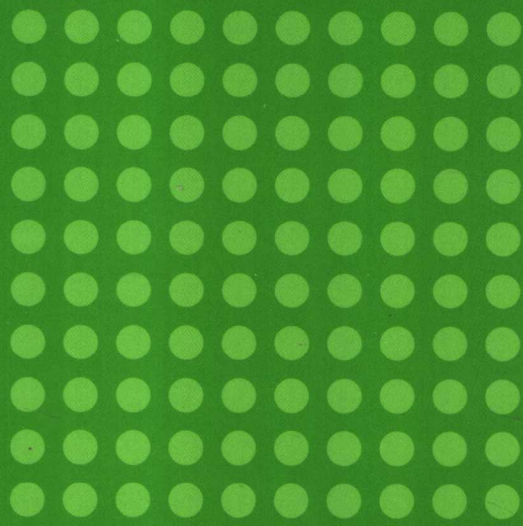


http://www.tup.com.cn

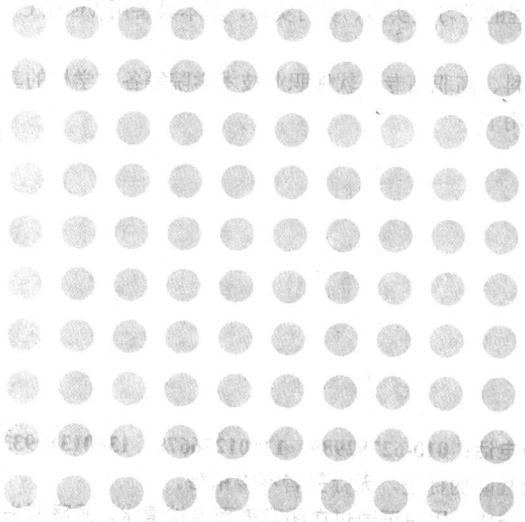
高等院校程序设计规划教材

Visual C++ 教程

郑阿奇 主编　丁有和 编著

清华大学出版社

高等院校程序设计规划教材



Visual C++ 教程

郑阿奇 主编　丁有和 编著

清华大学出版社
北京

内 容 简 介

本教程以 Visual C++ 6.0 中文版为平台，在 C++的基础上，从 Windows 编程入手，系统地介绍了 Visual C++编程基础和应用技术。内容包括：C++基础，Windows 编程基础，对话框，常用控件，菜单、工具栏和状态栏，框架窗口、文档和视图，图形、文本和打印，数据库编程以及 Visual C++高级应用。全书体现很强的应用特色。通过本教程的学习及实验、实习实训，基本具备用 Visual C++设计开发一个小的应用系统的能力。

本教程适合作为大学本科、高职高专、软件职业技术学院等各类学校的教材，也可作为 Visual C++的各类培训和用户学习参考用书。

图书在版编目(CIP)数据

Visual C++教程/郑阿奇主编；丁有和编著. —北京：清华大学出版社，2005.7
（高等院校程序设计规划教材）
ISBN 7-302-11017-4

I. V…　II. ①郑…　②丁…　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（2005）第 049481 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
　　http://www.tup.com.cn　　邮　　编：100084
　　社 总 机：010-62770175　　客户服务：010-62776969
责任编辑：张瑞庆
封面设计：孟繁聪
印 刷 者：北京密云胶印厂
装 订 者：三河市化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：24　字数：593 千字
版　　次：2005 年 7 月第 1 版　2005 年 7 月第1 次印刷
书　　号：ISBN 7-302-11017-4/TP・7301
印　　数：1～5000
定　　价：29.80 元

前言

介绍 Visual C++的书不少，但方便、实用、有特色的 Visual C++教材并不多，可供老师选择的余地比较少，为了解决这个实际问题，我们编写了此书。本系列教程首次提出“教程就是服务”的思想，总结近年来的教学和开发实践经验，以当前最流行的 Visual C++ 6.0 中文版的内容进行组织，详略结合，突出基本。既吸取现有教材中的合理的内容，又对主要内容的介绍有所创新。

为方便教学，本套丛书教学资源十分丰富。Visual C++课程包括以下配套内容。

（1）Visual C++教程：教程以“跟着学→模仿→自己应用”为思路，将使问题简单化。全书体现较强的应用特色，把介绍内容和实际应用有机地结合起来。选用的实例既不太大，这样程序不太长；同时实例又涉及一定的范围和意义，通过实例能消化教程的主要内容。为了解决用户对 Visual C++较高层的内容需要，在介绍有关基本知识后加入一个小规模、可运行的例子来消化、上机实验，以供用户模仿。

（2）Visual C++实训：内容包括实验和实习。实验内容是对教程内容的实训，并且在教程内容的基础上进一步提高。实习从一个应用系统开始逐步设计和组装，并把 Visual C++的基本内容包含进来。通过实验和实习实训，能轻松自如地用 Visual C++设计开发一个小的应用系统。

（3）Visual C++教程课件：在网上同步提供本课件下载。教师可据此备课和教学，它包含了本教程的主要内容，同时附本教程所有实例源代码。

（4）Visual C++应用系统：在网上同步提供包含教程和实验中形成的学生成绩管理系统的所有源文件，以及实习形成的人员信息管理系统的所有源文件。教师可据此在课堂演示，学生可据此上机模仿。

本教程不仅适合于教学，也非常适合于 Visual C++的各类培训和用 Visual C++开发应用程序的用户学习和参考。

本书由南京师范大学丁有和编写，郑阿奇统编、定稿。

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

2005 年 1 月

CONTENTS

目录

CHAPTER 1 第1章

C++基础

在传统的结构化程序设计方法中，数据和处理数据的程序是分离的。当对某段程序进行修改时，整个程序中所有与其相关的部分都要进行相应的修改，从而使程序代码的维护变得比较困难。面向对象的程序设计的本质是把数据和处理数据的函数当成一个整体封装到一个类中，此外类还具有继承和多态的特性。类的实例称为对象。C 是面向过程的语言，C++充分支持面向对象的程序设计。

本章归纳 C++面向对象的程序设计的主要内容，为学习 Visual C++奠定良好的基础。

1.1 简单 C++程序

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。下面看两个简单的 C++程序。

【例 1.1】 一个简单的 C++程序（Ex_Simple）。

```
#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：

注：本书中有灰色底纹的代码是需要用户输入的代码及屏幕显示的相应结果。

```
圆的面积为：314.159
Press any key to continue
```

这就是程序运行的结果。Press any key to continue 是程序运行后系统自动添加的，为简单起见，以后 C++运行结果中的这个内容不再在书中列出。

和 C 语言一样， 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{}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 条语句是用来声明 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，用来将用户输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 1 行#include <iostream.h>是 C++的编译指令，称为预处理指令。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。

下面再来看一个 C++程序。

【例 1.2】 一个求解一元二次方程的 C++程序（Ex_Root）。

```
#include <iostream.h>
#include <math.h>
int GetRoot(float a, float b, float c, double *root);    /* 声明一个全局函数 */
void main()
{
    float a = 2.0, b = 6.0, c = 3.0;                     // 定义并初始化变量
    double root[2];
    int n = GetRoot(a, b, c, root);                      // 调用函数
    if (n<1)
        cout<<"方程无根！";
    else {
        cout<<"方程有 "<<n<<" 根:\n";
        for (int i=0; i<n; i++)                          // 循环输出所有的根
            cout<<"根"<<i+1<<": "<<root[i]<<"\t";
    }
    cout<<endl;
}
// 求一元二次方程的根，函数返回根的个数
int GetRoot(float a, float b, float c, double *root)
{
    double delta, deltasqrt;
```

```
    delta = b*b - 4.0 * a * c;
    if (delta<0.0) return 0;                    // 无根
    deltasqrt = sqrt(delta);
    if (a!=0.0)
    {
        root[0] = (-b + deltasqrt)/(2.0 * a);
        root[1] = (-b - deltasqrt)/(2.0 * a);
    } else {
        if (b!=0.0)  root[0] = root[1] = -c/b;
        else   return 0;
    }
    if (root[0] == root[1]) return 1;
    else return 2;
}
```

程序运行后，结果如下：

```
方程有 2 根:
根1: - 0.633975   根2: -2.36603
```

程序中，GetRoot 是自己定义的函数，用来求解一个指定系数的一元二次方程的根。sqrt是一个库函数，用来返回一个数的平方根，使用该库函数时需要添加包含头文件math.h。

从上面两个程序可以看出，如果 C++程序不涉及“类”的概念，则其基本结构和语法是和 C 语言基本相同的，为此本书将 C++这些基本知识点列在附录中，以便查阅。

1.2　类和对象

类是面向对象程序设计的核心，它实际上是一种新的数据类型。类是对某一类对象的抽象，而对象是某一种类的实例，因此，类和对象是密切相关的。

1.2.1　从结构到类

在讨论类之前，先来看一个 C++结构类型示例，该类型的成员有学生姓名、学号、3门课成绩。

【例 1.3】 从结构到类的示例（Ex_StructToClass）。

```
#include <iostream.h>
struct STUSCORE {
    char  strName[12];              // 姓名
    char  strStuNO[9];              // 学号
    float  fScore[3];               // 3门课程成绩
};
float GetAverage(STUSCORE one)      // 计算平均成绩
```

```
{
    return (float)((one.fScore[0] + one.fScore[1] + one.fScore[2])/3.0);
}
void main()
{
    STUSCORE one={"LiMing", "21020501", {80,90,65}};
    cout<<one.strName<<" 的平均成绩为：  "<<GetAverage(one)<<"\n";
}
```

运行结果如下：

```
LiMing 的平均成绩为：  78.3333
```

从上述示例可以看出，学生 3 门课的平均成绩是通过程序中的全局函数 GetAverage 来计算，那么能否将此函数作为结构的一个成员呢？这时就需要使用“类”了。

1.2.2 类的定义

类的定义一般地分为声明部分和实现部分。声明部分是用来声明该类中的成员，包含数据成员的声明和成员函数的声明。成员函数是用来对数据成员进行操作的，又称为“方法”。实现部分是用来对成员函数的定义。概括地说，声明部分将告诉使用者“干什么”，而实现部分是告诉使用者“怎么干”。

C++中定义类的一般格式如下：

```
class <类名>
{
    private:
        [<私有数据和函数>]
    public:
        [<公有数据和函数>]
};
<各个成员函数的实现>
```

类体

其中，class 是定义类的关键字，class 的后面是用户定义的类名，通常用大写的 C 字母开始的标识符作为类名，C 用来表示类（Class），以与对象、函数及其他数据类型相区别。类中的数据和函数是类的成员，分别称为数据成员和成员函数。需要说明的是，由于数据成员是用变量来描述的，因此数据成员又可称为“成员变量”。

类中关键字 public 和 private 声明了类中的成员和程序其他部分的关系。对于 public 类成员来说，它们是公有的，能被外面的程序访问；对于 private 类成员来说，它们是私有的，只能由类中的函数所使用，而不能被外面的程序所访问。

<各个成员函数的实现>是类定义中的实现部分，这部分包含所有在类体中声明的函数的定义（即对成员函数的实现）。如果一个成员函数在类体中定义，实现部分将不出现。如果所有的成员函数都在类体中定义，则实现部分可以省略。需要说明的是，当类的成员函数的函数体在类的外部定义时，必须由作用域运算符::来通知编译系统该函数所属的类。

例如，下面定义的 CStuScore 类包含了 SetScore 和 GetAverage 成员函数，分别用来输

入成绩和返回计算后的平均成绩：

```
class CStuScore
{
public:                                          // 公有类型声明
    char  strName[12];                           // 姓名
    char  strStuNO[9];                           // 学号
    void  SetScore(float s0, float s1, float s2) // 成员函数：设置3门课成绩
    {
        fScore[0] = s0;        fScore[1] = s1;        fScore[2] = s2;
    }
    float  GetAverage();
private:                                         // 私有类型声明
    float  fScore[3];                            // 3门课程成绩
};                                               // 注意分号不能省略
float CStuScore::GetAverage()
{
    return (float)((fScore[0] + fScore[1] + fScore[2])/3.0);
}
```

类 CStuScore 中，成员函数 SetScore 是在类体中定义的，而 GetAverage 是类的外部定义的，注意两者的区别。另外，定义类时还应注意：

- 在 public:或 private:后面定义的所有成员都是公有或私有的，直到下一个 public:或 private:出现为止。若成员前面没有类似 public:或 private:，则所定义的成员是 private（私有），这是类的默认设置。
- 关键字 public 和 private 可以在类中出现多次，且前后的顺序没有关系。但最好先声明公有成员，后声明私有成员，因为 public 成员是用户最关心的。
- 除了 public 和 private 外，关键字 protected（保护）也可修饰成员的类型，它与 private 两者基本相似，但在类的继承时有所不同。
- 数据成员的类型可以是任意的，包含整型、浮点型、字符型、数组、指针等，也可以是另一个类的对象，但不允许对所定义的成员变量进行初始化。
- 尽量将类单独存放在一个文件中，或将类的声明放在.h 文件中，而将成员函数的实现放在与.h 文件同名的.cpp 文件中。以后将会看到，Visual C++ 6.0 为用户创建的应用程序框架中都是将各个类以.h 和同名的.cpp 文件来组织的。

1.2.3 对象的定义

一个类定义后，就可以定义该类的对象，如下面的格式：

```
<类名> <对象名列表>
```

其中，类名是用户已定义过的类的标识符，对象名可以有一个或多个，多个时要用逗号分隔。被定义的对象既可以是一个普通对象，也可以是一个数组对象或指针对象。例如：

```
CStuScore one, *two, three[2];
```

这时，one 是类 CStuScore 的一个普通对象，two 和 three 分别是该类的一个指针对象和数组对象。

一个对象的成员就是该对象的类所定义的成员，引用（访问）时可用下列方式：

<对象名>.<成员名>
<对象名>.<成员名>(<参数表>)

前者用来表示引用数据成员，后者用来表示引用成员函数。“.”是一个成员运算符，用来引用对象的成员。如：

```
one.strName,  three[0].GetAverage();
```

对于指针对象的成员引用可用下列方式：

<对象指针名>-><成员名>
<对象指针名>-><成员名>(<参数表>)

“->”也是一个成员运算符，它与“.”运算符的区别是：运算符“->”用来访问指针对象的成员，而“.”用来访问一般对象的成员。需要说明的是，下面的两种表示是等价的：

<对象指针名>-><成员名>
(*<对象指针名>).<成员名>

这对于成员函数也适用，例如 two->GetAverage()和(*two).GetAverage()是等价的，由于成员运算符“.”的优先级比取内容运算符“*”高，因此需要在*two 两边加上圆括号。

从上可以看出，C++的结构类型和类在很大程度上都是相同的。正因为如此，C++对结构类型进行了扩展，使其成为类的一种特殊形式。也就是说，在 C++的结构类型定义中除了 struct 和 class 关键词不同外，其余都相同。只不过，若结构成员前面没有 public:或 private:，则所定义的成员默认为 public（公有）类型。

1.3 类的成员及特性

C++的类有其独特的成员函数和特性，如构造函数、析构函数、常类型等，本节阐述这些内容。

1.3.1 构造函数

由于在类的定义中是不能对数据成员进行初始化的，因此为了能给数据成员自动设置某些初始值，这时就要使用类的特殊成员函数——构造函数。构造函数的最大特点是在对象建立时它会被自动执行，因此用于变量、对象的初始化代码一般放在构造函数中。

C++规定：构造函数必须与相应的类同名，它可以带参数，也可以不带参数，与一般的成员函数定义相同，而且可以重载。所谓重载，即允许多个同名的函数存在，但同名的各个函数的形参必须有所区别，要么是形参的个数不同，要么是形参的个数相同，而参数

类型有所不同。构造函数不能指定函数返回值的类型，也不能指定为 void 类型。例如：

```
class CStuScore
{
public:
    CStuScore(char str[12])                    // 第一个构造函数
    {   strcpy(strName, str);
    }
    CStuScore(char str[12], char strNO[9])     // 第二个构造函数
    {   strcpy(strName, str);
        strcpy(strStuNO, strNO);
    }
    char  strName[12];                         // 姓名
    char  strStuNO[9];                         // 学号class  CMeter
    ...
}
```

说明：

- 程序中的 strcpy 是 C++的一个库函数，用来复制字符串，使用时需要头文件 string.h。
- 实际上，在类定义时，如果没有定义任何构造函数，则编译器自动为类生成一个不带任何参数的默认构造函数。对于 CStuScore 类来说，默认构造函数的形式如下：

  ```
  CStuScore( )              // 默认构造函数的形式
  { }
  ```

- 由于构造函数的参数只能在定义对象时指定，因此有：

  ```
  CStuScore oOne("LiMing");
  ```

  它是自动调用第一个构造函数，使得 strName 内容为 LiMing。若有：

  ```
  CStuScore oTwo;
  ```

  则编译器会给出错误的提示，因为类 CStuScore 中已经定义了构造函数，默认构造函数将不再显式调用，因此在类中还要给出默认构造函数的定义，这样才能对 oTwo 进行初始化，此时对象的所有数据成员（成员变量）都被初始化为零或空。

1.3.2 析构函数

与构造函数相对应的另一种特殊的 C++成员函数是析构函数，它的功能是用来释放一个对象，在对象删除前，用它来做一些清理工作，它与构造函数的功能正好相反。

析构函数也要与相应的类同名，并在名称前面加上一个~符号。每一个类只有惟一的一个析构函数，没有任何参数，也不返回任何值，也不能被重载。例如：

```
class  CStuScore
```