



高等学校计算机科学与技术教材

- ❶ 原理与技术的完美结合
- ❷ 教学与科研的最新成果
- ❸ 语言精炼，实例丰富
- ❹ 可操作性强，实用性突出

Visual C++ 程序设计与实践：实验与指导

□ 梁普选 编著

清华大学出版社

● 北京交通大学出版社



高等学校计算机科学与技术教材

Visual C++ 程序设计与实践： 实验与指导

梁普选 编著

清华大学出版社

北京交通大学出版社

·北京·

内 容 简 介

本书是《Visual C++ 程序设计与实践》教材的实验指导书,按照《Visual C++ 程序设计与实践》的编写顺序组织了十一个相关实验。为了方便教师组织教学,在每一章的实验内容前增加了相应的实验示例,并给出详细的操作步骤。通过这些示例,启发学生的编程思路,提高学生程序设计能力。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13501256678 13801310933

图书在版编目(CIP)数据

Visual C++ 程序设计与实践:实验与指导/梁普选编著. —北京:清华大学出版社;北京交通大学出版社, 2007.9

(高等学校计算机科学与技术教材)

ISBN 978-7-81123-154-0

I. V… II. 梁… III. C 语言-程序设计-高等学校-教学参考资料 IV. TP312

中国版本图书馆 CIP 数据核字(2007)第 132669 号

责任编辑:谭文芳 特邀编辑:宋林静

出版发行:清华大学出版社 邮编:100084 电话:010-62776969 <http://www.tup.com.cn>

北京交通大学出版社 邮编:100044 电话:010-51686414 <http://press.bjtu.edu.cn>

印刷者:北京东光印刷厂

经 销:全国新华书店

开 本:185×260 印张:7.25 字数:182 千字

版 次:2007 年 9 月第 1 版 2007 年 9 月第 1 次印刷

书 号:ISBN 978-7-81123-154-0/TP·379

印 数:1~4 000 册 定价:13.00 元

本书如有质量问题,请向北京交通大学出版社质监组反映。对您的意见和批评,我们表示欢迎和感谢。

投诉电话:010-51686043, 51686008; 传真:010-62225406; E-mail: press@bjtu.edu.cn。

前 言

《Visual C++ 程序设计》是一门实践性很强的课程,学生只有通过系统而大量的上机操作,编制并调试 Visual C++ 程序,将编制代码与程序调试过程、运行结果与书本知识有机联系在一起,才能深刻理解 Visual C++ 程序的运行机制,获取程序调试经验。

本书作为《Visual C++ 程序设计与实践》的配套教材,主要内容包括:创建 Visual C++ 程序与常用控件使用;菜单、工具栏与对话框应用;文档-视图及用户自定义类;多文档及其应用;基本图形的绘制;位图处理与图形输出;数据库应用;多线程编程;动态链接库与应用及网络编程等程序设计技术。本书力求使学生通过上机实验教学环节更好地掌握 Visual C++ 程序设计的思路和编程方法。

编写本书的作者是教学第一线的教师,深感计算机教学、教材和实践环节的改革对培养学生学习能力的重要性,因此将多年的教学经验和教学实践充分融合在本教材的编写中,形成了本教材的编写体系。为了方便教师教学和进一步提高学生的编程能力,在每一章的实验内容前增加了相应的实验示例,给出详细的操作步骤。通过这些示例,引导学生一步步迈向编程之路,更好地完成实验教学任务。每章附有练习和思考题,启迪开拓学生的编程思路。书中不妥之处,敬请读者批评指正。

编者

2007 年 7 月

目 录

实验 1 熟悉 Visual C++ 集成开发环境	1
一、实验目的	1
二、重点和难点介绍	1
三、实验示例	2
示例 1 创建基于工作台的应用程序 Exam1	2
示例 2 类与对象	5
示例 3 派生	7
四、实验内容	9
思考题	9
实验 2 创建 Visual C++ 程序与常用控件使用	10
一、实验目的	10
二、重点和难点介绍	10
三、实验示例	11
示例 1 应用程序向导 AppWizard 及类向导 ClassWizard	11
示例 2 常用控件的使用	13
四、实验内容	20
思考题	21
实验 3 菜单、工具栏与对话框的应用	22
一、实验目的	22
二、重点和难点介绍	22
三、实验示例	22
示例 1 创建单文档应用程序	22
示例 2 使用默认菜单	25
示例 3 使用自定义菜单	27
示例 4 工具栏及非模态对话框	30
四、实验内容	32
思考题	33
实验 4 文档—视图及用户自定义类	34
一、实验目的	34
二、重点和难点介绍	34
三、实验示例	34
示例 1 学生成绩处理程序	34
示例 2 数据的保存	41
示例 3 学生成绩的链式存储	43
四、实验内容	43

思考题	44
实验 5 多文档及其应用	45
一、实验目的	45
二、重点和难点介绍	45
三、实验示例	45
示例 1 可编辑的多文档应用程序	45
示例 2 单文档多视图应用	47
示例 3 多文档多视图应用	51
四、实验内容	51
思考题	52
实验 6 基本图形的绘制	53
一、实验目的	53
二、重点和难点介绍	53
三、实验示例	53
示例 1 在指定的控件中绘制直方图	53
示例 2 在窗体客户区绘制折线图	55
示例 3 在窗体非客户区输出文本	57
四、实验内容	58
思考题	58
实验 7 位图处理与输出	59
一、实验目的	59
二、实验示例	59
示例 1 在客户区显示位图	59
示例 2 实现打印预览	62
三、实验内容	62
思考题	62
实验 8 数据库的应用	63
一、实验目的	63
二、重点和难点介绍	63
三、实验示例	63
示例 1 创建 Access 数据库及 ODBC 数据源	63
示例 2 使用 ODBC 实现数据访问	66
示例 3 ActiveX 控件的使用	71
示例 4 ADO 数据访问	75
四、实验内容	82
思考题	82
实验 9 多线程编程	83
一、实验目的	83
二、重点和难点介绍	83

三、实验示例	84
示例 1 两个计数器线程	84
示例 2 线程同步	86
示例 3 线程间数据交互	88
四、实验内容	89
思考题	89
实验 10 动态链接库与应用	90
一、实验目的	90
二、重点和难点介绍	90
三、实验示例	90
示例 1 进程载入时链接动态链接库	90
示例 2 显式装载的动态链接库	93
示例 3 综合实例	94
四、实验内容	94
思考题	94
实验 11 Socket 编程	95
一、实验目的	95
二、重点和难点介绍	95
三、实验示例	96
示例 1 TCP 协议编程	96
示例 2 聊天室	105
思考题	105

实验 1 熟悉 Visual C++ 集成开发环境

一、实验目的

- (1) 熟悉 Visual C++ 编程环境;
- (2) 了解 OOP 编程模式;
- (3) 了解程序调试的基本方法。

二、重点和难点介绍

Visual C++ 是程序员及软件爱好者最希望掌握的工具之一,它内嵌了微软基础类库 MFC——微软程序员几十年的精髓积累,在这些上百万行 C++ 优化代码的支持下,它将 Windows 应用程序的可视化设计、C++ 源程序代码及 Windows 资源的输入、编辑、编译、调试链接集成为一体,是微软公司推出的 Windows 平台下优秀的软件开发工具之一。它由主菜单、工具栏、控件工具箱及完成各种特定功能的窗口组成,其界面如图 1-1 所示。

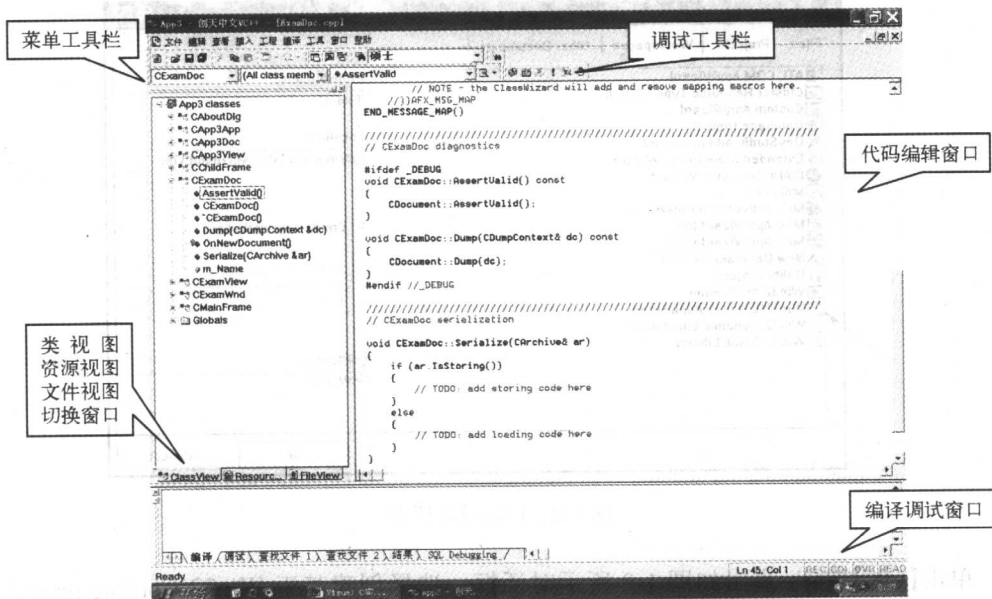


图 1-1 Visual C++ 集成开发环境

为了有效地开发 Windows 应用程序,必须熟悉 Visual C++ 编程环境。软件设计的一般规律是:根据用户需求定义软件功能模块,从中提取并定义用户数据结构,接着设计程序外观——界面设计,最后编写不同部件的程序代码,对程序进行动态调试,最终生成 Windows 环境下的可执行程序。提高调试程序效率无疑是学习编程的最佳途径之一。

Visual C++ 6.0 是 Visual Studio 6 的套件之一,它本身并不带帮助功能,微软公司为 Visual Studio 6 提供了一套 MSDN Library 帮助系统。MSDN(Microsoft Developer Network)是使用 Microsoft 开发工具或是以 Windows 或 Internet 为开发平台的开发人员的基本参考,它是一本集程序设计指南、用户使用手册及库函数于一体的电子词典,使用 MSDN Library 的帮助功能,不仅可以引导初学者入门,还可帮助各种层次的用户完成应用程序的设计。

三、实验示例

示例 1 创建基于工作台的应用程序 Exam1

(1) 创建“Win32 Console Application”应用程序 Exam1

打开【File】菜单,选择【New】菜单项,弹出【New】对话框,选择“Win32 Console Application”类型,并向【Project name】框中输入工程文件名“Exam1”,如图 1-2 所示。

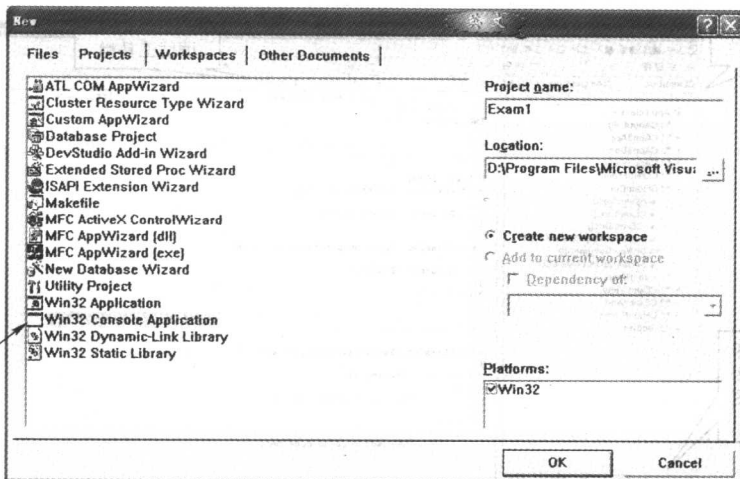


图 1-2 【New】对话框

单击【OK】按钮,弹出如图 1-3 所示对话框。选择创建基于 Win32 Console project 的“An empty project”应用程序 Exam1,如图 1-3 所示。

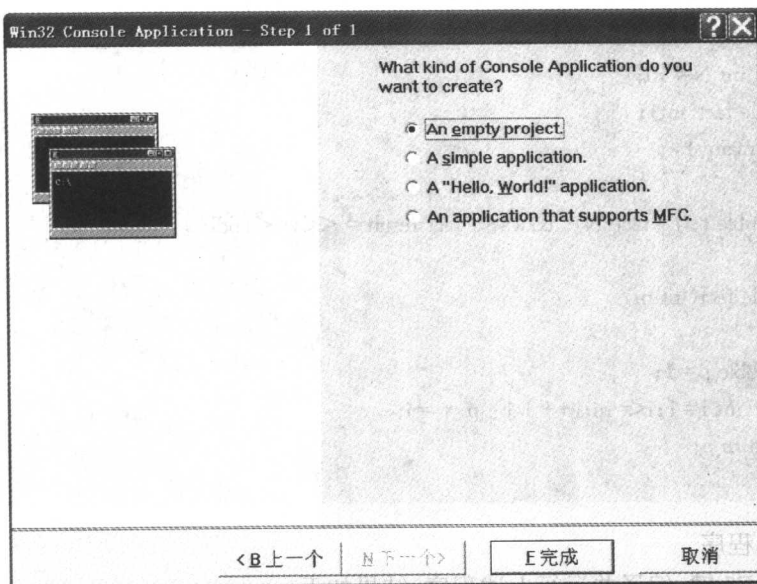


图 1-3 【Win32 Console Application-Step 1 of 1】对话框

(2) 添加 C++ 源程序文件

打开【File】菜单,选择【New】菜单项,弹出【New】对话框。选择【C++ Source File】项,向文件名框中输入“Exam1”(当前工程同名),选中【Add to project】复选框,如图 1-4 所示。单击【OK】按钮。

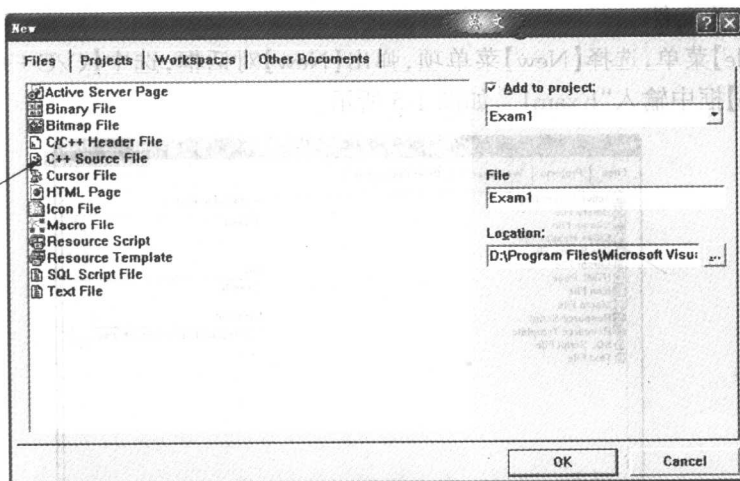


图 1-4 【New】对话框

(3) 输入代码

向代码编辑窗口中输入下列代码,计算 $n!$,并以文件名 Exam1.cpp 存盘。

```
#include <iostream.h>
const int N=10;
double fact(int);
void main()
{
    double t; t=fact(N); cout<<"fact result="<<t<<endl;
}
double fact(int m)
{
    double p=1;
    for(int i=1;i<=m;i++) { p *=i; }
    return p;
}
```

(4) 运行程序

按 Ctrl + F5 键,编译并运行上述程序,结果如下:

```
fact result=3.6288e+006
```

注意:观察工作空间中文件视图(FileView)的变化。

Visual C++ 应用程序结构将一个完整的源程序分为 C++ 头文件(.h)和实现文件(.cpp)。头文件包含全局常量定义及函数原型声明等,而.cpp 文件中包含的是函数定义的实现部分。

(5) 添加头文件

打开【File】菜单,选择【New】菜单项,弹出【New】对话框,选中【C/C++ Header File】项,并向【File】框中输入“Exam1”,如图 1-5 所示。

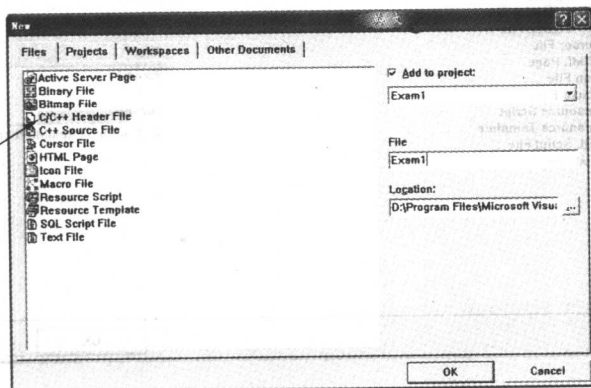


图 1-5 【New】对话框

单击【OK】按钮,以文件名“Exam1.h”存盘。

将 Exam1.cpp 文件中的前三条语句剪切到 Exam1.h 中,得到如图 1-6 所示的结果。

打开 Exam1.cpp 文件,在该程序首部添加下列语句: #include "Exam1.h"。

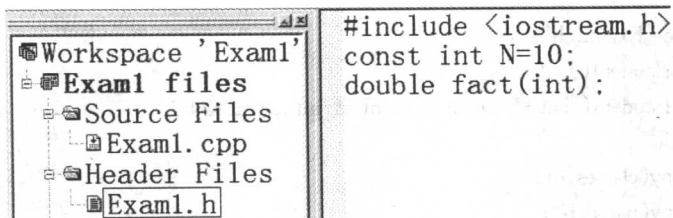


图 1-6 Exam1.h 文件内容

(6) 运行程序

按 Ctrl + F5 编译并运行该程序,得到运行程序结果:

```
fact result=3.6288e+006
```

小结: Visual C++ 的程序结构将全局常量定义及函数原型声明等书写在头文件中,实现文件书写函数定义的实现部分。

示例 2 类与对象

面向对象程序的三个特征是数据封装、代码继承与方法多态。本示例定义了学生类 student,实现了对学生信息的封装,主函数中定义学生对象数组,实现了对学生信息的管理。

- ① 创建名为“Exam2”的 Win32 Console project 的“empty project”。
- ② 添加头文件 Exam2.h。
- ③ 向头文件中输入如下代码:

```
#include <iostream.h>
#include <string.h>
class student{
private:
    char classes[32];           //班级
    char name[32];             //姓名
    int score1,score2,score3;   //三门课成绩
public:
    student();                 //默认的构造函数
    student(char *,char *,int,int,int); //重载的构造函数
    ~student();               //析构函数
    char * get_name();         //得到学生姓名
    void display();            //显示输出
};
```

说明:学生类中包含 5 个私有的数据成员,用于定义学生类的基本信息,同时声明两个重载的构造函数,用于学生数据的初始化,display 函数用于学生信息的输出。

- ④ 向当前工程中添加 Exam2.cpp,向该文件中输入源程序如下:

```

#include "Exam2.h"
student::student() {}
student::student(char * c, char * n, int s1, int s2, int s3)
{
    strcpy(classes, c);
    strcpy(name, n);
    score1 = s1; score2 = s2; score3 = s3;
}
student::~~student() {}
char * student::get_name()
{ return name; }
void student::display()
{
    cout<<"classes:"<<classes<<endl;    cout<<"name:"<<name<<endl;
    cout<<"score1:"<<score1<<endl;    cout<<"score2:"<<score2<<endl;
    cout<<"score3:"<<score3<<endl;
}

void main()
{
    student xk061[3];
    char bj[32], n[32];
    int s11, s22, s33;
    for(int i=0; i<3; i++)                //输入学生信息
    {
        cout<<"please enter classes, name, score1, score2, score3:"<<endl;
        cin>>bj>>n>>s11>>s22>>s33;
        xk061[i] = student(bj, n, s11, s22, s33);
    }
    for(i=0; i<3; i++)
    { xk061[i].display(); }                //显示全部学生信息
    cout<<"please enter seek name:"<<endl;
    cin>>n;                                //输入待查找学生姓名
    for(i=0; i<3; i++)
    {
        if(strcmp(xk061[i].get_name(), n) == 0)
            break;
    }
    if(i<3)                                //找到了则显示
        xk061[i].display();
    else
        cout<<"No found! ";
}

```

运行上述程序并解释运行结果。

示例 3 派生

首先定义基类 point,接着定义 line 类,它派生自 point 类。

(1) 定义基类 point

创建基于 Win32 Console project 的“empty project”,文件名为 Exam3,接着向当前工程中添加 Exam3.cpp 文件。向该文件中输入如下代码,其中包含两个数据成员(x1,y1)表示 point 坐标。构造函数实现了 point 对象的初始化。

```
#include <iostream.h>
#include <math.h>
class point                                //基类
{
protected:
    int x1,y1;
public:
    point(int a, int b)                    //构造函数
    {
        x1 = a; y1 = b;
    }
};
```

(2) 定义派生类 line

两点定一线段,line 类派生自 point 类,线段起始点源自 point 类,而终止点由 line 类定义。

```
class line:public point    //定义派生 line,它公有继承了基类 point
{
private:
    int x2,y2;
public:
    line(int a1,int b1,int a2,int b2):point(a1,b1)    //派生类构造函数
    {
        x2 = a2; y2 = b2; }
    double length()                                    //计算线段长度
    {
        return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2)); }
};
void main()
{
    line L1(1,1,2,2);                                //定义了起始点(1,1),终止点(2,2)
                                                         的线段 L1
    cout<<L1.length()<<endl;                        //计算线段 L1 的长度
}
```

程序运行结果:

1.41421

(3) 添加头文件

将类定义放在头文件 Exam31.h 中。

向当前工程中添加头文件 Exam31.h,接着将下列代码由 Exam3.cpp 中剪切到 Exam31.h 文件中。

```
class point{
protected:
    int x1,y1;
public:
    point(int a, int b);
};
class line:public point
{
private:
    int x2,y2;
public:
    line(int a1,int b1,int a2,int b2);
    double length();
};
```

注意:构造函数及成员函数仅保留函数原型声明部分,而函数实现在.cpp 文件中。

(4) 添加源文件 Exam31.cpp 文件

向当前工程中添加文件 Exam31.cpp,接着向该文件中输入如下代码:

```
#include "exam31.h"
#include "math.h"
point::point(int a,int b)
{
    x1=a; y1=b; }
line::line(int a1,int b1,int a2,int b2):point(a1,b1)
{
    x2=a2; y2=b2; }
double line::length()
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
```

Exam3.cpp 文件中程序如下:

```
#include <iostream.h>
#include "Exam31.h"
void main()
{
    line l1(1.0,1.0,2.0,2.0);    cout<<l1.length()<<endl;
}
```

(5) 运行程序

按 Ctrl + F5 键, 程序运行结果:

1.41421

四、实验内容

以示例 2 为基类, 派生出 employee 类和 manager 类, 各自添加 get_salary() 成员函数, 其功能为获取工资。由于雇员和经理的工资计算方法不同, 由此体现出学生类的多态性。

思考题

- (1) OOP 有哪些基本特征?
- (2) 如何组织用户数据?
- (3) 数据成员的访问权限如何界定?
- (4) 缺省构造函数有何作用?
- (5) 基类和派生类之间存在何种关系?

实验 2 创建 Visual C++ 程序与常用控件使用

一、实验目的

- (1) 应用程序向导 AppWizard 及类向导 ClassWizard 的使用方法;
- (2) 常用控件的使用方法;
- (3) Visual C++ 程序结构及 Visual C++ 程序运行机制。

二、重点和难点介绍

使用应用程序向导 AppWizard 可创建基于 MFC 的三种不同风格的 Windows 应用程序框架,它们分别是:对话框类型、单文档类型和多文档类型应用程序。

使用类向导 ClassWizard 可向应用程序框架中添加用户自定义类,数据成员,成员函数及消息宏路由等。

常用的控件有:命令按钮 Button、文本框 TextBox、静态文本 StaticText、单选按钮 Radio Button、复选框 CheckBox、列表框 ListBox、组合框 ComboBox。各控件所属类见表 2-1。

表 2-1 控件类

控 件 名	命 令 按 钮	文 本 框	列 表 框	组 合 框
所属类	CButton	CEdit	CListBox	CComboBox

其中静态文本常用于输出信息;文本框既可接受用户的输入,又可输出信息;单选按钮实现多选一逻辑;复选框实现多选的逻辑;而列表框可将相关信息组织在一起供用户选择;组合框将文本框和列表框两个控件的功能合二为一。

MFC 的原始基类是 CObject,其中包含了支持 Windows 平台程序运行的大部分机制。MFC 类库中常用的类有:对话框类 CDialog、主框架类 CFrameWnd、文档类 CDocument、视图类 CView 及应用类 CWinApp。由这些类派生出基本程序结构,构成了 Visual C++ 程序框架。

Visual C++ 程序框架中唯一的全局对象 theApp 是程序运行的入口,其中包含 WinMain 函数,窗口初始化及消息循环都在该对象中得以实现。文档类用于组织用户数据,视图类支持界面和用户接口,接收用户消息并及时处理,而主框架类将应用类、文档视图类组装在一起,构成一个完整的窗口界面。