第3章 chapter 3

运算符和表达式

本章导读：C语言本身没有提供输入输出语句，它的输入输出功能由函数来实现，本章首先介绍几种常用的输入输出函数。C语言在解决问题中不仅要考虑需要哪些数据，还要考虑对数据的操作，以达到求解问题的目的。这就需要依靠具有一定运算功能的运算符将运算对象连接起来，并且以符合C的语法规则构成一个指定运算过程的式子即表达式来完成数据的处理，表达式中的运算对象包括常量、变量和函数等。本章重点介绍C语言中的运算符和表达式。

学习目标：

- 掌握各输入输出函数；
- 了解C语言运算符与表达式的概念；
- 掌握基本运算符的功能；
- 掌握各类运算符的优先级和结合性；
- 理解表达式的概念。

3.1 数据的输入与输出

在程序运行过程中，作为程序加工处理的对象——数据，有时候需要从外部设备（例如键盘）上得到一些原始数据。程序计算结束后，通常要把计算结果发送到外部设备（例如显示器）上，以便人们对结果进行分析。把程序从外部设备上获得数据的操作称为“输入”，而把程序发送数据到外部设备的操作称为“输出”。不像其他的高级语言，C语言没有专门的输入、输出语句，其操作是通过调用C语言的库函数来实现的。

scanf()函数和printf()函数是格式输入输出函数，它们能够一次性输入输出一个或多个数据。getchar()函数和putchar()函数是字符输入输出函数，它们每次只能输入输出单个字符。

scanf()、printf()和getchar()、putchar()等函数的声明包含在stdio.h头文件中，所以在使用这些函数的源文件开头要加上预编译命令#include<stdio.h>或#include "stdio.h"，stdio是standard input &outupt的缩写。

考虑到输入输出函数使用频繁，系统允许在使用这两个函数时可不加以上预编译

命令。

3.1.1 格式输出函数

printf 函数称为格式输出函数，其关键字最末一个字母 f 即为“格式”(format)之意，它的作用是向计算机系统默认的输出设备(一般指显示器)输出一个或多个任意指定类型的数据。

在前面的例题中我们已多次使用过这个函数。下面我们详细讨论。

1. printf 函数调用的一般形式

printf 函数调用的一般形式为：

```
printf(格式控制字符串,输出表列)
```

例如：

```
printf("f=%d,z=%d ",f,z);
```

其中“格式控制字符串”是用双引号括起来的字符串，用于指定输出格式。包括格式说明符、转义字符和普通字符三种形式。

格式说明符：由“%”和格式字符组成，例如“%6d”、“%8.2f”和“%c”等，这些字符用来控制数据的输出格式。

(1) 转义字符：这些字符通常用来控制光标的位置，例如“\n”、“\t”等。

(2) 普通字符：除格式说明符和转义字符以外的其他字符，这些字符原样输出，例如上例中的“f=”、“,”、“z=”。

(3)“输出表列”由若干个输出项构成，输出项之间用“,”隔开，每个输出项既可以是常量、变量、表达式，也可以没有输出项。没有输出项时，一般用来输出一些提示信息。

例如：

```
printf("请输入两个整数,以逗号隔开。\n");
```

printf()函数可以输出常量、变量和表达式的值。但格式控制字符串中的格式指示符必须按从左到右的顺序，与输出表列中的每个数据一一对应，否则出错。

例如，有如下语句：

```
int a=4,b=5;
printf("a=%d,b=%d\n",a,b);
```

其输出过程如图 3-1 所示，在格式控制字符串中第一个格式控制符“%d”与输出表列中的“a”变量对应，输出时第一个“%d”位置处将被“a”的值替换；第二个格式控制符“%d”与输出表列中的“b”变量对应，输出时第一个“%d”位置处将被“b”的值替换。

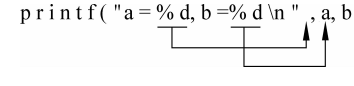

图 3-1 输出时格式控制符与变量的对应关系

2. 格式说明符

格式说明符用于指定对应输出项的输出格式，其一般形式为：

[标志][输出最小宽度][.精度][长度]类型

其中方括号[]中的项为可选项。

各项的意义介绍如下。

(1) 类型：类型字符用以表示输出数据的类型，其格式符和意义如表 3-1 所示。

表 3-1 常用的格式字符及说明

格式字符	意义
d	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 0)
x 或 X	以十六进制形式输出无符号整数(不输出前缀 0x)
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e 或 E	以指数形式输出单、双精度实数
g 或 G	以%f 或%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

输出示例如表 3-2 所示。

表 3-2 输出示例图

格式符	示例	运行结果
d	int a=123;printf("%d",a);	123
x(或 X)	int a=123;printf("%x",a);	7B
o	int a=123;printf("%o",a);	173
u	int a=80;printf("%u",a);	80
ld	long a=123; printf("%ld",a);	123
c	char c=69; printf("%c",c);	E
s	printf("%s","China");	China

(2) 标志：标志字符为－、＋、＃、空格 4 种，其意义如表 3-3 所示。

表 3-3 常用标志字符及说明

标 志	意 义
－	结果左对齐，右边填空格
＋	输出符号(正号或负号)
空格	输出值为正时冠以空格，为负时冠以负号
#	对 c,s,d,u 类无影响；对 o 类，在输出时加前缀 0；对 x 类，在输出时加前缀 0x；对 e,g,f 类当结果有小数时才给出小数点

(3) 输出最小宽度：用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度，则按实际位数输出；若实际位数少于定义的宽度，则补以空格或 0。

(4) 精度：精度格式符以“.”开头，后跟十进制整数。本项的意义是：如果输出数字，则表示小数的位数；如果输出的是字符，则表示输出字符的个数；若实际位数大于所定义的精度数，则截去超过的部分。

(5) 长度：长度格式符为 h,l 两种，h 表示按短整型量输出，l 表示按长整型量输出。

说明：格式字符紧跟在“%”后面就作为格式字符，否则将作为普通字符使用(原样输出)。输出示例如表 3-4 所示，“Θ”表示一个空格。

表 3-4 输出示例

输出函数调用	输 出 结 果
printf("%d",42);	42
printf("%5d",42);	ΘΘΘ42
printf("%3d",1234);	1234
printf("%－5d",42);	42ΘΘΘ
printf("%05d",42);	00042
printf("%d%d",42,54);	4254
printf("%5d%5d",42,54);	ΘΘΘ42ΘΘΘ54
printf("%f",123.54);	123.540000
printf("%12f",123.54);	ΘΘ123.540000
printf("%8.3f",123.55);	Θ123.550
printf("%8.1f",123.55);	ΘΘΘ123.6
printf("%8.0f",123.55);	ΘΘΘΘΘ124

3.1.2 格式输入函数

scanf 函数称为格式输入函数，即按用户指定的格式从键盘把数据输入到指定的变量之中。scanf()函数可以用于所有类型数据的输入，使用不同的格式转换符可以将不同类型的数据从标准输入设备读入内存。

1. scanf 函数的一般形式

scanf 函数的一般形式为：

```
scanf("格式控制字符串",地址列表);
```

其中，格式控制字符串的作用与 printf 函数相同，但不能显示非格式说明符，也就是不能显示提示字符串。

地址列表是由若干个地址给出的列表，可以是变量的地址，或字符串的首地址。地址是由地址运算符“&”后跟变量名组成的。

例如：

&a，表示变量 a 的地址

&b，表示变量 b 的地址

“&”是一个取地址运算符，&a 是一个表达式，其功能是求变量的地址。

这个地址就是编译系统在内存中给 a，b 变量分配的地址。在前面的章节中我们已经把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是 C 编译系统分配的，编程者不必关心具体的地址是多少。

例如，有以下语句：

```
int a, b;
scanf("%d%d", &a, &b);
```

程序运行时，从键盘输入数据 34，按下回车键后，scanf 函数开始读数据。将数值 3 存入到 a 变量的地址中，将数值 4 存入到 b 变量的地址中。其格式控制符与变量的对应关系如图 3-2 所示。

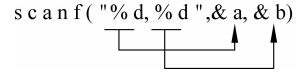

图 3-2　输入时格式控制符与变量的对应关系

2. 格式说明符

格式说明符与 printf()函数中的格式说明符相似，也是以“%”开始，以一个格式说明符结束，中间可以插入附加的字符。

格式说明符的一般形式为：

```
[*][输入数据宽度][长度]类型
```

其中有方括号[]的项为任选项。各项的意义如下。

(1) 类型：表示输入数据的类型，其格式符和意义如表 3-5 所示。

表 3-5　输入数据类型

格　式	字 符 意 义	格　式	字 符 意 义
d	输入十进制整数	x	输入十六进制整数
o	输入八进制整数	u	输入无符号十进制整数

续表

格 式	字 符 意 义	格 式	字 符 意 义
f 或 e	输入实型数(用小数形式或指数形式)	s	输入字符串
c	输入单个字符		

(2)“*”符：用以表示该输入项，读入后不赋予相应的变量，即跳过该输入值。

如：

```
scanf("%d%*d%d",&a,&b);
```

当输入为1 2 3时，把1赋给变量a，2被跳过，3赋给变量b。

(3) 宽度：用十进制整数指定输入的宽度(即字符数)。

例如：

```
scanf("%5d",&a);
```

输入：12345678

只把12345赋给变量a，其余部分被截去。

又如：

```
scanf("%4d%4d",&a,&b);
```

输入：12345678

将把1234赋予a，而把5678赋予b。

(4) 长度：长度格式符为l和h，l表示输入长整型数据(如%ld)和双精度浮点数(如%lf)。h表示输入短整型数据。

使用scanf函数还必须注意以下几点：

① scanf函数中没有精度控制，如scanf("%5.2f",&a);是非法的。不能企图用此语句输入小数为2位的实数。

② scanf中要求给出变量地址，如给出变量名则会出错。如scanf("%d",a);是非法的，应改为scnaf("%d",&a);才是合法的。

③ 在输入多个数值数据时，若格式控制串中没有非格式字符作输入数据之间的间隔则可用空格，Tab或回车作间隔。

④ 输入数据时，遇到以下情况时系统认为该数据输入结束。

- 遇到空格，或者按回车键或Tab键。
- 遇到输入域宽度结束。例如%3d，只取3列。
- 遇到非法输入。例如，在输入数值数据时，遇到字母等非数值符号。

⑤ 在输入字符数据时，若格式控制串中无非格式字符，则认为所有输入的字符均为有效字符。

例如：

```
scanf("%c%c%c",&a,&b,&c);
```

输入为：

```
d  e  f
```

则‘d’赋给变量 a，‘ ’赋给变量 b，‘e’赋给变量 c。

只有当输入为：

```
def
```

时，才能把‘d’赋给变量 a，‘e’赋给变量 b，‘f’赋给变量 c。

如果在格式控制中加入空格作为间隔，如：

```
scanf ("%c %c %c",&a,&b,&c);
```

则输入时各数据之间可加空格。

⑥ 格式字符串中出现的普通字符(包括转义字符形式的字符)务必原样输入。

例如：

```
scanf("%d,%d,%d",&a,&b,&c);
```

其中用非格式符","作间隔符，故输入时应为：

```
5,6,7
```

又如：

```
scanf("a=%d,b=%d,c=%d",&a,&b,&c);
```

则输入应为：

```
a=5,b=6,c=7
```

⑦ 如输入的数据与输出的类型不一致时，虽然编译能够通过，但结果将不正确。

3.1.3 字符输出函数

putchar() 函数是字符输出函数，其功能是在显示器上输出单个字符。

putchar() 函数的一般形式为：

```
putchar(c)
```

输出字符变量 c 的值，c 可以是字符型变量，也可以是整型变量。

例如：

```
putchar('A');          (输出大写字母 A)
putchar(x);            (输出字符变量 x 的值)
putchar('\101');       (也是输出字符 A)
putchar('\n');         (换行)
```

对控制字符则执行控制功能，不在屏幕上显示。

3.1.4 字符输入函数

getchar()函数的功能是从键盘上输入一个字符。

其一般形式为：

```
getchar();
```

通常把输入的字符赋给一个字符变量，构成赋值语句，如：

```
char c;
c=getchar();
```

【例 3-1】 输入单个字符。

```
//Exam ch03-1.c
#include<stdio.h>
void main()
{
    char c;
    printf("input a character\n");
    c=getchar();
    putchar(c);
}
```

以上程序将从键盘获取一个字符，并把它赋给变量 c，然后再输出到显示器。

3.2 运算符和表达式的概念

C 语言中运算符和表达式数量之多，在高级语言中是少见的。正是丰富的运算符和表达式使 C 语言功能十分完善。这也是 C 语言的主要特点之一。

1. 数据类型和运算符

不同的数据类型都提供了相应的运算符，以实现该类型数据的处理，也就是说，任何一个运算符都不可能脱离数据类型而单独存在。例如，short int 类型的运算符有＋、－、＊和％等。

2. 运算符的分类

运算符按照参加运算的运算数的个数来分，可以分为：

(1) 一元运算符或单目运算符：参加运算的运算数只有一个。

(2) 二元运算符或双目运算符：参加运算的运算数有两个。

以此类推。

运算符按照其功能分类如下。

(1) 算术运算符：用于各类数值运算。包括加(+)、减(-)、乘(*)、除(/)、求余(或称模运算%)、自增(++)、自减(--)共7种。

(2) 关系运算符：用于比较运算。包括大于(>)、小于(<)、等于(==)、大于等于(>=)、小于等于(<=)和不等于(!=)6种。

(3) 逻辑运算符：用于逻辑运算。包括与(&&)、或(||)、非(!)三种。

(4) 位操作运算符：参与运算的量，按二进制位进行运算。包括位与(&)、位或(|)、位非(~)、位异或(^)、左移(<<)、右移(>>)6种。

(5) 赋值运算符：用于赋值运算，分为简单赋值(=)、复合算术赋值(+=、-=、*=、/=、%=)和复合位运算赋值(&=、|=、^=、>>=、<<=)3类共11种。

(6) 条件运算符：这是一个三目运算符，用于条件求值(?:)。

(7) 逗号运算符：用于把若干表达式组合成一个表达式(,)。

(8) 指针运算符：用于取内容(*)和取地址(&)二种运算。

(9) 求字节数运算符：用于计算数据类型所占的字节数(sizeof)。

(10) 特殊运算符：有括号()，下标[]，成员(→、.)等几种。

3. 运算符的优先级和结合性

C语言中，运算符的运算优先级共分为15级。1级最高，15级最低。在表达式中，优先级较高的先于优先级较低的进行运算。而在一个运算量两侧的运算符优先级相同时，则按运算符的结合性所规定的结合方向处理。

C语言中各运算符的结合性分为两种，即左结合性(自左至右)和右结合性(自右至左)。例如，算术运算符的结合性是自左至右，即先左后右。如有表达式x-y+z则y应先与“-”号结合，执行x-y运算，然后再执行+z的运算。这种自左至右的结合方向就称为“左结合性”。而自右至左的结合方向称为“右结合性”。最典型的右结合性运算符是赋值运算符。如x=y=z，由于“=”的右结合性，应先执行y=z再执行x=(y=z)运算。C语言运算符中有不少为右结合性，应注意区别，以避免理解错误。

4. 表达式

表达式指用运算符和圆括号将常量、变量和函数调用连接起来的有意义的式子。单个常量、变量和函数调用也可看成一个表达式。例如，a=7*b和b=a>b都是一个表达式。

表达式要按照规定的运算规则进行运算，并会得到一个结果，即表达式的值。

注意：在C中，凡是表达式都有一个值，例如，a=2是一个表达式，其运算结果是a的值即2。

C语言可分为算术表达式、赋值表达式、关系表达式、逗号表达式和逻辑表达式等。

3.3 赋值运算符和赋值表达式

1. 赋值运算符

简单赋值运算符记为“=”,是一个双目运算符。由“=”连接的式子称为赋值表达式。其一般形式为:

```
变量=表达式
```

例如:

```
x=a+b
w=sin(a)+sin(b)
y=i+++--j
```

赋值表达式的功能是计算表达式的值再赋予左边的变量。

注意:优先级仅比最低级的逗号运算符高,右结合性。

因此 a=b=c=5 可理解为 a=(b=(c=5))。

在其他高级语言中,赋值构成了一个语句,称为赋值语句。而在 C 中,把“=”定义为运算符,从而组成赋值表达式。凡是表达式可以出现的地方均可出现赋值表达式。

例如,式子 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 语言规定,任何表达式在其末尾加上分号就构成语句。因此如 x=8;a=b=c=5;都是赋值语句,在前面各例中我们已大量使用过了。

2. 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。整型赋予字符型,只把低 8 位赋予字符量。

【例 3-2】

```
//Exam ch03-2.c
#include<stdio.h>
void main()
{
```

```
    int a,b=322;
    float x,y=8.88;
    char c1='k',c2;
    a=y;
    x=b;
    a=c1;
    c2=b;
    printf("%d,%f,%d,%c",a,x,a,c2);
}
```

【程序运行】

```
107,322.000000,107,B
```

本例表明了上述赋值运算中类型转换的规则。a 为整型，赋予实型量 y 值 8.88 后只取整数 8。x 为实型，赋予整型量 b 值 322，后增加了小数部分。字符型量 c1 赋予 a 变为整型，整型量 b 赋予 c2 后取其低 8 位成为字符型(b 的低 8 位为 01000010，即十进制 66，按 ASCII 码对应于字符 B)。

3. 复合的赋值运算符

在赋值符“=”之前加上其他二目运算符可构成复合赋值符，如+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=。

构成复合赋值表达式的一般形式为：

```
变量  双目运算符=表达式
```

它等效于

```
变量=变量 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
```

复合赋值符这种写法，对初学者可能不习惯，但十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

3.4 算术运算符和算术表达式

3.4.1 基本算术运算符

1. 基本算术运算符

基本算术运算符的性质如图 3-3 所示。

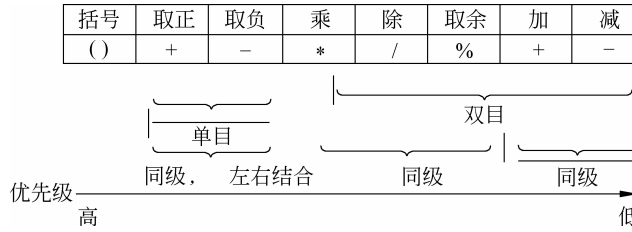

图 3-3 算术运算符

【例 3-3】

```
//Exam ch03-3.c
#include<stdio.h>
void main()
{
    printf("\n\n%d,%d\n",20/7,-20/7);
    printf("%f,%f\n",20.0/7,-20.0/7);
}
```

【程序运行】

```
2,-2
2.857143,-2.857143
```

【程序说明】

本例中，20/7，－20/7 的结果均为整型，小数全部舍去。而 20.0/7 和－20.0/7 由于有实数参与运算，因此结果也为实型。

注意：求余运算符(模运算符)“%”是双目运算，具有左结合性。要求参与运算的量均为整型。求余运算的结果等于两数相除后的余数。

2. 算术表达式

用算术运算符和括号将运算对象(也称操作数)连接起来的符合 C 语法规则的式子称为算术表达式。参加算术运算的运算数可以是整型数据、实型数据和字符型数据。

以下是算术表达式的例子：

```
a+b
(a*2)/c
(x+r)*8-(a+b)/7
++i
sin(x)+sin(y)
(++i)-(j++)+(k--)
```

3.4.2 自增、自减运算符

自增 1 运算符：记为“＋＋”，其功能是使变量的值自增 1。

自减1运算符：记为“－－”，其功能是使变量的值自减1。

自增1，自减1运算符均为单目运算，都具有右结合性。有如表3-6所示的几种形式。

表3-6　自增、自减运算符的几种形式

表　达　式	前缀运算	后缀运算
x＝x＋1 或 x＋＝1	＋＋x	x＋＋
x＝x－1 或 x－＝1	－－x	x－－

可以由自增和自减运算构成以下4种运算。

(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。

注意：自增、自减运算对象只能是变量而不能是表达式或常量。

【例3-4】

```
//Exam ch03-4.c
#include<stdio.h>
void main()
{
    int i=8;
    printf("%d\n",++i);
    printf("%d\n",--i);
    printf("%d\n",i++);
    printf("%d\n",i--);
    printf("%d\n",-i++);
    printf("%d\n",-i--);
}
```

【程序运行】

```
9
8
8
9
-8
-9
```

【程序说明】

i的初值为8，第2行i加1后输出故为9；第3行减1后输出故为8；第4行输出i为8之后再加1(为9)；第5行输出i为9之后再减1(为8)；第6行输出－8之后再加1(为9)，第7行输出－9之后再减1(为8)。

3.5 关系运算符和关系表达式

在程序中经常需要比较两个量的大小关系，以决定程序下一步的工作。比较两个量的运算符称为关系运算符。

1. 关系运算符

在C语言中有如表3-7所示的关系运算符。

表3-7 关系运算符

运算符	名称	示例	运算功能	优先级	结合性
<	小于	x<0	判断x是否小于0	相同(高)	左结合性
<=	小于等于	x<=0	判断x是否小于等于0		
>	大于	x>0	判断x是否大于0		
>=	大于等于	x>=0	判断x是否大于等于0		
==	等于	x==0	判断x是否等于0	相同(低)	
!=	不等于	x!=0	判断x是否不等于0		

关系运算符都是双目运算符，其结合性均为左结合。关系运算符的优先级低于算术运算符，高于赋值运算符。在6个关系运算符中，<、<=、>、>=的优先级相同，高于==和!=、==和!=的优先级相同。

2. 关系表达式的使用格式

关系表达式的一般形式为：

表达式 关系运算符 表达式

例如：

```
a+b>c-d
x>3/2
'a'+1<c
-i-5*j==k+1
```

都是合法的关系表达式。由于表达式也可以同时又是关系表达式，因此也允许出现嵌套的情况。例如：

```
a>(b>c)
a!=(c==d)
```

关系表达式的值只有两个，分别是“真”和“假”，用“1”和“0”表示。

例如，5>0的值为“真”，即为1。(a=3)>(b=5)由于3>5不成立，故其值为假，即

为0。

【例 3-5】

```
//Exam ch03-5.c
#include<stdio.h>
void main()
{
    char c='k';
    int i=1,j=2,k=3;
    float x=3e+5,y=0.85;
    printf("%d,%d\n",'a'+5<c,-i-2*j>=k+1);
    printf("%d,%d\n",1<j<5,x-5.25<=x+y);
    printf("%d,%d\n",i+j+k==-2*j,k==j==i+5);
}
```

【程序运行】

```
1,0
1,1
0,0
```

【程序说明】

在本例中求出了各种关系运算符的值。字符变量是以它对应的ASCII码参与运算的。对于含多个关系运算符的表达式，如k==j==i+5，根据运算符的左结合性，先计算k==j，该式不成立，其值为0，再计算0==i+5，也不成立，故表达式值为0。

3.6 逻辑运算符和逻辑表达式

1. 逻辑运算符及其优先次序

C语言中提供了三种逻辑运算符，如表3-8所示。

表 3-8 逻辑运算符

运算符	名称	示 例	运算功能	优先级	结合性
!	逻辑非	!(x>0)	判断x是否不大于0	高	右结合
&&	逻辑与	x>0&&y>0	判断x和y是否同时大于0	较高	左结合
||	逻辑或	x>0||y>0	判断x或y是否大于0	低	

与运算符&&和或运算符||均为双目运算符。非运算符!为单目运算符。逻辑运算符和其他运算符优先级的关系可表示为如图3-4所示。

“&&”和“||”低于关系运算符，“!”高于算术运算符。按照运算符的优先顺序可以得出：

a>b && c>d 等价于 (a>b)&&(c>d)
!b==c || d<a 等价于 ((!b)==c) || (d<a)
a+b>c&&x+y<b 等价于 ((a+b)>c)&&((x+y)<b)

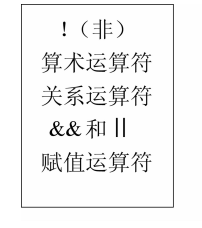

图 3-4 其他运算符优先级关系

2. 逻辑运算的值

逻辑运算的值也有“真”和“假”两种，用“1”和“0”来表示。其运算的真值表如表 3-9 所示。

表 3-9 逻辑运算真值表

左操作数 a	右操作数 b	a&&b	a||b	!a
非 0	非 0	1	1	0
非 0	0	0	1	0
0	非 0	0	1	1
0	0	0	0	1

其求值规则如下：

(1) 与运算 &&：参与运算的两个量都为真时，结果才为真，否则为假。

例如，5>0 && 4>2，由于 5>0 为真，4>2 也为真，相与的结果也为真。

(2) 或运算 ||：参与运算的两个量只要有一个为真，结果就为真。两个量都为假时，结果为假。

例如，5>0 || 5>8 由于 5>0 为真，相或的结果也就为真。

(3) 非运算!：参与运算量为真时，结果为假；参与运算量为假时，结果为真。

例如，!(5>0)的结果为假。

虽然 C 编译在给出逻辑运算值时，以“1”代表“真”，“0”代表“假”。但反过来在判断一个量是为“真”还是为“假”时，以“0”代表“假”，以非“0”的数值作为“真”。

例如，由于 5 和 3 均为非“0”因此 5&&3 的值为“真”，即为 1。

3. 逻辑表达式

逻辑表达式的一般形式为：

表达式 逻辑运算符 表达式

其中的表达式可以同时又是逻辑表达式，从而组成了嵌套的情形。

例如，(a&&b)&&c 根据逻辑运算符的左结合性，上式也可写为：a&&b&&c

逻辑表达式的值是式中各种逻辑运算的最后值，以“1”和“0”分别代表“真”和“假”。

【例 3-6】

```
//Exam ch03_6.c
#include<stdio.h>
void main()
```

```
{
    char c='k';
    int i=1,j=2,k=3;
    float x=3e+5,y=0.85;
    printf("%d,%d\n",!x*!y,!!!x);
    printf("%d,%d\n",x||i&&j-3,i<j&&x<y);
    printf("%d,%d\n",i==5&&c&&(j=8),x+y||i+j+k);
}
```

【程序运行】

```
0,0
1,0
0,1
```

【程序说明】

本例中!x和!y分别为0,!x*!y也为0,故其输出值为0。由于x为非0,故!!!x的逻辑值为0。对x||i&&j-3,先计算j-3的值为非0,再求i&&j-3的逻辑值为1,故x||i&&j-3的逻辑值为1。对i<j&&x<y,由于i<j的值为1,而x<y为0故表达式的值为1、0相与,最后为0,对i==5&&c&&(j=8),由于i==5为假,即值为0,该表达式由两个与运算组成,所以整个表达式的值为0。对于式x+y||i+j+k由于x+y的值为非0,故整个或表达式的值为1。

3.7 条件运算符和条件表达式

如果在条件语句中,只执行单个的赋值语句时,常可使用条件表达式来实现。不但使程序简洁,也提高了运行效率。

1. 条件运算符的使用格式

条件运算符为?和:,它是一个三目运算符,即有三个参与运算的量。由条件运算符组成条件表达式的一般形式为:

```
表达式1?  表达式2:  表达式3
```

2. 条件表达式的使用规则

其求值规则为:如果表达式1的值为真,则以表达式2的值作为条件表达式的值,否则以表达式3的值作为整个条件表达式的值。条件表达式通常用于赋值语句之中。

例如条件语句:

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

可用条件表达式写为

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

执行该语句的语义是：如果 a＞b 为真，则把 a 赋予 max，否则把 b 赋予 max。

使用条件表达式时，还应注意以下几点：

(1) 条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值符。因此 max＝(a＞b)?a：b 可以去掉括号而写为 max＝a＞b?a：b。

(2) 条件运算符? 和：是一对运算符，不能分开单独使用。

(3) 条件运算符的结合方向是自右至左。例如，a＞b?a：c＞d?c：d 应理解为 a＞b?a：(c＞d? c：d)。这也就是条件表达式嵌套的情形，即其中的表达式 3 又是一个条件表达式。

3.8 逗号运算符和逗号表达式

在 C 语言中逗号“,”也是一种运算符，称为逗号运算符。其功能是把两个表达式连接起来组成一个表达式，称为逗号表达式。

1. 逗号运算符的使用格式

其一般形式为：

```
表达式 1,表达式 2
```

2. 逗号表达式的使用规则

其求值过程是分别求两个表达式的值，并以表达式 2 的值作为整个逗号表达式的值。

对于逗号表达式还要说明以下两点：

(1) 逗号表达式一般形式中的表达式 1 和表达式 2 也可以又是逗号表达式。

例如，

```
表达式 1,(表达式 2,表达式 3)
```

形成了嵌套情形。因此可以把逗号表达式扩展为以下形式：

```
表达式 1,表达式 2,…,表达式 n
```

整个逗号表达式的值等于表达式 n 的值。

(2) 程序中使用逗号表达式，通常是要分别求逗号表达式内各表达式的值，并不一定要求整个逗号表达式的值。

【例 3-7】

```
//Exam ch03-7.c
#include<stdio.h>
```

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

【程序运行】

```
y=10,x=6
```

【程序说明】

本例中,y等于整个逗号表达式的值,也就是表达式2的值,x是第一个表达式的值。

注意:并不是在所有出现逗号的地方都组成逗号表达式,如在变量说明中,函数参数表中逗号只是用作各变量之间的间隔符。

3.9 求字节运算符

sizeof运算符是一个单目运算符,它返回变量或数据类型的字节长度。其一般形式为:

```
sizeof(类型标识符)或 sizeof(变量名)
```

例如:

```
sizeof(double)表达式值为 8;
sizeof(int)表达式的值为 4
```

设有下列程序段:

```
float f;
int i;
i=sizeof(f);
```

则变量i的值为4。

3.10 位运算符

前面介绍的各种运算都是以字节作为最基本位进行的,但在很多系统程序中常要求在位(bit)一级进行运算或处理。C语言提供了位运算的功能,这使得C语言也能像汇编语言一样用来编写系统程序。

1. 位运算符

C语言提供了6种位运算符,如表3-10所示。

表 3-10 位运算符

运算符	含义	示例	运 算 功 能
&	按位与	a&b	若 a 与 b 相应位都为 1,该位结果为 1
|	按位或	a|b	若 a 与 b 相应位都为 0,该位结果为 0
^	按位异或	a^b	若 a 与 b 相应位相同,该位结果为 0
~	取反	~a	若 a 的相应位为 1,该位结果为 0
<<	左移	a<<2	将 a 的二进制数左移 2 位,右补 0
>>	右移	a>>2	将 a 的二进制数右移 2 位,左补 0 或 1

注意:位运算要求运算对象只能是整型或字符型数据,不能为实型数据。

2. 按位与运算

按位与运算符“&”是双目运算符。其功能是参与运算的两数各对应的二进制位相与。只有对应的两个二进制位均为 1 时,结果位才为 1,否则为 0。参与运算的数以补码方式出现。

例如,9&5 可写算式如下:

```
  00001001  (9 的二进制补码)
&.00000101  (5 的二进制补码)
  00000001  (1 的二进制补码)
```

可见 9&5=1。

按位与运算通常用来对某些位清 0 或保留某些位。例如把 a 的高 8 位清 0,保留低 8 位,可作 a&255 运算(255 的二进制数为 0000000011111111)。

【例 3-8】

```
//Exam ch03-8.c
#include<stdio.h>
void main()
{
    int a=9,b=5,c;
    c=a&b;
    printf("a=%d\nb=%d\nc=%d\n",a,b,c);
}
```

【程序运行】

```
a=9
n=5
c=1
```

3. 按位或运算

按位或运算符“|”是双目运算符。其功能是参与运算的两数各对应的二进制位相