第 3 章　C++编程语言基础

如同在第 2 章中所提到的那样，Visual C++只是一个开发工具，并不能实现软件应具备的所有功能。因此，了解 C++编程语言的各种特性以及基础语法，对于游戏编程高手而言是非常必要的。

因此，本章将主要帮助读者熟悉 C++编程语言的优点及基本语法。把这些基本语法与 Visual C++开发工具相结合就是巩固所学知识的最好方法。但愿本章所列举的材料能够很好地帮助读者理解前一章中所学习的内容，同时能够建立起一定的 C++编程语言基础。如果您已经是一位 C++编程人员，那么请跳过本章，继续学习后面的内容。如果只是一个初学者，那么请一定要详细阅读这一章，并对每一节的示例代码进行实践。只有这样才能够真正的走进 C++程序设计的大门。

其实，有关 C++编程语言的内容在前面已经有所涉及，如第 2 章中的 HelloWorld 程序就演示了编程语言是如何实现文字输出的。本章将通过一些更加详细的内容来帮助读者了解 C++编程语言。为了保证这些应用程序代码能够正常运行，可能在前面的代码简单的涉及后面代码中的内容，读者只需要明白代码中本节所包含的内容即可。在阅读完本章的全部内容后，再回过头来看就会明白。

本章主要涉及的内容如下：

- C++编程语言的优点及其发展过程：了解 C++编程语言的优点及其发展。
- C++中的各种字符：掌握什么是 C++中的关键字、标识符等概念。
- 常用数据类型：明白常用数据类型的范围及其定义。
- 运算符与表达式：知道什么是表达式、运算符，及其如何使用。
- 常量与变量：掌握变量与常量的定义及区别。
- 控制语句：掌握基本的语法，并熟悉相关的控制语句。
- 数组与指针：知道如何使用数组及指针。
- C++类的特性：了解 C++中类的特性及其主要的成员。
- 运算符的重载：了解其基本方法，明白其优点。
- 编程规范：掌握基本的编程规范，对自己编程或者阅读别人的代码大有裨益。

3.1　C++编程语言是什么

在学习 C++这门编程语言之前，确实有必要了解一下这种编程语言的发展过程、优点及特性，这样才能对一种编程语言有一个全面的了解。

3.1.1　C++语言的由来

C++语言起源于 C 语言。在 1973～1979 年间，C 语言迅速成为应用最广泛的系统程序设计语言。然而，由于 C 语言也存在一些缺陷，例如类型检查机制相对较弱、缺少支持代码重用的语言结构等，造成用 C 语言开发大程序比较困难。为了克服 C 语言存在的缺点，在 1980 年，由美国贝尔实验室在 C 语言的基础上，开始对 C 语言进行改进和扩充，并将“类”的概念引入了 C 语言，构成了最早的 C++语言（1983 年）。

后来 C++中又引进了运算符重载、引用、虚函数等许多特性，并使之更加精炼。由贝尔实验室开发出的这种过程性与对象性相结合的程序设计语言，直到 1983 年正式取名为 C++。以后又经过不断的完善和发展，由美国国家标准化协会 ANSI 和国际标准化组织 ISO 一起进行了标准化工作，并于 1998 年正式发布了 C++语言的国际标准（ISO/IEC:98-14882）成为目前的 C++语言。

简单地说，C++语言是在 C 语言的基础上引入了面向对象的机制而形成的一门计算机编程语言。C++继承了 C 语言的大部分特点：一方面，C++语言将 C 语言作为其子集，使其能与 C 语言相兼容；另一方面，C++语言支持面向对象的程序设计，如类的概念和性质。这就是对 C 语言的重要改进。

3.1.2　C++语言的特点

C++语言的特点大致有如下 3 点：

- C++语言是一种面向对象的程序设计语言。其模仿了人们建立现实世界模型的方法。C++语言的基础是对象和类。现实世界中客观存在的事物都被称为对象。例如，一辆汽车、一家百货商场等。C++中的一个对象就是描述客观事物的一个实体，其是构成信息系统的基本单位。类（class）是对一组性质相同对象的描述，是用户定义的一种新的数据类型，也是 C++语言程序设计的核心。
- C++是 C 语言的超集。其不仅包含了 C 语言的大部分特性，例如指针、数组、函数、语法等。其还包含面向对象的特点，例如封装、继承、多态等。
- C++是程序员和软件开发者在实践中创造的。

注意：由于 C++语言来源于 C 语言，所以不管哪种 C++结构都可以用 C 语言实现。但 C 语言的结构，C++不一定可以实现。

3.2　C++中的各种字符

世界上所有的语言都有自己的字符表示和组合，像平时常常接触到的中文和英文都是如此。所以为了让大家都能够使用 C++编程语言，其设计者也给这种语言定义了自己的字符表示和组合。从本节起，才真正开始介绍 C++编程语言的基础内容。

3.2.1　标识符与关键字

在 C++中，有一套用来表示程序中的变量、常量、数据类型及语法关键字的符号，这些符号被统称为标识符。

标识符的命名有如下几点规则需要遵循：

- 标识符的第一个字符必须是字母或者下划线。
- 标识符中不应有除字母、数字和下划线以外的字符。
- 标识符的长度一般不超过 31 个字符。
- C++中的标识符可以大写，也可以小写。不过大写和小写是有区别的，即对应的大小写字母会被当作不同的标识符。例如 Good 和 good 是被当作不同的标识符，Hello 和 HELLO 也是不同的标识符。

例如：下面这些就是合法的标识符的例子。

```
Inno  Result  Good  At_This  GG_88  nCount
```

而如下这些就是不合法的标识符例子。

```
2SD  A!bc  GG*88  good-bye
```

在 C++的标识符中，有些单词组合是不能由用户声明的，其是由 C++编程语言本身保留使用，具有特殊的含义。一般用于表示固定语句、预定义类型说明、预定义函数等。这种标识符被统称为关键字或者保留字。

有了这些关键字，C++编译器才能正确识别输入的程序代码是如何分隔的，这就好像写应用文时为了突出重点，常常把关键字或者词进行标注一样。表 3.1 列出了一些 C++中常用的关键字（只是一部分），请读者注意。

表 3.1　常用关键字

public	private	class	delete
new	const	void	unsigned
int	long	short	char
true	false	float	double
main	try	this	Enum
bool	sizeof	operator	Struct
_asm	_int8	_int16	_int32

注意：在 Visual C++中，为了让用户阅读方便，所有的保留关键字都会被自动高亮标识出来。读者可以在 Visual C++的源文件中输入一些关键字试一试。

3.2.2　分隔符与注释符

C++中除了 3.2.1 节所说的标识符外，还有两种起特殊作用的符号。一种是用来分隔代码语句的，被称为分隔符；另一种是起说明作用的，被称为注释符。

1．划分语句的分隔符

其中分隔符又被称为 C++中的标点符号。用来将单词或者程序分隔，其表示某个程序的结束和另一个程序的开始。C++中包括如下几种分隔符。

- 空格符：用来作为单词与单词之间的分隔符。
- 逗号：用来作为说明多个变量的分隔符，或者多个参数（将在后面的章节讲解）之间的分隔符。
- 分号：用来作为 C++中语句的结束分隔符。
- 花括号：用来构造程序实体的分隔。

2．使语句无效的注释符

注释在程序代码中起到对程序语句注解和说明的作用。其目的是为了代码设计者或审查者方便阅读程序代码。对计算机而言其是无效的，在程序编译时，注释会被自动从程序代码中忽略掉。换句话说，这就好比大家在读书时所做的读书笔记，可以标记在书上，也可以标记在笔记本上，但对书的内容（代码）并没有任何影响，只是起一个辅助说明的作用。

在 C++中采用如下两种注释方法。

（1）使用/*和*/括起来进行注释，在/*和*/之间的字符都被作为注释符处理，适用于多行注释信息，如图 3.1 所示。图 3.1 中从/*开始一直到*/结束的两行字符都是被当作注释信息处理的。

```
void CList::disp()
{
    CNode * p = pHead;                  /*先把结点对象指针指向根结点*/
    for(int i=1; i<=length; i++)
    {                                   /*循环整个链表
                                        判断是否是有效指针*/
        if(p!=NULL)
        {
        cout<<p->data<<",";             /*输出当前结点中的数据*/
        p=p->pNext;                     /*把当前结点移到下一结点*/
        }
        if(i%10==0)                     /*当等于10个数时,输出换行*/
        {
            cout<<endl;
```

图 3.1　多行注释

（2）使用“//”，从“//”开始直到所在行的行尾，所有字符都被当作注释处理。适用于单行注释信息。这种方法已经遇到过多次，如图 3.2 所示。

```
    length = 0;
}
CList::CList(int n)
{
    pHead = new CNode(n);
    length = 1;
}
void CList::insert(int n)
{
    CNode * p = new CNode(n);           //创建新结点对象
    p->pNext = pHead;                   //把新结点的指向下一结点的指针指向根结点
    pHead = p;                          //把新结点变成要根结点
    length++;
}
void CList::remove()
{
    if(length > 0)
    {
        CNode * p = pHead;              //把结点p指针指向链表根结点
        pHead = pHead->pNext;           // head指向链首结点的下一个结点
        delete p;                       //删除p指向的结点
        length--;                       //长度减 1
```

图 3.2　单行注释

在真实的程序编程中，上面介绍的注释方法可以根据不同的情况进行选用，增强了注

释和阅读的灵活性。

注意：好的注释，能够提高程序设计或者代码阅读的效率。

3.3　C++中的常用数据类型

在 C++程序中数据的类型分为很多种，这就好比人的个头有高低、体重有胖瘦、皮肤有黑白一样。不同类型的数据，在数据存放的空间和处理数据的时间上都是各不相同的。本节的主要内容就是介绍 C++中常用的基本数据类型、声明关键字和格式。

注意：在游戏设计中对于数据的类型尤为重视。因为采用的数据类型不同，游戏最后的运行速度和处理效果是有质的差别的。

3.3.1　整数型数据

所谓的整数型数据类是指数据是没有小数部分的，其值可以是正，也可以是负，简称为整数型。例如日常生活中用到的 12、–22、1234、2009、–999、–2009、0 等都是整数型的。其中 12、1234、2009 等又被称为正整数，0 一般也被当作正整数处理。而–22、–999，–2009 等被称为负整数。在 C++中用 int 这个关键字来声明一个存放整数型数据的变量（变量这个概念将在 3.4 节中进行讲解）。例如：

```
int nCount;                                    //声明一个整数型变量 nCount
int nNum;                                      //声明一个整数型变量 nNum
int abc;.                                      //声明一个整数型变量 abc
```

声明整数型时要进行初始化，其方法如下：

```
int nCount=100;
```

说明：在 C++程序中所有数据类型都是通过关键字来声明的。

知道了如何声明一个整数型变量，那么一个整数型在计算机中可以表示的数值的大小范围是多少呢？如表 3.2 所示。

表 3.2　有符号整数范围

计算机系统	最　大　值	最　小　值
16 位机	32 767	–32 768
32 位机	2 147 483 647	–2 147 483 648

注意：整数变量能存储的最大值，是由计算机给其分配的存储空间的大小决定的。在不同的计算机系统中，其表示的范围是不同的。例如，早期的 16 位计算机是使用 2 个字节，而 32 位计算机则是使用 4 个字节。一般，现在的计算机都是 32 位的，不过也有 64 位的。

虽然整数型能表示的数已经比较大，但也有不够用的情况。例如，在有的游戏中表示银河系两个星球之间的距离是多少千米，或者太阳系的直径是多少米等数据时，整数型变量就不够了。为此 C++中就引入了长整数类型。用 long 这个关键字来声明一个长整数型变量。例如：

```
long lLen;                                  //声明长整数型变量 iLen
long lCount;                                //声明长整数型变量 iCount
long lTime;                                 //声明长整数型变量 iTime
```

长整数型变量，其存储数值的范围如表 3.3 所示。

表 3.3　长整数范围

计算机系统	最　大　值	最　小　值
16 位机	2 147 483 647	–2 147 483 648
32 位机	2 147 483 647	–2 147 483 648

从上面的表中会发现一个问题，长整数和整数其实只是在 16 位计算机上有表示范围的差异，而在 32 位计算机上是无差异的。这是因为在 32 位计算机上，一般把长整数和整数的表示范围设置为相同大小。

在计算机中长整数可能会降低程序执行的速度，所以有时候为了节约空间和提高程序执行速度，C++中又引入了短整数。在 C++中用 short 这个关键字来声明一个短整数型变量，其数值的范围在 16 位和 32 位计算机上都是一样的，为 32 767～–32 768。例如：

```
short sCount;                               //声明短整数型变量 sCount
short sNum;                                 //声明短整数型变量 sNum
short sLen;                                 //声明短整数型变量 sLen
```

其实，在日常生活中大家一般使用的都是正整数，即数学中的自然数，负数是比较少使用的。所以计算机为了在不增加存储空间和不影响执行速度的前提下，在存储数据时，直接把存储数据的符号去掉，这样变量能表示的最大值就扩大了，这种存储的数据在 C++中就被称为无符号数据类型。

C++中声明无符号整数型变量，是在 int 关键字前加上 unsigned 关键字。例如：

```
unsigned int unCount;                       //声明无符号整数型变量 unCount
unsigned int unNum;                         //声明无符号整数型变量 unNum
unsigned int abc;                           //声明无符号整数型变量 abc
```

既然是把最大值扩大了，那么其数值范围也不同了，无符号整数型变量存放的数值范围如表 3.4 所示。

表 3.4　无符号整数范围

计算机系统	最　大　值	最　小　值
16 位机	65 535	0
32 位机	4 294 967 295	0

既然有无符号的整数型变量，那么也就有无符号的长整数和短整数型变量，其声明都是在原关键字前加上 unsigned 关键字。例如：

```
unsigned long ulCount;                  //声明无符号的长整数型变量 uiCount
unsigned long ulNum;                    //声明无符号的长整数型变量 uiNum
unsigned short usLen;                   //声明无符号的短整数型变量 usLen
unsigned short usCount;                 //声明无符号的短整数型变量 usCount
```

无符号短整数其数值范围在 16 位和 32 位计算机上都是 65 535～0。而无符号的长整数其数值范围如表 3.5 所示。

表 3.5　无符号长整数范围

计算机系统	最　大　值	最　小　值
16 位机	4 294 967 295	0
32 位机	4 294 967 295	0

3.3.2　实数型数据

实数型数据是由一个整数部分、一个小数部分以及可选的后缀组成的。例如，3.1415、0.5、0.875 等都是实数型数据。在 C++中把实数型数据按照其表示的数值范围进行分类，可分为单精度、双精度和长双精度 3 种数据类型。

（1）声明一个单精度的实数变量使用 float 关键字，所以单精度的实数型又被称为浮点型。例如：

```
float half;                             //声明单精度的实数变量 half
float pi;                               //声明单精度的实数变量 pi
float num;                              //声明单精度的实数变量 num
```

单精度实数类型一般是以 32 位进行存储表示的，其数值表示范围为 3.4e+38～3.4e–38，最多只提供 7 位有效数字。

（2）双精度数据类型和单精度数据类似，因为其要占据的存储空间是单精度数据类型的两倍，所以被称为双精度数据类型。声明一个双精度类型的实数变量，使用 double 关键字。例如：

```
double dbNum1;                          //声明双精度的实数变量 dbNum1
double dbNum2;                          //声明双精度的实数变量 dbNum2
double dbNum3;                          //声明双精度的实数变量 dbNum3
```

双精度实数类型是以 64 位进行存储表示的，其数值范围为 1.7E–308～1.7E+308，最多可提供 16 位的有效数字。

（3）长双精度数据类型，理论上是双精度占用存储空间的两倍，但在实际中还是只占用 64 位的存储空间。声明一个长双精度实数变量，使用 long double 关键字。例如：

```
long double dbNum1;                     //声明长双精度的实数变量 dbNum1
long double dbNum2;                     //声明长双精度的实数变量 dbNum2
long double dbNum3;                     //声明长双精度的实数变量 dbNum3
```

长双精度因为其占用的存储空间和双精度相同，所以其实数类型的数值表示范围还是为 1.7E–308～1.7E+308，最多可提供 16 位有效数字。

实数在 C++中也可以包含一个指数形式的数值，其类似于数字的科学计数表示法。

例如：

```
1.5e3 其中 1.5 是尾数部分，3 是指数部分，表示的数值为 1500。
1.23e-2 其中 1.23 是尾数部分，-2 是指数部分，表示的数值为 0.0123
```

注意：字母 e 或 E 之前（即尾数部分）必须是有数字的。e 或 E 后面的指数部分必须是整数。

所有的实数型变量在声明时，对其进行初始化方法如下：

```
float half = 11111.511;
double dwSize = 11111.105;
```

由于 float 型的变量只能存储 4 个字节，有效数字为 7 位。所以其中 half 中的最后一位小数是不起作用的，即其值为 11111.51。但如果是 double 型的，则 dwSize 中是能够存储全部 11111.105 这个数的。对于这一点，读者只需要记住就行了，不必细究。

3.3.3　字符型数据

通常字符型数据变量是用来存储计算机中的字符的。例如：‘A’，‘#’，‘z’，‘1’等都是字符型的数据。在计算机中，字符型变量并不是直接存储字符，而是存储字符的 ASCII（American Standard Code for Information Interchange，美国标准信息交换码）码值。

ASCII 码是目前计算机中用得最广泛的字符集及编码，是由美国国家标准局（即 ANSI）制定的，其已被国际标准化组织（ISO）定为国际标准，称为 ISO 646 标准。适用于所有拉丁文字字母。

根据 ASCII 码标准，数值 65 代表大写的‘A’，而 97 则代表小写字母‘a’，表 3.6 列出了 ASCII 码表中前 128 个编码的十进制表示，读者可以不必记住这些代码值，但必须理解每个 ASCII 码都与一个特定的数值相对应的这个概念，ASCII 码表如表 3.6 所示。

表 3.6　ASCII码表

十进制	符　号	十进制	符　号	十进制	符　号	十进制	符　号
0	NUT	13	CR	26	SUB	39	,
1	SOH	14	SO	27	ESC	40	(
2	STX	15	SI	28	FS	41	)
3	ETX	16	DLE	29	GS	42	*
4	EOT	17	DCI	30	RS	43	+
5	ENQ	18	DC2	31	US	44	,
6	ACK	19	DC3	32	（space）	45	-
7	BEL	20	DC4	33	!	46	.
8	BS	21	NAK	34	”	47	/
9	HT	22	SYN	35	#	48	0
10	LF	23	TB	36	$	49	1
11	VT	24	CAN	37	%	50	2
12	FF	25	EM	38	&	51	3

续表

十进制	符　号	十进制	符　号	十进制	符　号	十进制	符　号
52	4	71	G	90	Z	109	m
53	5	72	H	91	[	110	n
54	6	73	I	92	\	111	o
55	7	74	J	93	]	112	p
56	8	75	K	94	^	113	q
57	9	76	L	95	—	114	r
58	:	77	M	96	、	115	s
59	;	78	N	97	a	116	t
60	<	79	O	98	b	117	u
61	=	80	P	99	c	118	v
62	>	81	Q	100	d	119	w
63	?	82	R	101	e	120	x
64	@	83	X	102	f	121	y
65	A	84	T	103	g	122	z
66	B	85	U	104	h	123	{
67	C	86	V	105	i	124	|
68	D	87	W	106	j	125	}
69	E	88	X	107	k	126	~
70	F	89	Y	108	l	127	DEL

看一下这张表，会发现其中有不少奇怪的字母组合，这些字母组合的含义如表 3.7 所示。

表 3.7　特殊字符的含义

NUL	空	VT	垂直制表	SYN	空转同步
SOH	标题开始	FF	走纸控制	ETB	信息组传送结束
STX	正文开始	CR	回车	CAN	作废
ETX	正文结束	SO	移位输出	EM	纸尽
EOY	传输结束	SI	移位输入	SUB	换置
ENQ	询问字符	DLE	空格	ESC	换码
ACK	承认	DC1	设备控制 1	FS	文字分隔符
BEL	报警	DC2	设备控制 2	GS	组分隔符
BS	退一格	DC3	设备控制 3	RS	记录分隔符
HT	横向列表	DC4	设备控制 4	US	单元分隔符
LF	换行	NAK	否定	DEL	删除

字符型在计算机中以 8 位进行存储，其表示数值的范围为 127～–128。在 C++中声明一个字符型的变量，使用关键字 char，例如：

```
char no;                                    //声明一个字符型变量 no
char yes;                                   //声明一个字符型变量 yes
```

```
char tmp;                                    //声明一个字符型变量 tmp
```

字符型的数据都是使用单引号引起来，例如‘Y’。如果要在声明字符型变量时进行初始化，其方法如下。

```
char no='N';                                 //声明字符变量 no，并初始化为 N
```

在 C++中还有一些是由单引号引起来的，但由多个字符所组成的特殊字符，C++把这些特殊字符当作单一字符来处理，称为转义字符。转义字符都具有特殊的功能，如表 3.8 所示。

表 3.8　转义字符

转 义 字 符	说　　明
\a	蜂鸣
\b	回退键
\f	换页
\n	换行
\r	回车换行
\t	水平制表
\v	垂直制表
\\	反斜杠
\’	单引号
\”	双引号
\?	问号
\nnn	八进制位模式，nnn 是一个八进制数
\xnn	十六进制位模式，xnn 是一个十六进制数

前面在讲解整数型时，知道分为“有符号”和“无符号”两种。其实字符型数据也是如此。如果要声明一个无符号的字符型数据，只需要在 char 关键字的前面加上 unsigned 关键字，其表示的数值范围就变为 255～0。例如：

```
usigned char no;                             //声明无符号字符型变量 no
usigned char id;                             //声明无符号字符型变量 id
```

现在再来了解另一种字符串型数据，所谓字符串就是由 0 个或多个字符组成的有限序列。例如 abcde、Hello、1234 这些都是字符串。字符串就相当于一个字符数组。

注意：在表示单独字符的时候，使用单引号，而在表示字符串或多于一个字符的时候必须使用双引号。

3.3.4　布尔型数据

布尔型数据即是逻辑型的简单数据值，其只有包含两个值：false（假）和 true（真），前者序号为 0，后者序号为 1。在 C++中，虽然布尔类型的数据量最少，但用途却非常广泛，其主要用于程序设计中的流程控制和逻辑判断。

声明一个布尔型的变量，使用 bool 关键字，例如：

```
bool bFlg;                                    //声明一个布尔型变量 bFlg
bool bOpen;                                   //声明一个布尔型变量 bOpen
bool bClose;                                  //声明一个布尔型变量 bClose
```

在声明布尔型的变量时，也可以进行初始化，其方法如下。

```
bool no = false;                              //在定义布尔型变量 no 时，初始化为 false
```

注意：在 C++中，只有 0 才能表示为假。其他任何数值代表布尔型变量时都表示真，包括负数。

总之，在设计游戏需要的数据类型时，应尽量满足游戏设计的要求。比如要计算有小数参与的运算，就应该选择实数类型，而不能选择整数型。而在没有小数参与的运算时，就应该选择整数型。

3.4　C++中的常量与变量

前面的数据类型一节中，常常会提到“变量”，在本节将会对其进行详细的说明。不仅如此，还将学习到另一个概念——“常量”。

3.4.1　变量的定义

变量就是在内存中，可以不断地被程序操作的、有名字的存储区。在代码中可以只使用一个变量，也可以使用多个变量。在使用变量前，必须先要创建一个变量。创建变量的 C++语句称为变量的声明，在前面已经见过。其格式如下：

```
变量数据类型  变量名;
int nLen;                                     //声明一个整数型的变量 nLen
```

也可以使用如下格式同时声明 n 个同类型的变量，但要注意一定是同类型的才能如此。

```
变量数据类型  变量名 1,变量名 2,…,变量名 n;
int a,b,c;                                    //同时声明三个整数型变量 a,b,c
```

不同类型的变量不能在同一语句中进行声明，下面的方式是错误的。例如：

```
int  len,short  count;                        //在同一行声明不同类型变量，这是错误的
```

通过前面的学习，知道了 C++中的每一个变量都有特定的类型，该类型决定了变量的内存大小和布局，以及能够存储于该内存中的值的取值范围和可应用于该变量上的操作集。这样计算机就可以正确理解变量中的数据含义了。

在前面的声明格式中看到的“变量名”就是变量的名字，就像每个人都有自己的名字一样，给变量起名的意义是为了区分不同的变量。在 C++中的变量名称是不能随便取的，除了必须遵循标识符命名规则外，还应该尽量遵循下面几条编程经验，虽然不是必须遵循

的，但为了方便对代码的阅读和理解，在程序设计时应尽量采用。

- 见名见意，指当看到这个变量的名称时就能望名知意，这样能便于读者阅读和进行程序设计。例如，用 count 来作为计数器变量名，用 len 作为长度变量名，用 age 作为年龄变量名等。
- 尽量不用汉语拼音。例如，声明一个姓名变量，应该使用 name 为变量名，而不是用 xingming 作为变量名。
- 命名不宜过长。用很长一串字符来命名会减慢编程的速度，也会使阅读程序困难。不要用过长的名称来命名变量，一般采用缩写来命名，例如，用 init 来表示初始化等。
- 采用驼峰标记法和匈牙利标记法来命名变量。

声明变量之后必须对其进行初始化才能使用，否则会导致程序运算错误。对变量进行初始化后，变量的值才会有效。

```
变量名=初始值;
```

也可以在声明变量的时候进行初始化，其格式如下：

```
变量类型 变量名=初始值;
```

要注意，在变量初始化的时候，设置的初始值一定要符合变量的数据类型。例如：

```
int nPI = 3.1415                                    //错误的初始化
int nCount = 100                                    //正确的初始化
```

3.4.2　常量的定义

与变量相对的就是常量。变量的值可以在程序中随时而改变，而常量的值是不能被改变的，所以其被称为常量。比如在游戏设计中，设置常常会把圆周率 π 设置为常量。其值就默认等于 3.141 592 7。

在 C++中常量分为如下两种：

- 文字常量，例如整数 888、字母 b 等都是文学常量。
- 自定义常量，即自己声明的常量。

自定义常量的声明格式与变量声明差不多，只是在语句的最前面多了一个 const 关键字来说明这是一个常量。例如：

```
const 数据类型 常量名=文字常量;
const int MAX=1000;                          //声明一个 int 型的常量 MAX，其值为 1000
const double PI = 3.1415927f;                //声明一个 float 型的常量
```

常量必须在声明时进行初始化，并且在除声明语句外，在程序的任何地方不能再对其进行赋值。这是和变量不同的地方之一。

声明一个单精度实型常量，需要该文字常量的最后加上 F 或者 f；如果要表示长双精度实型常量，则要在该文字常量的最后加上 L 或者 l。例如：

```
1.05f;                                       //声明的是单精度实型常量
```

```
1.75l                                    //声明的是长双精度实型常量
2.55F;                                   //因为采用了 F 说明，所以是单精度常量
3.75L;                                   //因为采用了 L 说明，所以是长双精度常量
```

事实上，只要在游戏程序中不需要改变的值，都应用常量来表示。这样做可以提高程序稳定性和严谨性。

3.5　C++中的运算符与表达式

除了变量和常量外，在 C++中还有其自身支持的、操作变量的运算符和表达式。运算符是表示对数值进行一种运算的符号。其对操作数进行运算，操作数可以是一个数值、变量或者常量。比如，大家平时接触到的数学中的运算符（+、–）及比较运算符（>、<、=）等。这些在 C++中也是支持的。

C++的运算符范围很广，有带一个操作数的，也有带两个操作数的；有些是专用的，而有些是由其他运算符派生出来的；甚至有些是重载的（将在后面的章节进行讲解）。所以很难找到一个程序能把所有的运算符都用上。

表达式是与运算符对应的，其代表的是由运算符和操作数组成的式子。由于运算符很丰富，因此表达式的种类也很多。最简单的表达式是常量或者变量。

在本章中，笔者只列举出常用的几种简单的运算符和表达式。同时在示例代码中给出其使用方法。

3.5.1　赋值运算符

在程序设计中最常用到的赋值运算符。C++规定赋值运算符为=，其作用是将一个数据赋值给一个变量，类似于数学中的等于符号。如 num1=1 的作用是把字符常量“1”赋值给变量 num1。也可以将一个表达式的值赋值给一个变量，由赋值运算符将一个变量和一个表达式连接起来的语句被称为“赋值表达式”，格式为：

```
左值=表达式;
```

赋值运算符的操作过程如下：

（1）计算右边表达式的值。

（2）把右边计算出来的值，存放在左值之中。

左值可以理解为变量或者声明语句中的自定义常量。例如：

```
a = 11;                                  //赋值给左值变量 a，值为 11
const int A=10;                          //赋值给左值常量 A，值为 10
```

3.5.2　算术运算符

在 C++中支持的运算符如表 3.9 所示。

表 3.9　算术运算符

运　算　符	说　　明	例　　子
+	加法运算符，或者表示正值	2+3　+8
–	减法运算符，或者表示负值	5–2　–88
*	乘法运算符	7*2
/	除法运算符	9/3
%	取模运算符，又称取余运算符	10%3 的结果是 1

算术运算符的计算方法同数学中的基本一致。不过要注意：当除号两边的数为整数型数据时，其结果为整数。如 5/3 的结果值为 1，舍去小数部分。但是如果除数或者被除数中有一个为负值，则舍去的方向是不固定的。例如，–5/3 有的电脑上–1，有的是–2。

下面的例子说明了算术运算符的使用方法，如代码 3.1 所示【代码参考：光盘的源代码\C03\Demo1.dsp】

代码 3.1　算术与赋值运算符示例

```
#include <windows.h>                    //一个 Windows 应用程序应该包含的头文件
#include <stdio.h>                      //标准输入输出流文件
LRESULT CALLBACK WinSunProc             /*声明一个回调函数*/
(
 HWND hwnd,                             /*窗口的句柄*/
 UINT uMsg,                             /*窗口的消息*/
 WPARAM wParam,
 LPARAM lParam
 );
int WINAPI WinMain
(
 HINSTANCE hInstance,                   //实例句柄，当前应用程序的实例句柄
 HINSTANCE hPrevInstance,               //默认这个参数为 NULL
 LPSTR lpCmdLine,                       //储存一个命令行参数
 int nCmdShow)
{
    WNDCLASS wndcls;                    //定义一个窗口对象
    wndcls.cbClsExtra=0;                //指定额外内存空间
    wndcls.cbWndExtra=0;                //指定额外内存空间
                                        //指定窗口背景色
    wndcls.hbrBackground=(HBRUSH)GetStockObject(WHITE BRUSH);
                                        //设置光标样式
    wndcls.hCursor=LoadCursor(NULL,IDC_CROSS);
                                        //设置图标样式
    wndcls.hIcon=LoadIcon(NULL,IDI_ERROR);
    wndcls.hInstance=hInstance;         //指定窗口实例句柄
    wndcls.lpfnWndProc=WinSunProc;      //指定窗口函数，即窗口主处理函数
    //窗口类名称
    wndcls.lpszClassName="Visual C++ Game";
    wndcls.lpszMenuName=NULL;           //菜单
    wndcls.style= CS HREDRAW|CS VREDRAW;
    RegisterClass(&wndcls);             //注册窗口类
    HWND hwnd;                          //声明窗口句柄
    hwnd=CreateWindow                   /*创建窗口，但这里的窗口是不会显示的*/
        (
        "Visual C++ Game",              /*已注册窗口类的名称*/
        "Visual C++ 游戏开发",          /*窗口标题*/
```

```
38          WS_OVERLAPPEDWINDOW,          /*窗口风格*/
39          200,                          /*窗口位置的横坐标*/
40          200,                          /*窗口位置的纵坐标*/
41          600,                          /*窗口的宽度*/
42          400,                          /*窗口的高度*/
43          NULL,
44          NULL,
45          hInstance,                    //实例句柄
46          NULL);
47      //在这里真正显示窗口
48      ShowWindow(hwnd,SW_SHOWNORMAL);
49      UpdateWindow(hwnd);               //更新显示
50      MSG msg;
51      while(GetMessage(&msg,NULL,0,0))
52      {
53          TranslateMessage(&msg);       //转换键盘消息
54          DispatchMessage(&msg);        //分派消息
55      }
56      return 0;
57  }
58  LRESULT CALLBACK WinSunProc(
59                              HWND hwnd,              /*窗口句柄*/
60                              UINT uMsg,              /*消息*/
61                              WPARAM wParam,          /*参数 1*/
62                              LPARAM lParam           /*参数 2*/
63                              )
64  {
65      char tmsg[128] = {0};
66      int num1, num2, num3, num4,num5;    //声明 5 个变量
67      num1 = 3+8;                         //加法运算
68      num2 = 10-7;                        //减法运算
69      num3 = 100*33;                      //乘法运算
70      num4 = 155/5;                       //除法运算
71      num5 = 9%2;                         //取模运算
72                                          //把运算符和结果输出到 tmsg 中
73      sprintf(tmsg, "3+8=%d 10-7=%d 100*33=%d 155/5=%d 9%%2=%d",
74              num1, num2, num3, num4, num5);
75      switch(uMsg)                        /*判断消息类型*/
76      {
77      case WM_PAINT:                      /*更新窗口消息*/
78          HDC hDC;                        /*定义 DC 设备*/
79          PAINTSTRUCT ps;
80          hDC=BeginPaint(hwnd,&ps);       /*得到设备 hDC*/
81          TextOut(hDC,150,0,tmsg,strlen(tmsg));
82          EndPaint(hwnd,&ps);
83          break;
84      case WM_CLOSE:                      /*当单击“关闭”按钮时，产生关闭消息*/
85          if(IDYES==MessageBox(hwnd,"是否真的结束？","游戏开发",MB_
            YESNO))
86          {
87              DestroyWindow(hwnd);        /*单击“确定”按钮，销毁窗口*/
88          }
89          break;
90      case WM_DESTROY:                    /*销毁窗口消息*/
91          PostQuitMessage(0);             /*退出程序*/
92          break;
```

```
93      default:
94          //在 default:处必须调用 DefWindowProc，这是 Windows 内部默认的消息处
            理函数
95          return DefWindowProc(hwnd,uMsg,wParam,lParam);
96      }
97      return 0;
98  }
```

代码解析：第 71 行，是用%运算符来表示这是一个取模运算。其意思是用来得到除法运算后的余数，即数学中的取余运算。其方法是先用 9 除以 2，得到商是 4，余数是 1，再把商丢弃，最后得到余数，并保存到变量 num5 中。第 73 行是把变量的结果格式化输出到字符数组 tmsg 中，格式化输出的方法是属于 C 语言的内容，请参见相关书籍，这里不再介绍。代码编译运行的结果如图 3.3 所示。

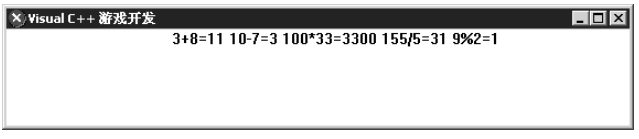

图 3.3　赋值与算术运算符示例运行结果

在 C++中用算术运算符和括号将运算对象连接起来的、符合 C++语法规则的语句，被称为算术表达式。运算对象包含常量、变量等。如代码 3.1 中的 3+8、10–7、num2*num1、num3/5，以及 num3%2 这些语句都是算术表达式。

3.5.3　自增与自减运算符

有两个特殊的运算符++和– –，在 C++中被称为自增和自减运算符，运算符++的作用是使变量的值增加 1。其表达式如下：

```
a++;            //相当于 a=a+1
++a;            //相当于 a=a+1
```

a++和++a 的作用相当于 a=a+1 都是将 a 的值加 1，但也有些不同。因为 a++是先使用 a 的值，再执行 a=a+1。而++a 是先执行 a=a+1，然后使用 a 的值。如果使用另一个变量来存放运算结果，这两种形式的不同就容易理解了。

例如：现在假定 a 的初值是 30。

```
int a=30;
int b=a++;
```

上面的语句是先将 a 的值 30 赋值给 b，然后 a 增加 1。最后的结果是变量 b 的值为 30，而变量 a 的值为 31。如果把语句修改为：

```
Int a=30;
int b=++a;
```

上面的语句是先将 a 增加 1，然后将 a 的新值 31 放在 b 中，最后的变量结果是变量 b 的值为 31，而变量 a 的值仍为 31。

在这里，把++a 念为“a 先加”而把 a++念为“a 后加”。好了，相信现在大家应该明

白++的作用了吧，其实另外一个运算符– –正好和++的作用相反，是将变量的值减少 1。但运算过程是一样的，这里不再复述。

自增（++）和自减（– –）运行符，只能用于变量，而不能用于常量或者表达式，如“10++”或者（a*c）++都是不合法的。因为 10 是常量，常量的值不能被改变。而“（a*c）++”是一个表达式，相加之后的结果没有变量可供存放。

3.5.4　复合运算符

复合运算符是对赋值运算符的扩展，其是在“=”赋值符之前加上其他运算符，就构成了复合运算符。使用复合运算符可以简化程序，使程序看上去精练，提高代码的编译效率。其表达式格式如下：

```
变量　运算符=表达式;
```

例如：

```
b+=4;        //等价于 b=b+4
b-=4;        //等价于 b=b-2
```

以“b+=4”为例来说明，其相当于先使变量 b 加 4，然后再重新赋值给变量 b。同理，“b–=4”是先使变量 b 减 4，然后再赋值给变量 b。为了便于记忆，读者也可以这样理解：

（1）a+=b，其中 a 为变量，而 b 为表达式。

（2）把 a+移动到=的右侧，变成=a+b。

（3）在=的左边补上变量名，变成 a=a+b。

即变成如下格式：

```
变量 = 变量 运算符 表达式
```

如果表达式是由几个表达式组成的，则相当于该表达式是有括号的。例如：

- a*=3*（2+3）。
- =a*（3*（2+3））。
- a=a*（3*（2+3）），不要写成 a=a*3*（2+3）。

凡是只有两个操作数的运算符，都可以与赋值运算符组成复合运算符。在 C++中规定了如下所示两类共 10 种复合运算符。

- 算术类：+=、– =、*=、/=、%=。
- 位运算类：<<=、>>=、&=、^=、|=。

3.5.5　位运算符

在计算机内所有的数据，总是用 0 和 1 的组合进行存储的，也就是二进制。而 8 个二进制位构成一个字节，两个字节构成一个字，也就是 16 位。有时按位来操作数据是很必要的，程序员可以通过改变存储在位、字节或者字中的 0 和 1 来改变它的值，这也就是位操作的由来。其表达式如下：

变量 位运算符 变量或者常量

1．左位移运算符（<<）

左位移运算符是将一个字中的数据位左移指定的位数。左移位一次相当于对当前这个数值每次乘 2。其实现的方法如下：

（1）在 32 位计算机中，7 被表示为 0000 0000 0000 0111。

（2）左移 4 位，变成 0000 0000 0111 …。

（3）空位用 0 来填充，变成 0000 0000 0111 0000。

（4）这个二进制数的数值正好是 112。

2．右位移运算符（>>）

右位移运算符刚好和左位移相反，它是将一个字中的数据位右移指定的位数。实现方法如下：

（1）在 32 位计算机中，128 被表示为 0000 0000 1000 0000。

（2）右移 2 位，变成…0 0000 0001 0000。

（3）空位用 0 来填充，变成 0000 0000 0001 0000。

（4）这个二进制数的数值正好是 16。

不过要注意，在右移位时，只有移位的变量是正数时，才会用 0 作为填充位，而如果变量是负数的话，右移位的结果是无法预料的，左移位无此限制。右移位一次相当于对当前数值每次除 2。

3．按位与运算符（&）

按位与运算符（&）是用来得到两个整数操作数的逻辑乘积。当被“与”的两位是 1 时结果是 1，否则结果为 0，如表 3.10 所示。

表 3.10　按位与示例表

操作位 1	操作位 2	结　果
0	0	0
0	1	0
1	0	0
1	1	1

现在设定 n 的初值为 250，m 的初值为 86。那么这两个数进行按位与运算后，其结果如表 3.11 所示。

表 3.11　按位与运算结果

数　值	说　明
1111 1010	250 二进制
0101 0110	86 二进制
0101 0010	对齐后，根据表 5.2 方法竖式运算，结果为 82

4．按位或运算符（|）

按位或运算符（|）是用来得到两个整数操作数的逻辑和。当被“或”的两位是 0 时结果是 0，否则结果为 1，如表 3.12 所示。

表 3.12　按位或示例表

操作位 1	操作位 2	结　　果
0	0	0
0	1	1
1	0	1
1	1	1

现在设定 n 的初值为 250，m 的初值为 86，进行按位或运算过程和结果如表 3.13 所示。

表 3.13　按位或运算结果

数　　值	说　　明
1111 1010	250 二进制
0101 0110	86 二进制
1111 1110	对齐后，根据表 5.4 方法竖式运算，结果为 254

5．按位异或运算符（^）

按位异或运算符（^）是用来得到两位操作数之间的异或结果的，其示例见表 3.14。

表 3.14　按位异或示例表

操作位 1	操作位 2	结　　果
0	0	0
0	1	1
1	0	1
1	1	0

现在设定 n 的初值为 250，m 的初值为 86，进行按位异或运算过程和结果如表 3.15 所示。

表 3.15　按位异或运算结果

数　　值	说　　明
1111 1010	250 二进制
0101 0110	86 二进制
1010 1100	对齐后，根据表 5.4 方法竖式运算，结果为 172

3.5.6　关系运算符

关系运算符与数学的比较运算符的运算方法相同，但标记格式不同。C++提供了 6 种

关系运算符，如表 3.16 所示。

表 3.16　关系运算符

运　算　符	说　　明	例　　子
<	小于	a<10
<=	小于等于	a<=10
>	大于	b>10
>=	大于等于	b>=200
==	等于	a==b
!=	不等于	a!=b

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，例如：

```
a>b                                  //a 大于 b
a+b<b+c                              //a+b 小于 b+c
(a=30)<(b=50)                        //变量 a 的值 30 小于变量 b 的值 50
b==a                                 //变量 b 和变量 a 相等
```

所有关系表达式的值是一个布尔值，即“真”（1）或者“假”（0）。例如，关系表达式“5==3”的值为“假”，而“5>3”的值为“真”。

3.6　C++中的控制语句

一个结构化程序设计需要 3 种基本结构：**连续结构、选择结构和循环结构**。在 C++中这 3 种结构分别对应基本语句、条件选择语句和循环语句。掌握好这 3 种语句的使用，对于程序设计是非常重要的。

和其他高级语言一样，C++的语句也是用来向计算机系统发出操作指令的。这就好像操练时的“向左转”、“向右转”、“稍息”、“立正”等一系列口令（即语句）。有了这些口令大家才能走出整齐的队列。

3.6.1　基本语句

一条基本语句经过编译后，将产生几条机器指令。为实现特定功能的程序一般又包含若干条基本语句。通常把 C++的基本语句分为如下两种。

1．简单的基础语句

在一个 C++程序中有许多表达式。有由算术运算符组成的算术表达式、由赋值运算符组成的赋值表达式、由位运算符组成的位运算表达式等。例如：

```
x=x+4                                //算术表达式
x=7*y                                //赋值表达式
y=4,x=3,n/55
x++,y--
x|y +n^m                             //位运算表达式
```