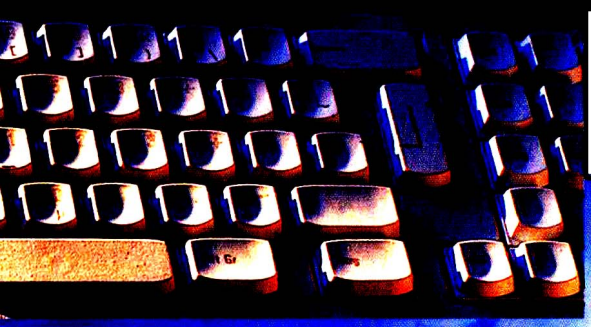
万水计算机实用教程系列

Visual C++ 6.0

编程实用教程

梁　维　　主编
门槛创作室　　编著



中国水利水电出版社

万水计算机实用教程系列

Visual C++ 6.0 编程实用教程

梁 维 主编

门槛创作室 编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书系统地介绍了 Visual C++ 6.0 的开发环境和使用 Visual C++ 6.0 进行程序开发的方法。

本书分为基础篇与应用篇两个部分，系统讲解了 Visual C++的使用方法。基础篇介绍了 C++语言的数据和表达式、语句和函数等 C++的语言特点，以及 Visual C++ 6.0 开发环境的基础知识、MFC 类库。应用篇从多方面进行程序开发的目的出发，介绍了 Visual C++ 6.0 中多种应用工具。为了提高本书的实用价值，本书还系统介绍了当前程序开发中比较热门的 ActiveX 控件和与数据库相联系的方法。

本书力求简洁明了地介绍各种基础和高级话题，并在书中提供了许多具有特点的小例子。因此，本书不仅可以为初学者提供一个入门的捷径，也可以为深层次的程序开发者作为参考。

图书在版编目(CIP)数据

Visual C++ 6.0 编程实用教程/梁维主编；门槛创作室编著. －北京：中国水利水电出版社，1999.9

（万水计算机实用教程系列）

ISBN 7-5084-0014-3

I. V … II.①梁… ②门… III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(1999)第 33381 号

书　　名	Visual C++ 6.0 编程实用教程
作　　者	梁　维　主编　门槛创作室　编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044）
	网址：www.waterpub.com.cn
	E-mail：sale@waterpub.com.cn
	电话：（010）63202266（总机）、68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京门槛创作室 WORD 输出中心
印　　刷	北京市天竺颖华印刷厂
规　　格	787×1092 毫米　16 开本　20.5 印张　460 千字
版　　次	1999 年 9 月第一版　1999 年 9 月北京第一次印刷
印　　数	0001—5000 册
定　　价	全套定价：130.00 元　本册定价：25.00 元

凡购买我社图书，如有缺页、倒页、脱页者，本社发行部负责调换

编委会

目　录

第一篇　基 础 篇

第二篇 应 用 篇

第一篇

基 础 篇

本篇导读

本篇的内容是本书中的基础部分，分为四个部分：C++语言基础、面向对象的程序设计、Windows 编程与 MFC 库、Visual C++ 6.0 的界面及开发工具。在“C++语言基础”这一章中，向读者介绍了 C++的起源和特点、C++程序结构、C++的数据和表达式、语句和函数、数组和指针、结构等内容，读者可以从中了解 C++的语言特点。在“面向对象的程序设计”这一章中，介绍了对象和类、继承和派生、多态性和虚函数、运算符重载等类/对象的属性和关系以及类/对象进行编程的语法特点，使读者获得面向对象程序设计的基本概念。在“Windows 编程与 MFC 库”这一章中，介绍了 Windows 编程的模式、Windows 编程特有的消息/响应机制和 Visual C++2.0 以上版本所提供的 MFC 类库的基本构成。在第 4 章“Visual C++ 6.0 的界面及开发工具”中，对 Visual C++ 6.0 的安装、界面、系统提供的工具、开发环境及 MFC 库对此提供的帮助进行了介绍，使得读者对 Visual C++ 6.0 的使用环境更加熟悉。本章的最后用 AppWizard 创建了一个简单的小例子。通过本篇的学习，读者可以掌握 Visual C++ 6.0 的基本内容。

第1章　C++语言基础

本章首先对C++语言的发展和特点进行了简要回顾，而后系统地介绍了C++语言的语句、语法，并通过几个简单的小程序说明C++程序的一般流程与结构。

1.1　C++的起源和特点

1972年，贝尔实验室的Dennis Richie首先设计并使用了C语言。C语言在过去B和BCPL语言的基础上发展起来，并很快成为Unix操作系统的描述语言。随着Unix系统成功而广泛的应用，C语言也随之成为一种普遍流行的语言。这种普遍流行不仅得利于Unix的广泛传播，而且更重要的应归功于C语言自身所具有的优势和特点。正是这些特点构成了C语言顽强的生命力。这些特点主要包括以下几个方面：

- 模块化的程序设计

结构化的程序语言设计要求有严谨的逻辑结构，其中包括顺序、选择、循环三种基本结构。C语言不仅提供了这三种基本结构，还提供了大量模块化的函数。这些函数是C语言模块化程序设计的基本单位。

- 高效的可移植性

C语言的所有语句不依赖于计算机的硬件设备，所有的输入和输出操作只是由独立于C之外的系统提供的库函数来完成。具有不同硬件设备的计算机之间程序的移植就变得非常方便。

- 丰富的运算符和数据类型

丰富的运算符和数据类型使程序设计得到简化，但同时也使初学者难以理解和掌握丰富的功能。

- 高效灵活的设计思想

C语言的设计目的是为了满足系统程序设计的需要，因此具有极高的表达能力，能够充分描述系统软件的各方面特性。由C语言生成的机器代码短小、紧凑而且高效。

C++是在C语言的基础上发展起来的。它在保持了C语言简洁、高效优点的基础上，引入了Simula 67、Algol 68和BCPL语言中的一些概念，改进、扩充了C语言，增加了类与对象的概念，开始全面支持面向对象的程序设计。

C++与C语言兼容。C语言中众多的库函数和过去形成的代码都可以在C++的语言环境中运行。C++语言编写的程序可读性好、结构合理，可直接在程序中映射问题空间的结构。比较而言，C++在程序的模块化上有了一个质的飞跃。C++与C语言的主要优、缺点如表1.1所示。

表1.1　C++与C语言的主要优、缺点比较

语言	检查机制	代码复用	程序规模
C语言	弱	不支持代码复用的语言结构	不适合开发大型程序，复杂程度难以控制

（续表）

语言	检查机制	代码复用	程序规模
C++	强，有专门的机制检查类	类的继承和派生极强的支持代码复用	适于中、大型程序开发，维护性、扩充性、可靠性强

1.2 C++程序结构

首先考虑下面这个简单的C++程序：

```
#include <iostream.h> //C++语言程序的一个系统文件
main()                //C++语言程序的主体函数
  {
    cout<<"Let's begin studying C++ language!";    //一条执行语句
  }
```

在这个简单的C++程序中，#include是一条编译指令，目的是把系统文件iostream.h插入程序中，使程序可以使用iostream.h中所定义的标准输入和输出操作。Cout是标准输出流，一般指屏幕。运算符<<把右边由双引号引用的内容输出到屏幕上。此程序由C++编译器编译后，可形成一个可执行文件。执行此可执行文件，可在屏幕上得到以下结果：

Let's begin studying C++ language!

由以上这个简单的例子可以初步了解C++语言程序的大致语法结构。

首先，C++程序以main开始的部分定义了一个函数，该函数规定了程序的功能。Main是函数名，其后必须紧跟一对圆括号。所有的C++程序都必须有一个名为main的主函数，程序执行的开始点是main函数中的第一条语句。

另外，一个C++函数中的任何成分都要括在一对花括号中，函数main后面的右花括号后要紧跟一个左花括号，表示函数从这里开始，最后的右花括号表示函数在这里结束。花括号括起来的部分称为函数体，函数名main和它后面的一对圆括号称为函数头。函数体由一系列的C++语句组成，这些语句描述这个函数怎样实现它的功能。

为了增强程序的可读性，C++为程序员提供了语言注释的功能。C++语言有两种注释的方法。符号//告诉编译器，//之后的本行所有内容都是注释；符号/*~~~*/告诉编译器，在/*和*/之间的内容都是注释。//适用于比较短的只占一行的注释，而/*~~~*/较适用于长的占用多行的注释。

虽然C++语言的书写格式比较自由，允许空格和回车起到同样的分隔作用，一个合格而且严谨的程序员应恰当地使用空格，以增强程序的可读性。

1.3 数据和表达式

程序是由数据和处理数据的操作组成的。在理解C++语言程序之前，必须了解C++语言的基本数据类型和对数据的简单处理方式——运算和表达式。

1.3.1 数据类型

C++语言程序所处理的数据都具有一种数据类型，类型规定了它们的存储表示以及可

以对它们进行的操作。类型名可以是一个标识符、一个关键字或者是一个表达式。基本数据类型是C++类型系统的基本组成部分。这些基本数据类型包括字符类型、整数类型、浮点类型、双精度类型和无值类型，相应的关键字是char、int、float和void。这些数据类型的位数和相应的取值范围如表1.2所示。

表1.2　C++语言数据类型

类型名	位数	范围
char	8	-128~127
int	16	-32768~32767
float	32	3.4E-38~3.4E+38
double	64	1.7E-308~1.7E+308
void	0	无值

字符类型的变量用于保存8位ASCII字符，整型变量用于保存整数，浮点类型和双精度类型的变量用于保存带小数点的数，但双精度类型比浮点类型有更大的值域。除void类型外，其他类型前面可以加类型修饰符用于改变基本类型的含义，以适用于各种情况下的需要。C++的类型修饰符是：signed(有符号的)、unsigned(无符号的)、short(短整)和long(长整)。修饰符signed和unsigned可以用于字符类型和整型，short和long可以用于整型，long还可以用于双精度类型。表1.3给出了基本数据类型和类型修饰符的组合情况。

表1.3　基本数据类型和类型修饰符

类型名	位数	范围
char	8	-128~127
signed char	8	-128~127
unsigned char	8	0~255
short int	16	-32768~32767
signed short int	16	-32768~32767
unsigned short int	16	0~65535
int	16	-32768~32767
signed int	16	-32768~32767
unsigned int	16	0~65535
long int	32	-2，147，483，648~2，147，483，647
signed long int	32	-2，147，483，648~2，147，483，647
unsigned long int	32	0~4，294，967，295
float	32	3.4E-38~3.4E+38
double	64	1.7E-308~1.7E+308
long double	80	3.4E~1.1E+4932

除了这些基本数据类型以外，C++的数据类型还包括常量(整常量、浮点常量和字符串常量)、类、结构、数组和指针等。其中类、结构本书将在以后进行讨论，本节将讨论表1.2中的简单数据类型和数组与指针。

C++语言在对变量使用之前，必须对此变量进行说明。常用的说明方法为简单说明，

例如：

```
float var1,var2,var3;
```

此语句对变量var1，var2与var3进行了简单说明，声明了三个浮点型变量。下面是C++语言中一些简单说明的例子。

1.3.1.1　整数类型

C++的基本整数类型包括三种：

1. short

short是一种短整型变量，它占据了二个字节的位置，与下面的int整型变量一样，取值范围为-32768~32767。下面是一些short变量的声明：

```
short I=8;
short I，J，K;
```

2. int

int是一种最常用的整型变量，大小为二个字节，取值范围为-32768~32767。下面是一些int变量的声明：

```
int I=9;
int I，J，K;
```

3. long

long是一种长整型变量，大小为四个字节，取值范围为-2,147,483,648~2,147,483,647，下面是一些long变量的声明：

```
long I=298;
long I，J，K;
```

1.3.1.2 浮点数类型

1. float

float是一个占据四个字节数的浮点型变量，取值范围为3.4E-38~3.4E+38。下面是一些float变量的声明：

```
float a=0.9;
float a,b,c;
```

2. double

double是一个八个字节数的双精度浮点型变量，取值范围为1.7E-308~1.7E+308，下面是一些double变量的声明：

```
double a=0.0009;
double a,b,c;
```

1.3.1.3　字符类型

char

C++语言中字符型变量声明只有一种，即char。它只占据一个字节数。标准ASCII字符集的取值范围在0~127之间，因此一个字节数已足以表示所有的ASCII码。下面是一些char型变量的声明：

```
char string1;
char string2='a';
```

```
char string3='\n';
```

1.3.1.4　数组和指针

1. 数组

C++具有使程序员能够定义并使用有序数据的能力，这就是数组。数组是一组相同类型变量的集合，其中的每一个变量称为数组的元素。数组可分为一维数组和多维数组。声明数组的一般形式为：

```
type data[size];
```

其中type是数组元素的类型，但不能是void类型和函数类型，size是整型数，用来说明数组data的大小，方括号是数组类型说明符。在使用数组前，必须对数组进行声明。下面是一些对数组的声明：

```
int num[3];              //声明了一组整数型一维数组
float sum[9];            //声明了一组浮点型一维数组
int trees[3][4];         //声明了一组整数型二维数组
float results[5][6];     //声明了一组浮点型二维数组
```

一维数组包含的元素个数与其下标数一致，而多维数组包含的变量个数是其下标的连乘积。如：

```
int num[3];          //声明了一组包含三个元素的整型数组
float results[5][6]; //声明了一组包含 5×6 个元素的浮点型数组
```

数组的下标应从 0 开始，到其下标减 1 为止。如下面语句：

```
int num[3];
```

其中包含的元素为 num[0]、num[1]、num[2]，而没有 num[3]。若忽视这一点，将使数组的使用产生难以估计的错误。

数组的初始化有许多方式，下面是对于静态数组常用的初始化方式：

```
int num[3]={1,2,3};                          //对整型数组的简单赋值
float results[2][3]={ {1.9,2.4},{3.5,4.1}};  //对浮点型数组的分组赋值
int a[ ]={2,4,5};                            //对整型不定元素个数的数组赋值
int b[10]={1,2,3};                           //对整型数组的不完全赋值
```

由上面代码可以看出，对数组的赋值可以采取固定赋值、分组赋值、不定个数赋值与不完全赋值的方式。其中，在不完全赋值时，剩下的数组元素将被缺省赋值为0。但在大多数情况下，使用前最好将它们赋上初值0，以免发生错误。

2. 指针

指针是C++语言中一种强大而有力的工具，但同时也因为它过于强大的特性，使用指针存在着巨大的危险性。在大多数情况下，指针变量是一种存放内存地址的变量，而该地址是内存中另一个变量的首地址。如果一个变量包含的是另一个变量的地址，可以称第一个变量指向第二变量。如果一个变量要存放另一个变量的地址，这个变量必须声明为指针类型的变量。声明指针变量的一般形式为：

```
type *name;
```

其中name是指针变量的名字，type可以是任何类型，符号“*”在声明语句中是指针类型的说明符，而type用于声明指针可以指向哪种类型的变量，同时规定指针在表达式中参与运算的形式。下面是一个指针变量的声明：

```
int *pointer,      //声明了一个可保存整型变量地址的指针变量
```

关于指针的运算符有三种：取地址运算符*、引用运算符&和下标运算符[ ]。下标运算符已经在数组中用过，下面只介绍地址运算符*和引用运算符&。地址运算符*表示取变量中的内容，引用运算符表示取变量所在的地址。下面是一些指针运算符的例子：

```
#include <iostream.h>
main()
{
    int num=2;                          //声明一个整型变量 num
    int *pointer=&num;                  //声明一个指针变量并把 num 所在的地址赋给 pointer
    cout<<"num="<<num<<"\n";            //输出 num 的值
    cout<<"*pointer="<<pointer<<"\n";   //输出 pointer 的值
    cout<<"*pointer="<<*pointer;        //输出*pointer 的值
}
```

程序运行结果如下：

```
num=2
pointer=0x0065FDF4
*pointer=2
```

由此可见，pointer代表了num的地址，而*pointer则代表了num的内容。

指针变量与其他变量一样，可进行运算，并可用new与delete语句在程序中进行动态分配，这些内容本书将在后面进行介绍。

1.3.2　表达式

运算是对数据进行加工的过程，描述各种不同运算的符号称为运算符，参与运算的的数据称为操作数。运算符按性质分，可分为算术运算符、逻辑运算符、关系运算符、位运算符和其他运算符。表达式是各种运算符和操作数的结合。对一个表达式求值时，存在着优先级和结合性的问题，运算符的优先级和结合性决定着一个表达式的求值顺序。一个表达式的求值顺序影响着程序中变量的数值，进而影响着程序的整个流程。C++语言中包含着一些特殊的运算符和表达式，极大方便了程序员的工作，但同时也容易引起错误。表1.4列出了C++语言的基本运算符和特殊操作符。

表1.4　C++语言的基本运算符和特殊操作符

优先级	运　　算　　符	结合性
1	( )　->　[　]　::　.　*　-　>　*　&(引用)	右结合
2	*(递引用)　&(取地址)new delete！～　++　——　-sizeof	左结合
3	*　/　%	右结合

（续表）

优先级	运　　算　　符	结合性
4	+ —	右结合
5	<< >>	右结合
6	< <= > >=	右结合
7	== !=	右结合
8	&	右结合
9	^	右结合
10	|	右结合
11	&&	右结合
12	||	右结合
13	?:	左结合
14	= += -= *= /= %= <<== >>== &= ^= |=	左结合
15	,	右结合

其中，基本运算符包括算术运算符、逻辑运算符、关系运算符、位运算符、条件运算符、逗号运算符、sizeof运算符，说明如下：

1. 算术运算符

算术运算符包括＋、－、＊、／、％。其中％是取模运算符，a％b的结果是a被b除的余数

2. 逻辑运算符

逻辑运算符包括&&(与)、||(或)、！(非)。

3. 关系运算符

关系运算符包括<(小于)、<＝(小于或等于)、>(大于)、>＝(大于或等于)。

4. 位运算符

位运算符包括&(位与)、｜(位或)、^(位异或)、<<(左移)、>>(右移)、～(求反)。

5. 条件运算符

条件运算符包括？、:。

下面是两个条件运算符的例子：

```
I<j?I+100:j-50      //如果 I<j，则 I+100，否则 j-50
e1?e2:e3            //如果表达式 e1 成立，计算 e2 表达式，否则计算 e3 表达式
```

逗号运算符：　，(只用于隔开几个表达式)

sizeof运算符：　sizeof　(只用于计算它的操作数在内存中所占用的字节数)

1.4　语句和函数

本章介绍C++的流程控制语句和系列封装函数。C++的流程控制语句决定程序执行时的结构，而函数的封装、调用和传递参数也同样影响着程序的流程。