计 算 机 基 础 课 程 系 列 教 材

# Visual C++教程

郑阿奇　主编
丁有和　编著

机械工业出版社
China Machine Press

计算机基础课程系列教材

# Visual C++教程

郑阿奇 主编
丁有和 编著

机械工业出版社
China Machine Press

本书以基本熟悉C为基础，着重介绍C++面向对象程序设计和利用Visual C++ 6.0(中文版)进行应用开发，分为“教程”和“实验与实习”两部分。内容主要包括：C/C++语言概述、C++面向对象程序设计基础、C++面向对象程序设计进阶、对话框、常用控件、框架窗口界面设计、文档和视图、图形和文本、数据库编程以及多媒体应用等。只要阅读本书，结合上机操作指导进行练习，就能在较短的时间内基本掌握Visual C++及其应用技术。

本书可作为大学本科、高职高专有关课程的教材，也可供广大Visual C++ 6.0用户自学和参考。



**图书在版编目`（CIP）数据**

Visual C++教程/郑阿奇主编. -北京：机械工业出版社，2004.9
（计算机基础课程系列教材）
ISBN 7-111-14676-X

Ⅰ.V… Ⅱ.郑… Ⅲ.C语言-程序设计-高等学校-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第057919号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：李云静
北京中兴印刷有限公司印刷 · 新华书店北京发行所发行
2004年9月第1版第1次印刷
787mm × 1092mm 1/16 · 26 印张
印数：0 001 - 5 000 册
定价：35.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

# 前　言

先学C语言后学C++和Visual C++已经成为许多高等学校教学的一种模式，但在C语言的基础上学习C++和Visual C++的好教材很少。我们在汲取以前编写C++和Visual C++教材的成功经验的基础上，结合近年来C++和Visual C++的教学实践，以初步熟悉C语言为基础，编写了本教材。

本教材首先复习C，对于C++中对C的扩展内容进行了特别说明和详细介绍，并通过部分例子进行消化。同时初步熟悉一下Visual C++ 6.0中文版开发环境，并练习控制台应用程序的创建和调试方法。在复习C和C++的基本内容的同时初步跨进了Visual C++之门。然后对C++面向对象程序设计分C++基础和C++进阶两大章进行系统介绍，这里也不急于与Visual C++紧密联系，而是在控制台应用程序方式下调试所有C++上机程序。我们选用的例子既完整又不太大，并加以注释。通过这个过程在逐渐掌握C++的同时，初步熟悉了Visual C++开发环境。从第4章开始逐步进入Visual C++，并不断深入。从简单的Windows编程到MFC，一般在讲解内容后紧跟实例，操作步骤清晰易懂，程序完整且一般都能上机调试通过。这部分的实验与教程配合，一步一步引导读者进行操作和编程，然后提出问题思考和在原来基础上让读者自己进行操作修改和扩充编程练习，最后通过实习进行综合检验。这个过程实际上就是:“模仿”→“理解”→“修改/扩充”→“练习”→“检验”。

本教程不仅适合于教学，也非常适合于用Visual C++ 6.0开发应用程序的用户学习和参考。只要阅读本书，结合上机操作指导进行练习，就能在较短的时间内基本掌握Visual C++及其应用技术。

本书由丁有和（南京师范大学）编写，郑阿奇（南京师范大学）对全书进行统编定稿。郑进、周怡君、胡勇、周俊生等同志对本书的编写提供了帮助，在此一并表示感谢！

本书配有教学课件，读者可到www.hzbook.com本书网页下载。

由于作者水平有限，不当之处在所难免，恳请读者批评指正。

作者E-mail: zhengaqi163@hotmail.com

编　者

2004.5

# 本书约定

1) 在C++的说明语句、运算符、函数、类等格式中，尖括号(< >)中的内容是必需的，方括号([ ])中的内容是可选的。例如：

```
if (<表达式>)       <语句1>
[else        <语句2>]
```

2) 在程序代码中，凡是带底纹的代码都是读者需要手动键入的代码及有关运行结果，并且凡是以Ex_开头的程序都可以上机操作。例如：

```
BOOL CFirstDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    CStatic* pWnd = (CStatic*)GetDlgItem(IDC_STATIC_1);
    pWnd->SetWindowText("这是我的第一个对话框!");
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

3) 若运行过程中还需要用户手工键入数据，则用下划线表示。例如：

```
Please input two integer numbers: 10  123↵
```

其中，“↵”符号表示Enter键（回车键）。

4) 在本教材和实验中，如没有特别的说明，“单击鼠标”、“右击鼠标”以及“双击鼠标”分别表示“单击鼠标左键”、“单击鼠标右键”以及“双击鼠标左键”。

5) 在本教材和实验中，如没有特别的说明，用MFC AppWizard创建应用程序就是在“新建”对话框的“工程”页面中选择“MFC AppWizard(exe)”向导类型来创建应用程序。

# 目　录

## 第二部分 实验与实习



## 附 录

# 第一部分 教 程

## 第 1 章 C/C++语言概述

C语言是在1970年由AT&T贝尔实验室推出的，随着当时微型计算机的日益普及，出现了许多C语言版本，也出现了许多不一致的地方。为了改变这种情况，美国国家标准研究所（ANSI）为C语言制定了一套ANSI标准，成为现行的C语言标准。尽管C语言具有简洁高效、良好的可读性和可移植性等许多优点，但为了能满足运用面向对象方法开发软件的需要，1983年贝尔实验室对C语言进行了扩充和完善，开发出了支持面向对象程序设计的C++语言。

本章主要复习C语言的基础内容，同时，凡C++对C语言扩展的内容将予以注明。需要说明的是，在学习本章内容之前最好先做实验0。

### 1.1 从C到C++的程序结构

和C语言一样，一个C++程序也是由预处理命令、语句、函数、变量（对象）、输入与输出以及注释等几个基本部分组成的。下面先来看一个简单的C++程序（注意与C语言的区别）。

[例Ex_Simple] 一个简单的C++程序

```
// C++程序的基本结构
#include <iostream.h>
void main()
{
    double r, area;                          // 声明变量
    cout<<"输入圆的半径: ";                  // 显示提示信息
    cin>>r;                                  // 从键盘上输入变量r的值
    area = 3.14159 * r * r;                  // 计算面积
    cout<<"圆的面积为: "<<area<<"\n";        // 输出面积
}
```

该程序经编译、连接、运行后，屏幕上显示:

```
输入圆的半径:
```

此时等待用户输入，当输入“10”并按Enter键后，屏幕显示:

```
圆的面积为: 314.159
```

这就是程序运行的过程。

和C语言一样，代码中的main表示主函数，每一个C++程序都必须包含一个且只能包含一个main函数。main函数体是用一对花括号“{”和“}”括起来的，函数体中包括若干条语句，每一条语句都以分号“；”作为结束的标志。

在本例中，main函数体的第1条语句用来声明r和area两个变量；第2条语句是一条输出语句，它将双引号中的内容输出到屏幕上，cout表示标准输出流对象，用于屏幕输出；“<<”是插入符，它将后面的内容插入到cout中，即输出到屏幕上；第3条语句是一条输入语句，cin表示标准输入流对象，用于键盘输入，“>>”是提取符，用来将用户键入的内容保存到后面的变量r中；最后一条语句是采用多个“<<”将字符串和变量area的内容输出到屏幕中，后面的“\n”是换行符，即在输出内容后回车换行。

程序的第2行“#include <iostream.h>”是C++的编译指令，称为预处理命令。iostream.h文件是一个标准输入/输出流的头文件，由于程序中用到了输入/输出流对象cin和cout，故需要包含该头文件。

从代码中可以看出，C++用标准输入输出的头文件iostream.h替代了C语言的stdio.h，用cin、cout和操作运算符>>、<<等实现并扩展了C语言的scanf和printf函数功能。

## 1.2　程序书写规范

C++程序的书写规范基本与C相同。下面从标识符命名、缩进和注释这几个方面进行简单介绍。

**1. 标识符命名**

标识符是用来标识变量名、函数名、数组名、类名、对象名、类型名、文件名等的有效字符序列。标识符命名需要遵守合法性、有效性和易读性的原则。

(1) 合法性

C++规定标识符由大小写字母、数字字符（0~9）和下划线组成，且第一个字符必须为字母或下划线。任何标识符中都不能有空格、标点符号、运算符及其他非法字符。标识符的大小写是有区别的，并且不能和系统的关键字同名，以下是常用的C++标准关键字（C语言本身有32个，斜体为C++新增加的）：

| | | | | | |
|---|---|---|---|---|---|
| *asm* | auto | *bool* | break | case | *catch* |
| char | *class* | const | continue | default | *delete* |
| do | double | else | enum | extern | float |
| for | *frient* | goto | if | *inline* | int |
| long | *mutable* | *namespace* | *new* | *operator* | *private* |
| *protected* | *public* | register | return | short | signed |
| sizeof | static | struct | switch | *template* | *this* |
| *throw* | *try* | typedef | union | unsigned | *using* |
| *virtual* | void | volatile | while | | |

(2) 有效性

虽然，标识符的长度（组成标识符的字符个数）是任意的，但最好不要超过32个，因为有的编译系统只能识别前32个字符，也就是说前32个字符相同的两个不同标识符被有的系统认为是同一个标识符。

(3) 易读性

在定义标识符时，若能做到“见名知意”就可以达到易读性的目的。

另外，为了增强程序的可读性，许多程序员采用“匈牙利标记法”来定义标识符。这种方法是：在每个变量名前面加上表示数据类型的小写字符，变量名中每个单词的首字母均大写。例如：用nWidth或iWidth表示整型（int）变量。

**2. 缩进和注释**

程序在书写时不要将程序的每一行都由第一列开始，而应在语句前面加进一些空格，称为“缩进”，或是在适当的地方加进一些空行，以提高程序的可读性。在本书中，采用下列缩进形式：

每个花括号占一行，并与使用花括号的语句对齐。花括号内的语句采用缩进书写格式，缩进量为四个字符（一个缺省的制表符）。

同缩进的目的一样，注释也是为了提高程序的可读性。注释本身对编译和运行并不起作用。在程序中，凡是放在“/*.....*/”之间或以“//”开头的行尾的内容都是注释的内容，其中，/*.....*/注释方式可以出现在程序中的任何位置。一般来说，注释应在编程的过程中进行，且注释内容一般有：源程序的总体注释（文件名、作用、创建时间、版本、作者及引用的手册、运行环境等）、函数注释（目的、算法、使用的参数和返回值的含义、对环境的一些假设等）及其他的少量注释。一般不要陈述那些一目了然的内容，以免影响注释的效果。

## 1.3 数据类型

和C语言一样，C++的数据类型也分为基本类型、派生类型以及复合类型三类。基本数据类型是C++系统的内部数据类型，如整型、浮点型等。派生类型是将已有的数据类型定义成指针或引用。而复合类型是根据基本类型和派生类型定义的复杂数据类型（又可称为构造类型），如数组、类、结构体和共用体等。

### 1.3.1 基本数据类型

C/C++的基本数据类型有字符型（char）、整型（int）和浮点型（float、double）三种。这些基本数据类型还可用short、long、signed和unsigned来修饰。表1-1列出了C++中的基本数据类型，其字宽（以字节数为单位）和取值范围是在Visual C++ 6.0时的情况。

需要注意的是：

1) C++还可以有布尔型（bool），即值为true或false。事实上，在计算机内，编译系统将true表示成整数1，false表示成整数0，因此也可把布尔型看成是一个整型。

2) 无符号（unsigned）和有符号（signed）的区别在于数值最高位的含义。对于signed类型来说，最高位是符号位，其余各位表示数值大小；而unsigned类型的各个位都用来表示数值大小；因此相同基本数据类型的signed和unsigned的数值范围是不同的。例如，无符号字符型值的范围为0~255，而有符号字符型值的范围为 -128 ~ -127。

3) char、short、int和long可统称为整型。缺省时，char、short、int和long本身是有符号（signed）的。

表1-1　C++的基本数据类型

| 类 型 名 | 类型描述 | 字　宽 | 范　围 |
|---|---|---|---|
| char | 字符型 | 1 | -128~127 |
| unsigned char | 无符号字符型 | 1 | 0~255(0xff) |
| signed char | 有符号字符型(与char相同) | 1 | -128~127 |
| short [int] | 短整型 | 2 | -32 768~32 767 |
| unsigned short [int] | 无符号短整型 | 2 | 0~65 535(0xffff) |
| signed short [int] | 有符号短整型(与short int相同) | 2 | -32 768~32 767 |
| int | 整型 | 4 | -2 147 483 648~2 147 483 647 |
| unsigned [int] | 无符号整型 | 4 | 0~4 294 967 295(0xffffffff) |
| signed [int] | 有符号整型(与int相同) | 4 | -2 147 483 648~2 147 483 647 |
| long [int] | 长整型 | 4 | -2 147 483 648~2 147 483 647 |
| unsigned long [int] | 无符号长整型 | 4 | 0~4 294 967 295(0xffffffff) |
| signed long [int] | 有符号长整型(与long int相同) | 4 | -2 147 483 648~2 147 483 647 |
| float | 单精度浮点型 | 4 | 3.4E-38~3.4E+38 |
| double | 双精度浮点型 | 8 | 1.7E-308~1.7E+308 |
| long double | 长双精度浮点型 | 8 | 1.7E-308~1.7E+308 |

注：表中[int]表示可以省略，即在int之前有signed、unsigned、short、long时，可以省略int关键字。

### 1.3.2 常 量

根据程序中数据的可变性，数据可以分为常量和变量两大类。

在程序运行过程中，其值不能被改变的量称为常量。常量可分为不同的类型，如1、20、0、-6为整型常量，1.2、-3.5为浮点型常量，'a'、'b'为字符常量。常量一般从其字面形式即可判别。

下面简单介绍几种不同数据类型常量的表示方法。

**1. 整型常量**

整型常量可以用十进制、八进制和十六进制来表示。

十进制整型常量即十进制整数，如34、128等。

八进制整型常量是以0开头的数，它由0至7的数字组成。如045，即$(45)_8$，表示八进制数45，等于十进制数37；-023表示八进制数-23，等于十进制数-19。

十六进制整型常量是以0x或0X开头的数，它由0至9、A至F或a至f组成。例如0X7B，即$(7B)_{16}$，等于十进制的123，-0x1a等于十进制的-26。

需要注意的是：

1) 整型常量中的长整型（long）要以L或小写字母l作为结尾，如3276878L、496l等。

2) 整型常量中的无符号型（unsigned）要以U或u作为结尾，如2100U、6u等。

**2. 浮点型常量**

浮点型常量即实数，它有十进制数或指数两种表示形式。

十进制数形式是由整数部分和小数部分组成的（注意必须有小数点）。例如0.12、.12、1.2、12.0、12.、0.0都是十进制数形式。

指数形式采用科学表示法，它能表示出很大或很小的浮点数。例如1.2e9或1.2E9都代表$1.2 \times 10^9$，注意字母E（或e）前必须有数字，且E（或e）后面的指数必须是整数。

若浮点型常量是以F（或f）结尾的，则表示单精度类型（float）；以L（或小写字母l）结尾的，表示长双精度类型（long double）。若一个浮点型常量没有任何说明，则表示双精度类型（double）。

**3. 字符常量**

字符常量是用单引号括起来的一个字符。如‘A’、‘g’、‘%’、‘␣’（␣表示空格，以下同）等都是字符常量。注意‘B’和‘b’是两个不同的字符常量。

除了上述形式的字符常量外，C/C++还可以用一个“\”开头的字符来表示特殊含义的字符常量。例如前例中‘\n’，代表一个换行符，而不是表示字母n。这种将反斜杠(\)后面的字符转换成另外意义的方法称为转义表示法，‘\n’称为转义字符。表1-2列出了常用的转义字符。

**表1-2 常用转义字符**

| 字符形式 | 含 义 |
|---|---|
| \a | 响铃 |
| \b | 退格(相当于按Backspace键) |
| \f | 进纸(仅对打印机有效) |
| \n | 换行 |
| \r | 回车(相当于按Enter键) |
| \t | 水平制表(相当于按Tab键) |
| \v | 垂直制表(仅对打印机有效) |
| \' | 单引号 |
| \" | 双引号 |
| \\ | 反斜杠 |
| \? | 问号 |
| \ooo | 1到3位八进制数所代表的字符 |
| \xhh | 1到2位十六进制数所代表的字符 |

表1-2中，‘\ooo’和‘\xhh’都是用ASCII码表示一个字符，例如‘\101’和‘\x41’都是表示字符‘A’。

**4. 字符串常量**

和C语言一样，C++除了允许使用字符常量外，还允许使用字符串常量。字符串常量是一对双引号括起来的字符序列。例如:

```
"Hello, World!\n"
"C++语言"
"abcdef"
```

等等都是字符串常量。字符串常量中还可以包含空格、转义字符或其他字符。并且必须在同一行书写，若一行写不下，则需要用‘\’来连接，例如:

```
"ABCD\
EFGHIGK..."
```

不要将字符常量和字符串常量相混淆，它们的主要区别如下:

1) 字符常量是用单引号括起来的，仅占一个字节；而字符串常量是用双引号括起来的，至少占用两个字节。例如‘a’是字符常量，占用一个字节用来存放字符a的ASCII码值，而“a”是字符串常量，它的长度不是1而是2，除了字符a之外，它的末尾还有个‘\0’字符。每个字符串的末尾都有一个这样的字符，一定要注意。

2) 字符常量实际上是整型常量的特殊形式，它可以参与常用的算术运算；而字符串常量则不能。例如:

```
int  b='a'+3;              // 结果b为100，这是因为'a'的ASCII码值97参与了运算
```

**5. 符号常量**

在C++中，除了用C语言的#define定义符号常量外，还常常用const来定义符号常量。

### 1.3.3 变量

变量是指在程序执行中其值可以改变的量。变量的作用是存储程序中需要处理的数据，它可以放在程序中的任何位置上。但无论如何，在使用一个变量前必须先定义这个变量。变量有三个基本要素：C++合法的变量名、变量类型和变量的数值。

**1. 变量的定义**

定义变量是用下面的格式语句进行定义的:

```
<类型> <变量名列表>;
```

需要说明的是:

1) 可以将同类型的变量定义在一行语句中，不过变量名要用逗号(,)分隔。但在同一个程序块中，不能有两个相同的变量名。

2) 注意在C++中没有字符串变量类型，字符串是用字符类型的数组或指针来定义的。

3) 与C语言相比，C++变量的定义比较自由。例如，可以在for语句中定义一个变量（以后还会讲到）。

**2. 变量的初始化**

程序中常需要对一些变量预先设置初值，这一过程称为初始化。在C/C++中，可以在定义变量时同时使变量初始化。例如:

```
double    x=1.28;                    // 指定x为双精度浮点变量，初值为1.28
int       nNum1, nNum2=3, nNum3;     // 指定某一个变量的初值
```

C++变量的初始化还有另外一种形式，它与C语言不同。例如:

```
int nX(1), nY(3);
```

表示nX和nY是整型变量，它们的初值分别为1和3。

### 1.3.4 数据类型转换

C/C++采用两种方法对数据类型进行转换，一种是“自动转换”，另一种是“强制类型转换”。

**1. 自动转换**

自动转换是将数据类型从低到高的顺序进行转换，如图1-1所示。

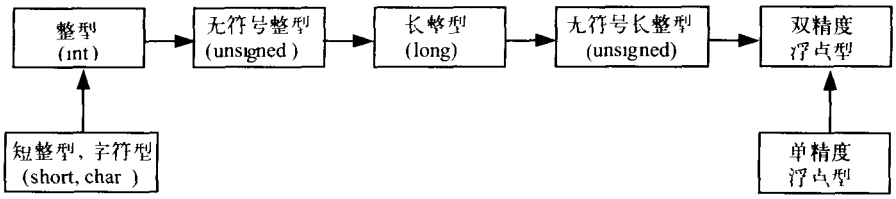


图1-1 类型转换的顺序

**2. 强制类型转换**

强制类型转换是在程序中通过指定数据类型来改变图1-1所示的类型转换顺序，将一个变量从其定义的类型改变为另一种新的类型。强制类型转换有下列两种格式

```
(<类型名>)<表达式>
<类型名>(<表达式>)
```

这里的“类型名”是任何合法的C/C++数据类型，例如float、int等。通过类型的强制转换可以将“表达式”转换成指定的类型。

### 1.3.5 数组

C/C++可以允许程序员按一定的规则进行数据类型的构造，如定义数组类型、枚举类型、结构体类型和共用体类型等，这些类型统称为构造类型。

数组是相同类型的元素的有序集合，每个元素在数组中的位置可用统一的数组名和下标来惟一确定。

**1 数组的定义**

定义一个数组可按下列格式进行

```
<类型> <数组名> [<常量表达式1>][<常量表达式2>].....
```

<数组名>后面的常量表达式用于确定数组的维数和大小。例如：

```
int     a[10];
float   b[2][3];
```

分别定义了整型一维数组a和浮点型二维数组b。

一般地，表示某维大小的常量表达式中不能包含变量，但可以包括常量和符号常量，其值必须是一个确定的整型数值，且数值大于1。

**2. 数组元素的引用**

数组定义后，就可以引用数组中的元素，引用时按下列格式:

```
<数组名> [<下标>].....
```

例如a[0]、b[5]等，这里的0和5是数组的下标，a和b是定义过的数组名。

**3. 数组的赋值**

数组中的元素既可以在数组定义的同时赋初值，即初始化，也可以在定义后赋值。例如：

```
int   b[5]={1, 2};
```

是将数组b的元素b[0]、b[1]分别赋予1、2的值。有时，在对全部数组元素赋初值时，可以不指定数组的长度，例如:

```
int   c[]={1, 2, 3, 4, 5};
```

系统将根据数值的个数自动定义c数组的长度，这里是5。

对于二维或多维数组也可同样进行初始化。需要说明的是:

1) 初始化数组的值的个数不能多于数组元素个数，初始化数组的值也不能通过跳过逗号的方式来省略，这在C中是允许的，但在C++中是不允许的。例如:

```
int f[5] = {1, , 3, 4, 5};              // 错误，初始化值不能省略
int g[5] = {1, 2, 3, };                 // 在Visual C++中正确，它忽略最后一个值的逗号
int h[5] = { };                         // 语法格式错误
```

2) 对于二维数组来说，如果对全部元素都赋初值，则定义数组时对第一维的大小可以忽略，但第二维的大小不能省。例如:

```
int k[][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
// 它等价于int k[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

3) 只对部分元素赋初值，可有两种说明方式: 一种是以“行”为单位，依次列出部分元素的值; 另一种是以数组元素的排列顺序依次列出前面部分元素的值。例如:

```
int m[3][4] = {{1, 2}, {3}, {4, 5, 6}};
```

没有明确列举元素值的元素，其值均为0，即等同于:

```
int m[3][4] = {{1, 2, 0, 0}, {3, 0, 0, 0}, {4, 5, 6, 0}};
```

又如:

```
int n[3][4] = {1, 2, 3};
```

即n[0][0] = 1，n[0][1] = 2，n[0][2] = 3，其余的各个元素的初值为0。

[例Ex_ArraySort] 把5个整数按从小到大的次序排列

```
#include <iostream.h>
const int ARRAYMAX=5;       // 用const定义一个符号常量
void main()
{
     int a[ARRAYMAX]={20, 40, -50, 7, 13};
     // 每次取余下数据的最小一个
     for(int i=0; i<ARRAYMAX; i++){
          for(int j=i+1; j<ARRAYMAX; j++){
               if(a[i]>a[j]){
                    // 交换数据
                    int temp = a[j];      a[j] = a[i];      a[i] = temp;
               }
          }
          cout<<a[i]<<" ";
     }
}
```