高等学校计算机应用规划教材

C 语言程序设计

(第四版)

谭浩强 主编

杨秀菊 付长凤 杨岚 吴宗波 副主编

清华大学出版社

北京

内 容 简 介

C 语言课程是我国许多高校为学生开设的第一门程序设计语言课程。C 语言具有很强的实用性，它既可用来编写系统软件，也可用来编写各种应用软件。

本书主要内容包括：C 语言概述，数据类型、运算符与表达式，程序设计初步，选择结构程序的设计，循环结构程序的设计，数组，函数，预处理命令，指针，结构体与其他数据类型，位运算，文件等。书中涵盖了大量的程序设计实例，通过对实例的学习，读者能够更好地掌握运用 C 语言进行程序设计的方法和技巧。

本书既可作为高等院校应用型本科专业学生的教材，也可供自学者以及参加 C 语言计算机等级考试的考生阅读参考。

为了使读者更好地掌握 C 语言，清华大学出版社还出版了与本教材配套的学习指导与实验辅导教材：《C 语言程序设计学习指导与实验教程(第四版)》。

本书对应的电子课件、实例源代码和习题答案可以通过 http://www.tupwk.com.cn/downpage 网站下载。

图书在版编目(CIP)数据

C 语言程序设计 / 高禹　主编. -4 版. -北京：清华大学出版社，2018
(高等学校计算机应用规划教材)
ISBN 978-7-302-49485-0

Ⅰ.①C…　Ⅱ.①高…　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(2018)第 017658 号

责任编辑：胡辰浩　袁建华
装帧设计：孔祥峰
责任校对：成凤进
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张**：15　　**字　　数**：346 千字
版　　次：2011 年 1 月第 1 版　　2018 年 2 月第 4 版　　**印　　次**：2018 年 2 月第 1 次印刷
印　　数：1～3500
定　　价：43.00 元

产品编号：

前　言

C 语言是被人们广泛使用的一种计算机语言。由于它功能丰富，灵活性强，可移植性好，语言简洁，应用面广，因此深受广大用户的喜爱。C 语言具有较强的实用性，它既可以用来编写系统软件，也可以用来编写各种应用软件。

C 语言程序设计既是计算机专业的必修课程，也是国内许多高校为非计算机专业学生开设的一门程序设计语言课程。对于从未接触过程序设计语言的学生来说，在规定的有限学时内，掌握好 C 语言具有一定的难度。作者在编写本书时，根据多年从事 C 语言教学的经验，充分地考虑到了以上实际情况。

本书的编写具有如下主要特点：

(1) 本书的内容编排充分考虑到高等院校培养应用型本科专业学生和初学者的要求。

(2) 本书选择学生容易理解的问题作为实例，结合知识点讲解程序设计的方法和技巧。

(3) 本书本着实用的原则，重点放在如何使用 C 语言来解决实际问题。在丰富的例题中包含了各种常见问题，对于例题中出现的各种算法都有较详细的解释。

(4) 与本书相配套，我们编写了《C 语言程序设计学习指导与实验教程(第四版)》，对各章知识的要点和难点整理归纳和深入分析，为读者准备了各种类型的习题，并且给出了习题的参考答案。还为读者设计了各种上机实验项目并详细说明了各实验的目的和内容。

(5) 本书内容覆盖了“C 语言计算机等级考试”的内容。

全书共分 12 章：第 1 章介绍了 C 语言的发展历史、特点及源程序结构；第 2 章介绍了 C 语言的基本数据类型、运算符和表达式；第 3 章介绍了 C 语言基本的输入输出操作和顺序结构程序设计；第 4 章介绍了 C 语言的选择结构程序设计；第 5 章介绍了 C 语言的循环结构程序设计；第 6 章介绍了 C 语言的数组；第 7 章介绍了 C 语言函数的调用、变量的存储类别；第 8 章介绍了 C 语言的预处理命令；第 9 章介绍了 C 语言指针的使用；第 10 章介绍了 C 语言的结构体和其他数据类型；第 11 章介绍了 C 语言的位运算；第 12 章介绍了 C 语言文件的概念及操作。

本书条理清晰、语言流畅、通俗易懂、实用性强。本书既可以作为高等院校应用型本科专业学生的教材，也可以供自学者以及参加 C 语言计算机等级考试的考生阅读参考。

除主编和副主编外，参加本书编写工作的还有许瑞斌、许戈静、何天兰、张梅娇、林玉梅、林博艺、苏乐辉、苏延平、郭新华、黄丽凤、王广伟、李鑫、毕振波、郑芸、张建科、张威、张艳艳、侯志凌、黄海峰和管林挺等人。感谢参加编写工作的泉州信息工程学院的老师。

由于编者水平有限，书中难免存在错误与不足，诚恳欢迎读者批评指正。我们的联系方式为电子邮箱：huchenhao@263.net，电话：010-62796045。本书对应的电子课件、实例源代码和习题答案可以通过 http://www.tupwk.com.cn/downpage 网站下载。

编　者

2018 年 1 月

目　　录

第1章　C语言概述

C 语言是一种优秀的结构化程序设计语言。它具有语言简洁、紧凑，使用灵活和可移植性强等优点，深受广大编程人员的喜爱，并得到广泛应用。

本章主要介绍了 C 语言的发展简史、C 语言的特点，以及 C 程序的编译、链接和运行。

1.1　C 语言的发展历史简介

C 语言是由美国贝尔实验室的 Dennis Ritchie 于 1972 年开发出来，并首次在 UNIX 操作系统的 DEC PDP-11 计算机上使用的计算机语言。它由早期的 B 语言发展演变而来。在 1970 年，贝尔实验室的 Ken Thompson 根据 BCPL(Basic Combined Programming Language)语言设计出了较简单且接近硬件的 B 语言。但由于 B 语言过于简单，功能有限，无法满足人们的需要，所以 Dennis Ritchie 在此基础上又开发了 C 语言。C 语言既继承了 B 语言的优点，又克服了它的缺点。

最初的 C 语言只能在大型计算机上执行，随着微型计算机的日益普及，它逐渐被移植到微机上来，并且出现了许多不同版本的 C 语言。但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，1983 年美国国家标准化协会 ANSI(American National Standards Institute)为 C 语言制定了标准，即 ANSI C。1987 年，ANSI 公布了 C 语言新的标准；1989 年，ANSI 又公布了一个新的 C 语言标准——C89，现在流行的各种 C 语言版本都是以它为标准的。1990 年，国际标准化组织 ISO(International Standards Organization)接受 C89 作为国际标准，通常称为 C90。1999 年，ISO 对 C 语言标准进行了修订，在基本保留原来的 C 语言特征的基础上增加了一些面向对象的特征，简称为 C99。目前流行的 C 语言编译系统大多是以 C89 为基础进行开发的。微机上正在使用的 C 语言编译系统有 Turbo C、WIN TC(Turbo C 的 Windows 版本)、C-Free、Visual C++等。

1.2　C 语言的特点

C 语言凭借其强大的功能，早已成为最受欢迎的语言之一。许多著名的软件都是用 C 语言编写的。C 语言具有如下一些特点。

(1) 语言简洁、紧凑，使用方便、灵活，具有丰富的运算符和数据结构。C 语言一共只有 32 个关键字，9 种控制语句，34 种运算符。C 语言把括号、赋值、强制类型转换等都作为运算符处理，从而使得 C 语言的运算类型极其丰富，表达式类型多样化。C 语言的数

据类型有：整型、实型、字符型、枚举类型、数组类型、指针类型、结构体类型、共用体类型等，使用这些数据类型可以实现各种复杂的数据结构运算。

(2) C 语言允许直接访问物理地址，能进行位操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此，C 语言既具有高级语言的功能，又具有机器语言和汇编语言的许多功能，可用来编写系统软件。C 语言既是成功的系统描述语言，又是通用的程序设计语言，人们通常称之为“中级语言”，即它兼有高级语言和汇编语言的特点。

(3) C 语言具有结构化的控制语句(如 if…else 语句、while 语句、do…while 语句、switch 语句、for 语句)，用函数作为程序模块以实现程序的模块化，是结构化的理想语言，符合现代编程语言风格的要求。

(4) 语法限制不太严格，程序设计自由度大。例如，对数组下标越界不作检查，由程序编写者自己来保证程序的正确性；对变量的类型使用比较灵活，如整型数据与字符型数据以及逻辑型数据可以通用。一般的高级语言语法检查比较严格，能检查出几乎所有的语法错误。而 C 语言允许程序编写者有较大的自由度，因此放宽了语法的检查。程序员应当仔细检查程序，来保证其正确，而不要过分依赖 C 编译程序来检查错误。

(5) 用 C 语言编写的程序可移植性好(与汇编语言相比)。在某一操作系统编写的程序，基本上不做任何修改就能在其他类型的计算机和操作系统上运行。

(6) 生成目标代码质量高，程序执行效率高。一般只比汇编程序生成的目标代码效率低 10%~20%。

C 语言的以上特点，使得 C 语言功能强大、应用广泛。用 C 语言可以编写出任何类型的程序，它既可用来编写系统软件，也可以用来编写各种应用软件。但同时 C 语言对编程人员也提出了更高的要求，与学习其他的高级语言相比，编程人员学习 C 语言必须将更多的心思花费在学习语法上，尤其是指针的应用，常让初学者摸不着边际。但是，一旦熟悉了 C 语言的语法，便可以享受到 C 语言所带来的便利性与快捷性。

1.3　C 语言源程序举例

通过第 1.2 节的介绍，读者应已了解了一些 C 程序的特点。下面通过几个简单的 C 程序实例，进一步分析 C 程序的结构特点。

例 1.1　在屏幕上显示两行信息，分别是“How are you！”和“Welcome you!”。

程序代码如下：

```
# include <stdio.h>
int main()
  {
      printf("How are you!\n");
      printf("Welcome you!");
      return 0;
      }
```

程序运行的结果是输出如下两行文本信息：

```
How are you!
Welcome you!
```

C 程序是由许多函数组合而成的，而在某个函数里面又可以再调用其他函数。

在上面的程序中，main 表示“主函数”，每一个 C 程序都必须有一个 main 函数，它是程序执行的入口。main 前面的 int 表示函数的返回类型，即 main 函数为基本整型类型。

上面程序中的一对大括弧{ }括起来的部分称为函数体。函数体内的 printf 是 C 语言中的输出函数，双引号内的字符串按原样输出。\n 是换行符，即在输出“How are you!”之后回车换行，然后在屏幕的下一行输出“Welcome you!”。每个语句结尾为一个分号。函数体内的 return 语句为主函数结束时的返回值。由于 main 函数的类型为 int，因此返回值必须为一个整型值。一般而言，返回值为 0 表示正常返回。

上面程序中的# include <stdio.h>表示把尖括号<>内的 stdio.h 文件包含到本程序中。stdio 为 standard input/output 的缩写，即标准输入输出。C 语言中有关输入输出函数的格式均定义在这个文件里。

例 1.2　计算两个整数之和，并在屏幕上显示计算结果。

程序代码如下：

```
#include <stdio.h>
int main ()                          /*主函数*/
{ int a,b,sum;                       /*定义变量*/
   a=111;   b=222;                   /*为变量赋值*/
   sum=a+b;                          /*求两数之和*/
   printf ("sum is: %d", sum);       /*输出 sum 的值*/
   return 0;
}
```

程序运行的结果是输出两个整数的和 sum，显示结果如下：

```
sum is: 333
```

在上面程序中，/*…*/表示注释部分。为了便于理解，程序员可用汉字表示注释，当然也可以用英语或汉语拼音作注释。注释只是用于解释程序，对编译和运行不起任何作用。

在上面程序中，在函数体内(即一对大括号之间)的第一行是变量定义部分，定义了 3 个 int 型变量；第二行是两个赋值语句，使变量 a 和 b 的值分别为 111 和 222；第三行使变量 sum 的值为 a 和 b 之和；第四行 printf 是输出函数，其中的%d 表示输出 sum 时的数据类型和格式为“十进制整数类型”。在执行输出时，此位置上代以一个十进制整数值，printf 函数中括弧内最右端的 sum 是要输出的变量，现在它的值是 333，因此输出的计算结果为“sum is: 333”。

例 1.3　输入变量 a 和 b 的值，调用自定义函数计算 a 和 b 的和，并在屏幕上输出结果。

程序代码如下：

```
#include <stdio.h>
int sumab (int x, int y);                    /*函数声明*/
int main ()                                  /*主函数*/
{ int a,b,sum;                               /*定义变量*/
   printf("请输入变量 a 与 b 的值:");          /*提示信息*/
   scanf ("%d   %d",   &a, &b);              /*输入变量 a 和 b 的值*/
   sum=sumab(a,b);                           /*调用 sumab 函数*/
   printf("a 与 b 的和等于%d", sum);          /*输出 sum 的值*/
   return 0;
}
int sumab (int x, int y)                     /*定义 sumab 函数，并定义形参 x、y */
{ int z;
   z=x+y;
   return z;
}
```

该程序由两个函数组成，即由主函数 main 和函数 sumab 组成。

函数 sumab 的功能是求两个整数之和并返回给主函数。它是一个用户自定义函数，它有两个 int 型的形参 x 和 y，它是一个具有 int 型类型返回值的函数。main 函数前面的函数声明语句“int sumab (int x, int y);”表明 sumab 是一个有两个 int 型的形参并返回一个 int 型值的函数。这样的函数声明叫作函数原型，它要与函数的定义和调用相一致。

本程序的执行过程如下：首先在屏幕上显示提示字符串“请输入变量 a 与 b 的值：”，请用户输入两个数；用户输入两个数并按回车后，由 scanf 函数语句接收这两个数并存入变量 a、b 中；然后调用 sumab 函数，把 a 和 b 的值传递给 sumab 函数的参数 x 和 y。在 sumab 函数中，计算 x 和 y 之和并赋给变量 z，并由 return 语句把变量 z 的值返回给主函数 main，然后赋值给变量 sum，最后由 printf 函数在屏幕上输出 sum 的值。

从以上 3 个例子可以看出，C 源程序的结构特点如下。

(1) 一个 C 语言源程序由一个或多个源文件组成。每个源文件由一个或多个函数构成，其中有且仅有一个主函数(main 函数)。

(2) 一个函数由函数首部(即函数的第一行)和函数体(即函数首部下面的大括号内的部分)组成。函数首部包括函数类型、函数名和放在圆括号内的若干个参数。函数体由声明部分和执行部分组成。

(3) C 程序书写格式自由，一行内可以写多条语句，一个语句也可以分写在多行中，每个语句必须以分号结尾。

(4) 程序的注释内容放在“/*”和“*/”之间，在‘/’和‘*’之间不允许有空格；注释部分允许出现在程序中的任何位置。

1.4　C 程序的编辑、编译、链接和运行

1. 编辑程序

用编辑软件将 C 源程序输入计算机，经修改无误后，保存为一个文件。C 源程序文件

的后缀为“.c”。可用于编写 C 源程序的编辑软件有很多，DOS 环境下，可以使用 Turbo C；Windows 环境下，可以使用 WIN TC(Turbo C 的 Windows 版本)，也可以使用 C-Free，还可以使用 Visual C++。

2. 编译程序

在 WIN TC 或 C-Free 或 Visual C++下，将后缀为“.c”的源程序编辑并保存之后，通过快捷键或者选择菜单的方式进行编译，编译的过程是把 C 源程序代码转换为计算机可识别的代码。如果在编译过程中发现源程序有语法错误，则系统会显示出错信息，告诉用户源程序有错误，然后用户可以重新修改源程序再进行编译，如此反复直至编译通过为止。编译通过后生成目标程序，目标程序的文件名与源程序相同，但后缀为“.obj”。

3. 链接程序

链接程序是指将目标程序和库函数或其他目标程序相链接。在 WIN TC 或 C-Free 或 Visual C++下通过快捷键或选择菜单的方式进行链接，即可以生成可执行程序。可执行程序的文件名与源程序相同，但后缀为“.exe”。

4. 运行程序

只要输入可执行文件的文件名即可运行程序。在 C-Free 或 WIN TC 或 Visual C++下通过快捷键或选择菜单的方式即可运行程序。

上述的编辑、编译、链接、运行程序的过程如图 1.1 所示。

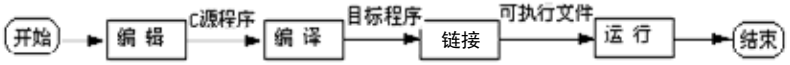

图 1.1　C 程序的执行过程示意图

1.5 习　　题

1. 简述 C 程序的结构特点。

2. 分析例 1.3 程序的结构。

3. 分别编写完成如下任务的程序，然后上机编译、链接并运行。

(1) 输出两行字符，第 1 行是“The computer is our good friends!”，第 2 行是“We learn C language.”。

(2) 从键盘输入变量 a、b 的值，分别计算 a+b、a–b 的值，将计算结果分别存放在变量 c、d 中，最后输出计算结果。

第2章　数据类型、运算符与表达式

本章主要介绍 C 程序中经常用到的常量、变量、基本数据类型(整型、实型、字符型)、运算符(算术运算符、赋值运算符、强制类型转换运算符、自增自减运算符、逗号运算符、求字节数运算符)和表达式(算术表达式、赋值表达式、逗号表达式)等内容。

2.1　C 语言的数据类型

对于在程序中要使用的每一个数据，都要指定其数据类型。C 语言提供了如图 2.1 所示的数据类型。

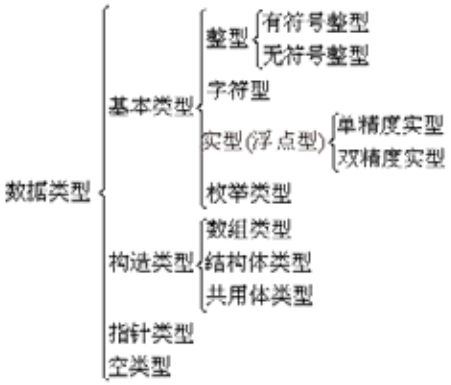

图 2.1　C 语言数据类型

本章将介绍整型、实型和字符型数据的用法。

2.2　常量和变量

2.2.1　常量

在程序运行过程中，其值不能被改变的量称为常量。

常量分为以下几种：

(1) 整型常量(如 369、0、–547)

(2) 实型常量(如 2.71828、–9.8、3.14159)

(3) 字符常量(如'A'、'a'、'#'、'3')

(4) 符号常量

可以用一个标识符代表一个常量。例如，在程序开始若有这样的预处理命令# define N

10，那么 C 预处理程序会将程序中的 N 用 10 代替。

2.2.2　变量

在程序运行过程中，其值可以被改变的量称为变量。

在使用某变量之前，必须先定义该变量，就是为该变量命名并声明其数据类型。根据定义，编译系统在内存中为该变量分配存储单元，在该存储单元中存放该变量的值。

用来标识变量名(或符号常量名、函数名、数组名、类型名、文件名)的有效字符序列称为标识符。C 语言规定，标识符只能由英文字母、数字和下画线这 3 种字符组成，并且第一个字符必须是字母或下画线。

注意：

大写英文字母和小写英文字母是不同的字符。例如，aver 和 Aver 是两个不同的标识符。在命名变量时一般用小写英文字母。

变量定义的一般格式如下：

```
[存储类型]　数据类型　　变量名 1[,  变量名 2……];
```

示例代码如下：

```
int    a, b, number, sum;
```

在定义变量的同时，对变量进行赋初值的操作称为变量初始化。变量初始化的一般格式如下：

```
[存储类型]  数据类型　变量名 1[=初值 1][,  变量名 2[=初值 2]……];
```

示例代码如下：

```
int    width=25, length=36, area;
```

注意：

格式中放在中括号内的内容可以省略，本书后文中都是如此。

2.3　整 型 数 据

2.3.1　整型常量

整型常量即整常数，在 C 语言中，整型常量可以用如下 3 种形式表示。

(1) 十进制，如 456、0、–789。

(2) 八进制(以数字 0 开头)，如 0123，即$(123)_8$，等于十进制的 83。

(3) 十六进制(以数字 0+小写字母 x 开头)，如 0x23，即$(23)_{16}$，等于十进制的 35。

2.3.2　整型变量

整型变量可分为有符号整型变量和无符号整型变量两大类，根据变量的取值范围，每类可分为基本整型、短整型、长整型三种类型。

归纳起来，共有六种整型变量，如下所示：

有符号基本整型　[signed] int
有符号短整型　[signed] short [int]
有符号长整型　[signed] long [int]
无符号基本整型　unsigned [int]
无符号短整型　unsigned short [int]
无符号长整型　unsigned long [int]

其中，方括弧内的部分可以省略，如 unsigned long [int]与 unsigned long 等价。

例如，下面分别定义了有符号基本整型变量 a 和 b、无符号长整型变量 c 和 d：

```
int a, b;
unsigned long c, d;
```

数据在内存中是以二进制形式存放的。若不指定是无符号型 unsigned 或者指定是有符号型 signed，则存储单元的最高位是符号位(0 为正，1 为负)。若指定是无符号型 unsigned，则存储单元的全部二进制位(bit)都用来存放数本身，而不包括符号。

整型数以二进制补码形式存放于内存中。

对于二进制正数，它的原码、反码和补码都相同。例如，若定义“short a=9; ”，则 a 的原码、反码和补码都是 0000000000001001。

对于二进制负数，它的原码是这样的：符号位是 1，数值部分用二进制的绝对值表示；它的反码是这样的：将其原码除符号位外，其余各位按位取反，即 1 都换成 0，0 都换成 1；它的补码是这样的：将其反码在最低位加 1。

例如，若定义“short b= –9; ”，则 b 的原码、反码和补码分别是：1000000000001001、1111111111110110、1111111111110111。

C 标准没有具体规定以上各类数据所占内存大小，只要求 long 型数据不短于 int 型，short 型不长于 int 型，怎样实现由计算机系统自行决定。例如，在微机上，使用 Turbo C 软件时，short 型和 int 型各占 2 个字节，long 型占 4 个字节。使用 Visual C++和 C-Free 软件时，short 型占 2 个字节， int 型和 long 型各占 4 个字节。

对于有符号整型变量，2 个字节的取值范围在-2^{15} 至$(2^{15}-1)$之间，即在–32768 至 32767 之间；4 个字节的取值范围在-2^{31} 至$(2^{31}-1)$ 之间，即在–2147483648 至 2147483647 之间。

对于无符号整型变量，2 个字节的取值范围在 0 至$(2^{16}-1)$ 之间，即在 0 至 65535 之间；4 个字节的取值范围在 0 至$(2^{32}-1)$ 之间，即在 0 至 4294967295 之间。

为整型变量赋值时，若超出了规定的取值范围，就会发生“溢出”现象，而程序运行时并不报错。因此，要根据实际情况，准确选择变量的类型，避免超出取值范围。

2.3.3　整型数据的输入输出

可以使用 scanf 函数和 printf 函数进行数据的输入与输出。

scanf 函数的功能是按照指定格式将标准输入设备输入的内容送入变量中，printf 函数的功能是按照指定格式在标准输出设备上显示数据。“指定格式”需要使用格式说明符%和格式字符，显示整型数的格式字符有英文字母 d、o、x、u 等。

具体含义如下：

%d——表示把数据按十进制整型输入(输出)；

%o——表示把数据按八进制整型输入(输出)；

%x——表示把数据按十六进制整型输入(输出)；

%u——表示把数据按无符号整型输入(输出)。

除了%d 格式外，上面的其他几种格式都将数据作为无符号数进行输入(输出)。

如果输入(输出)的是长整型数，一定要在转换字符的前面加上字符 l(字符 L 的小写)，例如使用%ld 输入(输出)十进制长整型。

例 2.1　整型数据的输出。

```
#include <stdio.h>
int main()
{ int a=200,b=100,c;
   c=a+b+15;
   printf("%d,%d,%d,%d\n", a,b,c,a–b–70);
   printf("%o,%o,%o,%o\n", a,b,c,a–b–70);
   printf("%x,%x,%x,%x\n", a,b,c,a–b–70);
   return 0;
}
```

输出结果如下：

```
200, 100, 315, 30
310, 144, 473, 36
C8, 64, 13b, 1e
```

例 2.2　整型数据的输入。

```
#include <stdio.h>
int main()
{ int a,b,c;   unsigned d；long e;
   scanf("%d,%o,%x,%u,%ld ", &a,&b,&c,&d,&e);
   printf("%d,%d,%d,%u,%ld \n", a,b,c,d,e);
   return 0;
}
```

若输入为：　　10, 10, 10, 65533, 654321 ↙(回车符)

则输出结果为：　10, 8, 16, 65533, 654321

2.4 实 型 数 据

2.4.1 实型常量

实数又称为浮点数，有以下两种表示形式。

(1) 十进制小数形式：由数字和小数点组成(必须有小数点)，如 3.14159、0.0、9.0、.12345、–9.8 等。

(2) 指数形式：<尾数>E(或 e)<指数>，如 1.23E003、2.71e–005(分别代表 1.23×10^{3}、2.71×10^{-5})等。注意：E(或 e)的两边必须有数字，且后面的指数必须是整数。

一个实数有多种指数形式。例如，314.159 可以表示为 3141.59E–001、314.159E000、3.14159E+002、0.314159E+003 等。其中的 3.14159E+002 被称为“规范化的指数形式”，即小数点左边有且只有一位非零数字。

2.4.2 实型变量

实型变量分为单精度型和双精度型，有的 C 版本还支持长双精度型(long double)。

(1) 单精度型：类型说明符为 float，在内存中占 4 个字节(32 位)，有效数字的个数是 7 位十进制数字，数值范围为-3.4×10^{-38}~3.4×10^{38} 。

(2) 双精度型：类型说明符为 double，在内存中占 8 个字节(64 位)，有效数字的个数是 15 位十进制数字，数值范围为-1.7×10^{-308}~1.7×10^{308} 。

2.4.3 实型数据的输入输出

可以使用%f 和%e 来控制输入(输出)float 类型的数据，使用%lf 和%le 来控制输入(输出) double 类型的数据。

例 2.3　实型数据的输入输出。

```
#include <stdio.h>
int main()
{ float a,b; double x,y;
   scanf("%f,%e,%lf,%le", &a,&b,&x,&y);
   printf("%f,%e,%lf,%le \n", a,b,x,y);
   return 0;
}
```

若输入为：

```
3.1415, 314.15, 123.456, 12345.6 ↙
```

则输出结果为：

```
3.141500, 3.141500e+002, 123.456000, 1.234560e+004
```

若输入为：

3.1415926, 567.456789, 123456789.123456789, 123456.7898765 ↙

则输出结果为：

3.141593, 5.674568e+002, 123456789.123457, 1.234568e+005

从结果可知：对于十进制小数形式，单精度型和双精度型的有效数字分别是 7 位和 15 位。对于十进制指数形式，都是 7 位有效数字。

2.5　字符型数据

2.5.1　字符型常量

C 语言的字符型常量是用一对单引号括起来的单个字符。例如，'A'、'3'、'a'、'?'等都是字符型常量。注意'A'和'a'是不相同的。

还有一种特殊形式的字符型常量，就是以转义符'\'开头的一些字符构成的转义序列。例如，'\n'表示“回车换行”。常见的转义字符如表 2.1 所示。

表 2.1　常见的转义字符及其含义

字 符 形 式	含　义
\a	警告声
\b	退格，将当前位置移到前一列
\f	换页，将当前位置移到下一页开头
\n	换行，将当前位置移到下一行开头
\r	回车，将当前位置移到本行开头
\t	横向跳格，跳到下一个 tab 位置
\\	反斜线字符
\’	单撇号字符
\”	双撇号字符
\ddd	1 到 3 位 8 进制数所代表的字符
\xhh	1 到 2 位 16 进制数所代表的字符
\0	字符串终止字符

表中\ddd 表示 1 到 3 位 8 进制数所代表的字符，如'\101'代表字符'A'，'\43'代表字符'#'，'\77'代表字符'？'等。

表中\xhh表示1到2位16进制数所代表的字符，如'\x61'代表字符'a'，'\x23'代表字符'#'。

表中\t 表示横向跳格，跳到下一个制表位置，一个制表区占 8 列，执行'\t'就是将当前位置跳到下一个制表区的开头。

注意'\r'和'\n'区别，'\r'是将当前位置移到本行开头，'\n'是将当前位置移到下一行开头。

例 2.4　转义字符的使用。

```
#include <stdio.h>
int main()
{ printf("A\102\x43\\DE\t\b\b\x23\43\x61\x62\n");
   printf("\'A\'\53\"\101\"\t\b\43\141\142\x63\b\x64\n");
   return 0;
}
```

输出结果如下：

```
ABC\DE##ab
'A'+"A"#abd
```

2.5.2　字符串常量

字符串常量是用一对双引号括起来的若干个字符，如"How are you"，"No.345"等。C 编译程序在存储字符串常量时自动采用字符'\0'作为字符串结束标志。字符'\0'的 ASCII 码值为 0，它不引起任何控制动作，也不是一个可显示的字符。因此，字符串"Good"在内存中占 5 个字节数，而不是 4 个，如图 2.2 所示。

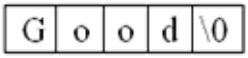

图 2.2　存储字符串

注意：

'A'和"A"是不同的。'A'是一个字符常量，在内存中占 1 个字节数；而"A"是字符串常量，在内存中占 2 个字节数，包含'A'和'\0'这两个字符。

2.5.3　字符型变量

字符型变量的类型说明符为 char。例如，“char c1,c2；”定义了两个字符型变量。

字符型变量用来存储字符常量，一个字符型变量只能存储一个字符，即只能存储一个字节的信息，就是说一个字符型变量在内存中占一个字节。例如，可以用如下语句给上面定义的字符变量 c1、c2 赋值：

```
c1='A'；c2='B';
```

将一个字符常量放到一个字符型变量中，实质上是将该字符常量对应的 ASCII 代码放到了字符型变量中，系统为字符型变量所分配的一个字节的存储单元中，存放的是该字符常量的二进制形式的 ASCII 码。例如，'A'的 ASCII 码是 65，65 的二进制形式是 01000001，所以系统为 c1 所分配的一个字节中，存放的是 01000001。

2.5.4　字符数据的输入输出

可以使用%c 控制输入(输出)char 类型的数据。

例 2.5　字符型变量值的输入输出。

```
#include <stdio.h>
int main()
{ char c1,c2,c3='P';   scanf("%c", &c1);     c2='D';
   printf("%c%c%c" ,c1,c2,c3);
   printf("，%c，%c，%c \n " ,c1,c2,c3);
   return 0;
}
```

若输入为：

```
G ↙
```

则输出结果为：

```
GDP, G, D, P
```

从例 2.5 可以看到为字符型变量赋值的不同方式。

例 2.6　将大写英文字符转换为小写英文字符。

```
#include <stdio.h>
int main()
{ char c1,c2;
   printf("请输入两个大写英文字符：");
   scanf("%c, %c",&c1,&c2);
   printf("%c%c", c1, c2);
   c1=c1+32;   c2=c2+32;
   printf("%c, %c\n " , c1, c2);
   return 0;
}
```

若输入为：

```
A, B ↙
```

则输出结果为：

```
AB      a, b
```

执行“scanf("%c,%c ",&c1,&c2);”语句，从键盘输入字符 A 和 B 后，c1 的值为'A'(ASCII 码值是 65)，c2 的值为'B'(ASCII 码值是 66)。

执行“printf("%c%c" ,c1,c2);”语句后，屏幕上显示了 AB。

接着执行“c1=c1+32;c2=c2+32;”两个语句，执行后 c1 的 ASCII 码值变为 97，c2 的 ASCII 码值变为 98。97 和 98 分别是字符'a'和'b'的 ASCII 码值，即 c1 和 c2 中存储的分别是'a'、'b'。

所以，最后执行“printf("%c,%c \n ",c1,c2);”语句后，屏幕上显示的是“a,b”。

同样地，也可以实现小写英文字符到大写英文字符的转换。

2.6　算术运算符和算术表达式

2.6.1　算术运算符

基本的算术运算符有如下 5 种：

- +：加法运算符或正值运算符，如 13+25、+9。
- –：减法运算符或负值运算符，如 32–15、–2。
- *：乘法运算符，如 4*7、5.6*7.8。
- /：除法运算符，如 32/5、1.23/3.45。
- %：求余数运算符，或称取模运算符，如 8%5 的值为 3。

关于除法运算符/，若是两个整数相除，其商为整数，小数部分被舍弃。例如，5/2 的结果不是 2.5，而是 2；12/24 的结果是 0。若除数和被除数中有一个是浮点数(实数)，则与数学的运算规则相同。例如，6/4.0、6.0/4、6.0/4.0 的结果都是 1.5。

关于求余数运算符%，要求%两侧的操作数均为整型数据，结果的符号与%左边的符号相同。例如，16%4 的结果是 0，–17%4 的结果是–1，18%–4 的结果是 2，–19%–4 的结果是–3。可以使用%运算符来判断一个数能否被另一个数整除。

2.6.2　算术表达式

1. 算术表达式的概念

用算术运算符和圆括号将运算对象(常量、变量和函数等)连接起来的、符合 C 语言语法规则的式子，称为 C 算术表达式。

单个常量、变量或函数，可以看作是表达式的一种特例。

例如，数学表达式(2x+3y)÷(4xy)，写成 C 语言的算术表达式，应该是(2*x+3*y)/(4*x*y)或(2*x+3*y)/4/x/y，而不能是(2x+3y)/(4xy)，也不能是 2*x+3*y/(4*x*y)，也不能是(2*x+3*y)/4*x*y。

算术表达式的结果不应该超过其能表示的数的范围。例如，short 型数的范围是–32768 至 32767，下面程序中的算术表达式 x+y 的值超过了 32767，赋给 z 显然是错误的。

```
main()
{short   x, y, z;
   x=31500; y=24600;   z=x+y;
   printf("%d", z);
}
```

如果将 x、y 和 z 定义为 long 型，就没有问题了。

2. 算术运算符的优先级与结合性

表达式求值时，按运算符的优先级别高低，按次序执行。算术运算符的优先级是：先乘除，后加减；求余运算的优先级与乘除相同；函数和圆括号的优先级最高。

所谓结合性，是指当一个操作数两侧的运算符具有相同的优先级时，该操作数是先与左边的运算符结合，还是先与右边的运算符结合。自左至右的结合方向，称为左结合性；反之，称为右结合性。

算术运算符的结合方向是“自左至右”。例如，在执行 a–b+c 时，变量 b 先与减号结合，执行 a–b；然后再执行+c 运算。

2.6.3　不同数据类型间的混合运算

在 C 语言中，整型、实型和字符型数据之间可以进行混合运算。

如果一个运算符两侧操作数的数据类型不同，则系统按“先转换、后运算”的原则，首先将数据自动转换为同一类型，然后在同一类型数据间进行运算。

有两种转换方式：自动转换和强制转换。

1. 自动转换

自动转换就是系统根据规则自动地将两个不同数据类型的运算对象转换为同一数据类型。自动转换又称隐式转换。自动转换的规则如图 2.3 所示。

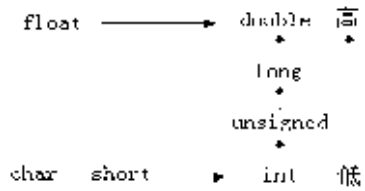

图 2.3　自动转换规则

在图 2.3 中，横向向右的箭头表示的是必须转换。char 和 short 型必须转换成 int 型参与运算，float 型必须转换成 double 型参与运算(即使是两个 float 型数据相加，也要先转换成 double 型，然后再相加)。

在图 2.3 中，纵向箭头表示的是当运算对象为不同类型时转换的方向。例如，若 int 型与 double 型数据进行混合运算，则先将 int 型数据转换成 double 型，然后进行运算，结果为 double 型。纵向箭头的方向只是表示数据类型的高低，由低向高转换，不要理解为 int 型先转换成 unsigned 型，然后再转换成 long 型，然后再转换成 double 型。

注意：

自动转换只是针对一个运算符两侧的两个运算对象，而不能对表达式中的所有运算符涉及的运算对象做一次性的自动转换。例如，表达式 6.0/5+4.32 和表达式 6/5+4.32，前者的值是 5.52，后者的值是 5.32。因为 6.0/5 是先将 5 转换成实型后再进行运算，值是 1.2，然后与 4.32 相加，值是 5.52。而 6/5 是按 int 型运算，值是 1，再与 4.32 相加，值是 5.32；不要理解成将 6/5+4.32 中的每个数都转换成实型后再运算。

2. 强制转换

编写程序时，可以利用强制类型转换运算符将一个表达式的值转换成所需类型，强制

转换的格式如下：

(类型名)(表达式)

示例如下：

(float)a　　将 a 转换成 float 型，注意不能写成 float(a)。
(int)3.45　　将 3.45 转换成 int 型。
(double)(15%6)　将 15%6 的值转换成 double 型。
(float)(x+y)　　将 x+y 的值转换成 float 型，注意不能写成(float)x+y。

2.7　赋值运算符和赋值表达式

2.7.1　赋值运算符

1. 普通赋值运算符

普通赋值运算符是“=”，作用是将运算符右侧表达式的值赋给运算符左侧的变量。

例如，“x=1.23;”的作用是将常量 1.23 赋给变量 x；“y=3*x+5.26;”的作用是将表达式 3*x+5.26 的值赋给变量 y；“x= x+1;”的作用是将变量 x 原来的值加 1 后再赋给变量 x，若变量 x 原来的值是 2，则执行 x= x+1 后，变量 x 的值是 3。

2. 复合赋值运算符

复合赋值运算符是在普通赋值运算符=的前面加上其他运算符。复合算术赋值运算符有如下 5 个：

+=、–=、*=、/=、%=

另外，还有 5 种复合赋值运算符(<<=、>>=、&=、∧=、|=)，将在后面章节中介绍。

复合算术赋值运算符的使用规则为：Xop=Y 等价于 X=XopY，其中 X 代表被赋值的某个变量，op 代表+或–或*或/或%，Y 代表某个表达式。例如，下面左边的代码等价于右边的代码：

```
a+=9        a=a+9
b*=c+5      b=b*(c+5)       (注意，不等价于 b=b*c+5)
d/=2*e–7    d=d/(2*e–7)     (注意，不等价于 d=d/2*e–7)
```

2.7.2　赋值表达式

由变量、赋值运算符和表达式连接起来的式子称为赋值表达式。赋值表达式的值就是被赋值的变量的值。

例如，“a=123”是一个赋值表达式，“a=123”这个赋值表达式的值就是 a 的值，而 a 的值是 123，所以“a=123”这个赋值表达式的值就是 123。

“b+=456”也是一个赋值表达式，“b+=456”这个赋值表达式的值就是 b 的值。因为

“b+=456”等价于“b=b+456”，若 b 的初值是 300，则执行“b=b+456”后，b 的值是 756，所以“b+=456”这个赋值表达式的值就是 756。

下面是赋值表达式的其他几个例子：

```
x=(y=23)+(z=17)–8;
(x 的值是 32, 所以赋值表达式的值是 32)
x/=8*(y=2) ;
(若 x 的初值是 32, 执行 x/=8*(y=2)后, x 的值是 2, 所以赋值表达式的值是 2)
y1=y2=y3=8;
(执行 y1=y2=y3=8 后, y1、y2、y3 的值都是 8, 所以赋值表达式的值是 8)
```

赋值表达式的后面加上分号“；”，就成为了赋值语句。

赋值表达式也可以在赋值语句之外的其他语句中出现。示例如下：

```
if ((ch=getchar())!= '\n')    printf("%c",ch);
```

上面语句中出现了赋值表达式“ch=getchar()”(函数 getchar()在第 3 章介绍)，ch 的值就是赋值表达式 ch=getchar()的值。该语句的含义是：若 ch 值不等于'\n'，则输出 ch 的值。

2.7.3　赋值表达式的类型转换

当赋值运算符左边的变量数据类型与右边的表达式的数据类型不相同时，需要进行数据类型转换。系统会把右边的数据转换成左边数据类型的数据。

转换后可能会发生数据丢失现象。例如，左边为 short 型，右边为 long 型，由于 long 型在内存中所占二进制位数(32 位)大于 short 型在内存中所占二进制位数(16 位)，造成 long 型的高 16 位无法复制到 short 型变量中，因此可能丢失数据。同理，左边为 float 型，右边为 double 型，也可能丢失数据。

下面分几种情况讨论。

1. 整型和字符型之间的转换

(1) 字符型数据赋给整型变量

由于字符型数据在内存中占 8 位，而整型变量在内存中至少占 16 位，因此，将字符型数据的 8 位放到整型变量的低 8 位中。对整型变量其他位的处理：有的系统是对整型变量其他位补 0；有的系统是根据字符型数据的最高位的值来决定补 1 还是补 0。

例如，Visual C++和 C-Free 以及 Turbo C 是根据字符型数据的最高位的值来决定补 1 还是补 0。若字符型数据的最高位是 0，则对整型变量其他位补 0；若字符型数据的最高位是 1，则对整型变量其他位补 1。

(2) 整型(int 或 short 或 long)数据赋给字符型变量

由于字符型数据在内存中占 8 位，所以只将整型数据的低 8 位送到字符型变量中。

例如，若将十进制 short 型数据 322 赋给字符型变量 ch，因为 322 的二进制形式是 0000000101000010，它的低 8 位是 01000010(十进制形式是 66)，所以字符型变量 ch 的值的二进制形式是 01000010(即 66)，若执行“printf("%c",ch);”语句则输出字符'B'，因为'B'的 ASCII 码是 66(十进制)。

例如，若将十进制 short 型数据 65 赋给字符型变量 ch，因为 65 的二进制形式是 0000000001000001，它的低 8 位是 01000001(十进制形式也是 65)，所以字符型变量 ch 的值的二进制形式是 01000001，若执行“printf("%c",ch);”语句，将输出字符'A'。

2. 整型之间的转换

(1) short 型数据赋给 long 型变量

将 short 型数据的 16 位二进制代码送到 long 型变量的低 16 位中，如果 short 型数据值为正(符号位是 0)，则 long 型变量的高 16 位补 0；如果 short 型数据值为负(符号位是 1)，则 long 型变量的高 16 位补 1。高 16 位补 0 或 1 称为符号扩展。

(2) long 型数据赋给 short 型变量

只将 long 型数据中的低 16 位送到 short 型变量中。

(3) unsigned short 型数据赋给 long int 型变量

此时不存在符号扩展问题，只需将 long int 型变量的高位补 0。

(4) unsigned 型数据赋给占二进制位数相同的其他整型变量

将 unsigned 型数据的内容原样送到其他整型变量中，如果范围超过其他整型变量允许的范围，则会出错。例如，若 a 是 unsigned short 型变量，a=65535，而 b 是 short 型变量，若执行“b=a;”，由于 a 的二进制形式是 1111111111111111，所以 b 的二进制形式也是 1111111111111111，由于最高位(符号位)是 1，因此 b 成了负数，根据补码知识，可知 b 是–1，执行“printf("%d",b);”将输出–1。

(5) 非 unsigned 型的整型数据赋给占二进制位数相同的 unsigned 型变量

此时也是原样照赋(最高的符号位也一起传送)。例如，若 a 是 unsigned short 型变量，b 是 short 型变量，b=–1。若执行“a=b;”，由于 b 的二进制形式是 1111111111111111，所以 a 的二进制形式也是 1111111111111111，执行“printf("%d",a);”，将输出 65535。

其他各种整型间的转换，方法与上面基本相同。即：将占位数少的变量赋值给占位数多的变量时，原值传送到占位数多的变量的低位，此时注意其他高位的符号扩展问题。而将占位数多的变量赋值给占位数少的变量时，可能丢失数据。

3. 实型与整型之间的转换

(1) 整型数据赋给实型变量

系统先将整型数据转换成单(或双)精度实型数据，保持数值不变，然后再赋值给实型变量。

(2) 实型数据赋给整型变量

单(或双)精度实型数据赋给整型变量时，舍弃实型数据的小数部分，将整数部分赋给整型变量。例如，若 n 是 int 型变量，执行“n=5.67;”的结果是 n 的值为 5，执行“printf("%d", n);”将输出 5。

4. 实型之间的转换

(1) float 型数据赋给 double 型变量

此时保持数值不变，存放到 double 型变量中，在内存中以 64 位二进制形式存储。

(2) double 型数据赋给 float 型变量

此时截取 double 型数据的前 7 位有效数字，存放到 float 型变量中，在内存中以 32 位二进制形式存储。此时可能会丢失数据，注意数值范围不要溢出。

2.8　其他运算符和表达式

2.8.1　自增、自减运算符

自增和自减运算符都是单目运算符，自增运算符(++)的作用是使变量的值增 1，自减运算符(– –)的作用是使变量的值减 1。

对于 int 型变量 i，++i 和 i++都等价于 i=i+1，– –i 和 i– –都等价于 i=i–1。

++i 和– –i 是前缀表示法，i++和 i– –是后缀表示法。前缀表示法是将 i 值先增/减 1，再将 i 在表达式中使用；后缀表示法是先在表达式中使用 i 的值，然后再将 i 值增/减 1。

例 2.7　自增、自减运算符的使用。

```
#include <stdio.h>
int main()
{ int i,j,k;   i=6;
  j=++i;   /*表达式++i 的值是 7 */
  k=i++;   /*表达式 i++的值是 7 */
  printf("%d,%d,%d\n " ,j,k,i);
  i=–6;
  j=– –i;   /*表达式– –i 的值是–7 */
  k=i– –;   /*表达式 i– –的值是–7 */
  printf("%d, %d, %d\n " ,j,k,i);
  return 0;
}
```

输出结果如下：

```
7, 7, 8
–7, –7, –8
```

需要注意以下几点。

(1) 自增、自减运算符，不能用于常量和表达式。例如，++6、– –(i+3*j)等都是非法的。

(2) 自增、自减运算符的优先级高于算术运算符，与单目运算符–(取负)和！(逻辑非)的优先级相同，结合方向自右至左。例如，–a++等价于–(a++)。

(3) 像“printf("%d,%d \n",i,i++);”这样出现“i,i++”的语句，在不同的编译系统中结果是不同的。若 i 的值是 6，按从左至右求值，输出“6,6”；按从右至左求值，输出“7,6”。Turbo C 和 C-Free 是按从右至左求值的，而 Visual C++是按从左至右求值的。

(4) 自增或自减运算符最好单独使用，避免自增或自减运算符与其他运算符混合使用。像 i++++++j 这样很难理解的表达式，应该避免使用。

2.8.2　逗号运算符和逗号表达式

C 语言还提供了逗号运算符，逗号将两个表达式连接起来，形成一个表达式，称为逗号表达式。它的一般形式如下：

表达式 1，表达式 2

逗号表达式的求值过程是：先求表达式 1 的值，再求表达式 2 的值，并将表达式 2 的值作为逗号表达式的值。

例如，逗号表达式“8–3,6+5”的值是 11，因为表达式 6+5 的值是 11。

再如，逗号表达式“k=2*3,++k”的值是 7，因为第一个表达式 k=2*3 的值是 6，k 的值也是 6，所以第二个表达式++k 的值是 7。注意，赋值运算符的优先级高于逗号运算符，所以“k=2*3,++k”是逗号表达式，千万不要将其理解为“k=(2*3,++k)”。

逗号表达式“a=6,a+=9”的值是 15。因为第一个表达式 a=6 的值是 6，a 的值也是 6，所以第二个表达式 a+=9(等价于 a=a+9)的值是 15。

一个逗号表达式可以与另一个表达式组成新的逗号表达式。例如，“(k=2*3,++k)，4*k”就是这样的逗号表达式。对于这样的逗号表达式，先计算逗号表达式“(k=2*3,++k)”的值，再计算表达式“4*k”，“4*k”的值也就是“(k=2*3,++k)，4*k”的值，而“4*k”的值是 28，所以这个逗号表达式的值是 28。

逗号表达式的一般形式可以扩展为如下形式：

表达式 1, 表达式 2, ……, 表达式 n

求这个逗号表达式的过程是：自左至右，依次计算每个表达式的值，最后计算出的表达式 n 的值即为整个逗号表达式的值。

例如，逗号表达式“3*5,6+4,10/2”和“n=3,n++,n*5%9”的值分别为 5 和 2。

注意：

并不是任何地方出现的逗号，都是逗号运算符。很多情况下，逗号仅用作分隔符。例如，函数的参数是用逗号分隔的。像输入函数“scanf("%d,%d",&x,&y);”中的逗号，以及输出函数“printf("%d,%d",x,y);”中的逗号，都是用作分隔符。

例 2.8　逗号表达式的使用。

```
#include <stdio.h>
int main()
{ int m,n,i,j,k=5;
   i=(j=6,j++,k+j);
   printf("%d, %d \n", (n=3*4, m=n+5), i);
   printf("%d, %d, %d, %d \n ", n, m, j, (n, m,j));
   return 0;
}
```

输出结果如下：

```
17, 12
12, 17, 7, 7
```

程序中的(n=3*4,m=n+5)是一个逗号表达式，值为 17；(n,m,j)也是一个逗号表达式，值为 7(变量 j 的值)。

2.8.3　求字节数运算符

求字节数运算符 sizeof 是一个比较特殊的单目运算符，用它可以求各种数据类型所占的字节数。某一个数据类型在不同的计算机系统中可能占用不同长度的内存空间，使用求字节数运算符 sizeof，就可以了解在自己所使用的计算机系统中，各种数据类型所占用的内存空间大小。

例 2.9　显示各种数据类型所占内存空间的字节数。

```
#include <stdio.h>
    int main()
    { int a; float b; double c; char d;
      printf("%d, %d, %d, %d \n",sizeof(a),sizeof(b),sizeof(c),sizeof(d));
      printf("%d, %d, %d\n ", sizeof(int),sizeof(unsigned int),sizeof(long int));
      printf("%d, %d, %d\n ", sizeof(float),sizeof(double),sizeof(char));
      return 0;
    }
```

使用 C-Free 和 Visual C++运行此程序，输出结果如下：

```
4, 4, 8, 1
4, 4, 4
4, 8, 1
```

而使用 Turbo C 运行此程序，输出结果如下：

```
2, 4, 8, 1
2, 2, 4
4, 8, 1
```

2.9　习　　题

1. 阅读程序，写出运行结果。

(1)

```
# include <stdio.h>
main()
{ int a,b,d=241;   a=d/100%9;   b=a*d;   printf("%d, %d", a, b); return 0;}
```

(2)

```
#include <stdio.h>
main()
{short a; unsigned short b=65535;   a=b; printf("%d,%d", a, b); return 0;}
```

(3)

```
# include <stdio.h>
main()
{ char c='A';    printf("%d,%o,%x,%c\n",c,c,c,c); return 0;}
```

(4)

```
# include <stdio.h>
main()
{float f=3.1415927;    printf("%f,%e\n",f,f); return 0; }
```

(5)

```
# include <stdio.h>
main()
{int i,j,x,y; i=5; j=7; x=++i; y=j++;    printf("%d,%d,%d,%d",i,j,x,y); return 0;}
```

(6)

```
# include <stdio.h>
main()
{long int a=123,b=456,c,d; c=--b; d=(a--, a+c); printf("%ld,%ld,%ld,%ld"a,b,c,d); return 0;}
```

2. 编写程序。

(1) 利用变量 k，将两个变量 m 和 n 的值交换。

(2) 输入一个整数 n，输出 n 除以 3 的余数。

(3) 输入一个三位整数 n，把 n 倒着输出(如输入 672，输出 276)。

(4) 输入一个三位整数 n，求 n 的三位数码之和。