第 2 章　数据类型、运算符和表达式

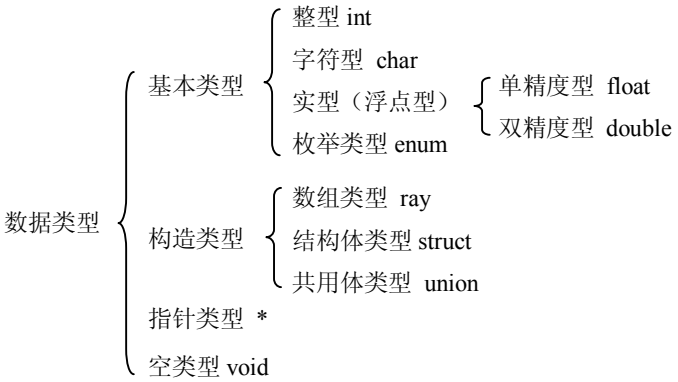
C 语言提供了丰富的数据类型和运算符，这就为编程提供了基础数据。程序员通过自己的算法对数据进行处理，达到自己的目的。

2.1　C 语言的数据类型

一个程序应包括数据结构和算法。数据结构是在程序中要指定数据的类型和数据的组织形式。算法是如何对数据结构进行处理以达到编程者的目的的想法，由于每个人的想法不一样，所以程序代码有可能不相同，但最后都能达到同样的目的。以学生回家为例，学生是数据结构，由于现在的交通很便利，坐火车、坐汽车或者骑自行车等方法都能够使学生到家，所以如何回家就是算法的体现。根据不同的编程语言，程序的概念可以表示为：

程序=算法+数据结构+程序设计方法+语言环境

C 语言的数据结构是以数据类型的形式出现的。C 语言的数据类型如下：



使用上面的数据类型可以完成各种复杂的的数据结构。C 语言中的数据有常量与变量之分，在程序中对用到的所有数据都必须指定其数据类型。

2.2　常量

在程序执行过程中，其值始终保持不变的量称之为常量。常量具有一定的类型，类型由其表示形式决定。在 C 语言中，常量可以分为整型常量、实型常量、字符型常量、字符串常量、符号常量等。

2.2.1　整型常量

整型常量就是整型常数，它是一种从字面即可判断其值的常量。C 语言中的常量可以用十进制、八进制和十六进制表示。

（1）十进制整型常量：由数字 0～9 组成且第一个数不能为 0 的整数，如 123、-456、0。

（2）八进制整型常量：以 0 开头的由 0～7 组成的整数，如 0123 表示八进制数 123。

（3）十六进制整型常量：以 0x 开头的由数字 0～9 和字母 a～f（A～F）组成的的数，如 0x123 代表数十六进制的 123。

整型常量的书写一定要合法，例如，078 和 3a 就不合法。其中 078 表示八进制数，但包含了 0～7 以外的数 8。3a 表示十进制数，但包含了 0～9 以外的字符 a。

2.2.2　实型常量

实数又称浮点数，就是日常生活中的带小数点的数。实型常量有两种表示形式：

（1）十进制小数形式。该形式的实型常量由数字和小数点组成（注意必须包含小数点），如.123、123.、123.0、0.0 都是十进制小数。其中 123.0 末尾的数字 0 可以省略，.123 小数点前面的 0 可以省略，0.0 可以省略其中一个 0 但不能都省略。

（2）指数形式。如 123e3 或 123E3 都代表 123×10^3，但应注意字母 e（或 E）之前必须包含数字，且 e 后面的指数必须为整数，如 e3，2.1e3.5，.e3，e 等都不是合法的指数形式实型常量。

一个实数可以有多种指数表示形式。例如 123.456 可以表示为 12.3456e1，1.23456e2，0.123456e3 等。其中 1.23456e2 称为“规范化的指数形式”，即在字母 e（或 E）之前的小数部分中，小数点左边有且只能有一位非零的数字。一个实数在用指数形式输出时，是按规范化的指数形式输出的。例如，指定将实数 5689.65 按指数形式输出，必然输出 5.68965e+003。

2.2.3　字符型常量

由字符组成的、其值不能被改变的量称为字符型常量。根据组成结构的不同，字符型常量可分为字符常量、转义字符和字符串常量 3 种。

1. 字符常量和转义字符

C 语言的字符常量是用单引号“'”括起来的一个字符。如'a'，'x'，'D'，'?'，'$'等都是字符常量。

除了以上形式的字符常量外，C 还允许使用一种特殊形式的字符常量，即以一个“\”符号开头的字符序列。例如，前面已经遇到过的 printf 函数中的'\n'，它代表一个换行符。这是一种控制字符，在屏幕上是不能显示的，在程序中也无法用一个一般形式的字符表示，只能采用特殊形式来表示。“\”使后面的一个字符失去了原有的意义，因此又称作转义字符。例如，'\n'中的“n”不代表字母 n，而作为回车换行符处理。转义字符是一种特殊形式的字符常量，主要用来表示控制代码和图形符号。常用的转义字符如表 2.1 所示。

表 2.1　常用转义字符

字符形式	等价于	ASCII 码	意义
\a	\x07	7	鸣铃
\b	\x08	8	退格，将当前位置移到前一列
\f	\X0C	12	走纸换页

续表

字符形式	等价于	ASCII 码	意义
\n	\X0A	10	回车换行
\r	\X0D	13	回车，将当前位置移到本行开头
\t	\X09	9	水平制表（跳到下一个 tab 位置）
\\	\X5C	92	反斜杠字符“\”
\'	\X27	39	单引号字符
\"	\X22	34	双引号字符
\ddd			ddd 代表 1～3 位八进制数所代表的字符
\xhh			hh 代表 1～2 位十六进制数所代表的字符

表 2.1 中，\ddd 是用八进制数表示的 ASCII 码。例如，用八进制数'\102'（相当于十进制数 66）代表 ASCII 字母“B”；用'\033'代表 Esc 键。从 ASCII 码表中能够查到常用的转义字符。

\xhh 是用十六进制数表示的 ASCII 码。例如，用'\x1B'同样可以代表 Esc 键。

由此可知，用转义字符可以表示任一 ASCII 码；并且任一 ASCII 码可以有多种表示方法。

有关字符常量的几点说明如下：

（1）单引号本身只作定界符使用，而不是字符常量的一部分。

（2）因字符“'”和“\”在字符常量中有特殊用途，因此它们不能直接作为字符常量使用。若要将这两个字符用作字符常量，应写为'\"'和'\\'，即在这些字符前再加一个反斜杠“\”。

（3）英文字母区分大小写。注意，'a'和'A'是不同的字符常量。

（4）字符常量具有数值，其值对应于 ASCII 码值，是 0～255 之间的整数。例如，'A'的值是 65，'a'的值是 97。因此，字符型常量与整型常量可以混合使用。在不至于引起混淆的情况下，0～255 之间的整数可以用字符常量表示。例如，63 可写成'A'-2，反之亦然。

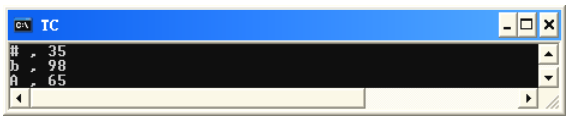
（5）输出字符常量时的输出格式控制符为%c。

例 2.1　字符型常量与整型常量的混合使用

要求程序输出字符“#”以及其 ASCII 码值，求'a'+1 的值并输出其 ASCII 字符，输出整数 65 以及该数字所对应的 ASCII 字符。

```
main()
{
   printf("%c, %d \n", '#', '#' );
   printf("%c, %d \n", 'a'+1, 'a'+1 );
   printf("%c, %d \n", 65, 65 );
}
```

运行结果如下。



2. 字符常量和字符串常量

字符常量是由一对单引号括起来的单个字符。C 语言除了允许使用字符常量外，还允许使用字符串常量。字符串常量是由一对双引号括起来的字符序列，如"How do you do."，"CHINA"，"a"，"$123.45"都是字符串常量。C 语言中可以直接输出一个字符串。如：

```
printf("How do you do.");
```

不要将字符常量与字符串常量混淆。'a'是字符常量，"a"是字符串常量，二者不同。那么，'a'和"a"究竟有什么区别？C 规定，在每一个字符串的结尾，需要添加一个字符串结束标志，以便系统据此判断字符串是否结束。C 规定以字符'\0'作为字符串结束标志。'\0'是一个 ASCII 码为 0 的字符，是“空操作字符”，即它不引起任何控制动作，是一个不可显示的字符。例如，字符串"CHINA"实际上在内存中存储的是：

C	H	I	N	A	\0

它的长度不是 5 个字符，而是 6 个字符，最后一个字符为'\0'。但在输出时不输出'\0'。例如在 printf("how do you do.")中，输出时一个一个字符地输出，直到遇到最后的'\0'字符，系统就知道字符串结束，停止输出。注意，在写字符串时不必加'\0'，否则会画蛇添足。'\0'字符是系统自动加上的。字符串"a"，实际上包含两个字符：'a'和'\0'。

在 C 语言中没有专门的字符串变量，如果想将一个字符串存放在变量中，必须使用字符数组，即用一个字符型数组来存放一个字符串，数组中每一个元素存放一个字符。字符型数组将在后面章节中介绍。

2.2.4　符号常量

在程序设计中，对于某些有特定含义的、经常使用的常量可以用标识符来代替。用标识符代替常量，可增加程序的可读性和可维护性。

在 C 语言中，代表常量的标识符称为符号常量。符号常量在使用之前必须先定义，定义的一般格式为：

#define 标识符 常量

其中#define 表示宏定义的一个命令，宏定义必须写在一行上。标识符称为“宏名”，字符串称为“宏体”，其功能是把该标识符定义为其后的常量值。一经定义，在程序中出现的所有该标识符均代表该常量值。习惯上符号常量的标识符为大写字母，变量标识符为小写字母，以示区别。

例 2.2　宏定义的使用

```
#define PI 3.14159          /*宏定义命令，定义 PI 为 3.14159*/
main()
{
  float s,r;                /*将 s，r 定义为实型变量，用来存放实型数据*/
  r=5;                      /*5 赋值给 r*/
  s=PI*r*r;
  printf("s=%f\n",s);
}
```

运行结果如下。

本程序在主函数之前由宏定义命令定义 PI 为 3.14159，在程序中用 3.14159 代替 PI。s=PI*r*r 等效于 s=3.14159*r*r。应该注意的是，符号常量不是变量，它所代表的值在整个作用域内不能再改变。也就是说，在程序中，不能再用赋值语句对它重新赋值。

使用符号常量的好处是：

（1）含义清楚。如在上面的程序中，看到 PI 就可知道它代表“π”。因此定义符号常量名时应“见名知义”。

（2）在需要改变一个常量时能做到“一改全改”。

在定义符号常量时，应注意以下几点：

（1）如果程序中有多个符号常量，必须用多个命令行分别定义，即一行只能定义一个符号常量。例如：

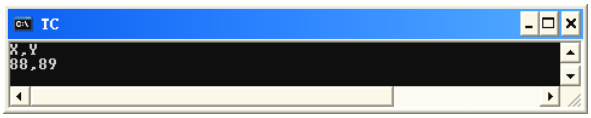
```
#define   PI   3.1415926
#define   TRUE   1
```

（2）符号常量代表的常量可以是前文介绍的各种形式的常量，还可以代表程序的其他语法成分。

例 2.3 符号常量的使用

```
#define   P   printf
main()
{
   char a,b;
   a='x';
   b='y';
   a=a-32;
   b=b-32;
   P("%c,%c\n%d,%d\n",a,b,a,b);
}
```

运行结果如下。



这里用 P 代表了库函数名 printf。

（3）符号常量的定义可以放在程序的任何位置，但必须在使用之前预先定义，因此一般放在程序的开始位置。

2.3 变量

在程序运行过程中，其值可以改变的量称为变量。一个变量必须有一个名字即变量名，变量名要遵守标识符的命名规则。变量名在内存中占据一定的存储单元，在该存储单元中存放变量的值。

下面分别介绍整型、实型（浮点型）、字符型变量。

2.3.1　整型变量

1. 整型变量的分类

整型变量为用来存放整型常量。根据数值的范围将变量定义为基本整型、短整型或长整型，分别表示为：

（1）基本整型，以 int 表示。

（2）短整型，以 short int 或 short 表示。

（3）长整型，以 long int 或 long 表示。

在 Turbo C 中一个 int 型的变量值的范围为-2^{15}～（2^{15}-1），即-32768～32767。可以将变量定义为“无符号”和“有符号”两种类型，即加上修饰符 unsigned 或修饰符 signed。如果既不指定为 signed，也不指定为 unsigned，则默认为有符号类型（signed）。在 Turbo C 中，可以使用以下 6 种整型变量，即：

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

C 语言中没有具体规定以上各类数据所占内存字节数，只要求 long 型数据的长度不短于 int 型，short 型不长于 int 型。具体如何实现，由各计算机系统自行决定。

Turbo C 系统对各类整型变量分配的内存空间如表 2.2 所示。

表 2.2　整型变量

数据类型	所占字节数（个）	所占位数（bit）
int	2	16
short	2	16
long	4	32
unsigned int	2	16
unsigned short	2	16
unsigned long	4	32

一个 unsigned int 型的变量值的范围为 0～65535。

在计算机系统内整型数据占两个字节。以 2 为例，将 2 转化为二进制数之后，它的存储形式为：

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

2. 整型变量的定义

C 规定在程序中所有用到的变量都必须先定义后使用，即“强制类型定义”。每个变

量必须要有确定的类型。整型变量定义的一般形式如下：

整型数据类型名　　变量名;

例如：

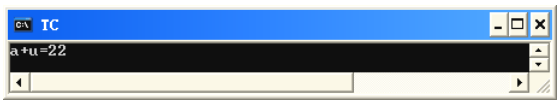
```
int a,b;                          （指定变量 a、b 为整型）
unsigned short c,d;               （指定变量 c、d 为无符号短整型）
long e,f;                         （指定变量 e、f 为长整型）
```

变量的定义一般放在一个函数开头的声明部分（也可以放在函数中的某一子程序内，但作用域只限于它所在的子程序）。C 语言允许在一个定义中定义多个变量，变量之间用逗号隔开。变量定义之后，编译系统就会为其分配相应的存储空间。

例 2.4　整型变量的定义与使用

```
main()
{ int a,b;              /*定义 a，b 为基本整型变量*/
  unsigned u;           /*定义 u 为无符号整型变量*/
  a=12;u=10;
  b=a+u;
  printf("a+u=%d\n",b);
}
```

运行结果如下。



可以看到不同种类的整型数据可以混合进行算术运算。在本例中是 int 型数据与 unsigned int 型数据进行相加相减运算。

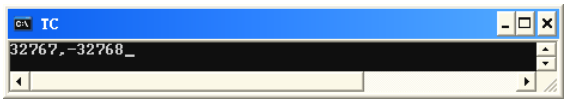
3. 整型数据的溢出

在 Turbo C 中一个 int 型变量的最大允许值为 32767，如果再加 1，会出现什么情况？

例 2.5　整型数据的溢出

```
main()
{ int a,b;
  a =32767;
  b=a+1;
  printf("%d,%d",a,b);
}
```

运行结果如下。



需要注意的是，一个整型变量只能容纳-32768～32767 范围内的数，无法表示大于 32767 的数。遇此情况就发生“溢出”，但运行时并不报错。溢出就好像日常生活中的表一样，达到最大值 24 点以后，又从最小值 0 点开始计时。所以，32767 加 1 得不到 32768，而得到-32768，这可能与程序编写者的原意不同。从这里可以看：C 的用法比较灵活，往往出现副作用，而系统又不给出“出错信息”，要靠程序员的细心和经验来保证结果的正

确。在本例中，将变量 b 改成 long 型就可得到预期的结果 32768。

2.3.2　实型变量

1. 实型变量的分类

C 语言中实型变量用来存放实型常量。实型变量分为单精度型（float）、双精度型（double）和长双精度型（long double）。

虽然分为 3 种，但是长双精度型（long double）在初学阶段用得较少，因此本书不作详细介绍。

ANSI C 并未具体规定每种类型数据的长度、精度和数值范围。有的系统将 double 型所增加的 32 位全用于存放小数部分，这样可以增加数值的有效位数，减少舍入误差。有的系统则将所增加的位（bit）用于存放指数部分，这样可以扩大数值的范围。表 2.3 列出的是计算机上常用的 C 编译系统（如 Turbo C，MS C，Borland C）的情况。应当了解，不同的系统的分配状况会有差异。

表 2.3　实型变量

数据类型	所占字节数（个）	所占位数（bit）	有效位数
float	4	32	6～7
double	8	64	15～16

2. 实型变量的定义

每一个实型变量在使用前都必须先定义。

实型变量定义的一般形式如下：

实型数据类型名　　变量名;

例如：

```
float x,y,          （指定 x，y 为单精度实数）
double z;           （指定 z 为双精度实数）
```

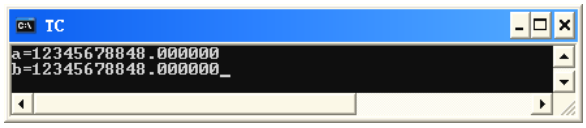
3. 实型数据的使用

由于实型变量是由有限的存储单元组成的，因此能提供的有效数字总是有限的，有效位以外的数字将被舍去，由此可能会产生一些误差。

例 2.6　实型数据的舍入误差

```
main()
{ float   a,b;
  a=123456.789e5;
  b=a+20;
  printf("a=%f\nb=%f",a,b);
}
```

运行结果如下。



程序内 printf 函数中的“%f”是输出一个实数时的格式符。程序运行时，输出 b 的值与 a 相等。原因在于 a 的值比 20 大很多，a+20 的理论值应是 12345678920，而一个实型变量只能保证有效数字是 7 位数字，后面的数字是无意义的，并不准确地表示该数。运行程序得到的 a 和 b 的值是 12345678848.000000，可以看到，前 8 位是准确的，后几位是不准确的，把 20 加到后几位上是无意义的。应当避免将一个很大的数和一个很小的数直接相加或相减，否则就会“丢失”较小的数。如果想得到预期效果，可以把 float 改为 double 类型。与此类似的问题有很多，读者在编程时应当注意。

2.3.3 字符型变量

1. 字符型变量的分类

字符型变量用来存放字符常量，C 语言的字符型变量分为有符号型（char）变量和无符号型（unsigned char）变量，如表 2.4 所示。

表 2.4 字符型变量

数据类型	所占字节数（个）	所占位数（bit）	取值范围
char	1	8	-128～127
unsigned char	1	8	0～255

2. 字符变量的定义

使用字符型变量时必须先定义。请注意字符型变量在内存中占用 1 个字节的存储空间，只能存放一个字符，所以不能在一个字符变量中存放一个字符串（包括若干字符）。

定义的一般形式如下：

字符型数据类型名　变量名;

例如：

```
char c1,c2;
```

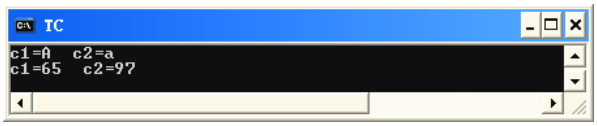
其中 c1 和 c2 为字符型变量，各可以存放一个字符。

3. 字符型变量的使用

例 2.7 向字符变量赋以整数

```
main()
{ char c1,c2;
  c1=65;
  c2=97;
  printf("c1=%c  c2=%c\n",c1,c2);
  printf("c1=%d  c2=%d\n",c1,c2);
}
```

运行结果如下。



本例中定义 c1，c2 为字符变量。但在程序第 3 行和第 4 行中，将整数 65 和 97 分别

赋给 c1 和 c2，它的作用相当于以下两个赋值语句：

```
c1='A';  c2='a';
```

因为'A'和'a'的 ASCII 码为 65 和 97，所以在程序的第 3 行和第 4 行中是把 65 和 97 两个整数直接存放到 c1 和 c2 内存单元中。“%c”是输出字符时必须使用的格式符。可以看到，字符型数据和整型数据是通用的。它们既可以以字符格式输出（用“%c”），也可以以整数形式输出（用“%d”）。但应注意字符数据只占一个字节，它只能存放 0～255 或-128～127 范围内的整数。本书对有符号数不做详细介绍。

例 2.8　大小写字母的转换

```
main()
{
   char c1, c2;
   c1='a';
   c2='b';
   c1=c1-32;
   c2=c2-32;
   printf("%c %c",c1,c2);
}
```

运行结果如下。

程序的作用是将两个小写字母 a 和 b 转换成大写字母 A 和 B。'a'的 ASCII 码为 97，而'A'为 65，'b'为 98，'B'为 66。从 ASCII 码表中可以看到每一个小写字母的 ASCII 码比它相应的大写字母的 ASCII 码大 32。C 语言允许字符型数据与整数型数据直接进行算术运算，即'A'+32 会得到 97，'a'-32 会得到 65。

通过以上示例可以看出，在 C 语言中，要求对所有用到的变量作强制定义，也就是“先定义，后使用”。这样做的目的是：

（1）凡未被事先定义的变量都不可用，这就能保证程序中变量名的使用正确。例如，如果在定义部分进行如下定义：

```
int abc;
```

而在执行语句中错写成 acb，如：

```
acb=30;
```

在编译时检查出 acb 未经定义，不作为变量名使用，因此输出“变量 acb 未经声明”的信息，便于用户发现错误，避免使用变量名时出错。

（2）每一个变量被指定为一确定类型，在编译时就能为其分配相应的存储单元。如指定 a、b 为 int 型，Turbo C 编译系统为 a 和 b 各分配两个字节，并按整数方式存储数据。

（3）指定每一变量属于一个类型，这就便于在编译时据此检查该变量所进行的运算是否合法。例如，整型变量 a 和 b 可以进行求余运算：

```
a%b
```

%是求余操作符，运算后可得到 a/b 的余数。如果将 a、b 指定为实型变量，则不允许进行求余运算，在编译时会显示有关的出错信息。

2.3.4 对变量赋初值

程序中常需要对一些变量赋初值，赋初值的方法如下：

（1）C 语言允许在定义变量的同时使变量初始化。如：

```
int a=3:                    /*指定 a 为整型变量，初值为 3*/
float f=3.56;               /*指定 f 为实型变量，初值为 3.56*/
char c='a';                 /*指定 c 为字符型变量，初值为'a'*/
```

（2）为被定义的变量的一部分赋初值。如：

```
int a,b,c=5;
```

表示指定 a、b、c 为整型变量，但只对 c 初始化，c 的初值为 5。

（3）对于几个变量同时赋初值。如：

```
int a=3,b=3,c=3;
```

表示 a、b、c 的初值都是 3。但不能写成：

```
int a=b=c=3;
```

初始化不是在编译阶段完成的，而是在程序执行本函数时完成的。

2.4 运算符和表达式

C 语言中运算符和表达式的数量很多。丰富的运算符和表达式使 C 语言的功能十分完善，这也是 C 语言的主要特点之一。

C 语言的运算符不仅具有不同的优先级，而且还有一个特点——结合性。在表达式中，各运算量参与运算的先后顺序不仅要遵守运算符优先级别的规定，还要受运算符结合性的制约，以便确定是自左向右进行运算还是自右向左进行运算。这种结合性是其他高级语言的运算符所没有的，因此也增加了 C 语言的复杂性。

2.4.1 算术运算符和算术表达式

（1）C 语言提供了丰富的算术运算符，能够完成各种复杂的运算。基本的算术运算符如下：

1）“+”加法运算符：双目运算符，即应有两个量参与加法运算，如 a+b，4+8 等。该运算符具有自左至右的结合性。此外，“+”也可以表示取正值运算符，属于单目运算符，具有自右至左的结合性。

2）“-”减法运算符：双目运算符。该运算符也可以表示取负值运算符，属于单目运算符，如-x，-5 等，具有自右至左的结合性。

3）“*”乘法运算符：双目运算符，具有自左至右的结合性。

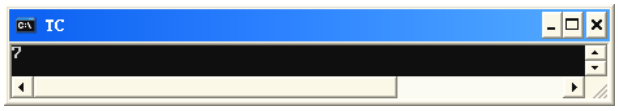
4）“/”除法运算符：双目运算符，具有自左至右的结合性。参与运算量均为整型时，结果也为整型，舍去小数。如果运算量中有一个是实型，则结果为双精度实型。

例 2.9 除法运算

```
main()
{
   int   a=15,b=2,c;
```

```
    c=a/b;
    printf("%d ",c);
}
```

运行结果如下。



参与运算量均为整型时，结果也为整型，舍去小数。如果运算量中有一个是实型，则结果为双精度实型。

5）“%”求余运算符（模运算符）：双目运算符，具有自左至右的结合性。该运算符要求参与运算的量必须为整型。求余运算的结果是两数相除后的余数，如 100%3 的结果为 1，1.5%2 是非法运算。

6）算术运算符中运算顺序和数学中的是一样的，先乘、除、求余，后加减。

（2）算术表达式。算术表达式是由算术运算符和括号连接起来的式子，运算对象可以是常量、变量和函数等。算术表达式的值可以是整型或实型。以下是算术表达式的例子：

a+b，　(a*2)/c，(x+r)*8-(a+b)/7，　sin(x)+sin(y)

其中 sin(x)是求解正弦函数，应用 sin(x)时应在程序的开始部分加上#include "math.h"。

在 C 语言中使用算术运算符应注意以下问题：

1）算术表达式中的“*”不能省略。如三角形的面积公式 s=ab/2 在 C 语言中应写为 s=a*b/2。

2）算术表达式中只能出现 C 语言中允许使用的运算字符。如数学面积公式 πr^2 中的 π，在 C 语言中没有这个字符，需要通过宏定义的方法实现。

3）算术表达式中只能通过()来改变运算对象的运算顺序。

2.4.2　自增、自减运算符

“++”为自增 1 的运算符，其功能是使变量的值自增 1。

“--”为自减 1 的运算符，其功能是使变量的值自减 1。

自增 1、自减 1 运算符均属于单目运算符，都具有右结合性。可有以下几种形式：

（1）++i：i 自增 1 后再参与其他运算。

（2）--i：i 自减 1 后再参与其他运算。

（3）i++：i 参与运算后，i 的值再自增 1。

（4）i--：i 参与运算后，i 的值再自减 1。

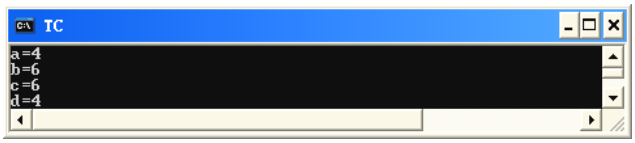
在理解和使用上容易出错的是 i++和 i--，特别是当它们出现在较复杂的表达式或语句中时，常常难于弄清，因此应仔细分析。

例 2.10　自增自减运算（一）

```
main()
{
   int i=4,a,b,c,d;
   a=i++;
   b=++i;
```

```
    c=i--;
    d=--i;
    printf("a=%d\nb=%d\nc=%d\nd=%d",a,b,c,d);
}
```

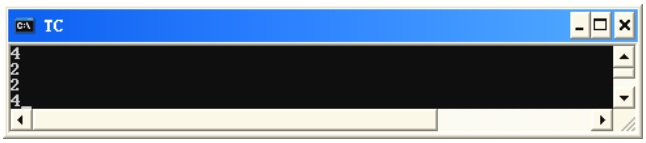
运算结果如下。



例 2.11　自增自减运算（二）

```
main()
{
   int i=4;
   printf("%d\n%d\n%d\n%d",++i,i++,--i,i--);
}
```

运行结果如下。



两个例子中的区别在于 printf 函数。在 C 语言中 printf 函数的参数应从后向前运算，在本例中的运算顺序为 i--，--i，i++，++i。

2.4.3　赋值运算符和赋值表达式

1. 赋值运算符

（1）简单赋值运算符“=”。在 C 语言中，“=”称为赋值运算符。它的作用是将一个数据赋给一个变量，与数学中的等号不同，它的结合性是自右至左的。赋值运算符的优先级仅高于逗号运算符。

（2）复合赋值运算符。C 语言中提供了+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=等复合运算符。复合运算符的构成方式是在赋值运算符“=”之前加上其他二目运算符。

2. 赋值表达式

由赋值运算符将变量和表达式连接起来的式子称为赋值表达式。

（1）简单赋值表达式的一般形式为：

变量=表达式

例如：

x=a+b

w=sin(a)+sin(b)

y=i++

赋值表达式的功能是计算表达式的值后再将其赋予表达式左边的变量。例如：

a=b=c=5

可理解为：

a=(b=(c=5))

（2）复合赋值表达式的一般形式为：

变量 双目运算符=表达式

例如：

a+=5 等价于 a=a+5

x*=y+7 等价于 x=x*(y+7)

r%=p 等价于 r=r%p

在 C 中，凡是表达式可以出现的地方均可出现赋值表达式。例如，表达式 x=(a=5)+(b=8) 是合法的。它的意义是把 5 赋予 a，8 赋予 b，再把 a，b 相加，和赋予 x，故 x 应等于 13。

在 C 语言中也可以组成赋值语句，按照 C 语言的规定，任何表达式在其末尾加上分号就构成了语句。因此“x=8;”，“a=b=c=5;”都是赋值语句，在前面各例中已大量使用过赋值语句了。

3. 赋值运算的类型转换

如果赋值运算符两边的数据类型不相同，系统将自动进行类型转换，即把赋值号右边的类型换成左边的类型。具体规定如下：

（1）实型赋予整型，舍去小数部分。

（2）整型赋予实型，数值不变，但将以浮点形式存放，即增加小数部分（小数部分的值为 0）。

（3）字符型赋予整型，由于字符型为一个字节，而整型为两个字节，故将字符的 ASCII 码值放到整型量的低 8 位中，高 8 位为 0。

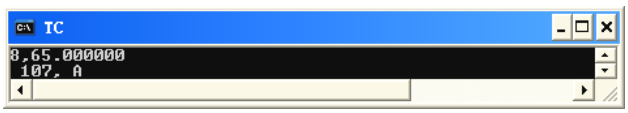
（4）整型赋予字符型，只把低 8 位赋予字符量。

总之，不同整型数据间的赋值归根到底是按照存储单元的存储形式直接赋给的。

例 2.12　赋值运算

```
main()
{
    int a,b=65;
    float x,y=8.88;
    char c1='k',c2;
    a=y;
    x=b;
    printf("%d,%f\n ",a,x);
    a=c1;
    c2=b;
    printf("%d, %c",a,c2);
 }
```

运行结果如下。



本例说明了赋值运算中的类型转换规则。a 为整型变量，被赋予实型量 y 值 8.88 后，舍去小数部分，只取整数 8。x 为实型变量，被赋予整型量 b 值 65 后，增加了小数部分。字符型变量 c1 的值赋予整型变量 a（直接把 c1 的一个字节放到了变量 a 的低字节中，高字节补 0），整型变量 b 的值赋予字符型变量 c2，是将整型变量 b 的低 8 位赋给字符型变量 c2（b 的低 8 位为 01000001，即十进制 65，按 ASCII 码对应于字符 A）。

2.4.4 逗号运算符和逗号表达式

（1）逗号运算符。C 语言中的逗号“,”是一种运算符，称为逗号运算符。

（2）逗号表达式。逗号表达式是把两个表达式连接起来而组成的一个表达式。一般形式为：

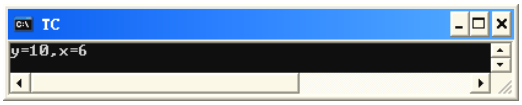
表达式 1，表达式 2…表达式 n

逗号表达式的结合方向是自左至右的，先求解表达式 1，然后求解表达式 2……最后求解表达式 n 的值，并将表达式 n 的值作为整个逗号表达式的值。

例 2.13 逗号表达式的使用

```
main()
{
  int a=2,b=4,c=6,x,y;
  y=((x=a+b),(b+c));
  printf("y=%d,x=%d",y,x);
}
```

运行结果如下。



本例中，“(x=a+b),(b+c)”是一个逗号表达式，y 的值是整个逗号表达式的值，也就是表达式(b+c)的值，x 是第一个表达式的值。对于逗号表达式还要说明几点：

1）程序中使用逗号表达式时，要分别求逗号表达式内各表达式的值。

2）并不是在所有出现逗号的地方都组成逗号表达式，如变量说明和函数参数表中逗号只是作为各变量之间的间隔符。

3）在本章涉及的运算符中逗号运算符的优先级最低，其次为赋值运算符，优先级最高的是算术运算符。

2.4.5 条件运算符

（1）条件运算符。C 语言中把“?:”称为条件运算符。条件运算符要求有 3 个操作对象，是 C 语言中唯一的三目运算符，它连接 3 个运算分量。

（2）条件表达式。条件表达式的一般形式如下：

表达式 1?表达式 2:表达式 3

它的执行过程是：先计算表达式 1，如果其值为真，则求解表达式 2 的值并将其作为结果值，否则求解表达式 3 的值作为结果值。例如：

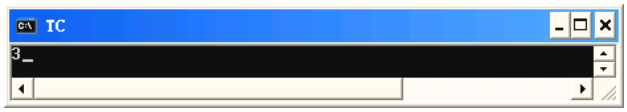
```
max=(a>b)?a:b;
```

该语句执行时，当 a>b 时条件成立，变量 max 取 a 值，否则取 b 值。

例 2.14　条件运算符的使用

```
main()
{
   int a,b;
   a=2;b=3;
   printf("%d",(a>b)?a:b);
}
```

运算结果如下。



条件运算符的优先级较低，只高于赋值运算符和逗号运算符。

2.4.6　长度运算符

C 语言中的 sizeof()称为长度运算符。长度运算符是单目运算符，主要用于计算变量或数据类型所占的内存字节数的大小。

长度运算符一般形式如下：

（1）sizeof(数据类型名)：计算数据类型在内存中所占的字节数。

（2）sizeof(变量名)：计算变量在内存中所占的字节数。

例 2.15　长度运算符的使用

```
main()
{ int a;
   float b;
   printf("%d,%d,%d,%d,%d",sizeof(a),sizeof(int),sizeof(b),sizeof(float),sizeof(double));
}
```

运行结果如下。

a 是整型变量，所占字节数为 2；int 是整型变量的数据类型，所占字节数为 2；b 是实型单精度变量，所占字节数为 4；float 是实型单精度变量类型，所占字节数为 4；double 是实型双精度类型，所占字节数为 8。

2.5　数据类型的转换

变量的数据类型是可以转换的。转换的方法有两种，一种是自动转换，一种是强制转换。

2.5.1 自动转换

自动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。自动转换遵循以下规则，如图 2-1 所示。

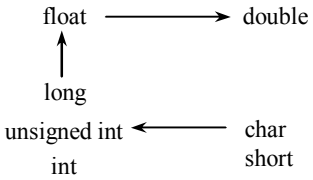


图 2-1 自动转换规则

（1）若参与运算的量的类型不同，则先转换成同一类型，然后进行运算。

（2）转换按数据长度增加的方向进行，以保证精度不降低。如 int 型和 long 型混合运算时，应先把 int 型转成 long 型后再进行运算。

（3）所有的浮点运算都是按双精度进行的，即使仅含有 float 单精度量的运算表达式，也要先转换成 double 型后再作运算。

（4）char 型和 short 型参与运算时，必须先转换成 int 型。

（5）在赋值运算中，赋值号两边的量的数据类型不同时，赋值号右边量将转换为左边量的类型。如果右边量的数据类型长度比左边长时，将丢失一部分数据，这样会降低精度。下面程序段表示了类型自动转换的规则。

例 2.16 类型的自动转换

```
main()
{
   int s;
   float a=5.5;
   s=a;
   printf("s=%d\n",s);
}
```

运行结果如下。

本例程序中，a 为实型变量，赋初值 5.5；s 为整型变量。在执行 s=a 语句后，由于 s 为整型，赋值结果仍为整型，舍去了小数部分，结果为 5。

2.5.2 强制类型转换

1. 强制类型转换运算符

“()”在 C 语言中称为强制类型转换运算符。一般形式为：

(类型说明符) (表达式)

2. 强制类型转换运算符的使用

强制类型转换运算符的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 转换为实型

(int)(x+y)　　把 x+y 的结果转换为整型

3. 在使用强制转换时应注意的问题

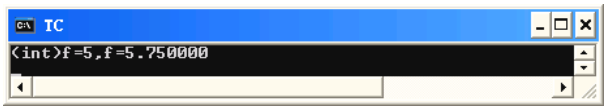
（1）类型说明符和表达式都必须加括号（单个变量可以不加括号），如把(int)(x+y)写成(int)x+y，则意为把 x 转换成 int 型之后再与 y 相加。

（2）无论是强制转换还是自动转换，都只是为了本次运算的需要而对变量的数据长度进行的临时性转换，而不改变数据说明时对该变量定义的类型。

例 2.17　强制转换

```
main()
{
  float f=5.75;
  printf("(int)f=%d,f=%f\n",(int)f,f);
}
```

运行结果为如下。



本例中的 printf 函数参数 f 虽强制转为 int 型，但只在运算中起作用，是临时的，而 f 本身的类型并不改变。因此，(int)f 的值为 5（删去了小数部分），而 f 的值仍为 5.75。

本章小结

本章首先介绍了数据类型的概念分类。C 语言的基本数据类型包括整型、实型、字符型和枚举型（后面的章节将会介绍）。根据在程序中值是否可以改变，基本数据类型分为常量和变量。常量包括整型常量、实型常量、字符型常量、字符串常量和符号常量等几种。整型常量可以用八进制、十进制和十六进制表示，实型常量可以用小数和指数形式表示，字符型常量可以用单引号括起来的量表示，字符串常量可以用双引号括起来的量表示。

变量必须先定义后使用。变量名必须符合标识符的命名规则。本章中涉及的变量有整型变量、实型变量和字符型变量。不同类型的变量所占的内存空间是不一样的。

C 语言提供了丰富的运算符，用它们可以组织各种类型的表达式。使用运算符一定要注意运算符的运算规则、优先级和结合性。

不同数据之间可以进行混合运算。在运算时，需要先将不同类型的数据要先转化成同一类型，然后再进行计算。数据类型的转换总是从低级别向高级别类型转换。

习　题

一、填空题

1．C 语言中基本的类型变量的关键字包括_____。

2．在 C 语言中，一个 char 型数据在内存中所占的字节数为_____；一个整型数据在内存中所占的字节数为_____；一个实型数据在内存中所占的字节数为_____。整型数据的取值范围是_____。

3．若 int a=2,b=3;float x=3.5,y=2.5;下面表达式的值分别为_____、_____。

```
(float)(a+b)/2+(int)x+(int)y
(a+b)%2+(int)x/(int)y
```

4．若 a 是 int 型变量，下面表达式的值为_____。

```
(a=4*5,a*2),a+6
```

二、选择题

1．设变量 a 是整型，f 是实型，i 是双精度型，则表达式 10+'a'+i*f 的值的数据类型为（　）。

A．int　　B．float　　C．double　　D．不确定

2．若有以下定义，则下列表达式中值为 3 的是（　）。

```
int k=7,x=12;
```

A．x%=(k%=5)　　B．x%=(k-k%5)

C．x%=k-k%5　　D．(x%=k)-(k%=5)

3．在 C 语言中一个 int 型数据在内存中占两个字节，则 unsigned int 型数据的取值范围为（　）。

A．0～255　　B．0～32767　　C．0～65535　　D．0～2147483647

4．若运行时将变量 x 赋值为 12，则以下程序的运行结果是（　）。

```
main()
{ int x,y;
  scanf("%d",&x);
  y=x>12?x+10:x-12;
  printf("%d\n",y);
}
```

A．0　　B．22　　C．12　　D．10

5．若有定义“int a=7;float x=2.5,y=4.7;”，则表达式 x+a%3*(int)(x+y)%2/4 的值是（　）。

A．2.500000　　B．2.750000　　C．3.500000　　D．0.000000

6．若以下变量均为整型，且 num=sum=7，则计算表达式“sum=num++,sum++,++num”后 sum 的值为（　）。

A．7　　B．8　　C．9　　D．10

7．下列程序的输出结果是（　）。

```
main()
{ double d=3.2; int x,y;
  x=1.2;
  y=(x+3.8)/5.0;
  printf("%d \n", d*y);
}
```

A．3　　B．3.2　　C．0　　D．3.07

8．有如下程序：

```
main()
{ int y=3,x=3,z=1;
  printf("%d %d\n",(++x,y++),z+2);
}
```

该程序的输出结果是（　）。

A．3 4　　B．4 2　　C．4 3　　D．3 3

9．若变量 a、i 已正确定义，且 i 已正确赋值，则下列语句中合法的是（　）。

A．a==1　　B．++i;　　C．a=a++=5;　　D．a=int(i);

10．若已定义 x 和 y 为 double 类型，则表达式“x=1,y=x+3/2;”的值是（　）。

A．1　　B．2　　C．2.0　　D．2.5

11．有以下程序：

```
main()
{ int i=10,j=1;
  printf("%d,%d\n",i--,++j);
}
```

该程序的输出结果是（　）。

A．9,2　　B．10,2　　C．9,1　　D．10,1

12．在 C 语言中，要求运算数必须是整型的运算符是（　）。

A．/　　B．++　　C．!=　　D．%

三、编程题

编写一个程序，要求从键盘上输入任意两个数后，求它们加、减、乘、除和求余的值。