的C语言程序学习其结构特点。

【例 1-1】在显示器上分行显示“Welcome you.”和“First C program.”。

程序如下所示。

```
#include <stdio.h>
void main()                                    /* 函数名 */
{                                              /* 函数体放在一对花括号中 */
    printf("Welcome you.\nFirst C program.\n");  /* 输出内容 */
}
```

运行结果如图1-1所示。

```
Welcome you.
First C program.
```

图1-1 例1-1运行结果

本例是一个最简单的C语言程序。

第1行是编译预处理命令include。当程序中要使用C系统提供的标准函数或其他文件时，一定要用include命令包含函数对应的文件，以实现在编译前将对应文件嵌入该处成为源程序的一部分。程序如果有输出语句，include命令中就必须包含标准输入/输出头文件stdio.h。

第2行是函数名main()。一个C语言程序由若干个函数组成，必须有一个用main()命名的主函数，程序从主函数开始运行，也结束于主函数。函数都要有类型说明，放在函数名前，void表示空类型，没有返回值。

第3～5行为函数体。函数体是程序的执行部分，由若干条语句构成，写在一对花括号“{ }”内，本例的函数体只有一条输出语句。C语言的输入与输出操作都是由系统函数完成的，printf()为标准输出函数。printf()函数的输出内容取决于第一对双引号" "内的字符串，一般原样输出该字符串，仅在出现以“%”开头的输出格式时有变化。“\n”是换行符。

在一对“/*”和“*/”内的内容为注释。程序编译时，碰到“/*”将不对后面的内容进行编译，直到出现“*/”为止。注释内容是对程序的解释，目的是提高程序的可读性，而不是执行代码。

【例 1-2】求 −5 的绝对值。

程序如下所示。

```
#include <stdio.h>
#include <math.h>
void main()
{
    int x,y;                              /* 定义程序中用到的变量 x①和 y*/
    x=-5;                                 /* 将初值 -5 赋给变量 x*/
    y=abs(x);                             /* 求 -5 的绝对值，将结果赋给变量 y*/
    printf("-5 的绝对值为：%d\n",y);          /* 输出变量 y 中的结果 */
}
```

运行结果如图1-2所示。

```
-5的绝对值为：5
```

图1-2　例1-2运行结果

本例使用了系统函数abs()求绝对值，因此，要包含对应的头文件math.h。%d是输出格式，表示输出一个十进制整数，输出时其出现的位置将代入逗号后面对应变量的值。本例的%d表示代入变量y的值5，并以十进制整数形式显示。


C++和C的关联

【例 1-3】从键盘输入任意 3 个整数，输出其中最大的数。

本例可以用两种形式编写程序代码。

程序1如下所示。

```
#include <stdio.h>
void main( )
{
    int x,y,z,max;
    scanf("%d%d%d",&x,&y,&z);                    /* 运行时，从键盘输入 3 个整数 */
```

①为与代码格式一致，本书涉及的变量均采用正体表示。

```
    if(x>y) max=x;else max=y;                    /* 比较 x 和 y，将大者赋给 max*/
    if(max<z) max=z;                             /* 比较 max 和 z，将大者赋给 max*/
    printf("%d,%d,%d 中的最大值为：%d\n",x,y,z,max);   /* 输出结果 */
}
```

运行结果如图1-3所示。

```
45 -34 167?
45,-34,167中的最大值为：167
```

图1-3 例1-3程序1运行结果

scanf()是系统提供的标准输入函数，该函数在程序运行时可为逗号后的各变量赋值，它要求变量名前必须加取地址符“&”。用户每次运行程序，可根据实际情况随机灵活地输入不同的变量值。本例程序运行时，从键盘输入3个整数，数与数之间用空格隔开，然后按回车键，系统顺次将其赋给变量x、y、z。

程序2如下所示。

```
#include <stdio.h>
void main()
{
    int max(int x, int y);              /* 对自定义函数 max()进行声明 */
    int a,b,c,d;
    scanf("%d,%d,%d",&a,&b,&c);
    d=max(a,b);                         /* 调用函数 max()找出 a,b 中的大者赋给 d*/
    d=max(d,c);                         /* 再次调用函数 max()找出 d,c 中的大者赋给 d*/
    printf("max=%d\n",d);
}
int max(int x, int y)                   /* 自定义函数 max()，从两个数中找出大者 */
{
    int m;
    if(x>y) m=x;                        /* 比较 x 和 y，将大者赋给 m*/
    else m=y;
    return(m);                          /* 返回找到的大数 */
}
```

运行结果如图1-4所示。

```
45,-34,167
max=167
```

图1-4 例1-3程序2运行结果

程序2中包括两个函数main()和max()。max()函数的功能是找到任意两个数中的大数，并将结果返回给调用它的函数。本例通过在主函数中两次调用max()函数，实现从3个数中找到最大数的功能。把一些功能相对独立的程序段编成一个函数，可避免使用重复代码，降低了代码量和复杂度，并可以反复调用，C语言就是通过函数实现了程序的模块化。

由上面的例题可以得出以下几点结论。

（1）C语言程序是由函数构成的。一个C语言程序不论由多少个文件组成、包含多少个函数，都有且仅有一个main()函数，即主函数。一个C语言程序总是从main()函数开始执行，而不论main()函数在程序中的什么位置，程序的执行也结束于主函数，其他函数通过函数调用执行。

（2）每个函数包括函数名和函数体两大部分。除主函数外，其他函数按C语言命名规则自主命名。函数体包含在一对花括号“{}”中，由若干条语句构成。

（3）C语言程序书写格式自由，可在一行内写多条语句，也可将一条语句写在多行上（上行末加续行符“\”）。每条语句必须以分号“;”结尾。

（4）程序代码一般用小写字母书写，C语言严格区分大小写。

（5）注释部分仅增加程序的可读性，是给用户看的，不会被编译，可以出现在程序的任意地方。写在双边注释符“/*……*/”中的注释内容可以占部分行、单行、多行；写在单边注释符“//”后的注释内容仅占一行，至行末为止。

（6）源程序中的预处理命令通常放在源程序的最前面，预处理命令之后不能加分号。

（7）函数体内的语句一般按四大功能顺次排列，即首先对变量与函数进行声明，然后为变量赋初值，再进行数据计算与处理，最后输出结果。

1.2.2 C语言程序的运行

使用高级语言编写的程序统称为源程序，源程序不能为计算机所识别。一个C语言程序需要经过编辑、编译、连接和运行四个步骤，才能得到程序结果，如图1-5所示。

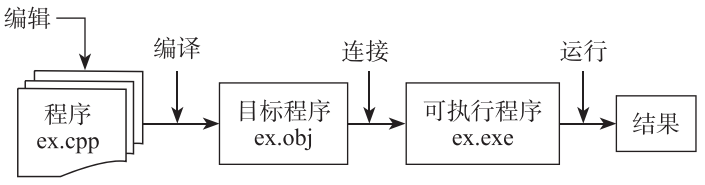

图1-5 C语言程序运行步骤

（1）编辑。通过纯文本编辑软件输入和编辑源代码，并保存成源程序文件。在Visual C++ 2010 Express环境下，源程序文件的扩展名为.c或.cpp，如ex.cpp。

（2）编译。通过编译系统将源程序代码编译成计算机能直接识别的二进制形式，即目标程序。在编译过程中，系统将自动检查源程序中的句法和语法错误。如果发现错误，则会报告错误类型及在程序中的位置，帮助用户修改错误；如果未发现错误，则自动生成对应的目标程序。目标程序是与源程序同主名的.obj文件，如ex.obj。

（3）连接。将形成的目标程序、库函数或其他目标程序连接装配成一个可执行程序。大多数的编译系统都允许使用一个命令同时完成编译和连接操作。可执行程序是与源程序同主名的.exe文件，如ex.exe。

（4）运行。运行可执行程序，得到程序结果。

1.2.3 使用Visual C++ 2010学习版集成开发环境运行C语言程序

集成开发环境（Integrated Development Environment，IDE）是集程序的编辑、编译、连接及运行等功能于一体的程序开发软件，利用IDE可以大大提高程序开发的效率。目前，用于C语言程序开发的集成环境有Turbo C、Borland C、Microsoft C和Microsoft Visual C++等。Microsoft Visual C++ 2010 学习版（简称VC++ 2010 Express）是Microsoft公司2010年推出的针对学生和教育机构设计的编程软件，不仅可以开发C++程序，也可以开发C程序。当文件以“.c”作为后缀名时，VC++ 2010 Express会用C语言语法编译处理文件；以“.cpp”作为后缀名的文件表明该文件是C++源文件（cpp是C Plus Plus的简写），VC++ 2010 Express会用C++语言语法编译处理该文件。

本书内容均针对VC++ 2010 Express编写。

在VC++ 2010 Express中，一个C应用程序被称为一个项目或工程（Project），它是由应用程序中所需要的所有文件组成的一个有机整体，一般包括源文件、头文件和资源文件等。项目能自动对其包含的文件进行分类和管理，从而大大减轻了程序员的负担。项目被置于解决方案（Solution）的管理之下。一个解决方案可以包含一个或多个项目，甚至是不同类型的项目，这些项目之间相互独立，但共用一个解决方案的环境设置。

1.在VC++ 2010 Express中运行一个C程序的步骤

（1）启动VC++ 2010 Express。选择“开始”→“Microsoft Visual C++ 2010学习版”命令。打开如图1-6所示的起始页。

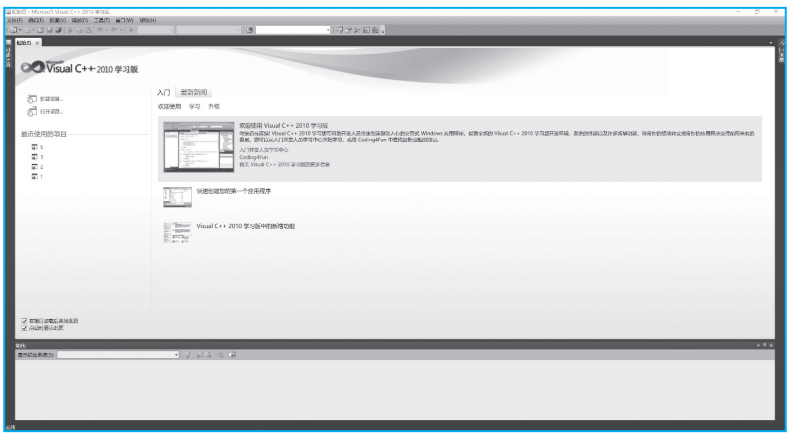

图1-6 Visual C++ 2010 Express起始页

（2）创建项目工作区。在VC++ 2010 Express起始页窗口中，选择“文件”→“新建”→“项目”命令，弹出如图1-7所示的“新建项目”对话框，选择“Visual C++”选项卡中的“Win32 控制台应用程序”选项，指定项目名称、项目存放路径，单击“确定”按钮，弹出“Win32 应用程序向导”对话框，单击“下一步”按钮，出现如图1-8所示的“应用程序设置”面板：“应用程序类型”选择“控制台应用程序”，“附加选项”选择“空项目”，单击“完成”按钮，创建项目和项目工作区，并显示如图1-9所示的“解决

方案资源管理器”界面。

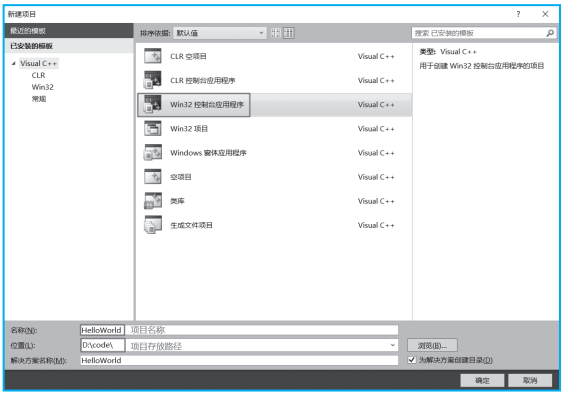

图1-7　“新建项目”对话框

图1-8　Win32应用程序向导“应用程序设置”面板

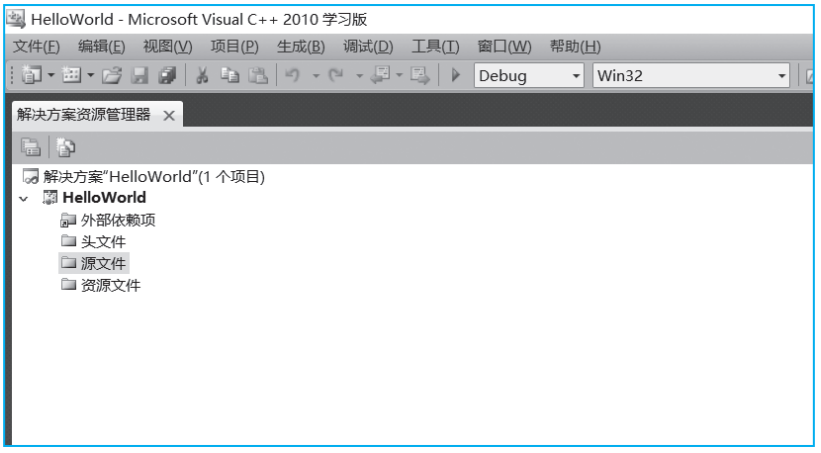

图1-9　“解决方案资源管理器”界面

提示

在“新建项目”对话框中可以创建多种类型的工程，作为初学者，可以选择新建“Win32 控制台应用程序”，从类型名可以看出创建的工程依赖控制台（命令提示符）工作在32位的Windows环境中。通过调用系统的控制台，可以进行数据的输入和显示等操作，使得编程人员可以和计算机进行交互。尽管运行控制台程序出现的是DOS窗口，没有图形用户界面友好美观，但是控制台应用程序可以避免初学者过多地关注界面开发，降低入门难度，使其将主要精力放在学习编程逻辑等方面。

HelloWorld项目创建完成后，解决方案名默认同为HelloWorld，在D:\code文件夹下会添加相应的解决方案主文件夹“D:\code\HelloWorld”和HelloWorld项目文件夹“D:\code\HelloWorld\HelloWorld”。

D:\code\HelloWorld文件夹下的HelloWorld.sln是解决方案文件，它包含了Test解决方案的配置信息、项目关联及其他设置。

D:\code\HelloWorld\HelloWorld文件夹是HelloWorld项目的文件夹，它包含了HelloWorld项目的源代码文件、头文件、资源文件等，如D:\code\Test\HelloWorld\HelloWorld.vcxproj是HelloWorld项目的项目文件，包含了项目的配置信息、编译选项、链接选项等信息。

（3）创建源文件。在如图1-9所示界面的解决方案下选择“源文件”，单击鼠标右键，选择“添加”→“新建项”命令，弹出如图1-10所示的“添加新项”对话框，选择“C++ 文件(.cpp)”，指定文件名为“hello.c”，单击“添加”按钮，创建源程序文件“hello.c”。

提示

①添加“C++ 文件(.cpp)”默认会创建.cpp文件，若要创建C文件，请在文件名后手动添加“.c”后缀，限定该文件为C语言文件。

②“hello.c”文件创建完成后，D:\code\HelloWorld\HelloWorld文件夹下将包含源文件“hello.c”。

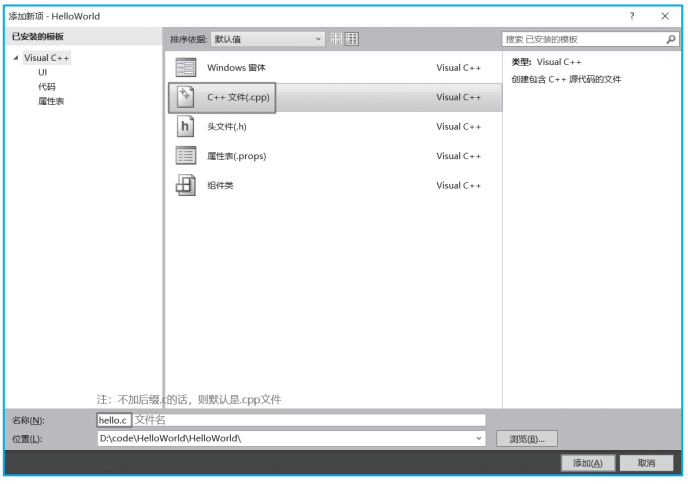

图1-10 “添加新项”对话框

（4）编辑源程序文件。在如图1-11所示的编辑窗口中输入、编辑源代码，并选择“保存”命令。

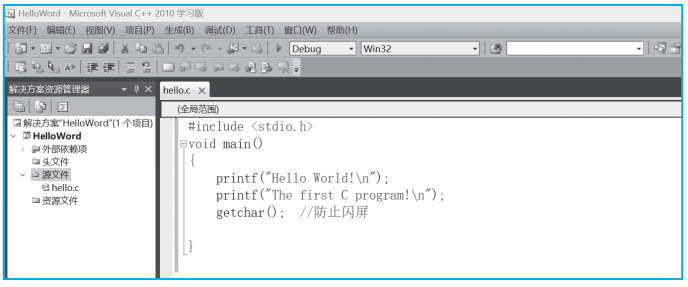

图1-11 编辑源程序文件

（5）编译。选择“生成”→“生成解决方案”命令编译并链接当前解决方案中的所有项目，编译结果会显示在下面的输出窗口中。如果编译有错误，必须修改错误重新编译，直至编译通过。

提示

①执行编译操作后，会分别在解决方案主文件夹下和项目文件夹下生成Debug文件夹。

②D:\code\HelloWorld\Debug文件夹通常用于存放整个解决方案的生成输出，包括所有项目的可执行文件、目标文件、库文件等。

③D:\code\HelloWorld\HelloWorld\Debug文件夹通常用于存放特定项目HelloWorld的生成输出，包括该项目的可执行文件、目标文件、中间文件等，与其他项目无关。

（6）运行。选择“调试”→“开始调试”命令，可执行文件将在控制台或Windows窗口中启动，运行结果如图1-12所示。

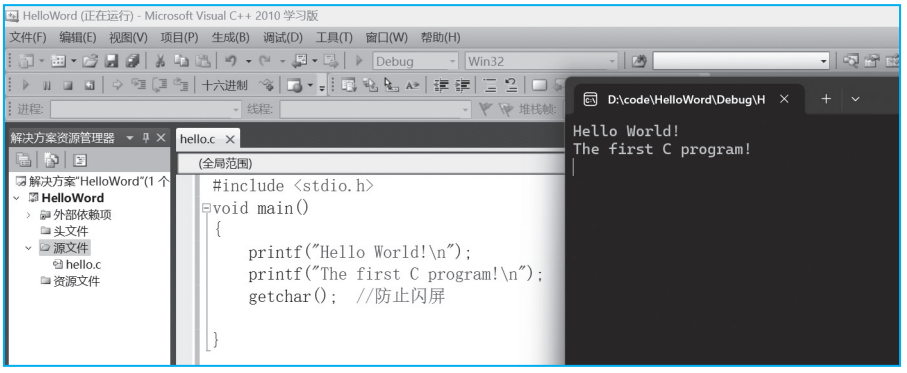

图1-12　文件运行结果

2. C程序调试错误

C程序调试时，可能会出现以下类型的错误。

（1）编译错误。编译错误是编译系统在编译时发现的错误，也叫语法错误。编译错误分为错误（Error）和警告（Warning）。Error通常是由关键字拼写错误、不可识别的标识符、语句末尾缺少分号、不配对的圆括号或花括号等引起的错误，必须修改，否则编译无法通过。Warning通常是由数据类型转换不当、不恰当的类型说明符等引起的错误，编译系统无法确定是否真正错误，便以警告的形式告知用户。一般来说，警告并不影响后续的编译、连接和运行，但有的警告如果不加以纠正，可能导致严重的运行错误。因此，对警告应分析原因，尽量予以消除。

有时，编译后会出现大片错误，这种情况往往是一种假象，实际可能是一两个错误连带引起的。因此，在调试程序时，应当首先纠正最前面的或容易发现的错误，然后重新编译，逐步减少错误量，而不是修改所有错误后再重新编译。

（2）运行错误。运行错误是在运行过程中发现的错误。比如对负数开平方根、数据

发生溢出等。

（3）逻辑错误。逻辑错误是使用了错误的算法、公式、数据等引起的错误。对于这种错误系统并不报错，必须由编程人员自行发现和纠正。

1.2.4　程序设计风格

程序设计风格是指编辑程序时所表现出来的特点、习惯和逻辑思路等。程序设计风格会深刻影响软件的质量和可维护性，良好的程序设计风格可以使程序结构清晰合理，使程序代码便于维护。培养良好的程序设计风格对于初学者来说非常重要。

程序设计的风格总体来说应强调简单和清晰，程序必须是可理解的，即做到“清晰第一，效率第二”。为提高程序的可阅读性，形成良好的程序设计风格，应注意和考虑以下几个因素。

1.源程序文档化

（1）标识符按意取名。具有实际意义的命名有助于对程序功能的理解。

（2）程序应加注释。正确的注释能够帮助理解程序。注释分为序言性注释和功能性注释两类。序言性注释通常置于每个模块的起始部分，给出程序的整体说明，主要内容有：程序标题、程序功能说明、主要算法、接口说明、程序位置、开发简历、程序设计者、复阅者、复审日期、修改日期等。功能性注释一般嵌在源程序内部，说明程序段或语句的功能及数据的状态。

（3）在程序中利用空格、空行、缩进等技巧可以使程序层次清晰，结构一目了然。

2.数据说明方法

为使数据定义更易于理解和维护，一般应注意以下几点。

（1）数据说明顺序规范化。数据说明的次序固定，可以使数据的属性更易于查找，从而有利于测试、纠错与维护。例如，按以下顺序进行数据说明：常量说明、类型说明、全程量说明、局部量说明。

（2）变量说明有序化。一条语句说明多个变量时，各变量名按字母顺序排列。

（3）对复杂的数据结构加注释，说明在程序实现时的特点。

3.语句的结构

语句结构应简单直接，不能为了追求效率而使代码复杂化。例如，为了便于阅读和理解，避免一行多条语句；不同层次的语句采用不同的缩进形式，使程序的逻辑结构和功能特征更加清晰；要避免复杂的判定条件和多重的循环嵌套；表达式中使用括号以提高运算次序的清晰度；等等。

4.输入和输出

在编写输入和输出程序时，应考虑以下六个原则。

（1）输入操作步骤和输入格式尽量简单。

（2）应检查输入数据的合法性、有效性，报告必要的输入状态信息及错误信息。

（3）输入一批数据时，使用数据或文件结束标志，而不要用计数来控制。

（4）交互式输入时，提供可用的选择和边界值。

（5）当程序设计语言有严格的格式要求时，应保持输入格式的一致性。

（6）输出数据表格化、图形化。

5.效率

程序的效率是指在完成任务时，以最小的时间消耗和资源占用（如存储、计算能力）实现最优性能，同时兼顾代码的可维护性与可扩展性。对效率的追求要明确以下几点。

（1）效率是一个性能要求，目标在需求分析时给出。

（2）追求效率应建立在不损害程序可读性或可靠性的基础上，要先使程序正确、清晰，再提高程序效率。

（3）提高程序效率的根本途径在于选择良好的设计方法、良好的数据结构算法，而不是靠编程时对程序语句进行调整。

使用C语言编写程序时除了必须遵守C语言的语法要求以外，还应遵守以下一系列的规约，这些规约不是强制性的，而是约定俗成的编程风格。

（1）所有代码必须采用ANSI 标准，保持代码简单清晰，避免使用复杂语句。

（2）程序结构清晰易懂，单个函数的程序行数建议不超过100行，代码行长度尽量不超过80个字符。每个语句行只做一件事情。

（3）尽量使用标准库函数和公共函数。程序根据要达成的目标设计，力求代码简单，避免出现冗余的代码。

（4）表达式中多使用括号可避免二义性。

（5）要有足够多的注释充分解释源代码。每个源程序文件都要有文件头说明，每个函数也都要有函数头说明。在适当处加空行或空格也是一种特殊的注释，可提高程序可读性。

（6）利用缩进来显示程序的逻辑结构，缩进量一致以“Tab”键为单位。

（7）循环、分支的层次不超过5层，避免出现不必要的分支，尽量避免使用goto语句。

（8）所有变量在调用前必须赋值。不要进行浮点数的比较，如10.0*0.1==1.0的比较不可靠。

1.3 程序设计算法及其描述方法

计算机程序是指为完成特定功能而编写的指令的集合，程序设计就是用计算机语言对所要解决的问题进行完整而准确的描述，但是程序设计并不是简单地编写程序代码的过程。程序可以表示为

程序 = 算法 + 数据结构 + 程序设计方法 + 语言工具和环境

程序设计方法是编写程序的指导思想，决定了以什么样的方式组织编写程序，也决定了一个程序的成功与否；语言工具和环境是编写程序的工具，负责制造程序；而算法则是

灵魂，是解决问题（处理数据）的方法和步骤；数据结构是算法加工的对象。

一个完整的程序设计过程大体可分为以下四个阶段。

（1）分析问题。充分研究与分析要解决的问题，明确问题的需求，即应用程序要实现什么功能、达到什么目的。

（2）设计算法。根据问题需求，设计具体算法，描述算法的具体实现步骤。解决同一问题可能有多种算法，应根据实际情况选择合适的算法。

（3）编写程序。根据选定的算法，选择适当的程序开发语言与环境，再按照算法编写程序代码。

（4）程序验证。进行程序的正确性证明、测试与调试。应设计各种情况下的数据验证，保证每一个条件的成立与不成立都被测试过，以证明程序正确无误。

可见，编写程序实际上就是在实现某种算法，算法在计算机程序设计中是非常重要的。

1.3.1 算法的概念及特征

算法（Algorithm）就是解决问题的步骤和方法，它能够对一定规范的输入，在有限时间内获得所要求的输出。解决一个问题的过程就是实现一种算法的过程。一个解题步骤、工作计划、生产流程、音乐乐谱等都可被称为“算法”。

如果一种算法有缺陷，或不适合某个问题，那么即使执行了这种算法也不会解决该问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。一种算法的优劣可以用空间复杂度与时间复杂度来衡量。

一种算法应该具有以下五个重要的特征。

（1）有穷性。一种算法应包括有限的操作步骤，能在执行有穷的操作步骤之后结束。

（2）确定性。算法的计算规则及相应的计算步骤必须是唯一确定的，既不能含糊其词，也不能有二义性。

（3）输入。一种算法有0个或多个输入，以刻画运算对象的初始情况。

（4）输出。一种算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

（5）可行性。算法中的每一个步骤都是可以在有限的时间内完成的基本操作，并能得到确定的结果。

任何简单或复杂的算法都是由基本功能操作和控制结构这两个要素组成的。计算机的基本功能操作包括以下四个方面。

（1）逻辑运算。与、或、非等。

（2）算术运算。加、减、乘、除等。

（3）数据比较。大于、小于、等于、不等于、大于等于、小于等于等。

（4）数据传送。输入、输出、赋值。

控制结构是指各操作之间的执行顺序。按照结构化程序设计的观点，任何复杂的算法都可以由顺序结构、选择（分支）结构和循环结构这三种基本结构来实现。这种结构化程

序结构清晰、易于进行正确性验证和纠错，提高了程序的可靠性，保证了程序的质量。

1.3.2 算法的描述工具

为了将算法正确地表示出来，需要使用算法描述工具。算法有多种描述工具，可概括为两大类：文字描述（语言描述）和图形描述。下面介绍几种常见的算法描述工具。

（1）自然语言：最简单的一种算法描述工具，如汉语、英语等。其优点是通俗易懂、易于掌握，但也有烦琐、容易产生歧义等缺点。

（2）流程图：使用不同的图框表示不同类型的操作，用带箭头的线表示操作的执行顺序。相对于自然语言来说，流程图更简洁、直观。

流程图常用图符如图1-13所示。三种基本控制结构的流程图如图1-14所示。

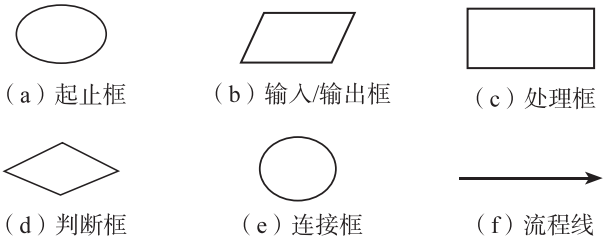

图1-13　流程图常用图符

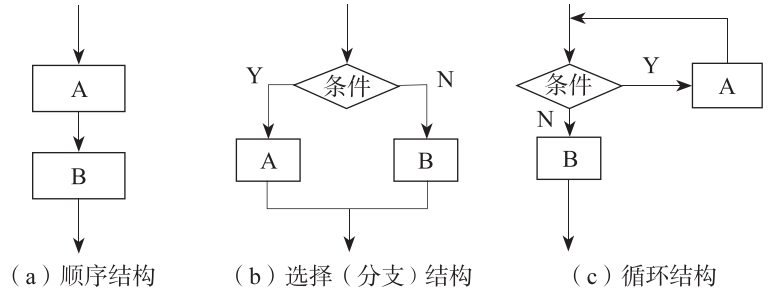

图1-14　三种基本控制结构的流程图

（3）N-S图：描述算法的另一种常见工具，主要省掉了流程图中的流程线，使得图形更紧凑。N-S图的基本结构描述形式如图1-15所示。N-S图的优点是能直观地用图形表示算法，去掉了导致程序非结构化的流程线；缺点是修改不方便。

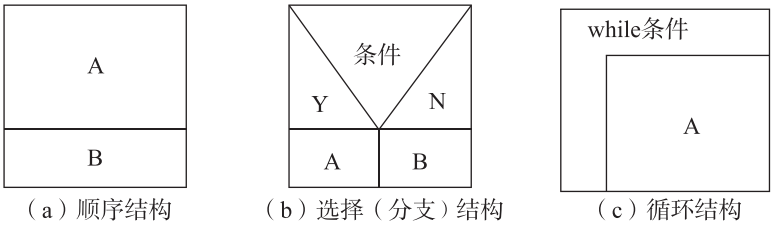

图1-15　三种基本控制结构的N-S图

（4）高级语言：用某种高级语言去描述算法，其好处是可以直接在计算机上运行，但受计算机语言严格的格式要求和语法限制，对用户要求较高。

（5）伪代码：介于自然语言与计算机语言之间的一种文字和符号相结合的算法描述工具，形式上跟计算机语言比较接近，但没有严格的语法规则限制。通常借助某种高级语言的控制结构，中间的操作既可以用自然语言又可以用程序设计语言描述，这样，既避免了严格的语法规则限制，又比较容易转换为程序代码。

在为具体问题设计算法时，选用哪种算法描述工具并不重要，重要的是一定要把算法描述得简洁、正确，不会产生理解上的“歧义”。

例如，计算如下分段函数：

$$y=\begin{cases} x+1 & x>0 \\ x-100 & x=0 \\ x\times 20 & x<0 \end{cases}$$

①使用算法语言描述如下。

第一步：输入变量x的值。

第二步：判断x是否大于0，若大于0，则y＝x+1，然后转第五步；否则进入下一步（第三步）。

第三步：判断x是否等于0，若等于0，则y＝x−100，然后转第五步；否则进入下一步（第四步）。

第四步：y＝x×20（如果第二、三步的条件不成立，则进入第四步时，x肯定小于0）。

第五步：输出变量y的值。

第六步：结束程序执行。

②使用流程图表示，如图1-16所示。

③使用N−S图表示，如图1-17所示。

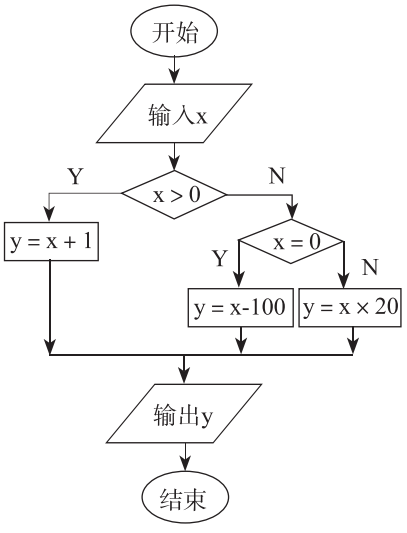

图1-16　分段函数算法的流程图

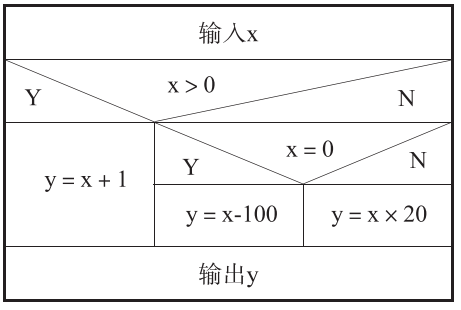

图1-17　分段函数算法的N−S图

④用高级语言C语言来描述的算法如下所示。

```
int x,y;
scanf("%d",&x);
if(x>0) y=x+1;
else if(x==0) y=x-100;
else y=x*20;
printf("x=%d,y=%d",x,y) ;
```

拓展练习

一、简答题

1.写出结构化程序设计的三种基本结构。

2.运行一个C语言程序必须经过哪四个步骤？

3.分别用流程图和N-S图描述算法：判断一个数是否能被3和5整除。

二、程序改错

指出并修改下面程序中的错误。

```
include studio.h
main{}
/* this program prints the number of weeks in a year. /*
(
    int s
    s:=52;
    print(There are s weeks in a year");
```

拓展练习1

第 2 章 数据类型、运算符和表达式

知识目标

1. 理解标识符、常量和变量的概念和区别，以及它们在 C 语言中的作用和用法。
2. 理解 C 语言整型数据、实型数据、字符型数据三种基本数据类型的特点和分类。
3. 理解运算符的概念和分类，包括算术运算符、关系运算符、逻辑运算符和位运算符等。
4. 掌握算术、赋值、逗号运算符的优先级和结合性规则，以及表达式的构建和求值方法。

知识目标

1. 能够在合适的应用场景中熟练声明和使用各种数据类型的常量和变量。
2. 能够正确选择合适的数据类型和运算符，设计并实现简单的表达式和语句。
3. 能够利用运算符的优先级和结合性规则，正确进行表达式计算，实现程序中的基本功能。

素质目标

1. 深入理解各种数据类型和运算符，提高程序设计的准确性和效率。
2. 提高对程序设计规范和代码质量的重视，培养良好的编程习惯和代码规范意识。
3. 强调在编程过程中的创新思维和问题解决能力，勇于探索和尝试新的思路和方法。

本章概述

程序设计语言中的基本元素，如数据类型、常量、变量、运算符、表达式等，不仅仅是代码的构建材料，更是反映社会需求和科技智慧的媒介。数据类型的选择直接关系到程序的性能和效率，选择合适的数据类型不仅是对计算机资源的合理利用，更是对环境的科学管理。本章将系统学习常量、变量、运算符、表达式等基本元素。

2.1 C语言的数据类型

程序的执行过程实际上是对数据进行处理，得到正确结果的过程。而这些需要处理的数据以及处理数据过程中产生的中间数据往往需要保存在内存中的某段存储单元内。对这些存储单元，要求“先定义，后使用”。在定义中，规定了该量的名称、数据类型、存储类型及作用域。系统会为不同数据类型的量分配不同大小的存储空间，按名称使用它们，并通过存储类型及作用域进一步规定了该量占据存储空间的时间与范围。

在C语言中，数据类型可分为基本类型、构造类型、指针类型、空类型四大类，如图2-1所示。

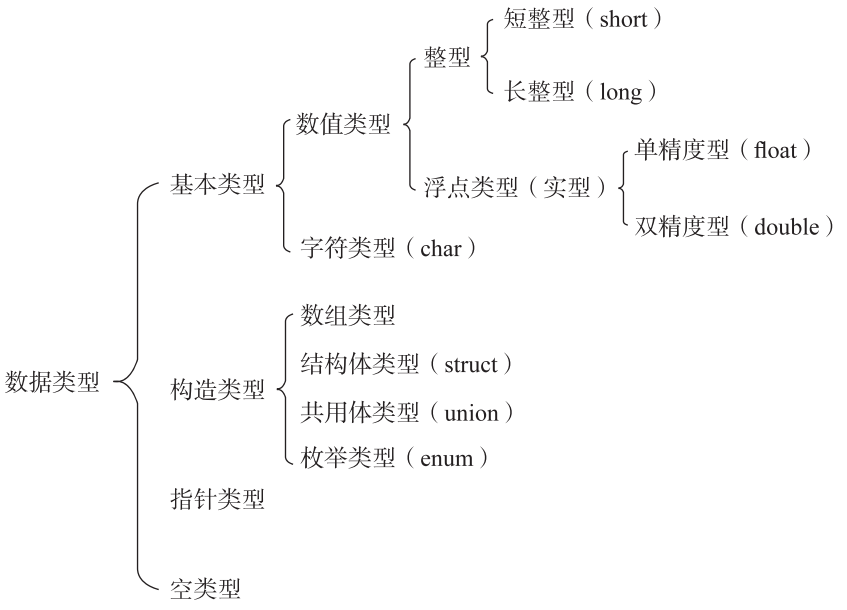

图2-1　数据类型分类

（1）基本类型。基本类型即值不可再分的数据类型，包括整型、浮点类型（实型）、字符类型等。一个基本类型的值就是单个数据。

（2）构造类型。构造类型是由已知的基本类型通过一定的构造方法构造出来的类型，包括数组、结构体、共用体、枚举等。一个构造类型的值可以分解成若干个“成员”或“元素”，每个“成员”都是一个基本数据类型或构造数据类型。

（3）指针类型。指针是存储单元地址的形象表示，指针类型的值用来表示某个数据在内存中的地址。通过指针访问内存取得数据，访问效率更高，可以直接操作硬件。

（4）空类型。空类型又称无值类型，通常用来描述函数无返回值。调用后不需要向调用者返回函数值的函数可以定义为“空类型”，其类型说明符为void。

本书的前5章只涉及基本数据类型，构造数据类型将在第6章和第9章中详细介绍，指针类型将在第8章中详细介绍。

2.2 标识符、常量和变量

对于基本数据类型量，按其取值是否可改变分为常量和变量两种。在程序执行过程中，其值不发生改变的量称为常量，其值可以改变的量称为变量。它们可与数据类型结合起来分类。例如，可分为整型常量、整型变量、浮点型常量、浮点型变量、字符型常量、字符型变量等。

在C语言程序中，直接常量是可以不经说明而直接引用的，而变量则必须先定义后使用。定义变量规定了变量的名称及类型，编译时不仅能为其分配相应的存储单元，还能够检查出变量名是否正确使用、对该变量的运算是否合法等。

2.2.1 标识符

在C语言中，有许多需要命名的对象，如符号常量名、变量名、数组名、函数名、文件名、类型名等。标识符就是名字，用以区分不同的对象。

C语言规定，标识符只能由英文字母、下划线、数字三种字符组成，且只能以字母和下划线开头。以下划线开头的标识符往往是系统变量或系统函数，由大写字母构成的标识符往往是符号常量，用户自定义的标识符应尽量避免采用上述形式。

在选择标识符时，应注意以下规则。

（1）严格区分大小写，即大小写字母表示不同的含义。例如，sum和Sum代表不同的标识符。

（2）标识符长度最好不超过8个字符。因为有的C语言编译系统只识别8个字符，即前8个字符有效。例如，变量名information1和information2的前8个字符相同，C语言编译系统可能会认为是同一个变量。

（3）尽量做到“见名知意”，即选择有含义的英文单词或其缩写词作为标识符，以增强程序的可读性，如用count、score、total、age分别命名存储数目、分数、总计、年龄值的变量。

（4）标识符不能使用C语言的关键字。这些关键字已被C语言系统使用，用户不能自定义使用。

由ANSI标准定义的关键字共有32个，分别为auto、double、int、struct、break、else、long、switch、case、enum、register、typedef、char、extern、return、union、const、float、short、unsigned、continue、for、signed、void、default、goto、sizeof、volatile、do、if 、while、static。


C语言中的关键字

下面列举几个正确和不正确的标识符。

正确：price5、_decision、key_board、FLOAT。

不正确：5price、bomb?、key.board、float。

2.2.2 直接常量和符号常量

在C语言中，常量分为直接常量和符号常量两种。

1.直接常量

直接常量也叫字面常量，包括如下几种。

（1）整型常量：如0、23、−3。

（2）浮点型常量：如1.2、−3.4。

（3）字符常量：如'a'、'1'、'\n'、'\101'。

（4）字符串常量：如"I am a student. "、"2220131234"。

2.符号常量

可以用一个标识符代表一个常量，该标识符称为符号常量。

符号常量在使用之前必须先定义，有以下两种定义形式。

形式一：

```
#define  标识符  字符串
```

这是一条预处理命令——宏命令，其功能是把标识符定义为其后的字符串。例如：

```
#define PI 3.14159
```

经过上面的定义，以后程序中所有出现符号常量PI的地方，在预处理阶段都会被3.14159取代。

形式二：

```
const  类型  标识符 = 常数
```

定义指定类型的标识符（符号常量）的值，且该值在程序运行过程中不可改变，例如：

```
const double PI=3.14159;
```

习惯上，符号常量的标识符采用大写字母，以区别于一般变量。使用符号常量可以将复杂的长字符串用简洁的标识符代替，而且能够做到“一改全改”。

【例 2-1】符号常量的使用。

程序如下所示。

```
#define PAI 3.14
#include<stdio.h>
void main()
{
```

```
    float r1=3,r2=4,s1,s2;
    s1=r1*r1*PAI;
    s2=r2*r2*PAI;
    printf("S1=%f\nS2=%.2f\n",s1,s2);
}
```

运行结果如图2-2所示。

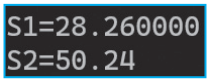

图2-2 例2-1运行结果

本例的功能为求半径为3和4的圆面积。因为面积值为实型数，所以定义为float型，对应的输出格式符为“%f”，默认数据保留6位小数，在“%”和“f”之间加“.2”，则输出时数据保留2位小数。

2.2.3 变量

变量的值在程序运行过程中是可以改变的，即可以多次赋值。C语言严格要求变量“先定义，后使用”。

定义变量的语句格式如下：

```
<数据类型><变量名1>,<变量名2>,<变量名3>,…;
```

例如：

```
int count;
float x,y;
double result;
char cc;
```

变量定义规定了被定义量的名称、性质、表示形式、占据存储空间的多少及构造特点。编译系统为不同类型的变量分配规定大小的存储空间。例如，在VC++ 2010 Express中，为每个int型变量分配4字节的存储空间，以后使用变量名访问该段存储单元，该变量的值只能是整型数据，不能超出−2 147 483 648～2 147 483 647范围。

变量定义后，其值不确定，需要为其赋一个确定的值后才能使用，例如：

```
int a;
a=100;
```

也可以在定义变量的同时为其赋初值——初始化。变量初始化的格式如下：

```
<数据类型><变量名1>=<常量表达式1>,<变量名2>=<常量表达式2>,<变量名3>,…;
```

例如：

```
int a=100;
```

在定义变量a的同时为其赋初值100。

也可以为部分变量赋初值，例如：

```
int a=100,b,c=500;
```

变量定义语句一般放在函数体的开头部分。

2.3 整型数据

在计算机中，整数是以其二进制的补码形式保存的。整型数据包括整型常量和整型变量。

2.3.1 整型数据的分类

整型变量的基本类型符为int。按照数值的表示范围，分为以下三类。

（1）基本类型：类型符为int。

（2）短整型：类型符为short int，可简写成short。

（3）长整型：类型符为long int，可简写成long。

在实际应用中，变量的值常常是正的，如学号、成绩、年龄等。为了充分利用存储空间，可省掉符号位的存储，将其分成以下两种类型。

（1）有符号型：类型符为signed。signed可省略，隐含为有符号型。

（2）无符号型：类型符为unsigned。

因此，归纳起来，整型数据有如下六种形式。

（1）有符号基本整型：[signed] int。

（2）无符号基本整型：unsigned int。

（3）有符号短整型：[signed] short [int]。

（4）无符号短整型：unsigned short [int]。

（5）有符号长整型：[signed] long [int]。

（6）无符号长整型：unsigned long [int]。

上述的方括号“[ ]”表示其中的内容可选，选与不选含义一样。例如：

```
int a;                  /* a 被定义为有符号整型变量 */
unsigned long c;        /* c 被定义为无符号长整型变量 */
```

编译系统不同，对各类整型数据长度的规定也不同。在VC++ 2010 Express中，短整型（short）为2字节，整型（int）和长整型（long）为4字节；在Turbo C中，短整型

（short）和整型（int） 为2字节，长整型（long）为4字节。C语言对于各类数据的长度没有明确规定，只要长整型的长度大于或等于整型、整型的长度大于或等于短整型即可。在VC++ 2010 Express中，整型数据的类型标识符、数据长度和数值范围如表2-1所示。

表2-1 VC++ 2010 Express中整型数据的类型标识符、数据长度和数值范围

类型	类型标识符	数据长度	数值范围
有符号整数	short	16位	−32 768～32 767
	int	32位	−2 147 483 648～2 147 483 647
	long	32位	−2 147 483 648～2 147 483 647
无符号整数	unsigned short	16位	0～65 535
	unsigned int	32位	0～4 292 967 295
	unsigned long	32位	0～4 292 967 295

使用无符号整数可以在长度不变的情况下扩大数值的表示范围。如果用2字节存储整型数据，则+13和−13的表示形式分别如下所示。

+13	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
−13	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1

对于有符号整数，最高位是数的符号，“0”代表“正”，“1”代表“负”；其余15位是尾数，用来存放数值，尾数最大可以表示的值为$2^{15}-1$。对于无符号整数，16位都用来表示数值，则尾数最大可以表示的值为$2^{16}-1$。

2.3.2 整型常量的表示

整型常量就是整常数。

1.整型常量的表示形式

在C语言中，整型常量有以下三种表示形式。

（1）十进制数：如220、−560、4 590。

（2）八进制数：以0作为前缀，如06、0105、05777。

（3）十六进制数：以0X或0x作为前缀，如0X0D、0XFF、0x4e。

2.整型常量的后缀

可以在整型常量的后面添加后缀来限定其类型。

（1）加一个“L”或“l”字母表示该数为long int型数，如22L、0773L、0Xae4l。

（2）加一个“U”或“u”字母表示该数为unsigned int型数。

2.3.3 整型变量的定义与使用

在C语言程序中，应该先定义一个变量，再为其赋初值，然后该变量才能参与运算，

而且在使用过程中其值不能超过该类型量的表示范围。

【例 2-2】整型变量的定义与混合运算。

程序如下所示。

```
#include <stdio.h>
void main()
{
    int a,b;
    signed short int c;
    short d=100;
    a=d-20;
    b=a+d;
    c=a+b+d;
    d=d-a+c-b;
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);
}
```

运行结果如图2-3所示。

```
a=80,b=180,c=360,d=200
```

图2-3　例2-2运行结果

本例中，变量c和d都是有符号短整型，在相应的定义语句中省略了signed和int。

【例 2-3】整型数据的溢出。

程序如下所示。

```
#include <stdio.h>
void main()
{
    int a=2147483647;
    unsigned int b=2,c,d;
    int e;
    c=a+b;
    d=a+b;
    e=a+b;
    printf("c=%d,d=%u,e=%u,%d\n",c,d,e,e);
}
```

运行结果如图2-4所示。

```
c = -2147483647, d = 2147483649, e = 2147483649, -2147483647
```

图2-4　例2-3运行结果

为什么程序中c的输出结果是−2 147 483 647呢？因为整数是以二进制的补码形式存储

的，有符号整型量的数值范围是−2 147 483 648～2 147 483 647。

变量a的值是+2 147 483 647，在计算机中的存储形式是：

0	1	1	…	…	…	1	1	1	1	1	1	1	1	1	1

变量b的值是2，在计算机中的存储形式是：

0	0	0	…	…	…	0	0	0	0	0	0	0	0	1	0

a+b的值为2 147 483 649，对于有符号整型数来说，超出了其表示范围，发生溢出错误。但是系统并不报告错误，继续按其内部规定操作，将a+b的值赋给c，则c的值是：

1	0	0	…	…	…	0	0	0	0	0	0	0	0	0	1

变量c的输出格式为“%d”，规定按有符号十进制整型数输出。最高位是数的符号位，1表示负数。对负数的补码再求补得到其原码，则c的原码是：

1	1	1	…	…	…	1	1	1	1	1	1	1	1	1	1

所以，c的输出结果是为−2 147 483 647。

变量d的输出格式为“%u”，规定按无符号十进制整型数输出，所以系统认为没有符号位，所有二进制位都是数值，表示为：

1	0	0	…	…	…	…	…	…	0	0	0	0	0	0	1

对应的十进制数的值是2 147 483 649。

实际上，系统在进行整型数运算时，不论是什么类型都用补码进行运算，至于结果如何表示，则取决于输出语句中的格式规定。

变量e以“%u”和“%d”两种格式输出。因为第一个e值的输出格式为“%u”，所以e的输出结果与d一样；第二个e值的输出格式为“%d”，所以输出结果也是−2 147 483 647。

由上述例子可以看出以下两点。

（1）C语言对数据类型检测不严格，不同类型的量可以参与运算并相互赋值。其中的类型转换是由编译系统自动完成的。有关类型转换的规则将在2.6.5节中介绍。

（2）当数值超过数据类型所能容纳的数值范围（溢出）时，系统不报错。

2.4 实型数据

实型也称浮点型。在C语言中，实型数据只采用十进制形式表示，所有实型数据均为有符号数，没有无符号的实型数据。

与整型数不同，浮点型数据是按照指数形式存储的。系统把一个浮点型数分成小数部分和指数部分分别存放。如3.141 59在内存中的十进制表示形式如图2-5所示，即+0.314 159 $\times 10^{1}$。

+	0.314 159	1
数符	小数部分	指数

图2-5　实型数在内存中的存储形式

2.4.1　实型数的分类

实型数分为单精度（float）、双精度（double）、长双精度（long double）三类。在VC++ 2010 Express中，实型数据的类型标识符、数据长度、有效数字和数值范围如表2-2所示。

表2-2　VC++ 2010 Express中实型数据的类型标识符、数据长度、有效数字和数值范围

类型标识符	数据长度	有效数字	数值范围
float	32位	6～7位	-3.4×10^{-38}～3.4×10^{38}
double	64位	15～16位	-1.7×10^{-308}～1.7×10^{308}
long double	64位	15～16位	-1.7×10^{-308}～1.7×10^{308}

2.4.2　实型常量

实型常量只采用十进制表示，有十进制数形式和指数形式两种表示方法。

1.十进制数形式

由数码0～9和小数点组成，小数点不可省略。例如：+29.56、－56.33、－6.8、.034、30.。

2.指数形式

由十进制数加阶码标志“e”或“E”及阶码（只能为整数，可以带符号）组成，其一般形式为

a E n

表示$a\times10^{n}$。其中，a为十进制数，不可省略；n为十进制整数。例如，2.1E5相当于2.1×10^{5}，3.7E－2相当于3.7×10^{-2}等。

以下这些实型常量不合法：345（无小数点）、E7（阶码标志E前无数字）、－5（无小数点）、53.－E3（负号位置不对）、2.7E（无阶码）、1e2.3（阶码为小数）。

系统输出指数形式的数据时，会自动转换为规范化形式输出，即小数点前保留一位有效数字。例如，12.34e2规范化后写成1.234e3。

【注意】所有实型常量默认为double型，若要表示float型常量，可以在常量的末尾加后缀“f”或“F”。例如，34.5为double型，34.5f为float 型。

2.4.3　实型变量

实型变量定义的格式和书写规则与整型相同。例如：

```
float a,f;                    /* a,f 被定义为单精度型变量 */
double b;                     /* b 被定义为双精度型变量 */
```

使用实型变量时要注意有效数字的位数，超过有效位数的值系统不能保证其准确性。

【例 2-4】浮点型变量数值的有效位数与舍入误差。

程序如下所示。

```
#include <stdio.h>
int main()
{
    int n,i;
    int factorial = 1, sum=0;
    float factorial_float = 1.0, sum_float = 0.0;
    double factorial_double = 1.0, sum_double = 0.0;
    printf("请输入一个整数：");
    scanf("%d", &n);
    // 使用整型计算 n 的阶乘及前 n 项阶乘和
    for (i = 1; i <= n; ++i) {
        factorial *= i;
        sum += factorial;
    }
    printf_s("整型占用字节数 %d, 计算 %d 的阶乘值：%d, 阶乘和：%d\n", sizeof(factorial),
n, factorial, sum);
    // 使用单精度浮点型计算 n 的阶乘及前 n 项阶乘和
    for(i = 1; i <= n; ++i) {
        factorial_float *= i;
        sum_float += factorial_float;
    }
     printf_s("单精度浮点型占用字节数：%d, 计算%d的阶乘值：%f, 阶乘和：%f\n",
sizeof(factorial_float), n, factorial_float, sum_float);
    // 使用双精度浮点型计算 n 的阶乘及前 n 项阶乘和
    for ( i = 1; i <= n; ++i)
    {
        factorial_double *= i;
        sum_double += factorial_double;
    }
     printf_s("双精度浮点型占用字节数：%d, 计算%d的阶乘值：%f, 阶乘和：%f\n",
sizeof(factorial_double), n, factorial_double, sum_double);
    return 0;
}
```

运行结果如图2-6所示。

```
请输入一个整数：13
整型占用字节数4,计算13的阶乘值：1932053504，阶乘和：-1839957479
单精度浮点型占用字节数：4,计算13的阶乘值：6227020800.000000,阶乘和：6749977088.000000
双精度浮点型占用字节数：8,计算13的阶乘值：6227020800.000000,阶乘和：6749977113.000000
```

图2-6　例2-4运行结果

从图2-6可知，在VC++ 2010 Express中，整型、单精度浮点型、双精度浮点型分别占用4、4和8个字节，当采用整型计算13的阶乘值时发生数据溢出，没有得到正确的阶乘值和阶乘和；单精度浮点型能够正确计算出13的阶乘值，但计算阶乘和时其精确度低于双精度浮点型，因为单精度浮点型变量的有效数字为7位，仅能保证前7位的数字是正确的，而双精度浮点型变量的有效数字为16位，从而可使得到的计算结果值更精确。

因此在编程中，我们要选择合适的数据类型，合理利用每一份内存和计算资源，践行节能和可持续发展理念，树立正确的价值观，减少资源浪费。党的二十大报告提出，要“协同推进降碳、减污、扩绿、增长，推进生态优先、节约集约、绿色低碳发展”。而在程序中仅声明必要的常量和变量，选择合适的数据类型，就是在优化程序性能，降低程序的时间复杂度和空间复杂度，推进程序完善、进化，从而达到节能的目的。

2.5 字符型数据

字符型数据是指由中文字符、英文字符、数字字符或其他ASCII码字符构成的，通常不需要计算的数据，字符长度为1个字符。

字符型数据是以字符ASCII码值的二进制形式在计算机中保存的，二进制长度为1字节。例如，大写字母A的ASCII码值为65，保存在计算机中的形式是01000001。

2.5.1 字符常量

字符常量是用单引号引起来的一个字符。例如，'a'、'A'、' '、'?'、'9'、'\n'等都是不同的合法字符常量。


常用ASCII码表

字符常量也可以用转义字符表示。转义字符必须以“\”开头，后面跟一个字符或该字符ASCII码值的八进制或十六进制形式。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。例如，'\102'（八进制）和'\X42'（十六进制）都表示大写字母B，'\" '表示一个双引号、'\n'表示换行。常用的转义字符及其含义如表2-3所示。

表2-3　常用的转义字符及其含义

转义字符	等价形式	含义
\n	\012	换行，将光标当前位置移到下一行的开头
\r	\015	回车，将光标当前位置移到本行的开头

续表

转义字符	等价形式	含义
\t	\011	制表键，将光标当前位置跳到下一个制表位
\f	\014	换页，将光标当前位置移到下页开头
\b	\010	退格，将光标当前位置移到前一列
\\	\134	一个反斜杠字符“\”
\'	\047	一个单引号字符“'”
\"	\042	一个双引号字符“"”
\ddd	—	1～3位八进制数作为ASCII码值所代表的字符
\xhh	—	以x开头的1～2位十六进制数作为ASCII码值所代表的字符

C语言字符集中的任何一个字符均可用转义字符来表示，只要知道该字符的ASCII码值，利用表2-3中的\ddd和\xhh格式就可以表达出来。例如，大写字母A的转义字符表示形式为'\101'或'\x41'。

转义字符一般用来表示那些在代码中不便于表示的控制字符。

【例2-5】转义字符的使用。

程序如下所示。

```
#include <stdio.h>
void main()
{
    int a,b;
    a=1; b=2;
    printf("\'%d\'\n\t%d    %d\n",a,a,b);
    a=6+2;
    printf("\\\141=6+2=\t\b%d\n",a);
}
```

运行结果如图2-7所示。

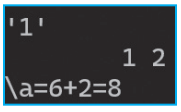

图2-7 例2-5运行结果

程序运行后，首先在第一列输出一个单引号“'”（\'），然后输出a的值1，紧接着再输出一个“'”，碰到“\n”换行；接下来的“\t”控制光标跳到下一个制表位，再输出a和b的值，碰到“\n”又换行；第二条printf语句先输出一个反斜杠“\”（\\），“\141”是a的转义字符，遇到“\t”跳到下一个制表位（与上一行的1对齐），但下一个转义字符“\b”又使其退回一格。

2.5.2 字符变量

字符变量的取值是字符常量，即单个字符。字符变量定义的格式和书写规则都与整型变量相同。例如：

```
char a;
```

字符在计算机中以其ASCII码值方式存储，因此也可以把它们看作整型数值。C 语言允许对整型变量赋予字符值，也允许对字符变量赋予整型值。在输出时，允许把字符变量按整型输出，也允许把整型变量按字符型输出。整型数占4字节，字符型数占1字节。当整型数按字符型处理时，只有低8位参与处理。

字符变量在参与数值运算时，使用的是其ASCII码值。例如，有整型变量a，执行语句“a='5'+5;”后，变量a的值是字符'5'的ASCII码值53加上5，即58。

【例 2-6】字符变量的赋值与输出。

程序如下所示。

```
#include <stdio.h>
void main()
{
    char a,b;
    int c;
    a='x';b='\171';c=122;
    printf("%c,%c,%c\n%d,%d,%d\n",a,b,c,a,b,c);
}
```

运行结果如图2-8所示。

```
x,y,z
120,121,122
```

图2-8 例2-6运行结果

从运行结果中可以看出，整型量与字符型量可以通用，输出结果取决于输出格式。“%c”表示按字符输出，“%d”表示按十进制数输出字符的ASCII码值。

【例 2-7】大小写字母的转换。

程序如下所示。

```
#include<stdio.h>
void main()
{
    int a='a',b='B',c,d;
    c=a-('a'-'A');
    d=b+32;
    printf("%c--->%c\n%c--->%c\n",a,c,b,d);
 }
```

运行结果如图2-9所示。

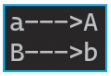

图2-9 例2-7运行结果

本例中，'a'-'A'实际上就是小写与大写字母ASCII码值的差值32，即97-65。将某大写字母的ASCII码值加32，即可转换得到其小写字母。

2.5.3 字符串常量

字符串常量是由一对双引号括起来的字符序列。例如，"China"、"Turbo C2.0 "、"3"、"$12.5" 、"中国"等都是合法的字符串常量。"3"因为使用了双引号，所以是字符串常量，'3'是字符常量。

字符串常量和字符常量是不同的量。两者主要有以下区别。

（1）字符常量由单引号括起来，字符串常量由双引号括起来。

（2）字符常量只能是单个字符，字符串常量则可以包含0个或多个字符。

（3）可以把一个字符常量赋给一个字符变量，但不能把一个字符串常量赋给一个字符变量。例如，语句“a="d";”是不合法的。在C语言中没有相应的字符串变量，但是可以使用一个字符数组来存放一个字符串常量，我们将在第6章中具体介绍。

（4）字符常量占1字节的内存空间。字符串常量占用的内存字节数等于字符串中的字符长度加1。增加的1字节中存放字符'\0'（ASCII码值为0的字符，也称空字符）。'\0'是字符串的结束标志，系统会自动在字符串末尾加空字符。例如，字符串 "China"在内存中所占的字节可表示为：

C	h	i	n	a	\0

字符常量'a'和字符串常量"a"虽然都只有一个字符，但在内存中的存储情况是不同的。

'A'在内存中占1字节，表示为：A。

"A"在内存中占2字节，表示为：A\0。

系统在对字符串进行操作时，遇到'\0'，即认为该字符串结束。因此，在字符串中要恰当使用字符'\0'。例如，语句“printf("this\0 is a student.");”的输出结果为“this”。

【例2-8】使用字符串常量打印图形。

程序如下所示。

```
#include <stdio.h>
void main()
{
    char a='*';
    printf("%s\n%c     %c\n%c     %c\n%s\n","******",a,a,a,a, "******");
 }
```

运行结果如图2-10所示。

图2-10　例2-8运行结果

输出格式“%s”表示输出字符串。

2.6 运算符及表达式

运算符丰富是C语言的一大特点。C语言的运算符不仅具有优先级，还有结合性，两者制约了运算的优先顺序。将操作数用运算符连接起来就构成了表达式。

2.6.1 运算符概述

1.运算符分类

根据运算符对操作数数量的要求，运算符可分为单目、双目、三目等几类。例如，负号运算符“−”是单目运算符，−a是对a进行一目求负操作；乘号“*”是双目运算符。

根据运算符的功能，可将其分为以下几类。

（1）算术运算符：+（加）、−（减）、*（乘）、/（除）、%（求余或取模）、−−（自减1）、++（自增1）。

（2）关系运算符：>（大于）、<（小于）、==（等于）、>=（大于等于）、<=（小于等于）、!=（不等于）。

（3）逻辑运算符：!（逻辑非）、&&（逻辑与）、||（逻辑或）。

（4）位运算符：&（按位与）、|（按位或）、～（按位非）、^（按位异或）、<<（按位左移）、>>（按位右移）。

（5）赋值运算符：=（简单赋值）、复合算术赋值（+=，−=，*=，/=，%=）、复合位运算赋值（&=，|=，^=，>>=，<<=）。

（6）条件运算符（?:）：属于三目运算符。

（7）逗号运算符（,）：用于把若干表达式组合成一个表达式。

（8）指针运算符：对内存直接操作，包括 *（取内容）和 &（取地址）。

（9）强制类型转换运算符（类型符）：用于数值类型的转换。

本章主要介绍算术运算符、位运算符、赋值运算符、逗号运算符、强制类型转换运算符的使用，第4章介绍关系运算符、逻辑运算符、条件运算符在选择结构程序设计中的使

用，指针运算符将在第8章中重点介绍。

2.运算符的优先级与结合性


运算符优先级及其结合性

所谓优先级是指不同运算符在表达式中参加运算的先后次序。在C语言中，括号内的表达式先进行运算，其余的除赋值运算符外，单目运算符的优先级高于双目运算符，而双目运算符的优先级又高于三目运算符。

一般地，运算符的优先级由高到低为：算术运算符 > 关系运算符 > 逻辑运算符 > 赋值运算符 > 逗号运算符。

如果表达式包含的运算符优先级别相同，那么运算方向是从左向右还是从右向左与运算符的结合方向有关。除单目运算符、条件运算符和赋值运算符的方向是从右向左外，其他运算符的结合方向都是从左向右。例如：a=b+c+d的运算顺序是先计算b+c的值，算出的结果加上d的值，然后将结果赋给变量a。

2.6.2 算术运算符和算术表达式

算术运算符及其运算优先级如表2-4所示。

表2-4 算术运算符及其优先级

运算符	作 用	优先级是否相同	结合方向	优先级
–	取负	是	从右向左	高
––	自减1			
++	自增1			
*	乘	是	从左向右	
/	除			
%	取模（取余数）			
+	加	是		
–	减			低

1.算术运算符的使用

（1）对于除法运算符“/”，如果参与运算的两个量均为整型，那么结果也为整型，会舍去小数；如果运算量中至少有一个为实型，则结果为双精度实型。

（2）对于取余运算符“%”，取余运算的结果等于两个数相除后的余数。要求参与运算的两个量均为整型。例如，6%5的值为1，65%10的值为5。

【例 2-9】除法与取余运算示例。

程序如下所示。

```
#include <stdio.h>
void main()
```

```
{
    float a,b,c,d;
    int e,f;
    a=2/3;
    b=2/3.0;
    c=-4/3;
    d=-4.0/3;
    e=2%3;
    f=7%3;
    printf("a=%.3f\nb=%.3f\nc=%.3f\nd=%.3f\n",a,b,c,d);
    printf("e=%d\nf=%d\n",e,f);
}
```

运行结果如图2-11所示。

```
a=0.000
b=0.667
c=-1.000
d=-1.333
e=2
f=1
```

图2-11　例2-9运行结果

运算前，C语言系统要将参加运算的操作数转换成同一类型的数据，转换方向是向长度较长的数据类型转换，运算结果的数据类型就是转换后的数据类型。

2和3均为整型数，2/3的结果也为整型数，舍掉小数，所以a的值为0。

2/3.0中有一个为浮点型数，运算结果也为浮点型数，所以b的值为0.667。

不论是正数还是负数，大多数系统处理整除时都是舍去小数取整数部分作为运算结果的，所以c的值为－1。

（3）除法运算符与取余运算符配合，可以拆分一个整型数得到其各个数位。

一般地，一个整数a某数位的值为：a / 位权 % 10。

例如，3 869百位上的数值为3 869 / 100 % 10 = 8。

【例 2-10】将一个三位数倒序显示。

程序如下所示。

```
#include <stdio.h>
void main()
{    int a,w0,w1,w2;
     printf("input:");
     scanf("%d",&a);
     w0=a%10;
     a=a/10;
     w1=a%10;
```

```
    a=a/10;
    w2=a;
    printf("output:%d%d%d\n",w0,w1,w2);
}
```

运行结果如图2-12所示。

```
input:123
output:321
```

图2-12 例2-10运行结果

运行该程序，从键盘输入123后按回车键，则输出321。

对于各个数位的分解，在这段代码中采用了相同的除10再取余的运算，便于将来应用到循环程序中，分解后原数a被改变。也可以采用下面的程序段，分解后原数a不变。

```
w0= a%10;
w1= a/10%10;
w2= a/100%10;
```

（4）自增1、自减1运算符只能用于变量，可以写在变量的前面，也可以写在变量的后面，但两者的意义有所不同。以应用于变量i为例，有以下几种形式。

①++i：变量i自增1后，再参与其他运算。

②--i：变量i自减1后，再参与其他运算。

③i++：变量i参与运算后，再自增1。

④i--：变量i参与运算后，再自减1。

【例2-11】自增1、自减1运算符的功能示例。

程序如下所示。

```
#include <stdio.h>
void main()
{   int i=2,j=-2,k,m;
    printf("%d,%d\n",i++,++j);
    printf("%d,%d\n",i--,--j);
    k=-i++;
    m=--j;
    printf("k=%d,m=%d,i=%d,j=%d\n",k,m,i,j);
}
```

运行结果如图2-13所示。

```
2,-1
3,-2
k=-2,m=-3,i=3,j=-3
```

图2-13 例2-11运行结果

在第一条printf语句中， i++是先输出i的值，i再加1；++j是j先自增1，再输出。所以输出结果为2和－1。输出后，i为3，j为－1。

在第二条printf语句中，i－－是先输出i的值，i再减1；－－j是j先自减1，再输出。所以输出结果为3和－2。输出后，i为2，j为－2。

在语句k=－i++;中，－i++要理解为－(i++)，因为“++”和“－－”运算符只能用于变量，－i是表达式。所以，i的值先取负，赋给k后再加1，所以k的值为－2，i自增1后的值为3。

（5）由于C语言对运算符、表达式书写和使用的规定不是很严格，灵活性较大，所以，为避免产生歧义和误解，应注意书写形式，并适当使用括号。

【例 2-12】自增 1、自减 1 运算符的表达式求值。

程序如下所示。

```
#include <stdio.h>
void main()
{    int i=5,j=5,k,p,q;
     k=i+++j;
     printf("i=%d,j=%d,k=%d\n",i,j,k);
     p=(i++)+(i++)+(i++);
     q=(++j)+(++j)+(++j);
     printf("i=%d,j=%d,p=%d,q=%d\n",i,j,p,q);
     printf("k1=%d,k2=%d\n",k++,k++);
     printf("k1=%d,k2=%d\n",++k,++k);
     printf("k=%d\n",k);
}
```

运行结果如图2-14所示。

```
i=6,j=5,k=10
i=9,j=8,p=18,q=24
k1=11,k2=10
k1=14,k2=14
k=14
```

图2-14　例2-12运行结果

对于i+++j，应理解为(i++)+j，而不是i+(++j)，所以，将i+j的值10赋给k后，i再自增1变为6。i+++j最好写成(i++)+j，以避免误解。

对于p=(i++)+(i++)+(i++)，应理解为三个i相加，故p的值为18，然后i再自增1三次，i最后的值为9。对于q=(++j)+(++j)+(++j)则不然，不同的C语言编译器理解不同。在Turbo C 2.0中，理解为j先自增1三次，变为8，然后三个8相加，q的值为24；而在VC++ 2010 Express中，q的值为22。因此，在实际应用中，应力求不写“q=(++j)+(++j)+(++j);”这样的代码，避免出现不同编译器下求值结果不一致的情况。可以改写为：

```
++j;
++j;
++j;
q=3*j;
```

第三、四条printf语句中的两个参数同样有运算次序问题。函数参数的运算次序是从右向左。在实际应用中也要注意避免误解，适当改写。

2.算术表达式和算术运算符的优先级、结合性

表达式是将常量、变量、函数用运算符连接起来构成的式子。一个表达式的计算结果及其类型就是这个表达式的值和类型，表达式的求值按运算符的优先级和结合性规定顺序进行。用算术运算符和括号将运算对象连接起来构成的式子是算术表达式。例如，a+b、(a+5)/c、(−b+sqrt(x))/(2*a)等等。

运算符的优先级是指不同运算符同时出现时运算的优先顺序。在C语言中，运算符的优先级别共分15级，优先级别高的运算符比优先级别低的运算符先计算。当一个运算量两侧的运算符优先级相同时，按运算符的结合性所规定的结合方向处理。结合性有左结合性和右结合性两种，大部分运算符是左结合的，典型的右结合运算符是赋值运算符。例如，x=y=10，由于运算符的右结合性，所以应先执行y=10，再执行x=(y=10)。

2.6.3 赋值运算符和赋值表达式

1.赋值运算符和赋值表达式的使用

由赋值运算符“=”连接的式子被称为赋值表达式。其一般形式为：

```
变量 = 表达式
```

其中，“表达式”部分可以是任意类型的表达式，具体可以是常量、变量或者表达式。赋值运算符左侧必须是变量名。例如，a+2=b+5是错误的赋值表达式。下列表达式均为合法的赋值表达式。

```
x='10'
x=w
y=sin(a)+b
a=(b=(c=3))
```

赋值表达式的功能是计算表达式的值，然后将值赋给“=”左边的变量。赋值运算符具有右结合性。

在进行赋值运算时，允许赋值运算符两边的数据类型不一致，但必须是数值或字符型数据。系统会自动进行类型转换，以保证“=”左边的变量类型不变，并尽量保证转换前后的值不变。例如，将浮点数据赋值给整型变量时，将舍去小数部分；将整型数据赋值给

浮点型变量时，值不变，保存形式变为浮点型，即增加小数部分（小数部分的值为0）。

【例 2-13】赋值表达式示例。

程序如下所示。

```
#include <stdio.h>
void main()
{    int a;
     float b;
     a=1234.56;
     b=123;
     printf("a=%d,b=%.2f\n",a,b);
}
```

运行结果如图2-15所示。

```
a=1234,b=123.00
```

图2-15　例2-13运行结果

将浮点数1234.56赋给整型量a后，舍去小数部分。浮点型量的输出格式符是“%f”。

赋值表达式也可以出现在其他语句中。例如，语句“printf("%d",a=b);”中的赋值表达式a=b表示先将b的值赋给变量a，然后输出变量a的值。

2.复合赋值运算符

在赋值运算符“=”前加上某些二目运算符，构成复合赋值运算符。有+=、−=、*=、/=、%=、<<=、>>=、&=、^=、|=共10种。

例如：

```
a+=b                    // 等价于 a=a+b
a % =b                  // 等价于 a=a % b
a/=b−c                  // 等价于 a=a/(b−c)
```

【注意】a/=b−c理解为 a=a/(b−c)，运算符右侧的表达式应整体参与计算。

复合赋值运算符有利于编译处理，能提高编译效率，产生质量较高的目标代码。

2.6.4　逗号运算符和逗号表达式

逗号运算符“,”也称顺序求值运算符。用逗号运算符把多个表达式连接起来的式子称为逗号表达式。逗号表达式的一般形式为：

```
表达式 1, 表达式 2, 表达式 3, …
```

逗号表达式的求值顺序是从左向右依次求表达式的值，将最右边表达式的值作为整个表达式的值。

逗号运算符的优先级别在所有运算符中是最低的。例如，执行语句“a=(1*2,3*4);”后，a的值为12，取第二个表达式3*4的结果赋给a。如果将语句改为“a=1*2,3*4;”形式，那么，因为逗号运算符的优先级低于赋值运算符，所以a的值为2。

前面出现的printf()函数中的“,”是函数参数分隔符，而不是逗号运算符。

【例 2-14】逗号表达式的运算示例。

程序如下所示。

```
#include <stdio.h>
void main()
{
    int a, b;
    a=10;
    b=(a=a-5,a/5);
    printf("b=%d\n",b);
}
```

运行结果如图2-16所示。

图2-16 例2-14运行结果

本程序中，先运算括号内的逗号表达式的值，然后将其赋给变量b。执行a=a-5后，a的值为5，再执行a/5结果为1，则逗号表达式的值为1，所以b的值为1。

2.6.5 不同数据类型之间的转换

在表达式中，参与运算的数据类型是可以相互转换的。转换的方法有两种：一种是自动类型转换，另一种是强制类型转换。

1.自动类型转换

自动类型转换发生在不同数据类型的量混合运算时，由编译系统自动完成。自动类型转换应遵循以下规则。

（1）若参与运算的数据类型不同，则先转换为同一类型，然后进行运算。运算结果的数据类型也就是转换后的数据类型。

（2）转换按数据长度增加的方向进行，以保证不降低精度。如int型和long型运算时，先将int型转换为long型再进行运算。

（3）所有的浮点运算都是以双精度进行的，即使仅含单精度量的表达式，也要先转换成双精度型，再进行运算。

（4）char型和short型参与运算时，必须先转换为int型。

在赋值运算中，赋值号两边量的数据类型不同时，赋值号右边量的数据类型将转换为

左边量的数据类型。如果右边量的数据类型长度比左边量长，则将丢失一部分数据，从而降低精度，丢失的部分按四舍五入规则向前舍入。不同数据类型之间的自动转换规则如图2-17所示。

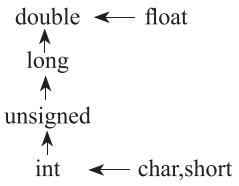

图2-17　不同数据类型之间的自动转换规则

2.强制类型转换

强制类型转换是通过类型转换运算来实现的。一般形式为：

```
(类型说明符)(表达式)
```

其功能是把表达式的运算结果强制转换成类型说明符所表示的类型。

例如：

```
(float) a     // 把 a 的值转换为 float 型
(int)(x+y)    // 把 x+y 的结果转换为整型
```

在使用强制类型转换时应注意以下问题：

（1）类型说明符和表达式都必须加括号（表达式是单个变量时可以不加括号），例如，如果将(int)(x+y)写成(int)x+y，则变成将x转换成int型之后再与y相加。

（2）无论是强制类型转换还是自动类型转换，都只是为了本次运算的需要而对变量的数据类型进行的临时性转换，不改变数据定义时对该变量所定义的类型。

【例 2-15】强制类型转换示例。

程序如下所示。

```
#include <stdio.h>
void main()
{
    float g=6.75;
    printf("(int)g=%d,g=%.2f\n",(int)g,g);
}
```

运行结果如图2-18所示。

```
(int)g=6,g=6.75
```

图2-18　例2-15运行结果

本例表明，g虽被强制转换为int型，但只在运算中起作用，是临时的，而g本身的数据类型并不改变。因此，(int)g的值为6（截去了小数），而g的值仍为6.75。

2.6.6 位运算

位运算是指对二进制位的运算，是对字节或字节中的实际位进行检测、设置或移位。位运算只适用于字符型和整型变量以及它们的变体，对其他数据类型不适用。除按位取反是单目运算符外，其他位运算符都是双目运算符。

位运算符有如下6种。

（1）&：按位与。

（2）|：按位或。

（3）^：按位异或。

（4）~：按位取反。

（5）>>：按位右移。

（6）<<：按位左移。

按位运算的结果取决于两个操作数的每一个相对应的二进制位。具体运算规则如下。

（1）与运算：两个二进制位均为1时，结果为1，否则为0。

（2）或运算：两个二进制位均为0时，结果为0，否则为1。

（3）异或运算：两个二进制位相同时，结果为0；不相同时，结果为1。

（4）求反：将二进制位按位取反，即1变0，0变1。

（5）移位：将每一个二进制位向右或向左移动一位，移位后，一端的位被“挤掉”，另一端空出的位以0填补，只要值不溢出，左移一位相当于原值乘以2，右移n位相当于原值除以2n。

例如，求-14&3的值。假设机器内部用2字节来表示数据，则结果为2。计算过程如下：

```
      1111111111110010  ----------------  -14的补码
&）   0000000000000011  ----------------  3的补码
      0000000000000010  ----------------  -14&3的补码
```

【例2-16】与、或运算的应用。

程序如下所示。

```
#include <stdio.h>
void main()
{    int a=0x16,b=0xf,c;
     int d=0x30,e=0x07,f;
     c=a&b;
     f=d|e;
     printf("c=%#x,f=%#x\n",c,f);
}
```

运行结果如图2-19所示。

```
c=0x6,f=0x37
```

图2-19 例2-16运行结果

利用按位与，可以取出某个数的某几位二进制值或保留某个数的某几位二进制值。十六进制数0x16（00010110）和0xf（00001111）相与，高4位被屏蔽掉，取得低4位，所以结果为6（00000110）。

输出格式符“%x”规定按十六进制输出数据，加上“#”，则数据会加上十六进制数前缀0x。

【例 2-17】交换两个变量的值。

程序如下所示。

```
#include <stdio.h>
void main()
{
    int a=3,b=5;
    printf("a1=%d,b1=%d\n",a,b);
    a=a^b;
    b=b^a;
    a=a^b;
    printf("a2=%d,b2=%d\n",a,b);
}
```

运行结果如图2-20所示。

```
a1=3,b1=5
a2=5,b2=3
```

图2-20　例2-17运行结果

本程序中，开始时a的值为3，b的值为5；处理后，a的值为5，b的值为3。处理过程如下：

```
        0000000000000011 ---------------- 3的补码（a）
^）     0000000000000101 ---------------- 5的补码（b）
        0000000000000110 ---------------- 6的补码赋值给a
^）     0000000000000101 ---------------- 5的补码（b）
        0000000000000011 ---------------- 3的补码赋值给b
^）     0000000000000110 ---------------- 6的补码（a）
        0000000000000101 ---------------- 5的补码赋值给a
```

利用异或，可以将某个数的某些二进制位翻转，即原来位值为1的变为0，原来位值为0的变为1。如果将某个数异或0xf，可将该数的后4位翻转。

拓展练习

1.写出下列代数式对应的C语言表达式。

（1）$6+\frac{(4+x)^2}{2+y}$

（2）$\frac{\sin^2(a+b)}{4x^2y}$

2.若x为float型，其原值为5，a=2,b=4.7，写出下列表达式运算后x的值。

（1）x=(int)(b−a)%3*a/4−a

（2）x=(x=b+1)+(int)(b)%10/2.0

（3）x+=x

（4）x−=x

（5）x*=x+x

（6）x/=x+x

（7）x+=x−=x*=x

（8）x%=x

（9）x=3*4,5*6

3.写出下面程序的运行结果。

```
#include <stdio.h>
void main()
{
    int a=2;
    printf("abcdefghijk\n");
    printf("lmnop/n");
    printf("I am a /n beginner of C! \n ");
    printf("I am a \n beginner of C\n");
    printf("%d + %d = %d\n",a,a,a+a);
}
```

4.写出下面程序的输出结果。

```
#include <stdio.h>
void main()
{
    int i,j;
    float s,b,a;
    char c;
    long m,n;
    i=5;j=−3;
    a=25.5;b=3.0;
    m=a/b;n=m+i/j;
    printf("%d\n",n);
}
```

拓展练习2