《面向对象语言与编程》

实验指导书

北京工商大学计算机与信息工程学院

(说明：本实验指导共有 10 个实验，任课教师可根据上课的实际情况和课时要求进行选择或适当调整)

实验一：Visual C++6.0 环境及程序调试方法 2 学时

一、实验目的

1. 熟悉 Visual C++6.0 的开发环境与特点。
2. 熟悉 Visual C++6.0 开发环境下的源程序编辑方法。
3. 掌握 Visual C++6.0 环境下源程序的编译、连接、运行与调试方法。
4. 学习断点的设置方法。
5. 学习 Visual C++6.0 的 Debug 调试功能。

二、实验任务

本实验包括以下 3 项任务。

1. 源程序的编辑

在 Visual C++6.0 开发环境下编辑一个简单、完整的 C++程序，这是开发各种软件必须进行的软件开发过程。

2. 源程序的编译、连接与运行

在 Visual C++6.0 环境下，对任务 1 所编辑的 C++源程序进行编译、连接和运行，观察程序的执行过程和各变量的值。

3．Debug 调试功能

对任务 1 所完成的源程序设置若干断点，掌握 Visual C++6.0 环境下断点的设置与取消方法，在 Debug 调试状态下，学习程序调试的各种相关方法，并通过 Debug 状态下的 Variable 窗口和 Watch 窗口观察程序的执行过程和结果。

三、实验指导

1. 创建项目（工程）并编辑源程序

（1）进入 Visual C++6.0 开发环境

单击“开始”菜单，从弹出的级联菜单中依次选择“程序”|Microsoft Visual Studio6.0 |Mcrosoft Visual C++6.0，如图 1-1 所示。

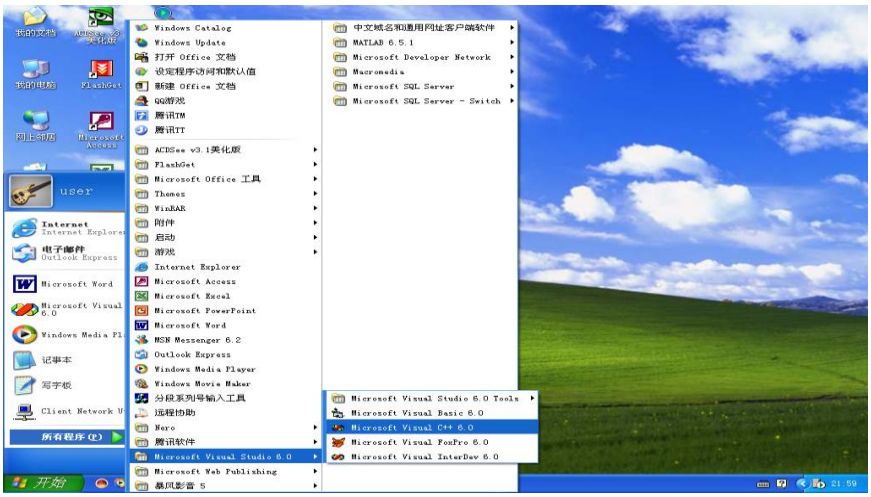


图 1-1 Visual C++6.0 开发环境的进入

单击 Microsoft Visual C++6.0 菜单项，出现 Visual C++6.0 开发环境的 Tip of the Day 提示窗口，如图 1-2 所示。该窗口显示了一条帮助信息，单击 Next Tip 按钮可显示下一条帮助信息。

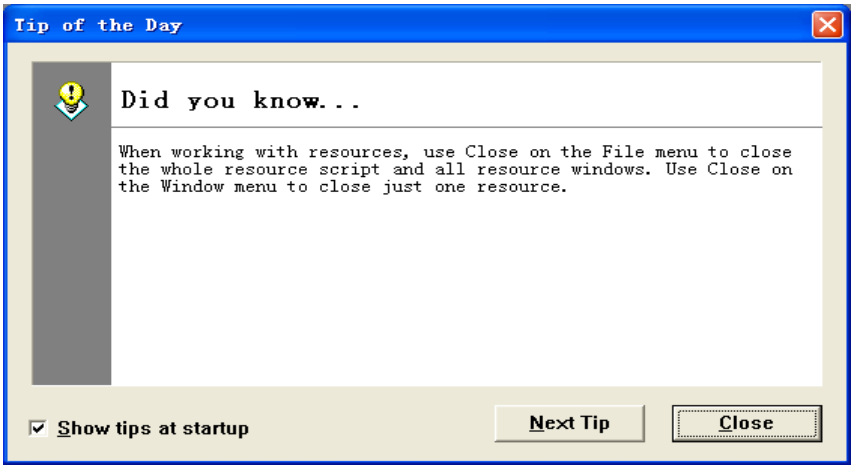


图 1-2　Tip of the Day 窗口

单击 Close 按钮关闭帮助窗口，进入如图 1-3 所示的 Visual C++6.0 开发环境主窗口。

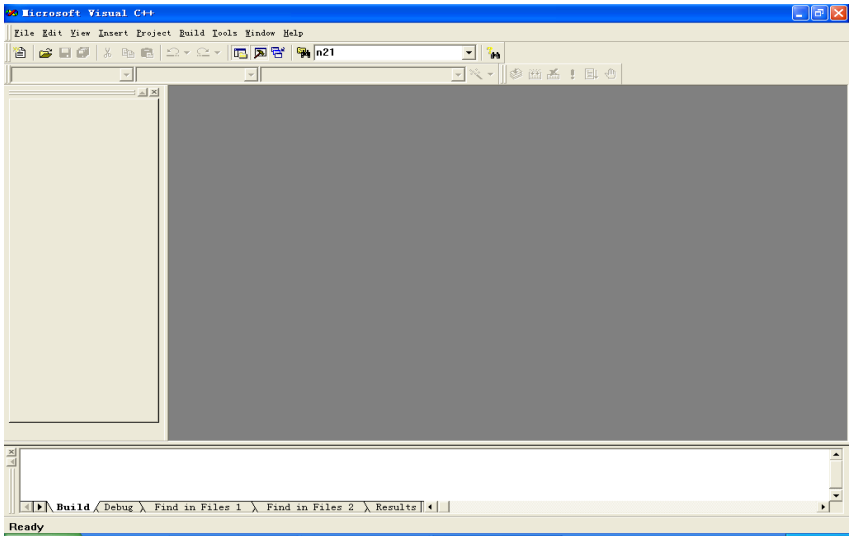


图 1-3　Visual C++6.0 开发环境主窗口

（2）创建一个 Visual C++项目

在开发环境主窗口中选择 File|New 菜单项，出现如图 1-4 所示对话框。单击 Projects 选项，选择 Win32 Console Application（Win32 控制台应用程序），在 Project name 框中输入欲创建的项目名 sy011，在 Location 框中输入（或单击“…”按钮选择已有的路径）保存项目的指定路径。单击 OK 按钮完成该窗口的设置。

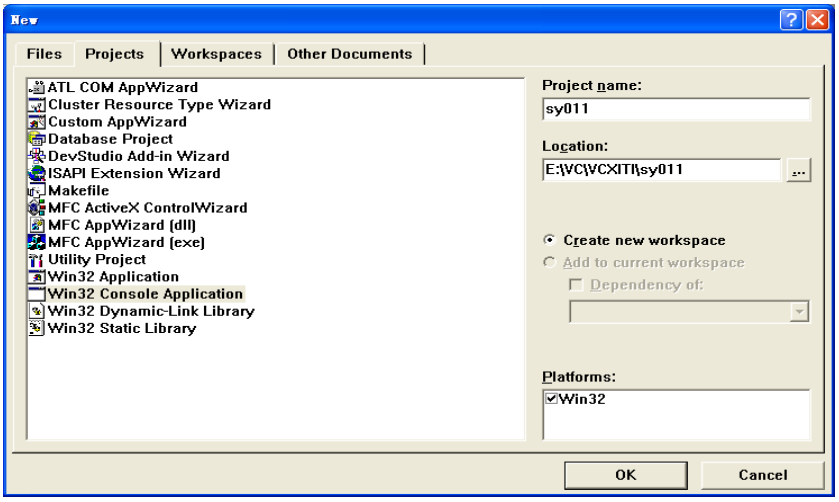


图 1-4　New 对话框

在弹出的 Win32 Console Application-Step 1 of 1 对话框中选择 An empty project 单选项，并单击 Finish 按钮，在弹出的 New Project Information 对话框中单击 OK 按钮，完成项目 sy011 的创建。

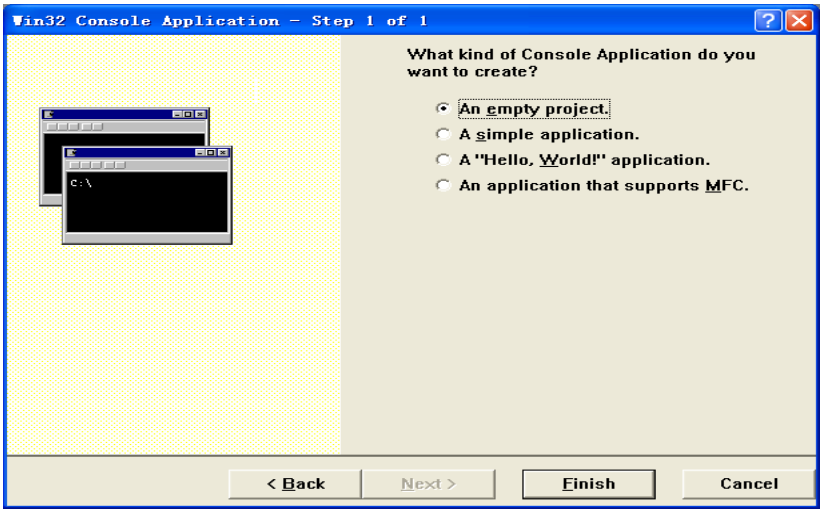


图 1-5　Win32 Console Application-Step 1 of 1

（3）编辑并建立 C++源程序

选择 Project|Add to Project|New 菜单项，弹出如图 1-6 所示的 New 对话框，在该对话框中选择 File 选项卡中的 C++ Source Files，在 File 框中输入文件名 sy011，其他接受默认选项。

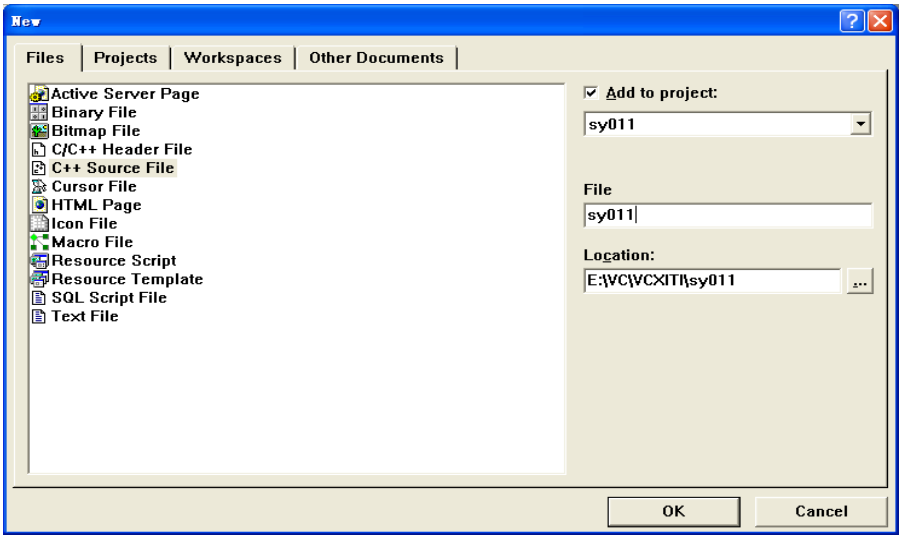


图 1-6　添加新源代码文件的 New 对话框

单击 OK 按钮，弹出如图 1-7 所示的编辑窗口，在该窗口中进行源程序代码的输入（图中已输入源程序）。

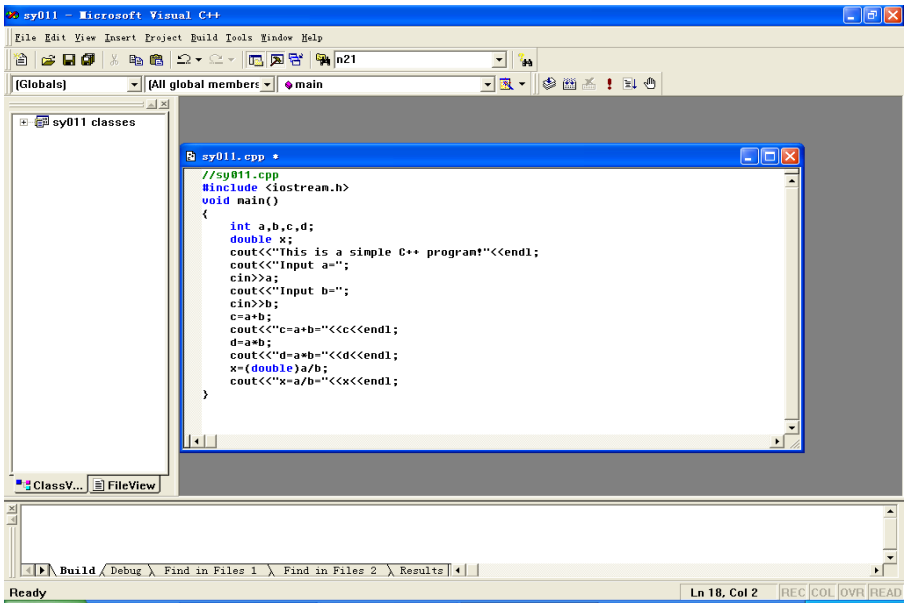


图 1-7　Visual C++6.0 源程序窗口

2. 编译并运行程序

（1）完成源程序的编辑以后，选择 File|Save 保存所编辑的源程序文件。

（2）选择 Build|Compile sy011.cpp 编译源程序，如果程序有语法错误，在屏幕正下方的状态窗口中将显示源程序的所有语法错误信息，根据这些错误提示信息在编辑窗口中修改源程序，直至编译时没有任何语法错误。

（3）选择 Build|Compile sy011.exe 连接 sy011 程序，如有错误信息显示，则先排除这些错误，如没有错误信息，则已经建立了 sy011.exe 可执行文件。

（4）选择 Build|Execute sy011.exe 选项，运行 sy011 程序。

3. 使用 Debug 调试程序

（1）设置断点。单击源程序中需要设置断点的代码行，按 F9 键即可在该行设置一个断点，如图 1-8 所示。同样的方法可设置其他断点。在已设置断点的代码行再次按 F9 键，即可取消所设置的断点。

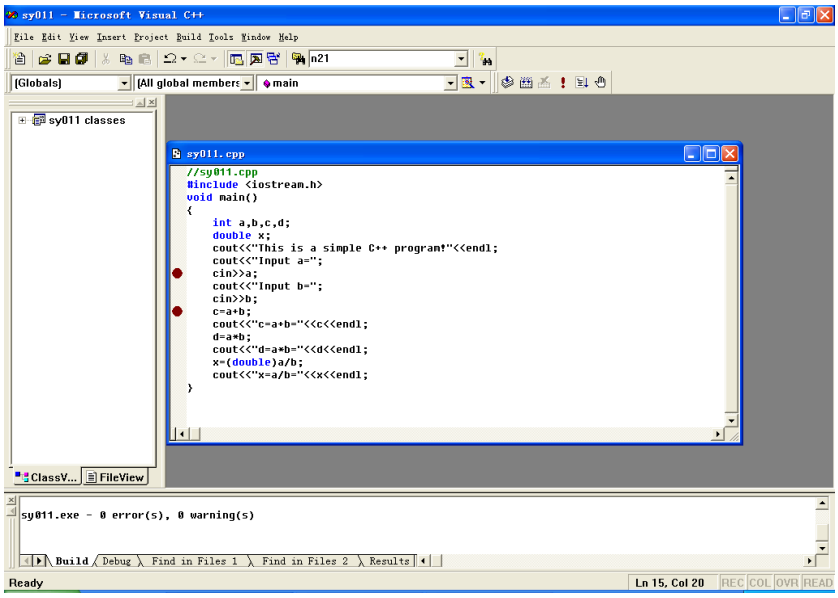


图 1-8　Visual C++6.0 断点设置窗口

（2）选择 Build|Start Debug|Go 菜单项（或按快捷键 F5），则进入 Debug 调试状态，这时 Build 菜单变为 Debug 菜单，如图 1-9 所示。

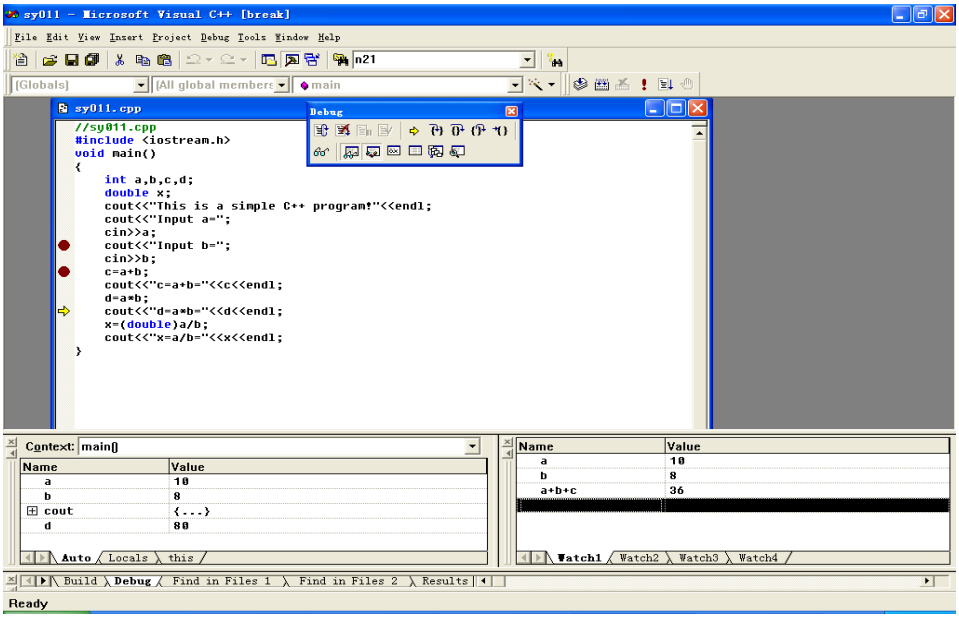


图 1-9　Visual C++6.0 源程序调试窗口

（3）在 Debug 工具栏中单击 Step Over 或 Debug 菜单中选择 Step Over，程序单步执行，在单步执行时给变量 a 赋值 10，给变量 b 赋值 8，这时可以从屏幕左下方 Variable 窗口的 Auto 页面观察当前语句和上一条语句使用的变量值，并可以显示使用 Step Over 或 Step Out 命令之后函数的返回值。

（4）在屏幕右下方的 Watch 窗口，有 Watch1、Watch2、Watch3 和 Watch4 四个页面，每一个页面均有一系列用户要查看的变量或表达式，用户可以将一组变量或表达式的值显示在同一个页面中。如单击 Watch1 页面 Name 下面的空行，并输入变量名 a，则在 Value 区域

显示变量 a 的值。同样的方法可以显示变量 b、表达式 a+b+c 以及其他变量与合法表达式的值。

（5）参照以上方法继续调试程序，进一步了解 Debug 菜单的其他调试功能，熟悉 Debug 的调试方法。

（6）各项调试结束以后，选择 Debug|Stop Debugging 菜单项（或快捷菜单 Shift+F5），结束程序调试。

（7）关闭项目工作区，退出 Visual C++6.0 开发环境。

选择 File|Close Workspace 关闭项目工作区，选择 File|Exit 可退出 Visual C++6.0 开发环境。

四、实验参考结果

程序 sy011 运行结果如下：

```
This is a simple C++ program!
Input a=10
Input b=8
c=a+b=18
d=a*b=80
x=a/b=1.25
```

实验二
函数的嵌套调用与递归调用 2 学时

一、实验目的

1. 掌握函数的定义与调用方法。
2. 掌握函数嵌套调用的概念与应用方法。
3. 掌握函数递归调用的概念及应用方法。
4. 学习 Visual C++6.0 的 Debug 调试功能及相关命令的使用方法。
5. 通过 Debug 功能了解函数的调用过程。

二、实验任务

本实验包括以下3项任务。

1．函数嵌套调用

利用函数嵌套调用求以下表达式的值：

$$f(x)=\frac{1-\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}{1+\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}$$

2．函数递归调用

利用函数的递归调用求以下表达式的值

m=n(n-1)(n-1)……×2×1=n!

3．Debug 调试功能

应用 Visual C++6.0 的 Debug 调试功能，观察函数嵌套调用时的执行过程。

三、实验指导

1．求解任务1

利用函数的嵌套调用实现求表达式的值。表达式 $f(x)=\dfrac{1-\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}{1+\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}$

可以分解为以下两个表达式：

$$\begin{cases} f(x)=\dfrac{1-y}{1+y} \\ y=\dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \end{cases}$$

根据实验任务和要求，程序代码如下：

```
//sy041.cpp
#include <iostream.h>
#include <math.h>
double fun1(double y);
double fun2(double x);
void main(void)
{
    double x;
    cout<<"输入函数自变量值 x=";
    cin>>x;
    cout<<"函数表达式的值 f(x)="<<fun1(x)<<endl;      //函数调用
}
```

```
double fun1(double y)
{
    double f;
    f=(1-fun2(y))/(1+fun2(y));                    //函数的嵌套调用
    return f;
}

double fun2(double x)
{
    double y;
    y=(exp(x)-exp(-x))/(exp(x)+exp(-x));
    return y;
}
```

程序sy041的main()函数调用了函数fun1和fun2，而函数fun1又调用了函数fun2，属于函数的嵌套调用。本程序所形成的嵌套调用关系及函数嵌套调用中的执行流程如图2-1所示：

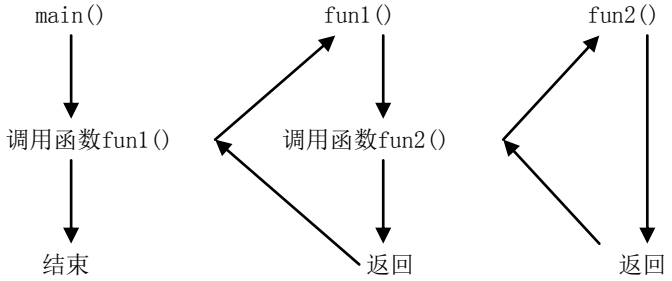


图2-1 函数的嵌套调用及执行过程

2. 求解任务 2

求 n!是一个经典的函数递归调用问题，利用函数的递归调用可以非常简便的实现对问题的求解。

例如，6!=6*5!，而5!=5*4!，…，0!=1。因此，可以应用下面的递归公式求解。

$$m!=\begin{cases}1 & (m=0)\\ m(m-1)! & (m>0)\end{cases}$$

求解 m!时需要使用(m-1)!，即采用函数的递归调用，递归结束条件是 m=0。

根据任务 2 的要求，程序代码如下：

```
//sy042.cpp
#include <iostream.h>
long fact(int m)
{
     long n;
    if (m<0) cout<<"输入数据不能小于 0,数据错误!"<<endl;
     else if (m==0||m==1) n=1;           //0!=1，1!=1
    else n=fact(m-1)*m;                  //函数的递归调用
return (n);
```

```
}

void main()
{
      long fact(int m);
      int m;
      long y;
     cout<<"输入正整数 m=";
     cin>>m;
      y=fact(m);
      cout<<m<<"!="<<y<<endl;
}
```

3．程序调试

选择 Build|Start Debug|Step Into 菜单项（或快捷键 F11），程序进入单步执行状态，此时，Visual Studio 中光标停留在 main()函数的入口处。将光标移到语句 cout<<"函数表达式的值 f(x)="<<fun1(x)<<endl 前，如图 2-2 箭头所指语句，从 Debug 菜单或 Debug 工具栏中单击 Run to Cursor，在程序运行窗口中给函数自变量 x 输入值 0.618，这时，观察变量 x 的值。

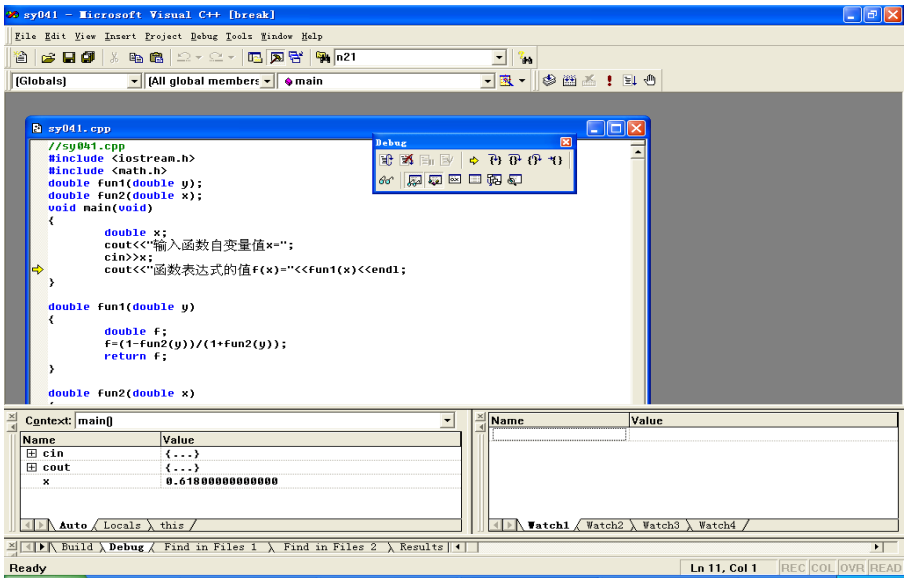


图 2-2　Debug 调试窗口之一

从 Debug 菜单或 Debug 工具栏中单击 Step Into，在 Visual Studio 中，光标停留在函数 fun1 的入口处，如图 2-3 所示。

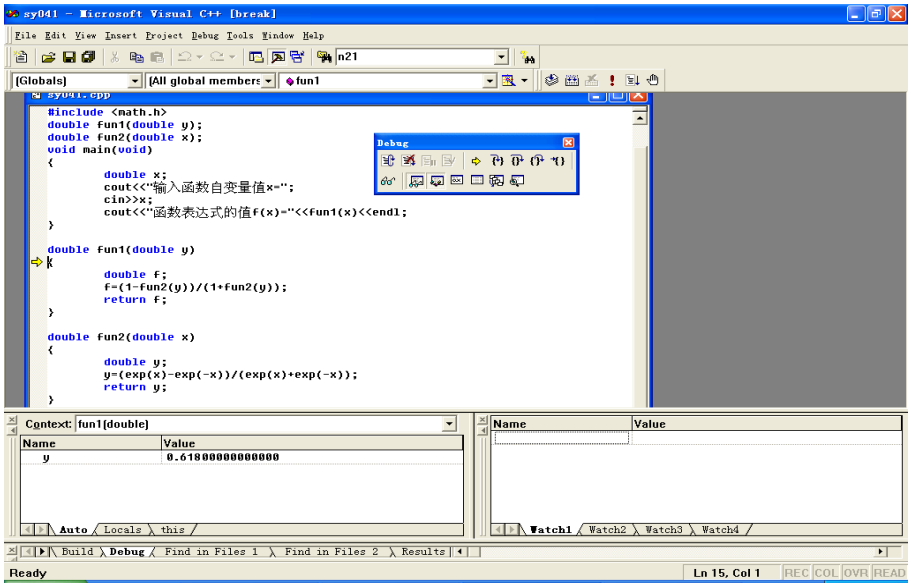


图 2-3　Debug 调试窗口之二

从 Debug 菜单或 Debug 工具栏中单击 Step Into，进入函数 fun2 内部单步执行，观察屏幕左下方的 Variable 窗口。反复单击 Step Into，连续 10 次单步执行之后，各变量的值如图 2-4 所示。

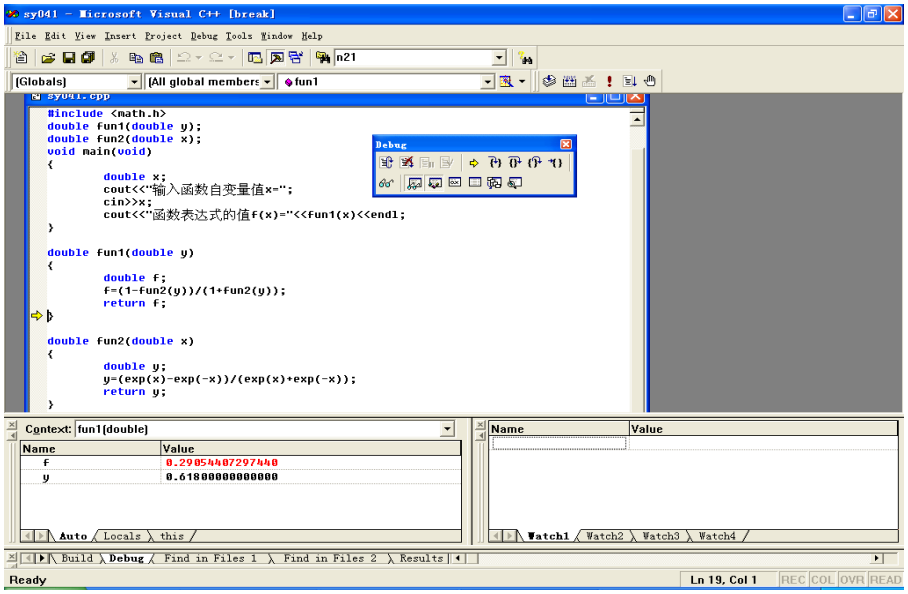


图 2-4　Debug 调试窗口之三

将光标移到 main()函数的结束处（即 main 函数右大括号左边），从 Debug 菜单或 Debug 工具栏中单击 Run to Cursor，则程序运行结束。

四、实验参考结果

程序 sy41 运行结果如下：

```
输入函数自变量值x=0.618
函数表达式的值f(x)=0.290544
```

程序 sy042 运行结果如下：

```
输入正整数m=6
6!=720
```

实验三
C++的头文件和多文件结构 2 学时

一、实验目的

1．掌握并熟练运用 C++多文件结构。
2．掌握多文件结构程序的编译、连接和运行过程。
3．理解和掌握 C++编译预处理指令的作用。

二、实验任务

设计一个项目（工程）文件，该项目由多个源程序文件组成，该程序具有多个函数，分别用于计算长方体、球体、圆柱体的体积，并将该程序的不同功能部分保存在不同的文件中，形成一个典型的多文件结构程序。

三、实验指导

C++程序设计中，既可以将程序源代码包含在一个程序文件中，也可以将不同的功能部分放在不同的文件中，形成多文件结构，C++的这种多文件结构方式有利于软件开发的分工与合作，当全部源程序完成以后，再进行编译、连接和运行。

对于大型的应用程序，一个项目文件往往包含多个头文件(.h)和源文件(.cpp)。一般将声明部分或说明部分（包括类的声明）形成为.h 文件，将函数的定义、类的实现及类的使用等形成为.cpp 文件。该项目文件中包括一个头文件 sy05.h 和 spheroid.cpp，cylinder.cpp，cuboid.cpp，sy05.cpp 等共 5 个文件。多文件项目的创建过程如下：

1. 输入项目名

在开发环境的主窗口中选择 File|New 菜单项，出现如图 3-1 所示对话框。单击 Projects 选项卡，选择 Win32 Console Application，在 Project name 框中输入项目名 sy05，在 Location 框中输入保存项目的指定路径（或单击“…”按钮选择已有的路径），然后单击 OK 按钮。

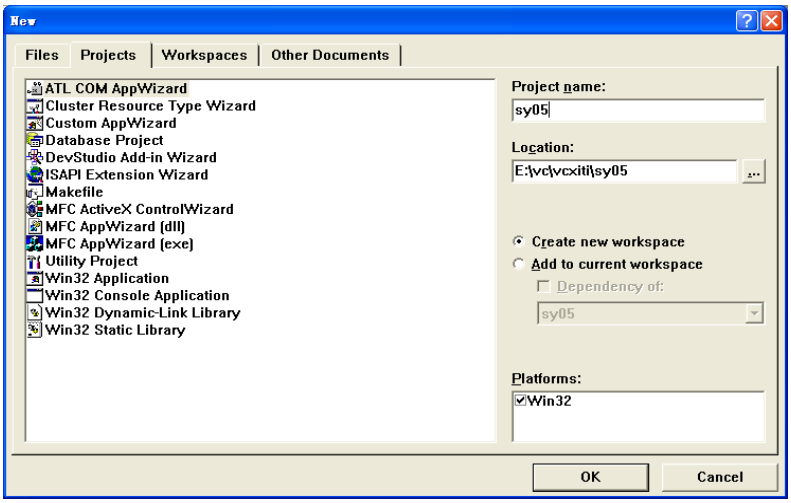


图 3-1　New 对话框

2. 完成项目创建

在弹出的 Win32 Console Application-Step 1 of 1 对话框中选择 An empty project 单选项，并单击 Finish 按钮，在弹出的 New Project Information 对话框中继续单击 OK 按钮，完成项目 sy05 的创建。

3．编辑并建立 C++源程序

（1）选择 Project|Add to Project|New 菜单项，弹出如图 3-2 所示的 New 对话框。

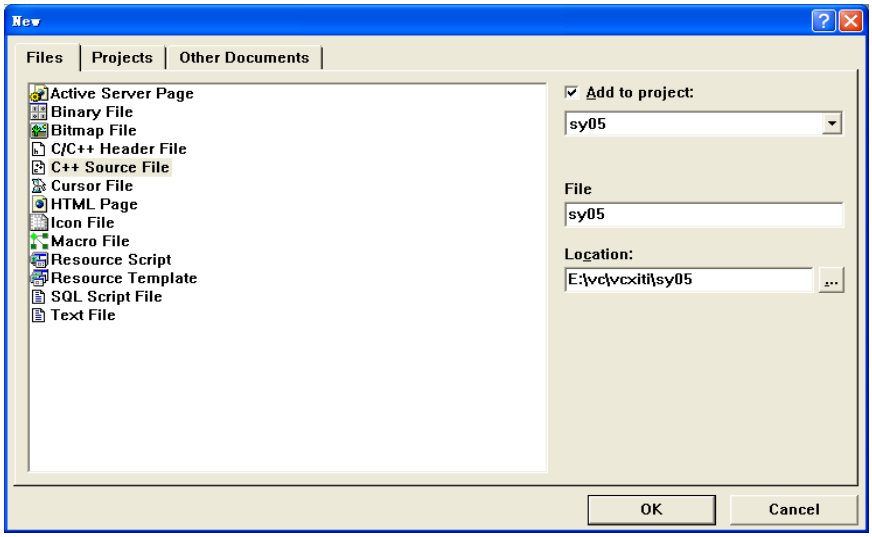


图 3-2　Visual C++6.0 源程序窗口

在该对话框中选择 File 选项卡中的 C++ Source Files，在 File 框中输入文件名 sy05，单击 OK 按钮，在弹出的编辑窗口中编辑 sy05.cpp 源程序文件。

```
//sy05.cpp
#include<iostream.h>
#include"sy05.h"
```

```
void main()
{
    cout<<"V1="<<cuboid(16.8,8.6,15.6)<<endl;
     cout<<"V2="<<spheroid(12)<<endl;
    cout<<"V3="<<cylinder(18,16)<<endl;
}
```

（2）重复步骤（1），在 File 框中输入文件名 cuboid.cpp（其他操作步骤相同），编辑 cuboid.cpp 源程序文件。

```
//cuboid.cpp
double cuboid(double x,double y,double z)   //求长方体体积
{
    return x*y*z;
}
```

（3）以同样的方法编辑文件 cylinder.cpp 和 spheroid.cpp，这两个文件源代码分别如下：

```
//cylinder.cpp
const double pi=3.1415926;
double cylinder(double h,double r)          //求圆柱体体积
{
   return h*pi*r*r;
}
//spheroid.cpp
const double pi=3.1415926;
double spheroid(double r)                  //计算球体体积
{
    return 4*pi*r*r*r/3;
}
```

（4）编辑 sy05.h 头文件。选择 Project|Add to Project|New 菜单项，弹出如图 3-3 所示的 New 对话框。

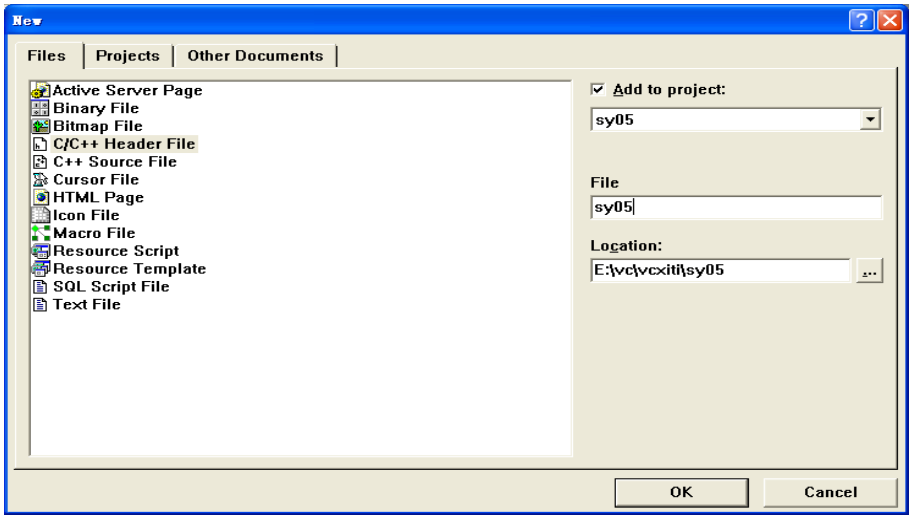


图 3-3　建立头文件的 New 对话框

在该对话框中选择 File 选项卡中的 C++ Header File，在 File 框中输入文件名 sy05，单击 OK

按钮，在弹出的编辑窗口中编辑 sy05.h 头文件。

```
//sy05.h
double cuboid(double x,double y,double z);
double cylinder(double h,double r);
double spheroid(double r);
```

4．编译、连接与运行

单击 sy05.cpp 源程序文件窗口，选择 Build|Compile sy05.cpp，然后分别选择 Build|Compile sy05.cpp 和 Execute sy05.exe 连接和运行程序。

四、实验参考结果

程序 sy05 运行结果如下：

```
V1=2253.89
V2=7238.23
V3=14476.5
```

实验四
构造函数与复制构造函数的执行 2 学时

一、实验目的

1. 掌握类与对象的定义与使用方法。
2. 掌握类的不同属性的成员的使用方法。
3. 掌握对象数组的概念与使用方法。
4. 掌握不同属性数据成员的使用。

5. 掌握构造函数与析构函数的概念及使用方法。
6. 观察构造函数与析构函数的执行过程。

二、实验任务

设计一个 Student（学生）类，并使 Student 类具有以下特点：

（1）该类具有学生姓名、学号、程序设计、信号处理、数据结构三门课程的成绩。

（2）学生全部信息由键盘输入，以提高程序的适应性，。

（3）通过成员函数统计学生平均成绩，当课程数量增加时，成员函数无需修改仍可求取平均成绩。

（4）输出学生的各科成绩与平均成绩。

（5）学生对象的定义采用对象数组实现。

（6）统计不及格学生人数。

（7）表格输出整齐美观。

（8）能以最方便的方式实现对课程数量和学生人数的修改。

三、实验指导

1．设计 Student 类

根据任务要求，Student 类应包括以下成员函数和数据成员：

`char name[20]`	表示学生姓名的字符数组
`char Id[10]`	表示学生学号的字符数组
`double course[N];`	表示 N 门课程的成绩
`double aver_score`	表示各学生平均成绩
`Student()`	构造函数
`~Student()`	析构函数
`void Display()`	输出各科成绩与平均成绩
`double Get_Aver()`	计算平均成绩
`bool No_pass()`	判断学生是否有不及格科目

2．定义常量

为实现课程数量和学生人数变化时程序修改的便利性，将学生人数 M 和课程数量 N 以 const 关键字定义为常数，当需要修改学生人数或课程数量时，只需修改常数的定义，程序的其他部分无需修改。

3．程序通用性分析

定义对象数组 s[M]表示学生人数，定义课程数组 course[M]表示 M 门课程的分数，这样既便于增强输出函数的适应性、也增加了程序对学生人数和课程数量变化的适应性。

4．程序代码设计

该程序代码如下：

```
//sy041.cpp
```

```
#include <iostream.h>
#include <string.h>
#include <stdio.h>
const int N=3;                              //N 表示课程数量
const int M=2;                              //M 表示对象数组大小
class Student                               //类 Student
{
public:
    Student();
    ~Student();
    void  Display();
    double Get_Aver();
    bool No_pass();
protected:                                  //保护成员
    char  name[20];
    char  Id[10];
    double course[N];
    double aver_score;
};

Student::Student()
{
    //cout<<"执行构造函数"<<endl;          //观察构造函数执行顺序时取消本行注释标志
    cin>>name>>Id;
    for(int i=0;i<N;i++)                    //输入学生成绩
    {
        cin>>course[i];
    }
}

Student::~Student()
{
    cout<<"执行析构函数"<<endl;
}

void Student::Display()                 //输出学生成绩
{
    cout<<name<<"   ";
    cout<<Id<<"    ";
    for(int i=0;i<N;i++)
    {
        cout<<course[i]<<"       ";
    }
    cout<<Get_Aver();
```

```
    cout<<endl;
}

double Student::Get_Aver()          //计算平均分数
{
    int i;
   aver_score=0;
   for(i=0;i<N;i++)
    {
      aver_score+=course[i];
    }
   aver_score=aver_score/N;
    return aver_score;
}

bool Student::No_pass()             //查找不及格学生
{
    for(int i=0;i<N;i++)
    {
      if(course[i]<60)
          return true;
    }
    return false;
}

void main()
{
    int i,no_pass=0;
    cout<<"输入学生信息,不同数据之间以空格分隔:"<<endl;
    cout<<"姓  名  学   号  程序设计  信号处理  数据结构"<<endl;
    Student s[M];                   //创建对象数组
    cout<<endl;
   cout<<endl<<"以下是输出信息与统计信息:"<<endl;
    cout<<"姓  名  学    号  程序设计  信号处理  数据结构  平均成绩"<<endl;
    for(i=0;i<M;i++)
    {
        s[i].Display();
        if(s[i].No_pass()==true)    //统计不及格人数
            no_pass++;
    }
   cout<<"不及格人数为:"<<no_pass<<"人."<<endl;
}
```

四、实验参考结果

1．程序 sy041 运行结果如下：

```
输入学生信息,不同数据之间以空格分隔：
姓  名  学    号  程序设计   信号处理   数据结构
李为民  010818     90       98         99
张晓利  010829     56       80         96

以下是输出信息与统计信息：
姓  名  学    号  程序设计   信号处理   数据结构   平均成绩
李为民   010818     90       98         97        95
张晓利   010829     56       80         96        73.33
不及格人数为：1人
执行析构函数
执行析构函数
```

2．构造函数与析够函数执行顺序

为观察构造函数的执行顺序，可以取消构造函数 Student 中被注释的输出语句，重新编译、连接和运行程序，由于表示对象数组大小的常数 M 为 2，因此，构造函数执行了 2 次（每构造一个对象数组的成员时执行一次构造函数），对象生存期结束时，系统自动调用析构函数，因此，析够函数同样应执行 2 次。

实验五
动态内存分配及数组、指针和引用 2 学时

一、实验目的

1. 掌握一维数组、二维数组的定义与应用。
2．掌握数组元素的初始化方法。
3．掌握指针及指针数组的使用方法。
4．掌握按数组方式、指针方式、地址方式输出数组元素的方法。
5．掌握存储空间的动态分配与使用。
6．通过 Debug 调试功能观察指针及其所指向存储单元的内容。

二．实验任务

本实验包括以下 3 项任务。

1．数组元素的访问与初始化

定义一个三行四列的二维数组，并将数组元素进行初始化为以下指定值：

{18,21,12,13,28,22,13,14,58,68,29,28}

并以下标方式、地址方式、指针方式分别输出该二维数组元素。

2．动态内存分配与使用

使用指针申请 N 个动态（N 为某一指定常数）存储空间，由键盘输入 N 个数据，并验证存储结果，将以上存储空间的数据进行平方运算，并将运算结果值赋值给数组 array。

3．Debug 跟踪程序的执行

使用 Debug 调试功能观察程序的执行过程（任务 2），观察指针的内容及其所指向对象的内容。

三、实验指导

1．数组元素的访问方法

根据任务 1 的要求，定义二维数组 array[3][4]，和指针数组*p[3]，这样可以分别实现以下标方式、地址方式、指针方式分别输出二维数组各元素。

程序代码如下：

```
//sy071.cpp
#include<iostream.h>
void main()
{
    int array[3][4]={18,21,12,13,28,22,13,14,58,68,29,28};
    int i,j,*p[3];
   cout<<"按数组下标方式输出二维数组:"<<endl;
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            cout<<array[i][j]<<" ";
        cout<<endl;
    }

   cout<<"按地址方式输出二维数组:"<<endl;
    for(i=0;i<3;i++)
    {
        for(j=0;j<4;j++)
            cout<<*(*(array+i)+j)<<" ";
        cout<<endl;
    }
    cout<<"按指针方式输出二维数组:"<<endl;
   for(i=0;i<3;i++)
```

```
    {
        p[i]=array[i];
        for(j=0;j<4;j++)
            cout<<*(*(p+i)+j)<<" ";
        cout<<endl;
    }
}
```

2．动态内存分配方法

根据任务 2 的要求，定义 double 型指针*ptr，使用 new 语句申请动态内存，为提高程序的通用性，动态内存空间的大小定义为常数 N。

程序代码如下：

```
//sy072.cpp
#include<iostream.h>
const int N=6;                  //N 表示动态申请存储空间的大小
void main()
{
    int i;
    double *ptr;
    ptr=new double[N];          //动态申请 N 个存储空间
    cout<<"输入 N 个数,不同数据之间以空格分隔:"<<endl;
    for(i=0;i<N;i++)
    {
        cin>>*(ptr+i);
    }
   cout<<"ptr="<<ptr<<endl;
   cout<<"输出 N 个数如下:"<<endl;
   for(i=0;i<N;i++)
    {
        cout<<*(ptr+i)<<" ";
    }
    cout<<endl;
   cout<<"N 个数平方以后输出如下:"<<endl;
    double array[N];
    for(i=0;i<N;i++)
    {
        array[i]=(*(ptr+i))*(*(ptr+i));
        *(ptr+i)=array[i];
        cout<<"array["<<i<<"]="<<array[i]
            <<",*(ptr+"<<i<<")="<<*(ptr+i)<<endl;
    }
}
```

3．程序调试

（1）选择 Build|Start Debug|Step Into 菜单项，或快捷键 F11，程序进入单步执行状态，在 Visual Studio 中，光标停留在 main()函数的入口处。

（2）从 Debug 菜单或 Debug 工具栏中单击 Step Over，程序进入单步执行状态，反复单击 Step Over，并按要求输入 N 个数据。

（3）当单步执行到如图 5-1 箭头所指向的语句时，在 Watch 窗口的 Name 列依次输入 *ptr 至 *(ptr+5)，观察指针 ptr 及其内容，在图 5-1 中，当单步执行到第 5 次循环时，ptr 的值为 0x00440050，通过 Watch 窗口可观察指针所指向的存储单元的值，其值与程序运行结果完全一致。

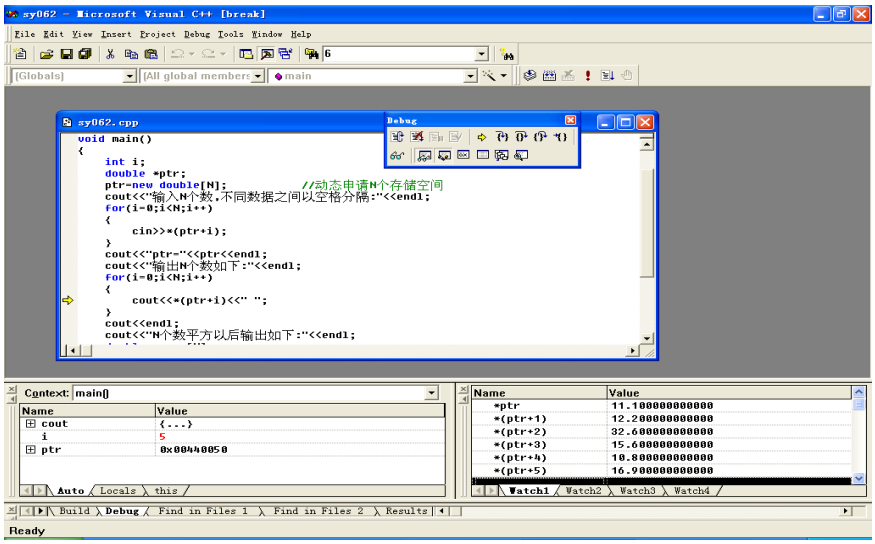


图 5-1　sy072 在调试状态下的单步执行过程之一

（4）继续单击 Step Over，当单步执行到如图 5-2 左边箭头所指向的语句时（第 5 次循环），指针及其所指向地址单元的值可通过图 5-2 右下方的 Watch 窗口观察。

（5）继续单步执行，直到程序结束，观察调试结果。

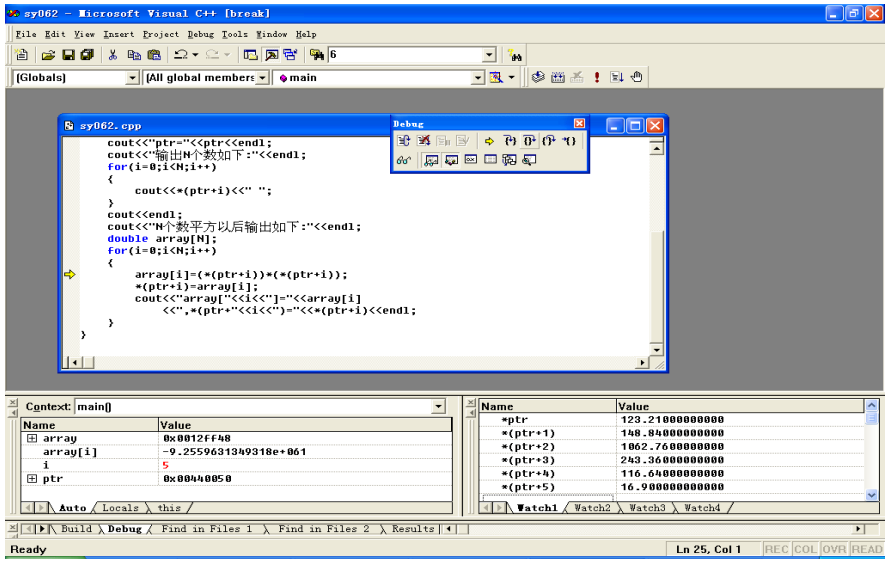


图 5-2　sy072 在调试状态下的单步执行过程之二

四、实验参考结果

程序 sy071 运行结果如下：

```
按数组下标方式输出二维数组：
18 21 12 13
28 22 13 14
58 68 29 28
按地址方式输出二维数组：
18 21 12 13
28 22 13 14
58 68 29 28
按指针方式输出二维数组：
18 21 12 13
28 22 13 14
58 68 29 28
```

程序 sy072 运行结果如下：

```
输入6个数,不同数据之间以空格分隔：11.1 12.2 32.6 15.6 10.8 16.9
输出6个数如下：11.1 12.2 32.6 15.6 10.8 16.9
ptr=0x00440050
5个数平方以后输出如下：
array[0]=123.21  *(ptr+0)=123.21
array[1]=148.84  *(ptr+0)=148.84
array[2]=1062.76  *(ptr+0)=1062.76
array[3]=243.36  *(ptr+0)=243.36
array[4]=116.64  *(ptr+0)=116.64
array[5]=285.61  *(ptr+0)=285.61
```

实验六
面向对象技术继承与派生机制的应用 4 学时

一、实验目的

1．掌握面向对象程序设计继承与派生机制的概念。
2．掌握类继承关系中构造函数的定义方法。
3．掌握派生类中构造函数与析构函数的执行顺序。

4．掌握根据具体问题建立类的层次关系的方法。

5．理解并掌握 public、protected 与 private 三种不同继承方式的特点与属性。

二、实验任务

定义点 Point 类作为基类，在此基础上派生出直线 Line 类和圆 Circle 类，在 Circle 类的基础上以不同的属性派生出圆柱 Cylinder 类和球 Spheroid 类。并要求基类和各派生类具有以下特点：

（1）Line 类含有计算直线长度和斜率的成员函数。

（2）Circle 类含有计算圆面积的成员函数。

（3）Cylinder 类含有计算圆柱表面积和体积的成员函数。

（4）Spheroid 类含有计算球体的表面积和体积的成员函数。

三、实验指导

定义直线 Point 类作为基类，在 Point 类的基础上以 public 方式派生出 Line 类和 Circle 类，在 Circle 类的基础上以 public 派生出 Cylinde 类，以 protected（不同的派生属性）派生出 Spheroid 类。

以上各类的成员函数如下：

Line 类成员函数：

```
double CalculateLength()          计算直线长度的成员函数
double CalculateSlope()           计算直线斜率的成员函数
```

Circle 类成员函数：

```
double CalculateArea()            计算圆面积的成员函数
double GetRadius()                获取圆半径的成员函数
```

Cylinde 类成员函数：

```
double CalculateVolume()          计算圆柱体积的成员函数
double GetHigh()                  获取圆柱高度的成员函数
double CalculateArea()            计算圆柱表面积的成员函数
```

Spheroid 类成员函数：

```
double CalculateArea()            计算球体表面积的成员函数
double GetC()                     获取 z 坐标值的成员函数
double CalculateVolume()          计算球体体积的成员函数
```

程序代码如下：

```
//sy06.cpp
#include<iostream.h>
#include<math.h>
const double pi=3.1415926;
class Point                       //定义基类 Point
{
public:
```

```
    Point(double x,double y)
    {
        x1=x;
        y1=y;
    }
protected:
   double x1,y1;
};

class Line:public Point            //定义 Line 类
{
public:
    Line(double x,double y,double X2,double Y2):Point(x,y),x2(X2),y2(Y2){}
   double CalculateLength();
   double CalculateSlope();
protected:
   double x2,y2;
};

double Line::CalculateLength()      //计算直线的长度
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}

double Line::CalculateSlope()       //计算直线的斜率
{
    return ((y1-y2)/(x1-x2));
}

class Circle:public Point          //定义 Circle 类
{
public:
    Circle(double x,double y,double R):Point(x,y),r(R){}
   double CalculateArea();
    double GetRadius();
protected:
   double r;
};

double Circle::CalculateArea()      //计算圆的面积
{
    return pi*r*r;
}
```

```
double Circle::GetRadius()
{
    return r;
}

class Cylinder:public Circle          //定义 Cylinder 类
{
public:
   Cylinder(double x,double y,double r,double h):Circle(x,y,r),h(h){}
   double CalculateVolume();
   double GetHigh();
   double CalculateArea();
protected:
   double h;
};

double Cylinder::CalculateVolume() //计算圆柱的体积
{
    return pi*r*r*h;
}

double Cylinder::GetHigh()
{
    return h;
}

double Cylinder::CalculateArea()    //计算圆柱的表面积
{
    return (2*pi*r*r+2*pi*r*h);
}

class Spheroid: protected Circle    //定义 Spheroid 类
{
public:
    Spheroid(double x,double y,double Z,double R):Circle(x,y,R),z(Z){}
    double CalculateArea();
    double GetZ();
    double CalculateVolume();
protected:
   double z;
};

double Spheroid::CalculateArea()    //计算球体的表面积
{
```

```
    return 4*pi*r*r;
}

double Spheroid::CalculateVolume()  //计算球体的体积
{
    return 4*pi*r*r*r/3;
}

double Spheroid::GetZ()
{
    return z;
}

void main()
{
    Line L(2,2,8,8);
    cout<<"派生类Line(直线)相关信息如下:"<<endl;
    cout<<"直线的长度为:"<<L.CalculateLength()<<endl;
   cout<<"直线的斜率为:"<<L.CalculateSlope()<<endl<<endl;
   Circle C(2,2,2.8);
   cout<<"派生类Circle(圆)相关信息如下:"<<endl;
    cout<<"圆的面积为:"<<C.CalculateArea()<<endl<<endl;
    Cylinder Cy(2,2,6.66,8.8);
   cout<<"派生类Cylinder(圆柱)相关信息如下:"<<endl;
   cout<<"圆柱的表面积为:"<<Cy.CalculateArea()<<endl;
   cout<<"圆柱的体积为:"<<Cy.CalculateVolume()<<endl<<endl;
   Spheroid  Sp(2,2,6.66,2);
   cout<<"派生类Spheroid(球)相关信息如下:"<<endl;
   cout<<"球的表面积为:"<<Sp.CalculateArea()<<endl;
   cout<<"球的体积为:"<<Sp.CalculateVolume()<<endl;
}
```

四、实验参考结果

程序 sy06 运行结果如下：

```
派生类Line(直线)相关信息如下:
直线的长度为:8.48528
直线的斜率为:1

派生类Circle(圆)相关信息如下:
圆的面积为:24.6301
```

派生类Cylinder(圆柱)相关信息如下:
圆柱的表面积为:646.939
圆柱的体积为:1226.26

派生类Spheroid(球)相关信息如下:
球的表面积为:50.2655
球的体积为:33.5103

实验七
利用纯虚函数实现多态性 4 学时

一、实验目的

1．深入理解面向对象程序设计关于多态性的概念与特征。
2．深入理解动态联编的概念与应用方法。
3．深入理解抽象类的概念与应用。
4．掌握纯虚函数在多态性中的应用。
5．掌握纯函数的定义方法，运用纯虚函数实现多态性。

二、实验任务

设计一个汽车类 Motor，该类具有可载人数、轮胎数、马力数、生产厂家和车主 5 个数据成员，根据 Motor 类派生出 Car 类、Bus 类和 Truck 类，其中 Bus 类除继承基类的数据成员之外，还具有表示车厢节数的数据成员 Number；Truck 类除继承基类的数据成员之外，还具有表示载重量的数据成员 Weight。每个类都具有成员函数 Display，用于输出各类对象的相关信息。

三、实验指导

根据任务要求，设计 Motor 类具有以下 5 个数据成员：

```
int N                  可载人数
int tire               轮胎数
double motor           马力数
char *str1             生产厂家
char *str2             车主
```

派生类 Car 具有与基类 Motor 相同的数据成员，而派生类 Bus 除继承基类的 5 个数据成员外，还具有表示公交车车厢节数的数据成员（Number），派生类 Truck 除具有以上 5 个

数据成员外，还具有表示载重量的数据成员（Weight）。成员函数 Display 定义为纯虚函数用于实现输出。

程序代码如下：

```
//sy071.cpp
#include<iostream.h>
#include<string>
class Motor                          //抽象类
{
public:
    Motor(int i,int j,int k,char *ch1,char *ch2)
    {
        N=i;
        tyre=j;
        motor=k;
        str1=new char[strlen(ch1)+1];//动态分配内存
        strcpy(str1,ch1);
        str2=new char[strlen(ch2)+1];//动态分配内存
        strcpy(str2,ch2);
    }
    virtual void Display()=0;        //纯虚函数 Display()
protected:                           //基类数据成员
    int N;
    int tyre;
    double motor;
    char *str1;
    char *str2;
};

class Car:public Motor
{
public:
    Car(int i,int j,double k,char *ch1,char *ch2):Motor(i,j,k,ch1,ch2){}
    void Display()
    {
        cout<<"小车的基本数据如下:"<<endl;
        cout<<"    该车可坐"<<N<<"人"<<endl;
        cout<<"    具有"<<tyre<<"个轮胎"<<endl;
        cout<<"    动力是"<<motor<<"马力"<<endl;
        cout<<"    生产厂家是"<<str1<<endl;
        cout<<"    小车的车主是"<<str2<<endl;
    }
};

class Bus:public Motor
```

```
{
public:
    Bus(int i,int j,int k,char *ch1,char *ch2,int n):Motor(i,j,k,ch1,ch2)
    {
       Number=n;
    }

    void Display()
    {
        cout<<"公交车的基本数据如下:"<<endl;
        cout<<"    核定载客数为"<<N<<"人"<<endl;
        cout<<"    该公交车是"<<Number<<"厢车"<<endl;
        cout<<"    具有"<<tyre<<"个轮胎"<<endl;
        cout<<"    动力是"<<motor<<"马力"<<endl;
        cout<<"    生产厂家是"<<str1<<endl;
        cout<<"    公交车的车主是"<<str2<<endl;
    }
protected:                              //派生类新增数据成员
    int Number;
};

class Truck:public Motor
{
public:
    Truck(int i,int j,int k,char *ch1,char *ch2,double w):
         Motor(i,j,k,ch1,ch2)
    {
        weight=w;
    }

    void Display()
    {
        cout<<"货车的基本数据如下:"<<endl;
        cout<<"    前排座位数为"<<N<<"个"<<endl;
        cout<<"    该车具有"<<tyre<<"个轮胎"<<endl;
        cout<<"    动力是"<<motor<<"马力"<<endl;
        cout<<"    载重量是"<<weight<<"吨"<<endl;
        cout<<"    生产厂家是"<<str1<<endl;
        cout<<"    货车的车主是"<<str2<<endl;
    }
protected:                                  //派生类新增数据成员
    double weight;
};
```

```
void main()
{
    Car car(4,4,58,"长春一汽集团","陈天晴");
    car.Display();
    cout<<endl;
    Bus bus(100,8,98,"北京客车厂","杨先锋",2);
    bus.Display();
    cout<<endl;
    Truck truck(2,6,90,"东风汽车公司","王  娟",8);
    truck.Display();
}
```

四、实验参考结果

程序 sy071 运行结果如下：

小车的基本数据如下：
该车可坐 4 人
具有 4 个轮胎
动力是 58 马力
生产厂家是长春一汽集团
小车的车主是陈天晴

公交车的基本数据如下：
核定载客数为 100 人
该公交车是 2 厢车
具有 8 个轮胎
动力是 98 马力
生产厂家是北京客车厂
公交车的车主是杨先锋

货车的基本数据如下：
前排座位数为 2 个
该车具有 6 个轮胎
动力是 90 马力
载重量是 8 吨
生产厂家是东风汽车公司
货车的车主是王　娟

实验八
函数模板的概念与应用　2 学时

一、实验目的

1．掌握模板的概念与应用。
2．掌握函数模板的定义与使用方法。

二、实验任务

用函数模板实现求 n 个数据的最小值，具体要求如下：
（1）能求不同数据类型的最小值。
（2）n 个数据由键盘输入。

三、实验指导

根据任务要求，可以采用函数模板求最小值，将求 n 个数据最小值的函数 FindMin 设计为函数模板，数据类型定义为模板参数。用数组表示 n 个数据，并将数组大小定义为常数，以适应数据长度变化时，程序修改的便利性。

程序代码如下：

```
// sy081.cpp
#include <iostream.h>
//模板说明
template <typename T>            //typename 与 class 等效，也可使用 class
T FindMin(T *p,int length)       //T 为模板参数
{
    T Min=*p;
    T Max=*p;
    for(int i=1;i<=length;i++)
        if(Min>p[i])
           Min=p[i];
    return Min;
}

const int N=5;                   //数组长度定义为常量
void main()
{
    int i,a[N];
    double d[N];
```

```
    cout<<"输入"<<N<<"个整型数据(不同数据之间以空格间隔):"<<endl;
    for(i=0;i<N;i++)
         cin>>a[i];
     cout<<"以上"<<N<<"个整型数的最小数是:"<<FindMin(a,N)<<endl;
    cout<<"输入"<<N<<"个双精度型数据(不同数据之间以空格间隔):"<<endl;
    for(i=0;i<N;i++)
         cin>>d[i];
    cout<<"以上"<<N<<"个双精度实型数的最小数是:"<<FindMin(d,N)<<endl;
}
```

四、实验参考结果

程序 sy081 运行结果如下：

```
输入5个整型数据(不同数据之间以空格间隔):
12 8 21 28 56
以上5个整型数的最小数是:8
输入5个双精度型数据(不同数据之间以空格间隔):
-6.8 -1.2 16 28 68
以上5个双精度实型数的最小数是:-6.8
```

实验九
文本文件与二进制文件的输入输出 2 学时

一、实验目的

1. 熟悉流的概念和流类库的层次结构
2. 握与文件相关的常用成员函数的使用方法。
3. 掌握文本的概念及其操作方法。
4. 掌握二进制文件的概念及其操作方法。

二、实验任务

本实验包括以下 2 项任务。

1．文本文件的操作

建立名为 testfile 的文本文件，先向该文件写入以下 2 行信息：
您好，您已经打开了 testfile 文件!
1000　2878
然后关闭文件，以输入模式打开 testfile 文件读取数据，并向计算机屏幕输出文件信息。

2．二进制文件的操作

建立名为 filebinary 的二进制文件，向该文件以二进制方式写入以下信息：
1280　binary file!
然后关闭文件，并再次打开 filebinary 文件读取文件数据，并向计算机屏幕输出文件信息。

三、实验指导

1．文本文件操作程序设计

根据任务 1 要求，先建立 ofsream 流类的对象 fileout，创建 testfile 文件，并写入所要求的字符与数据信息，然后使用成员函数 close 关闭文件，完成 testfile 文件的建立。

建立 ifsream 流类的对象 filein，使用成员函数 read 读取文件信息，实现向屏幕输出文件信息。

程序代码如下：

```
//sy091.cpp
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
void  test()
{
    ofstream fileout;                          //定义对象 fileout
    fileout.open("testfile");
    if(!fileout)
    {
        cerr<<"产生错误！无法打开 testfile 文件!"<<endl;
        exit(1);
    }
    fileout<<"您好，您已经打开了 testfile 文件!\n";  //写入数据
    fileout<<1000<<" "<<2878<<endl;              //写入数据
    fileout.close();                             //关闭文件

    ifstream filein;                             //定义对象 filein
    filein.open("testfile");                     //打开文件 testfile
    if(!filein)
    {
```

```
        cerr<<"产生错误！无法打开 testfile 文件!"<<endl;
        exit(1);
    }
    char str[50];
    int n1,n2;
    filein>>str>>n1>>n2;                          //读取文件信息
    cout<<str<<endl;
    cout<<n1<<" "<<n2<<endl;
    filein.close();
}

void main()
{
    test();
}
```

程序运行结束以后，可以用 Windows 记事本程序打开所建立的 testfile 文件查看文件信息，testfile 文件的内容与计算机屏幕所显示的内容完全相同。

2．二进制文件操作程序设计

根据任务 2 要求，先建立 ofsream 流类的对象 fileout，创建 testfile 文件，以二进制方式写入任务所规定的数据与字符信息，然后使用成员函数 close 关闭文件，完成二进制文件 filebinary 的建立。

建立 ifsream 流类的对象 filein，使用成员函数 read 读取信息，实现向屏幕输出文件信息。

程序代码如下：

```
//sy092.cpp
#include<fstream.h>
#include<iostream.h>
#include<string.h>
#include<stdlib.h>
void BinaryWrite()
{
    int n=1280;
    ofstream fileout;
    fileout.open("filebinary.dat",ios::out||ios::binary);
    if(!fileout)
    {
        cerr<<"产生错误！无法打开 testfile 文件!"<<endl;
        exit(1);
    }
    char str[]="binary file!";                //共 12 个字符
    fileout.write((char *)&n,sizeof(int));  //写入数据
    fileout.write(str,strlen(str));           //写入字符串
```

```
    fileout.close();
}

void BinaryRead()
{
    int n;
    char str[50];
    ifstream filein;
    filein.open("filebinary.dat",ios::in||ios::binary);
    if(!filein)
    {
        cerr<<"产生错误！无法打开 testfile 文件!"<<endl;
        exit(1);
    }
    filein.read((char *)&n,sizeof(int));      //读取数据
    filein.read(str,12);                      //读字符串
    cout<<n<<endl;
    for(int i=0;i<12;i++)
        cout<<str[i];
    cout<<endl;
    filein.close();
}

void main()
{
    BinaryWrite();
    BinaryRead();
}
```

四、实验参考结果

程序 sy091 运行结果如下：

```
您好，您已经打开了testfile文件!
1000 2878
```

程序 sy092 运行结果如下：

```
1280
binary file!
```

实验十
应用 AppWizard 向导生成应用程序 2 学时

一、实验目的

1．了解 Windows 应用程序设计过程。
2．掌握利用 AppWizard 生成 MFC 应用程序的步骤。

二、实验任务

本实验包括以下 2 项任务。
1．应用 AppWizard 向导生成一个具有编辑功能的 Windows 单文档应用程序 sy141。
2．应用 AppWizard 向导生成一个对话框应用程序 sy142。

三、实验指导

1．利用向导生成单文挡应用程序

根据任务 1 的要求，生成具有编辑功能的 Windows 单文档应用程序步骤如下：

（1）进入 Visual C++6.0 开发环境，选择 File|New 菜单项，在弹出的 New 对话框的 Projects 选项卡中选择 MFC AppWizard(exe)，在 Location 编辑框中输入文件欲保存的目录位置，或单击“…”按钮选择已有的目录。在 Project name 中输入应用程序名 sy141，并选中 Create new workspace 选项创建一个新工作区，单击 OK 按钮，出现如图 10-1 所示的 MFC AppWizard-Step 1 对话框。

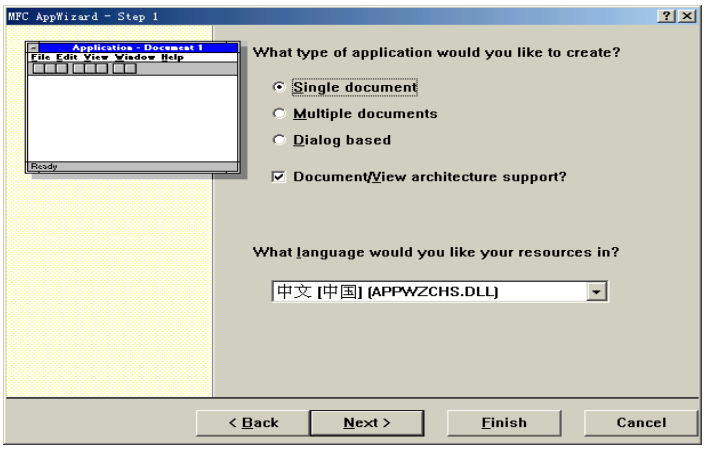


图 10-1　应用程序向导 MFC AppWizard-step 1 对话框

（2）选择 Single document（生成单文档应用程序），在 What language would you like your resources in?询问下面的列表框中选择“中文[中国]”，单击 Next 进入 step 2，如图 10-2 所示。

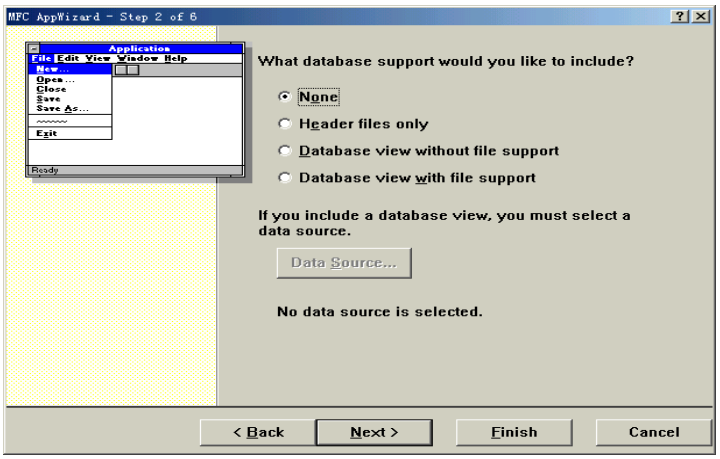


图 10-2　应用程序向导 MFC AppWizard-step 2 of 6 对话框

（3）在 What database support would you like to include？询问中选择 None，即接受默认设置。单击 Next 进入 step 3，如图 10-3 所示。

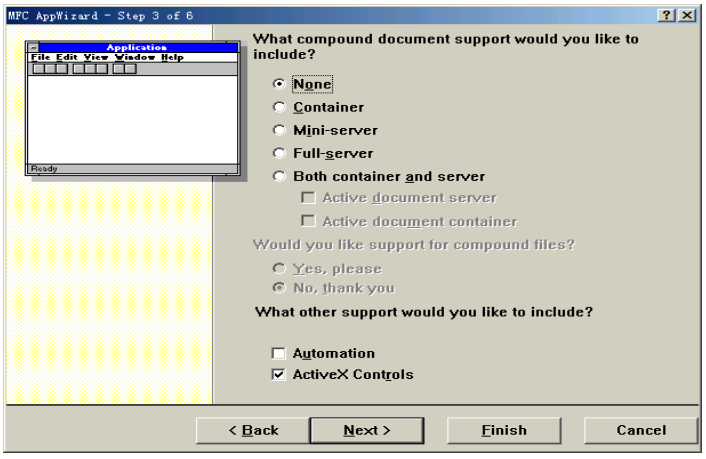


图 10-3　应用程序向导 MFC AppWizard-step 3 of 6 对话框

（4）在 What compound document support would you like to include?（选择 OLE 选项的复合文档）询问中选择 None，即不使用 OLE 特性，采用默认设置 None，并取消 ActiveX Controls，然后单击 Next 进入 step 4，如图 10-4 所示。

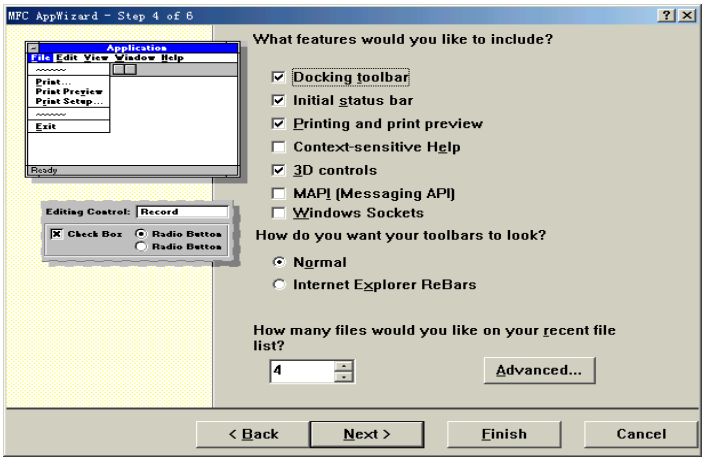


图 10-4　应用程序向导 MFC AppWizard-step4 of 6 对话框

（5）该对话框的 What features would you like to include?等选项用于选择应用程序图形界面的外观，如工具栏、状态栏、打印预览功能、使用 3D 控制外观、在线帮助、显示最近打开的文件数量等。该对话框各选项均采用默认设置，单击 Next 进入 step 5，如图 10-5 所示。

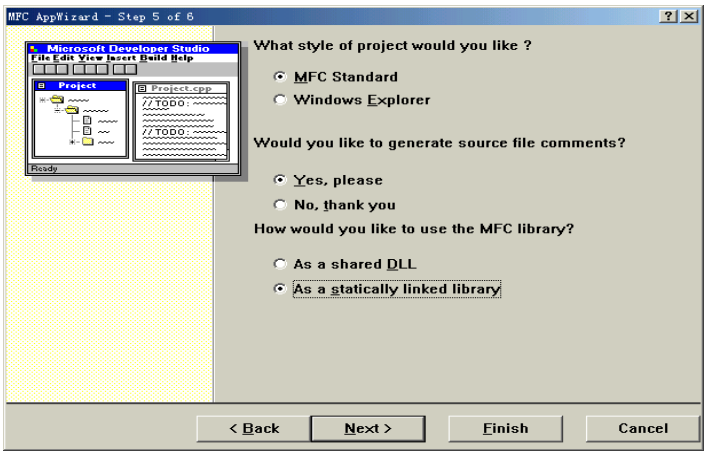


图 10-5 应用程序向导 MFC AppWizard-step 5 of 6 对话框

（6）在 step 5 对话框中选择 As a statically linked library 选项，其他接受默认设置，单击 Next 进入 step 6，如图 10-6 所示。

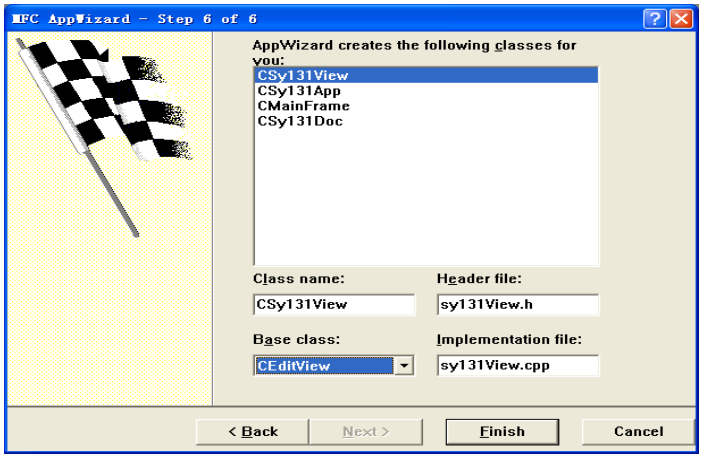


图 10-6 应用程序向导 MFC AppWizard-step 6 of 6 对话框

（7）step 6 对话框用于对默认类名、基类名、各源文件名进行选择。根据任务 1 要求，在 Base class 列表框中选择 CEditView 类作为基类（生成的应用程序 sy141 具有编辑功能），其他采用默认设置，单击 Finish 按钮，进入 New Project Information 对话框，单击 OK 按钮，完成应用程序创建。若不满意，单击 Cancel 按钮，返回前面步骤进行修改。

完成应用程序创建以后，单击 Build 菜单，选择 Build sy141.exe，当出现 sy141.exe-0 errors，0 warning 编译信息时，则表示已经正确生成了 sy141 应用程序。在 Built 菜单中选择 Execute sy141.exe 或 Ctrl+F5，则可运行该应用程序。

2．利用 AppWizard 向导创建对话框应用程序

与任务 1 创建单文挡应用程序的方法类似，在 Visual C++6.0 开发环境中，选择 File|New 菜单项，弹出 New 对话框，在 Projects 选项卡中选择 MFC AppWizard(exe)，在 Location 编辑框中输入文件欲保存的目录位置，在 Project name 中输入应用程序名 sy142，选中 Create new workspace 选项创建一个新工作区，单击 OK 按钮弹出 MFC AppWizard-Step 1 对话框，在该对话框中选择 Dialog Based 选项，单击 Next，进入下一步，在 MFC AppWizard-Step 3 of

4 中选择 As a statically linked library 选项，其他各步骤均接受默认设置。

完成全部对话框设置之后，单击 Build 菜单，选择 Build sy142.exe 构建 sy142 对话框应用程序。

四、实验参考结果

1．程序 sy141 运行结果如下：

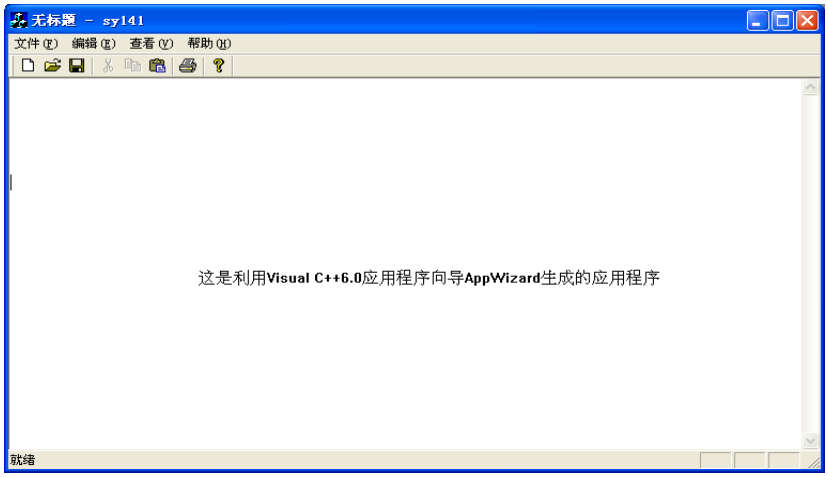


图 10-7　具有文本编辑功能（已输入信息）的单文挡应用程序界面

2．程序 sy141 运行结果如下：

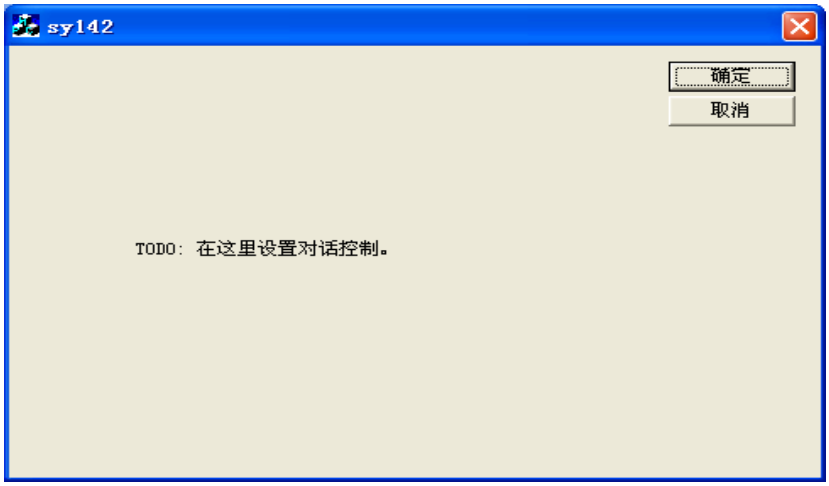


图 10-8　对话框应用程序运行界面