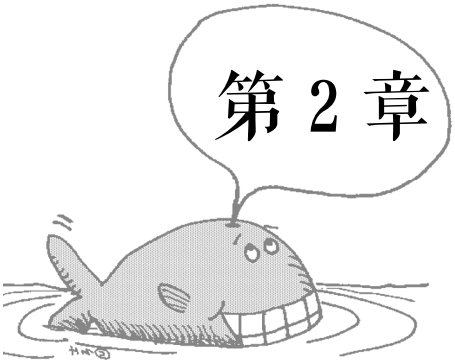


第 2 章

数据类型、运算符及表达式

在前一章中我们了解到 C 语言的发展、特点，学习了 C 语言程序的基本结构，知道 C 语言程序是由函数组成的，而函数体中的每条语句又是如何编写的，所以在程序设计之前，我们必须掌握一些基本知识。下面就从数据的变量与常量、C 语言所支持的数据类型、运算符及表达式开始学习吧。

2.1 C 语言的数据类型

程序处理的对象是数据，数据必须有类型。C 语言提供了丰富的数据类型，如图 2-1 所示。

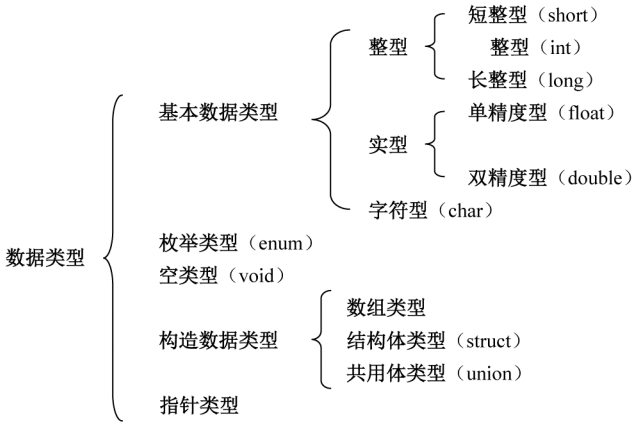


图 2-1　C 语言的数据类型

基本数据类型是构成 C 语言数据类型的最基本要素，平时实际编程中用到最多的实际上是整型、单精度型和字符型。灵活运用数据类型是学好程序设计的基本要求。

2.2 常量与变量

C 语言中的数据有常量和变量之分。常量是指在程序运行过程中，其值不能被改变的量。

变量是指在程序运行过程中，其值可以被改变的量。

2.2.1 常量

常量根据数据类型的不同也分为不同的类型。3、0、-14 是整型常量，3.5、-9.82 是实型常量，'a'、'*'是字符型常量，＂CHINA＂、＂3*5?7＂是字符串型常量。

1．整型常量

C 语言中整型常量可以用十进制、八进制、十六进制来表示。

（1）十进制整数：逢十进一，数字为 0～9，如：112，0，-67 等。

（2）八进制整数：逢八进一，数字为 0～7，为了与其他进制数区分，在数前加一个 0（注意：这是数字 0 而不是字母 o），如 032，-056 等。

（3）十六进制整数：逢十六进一，用 0～9、A～F（字母 A 代表 10，字母 B 代表 11，以此类推）表示，为了与其他进制数区分，在数前加一个 0x（数字 0 和字母 x），如 0x1A3，-0x67 等。

以上三种进制的数可以相互转换。十进制整数转换成八进制数或十六进制数可用除基数取余数倒数的方法；八进制、十六进制数转换成十进制数可采用权位相加法。来看如下程序。

【例 2.1】

```
#include<stdio.h>
main()
{
int a=112,b=0112,c=0x112;
printf("%d, %d, %d\n",a,b,c);
a=b=c=418;
printf("%d, %o, %x\n",a,b,c);
}
```

程序运行结果：

```
112, 74, 274
418, 642, 1a2
```

程序说明：

（1）0112 是一个八进制整数，要转换成十进制整数可使用权位相加法。

将位号从右向左依次标上：2　1　0

$$0\ 1\quad 1\quad 2 = 1\times 8^2+1\times 8^1+2\times 8^0=64+8+2=74$$

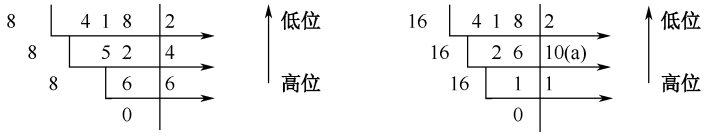
八进制数基数是 8，如果是十六进制数则基数要换成 16。

将位号从右向左依次标上：2　1　0

$$0x\quad 1\quad 1\quad 2 = 1\times 16^2+1\times 16^1+2\times 16^0=256+16+2=274$$

是不是很简单？想一想，你学会了吗？

（2）语句 a=b=c=418;的作用是将 a、b、c 的值都赋为 418，显然是十进制数。printf 函数中的“ %d ”表示以十进制整数输出，“ %o ”表示以八进制整数输出，“ %x ”表示以十六进制整数输出。这里存在十进制与八进制、十六进制相互转换的过程，如下所示。



注意

以“%o”输出的数在显示屏幕上或在理论分析程序中写出运行结果时是不加标识 0 的，同样，“%x”输出数时也不加 0x。

2．实型常量

实数就是带小数的数。实型常量只能由十进制数形式表示，通常有以下两种表示方法。

（1）十进制数形式：由数字（0～9）和小数点组成，如 3.14，-0.36，145.23 等。

（2）指数形式：类似于科学计数法，如要表示 45×10^2，需要写成 45e2 或 45E2，C 语言要求这种指数形式的字母 e 或 E 之前必须有数字且 e 或 E 后面的指数必须是整数。

例如，3.2e-3（3.2×10^{-3}）、2E5（2×10^5）、-0.5e-5（-0.5×10^{-5}）合法，12（没有小数点）、e3(e 之前没有数字)、2E-0.5（E 后面必须是整数）、-4.2e（e 后面没有数，说明没有阶码）不合法。

注意

幂为正数时，其正号（+）可以省略，如 2E5 相当于 2E+5；一个浮点数可以有多种指数表示形式。例如，12.34 可以表示为 12.34e0、1.234e1、123.4e-1、0.1234e2、0.01234e3、1234e-2 等。其中，1.234e1 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，小数点左边应有一位且只能有一位非零的数字。浮点数在用指数形式输出时，按规范化的指数形式输出。

3．字符型常量

C 语言的字符型常量是由单引号（单撇）括起来的一个字符。例如，'*'、'? '、'a'、'A'、'3' 等。这里'a'和'A'视为不同的字符常量。C 语言中还有一种以“ \ ”开头的特殊形式的字符常量，如表 2-1 所示。

表 2-1　转义字符及其功能

转 义 字 符	含义及功能
\n	换行，将当前位置移到下一行开头
\t	水平制表，跳到下一个 Tab 位置
\b	退格，将当前位置移到本行开头
\r	回车，将当前位置移到本行开头
\f	换页，将当前位置移到下页开头
\\	输出反斜杠\
\'	输出单撇'
\"	输出双撇 "
\ddd	输出 1～3 位八进制数所代表的字符
\xhh	输出 1～2 位十六进制数所代表的字符

表 2-1 中倒数第二行'\ddd'这种形式的转义字符，如'\101'代表 ASCII 码值为八进制数 101 的字符'A'，'\0'或'\000'代表 ASCII 码值为 0 的控制字符，指空操作用在字符串中。

注意

'7'和 7 不同，加单撇号的是字符常量，不加的是整型常量。它们在计算机中的存储方式完全不同，字符常量占一个字节（8 位二进制），存储的是该字符的 ASCII 码值，而整型常量理论上占 2 个字节（16 位二进制）的存储空间，在 VC++中被处理成 4 个字节。

4．字符串常量

C 语言的字符串常量是由一对双引号（双撇）括起来的字符序列。

字符串常量在内存中存储时，除了字符本身所占的字节数外，还在尾部自动增加一个转义字符'\0'作为字符串结束标识。

所以字符串"A"与'A'是有区别的，"A"是字符串常量，占 2 个字节；'A'是字符常量，占用 1 个字节。

5．宏

在 C 语言中可以使用宏定义命令#define 来定义一个常量。

语法格式：

```
#define  标识符  常量表达式
```

【例 2.2】已知圆半径值是 5.2，求圆面积。

```
#include<stdio.h>
#define  PI  3.14
main()
{
float r,s;
r=5.2;
s=PI*r*r;
printf("s=%f\n",s);
}
```

程序运行结果：

```
s=84.905594
```

程序说明：

（1）以“ %f ”输出，默认小数有 6 位，如果仅想输出 2 位小数位，则可用“ %.2f ”格式，圆点后面的 2 指 2 位小数位，注意格式中的小圆点不可省。

（2）程序使用#define PI 3.14 来定义 PI 的值是 3.14，如果想提高运算精度，可将 3.14 改为 3.14159。

使用符号常量的好处是“ 一改全改 ”。修改程序继续求圆的周长、圆的体积，上机调试并体验符号常量的方便性。

2.2.2 变量

变量是指在程序运行过程中其值可以改变的量。变量是用来存放数据的，每个变量都有一个确定的数据类型且占用内存一定的存储单元。常见的有整型变量、实型变量和字符型变量等。

（1）变量名、变量值要区分。

语句 int x=5,y=4;的作用是定义 x 和 y 两个整型变量且赋有初始值。x 和 y 是变量名，5 和

4 是整型常量，也是变量的值。

（2）不同类型的变量占不同字节数的存储单元。

理论上来说，int 变量占 2 个字节的宽度（1 个字节等于 8 个二进制位），long 变量占 4 个字节，float 占 4 个字节，double 占 8 个字节，而 char 占 1 个字节。当然，在不同的编译系统中可能有些变化，如 VC++ 6.0 中 int 占用 4 个字节处理。

（3）在 C 语言中，变量一定要“先定义，后使用”。

定义变量的一般形式：

```
类型    变量名表；
```

例如：

```
char  c1,c2;
float  x=3.14;
int  y;
```

（4）标识符是用来表示各种程序对象的，如变量名、函数名、文件名等。C 语言规定标识符只能是字母、数字、下画线且第一个字符必须是字母或下划线（_）。编译系统区分字母大小写，如 sum 与 SUM 是两个不同的变量名。与常量不同，习惯用小写字母表示变量名。例如，ab_c、sum、_name、china3 是合法的标识符，*123、3numbe 则不合法。

（5）关键字不能用作标识符，也不能作为变量名使用。

关键字又称保留字，是 C 语言预先声明的具有特定意义的单词。C 语言中的关键字共有 32 个，即

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	static	while
sizeof	struct	switch	typedef	union	unsigned	void	volatile

1．整型变量

整型变量分为以下 3 种类型。

（1）基本整型，用 int 表示，占 2 个字节的存储空间。

（2）短整型，用 short int 或 short 表示，占 2 个字节的存储空间。

（3）长整型，用 long int 或 long 表示，占 4 个字节的存储空间。

如果在 int、short、long 前面加上 unsigned 修饰符，则表示无符号整型、无符号短整型、无符号长整型，如果不加或加修饰符 signed，则表示有符号。

数据在内存中以二进制形式存放。例如，int a=10;语句的作用是定义了一个整型变量 a 并且赋初始值 10，则这个数据在内存中实际存放的是 10 的补码，如图 2-2 所示。

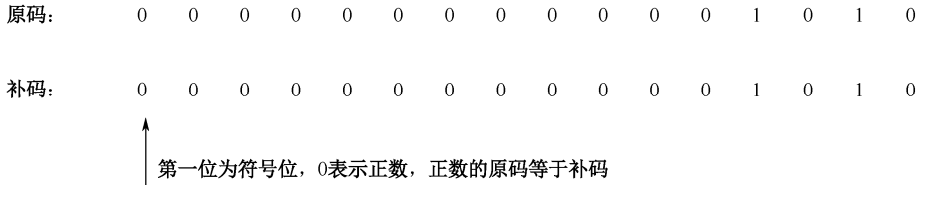


图 2-2

如果 a 的初始值改为-10，则在内存中实际存放的是-10 的补码，如图 2-3 所示。

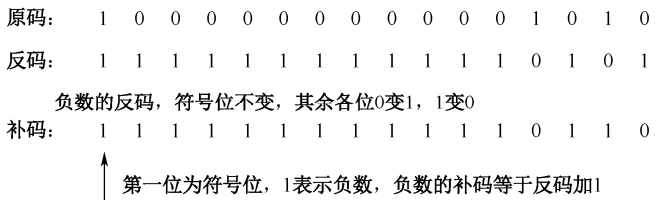


图 2-3

按照占 2 个字节来算，int 型数值范围是-32768～+32767，unsigned int 型数值范围是 0～65535。在 VC++ 6.0 中，为整型数据分配 4 个字节，其数值范围是多少，你能推算出来吗？当数据超出了所能表示的最大范围时会产生“溢出”。

【例 2.3】

```
#include<stdio.h>
main()
{
int a,b,c;
a=32;
b=a+14;
c=a-b;
printf("%d,%d,%d\n",a,b,c);
}
```

程序运行结果：

```
32, 46, -14
```

2．实型变量

实型变量分为以下 3 种类型。

（1）单精度型，用 float 表示，占 4 个字节的存储空间。

（2）双精度型，用 double 表示，占 8 个字节的存储空间。

（3）长双精度型，用 long double 表示，占 16 个字节的存储空间，但在 VC++ 6.0 中做 8 个字节处理，又因长双精度型使用得少，因此不再详细介绍。

【例 2.4】

```
#include<stdio.h>
main()
{
float x1,x2;
x1=23.0;
x2=3;
printf("%f,%f\n",x1*x2,x1/x2);
}
```

程序运行结果：

```
69.000000, 7.666667
```

【例 2.5】

```
#include<stdio.h>
main()
{
float x;
```

```
x=2345.123456;
printf("%f\n",x);
}
```

程序运行结果：

```
2345. 123535
```

程序说明：

一个实型常量可赋值 float 型或 double 型变量，float 型变量可接收 7 位有效数字，double 型变量可接收 15 位有效数字。程序的结果只有 2345.123 是有效的，后面的数字不准确，如果改成 double 型，运行结果就可以显示为 2345.123456。

3．字符型变量

字符型变量用 char 来定义，一个字符型变量只能存放一个字符常量。

例如，

```
char  c1='A'；
char  c2='a'；
```

这样的定义也可以写成：

```
char c1='A', c2='a'；
```

一个字符常量放到一个字符变量中，实际上是把该字符的 ASCII 码值存放到存储单元中。因为'A'的 ASCII 码值是 65，所以变量 c1 在内存中存放的是 65 的二进制数 01000001，而变量 c2 在内存中存放的是 97 的二进制数 01100001。由于字符的 ASCII 码表中字母是按顺序排列的，所以其他字母的 ASCII 码值可以推算出来，试想一下，字符'C'、'f'又是多少？注意，字符常量'0'的 ASCII 码值是 48。

【例 2.6】

```
#include<stdio.h>
main()
{
char c1,c2;
c1='A';
c2='a';
printf("%c, %c\n",c1,c2);
c1=c1+32;
c2=c2-32;
printf("%c, %c\n",c1,c2);
printf("%d, %d\n",c1,c2);
}
```

程序运行结果：

```
A, a
a, A
97, 65
```

程序说明：

（1）“ %c ”指以字符输出，所以第一条 printf 函数输出 A，a；因为小写字母的 ASCII 码值比大写字母的 ASCII 码值大 32，所以第二条 printf 函数输出 a，A；整型与字符型通用，字符型变量以“ %d ”输出时显示其 ASCII 码的值，所以第二条 printf 函数输出 97，65。

（2）字符型常量用单引号括起来，字符串常量是用双引号括起来的，如果程序中使用 char c1="A"，则会出现错误。"A"占 2 个字节的存储空间，而字符型变量 c1 的定义只开辟 1 个字节的存储空间，显然无法存储，如果要把一个字符串存放到变量中，则可以使用字符数组，这在后面章节中会介绍。

2.3 运算符与表达式

C 语言中提供了很丰富的运算符，可归类成以下几类。

（1）算术运算符：+、-、*、/、%、++、- -。

（2）关系运算符：>、<、>=、<=、==、! =。

（3）逻辑运算符：&&、||、!。

（4）赋值运算符：=。

（5）条件运算符：？ :。

（6）逗号运算符：,。

（7）求字节数运算符：sizeof。

（8）指针运算符：*和&。

（9）位运算符：<<、>>、～、｜、^、&。

（10）其他：强制类型转换运算符、分量运算符、下标运算符、函数调用运算符等。

表达式是用运算符和括号把操作数连接起来的式子，如 5+3。本章将重点介绍（1）～（7）的运算符及表达式的运算。

2.3.1 算术运算符与表达式

（1）+（加法运算符）、-（减法运算符）、*（乘法运算符），在运算功能中等同于数学上的 +、-、×。

例如，5+6，10-4，2*（-3）。

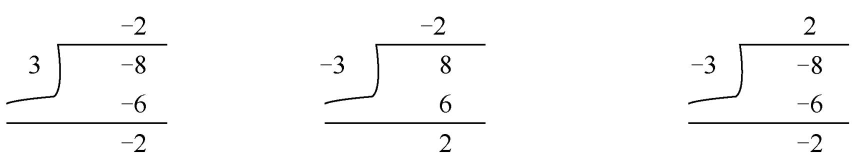
（2）/、%都是除法运算符。前者是整除，取整数部分，小数部分丢弃；后者取余，求余数，也称为模运算符，“ % ”运算符要求两边均为整型。

例如，8/2 的结果是 4，8/3 的结果是 2；

8%2 的结果是 0，8%3 的结果是 2；

想一想：（-8）%3，8%（-3），（-8）%（-3）的结果又是多少？

这种情况下可带符号一起建立除式来求解，如下所示。



是不是很简单，你学会了吗？请算一下（-9）/2，9/（-2），（-9）/（-2）的结果。

（3）++（自增运算符）和- -（自减运算符）的作用是使变量的值增 1 或减 1，只能用于变量，不能用于常量或表达式。

++有两种形式：++i 和 i++。

- -有两种形式：- -i 和 i- -。

【例 2.7】

```
#include<stdio.h>
main()
{
int i=3,j=4,m,n;
m=++i;
n=j--;
printf("i=%d\nj=%d\nm=%d\nn=%d\n",i,j,m,n);
}
```

程序运行结果：

```
i=4
j=3
m=4
n=4
```

程序说明：

（1）语句 m=++i;的作用是将 i 的值先加 1，后赋值给变量 m，这样可得到 i 的值是 4，m 的值也是 4。

（2）语句 n=j--;的作用是先取 j 的值作为表达式的值赋给变量 n，而后 j 的值减 1。这样就得到 n 的值是 4，j 的值是 3。

（3）++、- -的结合方向是“自右向左”。

（4）算术运算符的结合性是“从左向右”，先乘除后加减，有括号的先算括号里的。

（5）利用算术运算符可以写出一些数学表达式。

① $\frac{a-b}{a+b}$ 写成标准的 C 语言表达式是(a-b)/(a+b)；

② $2x^2+4y$ 写成标准的 C 语言表达式是 2*x*x+4*y，或者利用数学函数 pow，写成：2* pow(x,2)+4*y。

注意

如果程序中使用 C 语言提供的数学库函数，就应加一条头文件#include<math.h>。常用的数学函数有 sqrt（求平方根）、abs（求整数的绝对值）、fabs（求实数的绝对值）等。

2.3.2 关系运算符与表达式

（1）>（大于）、<(小于)运算符的作用是比较大小，如 5>3 的运算结果为 1，因为条件满足，为真，表达式的值为 1；反之，5<3 运算结果为 0，因为条件不满足，为假。

（2）>=（大于或等于）、<=（小于或等于）运算符的作用也是比较，例如，3>=3 运算结果为 1，因为大于或等于中只要一个满足，结果就为满足，值当然就是 1。

（3）==（等于）、!=（不等于）运算符的作用是比较运算符号两边的值是否相等，如果值一样，“==”运算的结果为 1，如果不同，则值为 0，“!=”运算则相反。

【例 2.8】

```
#include<stdio.h>
main()
{
int a,b,c,d,e;
a=2;
b=-3;
c=10;
d=(a+2)<b;
e=b>=(7-c);
printf("%d,%d\n",d,e);
}
```

程序运行结果：

```
0, 1
```

2.3.3 逻辑运算符与表达式

（1）&&（逻辑与）、||（逻辑或）运算符是双目运算符，它要求两边都有操作数。

||（逻辑或）运算符是单目运算符，结合性是“自右向左”，逻辑运算的结果只有真与假，在 C 语言中分别用 1 和 0 表示。

① &&（逻辑与）运算符两边的操作数相当于一个工程的两个任务，判断这个工程是否完成需要判断两个任务是否完成。如果两个任务都完成，则这个工程完成了，也就是两边的值都为真时，表达式才为真，运算结果是 1；如果有一个任务不完成，为假，表达式就为假，运算结果是 0。

例如，(a<b)&&(c==d)，假设有定义 int a=1,b=2,c=3,d=4;，算出&&的左边 a<b 值为 1，右边 c==d 值为 0，那么整个表达式的值为 0。

② ||逻辑或运算符两边的操作数也相当于一个工程的两个任务，判断这个工程是否完成需要判断两个任务中是否有任务完成，只要两个任务中有一个任务完成了，则表达式为真，运算结果是 1，如果都没完成，则运算结果为 0。

上例中，将逻辑与改为逻辑或，那么表达式的值为 1。

③ !（逻辑非）运算符是逻辑运算符中优先级别最高的。逻辑非的作用是取相反，如果!右边为真，则运算结果为 0；如果!右边为假，则运算结果为 1。

（2）三个逻辑运算符的优先级别是！（逻辑非）→&&（逻辑与）→||（逻辑或）。

例如，a||b && c　相当于　a||(b && c)。

（3）假定两边的操作数用 a 和 b 来表示，表 2-2 列出了各种逻辑值的运算规则。

表 2-2　逻辑运算的真值表

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	

【例 2.9】

```
#include<stdio.h>
main()
{
int x=12,y=-8,z=5;
printf("%d\n",x>=y&&z-5);
printf("%d\n",x*2-16||y);
printf("%d\n",!z/2&&(y+2));
}
```

程序运行结果：

```
0
1
0
```

（4）&&（逻辑与）与||（逻辑或）又称为短路运算符。&&的左边值是 0，则右边被短路，逻辑与的值是 0；||的左边值是 1，则右边也被短路，逻辑或的值是 1。

【例 2.10】

```
#include<stdio.h>
main()
{
int i,j,k,p;
i=j=k=-1;
p=++i&&j--&&k++;
printf("%d\n",p);
printf("%d,%d,%d\n",i,j,k);
i=j=k=3;
p=i++&&j--||--k;
printf("%d\n",p);
printf("%d,%d,%d\n",i,j,k);
}
```

程序运行结果：

```
0
0, -1, -1
1
4, 2, 3
```

2.3.4 赋值运算符与表达式

（1）赋值运算符是“ = ”，作用是将赋值号右边的值或表达式的值赋给赋值号左边的变量。例如：

```
x=5；
x=y+2；
x=(z=4)；相当于x=z=4；          //赋值的结合性是“ 自右向左 ”
```

（2）复合运算符（复合的赋值运算符）：在 C 语言中，在赋值运算符之前加上算术运算符或位运算符即可构成复合的赋值运算符。共有 10 种，分别如下：

+=、 - =、 *=、 /=、 %=、 <<=、 >>=、 &=、 ^=、 ｜=。

例如：

```
x+=5；相当于x=x+5；
z-=x+2；相当于z=z-（x+2）；      //算术运算优先
```

【例 2.11】

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

程序运行结果：

```
a= -112
c=1
```

2.3.5 逗号运算符与表达式

逗号运算符“，”是一个比较特殊的运算符，作用是将若干个表达式连接起来。

（1）一般形式如下。

```
表达式1，表达式2，……表达式n
```

例如，2*5，10/2。

整个逗号表达式的值是最后一个表达式的值，所以运算结果是 5。

（2）逗号运算符是所有运算符中级别最低的。

例如，a=b=2，2+7 相当于

（a=b=2），2+7。

该表达式的值是 9。

（3）逗号表达式可以嵌套，即一个逗号表达式可以与另一个逗号表达式组成一个新的表达式。

例如，（x=2-3，x*5），x+3 表达式的结果是 2。

① 括号先计算，（x=-1，-1*5），这个逗号表达式结果是-5，而 x=-1。

②（x=2-3，x*5），x+3 简化后是-5，-1+3 的结果为 2。

2.3.6 条件运算符与表达式

条件运算符是一个三目运算符，也是 C 语言中唯一的三目运算符，有三个操作对象。

（1）一般形式如下。

```
表达式1？表达式2：表达式3
```

（2）功能：判断表达式 1 的值是真或假，为真则执行表达式 2，为假则执行表达式 3。

例如：

```
max=(a>b)?a:b；
```

作用：判断 a 与 b 值的大小。如果 a>b 成立，则把 a 的值赋给变量 max；如果 a>b 不成立

则把 b 的值赋给变量 max。

（3）几点说明。

① 条件运算符“ ？ ”与“ :”不可分。

② 条件运算符的优先级低于关系运算符和算术运算符，但高于赋值运算符。

max=(a>b)?a:b； 等同于 max=a>b?a:b；

③ 条件运算符也可以嵌套，其结合方向是“ 自右向左 ”。

a>b?a:c>d?c:d 等同于 a>b?a:（c>d?c:d）。

【例 2.12】

```
#include<stdio.h>
main()
{
int a=10,b=20,c=30,d=40;
printf("%d",a>b?a:c>d?c:d);
}
```

程序运行结果：

```
40
```

④“ 表达式 2 ”和“ 表达式 3 ”不仅可以是数值表达式，还可以是赋值表达式或函数表达式。

【例 2.13】

```
#include<stdio.h>
main()
{
int x=10,y=20;
printf("%d",x==y?(x=100):(y=200));
}
```

程序运行结果：

```
200
```

【例 2.14】

```
#include<stdio.h>
main()
{
int  x=10;
x%2==0?printf("偶数!"):printf("奇数!");    //这是条件表达式语句
}
```

程序运行结果：

```
偶数！
```

2.3.7 求字节数运算符与表达式

C 语言提供了整型、实型、字符型等多种数据类型，但因使用的编译系统不同而造成有些数据类型的长度也不同。使用 sizeof 运算符可以正确了解各类型的长度。

语法格式：

```
sizeof(类型名或变量名)
```

功能：给出指定类型在内存中所占的字节数。

【例 2.15】在 VC++ 6.0 中调试以下程序。

```
#include<stdio.h>
main()
{
int x;
long y;
float z;
double k;
char c;
printf("int占%d个字节\n",sizeof(x));
printf("long占%d个字节\n",sizeof(y));
printf("float占%d个字节\n",sizeof(z));
printf("double占%d个字节\n",sizeof(k));
printf("char占%d个字节\n",sizeof(c));
}
```

程序运行结果：

```
int占4个字节
long占4个字节
float占4个字节
double占8个字节
char占1个字节
```

如果习惯使用 Turbo C 的运行环境，也可用以上例题的方法调试并查看各数据类型所占的字节数，比较一下它们的不同点。

2.3.8 指针运算符及位运算符

1．指针运算符

指针是 C 语言的特色，也是难点，后续章节中会详解，这里先来学习两个基本运算符：&（取地址运算符）、*（取指针目标运算符，其后跟随的必须是指针类变量）。

前面已经学习过，变量必须要“先定义，后使用”。定义变量的目的是给变量分配一个临时存储空间，除变量值与变量名外还会出现一个全新的概念——地址。

【例 2.16】

```
#include<stdio.h>
main()
{
int i=10;                    //定义整型变量i，为整型变量分配存储空间
printf("%d\n",i);
printf("%d\n",&i);           //取分配给变量i的首地址，以十进制形式输出
printf("%x\n",&i);
}
```

程序运行结果：

```
10
1638212
18ff44
```

程序说明：

（1）经过程序调试后，通过求&i 得到变量 i 的首地址是 18ff44，这是十六进制形式的。

（2）为什么说是首地址呢？一般而言，一个字节占 8 位，定义一个整型变量就得给相应的变量分配一个 2 个字节的存储空间，VC++ 6.0 中就分配 4 个字节，你明白了吗？取地址就用&运算符，而定义一个指针变量就要用*，后面有一章会专门介绍，这里不再详述。

2．位运算符

位运算符对操作数的各个二进制位进行运算，包括 6 种，分别是&(按位与)、|（按位或）、^（按位异或）、～（按位非）、<<（左移）、>>（右移）。

先来简单学习位运算的规律。

1）&（按位与运算符）

0 & 0=0　　0 & 1=0　　1 & 0=0　　1 & 1=1

与运算的特点是只有 1 与 1 才是 1，其余情况均为 0。与可以将某位或某几位二进制位清零，如果想让 11111111 的前 4 位清零而后 4 位不变，则可以和 00001111 相与（其值是 15）。

2）|（按位或运算符）

0 | 0=0　　0 | 1=0　　1 | 0=0　　1 | 1=1

或运算的特点是只有 0 或 0 才是 0，其余情况为 1。或可以将某位或某几位二进制位置 1，如果要把 00000000 的前 4 位置 1 后 4 位不变，则可以与 11110000 相或（其值是 240）。

3）^（按位异或运算符）

0 ^ 0=0　　0 ^ 1=1　　1 ^ 0=1　　1 ^ 1=0

这里教大家一个口诀：“ 异或异或求不同 ”。只有两个不同数的异或才是 1。

4）～（按位非运算符）

～ 0=1　　～ 1=0

按位非是单目运算符，其功能是对二进制数求反，即 0 变 1，1 变 0。

例如，～8 的运算实为～(00001000)，其结果是 11110111。

5）<<（左移运算符）

例如，a<<3 是指将<<左边的操作数的各二进位左移 3 位，左移时高位丢失，低位补 0。

假设变量 a 的值是 4（00000100），按表 2-3 操作。

表 2-3　左移操作

原　数	0	0	0	0	0	1	0	0
移 1 位	0	0	0	0	1	0	0	0
移 2 位	0	0	0	1	0	0	0	0
移 3 位	0	0	1	0	0	0	0	0

原数 00000100 是 4，移动 3 位后 00100000 是 32，变成原数的 8 倍（2^3），如此可总结出：如果左移 n 位，则移后的数就是原数的 2^n 倍。

6）>>（右移运算符）

右移运算符也是双目运算符，其功能是右移。注意，对于无符号数高位补 0；对于有符号数右移时，正数的最高位补 0，负数的最高位补 1 还是 0 取决于计算机系统。

【例 2.17】

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

```
void main()
{
int x=76,y=255,z=-2;
char c1='A',c2='b';
printf("%d,%x\n",x&y,x&y);
printf("%d,%x\n",x|y,x|y);
printf("%d\n",x^y);
printf("%d\n",~c1);
printf("%o\n",c1<<2);
printf("%x\n",z>>1);
}
```

程序运行结果：

```
76, 4c
255,ff
179
-66
404
ffffffff
```

通过运行以上程序得到一个结论，编译系统 VC++ 6.0 中右移运算是算术右移，补符号位，感兴趣的读者还可以在 Turbo C 上自己验证得到。

2.4 不同类型数据间的混合运算

整型和浮点型可以混合运算，字符型与整型又可以通用，所以整型、实型、字符型间可以混合运算，但需要转换，方法有自动类型转换和强制类型转换两种。

1．自动类型转换

1）赋值运算中的类型转换

例如：

```
int x;
x=3.14;
```

x 的值是 3。当实数赋给整型变量时，自动丢弃小数部分。

2）混合运算中的类型转换

例如：

```
2*'A'+1.5
```

不同类型的数据进行混合运算时，C 编译系统自动转换，其转换规则如图 2-4 所示。

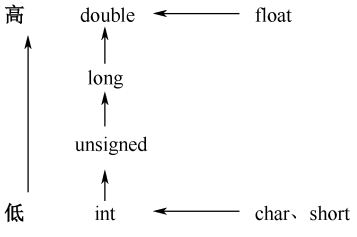


图 2-4

注意

箭头方向只表示数据类型级别的高低，由低向高转换。不要理解为 int 先转换成 unsigned int 型，再转换成 long 型，最后转换成 double 型。

上例中先做，2*'A'，将'A'转换成整型 65，运算结果为 2*65=130；再做加，130+1.5=131.5。运行结果是 131.5，数据类型就是 1.5 的类型。假定 1.5 是 float 型，那么最后结果就是 float 型，如果 1.5 是 double 型，那么最后结果就是 double 型。

2．强制类型转换

（1）一般形式如下。

```
（类型名）（表达式）
```

例如：

```
(float)x          //将x强制转换成float型
(int)(x+y)        //将x+y的值强制转换成int型
```

注意：表达式要用括号括起来。

```
(int) x+y         //只将x强制转换成int型，然后与y 值相加
```

（2）强制转换会得到一个所需的中间变量，原来的数据类型并未发生变化。

【例 2.18】

```
#include<stdio.h>
main()
{
float m;
int n;
m=-34.28;
n=(int)m;
printf("%d\n",n);
printf("%f\n",m);
}
```

程序运行结果：

```
-34
-34.280000
```

语句 n=(int)m;的功能是将 m 强制转换后得到-34 并赋给变量 n，m 的数据类型还是 float 型，值仍为-34.28。

（3）运算符的优先级与结合性：作为 C 语言的基础，首先要能正确算出表达式的值。附录 3 列出的是全部运算符的优先级和结合性规则。同一行具有相同优先级，上一行运算符的优先级高于下一行。同一行的运算符，按结合性规则计算。

（4）当表达式中既有自动类型转换又有强制类型转换时，又该如何分析呢？

【例 2.19】

```
#include<stdio.h>
main()
{
int x=1;
float y=4.2,z=-9.78,k;
char c='d';
```

```
    k=(float)x*2+((z>y)&&--x)-(int)(z-12.3)%(c-97);
    printf("%f\n",k);
    printf("%d\n",x);
    }
```

程序运行结果：

```
    3.000000
    1
```

分析步骤：

（1）求表达式(float)x*2+((z>y)&&--x)-(int)(z-12.3)%(c-97)，需先把优先级或顺序整理一下，(float)x*2+((z>y)&&--x)-(int)(z-12.3)%(c-97)，可分成 3 段。

（2）(float)x*2，先求(float)x，值为 1.0，做(float)x *2=2.0；

(z>y)&&--x，先求(z>y)，-9.78>4 不满足为假，值为 0，做 0&&--x 运算，由于&&是短路运算符，所以--x 被短路，不做运行，x 的值不变仍是 1，表达式(z>y)&&--x 的值是 0；

(int)(z-12.3)%(c-97)，先做(z-12.3)，-9.78-12.3=-22.08，被强制转换成-22，再做(c-97)，由于变量 c 的值是'd'，做'd'-97，字符型与整型通用，所以用字符 d 的 ASCII 码值 100-97=3，-22%3=-1。算下来，(int)(z-12.3)%(c-97)的值是-1。

（3）经过运算，表达式(float)x*2+((z>y)&&--x)-(int)(z-12.3)%(c-97)就成为 2.0+0-(-1)=3.0，所以整个表达式的值就是 3.0，以“%f”输出，保留 6 位小数。

本章小结

本章主要介绍了数据类型、常量、变量、运算符及表达式。

C 语言中常见的数据类型就是整型、实型、字符型。变量必须先定义后使用，选择数据类型时需要与常量类型相对应，变量的命名需要符合标识符的定义规则，只能包括下画线、字母、数字且必须以下画线或字母开头，关键字不能作为变量名且区分字母大小写。

C 语言中运算符丰富，常用的有算术运算符、关系运算符、逻辑运算符、赋值运算符、逗号运算符、条件运算符、求字节数运算符。用运算符和数据组成的式子就是表达式。不同类型数据间混合运算时一定要弄清优先级别和结合性。

本章习题

一、简答题

1. 常见的数据类型有哪些？请写出其对应的关键字。
2. 什么是常量？什么是变量？
3. 什么是标识符？定义时有什么规则？
4. 你知道常见的运算符有哪些？请分类写出。

二、填空题

1. 字符'F'占内存_____个字节，字符串"chinese"占内存_____个字节。
2. 定义一个变量 m 为整型且赋初始值是 78，应写成语句_____。