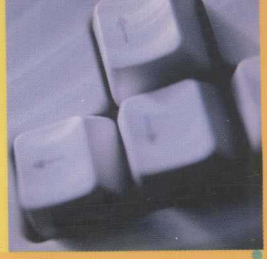
普通高等教育“十五”国家级规划教材

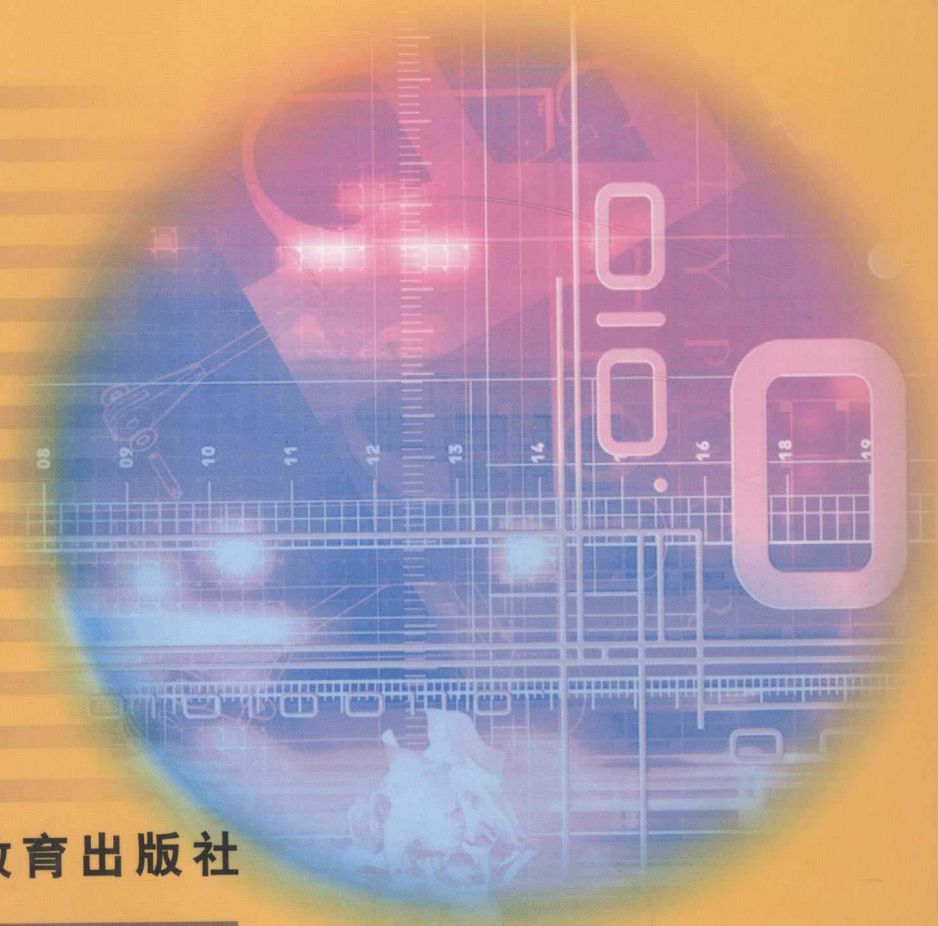
高等职业教育技能型紧缺人才培养试用



程序设计基础
——可视化及VC++实现

周晓云 主 编

陶 霖 陆 虹 副主编



高等教育出版社

普通高等教育“十五”国家级规划教材

高等职业教育技能型紧缺人才培养试用

程序设计基础

——可视化及VC++实现

周晓云　主编

陶　霖　陆　虹　副主编

高等教育出版社

内容提要

本书是普通高等教育“十五”国家级规划教材。作为C语言系列教材，本书是继程序设计基础——逻辑编程及C++实现、程序设计基础——面向对象及C++实现后的第3本。

本书共7章，主要内容包括：MFC编程基础、MFC应用程序初探、基于对话框的MFC应用程序、深入讨论对话框应用程序、单文档窗口应用程序、深入讨论单文档应用程序、调试。

本书适合于高等职业学校、高等专科学校、成人高校、本科院校举办的二级职业技术学院使用，也可供示范性软件职业技术学院、继续教育学院、民办高校、技能型紧缺人才培养使用，还可供本科院校、计算机专业人员和爱好者参考使用。

图书在版编目（CIP）数据

程序设计基础——可视化及VC++实现 / 周晓云主编.
北京：高等教育出版社，2004.7
ISBN 7-04-014769-6

Ⅰ.程…Ⅱ.周…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052233号

策划编辑	冯 英	**责任编辑**	严 亮	**封面设计**	王凌波
版式设计	胡志萍	**责任校对**	朱惠芳	**责任印制**	孔 源

出版发行	高等教育出版社	**购书热线**	010-64054588
社 址	北京市西城区德外大街4号	**免费咨询**	800-810-0598
邮政编码	100011	**网 址**	http://www.hep.edu.cn
总 机	010-82028899		http://www.hep.com.cn
经 销	新华书店北京发行所		
印 刷	北京铭成印刷有限公司		
开 本	787×1092 1/16	**版 次**	2004年7月第1版
印 张	13.25	**印 次**	2004年7月第1次印刷
字 数	310 000	**定 价**	17.00元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

出 版 说 明

为加强高职高专教育的教材建设工作，2000年教育部高等教育司颁发了《关于加强高职高专教育教材建设的若干意见》（教高司[2000]19号），提出了“力争经过5年的努力，编写、出版500本左右高职高专教育规划教材”的目标，并将高职高专教育规划教材的建设工作分为两步实施：先用2至3年时间，在继承原有教材建设成果的基础上，充分汲取近年来高职高专院校在探索培养高等技术应用性专门人才和教材建设方面取得的成功经验，解决好高职高专教育教材的有无问题；然后，再用2至3年的时间，在实施《新世纪高职高专教育人才培养模式和教学内容体系改革与建设项目计划》立项研究的基础上，推出一批特色鲜明的高质量的高职高专教育教材。根据这一精神，有关院校和出版社从2000年秋季开始，积极组织编写和出版了一批“教育部高职高专规划教材”。这些高职高专规划教材是依据1999年教育部组织制定的《高职高专教育基础课程教学基本要求》（草案）和《高职高专教育专业人才培养目标及规格》（草案）编写的，随着这些教材的陆续出版，基本上解决了高职高专教材的有无问题，完成了教育部高职高专规划教材建设工作的第一步。

2002年教育部确定了普通高等教育“十五”国家级教材规划选题，将高职高专教育规划教材纳入其中。“十五”国家级规划教材的建设将以“实施精品战略，抓好重点规划”为指导方针，重点抓好公共基础课、专业基础课和专业主干课教材的建设，特别要注意选择一部分原来基础较好的优秀教材进行修订使其逐步形成精品教材；同时还要扩大教材品种，实现教材系列配套，并处理好教材的统一性与多样化、基本教材与辅助教材、文字教材与软件教材的关系，在此基础上形成特色鲜明、一纲多本、优化配套的高职高专教育教材体系。

普通高等教育“十五”国家级规划教材（高职高专教育）适用于高等职业学校、高等专科学校、成人高校及本科院校举办的二级职业技术学院、继续教育学院和民办高校使用。

教育部高等教育司

2002年11月30日

前　言

本书是一本介绍 Visual C++程序设计的教材。学习本书应具备 C 语言程序设计基础知识，如对面向对象程序设计方法已有基本了解，则可以快速浏览第 1 章的内容，或跳过第 1 章直接进入第 2 章的学习；否则应仔细阅读第 1 章的内容，为学习 MFC（Microsoft Foundation Class Library）打好基础。

Windows 应用程序设计，彻底改变了传统的 DOS 应用程序的设计模式。在 DOS 模式下，应用程序是封闭的，从入口到出口，层层调用，环环相扣，构成一条完整的程序链。而 Windows 应用程序则采用了一种“事件驱动，消息激励”机制。在这种机制下，传统的程序结构形式被打破，大量的响应函数构成了程序的主体，当系统或用户触发了某个事件（如定时时间到、用户点击鼠标等），便会产生与该事件关联的消息，然后通过消息去激励相应的响应函数。从系统管理的角度看，这套机制有其合理性和有效性，但这使应用程序的结构发生了根本的变化，要求应用程序在运行机制上必须与系统实现无缝衔接，这无疑大大增加了编程的难度和工作量。为此，Visual C++提供了一个可视化的编程环境，帮助用户生成应用程序的结构框架，解决了应用程序启动、运行和退出机制的细节问题，以减轻编程人员的负担。值得庆幸的是，Visual C++内置了一个高度面向对象的基本类库，即 MFC。功能强大的 MFC 犹如浩瀚的海洋，其范围之广，几乎涉及了 Visual C++的各种应用。使用 MFC 能极大地提高编程的效率，并有利于提高程序的稳定性和可移植性。因此，目前许多专业程序员都使用 MFC 开发 Windows 应用程序。

根据编者多年从事教学和项目实践的经验，感到学习 MFC 最大的障碍是如何入门。MFC 经过层层封装，其内层已被封得严严实实，对于初学者来说总有窥一斑而难知全貌、知其然而不知其所以然的感觉，入门难度不言而喻。

因此，本书将充分考虑高职高专学生的特点，从实用的角度介绍 MFC 编程的基本方法，避免孤立、分散地介绍知识点，寓知识点于实例之中。通过一些典型实例的分析，对 MFC 中最常用的类进行详细剖析，揭示 MFC 应用程序编制的过程。本书的宗旨是：不求全、但求精，力图使学生尽快入门，能编写一些简单的应用程序，为进一步学习 MFC、运用 MFC 编写复杂的应用程序打下坚实的基础。

本书为有关章节配备了一些上机实训题，这些实训题一般可以通过模仿书中的例题完成，但又留有一定的思索空间。我们期望通过上机实训，能起到使学生举一反三、融会贯通的作用。

本书由上海第二工业大学周晓云担任主编，陶霖、陆虹担任副主编。本书的第 2、6、7 章由李丽萍编写，其他章节由周晓云编写。

限于编者水平，书中难免存在疏漏和不足之处，恳请广大师生和读者批评指正。

编　者

2004.5.25

目　录

第1章　MFC编程基础

本章提要	本章从封装性、继承性和多态性三个方面对面向对象程序设计方法作了简要的阐述，并通过具体示例揭示了面向对象程序设计的一般过程，为更好地学习、使用 MFC 类库进行 Windows 应用程序的开发打下基础。

Visual C++作为 Windows 应用程序设计领域中的主流，不但支持 C++语言的全部功能，而且还提供了完善的可视化的开发环境，集方便、灵活及高效于一体，成为深受专业程序设计人员青睐的开发设计工具。

Visual C++支持面向对象程序设计方法，其核心是一个高度面向对象、功能十分强大的 MFC（Microsoft Foundation Class Library）类库。本书并不是一本面向对象程序设计的入门教材，不打算对此花过多的篇幅。但出于更好地理解和使用 MFC 类库方面的考虑，本章将对面向对象程序设计方法作一个简要的描述和回顾。

1.1　面向对象程序设计方法

面向对象程序设计思想来自于对现实世界的认识和理解，现实世界是由各种各样的对象组成，同类型的对象，经过抽象，可以用一组相同的、能反映它们共同形态的属性和行为来描述。

在 C++语言中，面向对象程序设计方法是通过“类”的机制实现的。“类”作为一种用户自定义的数据类型，包含了两个部分：数据和函数。其中数据称作类的数据成员，它反映了对象的属性；函数称作类的成员函数，它反映了对象的行为。“类”是一种数据类型，用“类”定义的变量被称作对象。其实“类”并不神秘，它脱胎于 C 语言中的结构类型，是结构的拓展和延伸。

下面通过实例介绍有关类和对象的基本概念，演示如何创建、使用对象。对于具有 C 语言基础的人来说，跨入 C++殿堂、掌握面向对象程序设计方法，只不过一步之遥。

1.1.1　类的创建

考虑一个简单的例子：输入一个计算表达式，该表达式包含两个实数和一个运算符（+、−、*和/）。现编写一个类，计算表达式的值。头文件如下：

```
//-----------------------------------------------------------------------------------------------
//                                        实例 1_1
//                          文件名：ch1_1.h
//                          功　能：创建一个类，计算表达式的值
//-----------------------------------------------------------------------------------------------
```

```
#include <iostream.h>
#include <stdlib.h>

class calculate                    // 定义一个类，类名为：calculate
{
  public:                          // 定义公有访问控制属性
    void Init(char *expn);         // 初始化对象，从表达式中抽取参数并保存
    double GetResult();            // 根据保存的参数计算表达式的值
  protected:
    double n1, n2;                 // 保存表达式中两个参与运算的实数
    char c;                        // 保存表达式中的运算符
};

void calculate::Init(char *expn)
{
  int i;
  char str[20];
  for(i=0;
      *expn!='\0' && *expn!='+' && *expn!='-' && *expn!='*' && *expn!='/';
      str[i++]=*expn++);                    // 抽取第1个参与运算的数据
  str[i]='\0';
  n1=atof(str);                             // 将字符串转换为实数并保存到n1中
  c=*expn++;                                // 抽取运算符并保存到变量c中
  for(i=0; *expn!='\0'; str[i++]=*expn++);  // 抽取第2个参与运算的数据
  str[i]='\0';
  n2=atof(str);                             // 将字符串转换为实数并保存到n2中
}

double calculate::GetResult()
{
  switch(c)
  {
    case '+':                       // 运算符为“+”号
      return n1+n2;
    case '-':                       // 运算符为“-”号
      return n1-n2;
    case '*':                       // 运算符为“*”号
```

```
            return n1*n2;
        case '/':                          // 运算符为“/”号
            if(n2 = = 0)   break;
            return n1/n2;
    }
    cout <<"表达式出错！ \n";              // 非法的运算符
    return 0;
}
```

实例分析：

上述程序创建了一个 calculate 类，创建的步骤如下：

（1）本例面向的是计算表达式对象，该类表达式由 2 个参与运算的数值和 1 个决定运算规则的运算符组成，这 3 个参数决定了表达式的形态和计算结果，这些参数被称作对象的属性。在 calculate 类中，3 个数据成员 n1、n2 和 c 用于保存对象的属性；

（2）从表达式中抽取运算数据和运算符、计算出表达式的结果，需要有相应的操作函数来实现。为此，在 calculate 类中添加成员函数 Init()和 GetResult()，前者完成参数抽取并将抽取的结果保存到相应的数据成员中，而后者则根据数据成员的值实现计算功能；

（3）决定访问控制属性。按照常规，一般将数据成员定义为保护类型（Protected），以保护数据不受意外的破坏，成员函数则定义为公有类型（Public），作为对象向外开放的接口，使外部程序能通过这些接口操纵对象。

需要说明的是，建立类的过程是一个抽象的过程，同一个问题按什么原则抽象，往往因人而异。至于哪些部分向外开放、哪些部分隐蔽，同样应根据具体情况而定。

1.1.2 对象的定义

从实例 1_1 可看出，类是实际对象的抽象，是对用户定义的数据类型以及对数据操作方式的描述。对象则是类的实例化，是实际的变量。下面的例子将演示如何使用实例 1_1 创建的 calculate 类。

```
//-------------------------------------------------------------------------------------
//                                  实例 1_2
//              文件名：ch1_2.cpp
//              功  能：从键盘输入一个计算表达式，输出表达式的值
//-------------------------------------------------------------------------------------
    #include "ch1_1.h"                     // 该头文件含有 calculate 类的定义

    void main( )
    {
      char str[40];
      calculate cal1;                      // 定义第 1 个对象 cal1
```

```
    calculate cal2;                    // 定义第 2 个对象 cal2
    cout <<"请输入第 1 个表达式：";
    cin.getline(str, 39);
    cal1.Init(str);                    // 初始化第 1 个对象
    cout << "="<<cal1.GetResult( );    // 得到第 1 个对象的计算结果
    cout << "\n 请输入第 2 个表达式：";
    cout << flush;
    cin.getline(str, 39 );
    cal2.Init(str);                    // 初始化第 2 个对象
    cout << "="<<cal2.GetResult( );    // 得到第 2 个对象的计算结果
    cout << endl;
}
```

实例分析：

（1）本例定义了两个 calculate 类型的对象 cal1 和 cal2。生成应用程序时，类并不占用内存空间，只有创建了对象，系统才会根据类的定义为对象开辟相应的存储空间，用于保存对象的数据。

（2）每个对象都拥有自己的数据存储区域，即自己的数据成员，各自独立，互不干扰。键盘输入的第 1 个表达式的参数保存在对象 cal1 的数据区中，而第 2 个表达式的参数则被保存在对象 cal2 的数据区中。

（3）运行本程序后，如输入两个不同的表达式，则对象 cal1、cal2 通过调用 Init()、GetSult() 成员函数，得到各自不同的数据及运算结果，反映出对象的个性；如果输入两个相同的表达式，则对象 cal1、cal2 会有相同的数据及运算结果，反映出对象的确定性和惟一性。

下面是程序运行的结果：

```
请输入第 1 个表达式：128 / 3
=42.6667
请输入第 2 个表达式：12.4 * 4.2
=52.08
```

注：带下划线的内容表示通过键盘输入，下同。

从 C++语法的角度看，对象是一种变量，因此可以像其他变量一样，根据需要定义对象数组、对象指针及对象指针数组等。

1.1.3 构造函数和析构函数

类有两个特殊的成员函数：构造函数和析构函数。构造函数与类同名，并且没有返回值，它在创建一个对象时，由系统自动调用，一般用来执行对象的初始化操作。析构函数也与类同名，但必须加前缀“~”。析构函数也没有返回值，它在一个对象释放时（如退出对象作用范围

时）由系统自动调用。

下面对实例 1_1 略作改动，使用构造函数和析构函数来完成成员函数 Init( )、GetResult( ) 的操作。

```
//-------------------------------------------------------------------------------------------
//                                    实例 1_3
//                        文件名：ch1_3.h
//                        功  能：创建一个类，计算表达式的值
//                        说  明：使用构造函数和析构函数
//-------------------------------------------------------------------------------------------
    #include <iostream.h>
    #include <stdlib.h>

    class calculate                  // 定义一个类，类名为：calculate
    {
      public:                        // 定义公有访问控制属性
        calculate(char *expn);       // 构造函数，从表达式中抽取参数并保存
        ~calculate ( );              // 析构函数，根据保存的参数计算表达式的值
      protected:
        double n1, n2;               // 保存表达式中 2 个参与运算的实数
        char c;                      // 保存表达式中的运算符
    };

    calculate:: calculate (char *expn)        // 使用构造函数初始化数据成员
    {
      int i;
      char str[20];
      for(i=0;
          *expn!='\0' && *expn!='+' && *expn!='-' && *expn!='*' && *expn!='/';
          str[i++]=*expn++);                  // 抽取第 1 个参与运算的数据
      str[i]='\0';
      n1=atof(str);                           // 将字符串转换为实数并保存到 n1 中
      c=*expn++;                              // 抽取运算符并保存到变量 c 中
      for(i=0; *expn!='\0'; str[i++]=*expn++); // 抽取第 2 个参与运算的数据
      str[i]='\0';
      n2=atof(str);                           // 将字符串转换为实数并保存到 n2 中
    }

    calculate:: ~calculate()                  // 使用析构函数执行计算操作
```

```
    {
      switch(c)
      {
        case '+':                              // 运算符为“+”号
          cout << "=" << n1+n2 << endl;
          return;
        case '-':                              // 运算符为“-”号
          cout << "=" << n1-n2 << endl;
          return;
        case '*':                              // 运算符为“*”号
          cout << "=" << n1*n2 << endl;
          return;
        case '/':                              // 运算符为“/”号
          if(n2 = = 0)   break;
          cout << "=" << n1/n2 << endl;
          return;
      }
      cout << "表达式出错！\n";                // 非法的运算符
    }
```

相应的主函数为：

```
//-------------------------------------------------------------------------------------------
//                                    实例 1_4
//              文件名：ch1_4.cpp
//              功  能：从键盘输入一个计算表达式，输出表达式的值
//              说  明：使用构造函数和析构函数
//-------------------------------------------------------------------------------------------
    #include "ch1_3.h"              // 该头文件含有 calculate 类的定义
    void main( )
    {
      char str[40];
      cout << "请输入表达式：";
      cin.getline(str, 39);
      calculate cal(str);      // 构造函数被自动调用，str 为构造函数所需的输入参数
      // 注意当程序退出主函数时，析构函数被自动调用
    }
```

程序运行结果如下：

```
请输入表达式：64/4
=16
```

1.2 面向对象程序的特性

1.2.1 封装性

封装是面向对象程序设计方法的基本出发点，是面向对象程序的基本特性。所谓封装，就是从实际功能出发，将数据和对数据进行操作的函数结合成一个整体，并通过访问控制机制，对外隐蔽其实现功能的细节，用户只须按预定的规则调用其公开的接口即能得到预期的结果。

C++语言通过创建类来实现封装。封装时一般应注意以下几点：

（1）数据成员：传统的程序设计方法往往注重代码设计，把数据放在从属的地位；而面向对象程序设计方法以数据为中心，强调以数据引导代码。因此，在创建类的时候，首先应仔细分析面临的实际问题，抽象出各种情况下具体实例的共同属性，并为之定义相应的数据成员。例如，对于满足实例 1_1 要求的计算表达式，不管它如何变化，总可以抽象为 2 个实数和 1 个运算符，回顾实例 1_1 不难看出，整个 calculate 类就是围绕这 3 个数据设计的。

（2）成员函数：如果说数据成员决定了对象的状态和特征，那么对象的行为规律则取决于成员函数。成员函数的设计，既要考虑到实际功能的需要，又要体现以数据为中心的思想。以实例 1_1 为例，成员函数 Init( )从输入表达式中抽取数据和运算符，并保存到数据成员 n1、n2 和 c 中，数据成员是该函数行为的终点；而成员函数 GetResult( )则从数据成员 n1、n2 和 c 中获取数据进行计算，并返回计算的结果，数据成员是该函数行为的起点。这两个函数有各自的功能，同时又围绕数据成员 n1、n2 和 c 有机地结合起来，实现一个从输入表达式到输出计算结果的完整功能，数据成员就是这 2 个成员函数的纽带，是整个类的中心。

（3）访问控制：设置访问控制属性是为了隐蔽对象内部的细节，保护重要的数据不受破坏。C++语言中访问控制类型有 3 种：公有类型（Public）、保护类型（Protected）和私有类型（Private）。3 种类型区别如表 1-1 所示。

表 1-1 访问控制类型

类 型	说 明
Public	本类型的成员可供类外部、内部及派生类的函数访问
Protected	本类型的成员可供类内部及派生类的函数访问
Private	本类型的成员仅供类内部的函数访问

访问控制是封装的重要手段。对于需要隐蔽的信息，可以设置为私有类型；提供给外部访

问、并希望通过它们操纵对象的成员，应该设置为公有类型；至于那些需要隐蔽，但又希望给派生类留有直接访问途径的成员，则可以设置为保护类型。

下面的实例进一步演示如何通过创建类来实现封装。该类的功能是：从键盘输入 2 个整数，找出这 2 个整数之间所有各位数字之和等于 9 的整数。

```
//-------------------------------------------------------------------------------------------
//                                    实例 1_5
//          文件名：ch1_5.h
//          功  能：找出指定范围内所有各位数字之和为 9 的整数
//-------------------------------------------------------------------------------------------
    #include <iostream.h>

    class find
    {
      public:
        find( );                    // 构造函数，初始化对象
        int GetData(int d[ ]);      // 得到结果数据，返回值为数据的个数
      protected:
        int Process( );             // 查找满足条件的数据，结果存于 data 中
        int n;                      // 存放满足条件数据的个数
        int n1, n2;                 // 存放键盘输入的 2 个整数，即指定的范围
        int data[100];              // 存放查找到的数据
    };

    find::find( )
    {
      int t;
      cout << "请输入第 1 个整数：";
      cin >> n1;
      cout << "请输入第 2 个整数：";
      cin >> n2;
      if(n1>n2)                     // 查找的范围存于 n1、n2 中，且保证 n1<=n2
      {t=n1;  n1=n2;  n2=t;}
      n=0;                          // 查找前将 n 清零
      n=Process( );                 // 查找处理，返回满足条件数据的个数
    }

    int find::GetData(int d[ ])
    {
```

```
    int i;
    for(i=0; i<n; i++)          // 提取结果数据
      d[i]=data[i];
    return n;                   // 返回数据的个数
  }

  int find::Process( )
  {
    int i, j, k, s;
    for(i=n1; i<n2; i++)        // 依次搜索满足条件的数
    {
      j=i;   s=0;
      while(j>0)
      {
        k=j%10;                 // 依次得到个位数字、十位数字……等
        s+=k;                   // 计算各位数字之和
        j=j/10;                 // 为获取下一个高位数字做准备
      }
      if(s==9)   data[n++]=i;   // 保存满足条件的整数，且数据个数加 1
    }
    return n;                   // 返回满足条件整数的个数
  }
```

为了测试上述 find 类，编写主函数如下：

```
//-------------------------------------------------------------------------------------
//                                  实例 1_6
//      文件名：ch1_6.cpp
//      功   能：使用实例 ch1_5.h 找出指定范围内所有各位数字之和为 9 的整数
//-------------------------------------------------------------------------------------
  #include "ch1_5.h"
  void main( )
  {
    int data[100], n, m;
    find f;
    n=f.GetData(data);
    cout << "满足条件的数据如下：\n";
    for(m=0; m<n; m++)
    {
```

```
            cout << data[m] << '\t';
            if(m%5==4)   cout << endl;        // 每输出 5 个数据换行
        }
        cout << endl;
    }
```

程序运行结果如下：

```
请输入第 1 个整数：80
请输入第 2 个整数：300
满足条件的数据如下：
81      90      108     117     126
135     144     153     162     171
180     207     216     225     234
243     252     261     270
```

1.2.2 继承性

继承是面向对象程序设计方法的重要特性。所谓继承，就是在已存在的类的基础上创建新类，新的类不仅拥有原有类的全部功能和特征，并且还可以增加新的功能和特征。其中原有的类称为基类（父类），新建的类称为派生类（子类）。通过继承，可以有效地保护原有成熟的代码，提高代码的重用率，增强代码的可扩展性。因此面向对象的继承特性，可以极大地提高软件开发的效率。

在 C++语言中，派生类不仅可以继承基类中除构造函数、析构函数外的所有成员，而且还可以通过不同的继承方式更改所继承成员的访问控制属性。有 3 种继承方式：公有继承（Public）、保护继承（Protected）和私有继承（Private）。3 种继承方式的区别简述如下：

（1）公有继承：基类 Public 和 Protected 成员的访问属性在派生类中保持不变。

（2）保护继承：基类 Public 和 Protected 成员的访问属性在派生类中均为 Protected 类型。

（3）私有继承：基类 Public 和 Protected 成员的访问属性在派生类中均为 Private 类型。

在上述 3 种继承方式中，基类 Private 成员在派生类中均不可访问。

下面通过实例演示继承的使用方法。考虑问题：

要求创建一个类，该类从键盘输入 2 个整数，输出这 2 个整数之间所有各位数字之和为 9 的偶数。

回顾一下实例 1_5 不难看出，本例实际上是实例 1_5 在功能上的扩展，在实例 1_5 的基础上进行过滤，找出其中所有的偶数。可以单独编写一个类实现其功能，但更合理的选择是采用继承，通过继承可以重用 ch1_5.h 中经测试被证明是正确的代码，减少程序调试时间，提高开发效率。

实例代码如下：

```
//-------------------------------------------------------------------------------------------------
//                                        实例 1_7
//       文件名：ch1_7.h
```