第 1 章　C++语言概述

1.1　C++语言的起源和特点

1.1.1　C++语言的起源

正如从名字上可以猜测到的一样，C++语言是从 C 语言继承来的，但这种继承主要只是表现在语句形式、模块化程序设计等方面。如果从更重要的方面——概念和思想方面来看，C++源于早期的 SIMULA 语言，因为 C++语言的最大特征是支持“面向对象的程序设计”(面向对象的程序设计的概念见 1.1.2 节)。SIMULA 语言被广泛地用于系统仿真，设计它的主要目的是模仿现实世界的真实个体，而使用的主要手段是构造计算机领域的对象来表述现实的客体。由于 SIMULA 语言的应用领域并不十分广阔，更重要的一点是它缺乏强有力的开发工具支持，它并没有得到很大的重视。随后推出的另外一种面向对象语言 SMALLTALK 也没有取得太大的成功，很多人认为它没有提供给自己足够的灵活性和如同 C 或 BASIC 语言那样丰富的功能，最关键还在于它和人们早已得心应手的语言并不兼容。比如说，一个 C 程序员可能会对它的新特性退避三舍，因为 C 的特性对他是十分熟悉和亲切的，同时 C 的确是功能强大的，大多数人不愿放弃这些。

C++的产生正是为了解开这样的一个“情结”。面对越来越大，越来越复杂的系统，使用 C 语言已经感到力不从心了，但 C 语言作为应用域最为广泛的程序设计语言之一，又不能轻易放弃。必须有一种面向对象的程序设计语言，它对 C 语言有很高的兼容性，使得 C 程序员只需在原有的知识上进行一定的扩充，就能够方便地进行面向对象的程序设计。

1980 年起，Bell 实验室的 Bjarne Stroutstrup 博士及其同事开始为这个目标对 C 语言进行改进和扩充。由于这种被扩充和改进的 C 语言的大量特性与类(class)相关，它最初被开发者称为“带类的 C”。但很快人们就认识到这个称呼太片面了，这个“扩展了的 C”不仅以标准 ANSI C 作为子集保留了 C 语言的全部精华，同时又吸收了 SIMULA 67，ALGOL 68 和 BCPL 语言的许多特性，它已远远超过了 C 语言。随着这个语言的广泛应用和在各个领域取得成果的增多，它给程序设计带来的全新概念和表现出来的远大前景更加卓著，它的开发者因此将 C++这一名字赋与它。

1.1.2　C++语言的特点

1. C++是面向对象的程序设计语言

与过去的面向过程的程序设计语言比较，C++的最大特征在于它是面向对象的程序设计语言。所谓对象是现实世界中的实体，例如桌子、电视接收机、张三等等。具有共同行为和特征的实体的集合，可以被归纳成一类，因此每个对象都是属于某个类的对象，例

如，人是一个类，而每一个具体的人则是人这个类中的一个对象。面向对象的程序设计是程序设计的一种新思想，该思想认为程序是相互联系的离散对象的集合。面向对象的程序设计语言即是支持这种思想的程序设计语言。

2. 封装性

C++的封装性，是通过引入“类”而产生的，类将一定数据和关于这些数据的操作封装在一起。这个特点可以显著减少程序各模块之间的不良影响，这在多人协作性的程序开发中，好处尤为明显。

3. 继承性

C++的继承性，是指C++原有程序的代码可以方便地移植到C++的新程序中，而新程序在继承旧程序代码的同时可以增添自己的新内容。继承性使得程序代码的重用率得以很大提高，使得系统开发过程具有更好的连续性，易于应付用户对于软件不断发展的要求。

4. 多态性

C++的多态性，是指相似而实质不同的操作可以有相同的名称。例如，“和”的操作，可以是“整数和”也可以是“矢量和”，在C++中，这两种和的操作都可以简单地称为“和”。C++的多态性使得C++与人的思维习惯更趋一致，用C++编制的程序也更方便人的阅读。

1.2 C++语言与C语言的关系

1.2.1 C++语言与C语言的联系

C语言也诞生在AT&T的Bell实验室，1972年由Dennis Richie为UNIX设计了这个高级语言，今天C语言的使用已遍及到计算机的各个领域。

C语言有以下几个显著的特点：

第一，它是一种结构化语言，要求一个程序由众多的函数组成，程序的逻辑结构由顺序、选择和循环三种基本结构组成，适宜于大型程序的模块化设计。

第二，它可以部分取代汇编语言，同时具有很高的可移植性，这使得C语言程序在保证支持不同硬件环境的前提下，具有较高的代码效率。

第三，它提供了丰富的数据类型和运算，具有较强的数据表达能力，因而在许多不同的场合广泛应用。

总之，C语言反映了设计者追求高效、灵活，支持模块化设计，从而支持大规模软件开发的愿望。

C++语言保留了C语言设计者的良好愿望，并使得C语言语句成为C++语言的一个子集。一般，用C语言编写的程序可直接在C++编译器下编译。

1.2.2 C++语言与C语言的主要区别

首先，C++提出了类(class)的概念。类是数据和函数的集合，数据用来描述类所属

对象的状态，函数用来描述此类对象的行为。例如，大学生代表在大学读书的一类人，即大学生是一个类，每个具体的大学生都是这个类中的对象。大学生这个类中的数据可以是学生的姓名、性别、年龄、学校、专业、入学时间等等，描述此类对象的行为可以是入学、改换专业、毕业等。

C语言中的结构只是数据的集合，这种结构也可在C++语言中使用。不同的是C++语言将C语言中的“结构”概念扩充成近似于上述“类”的概念，即C++语言中的结构既可以有数据，也可以有函数。

C++语言沿用了C语言中的结构，概念上没有变化。

其次，下列关键字是C++语言新增的：

class，private，protected，public，this，new，delete，friend，operate，inline，virtual。

1.2.3 C++语言与C语言的细小区别

为较完整地讲述C++语言与C语言的关系，在此预先介绍它们之间的细微差别，读者可以在读完本书各章节后，再来阅读这部分。

1. C++语言在保留C语言原有注释方式的同时，增加了行注释。以“//”起始的，以换行符结束的部分是行注释。

2. const 关键字，用这个关键字修饰的标识符为恒值常量。它的引入可以替代C语言中的恒定义。比较下面两个语句：

```
#define Number 1
const int Number=1;
```

它们的功能相同，但后一语句在编译时，编译器对于用到 Number 的地方，将进行严格的类型检查。

3. 说明结构、联合和枚举变量时，不必在结构名、联合名和枚举名前加关键字 struct，union 和 enum 。例如：

```
/* C语言的说明 */
struct AStruct aS;
union  AUnion  aU;
enum   Bool   aBool;
// C++语言的说明
AStruct aS;
AUnion  aU;
Bool    aBool;
```

4. 变量的说明可以放在程序的任一位置上，例如：

```
for ( int i=0; i<100; i++ )
```

5. 提供了作用域运算符“::”，当有某一全局变量被一个局部变量遮挡时，运用作用域运算符仍可以操作该全局变量。例如：

```
...
int i;
```

```
main()
{ int i;
   i=5;        // 局部变量 i 赋值
   ::i=10;     // 全局变量 i 赋值
   ...
}
```

6. 标准输入输出一般不再使用C语言的 printf 和 scanf，而使用三个标准I/O流。它们是：cout(与标准输出设备相联)，cin(与标准输入设备相联)和 cerr(与标准错误输出设备相联)。在微机上，各设备一般分别为显示器、键盘和显示器。“<<”和“>>”分别被重定义为流的插入和提取操作，例如：

```
cout<<"Welcome!";              // 向显示器输出“Welcome!”
cin>>a;                        // 从键盘取得数据至 a
cerr<<"There is an error";     // 在显示器上立即显示错误信息
```

1.3 C++ 展 望

自从 1983 年 AT&T 的 Bell 实验室推出了它的C++标准之后，仅仅经历了短短的10年，C++的应用已广泛地深入到计算机技术的各个领域，并且取得了很大的成功。对象的概念越来越多地被人们所应用，例如，新一代具有革命性的微机和工作站操作系统 WINDOWS NT 就大量运用了这一概念。

面对愈来愈复杂的系统，愈来愈大型的软件，面向对象的程序设计从一个新的角度出发，力图通过使问题空间和解题空间保持更为有效的一致性，使得计算机软件更加有效和易于理解。作为一种成功的面向对象程序设计语言，C++是人们对于客观世界进行清晰和准确描述的有效手段，C++使得软件开发者能方便地仿真客观世界。

所以我们可以想见，今天的C++正如当年C语言一样，会给软件设计带来一次新的巨大进步，而它的应用会迅速地普及到计算机技术的各个领域，成为新的更为有效的工具。

需要指出的是，C++只是作为面向对象的程序设计方法的一种实现，在当今还有很多其它的程序设计语言也能支持面向对象程序设计，例如面向对象的 PASCAL 语言，以及前面已经提到的 SMALLTALK 语言等。因为面向对象方法学的提出是程序设计思想的革命，很多的专家学者都在为使得这种新思想的更完美实现而努力，不仅仅是软件在进行巨大的变革，计算机硬件也受到这个思想的影响而发生了变化。从对C++语言的以往使用和与相关语言的比较中得到的体会，我们认为它将在面向对象的程序设计语言中成为主流。

习 题

1. 简述C++语言的特点

第 2 章　数据类型和表达式

从某种角度看，程序是由数据和处理数据的操作组成的信息流。在介绍基本数据类型之前，我们先来看看C++的词法符号。

2.1　C++的词法符号

词法符号是程序中不可再分的最小单位，相当于我们自然语言中的单词。C++共有五种类型的词法符号，它们分别是关键字、标识符、常量、运算符以及标点符号。本节仅介绍关键字、标识符以及标点符号，常量及运算符将在后面的章节介绍。

2.1.1　关键字

关键字在计算机语言中有预定的含义。关键字又称为保留字，它们在C++语言中已被定义了特定的内容，因而不能再被用户重新定义而赋予新的内容。C++的关健字有：

asm	auto	break	case	char	class	const
continue	default	delete	do	double	else	enum
extern	float	for	friend	goto	if	inline
long	new	operator	protected	public	register	return
short	signed	sizeof	static	struct	switch	template
this	typedef	union	unsigned	virtual	void	while

它们的含义将在本书的以后章节中陆续介绍。

2.1.2　标识符

标识符是由程序员定义的名字，可以用作变量名、函数名、类名等。标识符由大小写英文字母、阿拉伯数字 0 ～ 9 和下划线组成。标识符的第一个字符规定必须是英文字母或下划线。

例如下面的标识符是合法的：

name，File，_error，err32，f8l_I3

而下面的标识符不合法：

2r，M＄100

有些语言对于标识符中英文字母的大小写不加区别，但C++则区别大小写。C++对于标识符的最大长度没有特别限制，它取决于所使用的编译器，如果标识符的长度超过具体编译器所能识别的最大长度，则编译器忽略多余的字符。在开始编制程序以前，通过查阅编译器手册可以确定这个最大长度。

另外，C++编译系统本身所使用的标识符都是以下划线开始的，为防止与系统发生

此为试读，需要完整PDF请访问：www.ertongbook.com

冲突，最好不要定义以下划线开始的标识符。如果一定有这样的用途，那么查阅编译器的运行库手册检查是否有冲突。

2.1.3 标点符号

C++的标点符号只起语法作用，不表示任何实际的操作。C++的标点符号有：

( ) { } , ; … ' "

它们的意义和用途将在本书的后续章节中介绍。

2.2 基本数据类型、变量与变量说明

程序中所处理的数据都具有一个特定的类型，类型规定了数据在内存中的存储格式，编译器通过它来决定每一个数据在内存中如何放置，任何变量在使用之前都必须说明其数据类型。描述类型的标识符称为类型名，类型名既可以是关键字，也可以是用户定义的标识符。C++预定义了一些基本的数据类型，其它的数据类型都可以视为基本数据类型的派生和发展，它们通过用户的定义产生。表 2-1 列出了 C++的基本数据类型。其中的字节数和值的范围特指 IBM-PC 系列微机而言(下同)。

表 2-1

名 称	类 型 名	字节数	值 的 范 围
字 符	char	1	−128 ～ 127
整 数	int	2	−32768 ～ 32767
浮 点	float	4	(+/−)3.4E−38 ～ (+/−)3.4E+38
双精度	double	8	(+/−)1.7E−308 ～ (+/−)1.7E+308
空	void	0	无值

除 void 类型外，其它的基本数据类型之前都可以加上一定的辅助类型修饰符，它们在一定程度上改变了基本数据类型的含义，给予基本数据类型一些特定的约束。C++的类型修饰符有：signed(有符号的)，unsigned(无符号的)，short(短的)，long(长的)。signed 和 unsigned 用于修饰字符类型和整数类型，short 和 long 用于修饰整数类型，long 还可用于修饰双精度类型。基本数据类型和修饰符的各种组合情况如表 2-2 表所示：

表 2-2

类 型 名	字节数	值 的 范 围
signed char	1	−128 ～ 127
unsigned char	1	0 ～ 255
short int	2	−32768 ～ 32767
signed short int	2	−32768 ～ 32767
unsigned short int	2	0 ～ 65535
signed int	2	−32768 ～ 32767

续表 2-2

类 型 名	字节数	值 的 范 围
unsigned int	2	0 ～ 65535
long int	4	－2147483648 ～ 2147483647
signed long int	4	－2147483648 ～ 2147483647
unsigned long int	4	0 ～ 4294967295
long double	10	(＋/－)3.4E－4932 ～ 1.1E＋4932

当有类型修饰符修饰 int 时，int 可省略，例如 long int 等效于 long 。

变量是程序运行中可以变化的量。类型名用来说明变量的数据类型，变量名用来标识一个特定的变量，它的说明格式为：

类型名　〈变量名，…，〉变量名；

例如：

```
int     i;              //    说明  i  为整数变量。
char ch;                //    说明  ch  为字符变量。
float x,y,z;            //    说明  x,y,z  都为浮点变量。
unsigned char a,b,c;    //    说明  a,b,c  都为无符号字符变量。
long length;            //    说明  length  为长整数变量。
```

与变量说明相关的一个关键字是 typedef，它用于用户定义一个类型，但要注意这些类型都是在 C＋＋标准类型的基础上定义的，即它只是为一些 C＋＋的类型提供更适合于用户的别名。它的形式如下：

typedef C＋＋定义类型 用户定义类型名称；

它的引入是为了使一些类型的说明简洁而明白，例如若需要一个整型变量来存放标识号，那么用下面的方法比用 int 要好一些：

```
typedef int ID_NUMBER;
ID_NUMBER StudentNumber;
```

2.3 常　　量

常量是用来表示固定数值(包括字符值)的词法符号。在程序中，常量一旦定义，其值就不能改变。常量都具有一定的数据类型。常量有两种——数值常量和 const 常量。

2.3.1 数值常量

1. 整数数值常量

整数数值常量即是整数，可以用十进制、八进制和十六进制表示。十进制不能以 0 开头，八进制必须以 0 开头，十六进制必须以 0x (或 0X)开头。例如：

193，－193，0765，－0765，0x34，－0x34，0X54，－0X54

任一整型数值常量可带字母 u(或 U)，l(或 L)，ul(或 lu 等)。u(或 U)表示 unsigned，

l（或 L）表示 long ，ul（或 lU 等）表示 unsigned long。例如：

2345756l（长整数 2345756），　　－6L（长整数－6）

65423uL（无符号长整数 65423），　04324u（无符号八进制整数 4324）

2. 浮点数值常量

浮点数值常量由整数部分和小数部分组成，允许科学计数法。浮点数值常量前面可带一个负号表示负数，尾部必须跟字母 f（或 F），浮点数值常量只采用十进制表示。例如：

23.5f，－23.5F，0.32E＋4f，－3.4e－3f

3. 双精度数值常量

双精度数值常量基本同浮点数值常量，只是末尾不能有 F（或 f）等字母，例如：

23.5，－23.5，0.32E＋4，－3.4e－3

4. 长双精度数值常量

基本同浮点数值常量，只是末尾所跟字母是 L（或 l）而不是 F（或 f），例如：

23.5L，－23.5l，0.32e＋4l，－3.4E－3L

5. 字符数值常量

由引号括起来的单个字符称为字符常量，例如："A"，"＃"，"＋?，' '（空格）。注意，不能有 "\"的字符数值常量。反斜杠被用作转意符，它与一些字母组合，可组成单个的字符，用来表示一些特殊的含义。例如：

\a	响铃	\r	回车符
\b	回退符（backspace 键）	\t	制表符（tab 键）
\f	进页符	\v	垂直制表符
\n	换行符		

要表示单引号、双引号、反斜杠，须表示成"\'"，"\"""，"\\"。另外，也可用反斜杠与八进制整数或十六进制整数的组合来表示 ASCII 字符集中的任意字符，例如：

"\050" 表示 "("，"\x41"表示 "A"等。

6. 字符串数值常量

字符串数值常量是由双引号括起来的字符序列，例如：" C＋＋ is a fun! "。可以简单地将字符串数值常量称为字符串常量或字符串。

2.3.2 const 常量

使用关键字 const 也是定义常量的一种方法，定义格式为：

const 数据类型 变量 ＝ 表达式；

例如：

```
const float f = 0.5f;
```

其中表达式的含义后面再述。上面语句使得编译器给该变量 f 拥有一个存储单元，存储单元的值即为表达式的值，该单元在变量 f 的有效期内，只允许读而不允许写，从而变量 f 成为一常量。编译器对它的处理与对变量的处理十分类似，但对企图修改它的任何语句都加以阻止，它相对于数值常量的优点在于易于修改，并且具有较好的安全性。

2.4 运算符与表达式

运算是对数据进行加工的过程，记述各种运算的符号称为运算符，而参与运算的数据称为操作数。仅对一个操作数作用的运算符称为一元运算符，对两个操作数作用的运算符称为二元运算符，对三个操作数作用的运算符称为三元运算符。

表达式是对一系列操作数及其运算的表述，例如小学课本中的四则混合运算 a+b*c-d 即为一表达式，其中 a,b,c,d 均为操作数，+,-,* 均为运算符。表达式的值及其类型与表达式的运算过程有关。当一表达式中存在有多种运算符时，为保证确定的运算顺序，必须规定运算符之间的优先顺序，即各运算符相对于其它运算符具有的一定优先级别。

C++的运算符可以分成以下几类：

(1) 算术运算符	+,-,*,/,%
(2) 关系运算符	>,<,<=,>=,==,!=
(3) 逻辑运算符	!,&&,||
(4) 位运算符	<<,>>,~,|,^,&
(5) 条件运算符	?:
(6) 逗号运算符	,
(7) 求字节数运算符	sizeof
(8) 赋值运算符	=,+=,-=,*=,/=,%=,<<=,>>=,&=,^=,|=
(9) 强制类型转换运算符	类型名
(10) 指针运算符	*,&
(11) 分量运算符	.,->
(12) 下标运算符	[]
(13) 其它	++,-- 等

本章仅讲述前面九种运算符。

2.4.1 算术运算符

算术运算符的含义与在一般算术运算中的含义大致相同。

“+”作为二元运算符，表示两个操作数的相加，例如 a+b。

“-”作为二元运算符，表示两个操作数的相减，在“-”之前的操作数称为被减数，在“-”之后的操作数称为减数，例如 a-b；“-”运算符亦可作为一元运算符，表示一个操作数的相反数，例如 -a。

“*”作为二元运算符，表示两个操作数的相乘，例如 a*b。

“/”作为二元运算符，表示两个操作数的相除，与“-”相似，在“/”之前的操作数称为被除数，在“/”之后的操作数称为除数，例如 a/b 表示 a÷b。仅当两个操作数均为整数时，“/”表示整除，例如 10/7=1，整除的结果为：先求被除数与除数绝对值的商，再将该商向

下取整，在所得到的整数上再冠以原来商的正/负符号，即得到原来两个整数整除的结果。例如，$-10/7=-1$ 。

“%”作为二元运算符，表示两个整数的模除，即 a%b 的结果是 a 被 b 除所得的余数，例如 $10\%7 = 3, -5\%7 = -5$。

2.4.2 关系运算符

所有的关系运算符都是二元运算符，各运算符的含义如下：

>（大于）　　<（小于）　　>=（大于或等于）

<=（小于或等于）　==（等于）　　!=（不等于）

关系运算符的结果为 int 类型，当操作数满足关系运算符所要求的比较关系时，结果为 1，表示真，否则为 0，表示假。（在 C++中的 int 型值，只要其不为 0，均表示真）。

2.4.3 逻辑运算符

逻辑运算符共有三个，它们是 &&（逻辑与），||（逻辑或），!（逻辑非）。其中 &&，|| 均为二元运算符，! 是一元运算符。例如：

p&&q，i||j，! t

当 p 和 q 的值全为真时，p&&q 为真，否则为假；当 i 或 j 只要有一个为真时，i||j 为真，而 i 与 j 全为假时，i|| 为假；当 t 为真时 ! t 为假，当 t 为假时，! t 为真。

2.4.4 位运算符

计算机中的数据都存储于一定的单元中，C++语言可以完成对这些单元的位操作，位操作一般是相对于字及字节的操作而言的，它的操作对象是这些字及字节中的某个或某几个位单元。一个字节包含八个位单元，一个位单元只有两种状态，即“0”和“1”。完成这些操作的运算符有：

&（位与）、|（位或）、^（位异或）、<<（左移）、>>（右移）、~（取反）

除 ~ 运算符为一元运算符外，其它都是二元运算符。各运算符的含义如下示：

0xb9 & 0x83	0xb9 | 0x83	0xb9 ^ 0x83	~0xb9
10111001	10111001	10111001	
& 10000011	| 10000011	^ 10000011	~ 10111001
10000001	10111011	00111010	01000110
结果为 0x81	结果为 0xbb	结果为 0x3a	结果为 0x46

&（位与）操作符：当 a&b 的 a 与 b 中对应位的值均为“1”时，在结果中的对应位的值也是“1”，否则结果中的相应位值为“0”。

|（位或）操作符：当 a|b 的 a 与 b 中对应位的值均不为“1”时，在结果中的相应位的值为“0”，否则结果中的相应位值为“1”。

^（位异或）操作符：当 a^b 的 a 与 b 中对应位的值互不相同时，即一个操作数的某

一位为“1”，另一个操作数的对应位为“0”时，则结果中的相应位为“1”，否则为“0”。

～(位取反)操作符：当 ～a 中对应位的值为“1”时，在结果中的对应位的值为“0”，否则结果中的相应位值为“1”。

移位运算符移动左边操作数的各位，移动位数由右边操作数指定，例如，0x9b＜＜2 的结果为 0xe4，0x9b＞＞3 的结果为 0x27 。左移时，低位补 0，高位溢出舍弃；右移时，高位补 0，低位移出舍弃。

2.4.5 条件运算符

条件运算符为三元运算符，格式为：

e1? e2:e3。

C++ 在对 e1? e2:e3 这一表达式求值时，先求 e1 的值，如果 e1 为真，则仅对 e2 求值，e2 成为该表达式的值；如果 e1 为假，则仅对 e3 求值，e3 成为该表达式的值。

例如，表达式 i＜5 ? 3 : 7，如果 i 的值小于 5，则表达式的值为 3，否则为 7。

2.4.6 逗号运算符

逗号运算符用于隔开一个表达式序列，这样的序列仍是一个表达式，其计算顺序从左至右依次进行，仅保留最后一个表达式的值，其它表达式的值都丢弃掉，这保留的最后一个表达式的值作为整个逗号表达式的值，例如：

j++，j＜5? 3:7 (其中 j++ 表示对 j 的加 1 运算，见后)

如果 j 的初始值为 4，则该逗号表达式的值为 7。

2.4.7 sizeof 运算符

sizeof 运算符为一元运算符，格式为 sizeof(e)，其中 e 为操作数。sizeof 运算符用于计算操作数在内存中所占的字节数，结果类型为 unsigned int 类型。

2.4.8 赋值运算符

赋值运算符为二元运算符，格式为：

e1 赋值运算符 e2 ；

赋值运算符的基本意义是将右边表达式(e2)的值赋给左边的操作数(e1)，从这一意义可知，左边的操作数只能是变量。这种可以出现在赋值运算符的左边的变量一般被称为左值，它的最主要的特性是能为值提供一个存储的空间。任何常量都不能出现在左值的位置上。

C++提供的赋值运算符如下：

＝，＋＝，－＝，＊＝，/＝，%＝，＜＜＝，＞＞＝，&＝，^ ＝，|＝

其中最基本的运算符为“ ＝ ”，其它的赋值运算符都是在“ ＝ ”运算符基础上的扩展。

1. “ ＝”运算符

“ ＝ ”运算符将右边表达式的值赋给左边的变量，例如：

a ＝ 100；　　a ＝ a＊2＋10；

此为试读，需要完整PDF请访问：www.ertongbook.com

如果上面两个语句顺序执行，则 a 的最后结果为 210。

“ = ”与一般数学运算中的“ = ”不完全相同，一般数学表达式的“ = ”表示一种推理的过程，通常要求在“ = ”两边的表达式相等。而赋值运算符将右边操作表达式的结果放入由左边变量标识的存储空间中去，二者不具有相等的意义。如上面的表达式 a = a * 2+10；，如果 a 中开始存放的值为 100，操作开始时将这个值读入处理器的寄存器中，当完成了运算之后，处理器又会将寄存器中的结果 210 放入由 a 指定的存储空间中去。可以看到，在整个操作过程中表达式 a 与 a * 2+10 的值不相等。

2. 扩展的赋值运算符

扩展的赋值运算符都有“其它运算符 = ”的形式，其含义为将运算符右边的操作数与左边的变量进行“其它运算符”的运算后，再将运算后的值赋给左边的变量。例如，已知 a= 1，b= 5，c= 7，x= 2，则如表 2-3 所示。

表 2-3

表 达 式	表达式的值
x = a+b * c	36
x += a+b * c	38
x -= a+b * c	34
x * = a+b * c	72
x /= a+b * c	18
x %= a+b * c	0
x <<= 1	4
x >>= 1	1
x &= 7	2
x |= 7	7
x ^ = 7	5

例如 x+ =a+b * c 的操作过程，程序先求表达式 a+b * c 的值，在这个例子中是 36，再由这个值与赋值运算符左边的变量做“其它运算符”定义的运算，这里是“+ ”，x 与 36 相加之后的值为 38，这个最后的结果被放入到 x 中去。其它的运算过程与此类似。

2.5 混合运算与类型强制转换

一个表达式中经常存在多种运算符，此时必须根据运算符的优先级确定运算顺序。进行混合运算时，各类运算符的优先级规则如下：

(1) sizeof 运算符、强制类型转换运算符、指针运算符、分量运算符、下标运算符高于算术运算符；

(2) 算术运算符高于逻辑运算符；

(3) 逻辑运算符高于条件运算符；

(4) 条件运算符高于赋值运算符；

(5) 赋值运算符高于逗号运算符。

(6) 算术运算符中“*”,“/”,“%”的优先级高于“+”,“-”;

(7) 逻辑运算符中! 的优先级最高,其次为 &&,|| 的优先级最低。

一个表达式中各操作数的计算顺序,总是隐含地先进行优先级高的运算,再进行优先级低的运算。如果要改变这种隐含的运算顺序或使运算顺序更加清析易读,可以加上括号(“(”和“)”),使得括号内的运算先进行。括号必须成对出现,可以嵌套使用。它的优先级是最高的。

当表达式中存在不同类型的操作数时,为求该表达式的值,C++编译器将自动对其中的一些操作数进行类型转换(称为隐式类型转换),以使一个二元运算符两边的类型一致。

例如,若变量 n 为整型,则表达式 n * 43.76 中运算符 * 两边的操作数类型不一致,43.76 为 double 型,n 为整型。对于这种情况,编译器将首先将 n 转换为 double 型,再求 n * 43.76 的值,若 n 为 3,则表达式的值为 131.28。一般的规则是将较简单的数据类型转换到较复杂的数据类型,所以这种类型转换也称为类型提升。

C++中隐式类型转换规则如下:

(1) 所有的 char,short 类型(unsigned short 例外)的数据转换成 int 类型的数据(这也适合于位运算符),该转换是无条件进行的,称作类型规范化。后面将见到的枚举常量在参与表达式运算时,其类型也总是 int。

(2) 除(1)以外的情况下,如果一个二元运算符两边操作数的类型不一致,总是将一个值域较小的类型向一个值域较大的类型转换。

(3) 对于赋值运算符,总是将赋值运算符右边表达式的类型转换成左边变量的类型,然后再赋给左边的变量。

除隐式类型转换外,也可对一个操作数进行强制类型转换,转换的方法是使用强制类型转换符。所谓强制类型转换符即是各种类型名。强制类型转换的语法是:

类型名(表达式) 或 **(类型名)表达式**

它将表达式的值类型强制转换为类型名所规定的类型。

综上所述,若 a,b 为 float 型,i,j,k 为 int 型,z 为 long double 型,t 为 double 型,则有表 2-4 所示的转换。

表 2-4

表 达 式	表达式的值或变量的值类型
i+'p'	int
i+j*a+b+k	float
t+k	double
i*j+k*(a+(b+c))*z	long double
t=i+'p'	double
(int)(a+b)+i	int
(int)a+b+i	float
long double (i+'p')	long double
i*(int)(a+b+z)	int

进行类型转换时，如果转换后的数据值域小于转换前的数据值域，则可能丢失数据精度。在浮点类型转换为整型时，小数部分被截断。尤其要注意的是，赋值运算符左边的变量类型只影响最后结果赋值时转换到的类型，而在运算过程中各个操作数的类型取决于操作中较高类型的操作数。例如求两个很大的整数的乘积，虽然两个整数都在 int 类型的值域中，但它们的结果超出了 int 类型的值域，而在 long int 类型的值域中，如果 a = b * c 中，b 和 c 的类型均是 int，那么即使用 long int 类型来定义 a，最后得到的结果也是一个溢出后可能错误的结果。这是因为在运算过程中的类型是 int 的缘故，在操作中被截断，最后的扩展已于事无补。

2.6 编写和运行简单的C++语言程序

2.6.1 编写简单的C++程序

从对C语言的继承性来看，可以简单地将C++语言程序理解成函数的集合。函数是一段顺序执行的语句，一个函数可以调用另一个函数，也可调用自身。函数和语句(语句是在表达式后跟“;”)中所用到的变量都必须在被使用之前予以说明。一个程序必须有唯一的一个 main() 函数，它是程序执行的起点。

函数是C++语言中的一个重要概念，其详细讨论见后，这里仅介绍一下它的格式。

函数的定义格式为：

函数返回值类型　　函数名(参数列表)

{

语句

……

}

最简单的C++程序仅有一个函数 main()，其返回值为空，参数列表为空，写成：

```
void main(void)
{
  语句
  ……
}
```

下面为一个完整的程序：

```
// A simple example                    ①
#include <iostream.h>                  ②
void main(void)                        ③
{
  int i,j;                             ④
  int add, mul;
  cout<<"C++ is a fun !";              ⑤
  i=3;
```

```
    j=4;
    add = i+j;
    mul = i*j;
    cout<<"\na+b = "<<add;                    ⑥
    cout<<      "\na*b = "<<mul;
}
```

运行结果为：

```
C++ is a fun !
a+b = 7
a*b = 12
```

程序中各语句解释如下：

①句：注释语句。C++语言的注释语句有两种，一种称为行注释语句，另一种称为段注释语句。行注释语句以“//”开头，“//”后一行内的任何字符都是注释字符；段注释语句以“/*”开头而以“*/”结束，处于“/*”和“*/”之间的字符都是注释字符。

②句：将文件 iostream.h 的全部源代码加入该语句的位置。iostream.h 也是C++语言源程序，它含有该语句之后的程序中一些标识符等的定义。一般，我们将一些定义集中在一个或几个文件中，然后用“ #include 文件名”的办法将其“包括”进来，这种文件一般“包括”在程序的开头，故称为“头文件”。请注意，有些编译器规定这种在行首有一个“#”标号的语句，一定要从一行的第一个字符位置写起，在它前面不要留有空格，否则编译器会报告出错。

③句：定义一个函数—— main()，按照该语句的定义，main() 的变量列表为空，返回值为空。main()函数是C++程序的入口，每一个C++程序中都必须定义 main()函数，而每一个程序的执行也都是从 main() 开始。

④句：main() 函数中所用到的变量说明。变量的说明格式为

类型名　　变量名序列；

语句“ int i,j; ”说明了两个整型变量 i,j。类似地也可有语句“float p,q”说明两个浮点型变量。

⑤句：输出语句。此处，“cout”和“<<”的含义在头文件 iostream.h 中被定义，“cout”表示标准输出设备(一般为显示器)；“<<”不是移位运算符，它被重新定义，此处表示输出运算(详见操作符重载和有关流的概念)。因此③句的含义为在标准输出设备上输出字符串“C++ is a fun !”。

⑥句：字符串中的“\n”是一个特别的字符，表示换行；“<<”和别的运算符一样，在一个表达式中可以被多次使用。⑥句使在“<<”之后的内容顺序出现在屏幕上。

从上面程序也可看出C++程序格式自由，空格符起分隔作用，除字符串中的空格外，连续的空格相当于一个空格，利用这个特点编写程序，可以使程序清晰易读。每一个语句用一个“;”结尾；多个语句可以写在同一行上，只要遇到一个“;”，C++编译器会自动将它们分割开来。请注意，不要因此而像写文章一样写程序，因为这样的程序可读性差。

下面程序的功能是，从键盘输入直角三角形的两直角边，求三角形的面积：

```
/* Example
    Getting the area of a triangle
*/
#include <iostream.h>
/* iostream.h 头文件中包含有关标准输入与输出的定义,
在有标准输入输出操作的程序中应把它包含进来 */
void main(void)
{
  float a,b,s;                          // 说明三个浮点型变量
  cout<<"Input the two edges : ";
  // 输入三角形的两直角边
  cin>>a;
  cin>>b;
  s = 0.5*a*b;                          /* 求面积 */
  cout<<endl<<"The area is : "<<s;
}
```

运行的结果如下：

```
Input the two edges : <3.0> <4.0> (3.0,4.0 是由用户输入的值)
The area is : 6
```

其中语句“ cin>>a;”表示从键盘输入一个浮点数至变量 a;“ cin”和“>>”类似于“cout”和“<<”,也是在头文件 iostream.h 中定义的,表示标准输入设备(一般为键盘)和输入运算;“endl”等同于“\n”,表示换行,它是在头文件“iostream.h”中定义的字符类型的常量。

在输入数据时,空格和逗号都可以作为分割变量的字符。但要注意在读入字符时,这些分割符号也会被作为字符读入,需要留心。另外还要注意输入的值一定要与接收该值的变量类型相匹配,否则会出现不可预测的结果。

注意,该程序使用了成对的“/*”与“*/”和单个的“//”,它们都表示注释。编译系统在编译时将忽略注释。

变量在说明时也可赋初值,例如,上面的程序可改写成：

```
#include <iostream.h>
void main(void)
{
  float a=3,b=4,s;                      // 给变量 a 赋初值 3,b 赋初值 4
  s = 0.5*a*b;
  cout<<endl<<"面积是:"<<s;
}
```

运行结果：

```
面积是:6
```

注意,上面程序中语句“cout<<endl<<”面积是:“<<s;”含有汉字字符串,可以这样做的前提是程序在汉字系统下执行。

2.6.2 编译、运行和调试C++程序

C++的编译系统有多种，如Turbo C++，Borland C++，Visual C++等，本书的程序全部是在Borland C++ 3.1集成环境下编译通过的程序，因此，我们仅介绍如何在Borland C++ 3.1集成环境下编译、运行和调试C++程序。

C++的运行环境也很多，本书的程序虽然一般可以在多种环境下运行，但它们依然特别针对PC机和DOS环境而言。

在DOS提示符下，运行文件"\borlandc\bin\bc"即进入Borland C++ 3.1集成环境。打开File菜单，选择New或Open子菜单，即可编辑新程序或已在磁盘上存在的程序，编辑完成后可选择File菜单的子菜单Save或Save as... 将程序存盘。对于一个处于激活窗口中的源程序，可以打开菜单Compile，再选择Compile(编译不连接)，Make(编译且连接)，Link(连接)及Build all(全部程序重新编译连接)。

源程序经编译、连接后生成可执行文件，它可在DOS提示符下运行。在集成环境下的源程序如果处于激活窗口中，则无论是否已生成可执行文件，均可打开菜单Run，再选择需要的子菜单命令Run(运行)，Go to Cursor(运行到光标处)，Trace into(跟踪运行)或Step Over(单步运行)。

在编译和连接程序之前，如果需要，可以打开Options菜单，选定需要的编译、连接参数，使得编译连接器按所使用的机器硬件、特别的程序语句、所希望的优化策略等编译、连接源程序。

在编译过程中带有调试信息(编译时使Options菜单中Debugger...子菜单的Source Debugging不处于none位置)的程序可以调试执行。集成环境中Debug即为调试菜单，选择该菜单中的命令，可以查看程序运行中变量的值，设置运行暂停位置等。

有关C++编译器的更为详细的信息，请查阅所使用的C++编译器手册。

2.7 变量名与函数名的命名法

为使程序具有良好的可读性，应仔细地为程序中所使用的各种变量和函数(函数的概念见后)命名，下面的命名方法供读者参考。

变量名由两部分构成——**前缀+主名**。前缀说明变量的类型，所有字母小写；主名说明变量的意义，第一个字母大写，其它字母小写。例如一个说明序号的整型变量，可以命名为nOrder，其中n为前缀，Order为主名。

函数名也由两部分构成，并采用动宾结构，各部分单词的第一个字母大写，宾语部分不是必需的。例如完成取得序号功能的函数，可命名为GetOrder；又如求三角形面积的函数，可命名为GetTriangleArea或者AreaTriangle，求绝对值的函数可命名为Abs。表2-5是本书使用的各种类型变量的前缀列表：

类型修饰符以各种简略字母表示：

u	unsigned
s	short

此为试读，需要完整PDF请访问：www.ertongbook.com