目　　录

第 1 章　C 语言程序设计概述

1.1　C 语言简介

1.1.1　C 语言的发展历程和趋势

C 语言是在 BCPL（Basic Combined Programming Language，基本的汇编程序设计语言）基础上发展而来的，BCPL 的原型是 ALGOL 60（也称 A 语言）。ALGOL 60 是计算机发展史上的首批高级语言，更适用于数值计算。1963 年，剑桥大学将 ALGOL 60 发展成为 CPL（Combined Programming Language，汇编程序设计语言）。CPL 比 ALGOL 60 接近硬件，但规模比较大，实现困难。1967 年，剑桥大学的 Matin Richards 对 CPL 进行了简化，形成了 BCPL。1970 年，美国贝尔实验室的 Ken Thompson 在 BCPL 的基础上做了进一步的简化，设计出了简单且更加接近硬件的 B 语言（取 BCPL 的第一个字母），并且采用该语言编写了第一个 UNIX 操作系统，在 PDP-7 上实现。1973 年，贝尔实验室的 D. M. Ritchie 在 B 语言的基础上设计出了 C 语言。C 语言既保持了 BCPL 和 B 语言的精练、接近硬件的优点，又克服了它们过于简单、无数据类型的缺点。开发 C 语言的目的在于尽可能降低它所写的软件对硬件平台的依赖程度，使之具有可移植性。同年，Ken Thompson 和 D. M. Ritchie 合作把 UNIX 系统的 90％以上的代码用 C 语言重新编写，完成 UNIX 第 5 版。1977 年，D. M. Ritchie 发表了不依赖具体机器的 C 语言编译文本《可移植的 C 语言编译程序》，简化了 C 语言移植到其他机器上的工作，推动了 UNIX 操作系统在各种机器上的实现。随着 UNIX 的广泛使用，C 语言先后移植到大、中、小型机上，得以迅速推广，很快风靡全球，成为世界上应用最广泛的高级语言。

1978 年，Brian W. Kernighan 和 D. M. Ritchie 联合撰写了影响深远的名著 *The C Programming Language*，成为第一个事实上的 C 语言标准。1983 年，美国国家标准化协

会（ANSI）成立了专门的委员会，根据当时存在的不同 C 语言版本进行改进和扩充，指定了 C 语言标准草案——83 ANSI C。1987 年 ANSI 又公布了新的标准——87 ANSI C。1989 年，ANSI 公布了更加完整的 C 语言标准——ANSI X3（也称 ANSI C 或 C89）。1990 年，国际标准化组织 ISO（International Organization for Standardization）接受 C89 为 ISO C 的国际标准（ISO/IEC 9899：1990）。1995 年，ISO 修订了 C 语言标准，称为 1995 基准增补 1（ISO/IEC/9899/AMD1：1995）。1999 年，ISO 又对 C 语言标准进行了修订，在原有基础上，增加了 C++中的一些功能，命名为 ISO/IEC 9899：1999。2011 年 12 月 8 日，ISO 正式发布了 C 语言的新标准 C11，提高了对 C++的兼容性，并加入对多线程的支持等功能，命名为 ISO/IEC 9899：2011。

在实际的使用中，目前不同公司对 C 语言编译系统的开发，并未完全实现最新的 C 语言标准，它们多以 C89 为基础开发。

1.1.2 C 语言的特点

C 语言具有较强的生命力，与其他编程语言相比，有着自己的特点，其主要特点如下：

1. 语言简洁、结构清晰

C 语言一共有 32 个关键字，9 种控制语句，程序书写形式自由灵活。C 语言程序通常由多个函数组成，便于模块化和结构化编程，使得编写的程序结构清晰明了、可读性强。

2. 表达能力强

C 语言不仅提供了丰富的运算符和数据类型，还提供了强大的功能库，使得程序员可以快速、灵活地编写程序，精确地控制计算机按照自己的意愿工作。

3. 高效率的编译性语言

C 语言生成的目标代码质量高，运行速度快。对于较大的程序，源代码可以分别存放，单独编译后再链接在一起，形成可执行文件。

4. 可移植性好

采用 C 语言编写的程序基本上可以不做修改，直接运行于各种型号的计算机和各种操作系统中。

5. 运算符和数据类型丰富

C 语言包含 34 种运算符，运算符种类丰富，表达式类型多样，使用灵活。

6. 语法限制少，设计自由度大

C 语言允许程序员有较大的自由度，编译时放宽了语法检查。例如，对数组下标越界不进行检查，整型量与字符型量可以通用，等等。

7. 允许直接访问物理地址

C 语言既具有高级语言的功能，又具备低级语言的很多功能，使它既是通用的程序设计语言，又是系统描述语言。C 语言能够直接访问物理地址，还能够进行位运算，实现了汇编语言的大部分功能，可以直接对硬件进行操作。

1.2　C 语言程序举例

C 程序到底是什么样子的？让我们一起来看几个简单的 C 程序，并尝试读懂这些程序的功能。

例 1.1　在屏幕上输出信息“This is my first c program.”。

程序如下：

```
#include<stdio.h>      /*编译预处理指令*/
int main(  )       //定义主函数
{     /*函数的开始标志*/
printf("This is my first c program.\n");       /*输出指定信息*/
return 0;       /*函数正常结束返回值为 0*/
}       /*函数的结束标志*/
```

例 1.1 程序运行结果如图 1.1 所示，其中第 1 行“This is my first c program.”为程序运行后的结果。第 2 行为 Visual C++ 6.0 编译系统在输出运行结果后自动输出的信息，即“按任意键继续”，按下键盘上的任意键后，运行结果窗口消失。

```
This is my first c program.
Press any key to continue
```

图 1.1　例 1.1 的运行结果

例 1.2　求两个整数的和。

程序如下：

```
#include<stdio.h>      //编译预处理指令
int main(  ) //定义主函数，C 程序的开始
{     //函数的开始标志
int a, b, sum;       //声明 a，b，sum 均为整型变量
a=222;       //将整数 222 放在变量 a 中存储
b=333;       //将整数 333 放在变量 b 中存储
sum=a+b;       //将整数 a 和 b 的和放在变量 sum 中存储
printf("sum is %d\n", sum);       //输出 sum 的值
return 0;       //函数正常结束返回值为 0
}       //函数的结束标志
```

例 1.2 程序实现的功能是求两个整数的和，程序的运行结果如图 1.2 所示。

```
sum is 555
Press any key to continue
```

图 1.2　例 1.2 的运行结果

例 1.3 利用函数调用求两个数的和。

程序如下：

```
#include<stdio.h>      /*文件包含预处理命令*/
void main( )      /*主函数*/
{
int add (int x, int y);      /*对被调用函数 add 的声明*/
int a, b, sum;      /*定义变量 a, b 和 sum*/
printf ("Please input two number (like: 3, 5): \n"); /*使用 printf 函数打印输入提示*/
scanf ("%d,%d", &a, &b);      /*使用 scanf 函数对变量 a, b 赋值*/
sum=add (a, b);      /*调用函数 add, 并将函数返回值赋给 sum*/
printf ("%d+%d=d\n", a, b, sum);      /*使用 printf 函数输出结果*/
}
int add (int x, int y)      /*定义函数 add, x、y 为形式参数*/
{
retum (x+y);      /*将 x+y 的和返回, 通过 add 带回到调用函数的位置*/
}
```

例 1.3 程序实现的功能是求从键盘上输入的两个整数之和，程序的运行结果如图 1.3 所示。

```
Please input two number<like:3,5>:
24,32
24+32=56
Press any key to continue
```

图 1.3 例 1.3 的运行结果

现在让我们一起来分析程序，以便对 C 程序有一个初步的了解。

C 程序是由函数构成的，一个 C 源程序虽然有且仅有一个 main 函数，但可以包含多个其他函数，当然也可以没有其他函数，仅有一个 main 函数。

例 1.3 程序是由一个 main 函数和一个 add 函数组成的 C 程序，我们将从每一行代码出发，探讨隐藏在代码背后的细节。

1. 文件包含预处理指令

#include <stdio.h>是一个 C 预处理指令，该行指令告诉编译器包含文件 stdio.h 中的全部信息，其作用相当于在程序中该行指令所在的位置键入了 stdio.h 的完整内容。stdio.h 文件是标准输入/输出头文件（standard input/output header file），由 C 编译系统提供，它包含了有关输入和输出的函数（如 printf，scanf 等）的信息以供编译器使用。在 C 语言中，头文件通常以 .h 作为扩展名。当然文件包含预处理指令也可以包含用户定义的其他文件，它的基本格式为

#include<文件名>

或者

#include “文件名”

这两者之间的主要区别是：使用<>时，编译系统到存放 C 库函数头文件的目录中去寻找要包含的文件，而使用“”时系统先在用户当前目录中去寻找要包含的文件，若找不到再按照<>的方式进行查找。

2. main 函数

C 程序中必须有一个 main 函数，它表示 C 程序中的主函数，C 程序的执行总是从该函数开始。如果在 main 函数中调用了其他函数，调用结束后流程将返回到 main 函数，在 main 函数中结束整个程序的运行。

void 指明了 main 函数的返回类型。void 表示“空”，意味着 main 函数的返回类型是“空”，即不返回任何类型的值。有时，也可用 int main 声明 main 函数，它表明 main 函数的返回类型是整型。

3. 注释

在程序中，有很多诸如“/ * 文件包含预处理 * /”这样的内容，通过阅读这些内容可以很好地帮助我们理解这个程序。包含在/ * 和 * /之间的部分就是程序注释，用于对程序代码进行说明，让我们更容易理解程序代码实现的功能，便于程序开发人员对代码进行更好的维护。在/ * 和 * /之间的所有内容都将被编译器所忽略。在写程序注释的时候，可以单放一行或者是多行。下面是一个多行注释的例子。

/ * 注释的第一行，

这一行仍然是注释。 * /

也可采用“//”加注释内容的方式进行单行注释。下面是一个单行注释的例子。

//注释到本行结束

4. 大括号与函数体

在例 1.3 程序中，不管是 main 函数还是 add 函数，都有一对大括号“{}”，这对大括号划定了 main 函数和 add 函数的界限。在 C 语言中，所有的函数都使用大括号来表示函数的开始和结束，在大括号之间的部分，就是函数体。函数体由若干语句构成，这些语句描述了函数的功能。

一个函数名后面必须跟一对圆括号，圆括号中为参数列表。函数参数可以有多个，也可以没有。函数体是紧跟函数首部下面的大括号内的部分，包括了实现函数特定功能的变量定义及若干执行语句。有时，函数体也可以没有变量定义和执行语句，只有一对大括号。

函数在被其他函数调用时，需要先声明。例如：在例 1.3 程序中，main 函数中代码为

```
int add (int x, int y);
```

就是对被调函数 add 的一个声明。对函数的声明需与函数定义的函数名，形式参数的个数、类型和顺序都保持一致。当然，在函数声明的语句中，也可省略具体的形式参数的名字，只给出形式参数的类型。如：

int add (int, int);

如果被调函数的定义出现在主调函数之前，则可以不加声明，因为编译系统已经知道了定义的被调函数，并根据函数定义的相关信息对函数的调用作出了正确性检查。

5. 赋值

“sum=add (a, b);”这行程序除了调用函数外，还有一个赋值的功能，表示将 add 函数的返回值赋值给变量 sum。在该语句之前有一个定义变量的语句“int a, b, sum;”，它表明在计算机内存中为变量 a，b 和 sum 分配存储空间，而这个赋值语句则将数值存放在变量 sum 的存储空间中。我们也可以根据需要修改 sum 的值，这正是将 sum 称为变量的原因。需要注意的是，赋值运算符“=”的结合顺序为自右向左，即将“=”右边表达式的结果赋值给左边的变量。赋值语句后面的分号也是必不可少的，C 程序的每个语句后面都需要有一个分号。

练 习 题

一、填空题

1. 每个源程序有且只有一个____函数，系统总是从该函数开始执行 C 语言程序。

2. C 源程序的基本单位是_____。

3. 在一个 C 源程序中，多行注释部分两侧的分界符分别是____和____。

4. 在 C 语言中，输入操作是由库函数____完成的，输出操作是由库函数____完成的。

二、选择题

1. 下列说法中正确的是（　　）。

A. C 语言程序由主函数和 0 个或多个其他函数组成

B. C 语言程序由主程序和子程序组成

C. C 语言程序由子程序组成

D. C 语言程序由过程组成

2. 下列说法中正确的是（　　）。

A. C 语言程序总是从第一个定义的函数开始执行

B. 在 C 语言程序中，要调用的函数必须在 main 函数中定义

C. C 语言程序总是从 main 函数开始执行

D. C 语言程序中的 main 函数必须放在程序的开始部分

3. 在 C 语言程序中，每个语句必须以（　　）结束。

A. 回车符　　B. 逗号　　C. 冒号　　D. 分号

三、编程题

1. 请编写一个 C 程序，输出以下信息：

   *

   Hello，my first C program!

   *

2. 请编写一个 C 程序，从键盘上输入两个整数，求出这两个数的积，并将结果输出在屏幕上（提示：C 语言中的乘法运算符是“*”）。

第 2 章　C 语言初步知识

2.1　简单 C 语言程序的构成和格式

为了了解 C 语言程序的构成和格式，下面先看一个简单的 C 程序例子。

例 2.1　求矩形的面积。

程序如下：

```
#include <stdio.h>
main ( )
{double a，b，area;
  a=1.2;      /*将矩形的两条边长分别赋给 a 和 b*/
  b=3.6;
  area=a*b;     /*计算矩形的面积并存储到变量 area 中*/
printf ("a=%f，b=%f，area=%f\n"，a，b，area)；/*输出矩形的两条边长和面积*/
}
```

执行以上程序后的输出结果如下：

```
a=1.200000，b=3.600000，area=4.320000
```

以上程序中，main 是主函数名，C 语言规定必须用 main 作为主函数名。其后的一对圆括号中间可以是空的，但这一对圆括号不能省略。程序中的 main 是主函数的起始行，也是 C 程序执行的起始行。每一个可执行的 C 程序都必须有一个且只能有一个主函数。一个 C 程序中可以包含任意多个不同名的函数，但只能有一个主函数。一个 C 程序总是从主函数开始执行。

在函数的起始行后面用一对花括号“{}”括起来的部分为函数体。函数体内通常有定义（说明）部分和执行语句部分。以上程序中的“double a，b，area;”为程序的定义部分，从“a=1.2;”到“printf ("a=%f，b=%f，area=%f\n"，a，b，area);”是程序

的执行部分。执行部分的语句称为可执行语句，必须放在定义部分之后，语句的数量不限，程序中由这些语句向计算机系统发出操作指令。

定义语句用分号“;”结束。在以上程序中只有一个定义语句，该语句对程序中所用到的变量 a，b，area 进行定义，说明它们为 double 类型的变量。

程序中“a＝1.2;”和“b＝3.6;”的作用是分别给矩形的两条边赋值，“area＝a＊b;”的作用是计算出矩形面积并赋给变量 area，“printf（"a＝%f，b＝%f，area＝%f\n"，a，b，area);”的作用是按格式把 a，b 和 area 的值输出到屏幕。C 程序中的每一条执行语句都必须用分号“;”结束，分号是 C 语句的一部分，不是语句之间的分隔符。

C 语言程序有比较自由的书写格式，但是过于“自由”的程序书写格式往往使人们很难读懂程序，初学者应该从一开始就养成良好的习惯，使编写的程序便于阅读。

2.2　标识符、常量和变量

2.2.1　标识符

在 C 语言中，有许多符号的命名，如变量名、函数名、数组名等，都必须遵守一定的规则，按此规则命名的符号称为标识符。合法标识符的命名规则是：标识符可以由字母、数字和下画线组成，并且第一个字符必须为字母或下画线。在 C 语言程序中，凡是要求标识符的地方都必须按此规则命名。以下都是合法的标识符：

area，PI，_ini，a_array，s1234，PlOlp

以下都是非法的标识符：

456P，Cade－y，w.w，a&b

在 C 语言的标识符中，大写字母和小写字母被认为是两个不同的字符，例如 page 和 Page 是两个不同的标识符。

对于标识符的长度（即一个标识符允许的字符个数），C 语言编译系统是有规定的，即标识符的前若干个字符有效，超过的字符将不被识别。不同的 C 语言编译系统所规定的标识符有效长度可能会不同。有些系统允许取较长的名字，读者在命名标识符时应当了解所用系统的具体规定。

C 语言的标识符可以分为以下三类。

1. 关键字

C 语言已经预先规定了一批标识符，它们在程序中都代表着固定的含义，不能另作他用，这些标识符称为关键字。例如，用来说明变量类型的标识符 int，double 以及 if 语句中的 if，else 等都已有专门的用途，它们不能再用作变量名或函数名。

2. 预定义标识符

所谓预定义标识符是指在C语言中预先定义并具有特定含义的标识符，如C语言提供的库函数的名字（如 printf）和预编译处理命令（如 define）等。C语言允许把这类标识符重新定义另作他用，但这将使这些标识符失去预先定义的含义。鉴于目前各种计算机系统的C语言都一致把这类标识符作为固定的库函数名或预编译处理中的专门命令使用，因此为了避免误解，建议用户不要把这些预定义标识符另作他用。

3. 用户标识符

由用户根据需要定义的标识符称为用户标识符，又称自定义标识符。用户标识符一般用来给变量、函数、数组等命名。程序中使用的用户标识符除要遵守标识符命名规则外，还应注意做到“见名知义”，即选择具有一定含义的英文单词或汉语拼音作为标识符，如 number，red，yellow，green，work 等，以增加程序的可读性。

如果用户标识符与关键字相同，则在对程序进行编译时系统将给出出错信息；如果用户标识符与预定义标识符相同，系统并不报错，只是该预定义标识符将失去原定含义，代之以用户确认的含义，这样运行时有可能会引发一些错误。

2.2.2 常量

所谓常量是指在程序运行过程中，其值不能被改变的量。在C语言中，有整型常量、实型常量、字符常量和字符串常量等类型。整型常量还可以进一步分为短整型常量、长整型常量等。

整型常量和实型常量又称数值型常量，它们有正值和负值的区分。基本整型常量只用数字表示，不带小数点，例如 12，－1，0 等。实型常量必须用带小数点的数表示，例如 3.14159，－2.71828，0.0 等。‘A’和‘d’则是字符型常量，而“NCRE”和“Beijing”是字符串常量。由此可见，常量的类型从字面形式上是可区分的，C编译程序就是以此来确定常量类型的。

2.2.3 符号常量

在C语言程序中，可以用一个符号名来代表一个常量，称为符号常量。这个符号名必须在程序中进行特别的“指定”，并符合标识符的命名规则。

例 2.2 计算圆面积。

程序如下：

```
#include "stdio.h"
#define PI 3.14159
main（ ）
{double r，s；
    r＝5.0；
```

```
    s= PI * r * r;
    printf ("s=%f\n", s);
}
```

执行以上程序后的输出结果如下：

```
s=78.539750
```

程序中用 # define 命令行（注意：不是语句）定义 PI 代表一串字符 3.14159，在对程序进行编译时，凡本程序中出现 PI 的地方，编译程序均用 3.14159 来替换。为了使之比较醒目，这种符号名通常采用大写字母表示。用 define 进行定义时，必须用“#”作为一行的开头，在 # define 命令行的最后不得加分号。有关 # define 命令行的作用，将在后续章节中介绍。

2.2.4　变量

所谓变量是指在程序运行过程中其值可以改变的量。程序中用到的所有变量都必须有一个名字作为标识，变量的名字由用户定义，它必须符合标识符的命名规则。如例 2.1 中的 a，b 和 area 就是由用户定义的变量名。

一个变量实质上是代表了内存中的若干个存储单元。在程序中，变量 a 就是指用 a 命名的若干个存储单元，用户对变量 a 进行的操作就是对该存储单元进行的操作；给变量 a 赋值，实质上就是把数据存入该变量所代表的存储单元中。

C 语言规定，程序中所有变量都必须先定义后使用。对变量的定义通常放在函数体内的前部，但也可以放在函数的外部或复合语句的开头。

像常量一样，变量也有整型变量、实型变量、字符型变量等不同类型。在定义变量的同时要说明其类型，系统在编译时就能根据其类型为其分配相应的存储单元。

2.3　整 型 数 据

2.3.1　整型常量

在 C 语言程序中，整型常量可以用十进制、八进制和十六进制等形式表示。

十进制基本常量用一串连续的数字表示，例如 32767，－32768，0 等。

八进制数也是用一串连续的数字表示，但其开头必须是数字“0”。例如 010，011，016 等都是合法的八进制数，与之对应的十进制数为 8，9，14。因此，在 C 程序中不能在一个十进制数前随意添加数字“0”。例如，不能把十进制数 11 写成 011。注意：八进制数必须用合法的八进制数字表示。例如，不能写成 018，因为数字 8 不是八进制数字。

十六进制数用数字 0 和字母 x（或大写字母 X）开头。例如 0x10，0Xde，0xf 等都是合法的十六进制数，与之对应的十进制数分别为 16，222，15。注意：十六进制数必须用合法的十六进制数字表示。十六进制数中的字母 a，b，c，d，e，f 既可以用小写也可以用大写。

在 C 程序中，只有十进制数可以是负数，而八进制和十六进制数只能是正数或 0。

整型常量又有短整型（short int）、基本整型（int）、长整型（long int）和无符号型（unsigned）等不同类型。

2.3.2 整型变量

整型变量也分为短整型、基本型、长整型和无符号型等类型。本节只介绍基本型的整型变量。

基本型的整型变量用类型名关键字 int 进行定义，例如：

```
int k;        /*定义 k 为整型变量*/
```

一个定义语句必须以一个“;”号结束。在一个定义语句中也可以同时定义多个变量，变量之间用逗号隔开。例如：

```
int i，j，k；/*定义 i，j，k 为整型变量*/
```

不同的编译系统为 int 变量开辟的内存单元大小不同。VC 为 int 变量开辟 4 个字节（32 个二进制位）的内存单元，并按整型数的存储方式存放数据，允许存放的数值范围是 −2 147 483 648～2 147 483 647。整型的变量只能存放整型数值。

当按上述方式定义变量 i，j 和 k 时，编译程序仅为 i，j 和 k 开辟存储单元，而没有在存储单元中存放任何初值。

C 语言规定，可以在定义变量的同时给变量赋初值，也称变量初始化。例如：

```
main（）
{int i=1，j=0，k=2；  /*定义 i，j，k 为整型变量，它们的初值分别为 1，0 和 2*/
……
}
```

2.3.3 整型数据的取值范围

不同的编译系统或计算机系统对这几类整型数所占用的字节数有不同的规定。表 2.1 列出了在 VC 中定义的整型数所占用的存储空间和取值范围。表中方括号内的单词可以省略，各单词排列的先后次序无关紧要。

在 VC 中可以在整型常量的后面加一个字母 l（L 的小写）或 L 表示长整型，例如：123L，345l，0L，123456L 等，这些常量在内存中占 4 个字节。

无论是短整型数还是长整型数，都被识别为有符号整数。无符号整数在数的末尾应该加上字母后缀 u 或 U，若是无符号长整型常量，则可以加后缀 lu 或 LU。无符号短整型常量的取值应在 0～65 535 范围内，无符号长整型常量的取值在 0～4 294 967 295 的范围内。

注意：无符号常量不能表示成小于 0 的负数，例如：－200U 是不合法的。

表 2.1　VC 中定义的整型数所占存储空间和取值范围

类型名称	占用的存储空间/B	取值范围
[signed] int	4	－2 147 483 648～ 2 147 483 647
[signed] short [int]	2	－32 768～ 32 767
[signed] long [int]	4	－2 147 483 648～ 2 147 483 647
unsigned [int]	4	0～4 294 967 295
unsigned short [int]	2	0～65 535
unsigned long [int]	4	0～4 294 967 295

2.3.4　整数在内存中的存储形式

计算机中，内存储器的最小存储单位称为“位（bit)”，由于只能存放 0 或 1，因此称为二进制位。大多数计算机把 8 个二进制位组成一个“字节（byte)”，并给每个字节分配一个地址。若干字节组成一个“字（word)”，用一个“字”来存放一条机器指令或一个数据。一个字含多少个字节随机器的不同而不同。一台计算机如果以 2 个字节（16 个二进制位）来存放一条机器指令，则称此计算机的字长为 16 位；如果以 4 个字节（32 个二进制位）来存放一条机器指令，则称此计算机的字长为 32 位。

通常把一个字节中的最右边一位称为最低位，最左边一位称为最高位。对于一个有符号整数，其中最高位（最左边的一位）用来存放整数的符号，称为符号位。若是正整数，最高位放置 0；若是负整数，最高位放置 1。

1. 正整数

当用两个字节存放一个 short 类型正整数时，例如正整数 5，其在内存中的二进制码为

0000000000000101

对于正整数的这种存储形式称为用“原码”形式存放。因此用两个字节存放 short 类型的最大正整数是

0111111111111111

它的值为 32 767。

2. 负整数

（1）负整数在内存中是以“补码”形式存放的。

取某个二进制数的补码，例如 10000101（十进制数－5）的补码，步骤如下：

①求原码的反码。把原码除符号位之外的二进制码按位取反，即把 1 变成 0，0 变成 1，即得到该原码的反码。例如 10000101 的反码为 11111010。

②把所得的反码加 1，即得到原码的补码。因此 11111010 加 1 得 11111011，这就是－5在内存中的二进制码。若用两个字节表示，即为

1111111111111011

（2）把内存中以补码形式存放的二进制码转化成十进制的负整数，步骤如下：

①先对除符号位之外的各位取反。例如有补码 11111010，取反后为 10000101。

②将所得二进制数转换成十进制数。例如，10000101 的十进制数为－5。

③对所求得的数再减 1。

通过以上分析可知，由两个字节存放的最小整数是 1000000000000000，它对应的十进制数为－32 768，而－1 在内存中存放的二进制码为 1111111111111111。

3. 无符号整数

用两个字节存放一个整数时，若说明为无符号整数，则最高位不再用来存放整数的符号，16 个二进制位全部用来存放整数，因此无符号整数不可能是负数。这时，若内存中存放的 16 个二进制位全部为 1，则它所代表的整数就不再是－1，而是 65 535。

2.4 实型数据

2.4.1 实型常量

实型常量又称实数或浮点数。在 C 语言中可以用两种形式表示一个实型常量。

1. 小数形式

小数形式是由数字和小数点组成的一种实数表示形式，例如 0.123，.123，123.，0.0 等都是合法的实型常量。注意：小数形式表示的实型常量必须要有小数点。

2. 指数形式

这种形式类似数学中的指数形式。在数学中，一个数可以用幂的形式来表示，如 2.302 6 可以表示为 $0.230\ 26\times10^{1}$，$2.302\ 6\times10^{0}$，23.026×10^{-1}等形式。在 C 语言中，则以“e”或“E”后跟一个整数来表示以 10 为底的幂数。2.302 6 可以表示为 0.23026E1，2.3026e0，23.026e－1。C 语言的语法规定，字母 e 或 E 之前必须要有数字，且 e 或 E 后面的指数必须为整数。如 e3，.5e3.6，.e3，e 等都是非法的指数形式。注意：在字母 e 或 E 的前后以及数字之间不得插入空格。

2.4.2 实型变量

C 语言中实型变量分为单精度型和双精度型两种，分别用类型名 float 和 double 进行定义。

单精度型变量定义的形式如下：

```
float a，b，c；
```

双精度型变量定义的形式如下：

```
double x，y，z；
```

在一般计算机系统中，为 float 类型的变量分配 4 个字节的存储单元，为 double 类型

的变量分配 8 个字节的存储单元，并按实型数的存储方式存放数据。实型的变量只能存放实型数，不能用整型变量存放一个实数，也不能用实型变量存放一个整数。

在 VC 中单精度实数（float 类型）的数值范围是-10^{38}～10^{38}，并提供 7 位有效数字位；绝对值小于10^{-38}的数被处理成零值。双精度实数（double 类型）的数值范围约在-10^{308}～10^{308}之间，并提供 15～16 位有效数字位，具体精确到多少位与机器有关；绝对值小于10^{-308}的数被处理成零值。因此 double 型变量中存放的数据要比 float 型变量中存放的数据精确得多。注意，在 VC 中，所有的 float 类型数据在运算中都自动转换成 double 型数据。

前面已经介绍过，在程序中一个实数可以用小数形式表示，也可以用指数形式表示。但在内存中，实数一律是以指数形式存放的。

注意：在计算机中可以精确地存放一个整数，不会出现误差，但整型数值的数值范围比实数小。实型数的数值范围较整型大，但往往存在误差。

2.5　算术表达式

2.5.1　基本的算术运算符

在 C 语言中，基本的算术运算符是＋，－，＊，/，%，分别为加、减、乘、除、求余运算符。这些运算符需要两个运算对象，称为双目运算符。除求余运算符%外，运算对象可以是整型，也可以是实型，如 1＋2，1.2＊3.2。

求余运算符的运算对象只能是整型。在%运算符左侧的运算数为被除数，右侧的运算数为除数，运算结果是两数相除后所得的余数。当运算数为负数时，所得结果的符号随机器的不同而不同。

“＋”和“－”也可用作单目运算符，运算符必须出现在运算数的左边。运算数可为整型，也可为实型，如－54，＋3.9。

说明：

（1）如果双目运算符两边运算数的类型一致，则所得结果的类型与运算数的类型一致。例如：1.0/2.0，其运算结果为 0.5；1/2，其运算结果为 0。

（2）如果双目运算符两边运算数的类型不一致，系统将自动进行类型转换，使运算符两边的类型达到一致后，再进行运算。

（3）在 C 语言中，所有实型数的运算均以双精度方式进行。若是单精度数，则在尾数部分添 0，使之转化为双精度数。

2.5.2 运算符的优先级、结合性和算术表达式

在 C 语言中，常量、变量、函数调用以及按 C 语言语法规则用运算符把运算数连起来的式子都是合法的表达式。凡是表达式都有一个值，即运算结果。

1. 算术运算符的优先级

算术运算符和圆括号的优先级高低次序如图 2.1 所示。

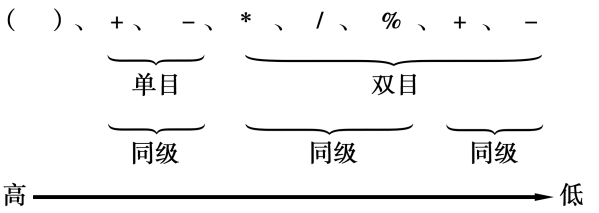

图 2.1 算术运算符和圆括号的优先级高低次序

2. 算术运算符和圆括号的结合性

以上所列的运算符中，只有单目运算符“+”和“-”的结合性是从右到左的，其余运算符的结合性都是从左到右。

例如：表达式（5+1）/2 的运算结果是 3，圆括号的优先级高于除号；表达式 5+1/2 的运算结果是 5，除号的优先级高于加号；表达式 5 * -2 的运算结果是-10，单目运算符“-”的优先级高于乘号，这个表达式与 5 *（-2）等价。

3. 算术表达式

用算术运算符和一对圆括号将运算数（或称操作数）连接起来的、符合 C 语言语法的表达式称为算术表达式。

算术表达式中，运算对象可以是常量、变量和函数等。例如：2+sqrt（c）* b。

在计算机语言中，算术表达式的求值规律与数学中四则运算的规律类似，其运算规则和要求如下：

（1）在算术表达式中，可使用多层圆括号，但左右括号必须配对。运算时从内层圆括号开始，由内向外依次计算表达式的值。

（2）在算术表达式中，若包含不同优先级的运算符，则按运算符的优先级由高到低进行运算；若表达式中运算符的级别相同，则按运算符的结合方向进行运算。例如表达式 a+b-c，因为加号和减号的优先级相同，它们的结合性为从左到右，因此先计算 a+b，然后把所得结果减去 c 的值。

2.5.3 强制类型转换表达式

强制类型转换表达式的形式如下：

（类型名）（表达式）

上述形式中，“(类型名)”称为强制类型转换运算符，利用强制类型转换运算符可以将一个表达式的值转换成指定的类型，这种转换是根据人为要求进行的。例如：表达式(int) 3.234 把 3.234 转换成整数 3；表达式 (double)(10%3) 把 10%3 所得结果转换成双精度数。

2.6　赋值表达式

2.6.1　赋值运算符和赋值表达式

在 C 语言中，赋值号“=”是一个运算符，称为赋值运算符。由赋值运算符组成的表达式称为赋值表达式，其形式如下：

变量名=表达式

赋值号的左边必须是一个代表某一存储单元的变量名，对于初学者来说，只要记住赋值号左边必须是变量名即可。赋值号的右边必须是 C 语言中合法的表达式。赋值运算的功能是先求出右边表达式的值，然后把此值赋给赋值号左边的变量，确切地说，是把数据存入以该变量为标识的存储单元中去。例如，a 和 b 都被定义成 int 类型变量：

```
a=10;        /* 把常量 10 赋给变量 a */
b=a;         /* 把 a 中的值赋给变量 b，a 中的值不变 */
```

在程序中可以多次给一个变量赋值，每赋一次值，与它相应的存储单元中的数据就被更新一次，内存中当前的数据就是最后一次所赋的那个数据。

说明：

(1) 赋值运算符的优先级别只高于逗号运算符，比任何其他运算符的优先级都低，且具有自右向左的结合性。因此，对于表达式 a=2+7/3，由于所有其他运算符的优先级都比赋值运算符高，所以先计算赋值运算符右边表达式的值，再把此值赋给变量 a。

(2) 赋值运算符不同于数学中的“等于号”，这里不是等同的关系，而是进行“赋予”的操作。

(3) 赋值表达式 x=y 的作用是，将变量 y 所代表的存储单元中的内容赋给变量 x 所代表的存储单元，x 中原有的数据被替换掉。赋值后，变量 y 中的内容保持不变。此表达式应当读作“把 y 的值赋给 x”，而不应读作“x 等于 y”。

(4) 在赋值表达式 x=x 中，虽然赋值运算符两边的运算对象都是 x，但出现在赋值号左边和右边的 x 具有不同的含义。赋值号右边的 x 表示变量 x 所代表的存储单元中的值。赋值号左边的 x 代表以 x 为标识的存储单元。该表达式的含义是取变量 x 中的值放到变量 x 中去。当然，这一操作并无实际意义。

表达式 n=n+1 也是合法的赋值表达式，其作用是取变量 n 中的值加 1 后再放回到变

量 n 中，即使变量 n 中的值增 1。

（5）赋值运算符的左侧只能是变量，不能是常量或表达式。a＋b＝c 就是非法的赋值表达式。

（6）等号右边的表达式也可以是一个赋值表达式。如 a＝b＝7＋1，按照运算符的优先级，将首先计算出 7＋1 的值 8，然后按照赋值运算符自右向左的结合性，把 8 赋给变量 b，最后再把变量 b 的值赋给变量 a。而表达式 a＝7＋1＝b 则是不合法的，因为在 7＋1＝b 中，在赋值号的左边不是一个变量。

（7）在 C 语言中，“＝”号被视为一个运算符，a＝19 是一个表达式，而表达式应该有一个值，C 语言规定最左边变量中所得到的新值就是赋值表达式的值。

（8）C 语言的赋值表达式可以作为语句中的某个成分出现在众多的语句或表达式中，从而使变量中的数值变化过程变得难以掌握。因此要求读者在学习过程中建立正确的概念，才能准确掌握赋值表达式的运算规律。

2.6.2　复合赋值表达式

在赋值运算符之前加上其他运算符可以构成复合赋值运算符。C 语言规定可以使用 10 种复合赋值运算符，其中与算术运算有关的复合赋值运算符有＋＝，－＝，＊＝，/＝，％＝（注意：两个符号之间不可以有空格）。复合赋值运算符的优先级与赋值运算符的优先级相同。表达式 n＋＝1 的运算规则等价于 n＝n＋1，表达式 n＊＝m＋3 的运算规则等价于 n＝n＊（m＋3），因为运算符“＋”的优先级高于复合赋值运算符“＊＝”。其他以此类推。

例 2.3　已有变量 a，其值为 9，计算表达式 a＋＝a－＝a＋a 的值。

因为赋值运算符与复合赋值运算符“－＝”和“＋＝”的优先级相同，且运算方向自右至左，所以：

（1）先计算“a＋a”，因 a 的初值为 9，所以该表达式的值为 18，注意 a 的值未变。

（2）再计算“a－＝18”，此式相当于“a＝a－18”，因 a 的值仍为 9，所以表达式的值为－9，注意 a 的值已为－9。

（3）最后计算“a＋＝－9”，此式相当于“a＝a＋（－9）”，因 a 的值此时已是－9，所以表达式的值为－18。

由此可知，表达式 a＋＝a－＝a＋a 的值是－18。

2.6.3　赋值运算中的类型转换

在赋值运算中，只有在赋值号右侧表达式的类型与左侧变量类型完全一致时，赋值操作才能进行。如果赋值运算符两侧的数据类型不一致，在赋值前，系统将自动先把右侧表达式求得的数值按赋值号左边变量的类型进行转换，也可以用强制类型转换的方式人为地进行转换后将值赋给赋值号左边的变量。这种转换仅限于数值数据之间，通常称为“赋值

兼容”。对于另外一些数据，例如后面将要讨论的地址值就不能赋给一般的变量，称为“赋值不兼容”。

在这里，特别需要指出的是在进行混合运算时整型数据类型之间的转换问题。

在 C 语言的表达式（不包括赋值表达式）中，如果运算符两边的整数类型不相同，将进行类型之间的转换。转换规则如下：

（1）若运算符两边一个是短整型，一个是长整型，则将短整型转换为长整型，然后进行运算。

（2）若运算符两边一个是有符号整型，一个是无符号整型，则将有符号整型转换成无符号整型，然后进行运算。

在 C 语言的赋值表达式中，赋值号右边的值先转换成与赋值号左边的变量相同的类型，然后进行赋值。应当注意：

（1）当赋值号左边的变量为短整型，右边的值为长整型时，短整型变量只能接受长整型数低位上两个字节中的数据，高位上两个字节中的数据将丢失。也就是说，右边的值不能超出短整型的数值范围，否则将得不到预期的结果。例如，若有以下语句：

```
short a;
unsigned long b;
b= 98304; a=b;
printf ("%d\n", a);
```

则 a 的值为－32768。因为 98304（二进制数 11000000000000000）已经超出短整型的数值范围（－32 768～32 767），a 截取 b 低 16 位的值（二进制数 1000000000000000），由于最高位为 1，因此 a 的值为－32768。

（2）当赋值号左边的变量为无符号整型，右边的值为有符号整型时，则把内存中的内容原样赋值。右边数值的范围不应超出左边变量可以接受的数值范围。同时需要注意，这时负数将转换为正数。例如，变量 a 被说明为 unsigned 类型，在进行了 a＝－1 的赋值操作后，将使 a 的值为 65535。

（3）当赋值号左边的变量为有符号整型，右边的值为无符号整型时，赋值的机制同上。这时若符号位为 1，将按负数处理。

2.7 自加、自减运算符和逗号运算符

2.7.1 自加运算符“++”和自减运算符“--”

(1) 自加运算符“++”和自减运算符“--”的运算结果是使运算对象的值增 1 或减 1。如 i++，相当于 i=i+1；i--，相当于 i=i-1。因此，自加或自减运算本身也是一种赋值运算。

(2) 运算符“+”和“-”是单目运算符，运算对象可以是整型变量，也可以是实型变量，但不能是常量或表达式，因为不能给常量或表达式赋值。因此，如 ++3，(i+j) --等都是不合法的。

(3) 用自加或自减运算符构成表达式时，既可以前缀形式出现，也可以后缀形式出现。例如：++i，--i，i++，i--等都是合法的表达式。无论是作为前缀运算符还是作为后缀运算符，对于变量本身来说自增 1 或自减 1 都具有相同的效果，但作为表达式来说却有着不同的值。例如，若变量 i 为 int 类型，且已有值为 5。若表达式为++i，则先进行 i 增 1 运算，i 的值为 6，表达式的值也为 6；若表达式为--i，则先进行 i 减 1 运算，i 的值为 4，表达式的值也为 4；若表达式为 i++，则表达式先取 i 的值为 5，然后 i 进行增 1 运算，使 i 的值为 6；若表达式为 i--，则表达式先取 i 的值为 5，然后 i 进行减 1 运算，使 i 的值为 4。

(4) 运算符“++”和“--”的结合方向是“自右至左”。例如，有一表达式 -i++，其中 i 的原值为 3。由于负号运算符与自加运算符的优先级相同，结合方向是“自右至左”，即相当于对表达式-(i++) 进行运算，此时自加运算符“++”为后缀运算符，(i++) 的值为 3，因此-(i++) 的值为-3，然后 i 自增为 4。

(5) 不要在一个表达式中对同一个变量进行多次诸如 i++或++i 等运算，例如写成：i++*++i+i--*--i，这种表达式不仅可读性差，而且不同的编译系统对这样的表达式将做不同的解释，进行不同的处理，因而所得结果也各不相同。

2.7.2 逗号运算符和逗号表达式

“,”是 C 语言提供的一种特殊运算符，用逗号将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 *n*

说明：

（1）逗号运算符的结合性为从左到右，因此逗号表达式将从左到右进行运算，即先计算表达式 1，然后计算表达式 2，依次进行，最后计算表达式 n。最后一个表达式的值就是此逗号表达式的值。例如：（i＝3，i＋＋，＋＋i，i＋5）这个逗号表达式的值是 10，i 的值为 5。

（2）在所有运算符中，逗号运算符的优先级最低。

练　习　题

一、选择题

1. 以下选项中正确的整型常量是（　　）。

A. 12.　　B. －20　　C. 1，000　　D. 456

2. 以下选项中正确的实型常量是（　　）。

A. 0　　B. 3.1415　　C. 0.329×102　　D. .871

3. 以下选项中不正确的实型常量是（　　）。

A. 2.607E－1　B. 0.8103e2　　C. －77.77　　D. 456e－2

4. 以下选项中不合法的用户标识符是（　　）。

A. abc.c　　B. file　　C. Main　　D. PRINTF

5. 以下选项中不合法的用户标识符是（　　）。

A. _123　　B. printf

C. A$　　D. Dim

6. C 语言中运算对象必须是整型的运算符是（　　）。

A. %　　B. /　　C. !　　D. * *

7. 可在 C 程序中用作用户标识符的一组标识符是（　　）。

A. void define WORD　　B. as _ b3　_123　If

C. For　－ abc　case　　D. 2c　DO　SIG

8. 若变量已正确定义并赋值，则符合 C 语言语法的表达式是（　　）。

A. a＝a＋7；　　B. a＝7＋b＋c，a＋＋

C. int（12.3%4）　　D. a＝a＋7＝c＋b

9. 以下叙述中正确的是（　　）。

A. a 是实型变量，C 允许进行赋值 a＝10，但不可以这样说：实型变量中允许存放整型值

B. 在赋值表达式中，赋值号左边既可以是变量也可以是任意表达式

C. 执行表达式 a＝b 后，在内存中 a 和 b 存储单元中的原有值都将被改变，a 的值已由原值改变为 b 的值，b 的值由原值变为 0

D. 已有 a＝3，b＝5，当执行了表达式 a＝b，b＝a 之后，a 中的值为 5，b 中的值为 3

二、填空题

1. 若 k 为 int 型变量且赋值 11，则运算 k＋＋后表达式的值为（　　），变量 k 的值为（　　）。
2. 若 x 为 double 型变量，则运算 x＝ 3.2，＋＋x 后表达式的值为（　　），变量 x 的值为（　　）。
3. 函数体由符号（　　）开始，到符号（　　）结束。函数体内的前面是（　　）部分，后面是（　　）部分。
4. C 语言中的标识符可分为（　　）、（　　）和预定义标识符三类。
5. 在 C 语言程序中，用关键字（　　）定义基本整型变量。

三、上机改错题

1. 请指出以下 C 程序的错误所在：

```
#include  stdio.h;
main (  );       /* main function */
  float r, s;       /* r is radius, s is area of circular */
  r=5.0;
  S=3.14159 * r*r;
  printf ("%f\n", s)
```

2. 请指出以下 C 程序的错误所在：

```
main      /*   main function   */
   { float a, b, c, v;  /*   a, b, c are sides, v is volume of cube   */
  a= 2.0; b=3.0; c=4.0
  v=a * b * c;
  printf ("%f\n", v)
  }
```

第3章　顺序结构

3.1　赋值语句

在赋值表达式的尾部加上一个“;”号，就构成了赋值语句，也称表达式语句。例如“a=b+c”是赋值表达式，“a=b+c;”则是赋值语句。“i++;”“i--;”“a=b=c;”“a=b，b=c;”等也是赋值语句。赋值语句是一种可执行语句，应当出现在函数的可执行部分。但需要注意，不要把变量定义时的赋初值和赋值语句混为一谈。

C语言中可由形式多样的赋值表达式构成赋值语句，用法灵活，因此读者首先应当掌握好赋值表达式的运算规律才能写出正确的赋值语句。

3.2　数据输出

把数据从计算机内部送到计算机外部设备上的操作称为“输出”。例如把计算机运算结果显示在屏幕上或打印在纸上，或者送到磁盘上保存起来。从计算机外部设备将数据送入计算机内部的操作称为“输入”。

C语言本身并没有提供输入输出语句，但可以通过调用标准库函数中提供的输入和输出函数来实现输入和输出。C语言提供了丰富的用于输入和输出的库函数。在VC环境下，在调用输入和输出的库函数之前要求在源程序中出现包含头文件stdio.h的命令行：

#include <stdio.h>

3.2.1 printf 函数的一般调用形式

printf 函数是 C 语言提供的标准输出函数，用来在终端设备上按指定格式进行输出。

printf 函数的调用形式如下：

printf（格式控制，输出项 1，输出项 2，…）

格式控制是字符串形式。

在 printf 函数调用之后加上“;”，则构成输出语句。例如：

printf（"a＝%d，b＝%d"，a，b）；

以上输出语句中，printf 是函数名，用双引号括起来的字符串部分 a＝%d，b＝%d 是输出格式控制，决定了输出数据的内容和格式。a，b 称为输出项，是 printf 函数的实参。

printf 函数中格式控制的作用如下：

（1）给输出项提供输出格式说明。输出格式说明的作用是将要输出的数据按照指定的格式输出。格式说明由“%”符号和紧跟在其后的格式描述符组成。当输出项为 int 类型时，用 d 作为格式描述字符，其形式为%d；当输出项为 float 或 double 类型时，用 f 或 e 作为格式描述字符，其形式为%f 或%e（对于 double 类型也可用%lf 或%le）。

（2）提供需要原样输出的文字或字符。除了格式转换说明外，字符串中的其他字符（包括空格）将按原样输出。例如，上例中的“a＝%d，b＝%d”，其中的“a＝”“,”和“b＝”都将原样输出，这样使得输出结果更具有可读性。若 a、b 的值分别为 3 和 4，则上例的 printf 输出的结果是：a＝3，b＝4。

printf 的各输出项之间要用逗号隔开（函数的各个参数之间必须用逗号隔开）。输出项可以是任意合法的常量、变量或表达式。printf 可以没有输出项，函数的调用形式将为 printf（格式控制），输出结果就是格式控制中的固定字符串。比如：“printf（"OK!"）;”将输出字符串：OK!。

对于 printf 函数的调用形式，请见下面的程序示例：

```
#include <stdio.h>
main（ ）
{ int i＝ 2518；
double a＝3.1415；
printf（"i＝%d，a＝%f，a＊10＝%e\n"，i，a，a＊10）；
}
```

运行后的输出结果为

i＝ 2518，a＝3.141500，a＊10＝3.141500e＋01

在以上 printf 函数的输出格式控制中，“i＝”按原样输出，在%d 的位置上输出变量 i 的值，接着输出一个逗号和“a＝”，在%f 的位置上输出变量 a 的值，又输出一个逗号和“a＊10＝”，在%e 的位置上输出表达式 a＊10 的值，最后的\n 是 C 语言中特定的转义字符“回车换行”，使得屏幕光标或打印机针头移到下一行的开头。

3.2.2　printf 函数中常用的格式说明

格式控制中，每个格式说明都必须以“%”开头，以一个格式字符作为结束，在此之间可以根据需要插入“宽度说明”、左对齐符号“－”、前导零符号“0”。

1. 格式字符

%后允许使用的格式字符和它们的功能见表 3.1。在某些系统中，可能不允许使用大写字母的格式字符，因此为了使程序具有通用性，在写程序时应尽量不用大写字母的格式字符。

表 3.1　%后允许使用的格式字符和它们的功能

格式字符	说　明
c	输出一个字符
d 或 i	输出带符号的十进制整型数。%ld 为长整型（16 位编译器上必须使用），%hd 为短整型，%I64d 为 64 位长整型（VC＋＋ 4.0 以上版本输出 _ int64 类型的整数）
o	以八进制格式输出整型数。%o 不带先导 0，例如十进制数 15 用%o 输出为 17；%＃o 加先导 0，例如十进制数 15 用%＃o 输出为 017
x 或 X	以十六进制格式输出整型数。%x 或%X 输出不带先导 0x 或 0X 的十六进制数，例如十进制数 2622 用%x 数据格式输出为 a3e，用%X 数据格式输出为 A3E；%＃x 或%＃X 输出带先导 0x 或 0X 的十六进制数，例如十进制数 2622 用%＃x 数据格式输出为 0xa3e，而用%＃X 数据格式输出为 0XA3E
u	以无符号十进制形式输出整型数
f	以带小数点的数学形式输出浮点数（单精度和双精度数）
e 或 E	以指数形式输出浮点数（单精度和双精度数），格式是：[－] m.dddddde＋xxx 或 [－] m.ddddddE ＋xxx。小数位数（d 的个数）由输出精度决定，隐含的精度是 6。若指定的精度为 0，则包括小数点在内的小数部分都不输出。xxx 为指数，保持 3 位，不足补 0。若指数为 0，输出指数是 000
g 或 G	由系统决定采用%f 格式还是采用%e（或%E）格式输出，以使输出宽度最小
s	输出一个字符串，直到遇到“\0”。若字符串长度超过指定的精度则自动突破，不会截断字符串
p	输出变量的内存地址
%	也就是%%形式，输出一个%

2. 长度修饰符

在%和格式字符之间，可以加入长度修饰符，以保证数据输出格式的正确和对齐。对于长整型数（long）应该加 l，即%ld；对于短整型数（short）可以加 h，即%hd。

3. 输出数据所占的宽度说明

当使用%d，%c，%f，%e，%s，…的格式说明时，输出数据所占的宽度（域宽）由系统决定，通常按照数据本身的实际宽度输出，前后不加空格，并采用右对齐的形式。也可以用以下三种方法人为控制输出数据所占的宽度（域宽），按照使用者的意愿进行输出。

（1）在%和格式字符之间插入一个整数常数来指定输出的宽度 n（例如%4d，n 代表整数 4）。如果指定的宽度 n 不够，输出时将会自动突破，保证数据完整输出。如果指定的

宽度 n 超过输出数据的实际宽度，输出时将会右对齐，左边补以空格，达到指定的宽度。

（2）对于 float 和 double 类型的实数，可以用“n1. n2”的形式来指定输出宽度（n1 和 n2 分别代表一个整常数），其中 n1 指定输出数据的宽度（包括小数点），n2 指定小数点后小数位的位数，n2 也称为精度（例如%12.4f，n1 代表整数 12，n2 代表整数 4）。

对于 f、e 或 E，当输出数据的小数位多于 n2 位时，截去右边多余的小数，并对截去部分的第一位小数做四舍五入处理；当输出数据的小数位少于 n2 时，在小数的最右边补 0，使得输出数据的小数部分宽度为 n2。若给出的总宽度 n1 小于 n2 加上整数位数和小数点（e 或 E 格式还要加上指数的 5 位），则自动突破 n1 的限制；反之，数字右对齐，左边补空格。

也可以用“.n2”格式（例如%.6f），不指定总宽度，仅指定小数部分的输出位数，由系统自动突破，按照实际宽度输出。如果指定“n1.0”或“.0”格式（例如%12.0f 或%.0f），则不输出小数点和小数部分。

对于 g 或 G，宽度用来指定输出的有效数字位数。若宽度超过数字的有效数字位数，则左边自动补 0；若宽度不足，则自动突破。不指定宽度，将自动按照 6 位有效数字输出，截去右边多余的小数，并对截去部分的第一位小数做四舍五入处理。

（3）对于整型数，若输出格式是“0n1”或“.n2”格式（例如%05d 或%.5d），则如果指定的宽度超过输出数据的实际宽度，输出时将会右对齐，左边补 0。

对于 float 和 double 类型的实数，若用“0n1. n2”格式输出（例如%012.4f），如果给出的总宽度 n1 大于 n2 加上整数位数和小数点（e 或 E 格式还要加上指数的 5 位），则数字右对齐，左边补 0。

对于字符串，格式“n1”指定字符串的输出宽度，若 n1 小于字符串的实际长度，则自动突破，输出整个字符串；若 n1 大于字符串的实际长度，则右对齐，左边补空格。若用“.n2”格式指定字符串的输出宽度，则若 n2 小于字符串的实际长度，将只输出字符串的前 n2 个字符。

注意：输出数据的实际精度并不完全取决于格式控制中的域宽和小数的域宽，而是取决于数据在计算机内的存储精度。通常系统只能保证 float 类型有 7 位有效数字，double 类型有 15 位有效数字。若你指定的域宽和小数的域宽超过相应类型数据的有效数字，输出的多余数字是没有意义的，只是系统用来填充域宽而已。

4. 输出数据左对齐

由于输出数据都隐含右对齐，如果想左对齐，可以在格式控制中的“%”和宽度之间加一个“－”号来实现。

5. 使输出数据总带＋号或－号

通常输出的数据如果是负数，前面有符号“－”，但正数前面的“＋”一般都省略了。如果要每一个数前面都带正负号，可以在“%”和格式字符间加一个“＋”号来实现。

各种输出宽度和不指定宽度情况下的输出结果见表 3.2（表中输出结果中的符号 j 代表一个空格），其中 k 为 int 型，值为 1234；f 为 float 型，值为 123.456。

表 3.2　各种输出格式下的输出结果

输出语句	输出结果
printf ("%d \ n", k);	1234
printf ("%6d \ n", k);	jj1234
printf ("%2d \ n", k);	1234
printf ("%f \ n", f);	123.456
printf ("%12f \ n", f);	jj123. 456000
printf ("%12.6f \ n", f);	jj123. 456000
printf ("%2.6f \ n", f);	123. 456000
printf ("%.6f \ n", f);	123. 456000
printf ("%12.2f \ n", f);	jjjjjj123. 46
printf ("%12.0f \ n", f);	jjjjjjjj123
printf ("%.f \ n", f);	123
printf ("%e \ n", f);	1. 234560e+002
printf ("%13e \ n", f);	1. 234560e+002
printf ("%13.8e \ n", f);	1. 23456000e+002
printf ("%3.8e \ n", f);	1. 23456000e+002
printf ("%.8e \ n", f);	1. 23456000e+002
printf ("%13.2e \ n", f);	jjjj1. 23e+002
printf ("%13.0e \ n", f);	jjjjjj1e+002
printf ("%.Oe \ n", f);	1.00E+02
printf ("%g \ n", f);	123. 456
printf ("%5g \ n", f);	123. 456
printf ("%10g \ n", f);	jjj123. 456
printf ("%g \ n", 123.456789);	123. 457
printf ("%06d \ n", k);	1234
printf ("%.6d \ n", k);	1234
printf ("%012.6f \ n", f);	00123. 456000
printf ("%013.2e \ n", f);	00001. 23 e+002
printf ("%s \ n"," abcdefg");	abcdefg
printf ("%10s \ n"," abcdefg");	jjjabcdefg
printf ("%5s \ n"," abcdefg");	abcdefg
printf ("%.Ss \ n"," abcdefg");	abcde
printf ("%－6d \ n", k);	1234jj
printf ("%－12.2f \ n", f);	123. 46jjjjjj
printf ("%－13.2e \ n", f);	1. 23e+002jjjj
printf ("%＋－6d%＋－12.2f \ n", k, －f);	＋1234－123. 46jjjjjj
printf ("%f%% \ n", 12.5);	12. 500000%

3.2.3 使用 printf 函数时的注意事项

(1) printf 函数的输出格式为自由格式，是否在两个数之间留逗号、空格或回车，完全取决于格式控制，如果不注意，很容易造成数字连在一起，使得输出结果没有意义。例如：若 k= 1234，f=123.456，则“printf ("%d%d%f\n", k, k, f);”语句的输出结果是：12341234123.456，无法分辨其中的数字含义。而如果改为“printf ("%d %d %f\n", k, k, f);”，其输出结果是：1234 1234 123. 456，看起来就一目了然了。

(2) 格式控制中必须含有与输出项一一对应的输出格式说明，类型必须匹配。若格式说明与输出项的类型不一一对应匹配，则不能正确输出，而且编译时不会报错。若格式说明个数少于输出项个数，则多余的输出项不予输出；若格式说明个数多于输出项个数，则将输出一些毫无意义的数字乱码。

(3) 在格式控制中，除了前面要求的输出格式，还可以包含任意的合法字符（包括汉字和转义符），这些字符输出时将“原样照印”。此外，还可利用\n（回车）、\r（回行但不回车）、\t（制表）、\a（响铃）等控制输出格式。

(4) 如果要输出%符号，可以在格式控制中用%%表示。

(5) printf 函数有返回值，返回值是本次调用输出字符的个数，包括回车等控制符。

(6) 尽量不要在输出语句中改变输出变量的值，因为可能会造成输出结果的不确定性。例如：“int k=8; printf ("%d,%d\n", k, ++k);”，输出结果不是 8，9，而是 9，9。这是因为调用函数 printf 时，其参数是从右至左进行处理的，将先进行++k 运算。

(7) 输出数据时的域宽可以改变。若变量 m、n、i 和 f 都已正确定义并赋值，则语句“printf ("%d", m, i);”将按照 m 指定的域宽输出 i 的值，并不输出 m 的值。而语句“printf ("%*.*f", m, n, f);”将按照 m 和 n 指定的域宽输出浮点型变量 f 的值，并不输出 m、n 的值。

3.3 数 据 输 入

scanf 函数是 C 语言提供的标准输入函数，其作用是从终端键盘上读入数据。

3.3.1 scanf 函数的一般调用形式

scanf 函数的一般调用形式如下：

scanf（格式控制，输入项 1，输入项 2，…）

在 scanf 函数调用之后加上“;”，则构成输入语句。

例如，若 k 为 int 型变量，a 为 float 型变量，y 为 double 型变量，可通过以下函数调用语句进行输入：

scanf ("%d%f%lf", &k, &a, &y);

其中 scanf 是函数名，双引号括起来的字符串部分为格式控制部分，其后的 &k，&a，&y 为输入项。

格式控制的主要作用是指定输入时的数据转换格式，即格式转换说明。scanf 的格式转换说明与 printf 的类似，也是由“%”开始，其后是格式字符。上例的%d、%f（或%e）、%lf（或%le）分别用于 int、float 和 double 型数据的输入。

输入项之间用逗号隔开。对于 int、float 和 double 型变量，在变量之前必须加 & 符号作为输入项（& 是 C 语言中的求地址运算符，输入项必须是地址表达式）。

3.3.2　scanf 函数中常用的格式说明

每个格式说明都必须用%开头，以一个“格式字符”作为结束。

通常允许用于输入的格式字符及其相应的功能见表 3.3。

表 3.3　用于输入的格式字符及其功能

格式字符	说　明
c	输入一个字符
d	输入带符号的十进制整型数
i	输入整型数，整型数可以是带先导 0 的八进制数，也可以是带先导 0x（或 0X）的十六进制数
o	以八进制格式输入整型数，可以带先导 0，也可以不带
x	以十六进制格式输入整型数，可以带先导 0x 或 0X，也可以不带
u	以无符号十进制形式输入整型数
f（lf）	以带小数点的数字形式或指数形式输入浮点数（单精度数用 f，双精度数用 lf）
e（le）	同上
s	输入一个字符串，直到遇到“\0”。若字符串长度超过指定的精度则自动突破，不会截断字符串

说明：

（1）在格式串中，必须含有与输入项一一对应的格式转换说明符。若格式说明与输入项的类型不一一对应匹配，则不能正确输入，而且编译时不会报错。若格式说明个数少于输入项个数，scanf 函数结束输入，则多余的输入项将无法得到正确的输入值；若格式转换说明个数多于输入项个数，scanf 函数也结束输入，多余的数据作废，不会作为下一个输入语句的数据。

（2）在 VC 环境下，输入 short 型整数，格式控制要求用%hd。要输入 double 型数据，格式控制必须用%lf（或%le）。否则，数据不能正确输入。

（3）在 scanf 函数的格式字符前可以加入一个正整数指定输入数据所占的宽度，但不可以对实数指定小数位的宽度。

（4）由于输入是一个字符流，scanf 从这个流中按照格式控制指定的格式解析出相应数据，送到指定地址的变量中。因此当输入的数据少于输入项时，运行程序将等待输入，

直到满足要求为止。当输入的数据多于输入项时，多余的数据在输入流中没有作废，而是等待下一个输入操作语句继续从此输入流读取数据。

（5）scanf 函数有返回值，其值就是本次 scanf 调用正确输入的数据项的个数。

3.3.3 通过 scanf 函数从键盘输入数据

当用 scanf 函数从键盘输入数据时，每行数据在未按下回车键（Enter 键）之前，可以任意修改。但按下回车键（Enter 键）后，scanf 函数即接受了这一行数据，不能再回去修改。

1. 输入数值数据

在输入整数或实数这类数值型数据时，输入的数据之间必须用空格、回车符、制表符（Tab 键）等间隔符隔开，间隔符个数不限。即使在格式说明中人为指定了输入宽度，也可以用此方式输入。例如：若 k 为 int 型变量，a 为 float 型变量，y 为 double 型变量，有以下输入语句：

```
scanf ("%d%f%le", &k, &a, &y);
```

若要给 k 赋值 10，a 赋值 12.3，y 赋值 1234567.89，输入格式可以是（输入的第一个数据之前可有任意空格）：

```
10 12.3 1234567.89<CR>
```

此处<CR>表示回车键。也可以是：

```
10<CR>
12. 3<CR>
1234567.89<CR>
```

只要能把 3 个数据正确输入，就可以按任何形式添加间隔符。

2. 指定输入数据所占的宽度

可以在格式字符前加入一个正整数指定输入数据所占的宽度。例如上例可改为：

```
scanf ("%3d%5f%5e", &k, &a, &y);
```

若从键盘上从第 1 列开始输入：

```
123456.789.123
```

用“printf ("%d %f %f\n", k, a, y);”打印的结果是：

```
123 456. 700000 89.120000
```

可以看到，由于格式控制是%3d，因此把输入数字串的前三位 123 赋值给了 k；由于对应于变量 a 的格式控制是%5f，因此把输入数字串中随后的 5 位数（包括小数点）456.7 赋值给了 a；由于格式控制是%5e，因此把数字串中随后的 5 位（包括小数点）89.12 赋值给了 y。

由以上示例可知，数字之间不需要间隔符，若插入了间隔符，系统也将按指定的宽度来读取数据，从而会引起输入混乱。除非数字是“粘连”在一起，否则不提倡指定输入数据所占的宽度。

3. 跳过某个输入数据

可以在%和格式字符之间加入“*”号，作用是跳过对应的输入数据。例如：

```
int x, y, z;
scanf ("%d%*d%d", &x, &y, &z);
printf ("%d%d%d\n", x, y, z);
```

若是输入：

12 34 56 78

则输出是：

12 56 78

系统将 12 赋给 x，跳过 34，把 56 赋给 y，把 78 赋给 z。

4. 在格式控制字符串中插入其他字符

scanf 函数中的格式控制字符串是为了输入数据用的，无论其中有什么字符，也不会输出到屏幕上，因此若想在屏幕上输出提示信息，应该首先使用 printf 函数输出。例如：

```
int x, y, z;
scanf ("Please input x, y, z:%d%d%d", &x, &y, &z);
```

屏幕上不会输出“Please input x, y, z:”，而是要求输入数据时按照一一对应的位置原样输入这些字符，必须从第一列起以下面的形式进行输入：

Please input x, y, z: 12 34 56

包括“Please input x, y, z:”中字符的大小写、字符间的间格等必须与 scanf 中的完全一致。这些字符又被称为通配符。

但如果使用以下的形式：

```
int x, y, z;
printf ("Please input x, y, z:");
scanf ("%d%d%d", &x, &y, &z);
```

运行时，由于 printf 语句的输出，屏幕上将出现提示“Please input x, y, z:”，只需按常规输入下面的数据即可：

12 34 56

如果在上面的 scanf 函数中，在每个格式说明之间加一个逗号作为通配符：

```
scanf ("%d,%d,%d", &x, &y, &z);
```

则输入数据时，必须在前两个数据后面紧跟一个逗号，以便与格式控制中的逗号一一匹配，否则就不能正确读入数据。例如，输入“12, 34, 56”能正确读入。输入“12, 34,　　56”也能正确读入，因为空格是间隔符，将全部被忽略掉。但输入“12　　, 34　　,56”将不能正确读入，因为逗号没有紧跟在输入数据后面。

需要提醒的是，为了减少不必要的麻烦，尽量不要使用通配符。

3.4 复合语句和空语句

3.4.1 复合语句

在 C 语言中，一对花括号“{}”不仅可用作函数体的开头和结尾的标志，也可用作复合语句的开头和结尾的标志。复合语句也可称为“语句块”，其语句形式如下：

{语句 1　语句 2　……　语句 n}

用一对花括号把若干语句括起来构成一个语句组。一个复合语句在语法上视为一条语句，在一对花括号内的语句数量不限。例如：

```
{a++; b*=a; printf (" b=%d\n", b);}
```

在复合语句中，不仅可以有执行语句，也可以有定义部分，定义本复合语句中的局部变量。

3.4.2 空语句

C 程序中的所有语句都必须由一个分号“;”作为结束。如果只有一个分号，如：

```
main ()
{    ;
     }
```

这个分号也是一条语句，称为“空语句”，程序执行时不产生任何动作。程序设计中有时需要加一个空语句来表示存在一条语句，但随意加分号也会导致逻辑上的错误，而且这种错误十分隐蔽，编译器也不会提示逻辑错误，初学者一定要小心，需要慎用。

3.5 程 序 举 例

例 3.1　以下程序由终端输入两个整数给变量 x 和 y；然后输出 x 和 y；在交换 x 和 y 中的值后，再输出 x 和 y。

程序如下：

```
#include <stdio.h>
```

```
main (  )
{int x, y, t;
printf ("Enter x&y: \n");
scanf ("%d%d", &x, &y);
printf ("x=%d y=%d\n", x, y);
t=x; x=y; y=t;
printf ("x=%d y=%d\n", x, y);
}
```

以下是程序运行情况：

```
Enter x&y:        (由第 4 行的 printf 输出)
123 456<CR>       (从键盘输入两个整数，<CR>代表按 Enter 键)
x= 123 y=456      (由第 6 行的 printf 输出)
x= 456 y=123      (由第 8 行的 printf 输出)
```

在程序中交换 x 和 y 两个变量中的数据时，不可以简单地用“x=y; y=x;”两条语句来实现，语句“x=y;”执行的结果将把 y 中的值复制到 x 中，使 y 和 x 变量中具有相同的值，x 中原有的值丢失，因此无法再实现两数的交换。为了不丢失 x 中原有的值，必须在执行“x=y;”前，把 x 中的值放到一个临时变量中保存起来（在此，通过“t=x;”来实现），在执行了“x=y;”之后，再把保存在临时变量中的值赋给 y（通过“y=t;”来实现）。

例 3.2　输入一个 double 类型的数，使该数保留小数点后两位，对第三位小数进行四舍五入处理，然后输出此数，以便验证处理是否正确。

程序如下：

```
#include <stdio. h>
main ()
{double x;
printf ("Enter x: \n");
scanf ("%lf", &x);
printf ("(1) x=%f\n", x);
x=x*100;
x=x+0.5;
x= (int) x;
x=x/100;
printf ("(2) x=%f\n", x);
}
```

运行结果如下：

```
Enter x:        (printf 输出提示信息)
123.4567<CR>    (从键盘输入 123.4567，<CR>代表 Enter 键)
```

（1）x＝123.456700　　（输出原始数据）

（2）x＝123.460000　　（输出对第三位小数进行四舍五入后的数据）

注意：在 scanf 函数中给 double 类型变量输入数据时，应该使用%lf 格式转换说明符，而输出时，对应的格式转换说明符可以是%lf，也可以用%f。

练　习　题

一、选择题

1. 若 a，b，c，d 都是 int 型变量且初值为 0，以下选项中不正确的赋值语句是（　　）。

A. a＝b＝c＝100；　　B. d＋＋；　　C. e＋b；　　D. d＝（c＝22）－（b＋＋）；

2. 下列选项中不是 C 语句的是（　　）。

A. {int i；i＋＋；printf（"%d\n"，i）；}

B. ；

C. a＝5，c＝10

D. {　；}

3. 合法的 C 语言赋值语句是（　　）。

A. a＝b＝58　　B. k＝int（a＋b）；

C. a＝58，b＝58　　D. －－i；

4. 有以下程序：

```
#include <stdio.h>
main()
{int x=10,y=3;
printf("%d\n",y=x/y);
}
```

执行后的输出结果是（　　）。

A. 0　　B. 1　　C. 3　　D. 不确定的值

5. 若变量已正确定义为 int 型，要给 a、b、c 输入数据，正确的输入语句是（　　）。

A. read（a，b，c）；

B. scanf（"%d%d%d"，a，b，c）；

C. scanf（"%D%D%D"，&a,%b,%c）；

D. scanf（"%d%d%d"，&a，&b，&c）；

6. 若变量已正确定义为 float 型，要通过输入语句“scanf（"%f　%f　%f"，&a，&b，&c）；”给 a 赋值 11.0，给 b 赋值 22.0，给 c 赋值 33.0，不正确的输入形式是（　　）。

A. 11，22，33　　B. 11.0，22.0，33.0

C. 11.0，22.0，33.0　　D. 11，22，33

7. 若变量 a、b、t 已正确定义，要将 a 和 b 中的数进行交换，以下选项中不正确的语句组是（　　）。

A. a= a+b，b=a-b，a= a-b；　　B. t=a，a=b，b=t；

C. a=t；t=b；b=a；　　D. t=b；b=a；a=t；

8. 若有正确定义语句：

```
double x=5.16894;
```

则语句“printf ("%f\n", (int) (x*1000+0.5) / (double) 1000);”的输出结果是（　　）。

A. 输出格式说明与输出项不匹配，输出无定值

B. 5.170000

C. 5.168000

D. 5.169000

9. 若有以下程序段：

```
int c1=1, c2=2, c3;
c3=c1/c2;
printf ("%d\n", c3);
```

执行后的输出结果是（　　）。

A. 0　　B. 1/2　　C. 0.5　　D. 1

二、填空题

1. 以下程序段中输出语句执行后的输出结果是（　　）。

```
int i=-200, j=2500;
printf (" (1)%d,%d", i, j);
printf (" (2) i=%d, j=%d\n", i, j);
printf (" (3) i=%d\nj=%d\n", i, j);
```

2. 变量 i，j，k 已定义为 int 型并均有初值 0，用语句

```
scanf ("%d", &i); scanf (" td", &j); scanf ("%d", &k);
```

进行输入时，从键盘输入：

1 2.3<CR>　　（<CR>代表 Enter 键）

则变量 i，j，k 的值分别是（　　）、（　　）、（　　）。

三、编程题

1. 编写程序，把 560 分钟换算成用小时和分钟表示，然后进行输出。

2. 编写程序，输入两个整数：1500 和 350，求出它们的商和余数并进行输出。

3. 编写程序，读入三个双精度数，求它们的平均值并保留此平均值小数点后一位数，对小数点后第二位数进行四舍五入，最后输出结果。