# 第 5 章　C 语言程序设计基础知识

程序处理的对象是数据，编写程序也就是描述对数据的处理过程。在编写程序的过程中，必然要涉及数据本身的描述以及数据运算的问题。

## 5.1　常量与变量

在 C 语言中，任何数据对用户呈现的形式有两种：常量和变量。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、实型常量、实型变量、字符常量、字符变量等。在程序中，常量是可以直接使用的，而变量则必须先定义后使用。

### 5.1.1　常量

常量是 C 语言中使用的基本数据对象之一。C 语言提供的常量如下：

常量
- 数值常量
  - 整型常量
    - 数值常量
    - 符号常量
  - 实型常量
- 字符常量
  - 字符常量
  - 字符串常量

以上是常量所具有的类型属性，这些类型决定了各种常量所占存储空间的大小和数的表示范围。在 C 程序中，常量是直接以自身的存在形式体现其值和类型的。例如，123 是一个整型常量，整型常量占两个字节；123.0 是实型常量，实型常量占 4 个字节。

在 C 程序中，常量除了以自身的存在形式直接表示之外，还可以用标识符即符号常量来表示。常量本身是一个较长的字符序列，且在程序中重复出现。例如，取常数 PI 的值为 3.141 592 7；如果 PI 在程序中多处出现，直接使用 3.141 592 7 的表示形式，势必会使编程工作变得繁琐，而且当需要把 PI 的值修改为 3.141 592 653 6 时，就必须逐个查找并修改，这样会降低程序的灵活性。因此，C 语言中提供了一种符号常量，即用指定的标识符来表示某个常量，在程序中需要使用该常量时就可直接引用标识符。

C 语言中对符号常量进行定义的形式如下：

```
#define 标识符 常量
```

其中，#define 是宏定义命令的专用定义符，标识符是对常量的命名，常量可以是前面介绍的几种类型常量中的任何一种。用指定的标识符来代表指定的常量，这个被指定的标识符就称为符号常量。例如，在 C 程序中，要用 PI 代表实型常量 3.141 592 7，用 W 代表字符串常量"Windows XP"，可用下面两个宏定义命令：

```
#define  PI  3.1415927
```

```
#define  W  "Windows XP"
```

宏定义的功能是:在编译预处理时,将程序中宏定义命令之后出现的所有符号常量用宏定义命令中对应的常量一一代替。例如,对以上两个宏定义命令,编译程序时,编译系统首先将程序中除这两个宏定义命令之外的所有 PI 替换为 3.141 592 7,所有 W 替换为 Windows XP。因此,符号常量通常也被称为宏替换名。

习惯上人们把符号常量名用大写字母表示,而把变量名用小写字母表示。【例 5-1】是符号常量的一个简单应用。其中,PI 为定义的符号常量,程序编译时,用 3.1416 替换所有的 PI。

**【例 5-1】** 已知圆半径 r,求圆周长 c 和圆面积 s 的值。

```
#define PI 3.1416
#include<stdio.h>
void main()
{
  float r,c,s;
  scanf("%f",&r);
  c=2*PI*r;  /*编译时用 3.1416 替换 PI*/
  s=PI*r*r;  /*编译时用 3.1416 替换 PI*/
  printf("c=%6.2f,s=%6.2f\n",c,s);
}
```

程序运行情况如下:

```
2↙
c= 12.57,s= 12.57
```

### 5.1.2　变量

变量是指在程序运行时其值可以改变的量。程序里的一个变量可以看成是一个存储数据的容器,它的功能就是存储数据。对变量的基本操作有以下两个:

(1)向变量中存入数据值,这个操作被称为给变量"赋值"。

(2)取得变量当前值,以便在程序运行过程中使用,这个操作称为"取值"。

**注意:**变量具有保持值的性质,也就是说,如果在某个时刻给某变量赋了一个值,此后使用这个变量时,每次调用的将总是这个值。

程序是通过变量名来使用变量的。要对变量进行"赋值"和"取值"操作,程序里的每个变量都要有一个变量名,变量名就是一个标识符。C 语言规定标识符只能由字母、数字和下划线 3 种字符组成,且第一个字符必须为字母或下划线。

合法的标识符,即合法的变量名如下:

sum、average、_total、class、day、month、student_name、tan、lotus_1_2_3、basic、li_ling。

不合法的标识符和变量名如下:

M.d.John、y 123、#33、3d64、a>b。

C 语言提供的基本变量类型表示如下:

变量 { 数值变量 { 整型变量 / 实型变量 } / 字符变量 }

变量是以标识符的形式来表示其类型的。在C语言中,用类型说明部分对变量进行定义,其定义形式如下:

类型说明符 变量名表;

其中,类型说明符是C语言中的一个有效的数据类型,如整型类型说明符int、字符型类型说明符char等。变量名表的形式是:

变量名1,变量名2,…,变量名n;

即用逗号分隔的变量名的集合,最后用一个分号结束定义。定义变量的这种语言结构称为"变量说明",下面是一些关于变量的说明:

- `int a,b,c;` /*声明a、b、c为整型变量*/
- `char cc;` /*声明cc为字符型变量*/
- `double x, y;` /*声明x、y为双精度实型变量*/

可见,一个定义中可以说明多个变量。而且,由于C语言是自由格式语言,把多个变量说明写在同一行也是允许的。但是为了程序清晰,人们一般不采用这种写法,尤其是初学者。在C程序中,不能用关键字做变量名。C语言中的关键字共32个:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

一般提倡用能说明变量用途的有意义的名字为变量命名,按照"见名知意"的原则,这样有助于提高程序的可读性,尤其是当程序比较大、程序中的变量比较多时,这一点就显得尤其重要。

C语言要求程序里使用的每个变量都必须首先定义,也就是说,首先需要声明一个变量的存在,然后才能够使用它。要定义一个变量需要提供两方面的信息:变量的名称和它的类型。其目的是由变量的类型决定变量的存储结构,以便使C语言的编译程序为所定义的变量分配存储空间。

## 5.2 数据类型

在C语言中,数据类型可分为基本数据类型、构造数据类型、指针类型和空类型四大类。基本数据类型是C语言最常见的数据类型,它的最主要的特点是其值不可以再分解为其他类型。本节主要介绍基本数据类型中的整型数据、实型数据和字符型数据。

### 5.2.1 整型数据

#### 1.整型数据在内存中的存放形式

在计算机中,整型数据的表示范围是有限的,它和计算机的字长有关,字长越长,表示范围越大。数据在内存中是以二进制的形式存放的。Turbo C++ 3.0为一个基本的整型数据分配两个字节的存储单元(不同的编译系统为整型数据分配的字节数是不相同的)。在计算机内部,数值是以补码的形式表示的。正整数的补码和该数的原码(即该数的二进制形式)

相同。负整数的补码是将该数的绝对值的二进制形式按位取反再加 1。

例如，求－1 的补码的过程如下。

1 的原码：

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

按位取反：

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

再加 1，得－1 的补码：

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

由此可知，在存放整数的存储单元中，最左面的一位是数值的符号位，如果该位为 0，则表示该数为正数，如果该位为 1，则表示为负数。数值采用补码表示的另一个优点是可以把加法和减法运算统一为加法运算，减少运算规则。关于补码的知识可以查阅相关的参考书。

**2. 整型数据分类**

C 语言提供了多种整型数据类型，用以适应不同的情况。不同类型的差别在于采用不同位数的二进制编码方式，占用不同的存储空间，所以有不同的数值表示范围。其中，

(1)基本整型：类型说明符为 int，在内存中占两个字节。

(2)短整型：类型说明符为 short int 或 short，所占字节数和取值范围均与基本整型相同。

(3)长整型：类型说明符为 long int 或 long，在内存中占 4 个字节。

以上 3 种类型隐含为有符号(signed)，只是 signed 可以省略不写。无符号型又可与上述 3 种类型匹配而构成：

(1)无符号基本整型：类型说明符为 unsigned int 或 unsigned。

(2)无符号短整型：类型说明符为 unsigned short。

(3)无符号长整型：类型说明符为 unsigned long。

各种无符号数所占的内存空间字节数与相应的有符号数相同。但由于省去了符号位，故能表示的最大数值比有符号数大，但不能表示负数。

以占有两个字节 16 位的基本整型数据来说，有符号基本整型数据最大表示 32 767，各位如下：

| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

无符号基本整型数据最大表示 65 535，各位如下：

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

在实际应用中，一些数据的值是正的(如学号、库存量等)，在这种情况下，为了充分利用变量的取值范围，可以将符合该类条件的变量定义为无符号整型变量。表 5-1 列出了 Turbo C++ 3.0 中各类整型数据所分配的内存字节数及其表示范围。

**表 5-1　整型数据的表示范围及占用字节数**

| 类型说明符 | 表示范围 | 占用字节数 |
|---|---|---|
| [signed] int | －32 768～32 767 即 $-2^{15}\sim(2^{15}-1)$ | 2 |
| unsigned int | 0～65 535 即 $0\sim(2^{16}-1)$ | 2 |

续表

| 类型说明符 | 表示范围 | 占用字节数 |
|---|---|---|
| [signed] short [int] | −32 768～32 767 即$-2^{15}$～$(2^{15}-1)$ | 2 |
| unsigned short [int] | 0～65 535 即 0～$(2^{16}-1)$ | 2 |
| [signed] long [int] | −2 147 483 648～2 147 483 647 即$-2^{31}$～$(2^{31}-1)$ | 4 |
| unsigned long [int] | 0～4 294 967 295 即 0～$(2^{32}-1)$ | 4 |

注:“[]”中的内容表示可以省略。

需要注意的是,使用不同的编译系统时,具体的表示范围有所差别。如 Visual C++ 6.0 为整型数据分配 4 个字节(32 位),其取值范围为−2 147 483 648～2 147 483 647。因此,使用时需要考虑自己所用的编译系统。

**3.整型变量的定义和使用**

C 程序中所有用到的变量都必须在程序中定义,即“强制类型定义”。对整型变量定义的一般形式为:

类型说明符 变量名标识符,变量名标识符,…;

例如,下面都是合法的整型变量定义。

- int a,b,c; /*声明 a、b、c 为整型变量*/
- long x,y; /*声明 x、y 为长整型变量*/
- unsigned p,q; /*声明 p、q 为无符号整型变量*/

C 语言中,在定义变量时,应注意以下几点:

(1)允许在一个类型说明符后,定义多个相同类型的变量,各变量名之间用逗号间隔,类型说明符与变量名之间至少用一个空格间隔。

(2)最后一个变量名之后必须以“;”结尾。

(3)变量定义必须放在变量使用之前,即先定义后使用。一般放在函数体的声明部分。

整型数据是首先接触到的数据类型,为了加深对整型数据的了解,下面给出两个例子。

**【例 5-2】** 整型变量的定义与使用。

```
#include<stdio.h>
void main()
{
  int a,b,c;     /*指定 a、b、c 为整型变量*/
  a=26;b=-24;
  c=a+b;
  printf("c= %d\n",c);
}
```

程序运行结果为:

```
c=2
```

**【例 5-3】** 整型数据的溢出。

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

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

程序运行结果为：

32767,－32768

输出 a 为 32767,而 b 为－32768。这是因为一个基本整型变量只能容纳－32 768～32 767 范围内的数,无法表示大于 32 767 或小于－32 768 的数。遇此情况就发生“溢出”。

**注意**:在 C 语言程序中,当进行数据运算时,要考虑数据溢出问题。为了防止数据溢出,需要定义合适的变量类型,如可将【例 5-3】的数据 a、b 定义为长整型。

#### 4. 整型常量的表示

在 C 语言中,整型数据的输入可以采用八进制、十六进制和十进制 3 种形式。各种进制的格式说明如下：

(1)十进制整型常量。十进制整型常量没有前缀,其数码为 0～9。例如,123、－568、65 535、0 都是合法的十进制整型常量。

(2)八进制整型常量。八进制数必须以 0 开头,即以 0 作为八进制数的前缀,数码取值为0～7。例如,014(十进制为 12)、0101(十进制为 65)、0177777(十进制为 65 535)。

(3)十六进制整型常量。十六进制整常量的前缀为 0X 或 0x,其数码取值为 0～9、A～F 或 a～f 共 16 个数码。例如,0X2C(十进制为 44)、0XBC(十进制为 188)、0XFFFF(十进制为 65 535)。

在程序中是根据前缀来区分各种不同进制数的,因此,在书写常量时不要把前缀弄错造成结果不正确。

### 5.2.2　实型数据

#### 1. 实型数据的表示

实型数据也被称为浮点型数据,实型数据的表示范围和精度是有限的。

C 语言中,实型常数有两种表示形式。

(1)十进制小数形式。它由数字和小数点组成,如 0.123、123.、123.0、0.0 等。当采用十进制小数形式表示实型数据时,小数点是必须的。

(2)指数形式。字母 e(或 E)之前必须有数字,且 e 后面的指数必须为整数。1e3、1.8e－3、－123e－6 都是合法的表示形式,而 e3、2.1e3.5、.e3、e 等的表示方法是错误的。对于指数形式的表示,在字母 e(或 E)之前的小数部分中,小数点左边应有一位(且只能有一位)非零的数字的表示方法称为“规范化的指数形式”。例如,123.456 可以表示为 123.456e0,12.3456e1,1.23456e2,0.123456e3,0.0123456e4,0.00123456e5 等,其中的 1.23456e2 称为“规范化的指数形式”。当按指数形式输出实型数据时,是按规范化的指数形式输出的。

#### 2. 实型数据在内存中的存放形式

一个实型数据一般在内存中占 4 个字节(32 位)。与整型数据的存储方式不同,实型数

据是按照指数形式存储的。系统把一个实型数据分成小数部分和指数部分，分别存放。实型数据在内存中存放格式如下：

| 数据符号 | 小数部分 | 指数部分 |
|---|---|---|

数据符号部分占一位；小数部分和指数部分所占的位数，没有具体的标准，由各 C 语言编译系统自定。一般的 C 编译系统用 4 个字节中的前 24 位表示小数部分，其中最高位为整个数的符号位，用后 8 位表示指数部分，其中最高位为指数的符号位。这种表示方法中，小数部分占的位数愈多，表示数值的有效数字愈多，精度愈高。指数部分占的位数愈多，则能表示的数值范围愈大。

**3. 实型数据的分类**

C 语言提供了 3 种用于表示实数的类型：单精度型(float)、双精度型(double)和长双精度型(long double)。表 5-2 列出了实型数据的长度和表示范围。表中的有效位是指数据在计算机中存储和输出时能够精确表示的数字位数。

**表 5-2 实型数据基本类型表**

| 类 型 | 位 数 | 表示范围 | 有效数字 |
|---|---|---|---|
| float | 32 | $10^{-37}$～$10^{38}$ | 6～7 位 |
| double | 64 | $10^{-307}$～$10^{308}$ | 15～16 位 |
| long double | 128 | $10^{-4\,931}$～$10^{4\,932}$ | 18～19 位 |

实型数据的变量定义的格式和书写规则与整型相同，例如：

- `float x、y;` /* 声明 x,y 为单精度实型量 */
- `double a,b,c;` /* 声明 a、b、c 为双精度实型量 */

**4. 实型数据的舍入误差**

由于实型数据是由有限的存储单元组成的，所以能提供的有效数字总是有限的。一个单精度实型数据只能保证的有效数字是 7 位，后面的数字是无意义的，并不能准确地表示该数。在实际运算中，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”小的数，出现舍入误差。

### 5.2.3 字符型数据

在 C 语言中，字符型数据包括字符和字符串两种，例如，'a'是字符，而"Windows"是字符串。

**1. 字符型数据在内存中的存储形式**

字符型数据在计算机中存储的是字符的 ASCII 码(见附录 A)，一个字符的存储占用一个字节。ASCII 码形式上就是 0～255 之间的整数，因此 C 语言中字符型数据和整型数据可以通用。例如，字符'A'的 ASCII 码值表示成二进制形式是 1000001，用二进制数表示时占用一个字节，在计算机中的存储形式如下：

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

字符'A'的 ASCII 码用十进制数表示是：整型数 65，在计算机中的存储形式如下：

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

可见，字符'A'的存储形式实际上就是一个整型数 65，所以它可以直接与整型数据进行算术运算、混合运算，可以与整型变量相互赋值，也可以将字符型数据以字符或整数两种形式输出。以字符形式输出时，先将 ASCII 码值转换为相应的字符，然后再输出；以整数形式输出时，直接将 ASCII 码值作为整数输出。

对于由多个字符组成的字符串，其存储形式和单个字符不同。字符串常量占的内存字节数等于字符串中字节数加 1。增加的一个字节中存放字符'\0'(ASCII 码值为 0)。这是字符串结束的标志。如字符串"Windows XP"在内存中的存储形式如下：

| W | i | n | d | o | w | s |  | X | P | \0 |
|---|---|---|---|---|---|---|---|---|---|---|

需要注意的是，字符常量和由单个字符组成的字符串常量虽然都只有一个字符，但在内存中的情况是不同的。

字符串和字符之间主要有以下区别：

(1)字符由单引号引起来，字符串由双引号引起来。

(2)字符只能是单个字符，字符串则可以含一个或多个字符。

(3)可以把一个字符赋予一个字符变量，但不能把一个字符串赋予一个字符变量。在 C 语言中没有相应的字符串变量。可以用一个字符数组来存放一个字符串变量。

(4)字符占一个字节的内存空间，字符串常量占的内存字节数等于字符串中字节数加 1。

### 2. 字符型数据的定义

字符型数据的类型说明符是 char。字符型数据的类型定义的格式和书写规则都与整型变量相同。例如：

```
char a,b; /* 声明 a、b 为字符类型变量 */
```

### 3. 转义字符

转义字符是一种特殊的字符常量。转义字符以反斜线“\”开头，后跟一个或几个字符。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，在前面的例题中 printf 函数的格式串中用到的“\n”就是一个转义字符，其意义是“回车换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。常用的转义字符及其含义见表 5-3。

**表 5-3　常用的转义字符及其含义**

| 转义字符 | 含　义 | ASCII 码(十六/十进制) |
|---|---|---|
| \0 | 空字符(NULL) | 00H/0 |
| \n | 换行符(LF) | 0AH/10 |
| \r | 回车符(CR) | 0DH/13 |
| \t | 水平制表符(HT) | 09H/9 |
| \b | 退格符(BS) | 08H/8 |
| \f | 换页符(FF) | 0CH/12 |
| \′ | 单引号 | 27H/39 |
| \" | 双引号 | 22H/34 |
| \\ | 反斜杠 | 5CH/92 |
| \ddd | 任意字符 | 1～3 位八进制 |
| \xhh | 任意字符 | 1～2 位十六进制 |

在 C 程序中使用转义字符\ddd 可以方便灵活地表示 1～3 位八进制数所代表的字符，\xhh 可以方便灵活地表示 1～2 位十六进制数所代表的字符。\ddd 表示斜杠后面跟 1～3 位八进制数，该八进制数的值即为对应的八进制 ASCII 码值。\xhh 后面跟 1～2 位十六进制数，该十六进制数为对应字符的十六进制 ASCII 码值。

使用转义字符时需要注意以下问题：

(1)转义字符中只能使用小写字母，每个转义字符只能看做一个字符。

(2)\f 换页符对屏幕没有任何影响，但会影响打印机执行响应操作。

(3)在 C 程序中，使用不可打印字符时，通常用转义字符表示。

**4.字符型数据应用举例**

**【例 5-4】** 给字符变量赋值。

```
#include<stdio.h>
void main()
{
  char a,b,c;
  a=120;
  b=121;
  c='a';
  printf("%c,%c,%c\n",a,b,c);
  printf("%d,%d,%d\n",a,b,c);
}
```

程序运行结果为：

```
x,y,a
120,121,97
```

本程序中定义 a、b、c 为字符型，但在赋值语句中可以赋以整型值或字符型值。从结果看，a、b、c 值的输出形式取决于 printf 函数格式串中的格式符：当格式符为%c 时，对应输出的变量值为字符；当格式符为%d 时，对应输出的变量值为整数。从程序运行结果可以看到，整型数据和字符型数据是通用的，需要注意的是字符型数据只占用一个字节。

**【例 5-5】** 字符变量与数值的关系。

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

程序运行结果为：

```
A,B
```

65,66

本程序中,a、b 被说明为字符变量并赋予字符值。C 语言允许字符变量参与数值运算,即用字符的 ASCII 码参与运算。由于大小写字母的 ASCII 码相差 32,因此运算后把小写字母换成大写字母,然后分别以整型和字符型输出。

【例 5-6】　转义字符的使用。

```
#include <stdio.h>
void main()
{
  printf("  ab  c\tde\rf\n");
  printf("hijk\tL\bM\n");
}
```

程序运行结果为:

```
f ab  c de
hijk    M
```

请读者运行该程序并思考输出结果的格式,以加深对转义字符的理解。

## 5.3　变量赋初值

在程序中常常需要对变量赋初值,以便使用变量。C 语言程序中有多种方法为变量赋初值。本节介绍在变量定义的同时给变量赋初值的方法,这种方法称为初始化。

在变量定义中赋初值的一般形式为:

类型说明符　变量 1=值 1,变量 2=值 2,…;

例如,下面都是合法的变量初始化语句。

- int a=5;
- int b=6,c=7;
- float x=3.14,y=3.6;
- char c1='a',c2='b';

关于变量赋初值的几点说明:

(1)C 语言允许在定义变量的同时使变量初始化。例如:

- int a=3; /* 声明 a 为整型变量,初值为 3 */
- float f=3.56; /* 声明 f 为浮点型变量,初值为 3.56 */
- char c='a'; /* 声明 c 为字符变量,初值为'a' */

(2)可以对被定义变量的一部分初始化。例如:

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

表示声明 a、b、c 为整型变量,但只对 c 初始化,c 的初值为 5。

(3)如果对几个变量赋以同一个初值,如将 a、b 和 c 赋初值为 3,应写为:

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

不能写为:

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

但这样写是正确的：

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

(4)变量初始化的一种更一般的形式是先声明变量类型，然后通过赋值语句赋值。例如：

```
int a;
a=3;
```

## 5.4 运算符与表达式

运算符是C语言用于描述数据运算的特殊符号。C语言具有丰富的运算符，这些运算符和基本数据对象结合可以构成各种表达式，这是其他任何程序设计语言所不可比拟的。其中有些运算符已超出了一般“运算符”的概念，这使得编写程序具有更大的方便性和灵活性，使程序简洁而高效；但同时，由于运算符的丰富也会产生不便于记忆、应用难度较高等问题。初学者一定要注意运算符、表达式和运算过程的使用规则，这些规则是编程的基本条件。本节主要介绍C语言的各种运算符和它们的形式和意义，以及如何用这些运算符构造表达式。

C语言的运算符按其在表达式中与运算对象的关系(连接运算对象的个数)可以分为3类：

(1)单目运算符：一个运算符连接一个运算对象。

(2)双目运算符：一个运算符连接两个运算对象。

(3)三目运算符：一个运算符连接三个运算对象。

### 5.4.1 算术运算符与表达式

#### 1.基本的算术运算符

C语言的算术运算符包括以下几种：

(1)加法运算符“+”：双目运算符或单目运算符(为正值运算符时)，如3+5、+8。

(2)减法运算符“-”：双目运算符或单目运算符(为负值运算符时)，如8-5、-3。

(3)乘法运算符“*”：双目运算符，具有左结合性，如3*5。

(4)除法运算符“/”：双目运算符。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(两数相除后的余数)“%”：双目运算，具有左结合性。要求参与运算的量均为整型。

**【例5-7】** 输出表达式的值。

```
#include<stdio.h>
void main()
{
  printf("%d,%d\n",20/8,-20/8);
  printf("%f,%f\n",20.0/8,-20.0/8);
```

```
    printf("%d\n",100%3);
}
```

程序运行结果为：

```
2,-2
2.500000,-2.500000
1
```

**2. 算术表达式**

C 语言的算术表达式由算术运算符、常量、变量、函数和圆括号组成，其基本形式与数学上的算术表达式类似。例如，3+5、12.34-23.65*2-5*(18%4+6)、x/(67-(12+y)*a)等都是合法的算术表达式。

使用算术表达式时应注意以下几点：

(1)双目运算符两侧运算对象的类型必须一致，所得结果的类型与运算对象的类型一致。如果类型不一致，系统将自动按转换规律先对操作对象进行类型转换，然后再进行相应的运算。

(2)算术表达式中的运算符是有优先级高低之分的，并遵循数学上“先乘除，后加减”的原则。“*、/、%”优先级相同，“+、-”优先级相同，“*、/、%”优先级高于“+、-”。

(3)用括号可以改变表达式的运算顺序，左右括号必须配对，多层括号都用圆括号“()”表示，运算时先计算内括号中表达式的值，再计算外括号中表达式的值。

**注意：**算术表达式中的运算对象可为常量、变量、函数调用等。其中，函数调用是指既可以调用系统定义的各类函数库中的函数，也可以调用自己编写的函数。

以数学函数的调用为例，C 语言把数学计算中常用的计算公式(或算法)抽象定义为一个个的函数，这些函数的集合构成了 C 语言的数学函数库，这样在程序中用到相应的函数时只要直接调用即可。例如，要计算 sin(x)+cos(y/2)，通过调用 C 语言数学函数库中的 sin 和 cos 函数，可直接写出算术表达式如下：

```
sin(x)+cos(y/2)
```

另外，C 语言中不含乘方运算符，对于乘方运算也要调用系统提供的函数库中的数学函数。

例如，将数学表达式 $\frac{a+b+c}{\sqrt{a}+b(\sin x+\sin y+\sin z)}$ 写成符合 C 语言规则的表达式。其 C 语言表达式如下：

```
(a+b+c)/(sqrt(a)+b*(sin(x)+sin(y)+sin(z)))
```

其中，sqrt(a)和 sin(x)、sin(y)、sin(z)都是数学库函数的调用，表达式中用了 3 层括号，以保证表达式运算顺序的正确性。

**3. 强制类型转换运算符**

可以利用强制类型转换运算符，将表达式转化为所需类型。强制类型转换的一般形式为：

```
(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 转换为实型*/
```

· (int)(x+y) /* 把 x+y 的结果转换为整型 */

自动转换发生在不同数据类型的量进行混合运算时,它由编译系统自动完成,自动转换遵循以下规则:

(1)若参与运算的量类型不同,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 型运算时,先把 int 型转换成 long 型后再进行运算。

(3)所有的浮点运算都是以双精度进行的,即使仅含 float 单精度量运算的表达式,也要先转换成 double 型,再进行运算。

(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,赋值号两边量的数据类型不同时,赋值号右边量的类型将转换为左边量的类型。如果左右两边的数据类型长度不一致时,按"长则截取,短则补足"的原则转换。

#### 4. 自增、自减运算符

自增"++"、自减"--"运算符是单目运算符,其作用是使变量的值增 1 或减 1,其优先级高于所有双目运算符。自增、自减运算的应用形式为:

· ++i;--i;运算符在变量前面,称为前缀形式,表示变量在使用前自动加 1 或减 1。

· i++;i--;运算符在变量后面,称为后缀形式,表示变量在使用后自动加 1 或减 1。

使用自增自减运算时应注意以下几点:

(1)++、--运算只能作用于变量,不能作用于表达式或常量。因为自增、自减运算是对变量进行加 1 或减 1 操作后再对变量赋新的值,而表达式或常量都不能进行赋值操作,所以下列语句形式都是不允许的:

```
x=(i+j)++;  5++;  (3*8)++;
```

如果有以下程序段:

```
int n=6;
printf("%d",-n++);
```

上述程序输出结果为-6。此时,n 的值为 7。

(2)++、--运算的前缀形式和后缀形式的意义不同。前缀形式是在使用变量之前先将其值增 1 或减 1;后缀形式是先使用变量原来的值,使用完后再使其值增 1 或减 1。例如,设 x=5,有:

· y=++x;等价于:先计算 x=x+1(结果 x=6),再执行 y=x,结果 y=6。

· y=x++;等价于:先执行 y=x,再计算 x=x+1,结果 y=5,x=6。

(3)用于++、--运算的变量只能是整型、字符型和指针型变量。

(4)++、--的结合性是自右向左的。

### 5.4.2 关系运算符与表达式

#### 1. 关系运算符

关系运算实际上是比较运算,关系运算符的作用是确定两个数据之间是否存在某种关系。C 语言有 6 种关系运算符,其有关的说明见表 5-4。

表 5-4　算术运算符

| 运 算 符 | 含　义 | 运算符类型 | 结合方向 | 简　例 |
|---|---|---|---|---|
| > | 大于 | 双目运算符 | 自左至右 | a>b、8>3 |
| >= | 大于等于 | 双目运算符 | 自左至右 | a>=b、3>=2 |
| < | 小于 | 双目运算符 | 自左至右 | a<b、3<8 |
| <= | 小于等于 | 双目运算符 | 自左至右 | a<=b、3<=b |
| != | 不等于 | 双目运算符 | 自左至右 | a!=b、3!=5%7 |
| == | 等于 | 双目运算符 | 自左至右 | a==b、3==5*a |

关系运算符的前 4 种运算符(>、>=、<、<=)优先级相同,后两种相同,前 4 种优先级高于后两种;同时,关系运算符优先级低于算术运算符,高于赋值运算符。

**2.关系表达式**

用关系运算符将两个表达式连接起来的式子称为关系表达式。它的一般形式为:

表达式 1 关系运算符 表达式 2

其中,关系运算符指明了对表达式所实施的操作。“表达式 1”和“表达式 2”是关系运算的对象,它们可以是算术表达式、关系表达式、逻辑表达式、赋值表达式和字符表达式。但一般关系运算要求关系运算符连接的两个运算对象为同类型数据。例如:

- a+b>3*c;　　/*两个算术表达式的值作比较*/
- (a=b)<(b=10%c);　　/*两个赋值表达式的值作比较*/
- (a<=b)==(b>c);　　/*两个关系表达式的值作比较*/
- 'A'!='a';　　/*两个字符表达式的值作比较*/

关系表达式只有两种可能的结果,即它所描述的关系成立或者不成立,所以一个关系表达式描述的是一种逻辑判断。若关系成立,说明关系表达式表述的关系是“真”的,称逻辑值为“真”,用 1 表示;若关系不成立,说明关系表达式表述的关系是“假”的,称逻辑值为“假”,用 0 表示。所以关系表达式的运算结果一定是逻辑值(用 1 或 0 表示)。

在进行关系运算时,应先计算表达式的值,然后再进行关系比较运算。例如,假定 a=2、b=3、c=4,则上述关系表达式的值为:

- a+b>3*c、(5>12)关系不成立,表达式结果值为 0(假)。
- (a+=b)<(b*=10%c)、(5<6)关系成立,表达式结果值为 1(真)。
- (a<=b)==(b>c)、(1==0)关系不成立,表达式结果值为 0(假)。
- 'A'!='a'、(65!=97)关系成立,表达式结果值为 1(真)。

以关系表达式 a+b>3*c 为例,因为算术运算符的优先级高于关系运算符,所以先计算 a+b 和 3*c 的值,结果分别为 5 和 12,再将 5 和 12 进行关系比较,其运算结果为 0。

在表达式中连续使用关系运算符时,要注意正确表达含义,注意运算优先级和结合性。例如,变量 x 的取值范围为“0≤x≤20”时,不能写成“0<=x<=20”。因为关系表达式“0<=x<=20”的运算过程是:按照优先级,先求出“0<=x”的结果,再将结果 1 或 0 进行“<=20”的判断,这样无论 x 取何值,最后表达式一定成立,结果一定为 1。这显然违背了原来的含义。此时,就要运用将要介绍的逻辑运算符进行连接,即应写为:x>=0&&x<=20。

### 5.4.3 逻辑运算符与表达式

#### 1.逻辑运算符

C 语言规定的 3 种逻辑运算符及其有关的说明见表 5-5。

表 5-5 逻辑运算符

| 运 算 符 | 含 义 | 运算符类型 | 结合方向 | 简 例 |
|---|---|---|---|---|
| && | 逻辑与 | 双目运算符 | 自左向右 | a&&b、3>8&&a==b |
| \|\| | 逻辑或 | 双目运算符 | 自左向右 | a\|\|b、3<=8\|\|a==b |
| ! | 逻辑非 | 单目运算符 | 自右向左 | !a、!a==b |

逻辑运算要求运算对象为“真”(1)或“假”(0)。这 3 种逻辑运算符的运算规则可用表 5-6 的真值表表示。

表 5-6 逻辑运算真值表

| a | b | a&&b | a\|\|b | !a | !b |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

#### 2.逻辑表达式

用逻辑运算符将关系表达式或逻辑量连接起来的式子就是逻辑表达式。在一个逻辑表达式中,可以含有多个逻辑运算符,其优先级是:“!”最高,“&&”次之,“||”最低;逻辑运算符中的“&&”和“||”低于关系运算符,“!”高于算术运算符。

例如,某程序中有如下说明:“int a=3,b=1,x=2,y=0;”则:

(1)(a>b)&&(x>y)的值为 1。

(2)a>b&&x>y 的值为 1。

(3)(y||b)&&(y||a)的值为 1。

(4)y||b&&y||a 的值为 1。

(5)!a||a>b 的值为 1。

其中,(1)(2)两式是等价的,因为“&&”运算优先级低于关系运算,故括号可以省略。(3)(4)两式结果虽然一样,但两式的含义不同。(3)式中由于括号的优先级高于“&&”,所以先计算“y||b”和“y||a”,再将两个结果进行“&&”运算。而(4)式由于“&&”的优先级高于“||”,故要先计算“b&&y”,其结果为 0,再计算“y||0”,其值也为 0,最后计算“0||a”,结果为 1。由此可见,运算符的优先级制约着表达式的计算次序。(5)式中,“!”的优先级高于“>”,而“>”的优先级高于“||”,故先计算“!a”,其值为 0,再计算“a>b”,其值为 1,最后计算“0||1”,值为 1。

### 5.4.4 赋值运算符与表达式

赋值运算符构成了 C 语言最基本、最常用的赋值语句,同时 C 语言还允许赋值运算符

“=”与 10 种运算符联合使用，形成复合赋值运算，使得 C 程序简明而精练。

### 1. 赋值运算符

赋值运算符用“=”表示，其功能是计算赋值运算符“=”右边表达式的值，并将计算结果赋给“=”左边的变量。例如：

- a=12.3; /* 直接将实型数 12.3 赋给变量 a */
- c=a*b; /* 将 a 和 b 进行乘法运算，所得到的结果赋给变量 c */

赋值运算符“=”与数学中的等号完全不同，数学中的等号表示该等号两边的值是相等的，而赋值运算符“=”是指要完成“=”右边表达式的运算，并将运算结果存放到“=”左边指定的内存变量中。可见，赋值运算符完成两类操作：一是计算，二是赋值。

### 2. 赋值表达式

由赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式。它的一般形式为：

变量名=表达式

对赋值表达式的求解过程：计算赋值运算符右边“表达式”的值，并将计算结果赋值给赋值运算符左边的“变量”。赋值表达式的值就是赋值运算符左边“变量”的值。例如，算术表达式(-b+sqrt(b*b-4*a*c))/(2*a)写成赋值表达式为：

x1=(-b+sqrt(b*b-4*a*c))/(2*a)

其中，x1 是变量，赋值号右边是算术表达式，x1 的值就是这个算术表达式的值，也就是该赋值表达式的值。以下均是赋值表达式：

- i=5　/* 将常数 5 赋值给变量 i，赋值表达式“i=5”的值就是 5 */
- a=3.5-b　/* 计算算术表达式 3.5-b 的值并赋值给变量 a */
- x=(a+b+c)/12.4*8.5　/* 计算算术表达式(a+b+c)/12.4*8.5 的值并赋值给变量 x */

### 3. 类型转换

在对赋值表达式求解的过程中，如果赋值运算符两边的数据类型不一致，赋值时要进行类型转换。其转换工作由 C 编译程序自动实现，转换原则是以“=”左边的变量类型为准，即将“=”右边的值的类型转换为与“=”左边的变量类型一致的类型。

**【例 5-8】** 赋值运算及数据类型转换。

```
#include<stdio.h>
void main()
{
  int i=65;     /* 声明整型变量 i 并初始化为 65 */
  float a=66.5,a1;   /* 声明实型变量 a 和 a1 并初始化 a */
  double b=123456789.123456789;   /* 声明双精度型变量 b 并初始化 */
  char c='A';                     /* 声明字符变量 c 并初始化为'A'*/
  printf("i=%d,a=%f,b=%f,c=%c\n",i,a,b,c);     /* 输出 i、a、b、c 的初
                                                  始值 */
  a1=i; i=a; a=b; c=i;
  printf ("i=%d,a=%f,a1=%f,c=%c\n",i,a,a1,c);   /* 输出 i、a、a1、c 赋
```

```
                                                    值以后的值 * /
}
```

程序运行结果为：

```
i=65,a=66.500000,b=123456789.123457,c=A
i=66,a=123456792.000000,a1=65.000000,c=B
```

由以上运行结果可见：

(1)将 float 型数据赋值给 int 型变量时，先将 float 型数据舍去其小数部分，然后再赋值给 int 型变量。例如，“i=a;”的结果是 int 型变量 i 只取实型数据 66.5 的整数部分 66。

(2)int 型数据赋给 float 型变量时，先将 int 型数据转换为 float 型数据，并以浮点数的形式存储到变量中，其值不变。例如，“a1=i;”的结果是整型数据 65 先转换为 65.000 000，再赋值给实型变量 a1。如果赋值的是双精度实数，则按其规则取有效数位。

(3)double 型实数赋给 float 型变量时，先截取 double 型实数的前 7 位有效数字，然后再赋值给 float 型变量。例如，“a=b;”的结果是截取 double 型实数 123 456 789.123 457 的前 7 位有效数字 1 234 567 赋值给 float 型变量。上述输出结果中，a=123 456 792.000 000 的第 8 位以后就是不可信的数据了。所以一般不使用这种把有效数字多的数据赋值给有效数字少的变量。

(4)int 型数据赋值给 char 型变量时，由于 int 型数据用两个字节表示，而 char 型数据只用一个字节表示，所以先截取 int 型数据的低 8 位，然后赋值给 char 型变量。例如，上述程序中，执行“i=a;”后 int 型变量 i 的结果是 66，而“c=i;”的结果是：截取 i 的低 8 位(二进制数 00000011)赋值给 char 型变量，将其 ASCII 码对应的字符输出即字符 B。

**4.复合赋值运算符和复合赋值表达式**

1)复合赋值运算符

C 语言规定，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，其中可与“=”形成复合赋值运算的运算符有：+、-、*、/、%、<、>、|、&、^。

所构成的复合赋值运算有：+=、-=、*=、/=、%=、<=、>=、|=、&=、^=。

2)复合赋值表达式

由复合赋值运算符将一个变量和一个表达式连接起来的式子称为复合赋值表达式。它的一般形式为：

变量名 复合赋值运算符 表达式

其功能是对“变量名”和“表达式”进行复合赋值运算符所规定的运算，并将运算结果赋值给复合赋值运算符左边的“变量名”。复合赋值运算的作用等价于：

变量名=变量名 运算符 表达式

即先将变量和表达式进行指定的组合运算，然后将运算的结果值赋给变量。例如：

- a*=3 等价于 a=a*3。
- a*=b+5 等价于 a=a*(b+5)。
- a--=1 等价于 a=a-1。

**注意**：a*=b+5 与 a=a*b+5 是不等价的，它实际上等价于 a*(b+5)，这里的括号是必需的。

C 语言提供了赋值表达式，它使赋值操作不仅可以出现在赋值语句中，而且可以以表达式的形式出现在其他语句中。例如：

```
printf("i= %d,s= %f\n",i=3 * 45,s-=3.14 * 12.5 * 12.5);
```

该语句直接输出赋值表达式 i=3 * 45 和 s-=3.14 * 12.5 * 12.5 的值，也就是输出变量 i 和 s 的值，在一个语句中完成了赋值和输出的双重功能。这就是 C 语言使用的灵活性。

**【例 5-9】** 复合赋值运算符的使用。

```
#include<stdio.h>
void main()
{
  int a=3,b=2,c=4,d=8,x;
  a+=b * c;
  b-=c/b;
  printf(" %d, %d, %d, %d\n",a,b,c * =2 * (a-c),d %=a);
  printf("x= %d\n",x=a+b+c+d);
}
```

程序运行结果为：

```
11,0,56,8
x=75
```

### 5.4.5　逗号运算符与表达式

逗号运算使用的运算符是“,”，其作用是将多个表达式连在一起构成逗号表达式。其形式为：

表达式 1,表达式 2,…,表达式 n

逗号表达式的优先级是所有表达式中最低的，其结合性是左结合性。对逗号表达式的求解过程是将逗号表达式中各表达式按从左至右的顺序依次求值，并将最右面的表达式结果作为整个逗号表达式的最后结果。例如：

```
y=(x=123,x++,x+=100-x);
```

括号内表达式是用“,”运算符连接的 3 个表达式，执行情况是将 123 赋给 x，然后执行 x++得 x 的值为 124，最后执行 x+=100-x 得 100，这个 100 就是该逗号表达式的求解结果，所以 y 的值是 100。

### 5.4.6　条件运算符与表达式

C 语言还提供唯一的三目运算符——条件运算符，条件运算符构成的表达式称为条件表达式。其形式为：

表达式 1? 表达式 2 : 表达式 3

条件运算符的“?”和“:”是成对出现的。条件表达式的运算功能是：先计算表达式 1 的值，若为非 0 值，则计算表达式 2 的值，并将表达式 2 的值作为整个条件表达式的结果；否则，计算表达式 3 的值，并将表达式 3 的值作为整个条件表达式的结果。例如：

```
(a>b)? a+b : a-b
```

当 a=8、b=4 时，求解条件表达式的过程如下：

先计算关系式 a>b，结果为 1，因其值为真，则计算 a+b 的结果为 12，这个 12 就是整个

条件表达式的结果。

条件运算符的优先级高于赋值运算符，但低于所有关系运算符、逻辑运算符和算术运算符，其结合性是自右向左结合。使用条件表达式可以使程序简洁明了，它常用在 if 语句中，不管条件是否成立，都要给同一个变量赋值的情况。例如：

```
if(a>b) max=a; else max=b;
```

可以简洁地表示为：

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

表示求变量 a 与 b 的较大值并赋给变量 max。

以上介绍了 C 语言的数据类型及一些运算符与表达式，需要说明的是，不同数据类型之间进行混合运算时要先进行数据类型转换。C 语言允许进行整型、实型、字符型数据的混合运算，但在实际运算时，要先将不同类型的数据转换成同一类型再进行运算。这种类型转换的一般规则是：

- 运算中将所有 char 型数据都转换成 int 型，float 型数据转换成 double 型。
- 低级类型服从高级类型，并进行相应的转换。数据类型的级别由低到高的排序表示如下：

  char→int→unsigned→long→float→double
- 赋值运算中最终结果的类型，以赋值运算符左边变量的类型为准，即赋值运算符右边值的类型以左边变量的类型为准，并进行相应的转换。

设有如下变量说明：

```
int a; float b; long d; double c; int e,f;
```

对于赋值语句：

```
f=e+'a'+a*b-c/d;
```

其运算次序和隐含的类型转换如下：

(1)计算 a*b，由于变量 b 为 float 型，所以运算时先由系统自动转换为 double 型，变量 a 为 int 型，两个运算对象要保持类型一致，变量 a 也要转换为 double 型，运算结果为 double 型。

(2)由于 c 为 double 型，将 d 转换成 double 型，再计算 c/d，结果为 double 型。

(3)计算 e+'a'，先将'a'(char 型)转换成 int 型数再与 e 相加，结果为 int 型。

(4)将第(1)步和第(3)步的结果相加，先将第(3)步的结果(int 型)转换成 double 型再进行运算，结果为 double 型。

(5)用第(4)步的结果减第(2)步的结果，结果为 double 型。

(6)给 f 赋值，先将第(5)步的结果 double 型转换为 int 型(因为赋值运算左边变量 f 为 int 型)，即将 double 型数据的小数部分截掉，转换成 int 型，然后进行赋值。其中，类型转换是由 C 语言编译系统自动完成的。

## 本章小结

本章主要介绍了 C 语言中有关数据类型、运算符与表达式的基本概念和规则。关于数据类型的主要内容有：常量与变量、各种类型数据的表示方法、数据的取值范围和数值的有

效位。关于运算符与表达式的主要内容有:运算符与运算对象、表达式及其表示、运算符优先级及结合性、算术运算(包括自加、自减运算)、关系运算、逻辑运算、条件运算、赋值运算、复合赋值运算、混合运算及赋值过程中的类型转换等。本章是 C 语言程序设计的基础部分,需要熟练掌握。

## 习　题　5

1. C 语言有哪些数据类型及运算符?
2. C 语言规定对所有用到的变量要“先定义后使用”,这样做有什么好处?
3. C 语言的表达式类型有哪些?
4. 请将下面各数用八进制和十六进制数(补码)表示:

(1)12　(2)－1　(3)65　(4)－10

5. 字符常量与字符串常量有什么区别?
6. 写出以下程序运行的结果。

(1)

```
#include <stdio.h>
void main()
{
  char c1=97,c2=98;
  int a=97,b=98;
  printf("%3c,%3c\n",c1,c2);
  printf("%d,%d\n",c1,c2);
  printf("\n%c,%c\n",a,b);
}
```

(2)

```
#include<stdio.h>
void main()
{
  int a=-1;
  printf("%d,%o,%x,%u\n",a,a,a,a);
}
```

(3)

```
#include<stdio.h>
void main()
{
  int i,j;
  i=3; j=4;
  printf("%d,%d\n",i++,++j);
  printf("%d,%d\n",i,j);
```

```
  printf("%d,%d\n",-i++,--++j);
}
```

(4)

```
#include<stdio.h>
void main()
{
  int a,b;a=5;
  a-=a*a;
  printf("a=%d\n",a);
  b=(a=3*5,a*4,a+5);
  printf("a=%d,b=%d\n",a,b);
}
```

7.求下面表达式的值。

(1)x+a%3*(int)(x+y)%2/4。设 x=2.5,a=7,y=4.7。

(2)(float)(a+b)/2+(int)x%(int)y。设 a=2,b=3,x=3.5,y=2.5。

(3)a+b>c&&b==c。设 a=3,b=4,c=5。

(4)a||b+c&&b-c。设 a=3,b=4,c=5。

(5)!(a>b)&&!c||1。设 a=3,b=4,c=5。

(6)!(x=a)&&(y=b)&&0。设 a=3,b=4。

(7)!(a+b)+c-1&&b+c/2。设 a=3,b=4,c=5。

8.a 和 n 已定义为整型变量,设 a=12,n=5。写出下面表达式运算后 a 的值。

(1)a+=a

(2)a=2

(3)a*=2+3

(4)a/=a+a

(5)a%=(n%=2)

(6)a+=a-=a*=a