第3章 基本数据类型和表达式

计算机解决各种实际问题的本质是对数据进行处理，程序中对数据处理由对应的程序语句来完成。因此，需要掌握数据在计算机中的表示和存储形式，掌握利用运算符、变量、常量按一定规则组成表达式对数据进行运算。只有掌握了这些基本知识，才能顺利地进行C++语言程序设计。

本章主要介绍C++语言程序设计的基本部分，包括基本数据类型、常量与变量、运算符与表达式、数据类型转换及常用库函数。

3.1 数据类型

程序处理的对象是数据，数据分为常量和变量，每个常量或变量都有数据类型。C++语言中的数据类型分为两大类：基本数据类型和导出数据类型。其中，基本数据类型是C++语言中预定义的类型，包含整型(int)、字符型(char)、布尔型(bool)、单精度浮点型(float)、双精度浮点型(double)和空类型(void)；导出数据类型也称自定义数据类型，是用户根据程序设计需要，按语法规则由基本数据类型构造的，包含数组、指针、结构体、联合体、枚举和类等。表3-1中列出了C++语言的基本数据类型。

表3-1 C++语言的基本数据类型

类　　型	字节数	取值范围
bool(布尔型或逻辑型)	1	true或false
[signed]char(有符号字符型)	1	−128～+127
unsigned char(无符号字符型)	1	0～255
[signed]int(有符号整型)	4	-2^{31}～$+2^{31}-1$(−2 147 483 648～+2 147 483 647)
unsigned[int](无符号整型)	4	0～$2^{32}-1$(0～4 294 967 295)
[signed]long[int](有符号长整型)	4	-2^{31}～$+2^{31}-1$(−2 147 483 648～+2 147 483 647)
unsigned long[int](无符号长整型)	4	0～$2^{32}-1$(0～4 294 967 295)
[signed]short[int](有符号短整型)	2	-2^{15}～$+2^{15}-1$(−32 768～+32 767)
unsigned short[int](无符号短整型)	2	0～$2^{16}-1$(0～65 535)
float(单精度浮点型)	4	-3.4×10^{-38}～$+3.4\times10^{38}$
double(双精度浮点型)	8	-1.7×10^{-308}～$+1.7\times10^{308}$
long double(长双精度型)	8/16	-1.7×10^{-4932}～$+1.7\times10^{4932}$
void(空值型)		

说明：

(1) 表3-1中的符号“[]”表示可选，其中的内容为默认，在定义变量时“[]”中的内容可以缺省。例如，signed char等同于char，表示有符号字符型；signed int等同于int，表示有

符号整型。

(2) 符号修饰符 signed、unsigned 与长短修饰符 short、long 连同类型名 int 可以随意组合。例如,unsigned long int、long unsigned int、int unsigned long 都表示无符号长整型。

(3) 空值型用于描述没有返回值的函数及通用指针类型。

(4) 表 3-1 中的字节数和取值范围是基于 32 位操作系统给出的。在 64 位操作系统中,long、unsigned long 为 8 字节,其余相同。

(5) 单精度型浮点数(float)的有效数字为 7 位,双精度型浮点数(double)的有效数字为 15 位,长双精度型浮点数(long double)的有效数字为 15 位(占 8 字节时)或 19 位(占 16 字节时)。

(6) 不同的编译系统对 long double 的处理方法不完全相同,有的分配 8 字节,有的分配 16 字节。

【例 3.1】 输入圆的半径,求圆的面积并输出。

```
#include <iostream>
using namespace std;
const double PI = 3.14159;
int main()
{   double r,c,area;                    //变量 r 表示圆的半径,c 表示圆的周长,area 表示圆的面积
    cout <<"输入圆的半径:";             //提示用户输入圆的半径
    cin >> r;
    c = 2 * PI * r;
    area = PI * r * r;
    cout <<"pi = "<< PI << endl;
    cout <<"圆的周长:"<< c << endl;
    cout <<"圆的面积:"<< area << endl;
    return 0;
}
```

例 3.1 中定义了双精度类型变量 r、c 和 area,运行时根据输入的 r 值计算 area 和 c。2 * PI * r 是表达式,表达式中有常量 PI 和 2。常量和变量有什么区别? 常量和变量应该如何定义? 表达式怎样表示? 这些问题将在本章逐一讲解。

3.2 常　　量

常量是指在程序运行过程中值不能被改变的量。常量分为字面常量(直接常量)和符号常量。不加任何说明就可直接使用的常量称为字面常量,例 3.1 中语句“c=2 * PI * r;”中的 2 是字面常量;用符号表示的常量称为符号常量,例 3.1 中的 PI 为符号常量。

3.2.1 字面常量

字面常量按类型分为整型常量、浮点型常量、字符型常量、字符串常量和布尔型常量,其值自动地决定了它的数据类型。例如,32 为整型常量中的 int 类型,32.0 为浮点型常量中的 double 类型。

1. 整型常量

整型常量可以采用十进制、八进制、十六进制的形式表示。

(1) 十进制整数：由数字 0～9 组成，是整型常量的默认表示形式，如－15、0、126 等。

(2) 八进制整数：以数字 0 开头，由数字 0～7 组成。例如，0126 表示八进制整数 126，即$(126)_8$，对应十进制数 $1\times8^2+2\times8^1+6\times8^0=86$；－015 表示八进制整数－15，即$(-15)_8$，对应十进制数－13。

(3) 十六进制整数：以 0x(或 0X)开头，由数字 0～9、字母 A～F(或 a～f)组成。例如，0x126 表示十六进制整数 126，即$(126)_{16}$，对应十进制数 $1\times16^2+2\times16^1+6\times16^0=294$；0XFF 表示十六进制整数 FF，即$(FF)_{16}$，对应十进制数 255。

对于整型常量，可以使用后缀 L 或 l 表示长整型数，使用 U 或 u 表示无符号整数。例如，126L、015L、0XFFL 表示长整型数，126U、0XFFU 表示无符号整数，0XFFUL、126LU 表示无符号长整型数。针对没有明确指定为长整型或无符号整型的常量，由编译系统根据值的大小自动识别。例如，取值范围在$-2^{31}\sim+2^{31}-1$(－2 147 483 648～＋2 147 483 647)的数识别为 int，取值范围在 $2^{31}\sim2^{32}-1$(2 147 483 647～4 294 967 295)的数识别为 unsigned long int。

2. 浮点型常量

浮点型常量是指含有小数点或包含有 10 的幂次的实数，也称为浮点数。浮点型常量有以下两种表示形式：

(1) 十进制小数形式：由数字 0～9 和小数点组成。例如，0.123、252.6、.12、－89. 都是合法的浮点型常量。

(2) 指数形式：又称为科学计数法，把浮点型常量用“尾数$\times10^{指数}$”的形式表示，在程序中基数 10 用字母 E(或 e)代替，表示为“尾数 E(e)指数”。例如，浮点数 252.62 可表示为 0.25262E3、2.5262e2。注意，在字母 E(或 e)前后必须有数字，且 E(或 e)之后的指数部分必须是整数。例如，12.3E、E5、1.3e1.2 都不是合法的浮点数。

需要说明的是，浮点型常量在 C++语言编译系统中默认按双精度浮点型(double)处理。只有在实数后面加上 F(或 f)才表示 float 型；在实数后面加上 L(或 l)表示 long double 型。例如：

```
3.14159              //double 型(默认)
3.14159f             //float 型(以 f 结尾)
3.14159L             //long double 型
1.23E5               //double 型(默认)
1.23E5f              //float 型(以 f 结尾)
1.23E5L              //long double 型
```

3. 字符型常量

用英文单引号括起来的单个字符称为字符型常量，如'a'、'A'、'0'、'％'都是合法的字符型常量。字符型常量在计算机内存储时占用 1 字节，实际在存储单元中以二进制的形式存放，存储的是该字符的 ASCII 码值。例如，字符'a'、'A'、'0'被存储在存储单元中的内容为 0110 0001、0100 0001、0011 0000，对应的 ASCII 码值分别为 97、65 与 48。

ASCII 码表中的大小写英文字母、数字等字符可直接用单引号括起来表示成字符型常量，但其中的 32 个控制字符(如回车、退格、换行等)、单引号、双引号、反斜杠(\)等无法用上述方法表示，这些字符需要使用 C++语言提供的“转义字符”形式来表示。

转义字符以转义符(\)开头,有以下几种情况:

(1) 所有的 ASCII 码字符都可以用“\”加一个整型常量来表示,该整型常量必须是一个不超过 3 位的八进制或不超过 2 位的十六进制整数,取值范围为 0～255。例如,'\062'、'\141'、'\x62'、'\X41'都是合法的字符型常量,分别表示字符常量'2'、'a'、'b'、'A'。虽然这种形式的转义序列可表示任意一个字符型常量,但其可读性差。

(2) 常见的但不能显示的 ASCII 字符采用特定字母前加“\”的形式表示,如\v、\t、\n 等。注意,此时“\”后面的字符均不是它本来的 ASCII 字符的意思。

(3) 某些具有特殊含义的字符用转义字符表示。例如,单引号(')是字符常量定界符,表示单引号时需用转义字符'\''。此外,双引号用'\"'表示,“\”字符用'\\'表示。

表 3-2 列出了 C++语言中预定义的常用转义字符及其含义。

表 3-2 C++语言中预定义的常用转义字符及其含义

转义字符	含　义	ASCII 码值(十进制)
\a	响铃(alert)	7
\b	退格(backspace),将当前位置移到前一列	8
\f	走纸(feed)换页,将当前位置移到下页开头	12
\n	换行(newline),将当前位置移到下一行开头	10
\r	回车(return),将当前位置移到本行开头	13
\t	水平制表符(HT)(跳到下一个 Tab 位置)	9
\v	垂直制表符(VT)	11
\\	代表一个反斜线字符''\'	92
\'	代表一个单引号(')字符	39
\"	代表一个双引号(")字符	34
\0	空字符(NULL)	0
\ddd	ASCII 码值为八进制 ddd 的字符	ddd(八进制)
\xhh	ASCII 码值为十六进制 hh 的字符	hh(十六进制)

【例 3.2】 转义字符的应用。

```
#include <iostream>
using namespace std;
int main()
{   cout<<'A'<<'\t'<<'B'<<'\n';
    cout<<'\101'<<'\011'<<'\102'<<'\12';
    cout<<'\x41'<<'\x9'<<'\x42'<<'\xA';
    cout<<'C'<<':'<<'\\'<<endl;
    return 0;
}
```

程序运行结果如下:

```
A       B
A       B
A       B
C:\
```

通过例 3.2 可以发现,前 3 行的输出效果相同。当输出字符 A 时,可以使用'A'、'\101'、

'\x41',显然用'A'可读性好。另外,在输出语句中,经常用'\t'和'\n'等转义字符来调整输出格式。其中,'\t'将当前光标移到下一个输出区(一个输出区占 8 列字符宽度);'\n'相当于按 Enter 键,表示将当前光标移动到下一行行首。要输出"\",则应该在语句中用转义字符'\\'。

4. 字符串常量

用英文双引号(" ")括起来的字符序列称为字符串常量。在存储字符串常量时,编译系统会自动在字符串末尾添加'\0'作为结束标记。编写程序时,通常用这一特性来判断字符串是否结束。

以下给出的都是合法的字符串:

```
"China"                 //字符串长度 5,占 6 字节
"a"                     //字符串长度 1,占 2 字节,注意与字符常量'a'的区别
""                      //空串,长度 0,占 1 字节
"你好"                  //中文字符串,长度 4,占 5 字节(一个汉字占 2 字节)
"a + b = "              //字符串长度 4,占 5 字节
"I\'m a student.\n"     //字符串长度 15,占 16 字节
```

其中,字符串"China"在存储空间的二进制表示形式如图 3-1 所示。

0100 0011	0110 1000	0110 1001	0110 1110	0110 0001	0000 0000

图 3-1　字符串"China"在存储空间的二进制表示形式

【例 3.3】 字符串常量测试示例。

```
#include <iostream>
using namespace std;
int main()
{    cout <<"I\'m a student."<< endl;
     cout <<"字符串长度:"<< strlen("I\'m a student.\n")<< endl;
     cout <<"字符串所占字节数:"<< sizeof("I\'m a student.\n")<< endl;
     return 0;
}
```

程序运行结果如下:

```
I'm a student.
字符串长度:15
字符串所占字节数:16
```

说明:

(1) 字符用单引号作定界符,而字符串用双引号作定界符。

(2) 字符常量只能表示单个字符,而字符串常量中包含零个(空串)或多个字符。

(3) 字符占 1 字节的存储空间,而字符串占的字节数等于字符串的长度加 1。

(4) 例 3.3 中的 sizeof 运算符用于求参数所占的字节数,strlen()函数用于求字符串的长度,具体用法见后续章节。

5. 布尔型常量

布尔型的值只有两个:true 和 false,分别对应逻辑值"真"和"假"。在 C++语言中,布尔

型数据可以参与执行算术运算、逻辑运算和关系运算。在算术运算中，把布尔型数据当作整型数据，true 和 false 分别当作 1 和 0；在逻辑运算中，把非 0 当作 true，把 0 当作 false；在关系运算中，成立的关系值表示为 true，不成立的关系值表示为 false。

3.2.2 符号常量

用标识符表示的常量称为符号常量或标识符常量。在 C++语言中，有两种方法定义符号常量。

(1) 使用编译预处理指令，格式如下：

```
#define 符号常量名 数值
```

(2) 使用常量说明符 const，格式如下：

```
const 数据类型 符号常量名 = 值;
```

例如：

```
#define PI 3.14159
const double PI = 3.14159;
```

两种方法的区别如下：用 #define 定义的符号常量是用一个符号代替一个指定值，在预编译时会把所有符号常量替换为所指定的值(把程序中出现的 PI 替换为 3.14159)。其没有类型，在内存中不存在以符号常量命名的存储单元。而用 const 定义的符号常量有数据类型，内存中存在以它命名的存储单元。

在程序中，通常将频繁使用的常量或具有特殊意义的数值(如上例中的 PI)定义成符号常量，这是一种良好的程序设计习惯。与字面常量相比，使用符号常量增强了程序的可读性，能做到“见名知意”。另外，若要调整程序中某个频繁出现的常量的值，对于字面常量来说要修改多处，很容易遗漏或出错；而对于符号常量来说，只需在定义处进行修改即可。

3.3 变　　量

变量是指在程序执行过程中值可以被改变的量。变量有 3 个基本要素：变量名、类型和值。从计算机系统实现角度来看，变量是用来存储数据的内存区域，变量名是该区域的名称或标识符，区域的大小由编译系统根据变量定义时的数据类型决定，值是存储在该区域的具体内容。

3.3.1 标识符和关键字

在 C++语言中，用来标识变量名、符号常量名、函数名、自定义类型名、类名等名称的有效字符序列称为标识符。简单地说，标识符就是某一实体的名字。C++语言中标识符的构成规则如下：

(1) 以字母或下画线“_”开头。

(2) 由字母、数字、下画线“_”组成，不得使用其他字符。

(3) 区分字母大小写，如 sum、Sum、SUM 是 3 个不同的标识符。

(4) 不能使用关键字,C++语言中的关键字详见表 3-3。

表 3-3 C++语言中的关键字

asm	const_cast	explicit	inline	public	struct	typename
auto	continue	export	int	register	switch	union
bool	default	extern	long	reinterpret_cast	template	unsigned
break	delete	false	mutable	return	this	using
case	do	float	namespace	short	throw	virtual
catch	double	for	new	signed	true	void
char	dynamic_cast	friend	operator	sizeof	try	volatile
class	else	goto	private	static	typedef	wchar_t
const	enum	if	protected	static_cast	typeid	while

关键字是预定义的具有特殊用途的一些名词,也称保留字,表示为 C++语言保留,不能用作标识符。例如,与数据类型有关的关键字有 bool、char、int、long、unsigned、float 和 double 等,与程序流程控制有关的关键字有 if、else、switch、case、default、while、do、for、break、continue 和 return 等,与动态内存管理有关的关键字有 new 和 delete,与类和对象有关的关键字有 class、private、protected、public 和 this 等。表 3-3 中的常用关键字及其用法将在后续章节中介绍。

结合上述规则,可知下面列出的是合法标识符:

```
Sum    area    _name    Max_Value1
```

而下面列出的均为不合法标识符:

```
6ab                  //不能以数字开头
Sum#                 //不能使用除大小写字母、数字、下画线以外的字符"#"
this                 //不能使用关键字
```

C++语言没有规定标识符的长度(字符个数),但各个具体的编译系统都有自己的规定。例如,Visual C++ 2010 规定标识符最长为 2048 个字符。在符合构成规则的前提下,标识符尽量采用简单且有含义的名字(如 student、area、flag、count、num 等),做到“见名知意”,提高程序可读性。

3.3.2 变量的定义

变量遵循“先定义后使用”的原则。定义变量的语句格式如下:

类型名 变量名 1, 变量名 2, 变量名 3, …, 变量名 n;

其中,类型名是变量的数据类型,它可以是 C++语言中预定义的数据类型,也可以是用户自定义的数据类型;变量名的命名需遵循标识符的构造规则。

可以同时定义 n 个相同类型的变量。例如:

```
int num1,num2;          //定义了 2 个 int 型变量
float x,y,z;            //定义了 3 个 float 型变量
double area;            //定义了 1 个 double 型变量
```

在C++语言中,同一变量不能重复定义。变量定义语句可以出现在程序中语句可出现的任何位置,只要在第一次使用该变量之前进行定义即可。但是,从程序可读性方面考虑,通常将变量定义语句放在函数体或程序块(复合语句)的开始处。

3.3.3 变量赋初值

变量首次使用时必须有明确的值,该值称为变量的初值。初值可以是常量,也可以是一个有确定值的表达式。如果未对变量赋初值,则该变量的初值是一个随机值,即该存储单元中的内容是不可知的。可用以下两种方法给变量赋初值。

1. 在定义变量的同时初始化

格式如下:

```
类型名 变量名 = 初值;
类型名 变量名(初值);
```

例如:

```
int a = 1,b = 2,c = 3;              //定义整型变量 a, b, c,并分别将其初始化为 1, 2, 3
char c1 = 'A',c2 = 'a';             //定义字符型变量 c1, c2,并分别将其初始化为'A','a'
double x,y = 12.6;                  //定义双精度型变量 x, y,并只将 y 初始化为 12.6
```

上述语句的等价形式如下:

```
int a(1),b(2),c(3);
char c1('A'),c2('a');
double x,y(12.6);
```

2. 变量定义后使用赋值语句赋初值

例如:

```
double r,area;
r = 3.6;                            //给变量 r 赋初值 3.6
area = 3.14159 * r * r;             //使用有确定值的表达式给变量 area 赋初值
```

3.3.4 变量的使用

变量定义后就可以多次使用。取一个变量的值,称为对变量的引用。对变量的赋值与引用统称为对变量的操作或使用。

3.4 运算符与表达式

运算符是用于描述对数据的操作、体现数据之间运算关系的符号,参与运算的数据称为操作数。根据运算符所需操作数的个数,可以把运算符分为单目运算符、双目运算符和三目运算符;根据运算符的功能,可以把运算符分为算术运算符、关系运算符、逻辑运算符、位运算符等。

表达式是由运算符、圆括号和操作数构成的合法式子,经过运算会得到某种类型的确定

值，其操作数可以是常量、变量或函数等。使用不同的运算符可以构成不同类型的表达式，如算术表达式、赋值表达式、关系表达式和逻辑表达式等。

C++语言提供了十分丰富的运算符，当在一个表达式中出现多种运算符时，其运算顺序需要考虑运算符的优先级和结合性。表 3-4 列出了常用运算符的优先级和同级别运算符的运算顺序，详见附录 B。表 3-4 中优先级别的数字越小，表示优先级越高。

表 3-4　常用运算符及其优先级和结合性

优先级	运　算　符	含　　义	操作数个数	结合性
1	()	圆括号或函数调用		自左向右
	. 、－＞	成员访问符		
	[]	下标运算符		
	::	作用域运算符		
	. ＊、－＞ ＊	成员指针运算符		
	&	引用		
2	＊	取变量运算符	1	自右向左
	&	取指针运算符		
	new	申请动态内存		
	delete	释放动态内存		
	!	逻辑非运算符		
	～	按位取反运算符		
	＋＋	自增运算符		
	－－	自减运算符		
	＋、－	正、负号运算符		
	sizeof	求占用内存字节长度		
3	＊、/、％	乘、除、取余(模运算符)	2	自左向右
4	＋、－	加、减	2	
5	＜＜、＞＞	左移位、右移位	2	
6	＜、＜＝、＞、＞＝	小于、小于或等于、大于、大于或等于	2	
7	＝＝、!＝	等于、不等于	2	
8	&	按位与运算符	2	
9	^	按位异或运算符	2	
10	|	按位或运算符	2	
11	&&	逻辑与运算符	2	
12	||	逻辑或运算符	2	
13	? :	条件运算符	3	
14	＝、＋＝、－＝、＊＝、/＝、％＝、＜＜＝、＞＞＝、&＝、^＝、|＝	赋值运算符、复合赋值运算符	2	自右向左
15	,	逗号运算符		自左向右

1. 优先级

运算符的优先级确定了运算的优先次序，当一个表达式中包含多个运算符时，先执行优先级高的，再执行优先级低的。例如，在表达式 a＊b＋c 中，运算符“＊”的优先级高于“＋”，则先进行 a＊b 运算，再将其结果加上 c。

2. 结合性

如果表达式中同时出现了优先级相同的运算符，则其运算顺序由运算符的结合性决定。运算符的结合性分为左结合和右结合。

(1) 左结合：在优先级相同的情况下，左边的运算符优先与操作数结合起来进行运算，即按从左到右的顺序计算。例如，表达式 a * b%c，先乘后取模。

(2) 右结合：在运算符优先级相同的情况下，右边的运算符优先与操作数结合起来进行运算，即按从右到左的顺序计算。例如，表达式 a=b * =c，先执行 b * =c，再执行 a=b。

值得注意的是，C++语言的运算符很多，其优先级和结合性较复杂，在书写比较复杂的表达式而又对运算次序把握不准时，可通过适当增加圆括号来明确指定表达式的求值顺序。

本章主要介绍算术运算符、关系运算符、逻辑运算符、自增与自减运算符、赋值与复合赋值运算符、sizeof 运算符、逗号运算符，其他运算符将在以后各章中陆续介绍。

3.4.1 算术运算符与算术表达式

1. 算术运算符

算术运算符主要用于数值型数据的算术运算。C++语言中的算术运算符共有 7 个，如表 3-5 所示。

表 3-5 算术运算符

优先次序	算术运算符	含　义	示例(a=7,b=2)	
			表达式	值
1	+	正值运算符，用于取正	+a	7
	−	负值运算符，用于取负	−a	−7
2	*	乘法运算符	a * b	14
	/	除法运算符	a/b	3
	%	取余运算符、模运算符	a%b	1
3	+	加法运算符	a+b	9
	−	减法运算符	a−b	5

说明：

(1) 算术运算符中，+(正值运算符)、−(负值运算符)属于单目运算符，其余均为双目运算符。取正操作的结果与操作数相同，所以正值运算符很少使用；取负操作是取操作数的相反值。

(2) %(取余运算符)用于求两个操作数作除法运算的余数，要求两个操作数均为整型数据，所得余数的符号与左操作数的符号相同。例如：

```
7%2                 //结果为1
-7%2                //结果为-1
7% -2               //结果为1
7.5 %2              //错误,无法正确运行
```

(3) /(除法运算符)用于求两个操作数的商。如果两个操作数均为整型，则作整除运算(去尾取整)；当操作数中至少有一个是浮点型数据时，则作数学中的除法运算。例如：

```
7/2                          //结果为 3
7.0/2                        //结果为 3.5(double 型)
7/2.0                        //结果为 3.5(double 型)
7.0f/2                       //结果为 3.5(float 型 )
```

(4) 在使用算术运算符时,需要注意运算结果的溢出(超出了对应类型的数据表示范围)问题。在除法运算时,若除数为 0 或运算的结果溢出,则系统认为产生了一个严重错误,将终止程序的执行;两个整数作加法、减法或乘法运算时,产生结果溢出时并不认为是一个错误,但这时的计算结果已不正确,编程时要格外注意。例如:

```
#include <iostream>
using namespace std;
int main()
{    int n = 60000;
     cout << n * n << endl;
     short int a = 32767, i = 1;    //short int 数据取值范围为 - 32768～ + 32767
     a = a + i;
     cout << a << endl;
     return 0;
}
```

程序运行结果如下:

```
- 694967296
- 32768
```

以上程序的运行结果显然是错误的,因为 n * n 的值超出了 int 型的数值范围,a+1 超出了 short int 型的数值范围,产生高位溢出。此类问题可以通过改变变量的数据类型来解决。例如,将语句"int n=60000;"改为"unsigned int n=60000;"、将语句"short int a=32767,i=1;"改为"int a=32767,i=1;"便可得到正确结果。

2. 算术表达式

算术表达式是由算术运算符、圆括号和操作数构成的符合 C++语言语法规则的式子。书写算法表达式时应注意:

(1) 所有运算要用指定的运算符表示,不能省略运算符。例如,数学中可以用 ab 表示 a 乘以 b,但在 C++语言中不可以省略运算符"*"。例如,描述数学式$\frac{ab}{2}$时,对应的算术表达式为 a * b/2。必要时,可添加圆括号来描述正确的运算次序。例如,描述数学式$\frac{xy}{2(c+d)}$时,如果只添加运算符,则对应的表达式为 x * y/2 * (c+d),这显然是错误的。此时需要添加圆括号来维持原有数学式的含义,其正确的表达式为 x * y/(2 * (c+d))。

(2) 数学式中特殊含义的值(如圆周率 π)需要写成具体的数值。例如,描述数学式 2πr 时,可以用字面常量 3.14 表示 π,写成 2 * 3.14 * r。在程序中,也可以先定义符号常量再使用,例 3.1 中的语句"const double PI=3.14159;"定义了符号常量 PI,求圆面积时可将表达式写为"PI * r * r"。

(3) 表达式求值时,表达式中的每个变量都应有确定的值。

3.4.2 关系运算符与关系表达式

1. 关系运算符

关系运算符用于比较两个操作数之间的关系是否成立，比较后返回的结果为 bool 型值，当给定关系成立时返回 true，不成立时返回 false。

需要特别注意的是，编译系统处理布尔型数据时，将 true 存储为 1，false 存储为 0。布尔型变量占 1 字节，存储的内容是 1 或 0。

C++语言中有 6 个关系运算符，如表 3-6 所示。

表 3-6 关系运算符

优先次序	关系运算符	含义	示例(a=7，b=2)	
			表达式	值(bool 型)
1	<	小于	a<b	false
	<=	小于或等于	a<=b	false
	>	大于	a>b	true
	>=	大于或等于	a>=b	true
2	==	等于	a==b	false
	!=	不等于	a!=b	true

说明：

(1) 关系运算符都是双目(二元)运算符。关系运算符的优先级低于算术运算符。例如，在表达式 a+b>c+d 中，算术运算符“+”的优先级高于关系运算符“>”，所以先计算 a+b 和 c+d 的值，再进行比较，等价于表达式(a+b)>(c+d)。

(2) 关系运算符可以连续使用，但此时的运算结果与想要表达的含义有可能截然不同，需引起注意。例如，当 a=5、b=4、c=3 时，关系表达式 a>b>c 的返回值为 false，因为运算符“>”的结合方向为从左到右，原表达式等价于(a>b)>c。其先计算 a>b，返回值为 true(实际存储单元中的内容是 1)；再比较 1>3，关系不成立，返回 false(实际存储单元中的内容是 0)。

(3) 描述相等关系时用“==”，不要误写成赋值运算符“=”。

2. 关系表达式

用关系运算符将操作数连接起来的式子称为关系表达式，其中操作数可以是变量、常量、算术表达式、函数等。

【例 3.4】 测试关系表达式的值。

```
#include <iostream>
using namespace std;
int main()
{    int a = 98,b = 66,c = 5;
     cout <<"测试关系表达式的值:";
     cout <<(a > b + c)<<'\t';     //①计算 b + c,得 71; ②a > 71 关系成立,返回 true,输出 1
     cout <<(a > b + 4!= c)<<'\t'; //①计算 b + 4,得 70; ②a > 70 关系成立,结果为 1; ③1!= c 成立,
                                   //输出 1
     cout <<('a' == a)<<'\t'; //字符常量'a'的 ASCII 码为 97,与变量 a 不相等,关系不成立,输出 0
```

```
    cout <<(a < b < c)<< endl;      //①a < b 关系不成立,结果为 0; ②0 < c 关系成立,输出 1
    return 0;
}
```

程序运行结果如下：

```
测试关系表达式的值:1       1       0       1
```

3.4.3 逻辑运算符与逻辑表达式

1. 逻辑运算符

关系表达式只能表示单一条件,但在实际编程中经常需要判断多个关系组合以后的成立情况,此时需要使用逻辑运算符来实现逻辑运算,完成逻辑判断。例如,要判断一个变量是否在给定区间,如判断 x∈(a,b),在 C++语言中不能写成 a＜x＜b,正确的表达式为 x＞a && x＞b,这里用逻辑与运算符描述两个关系同时成立。

C++语言中有 3 种逻辑运算符,即“!”(逻辑非)、“&&”(逻辑与)、“||”(逻辑或)。其中,逻辑非是单目运算符,逻辑与、逻辑或是双目运算符。逻辑运算符与算术运算符、关系运算符的优先级由高到低的顺序如下：

逻辑非(!)→算术运算符→关系运算符→逻辑与(&&)→逻辑或(||)

给定布尔值 a、b,逻辑运算符真值表如表 3-7 所示。

表 3-7　逻辑运算真值表

a	b	!a	a&&b	a||b
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

说明：

(1) 在 C++语言中,原则上逻辑运算符的操作数是布尔型的值,一般以关系运算的结果作为操作数,返回结果为 true 或 false。但事实上,C++语言允许逻辑运算的操作数是其他类型,这种情况下,C++语言进行自动类型转换,将非 0 值转换为 true,0 转换为 false。设 x＝5、y＝6、z＝－2,则表达式 x＜y && z 的返回值为 true。因为 x＜y 成立,返回 true;z 作为逻辑运算的操作数,是非 0 值,转换为 true; true && true 结果为 true。

(2) 逻辑与(&&)、逻辑或(||)可以进行短路求值。这两个运算符具有自左向右的结合性,如果根据运算符左边的计算结果能得到整个表达式的结果,那么运算符右边的表达式就不需要进行计算了,这个规则就是短路求值。根据真值表(表 3-7)不难发现,无论右操作数取何值,false &&(?)的结果一定是 false,而 true||(?)的结果一定是 true,这种情况下不再对右操作数进行计算。也就是说,当逻辑与(&&)的左操作数为 true,逻辑或(||)的左操作数为 false 时,表达式的最终结果由右操作数的值决定,这种情况下才需要计算右操作数的值。

2. 逻辑表达式

用逻辑运算符连接起来的式子称为逻辑表达式。编程时,需要逆转操作数的逻辑状态

时用逻辑非(!),即!true 为 false、!false 为 true;判断两个条件是否同时成立时用逻辑与(&&);需要至少有一个条件成立时用逻辑或(||)。

【例 3.5】 写出判断字符变量 ch 是否为字母的表达式。

分析:字母包含小写字母'a'～'z'及大写字母'A'～'Z',判断表达式为:

```
ch>='a'&&ch<='z'||ch>='A'&&ch<='Z'
```

【例 3.6】 写出判断某一年是否是闰年的表达式。闰年是指年份值能被 4 整除,但不能被 100 整除;或者能被 400 整除。年份用整型变量 year 表示。

分析:判断变量能否被一个数整除,可以用取余运算符(%),如果余数为 0 则表示能被整除,否则表示不能被整除。条件中能被 4 整除可用 year%4==0 表示,不能被 100 整除可用 year%100!=0 表示;能被 400 整除可用 year%400==0 表示。

综上,判断闰年的逻辑表达式如下:

```
(year%4==0 && year %100 != 0) || year%400==0
```

3.4.4 赋值运算符与复合赋值运算符

1. 赋值运算符与赋值表达式

赋值运算符"="的作用是将一个数据赋给一个变量。赋值表达式是由赋值运算符将一个变量和一个表达式连接起来的合法式子,其格式如下:

变量 = 表达式

其功能是将右边表达式的值存放到左边变量对应的内存单元中。注意,赋值运算符的左操作数只能是变量。

一个表达式应该有一个值,赋值表达式也是表达式,一样有值,它的值即为左边变量的值。例如,a=6,表示将 6 赋给变量 a,即变量 a 对应内存单元的值为 6,表达式的值也为 6;赋值表达式"a =6 * 5"的值为 30,执行表达式后,变量 a 的值是 30。

赋值运算符的优先级低于算术运算符、关系运算符和逻辑运算符。多个赋值运算符同时出现时,计算顺序是自右向左。

例如,设 a、b、c 均为整型变量,c 的初值为 5,有赋值表达式 a=b=c+3,其执行过程如图 3-2 所示。

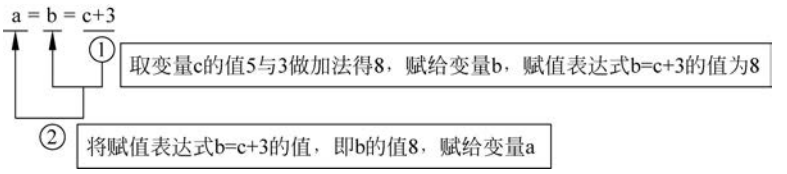

图 3-2 赋值运算执行过程

2. 复合赋值运算符

在 C++语言中,凡是二元(双目)运算符都可以与赋值运算符一起组合成复合赋值运算符。复合赋值运算符共有 10 个:+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=。

复合赋值运算符的优先级和结合性与赋值运算符相同。复合赋值运算符的使用格式

如下：

变量 @= 表达式

等价于：

变量=变量 @（表达式）

其中，@代表复合赋值运算符中的算术或位运算符。

例如：

```
a+=5                    //等同于a=a+5
a*=b+5                  //等同于a=a*(b+5)
a/=b-c*2                //等同于a=a/(b-c*2)
a%=b                    //等同于a=a%b
```

【例 3.7】 设整型变量 a 的初值为 10，执行表达式 a+=a-=a*a 后，求变量 a 的值。

分析：赋值运算符具有右结合性，运算顺序自右向左，表达式等同于

```
a=a+(a-=a*a)
```

其按以下顺序执行：

(1) a-=a*a：等同于 a=a-a*a，即 a=10-10*10，运算后 a=-90，表达式 a-=a*a 的值也为-90。

(2) a=a+(a-=a*a)：变量 a 的值加赋值表达式“a-=a*a”的值，即 a=-90+(-90)=-180，运算后变量 a=-180。

含有复合赋值运算符的表达式也属于赋值表达式。使用复合赋值运算符可简化表达式的书写，提高编译效率，生成的目标代码也较少。

3.4.5 自增运算符与自减运算符

除了复合赋值运算符之外，C++语言还提供了两个使用方便且效率高的运算符：自增(++)和自减(--)运算符，作用分别是使变量的值增 1 或减 1，它们都是单目运算符。自增(++)和自减(--)运算符使用时有前置和后置两种形式，前置是指将运算符置于变量之前，后置是指将运算符置于变量之后。例如：

```
++i;                    //前置,先执行i=i+1,再使用i的值
i++;                    //后置,先使用i的值,再执行i=i+1
--i;                    //前置,先执行i=i-1,再使用i的值
i--;                    //后置,先使用i的值,再执行i=i-1
```

说明：

(1) 当仅需将一个变量值增 1 或减 1，而不参与其他运算时(不与其他运算符同时出现在一个表达式中时)，自增(++)和自减(--)运算符前置和后置的作用相同。例如：

```
int i=6,x,y;++i;x=y=i;    //++前置,运行后结果:i=7,y=7,x=7
int i=6,x,y;i++;x=y=i;    //++后置,运行后结果:i=7,y=7,x=7
```

(2) 当自增(++)或自减(--)运算符与其他运算符出现在同一表达式中时，前置和后置的作用不同。

① ++(或--)前置时,表示先将对应变量的值增 1(或减 1),再使用该变量的值参与表达式的运算。

② ++(或--)后置时,表示先使用该变量的值参与表达式的运算,再将对应变量的值增 1(或减 1)。

例如,有语句"int i=6,x,y;",执行下列语句后的结果是不同的。

```
y = ++i;x = y;          //"y = ++i;"等同于"i = i + 1;y = i;",运行后结果:i = 7,y = 7,x = 7
y = i++;x = y;          //"y = i++;"等同于"y = i;i = i + 1;",运行后结果:y = 6,i = 7,x = 6
```

(3) 自增(++)和自减(--)运算符内包含赋值运算,所以其只能用于变量,不可用于常数或表达式。例如,2++、(x+y)--是非法的。

【例 3.8】 有语句"int i,x=5,y=6;",下列表达式执行后,变量的值分别是多少?

(1) i=++x==6||++y==7;

分析:按优先级关系,该表达式等同于 i=((++x==6)||(++y==7)),逻辑或(||)具有短路求值功能,当逻辑或(||)运算符左侧表达式的值为 true 时,右侧表达式不再执行。

① 逻辑或(||)运算符左侧表达式可以分解为 x=x+1,x==6,x 值为 6,表达式成立,左侧表达式值为 true。

② 因为逻辑或(||)左侧表达式为 true,所以右侧表达式++y==7 未执行,y 的值不变,仍为 6。

③ 执行逻辑或(||)运算,返回 true。

④ 执行赋值运算,i 的值为 1。其具体执行步骤如图 3-3 所示。

因此,上述语句执行后,i=1,x=6,y=6。

例 3.8 程序执行中,借助变量变化示意图(图 3-4)来分析各变量当前的值。列举出各个变量,根据程序的执行流实时更新各变量的值。

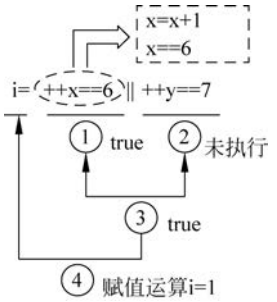

图 3-3 表达式(1)的执行步骤

i	x	y
	5	6
1	6	

图 3-4 表达式(1)变量变化示意图

(2) i=x++==5 && ++y==6;

分析:按优先级关系,该表达式等同于 i=((x++==5)&&(++y==6)),逻辑与(&&)具有短路求值功能,仅当逻辑与(&&)运算符左侧表达式的值为 true 时,才执行右侧表达式。

① 逻辑与(&&)运算符左侧表达式可以分解为 x==5,x=x+1,左侧表达式值为 true,x=6。

② 逻辑与(&&)运算符左侧表达式为 true,继续执行右侧表达式。因为++前置,变量先增 1,再运算,所以其可分解为 y=y+1,y==6,执行后 y=7,右侧表达式的值为 false。

③ 执行逻辑与(&&)运算,返回 false。

④ 执行赋值运算,i 的值为 0。其具体执行步骤如图 3-5 所示。

因此,上述语句执行后,i=0,x=6,y=7,其变量变化如图 3-6 所示。

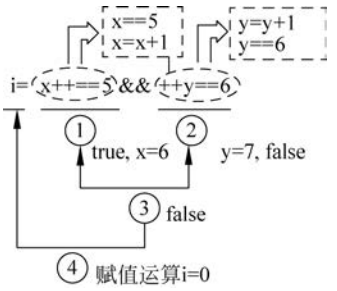

图 3-5 表达式(2)的执行步骤

i	x	y
	5	6
0	6	7

图 3-6 表达式(2)变量变化示意图

3.4.6 逗号运算符与逗号表达式

在 C++语言中,逗号可以作为分隔符,用于在定义语句中将若干变量隔开或调用函数时将若干个参数隔开,如"int i,j,k;"、pow(x,y);逗号也可以作为运算符,称为逗号运算符或顺序求值运算符。逗号运算符的优先级最低,具有自左向右的结合性。用逗号将若干个表达式连接起来构成逗号表达式,其格式如下:

表达式 1, 表达式 2, …, 表达式 n

程序执行时,按从左到右的顺序依次求出各表达式的值,并把最后一个表达式(表达式 n)的值作为整个逗号表达式的值。例如,有定义语句"int a=2,b,c;",则逗号表达式 a+=2,b=3+a,c=a*b 的计算顺序如下:先计算 a+=2,a 的值为 4;再计算 3+a,将其值 5 赋给 b;最后计算 a*b,将其值 20 赋给 c。整个逗号表达式的值也为 20。

3.4.7 sizeof 运算符

在 C++语言中,sizeof 运算符是单目运算符,用于计算操作数的数据类型所占的字节数。操作数可以是数据类型名、变量、常量、表达式。其格式如下:

sizeof(类型说明符|变量名|常量|表达式)

其中,类型说明符可以是预定义的基本数据类型,也可以是用户删除自定义的数据类型。

例如:

```
sizeof( int)                  //值为 4
sizeof( float)                //值为 4
sizeof('\100')                //字符型常量,占 1 字节,值为 1
sizeof('A' + 3.5)             //值为 8,因表达式'A' + 3.5 的结果类型为 double
sizeof( "Hello" )             //值为 6,因字符串"Hello"占 6 字节
sizeof( 3.0)                  //值为 8,因 3.0 是一个 double 型常量
short int x; sizeof(x)        //值为 2
```

3.4.8 条件运算符

条件运算符也称为三目(三元)运算符,是C++语言中唯一要求有3个操作数的运算符。条件运算符能够实现简单的选择功能。条件运算符的使用格式如下:

表达式1?表达式2:表达式3

其中,表达式可以是任意的符合C++语言语法规则的表达式。

条件运算符的运算规则如下:

(1) 计算表达式1。

(2) 如果表达式1的值为true(非0),则求出表达式2的值(不求表达式3的值),并把该值作为整个条件表达式的值。

(3) 如果表达式1的值为false(0),则求出表达式3的值(不求表达式2的值),并把该值作为整个条件表达式的值。

例如,有条件表达式如下:

```
min = a < b?a:b;                          //求两个数中较小的数,并赋值给 min
ch = ch > = 'A' && ch < = 'Z'?ch + 32:ch; //若 ch 是大写字母,则将其转换成小写字母;否则不变
```

条件运算符的优先级较低,仅高于赋值运算符、复合赋值运算符和逗号运算符,具有自右向左的结合性。

3.5 数据类型转换

C++语言支持不同类型数据之间的混合运算。运算时,先进行数据类型转换,转换成相同的数据类型,再进行运算。数据类型的转换有两种方式:自动类型转换和强制类型转换。

3.5.1 自动类型转换

自动类型转换又称隐式类型转换,由编译系统按类型转换规则自动完成。对于非赋值与赋值表达式来说,自动类型转换规则是不同的。

1. 非赋值表达式的类型转换

一般情况下,当算术运算符、关系运算符、逻辑运算符和位运算符两个操作数的数据类型不一致时,将根据数据类型由低到高的顺序进行转换。其顺序如图3-7所示。

图3-7 转换顺序

其转换规则如下:

(1) 当操作数为布尔型(bool)、字符型(char)或短整型(short)时,系统自动转换成整型数(int)参与运算。例如,有语句:

```
short s1 = 1,s2 = 2;
char ch1 = 'a',ch2 = 'b';
```

表达式 s1＋s2 的值为 3，类型为 int；表达式 ch1＋ch2 的值为 195，类型为 int。

（2）当两个操作数的类型不同时，则级别低的类型转换为级别高的类型，运算结果（表达式的值）的类型是表达式的最终类型。例如，表达式 5＋'a'＋6.5－3.65 * 'b'的运算结果为 double 型。

（3）对于整型数据，有符号类型和无符号类型数据进行混合运算，结果为无符号类型。例如，int 类型数据和 unsigned 类型数据的运算结果为 unsigned 类型，unsigned int 类型数据和 long 类型数据的运算结果为 unsigned long 类型。例如，有语句：

```
int i1 = -6;
unsigned int i2 = 65535;
long i3 = 5;
unsigned long int i4 = 4;
float f = 6;
```

则表达式 i1＋i2 的类型为 unsigned int，i2＋i3 的类型为 unsigned long int，i4＋f 的类型为 float 型。

2. 赋值表达式的类型转换

在赋值表达式中，当赋值号两边的数据类型不同但类型兼容时，则将赋值号右边量的类型转换为左边变量的类型。

其转换规则如下：

（1）将浮点型数据赋值给整型变量时，仅取其整数部分赋给整型变量；若整数部分的值超过整型变量的取值范围，则赋值的结果错误。例如：

```
double a = 65536.7;
int b = a;                  //赋值后变量 b 的值为 65536
short c = a;                //赋值后变量 c 的值为 0,赋值结果错误
```

（2）将整型数据赋给浮点型变量时，将整型数据转换成浮点型数据后，再赋值。

（3）将 double 型数据赋给 float 型变量时，要注意数值范围，溢出时赋值出错。

（4）将少字节整型数据赋值给多字节整型变量时，采用符号位扩展的方式，即低字节部分一一对应，高字节部分按少字节数据的符号位进行扩展。图 3-8 给出了 2 字节整型数据向 4 字节扩展的示例。

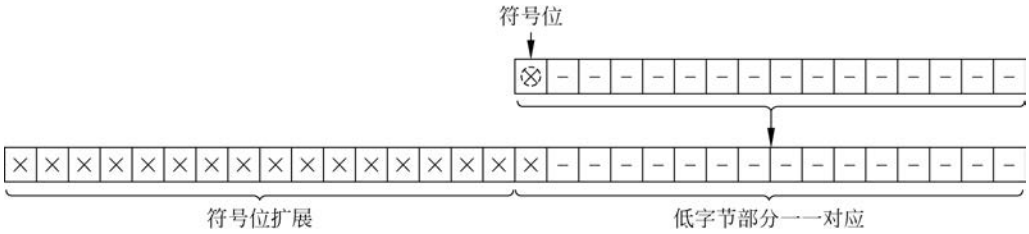

图 3-8　符号位扩展示例

例如：

```
short a = -1;
int b = a;          //赋值后 b 的值为 -1
```

（5）将多字节数据赋值给少字节变量时，则将多字节的低位部分一一赋值，高位字节部分舍去。例如：

```
int a1 = 65536;   //变量 a1 在内存中为 0000 0000 0000 0001 0000 0000 0000 0000
short b1 = a1;    //取变量 a1 的低 2 字节赋值给变量 b1,值为 0
int a2 = 65535;   //变量 a2 在内存中为 0000 0000 0000 0000 1111 1111 1111 1111
short b2 = a2; //取变量 a2 的低 2 字节赋值给变量 b2,值为 - 1( - 1 的补码为 1111 1111 1111 1111)
```

图 3-9 所示为 4 字节整型数据赋值给 2 字节变量。

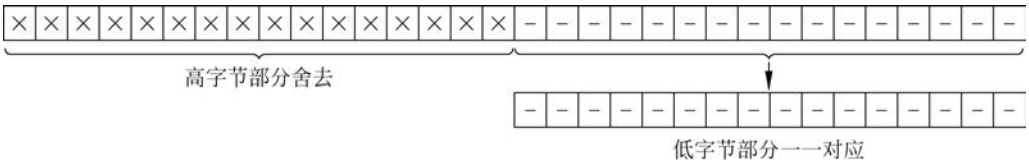

图 3-9　4 字节整型数据赋值给 2 字节变量

例如：

```
int a = 256;              //变量 a 在内存中为 0000 0000 0000 0000 0000 0001 0000 0000
char ch = a;              //取变量 a 的低 1 字节赋值给变量 ch,值为 0
short a1 = 353;           //变量 a1 在内存中为 0000 0001 0110 0001
char ch1 = a;             //取变量 a1 的低 1 字节赋值给变量 ch1,值为 97,即字符'a'
```

图 3-10 所示为 4 字节整型数据赋值给 char 型变量。

图 3-10　4 字节整型数据赋值给 char 型变量

(6) 将无符号字符型数据赋给整型变量时，字符型数据放在整型变量的低位字节，高位字节补 0；但将有符号字符型数据赋给整型变量时，将其放到整型变量的低位字节，高位字节扩展符号位。例如：

```
unsigned char ch = 255; //变量 ch 在内存中为 1111 1111,最高位的 1 不表示符号
short int a = ch;       //无符号字符型赋值给整型变量时,高位字节补 0,即 0000 0000 1111 1111,
                        //值为 255
char ch1 = 255;         //变量 ch1 在内存中为 1111 1111,其实际值为 - 1,最高位 1 为符号位
short int a1 = ch1;     //高位字节扩展符号位后为 1111 1111 1111 1111,值为 - 1
```

(7) 将无符号整型或长整型数据赋给整型变量时，若在整型的取值范围内，则不会产生错误；而当超出其取值范围时，则赋值的结果错误。

3.5.2　强制类型转换

强制类型转换也称显式类型转换，作用是将某种类型强制转换成指定的数据类型。其语法格式如下：

类型说明符(表达式)

或

(类型说明符) 表达式

例如：

```
double(a)                    //将变量 a 转换成 double 类型的中间数据
(int)(a + b)                 //将 a + b 运算的结果转换成 int 型
5/float(3)                   //将 3 转换成 float 型
```

一个变量在强制类型转换后，得到一个指定类型的中间变量，但该变量本身不会改变，还是原来的数据类型。

【例 3.9】 强制类型转换。

```
#include <iostream>
using namespace std;
int main()
{    float y = 6.8;
     int x;
     x = (int)y % 2;          //y 的值与类型均不变，生成中间值 6，取余后得 0，赋值给 x
     cout <<"x = "<< x <<'\t'<<"y = "<< y <<'\n';
     return 0;
}
```

程序运行结果如下：

```
x = 0     y = 6.8
```

例 3.9 中的变量 y 为 float 类型，而取余运算符(%)要求两个操作数均为整型，这种情况下若要正确完成运算，需要用强制类型转换符。

3.6 常用库函数

C++语言分门别类地提供了多个标准函数库和面向对象类库(简称库)，如常用的数学函数库 cmath、字符串处理函数库 cstring、输入/输出类库 iostream 等。每个库中提供了大量的函数，若在编程时需要使用库中的函数，只需在程序开头用编译预处理命令 #include 包含相关库的头文件即可。例如，求某个正整数的平方根，使用 C++语言提供的函数 sqrt()，则需要包含数学函数库的头文件 cmath，包含头文件的编译预处理命令为 #include < cmath >。

表 3-8 C++语言常用的库函数

C++头文件	类别	函数原型	功能简述
cmath	开平方根函数	double sqrt(double x)	求 x 的平方根
	取绝对值函数	int abs(int x)	别求整型数、长整型数、浮点数的绝对值
		long labs(long x)	
		double fabs(double x)	
	三角函数	double sin(double x)	分别求 x 的正弦、余弦、正切值，x 为弧度值。例如，求 sin(30°)时应把度数转换为弧度，可表示为 sin(3.14159 * 30/180)
		double cos(double x)	
		double tan(double x)	
		double asin (double x)	分别求 x 的反正弦、反余弦、反正切值
		double acos (double x)	
		double atan(double x)	

续表

C++头文件	类别	函数原型	功能简述
cmath	指数函数	double pow(double x,double y)	求 x^y，即 x 的 y 次幂
		double exp(double x)	求 e^x，即 e 的 x 次幂
	对数函数	double log(double x)	求 ln(x)
		double log10(double x)	求 $\log_{10}^x$
ctime	时间函数	time_t time(time_t * tloc)	返回从 1970 年 1 月 1 日至今所经历的时间(以 s 为单位)
cstdlib	伪随机函数	void srand(unsigned seed)	初始化伪随机数发生器。若每次的 seed 相同，则产生的伪随机数序列也相同。可配合 time()函数生成不同的 seed，如 srand(time(NULL))
		int rand()	产生 0～0x7fff 范围内的伪随机数。例如，rand()%100 产生[0,99]的随机数，rand()%(b－a＋1)＋a 产生[a,b]的随机数

为了便于函数的使用，通常将函数的主要特征用固定的格式来描述，这种固定的格式称为函数原型(具体内容在后续函数章节讲述)。如图 3-11 所示，以计算并返回指定参数的平方根的 sqrt()函数为例，其函数原型如下：

```
double sqrt(double x);
```

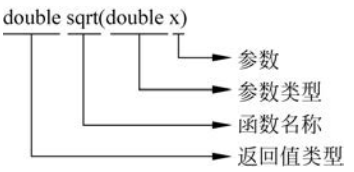

图 3-11　sqrt 函数原型

使用函数时，写下函数名称后，在其后加一对圆括号，括号内根据函数原型写下该函数所需的参数(多个参数用逗号分隔)，参数可以是常量、变量或表达式。例如，$\sqrt{6}$ 可以使用 sqrt(6)来描述，$\sqrt{(20a+5.0)}$ 可以使用 sqrt(20 * a＋5.0)来描述。

表 3-8 给出了部分常用的库函数，读者若需要了解更多的函数，可查阅对应的头文件，详见附录 C。

习　　题

一、选择题

1. 下列符号中可以定义为用户标识符的是_____。

A. float　　B. -ad　　C. _56　　D. 5_a

2. 下列符号中不可作为用户标识符的是_____。

A. Main　　B. _int　　C. sizeof　　D. abc

3. 下列选项中，正确的 C++语言标识符是_____。

A. 6_ _sum　　B. sum～6　　C. sum.6　　D. _sum_6

4. 下列选项中可以作为 C++语言中合法常量的是_____。

A. 80　　B. 080　　C. －80e2.0　　D. －80e

5. 下列选项中可以作为 C++语言中合法常量的是_____。

A. '\85'　　B. 0x80　　C. '\x102'　　D. '\085'

6. 下列选项中不是 C++语言中合法常量的是_____。

A. 0789　　B. 123UL　　C. 0XFF　　D. 1E－2

7. 下列为字符型变量赋初值的语句中不正确的是_____。

A. char a='a';　　B. char a="3";　　C. char a= 97;　　D. char a=0x61;

8. 已知“float x=5.6,y=5.2;”，则表达式(int)x+y 的值为_____。

A. 10.2　　B. 10　　C. 10.8　　D. 11.2

9. 若有语句“int i=3, j=2;”，则表达式 i/j+i%j 的结果是_____。

A. 1.5　　B. 3　　C. 2　　D. 2.5

10. 表达式(int)((float)9/2)－9%2 的值是_____。

A. 0　　B. 4　　C. 3　　D. 5

11. 下列运算符要求操作数必须为整型的是_____。

A. /　　B. ++　　C. !=　　D. %

12. 若变量已正确定义并具有初值，则下列表达式正确的是_____。

A. x=y%1.2　　B. x+y=z　　C. x=y++=z　　D. x=2+(y=z=2)

13. 转义字符是以_____开头的。

A. %　　B. &　　C. \　　D. '

14. 设“int m=1,n=2;”，则表达式 m++==n 的结果是_____。

A. false　　B. true　　C. 3　　D. 2

15. 设“int m=1,n=2;”，则表达式++m==n 的结果是_____。

A. false　　B. true　　C. 3　　D. 2

16. 下列程序段执行后的输出结果是_____。

```
int a, b, c = 246;
a = c/100 % 9;
b = ( - 1)&&( - 1);
cout << a <<','<< b << endl;
```

A. 2,1　　B. 3,2　　C. 4,3　　D. 2,3

二、填空题

1. 设有“int a=5,b=4;”，则逻辑表达式 a<=6 && a+b>8 的值为_____。

2. 设有“int a=10,b=3;”，则执行表达式 b%=b++||a++后，a=_____、b=_____。

3. 设有“int a=6,b=7;”，则执行表达式 a+=a－b 后，a=_____、b=_____。

4. 设有“int a=35,b=6;”，则表达式!a+a%b 的值是_____。

5. 设有“int x=1,y=2;”，则执行表达式“(x>y)&&(－－x>0);”后 x 的值

为_____。

6. 若有定义“int i=7; float x=3.1416; double y=3;”,则表达式 i+'a' * x+i/y 值的类型是_____。

7. 下列程序的运行结果是_____。

```
#include <iostream>
using namespace std;
int main()
{    int a,b,c;
     a = b = 1;
     c = a++,b++,++b;
     cout << a <<'\t'<< b <<'\t'<< c <<'\t'<< endl;
     return 0;
}
```

8. 将下列数学式写成 C++语言的表达式。

(1) (a−b)(a+b)。

(2) $\frac{-b+\sqrt{b^2-4ac}}{2a}$。

(3) $\frac{\sin x}{|x-y|}$。

(4) $\sqrt{s(s-a)(s-b)(s-c)}$。

9. 写出表示下列条件的 C++语言表达式。

(1) a 和 b 至少有一个是偶数。

(2) a 是 3 的倍数,b 是 5 的倍数。

(3) $0 \leqslant x < 100$。

(4) a、b、c 构成三角形的条件。

三、编程题

1. 从键盘输入一个 3 位正整数 n,求出它的各位数字之和 sum 并输出。

2. 用伪随机函数 rand()分别产生一个两位正整数和一个在[−10,10]区间内的整数并输出。