

《面向对象程序设计 (C++)》

实验指导书

山东理工大学计数学与统计学院

2008 年 09 月

目 录

第一部分 Visual C++实验环境介绍	2
一、Visual C++简介	2
1. Visual C++集成开发环境（IDE）	2
2. 向导（Wizard）	2
3. MFC 库	3
二、项目开发过程	3
三、集成开发环境	4
四、常用功能键及其意义	5
第二部分 上机实验内容	6
实验一 类和对象	6
实验二 派生类与继承	10
实验三 多态性	15
实验四 运算符重载	17
实验五 模板与异常处理	20
实验六 标准库和输入/输出流	22
实验七 综合应用——利用指针、模板设计通用链表以及类	25
实验八 用 VC 制作自己的屏幕保护	26
一、程序代码编写提示	26
二、代码编译调试	57
三、制作你自己的屏保	63
附录 程序编写风格规范参考	67

第一部分 Visual C++实验环境介绍

一、Visual C++简介

Visual C++是 Microsoft 公司的 Visual Studio 开发工具箱中的一个 C++程序开发包。Visual Studio 提供了一整套开发 Internet 和 Windows 应用程序的工具，包括 Visual C++，Visual Basic，Visual FoxPro，Visual InterDev，Visual J++以及其他辅助工具，如代码管理工具 Visual SourceSafe 和联机帮助系统 MSDN。Visual C++包中除包括 C++编译器外，还包括所有的库、例子和为创建 Windows 应用程序所需要的文档。

从最早期的 1.0 版本，发展到最新的 6.0 版本，Visual C++已经有了很大的变化，在 界面、功能、库支持方面都有许多的增强。最新的 6.0 版本在编译器、MFC 类库、编辑器以及联机帮助系统等方面都比以前的版本做了较大改进。

Visual C++一般分为三个版本：学习版、专业版和企业版，不同的版本适合于不同类型的 应用开发。实验中可以使用这三个版本的任意一种。

1. Visual C++集成开发环境（IDE）

集成开发环境（IDE）是一个将程序编辑器、编译器、调试工具和其他建立应用程序的 工具集成在一起的用于开发应用程序的软件系统。Visual C++软件包中的 Developer Studio 就是一个集成开发环境，它集成了各种开发工具和 VC 编译器。程序员可以在不离开该环境 的情况下编辑、编译、调试和运行一个应用程序。IDE 中还提供大量在线帮助信息协助程序员做好开发工作。Developer Studio 中除了程序编辑器、资源编辑器、编译器、调试器外，还有各种工具和向导（如 AppWizard 和 ClassWizard），以及 MFC 类库，这些都可以帮助程序员快速而正确地开发出应用程序。

2. 向导 (Wizard)

向导是一个通过一步步的帮助引导你工作的工具。Developer Studio 中包含三个向导， 用来帮助程序员开发简单的 Windows 程序，它们是：

AppWizard：用来创建一个 Windows 程序的基本框架结构。AppWizard 向导会一步步向 程序员提出问题，询问他所创建的项目的特征，然后 AppWizard 会根据这些特征自动生成 一个可以执行的程序框架，程序员然后可以在这个框架下进一步填充内容。AppWizard 支持 三类程序：基于视图/文档结构的单文档应用、基于视图/文档结构的多文档应用程序和基 于对话框的应用程序。也可以利用 AppWizard生成最简单的控制台应用程序（类似于 DOS 下用字符输入输出的程序）。

ClassWizard: 用来定义 AppWizard 所创建的程序中的类。可以利用 ClassWizard 在项目中增加类、为类增加处理消息的函数等。ClassWizard 也可以管理包含在对话框中的控件，它可以将 MFC 对象或者类的成员变量与对话框中的控件联系起来。

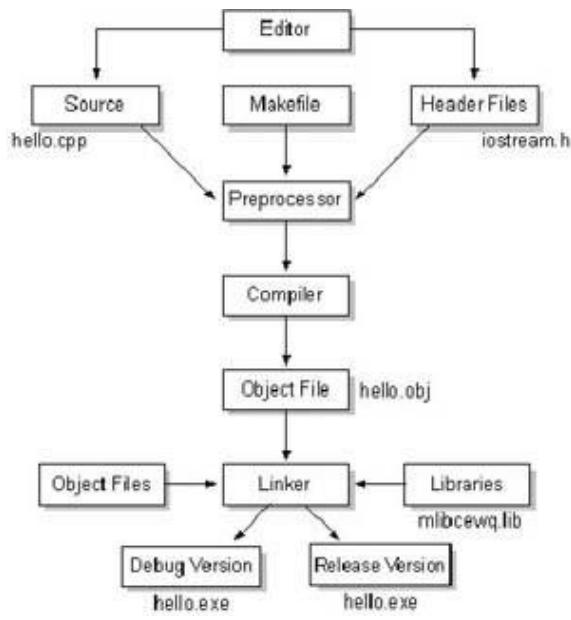
ActiveX Control Wizard: 用于创建一个 ActiveX 控件的基本框架结构。ActiveX 控件 是用户自定义的控件，它支持一系列定义的接口，可以作为一个可再利用的组件。

3. MFC 库

库 (library) 是可以重复使用的源代码和目标代码的集合。MFC (Microsoft Fundamental Classes) 是 Visual C++ 开发环境所带的类库，在该类库中提供了大量的类，可以帮助开发人员快速建立应用程序。这些类可以提供程序框架、进行文件和数据库操作、建立网络连接、进行绘图和打印等各种通用的应用程序操作。使用 MFC 库开发应用程序可以减少很多工作量。

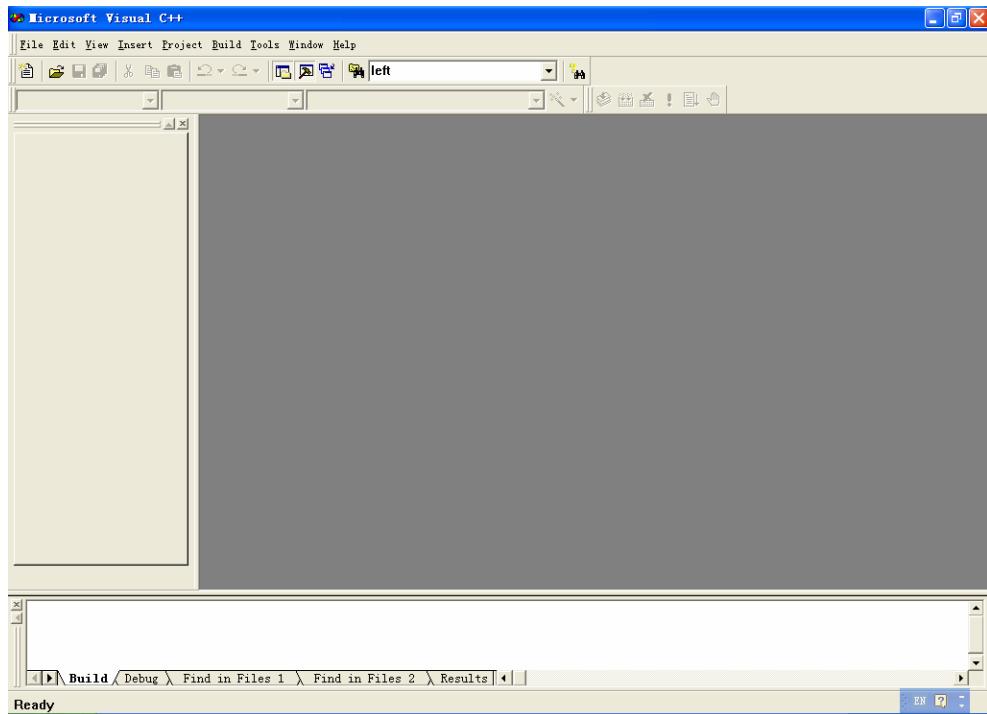
二、项目开发过程

在一个集成的开发环境中开发项目非常容易。一个用 C++ 开发的项目的通用开发过程可以用左图表示。建立一个项目的第一步是利用编辑器建立程序代码文件，包括头文件、代码文件、资源文件等。然后，启动编译程序，编译程序首先调用预处理器处理程序中的预处理命令（如 #include, #define 等），经过预处理器处理的代码将作为编译程序的输入。编译对用户程序进行词法和语法分析，建立目标文件，文件中包括机器代码、连接指令、外部引用以及从该源文件中产生的函数和数据名。此后，连接程序将所有的目标代码和用到的静态连接库的代码连接起来，为所有的外部变量和函数找到其提供地点，最后产生一个可执行文件。一般有一个 makefile 文件来协调各个部分产生可执行文件。可执行文件分为两种版本：Debug 和 Release。Debug 版本用于程序的开发过程，该版本产生的可执行程序有大量的调试信息，可以供调试程序使用，而 Release 版本作为最终的发行版本，没有调试信息，并且带有某种形式的优化。学员在上机实习过程中可以采用 Debug 版本，这样便于调试。选择是产生 Debug 版本还是 Release 版本的方法是：在 Developer Studio 中选择菜单 Build|Set Active Configuration，在弹出的对话框中，选择所要的类型，然后选择 OK 关闭对话框。Visual C++ 集成开发环境中集成了编辑器、编译器、连接器以及调试程序，覆盖了的开发应用程序的整个过程，程序员不需要脱离这个开发环境就可以开发出完整的应用程序。



三、集成开发环境

进入 Developer Studio 如果你使用的是 Visual C++ 6.0，则要进入 Developer Studio，需要单击任务栏中“开始”后选择“程序”，找到 Microsoft Visual Studio 6.0 文件夹后，单击其中的 Microsoft Visual C++6.0 图标，则可以启动 Developer Studio。Developer Studio 使用 Microsoft Developer Network (MSDN) 库（需要先行安装）作为它的联机帮助系统。其界面如下：



总的来说，窗口和命令接口（包括工具条和菜单条）是构成界面的最主要组成部分。通常有两种窗口：文档窗口和可附着（docking）窗口。文档窗口显示在文档窗口区，用于显示和编辑文档，其的大小和位置可以随其所处的Developer Studio窗口的改变而改变，可以最大化和最小化。可附着窗口可以附着于应用程序窗口的边界，也可以浮在屏幕上的任何位置。可附着窗口有：工作区（workspace）窗口，输出（output）窗口，调试窗口（包括variable, watch, local等窗口）等。文档窗口的位置、大小及是否可见和它所在的项目有关，docking窗口的位置、大小及是否可见则与项目进行的状态以及各种编辑和调试的操作有关。各种窗口和各种工具条以及菜单构成了界面的布局。一旦用户决定了一种界面布局，系统就会为一直为用户保持这种布局，直到用户下一次改变该布局为止。获得帮助信息大多数时候，你可以通过按F1得到上下文帮助。如在编辑文件时按F1可以得到有关编辑的帮助，在编译连接错误信息上按F1可以得到关于该错误的帮助信息。如果想系统地获得帮助，VC6中，可以通过选择菜单Help|Contents来启动MSDN查阅器，MSDN查阅器是一个功能强大的程序，可以方便地浏览、查找信息，要想知道具体如何使用MSDN查阅器，可以在MSDN查阅器中选菜单Help下的命令。Visual C++的编辑器Developer Studio包含一个功能强大的编辑器，可以编辑将被编译成Windows程序的Visual C++源文件。这个编辑器有点像字处理器，但是没有字处理器具备的复杂的排版、文本格式等功能，它注重的是如何帮助程序员快速高效地编制程序。它具有以下特点：自动语法。用高亮度和不同颜色的字来显示不同的语法成分，如注释、关键字和一般代码用不同的颜色显示自动缩进。帮助你排列源代码，使其可读性更强参数帮助。在编辑时用到预定义的windows函数时，可以自动为你显示函数参数集成的关键字帮助。能够使你快速得到任何关键字、MFC类或Windows函数的帮助信息（按F1即可）拖放编辑。能够用鼠标选择文本并自由拖动到任意位置自动错误定位。能自动将光标移动到有编译错误的源代码处。当你打开一个源代码文件时，就可以利用编辑器对其进行编辑。源代码文件在文档显示区显示，每个文件有独立的显示窗口。如果你选择用其他编辑器编辑源文件，必须将它以纯文本的方式保存。VC的编译器不能处理其中有特别格式字符的文件。

四、常用功能键及其意义

为了使程序员能够方便快捷地完成程序开发，开发环境提供了大量快捷方式来简化一些常用操作的步骤。键盘操作直接、简单，而且非常方便，因而程序员非常喜欢采用键盘命令来控制操作。下面是一些最常用的功能键，希望学员在实验中逐步掌握。

操作类型	功能键	对应菜单	含义
文件操作	Ctrl+N	File New	创建新的文件、项目等
	Ctrl+O	File Open	打开项目、文件等
	Ctrl+S	File Save	保存当前文件
编辑操作	Ctrl+X	Edit Cut	剪切
	Ctrl+C	Edit Copy	复制

	Ctrl+V	Edit Paste	粘贴
	Ctrl+Z	Edit Undo	撤消上一个操作
	Ctrl+Y	Edit Redo	重复上一个操作
	Ctrl+A	Edit SelectAll	全选
	Delete	Edit Delete	删除光标后面的一个字符
建立程序操作	Ctrl+F7	Build Compilercurrentfile	编译当前源文件
	Ctrl+F5	Build Runexe	运行当前项目
	F7	Build Buildexe	建立可执行程序
	F5	Build StartDebugging	启动调试程序
调试	F5	Debug Go	继续运行
	F11	Debug Stepinto	进入函数体内部
	Shift+F11	Debug Stepout	从函数体内部运行出来
	F10	Debug Stepover	执行一行语句
	F9		设置/清除断点
	Ctrl+F10	Debug Runtocursor	运行到光标所在位置
	Shift+F9	Debug QuickWatch	快速查看变量或表达式的值
	Shift+F5	Debug Stopdebugging	停止调试

第二部分 上机实验内容

实验一 类和对象

一、实验目的

- 1) 理解类和对象的概念，掌握声明类和定义对象的方法；
- 2) 掌握构造函数和析构函数的实现方法；
- 3) 初步掌握使用类和对象编制 C++ 程序；
- 4) 掌握对象数组、对象指针和 string 类的使用方法；
- 5) 掌握使用对象、对象指针和对象引用作为函数参数的方法；
- 6) 掌握类对象作为成员的使用方法；
- 7) 掌握静态数据成员和静态成员函数的使用方法。

二、实验内容和步骤

1、输入下列程序

题目 1：

```
#include<iostream>
using namespace std;
class Coordinate
{ public:
    Coordinate(int x1,int y1)
    {
        x=x1;
        y=y1;
    }
    Coordinate(Coordinate &p);
    ~Coordinate()
    {
        cout<<"Destructor is called\n";
    }
    int getx() {return x;}
    int gety() {return y;}
private:
    int x,y;
};
Coordinate::Coordinate(Coordinate &p)
{
    x=p.x;
    y=p.y;
    cout<<"copy-initialization Constructou is called\n";
}
int main()
{
    Coordinate p1(3,4);
    Coordinate p2(p1);
    Coordinate p3=p2;
    cout<<"p3=(“<<p3.getx()<<”,”<<p3.gety()<<”)\\n”;
    return(0);
}
```

(1) 写出程序的运行结果。

(2) 将 Coordinate 类中带有两个参数的构造函数进行修改，在函数体内增添下述语句：

```
cout<<" Constructor is called. \\n" ;
```

(3) 按下列要求进行调试：在主函数体内，添加下列语句：

```
Coordinate p4;
```

```
Coordinata p5(2);
```

调试程序时会出现什么错误？为什么？如何对已有的构造函数进行适当修改？

(4) 经过以上第(2)步和第(3)步的修改后,结合运行结果分析: 创建不同的对象时会调用不同的构造函数。

题目2: 设计一个 4×4 魔方程序,让魔方的各行值的和等于各列值的和,并且等于两对角线值的和。

示例:

31	3	5	25
9	21	19	15
17	13	11	23
7	27	29	1

各行、各列以及对角线值的和都是64。

[提示]求 4×4 魔方的一般步骤如下:

(1) 设置初始魔方的起始值和相邻元素之间的差值。例如上述魔方的初始魔方的起始值(first)和相邻元素之间的差值(step)分别为: first = 1, step = 2。

(2) 设置初始魔方元素的值。例如上述魔方的初始魔方为:

1	3	5	7
9	11	13	15
17	19	21	23
25	27	29	31

(3)生成最终魔方。方法如下:

- ①求最大元素值与最小元素值的和sum,该实例的sum是: $1 + 31 = 32$
- ②用32减去初始魔方所有对角线上元素的值,然后将结果放在原来的位置,这样就可求得最终魔方。本例最终魔方如下:

31	3	5	25
9	21	19	15
17	13	11	23
7	27	29	1

(4)本题的魔方类magic的参考框架如下:

```
class magic
{
public:
    void getdata();
    void setfirstmagic();
    void generatemagic();
    void printmagic();
private:
    int m[4][4];
```

```
    int step;
    int first;
    int sum;
};
```

题目 3：设计一个用来表示直角坐标系的 Location 类，在主程序中创建类 Location 的两个对象 A 和 B，要求 A 的坐标点在第 3 象限，B 的坐标在第 2 象限，分别采用成员函数和友元函数计算给定两个坐标点之间的距离，要求按如下格式输出结果。

```
A(x1,y1),B(x2,y2)
Distance1=d1
Distance1=d2
```

其中：x1、x2、y1、y2 为指定的坐标值，d1 和 d2 为两个坐标点之间的距离。

[提示]

类 Location 的参考框架如下：

```
class Location
{
public:
    Location(double,double);
    double Getx();
    double Gety();
    double distance(Location &);
    friend double distance(Location &, Location &);
private:
    double x,y;
};
```

实验二 派生类与继承

一、实验目的

- 1) 掌握派生类的声明方法和派生类构造函数的定义方法;
- 2) 掌握不同继承方式下，基类成员在派生类中的访问属性;
- 3) 掌握在继承方式下，构造函数与析构函数的执行顺序与构造规则;
- 4) 学习虚基类在解决二义性问题中的作用。

二、实验内容和步骤

1、输入下列程序

题目 1:

```
#include<iostream>
using namespace std;
class Base
{
public:
    void setx(int i){x = i;}
    int getx(){return x;}
public:
    int x;
};
class Derived:public Base
{
public:
    void sety(int i){y = i;}
    int gety(){return y;}
    void show()
    {
        cout<<"Base::x= "<<x<<endl;
    }
public:
    int y;
};
int main()
{
    Derived bb;
    bb.setx(16);
    bb.sety(25);
    bb.show();
    cout<<"Base::x= "<<bb.x<<endl;
    cout<<"Derived::y= "<<bb.y<<endl;
    cout<<"Base::x= "<<bb.getx()<<endl;
    cout<<"Derived::y= "<<bb.gety()<<endl;
    return 0;
}
```

- (1) 写出程序的运行结果。
- (2) 按以下要求，对程序进行修改后再调试，指出调试中出错的原因。
 - ① 将基类 Base 中数据成员 x 的访问权限改为 private 时，会出现哪些错误？为什么？
 - ② 将基类 Base 中数据成员 x 的访问权限改为 protected 时，会出现哪些错误？为什么？
 - ③ 在源程序的基础上，将派生类 Derived 的继承方式改为 private 时，会出现哪些错误？为什么？
 - ④ 在源程序的基础上，将派生类 Derived 的继承方式改为 protected 时，会出现哪些错误？为什么？

题目 2：编写一个学生和教师的数据输入和显示程序。学生数据有编号、姓名、性别、年龄、系别和成绩，教师数据有编号、姓名、性别、年龄、职称和部门。要求将编号、姓名、性别、年龄的输入和显示设计成一个类 Person，并作为学生类 Student 和教师类 Teacher 的基类。

[提示] 供参考的类结构如下：

```
class Person
{
    .....
};

class Student:public Person
{
    .....
};

class Teacher:public Person
{
    .....
};
```

题目 3：按要求阅读、编辑、编译、调试和运行以下程序。

- (1) 阅读、编辑、编译、调试和运行以下程序，并写出程序的运行结果。

```
#include<iostream>
using namespace std;
class MyArray
{
public:
    MyArray(int leng);
    ~MyArray();
    void Input();
    void Display(string);
protected:
    int *alist;
    int length;
};
```

```

MyArray::MyArray(int leng)
{
    if(leng<=0)
    {
        cout<<"error length";
        exit(1);
    }
    alist = new int[leng];
    length = leng;
    if(alist==NULL)
    {
        cout<<"assign failure";
        exit(1);
    }
    cout<<"MyArray类对象已创建。" << endl;
}
MyArray::~MyArray
{
    delete[] alist;
    cout<<"MyArray类对象被撤销。" << endl;
}
void MyArray::Display(string str)
{
    int i;
    int *p = alist;
    cout<<str<<length<<"个整数:";
    for(i=0;i<length;i++,p++)
    {
        cout<<*p<<" ";
    }
    cout<<endl;
}
void MyArray::Input()
{
    cout<<"请键盘输入" << length << "个整数: ";
    int i;
    int *p = alist;
    for(i=0;i<length;i++,p++)
    {
        cin>>p;
    }
}
int main()
{
    MyArray a(5);
    a.Input();
    a.Display("显示已输入的");
    return 0;
}

```

(2) 声明一个类 SortArray 继承类 MyArray, 在该类中定义一个函数, 具有将输入的整数从小到大进行排序的功能。

[提示]

供参考的类结构如下:

在第 (1) 步的基础上可增加下面的参考框架:

```
class SortArray:public MyArray
{
public:
    void Sort();
    SortArray(int leng):MyArray(leng)
    {
        cout<<"SortArray类对象已创建。"endl;
    }
    virtual ~SortArray();
};

SortArray::~SortArray()
{
    cout<<"SortArray类对象被撤销。"endl;
}

void SortArray::Sort()
{
//请自行编写Sort函数的代码, 将输入的整数从小到大排序。
}

//并将主函数修改为:
int main()
{
    SortArray a(5);
    s.Input();
    s.Display("显示排序以前的");
    s.Sort();
    s.Display("显示排序以后的");
    return 0;
}
```

声明一个类 ReArray 继承类 MyArray, 在该类中定义一个函数, 具有将输入的整数进行倒置的功能。

[提示]

在第 (1) 步的基础上可增加下面的参考框架:

```
class ReArray:public MyArray
{
public:
    void reverse();
    ReArray(int leng);
    Virtual ~ReArray();
};
```

请读者自行编写构造函数、析构函数和倒置函数 ReArray, 以及修改主函数。

(3) 声明一个类 AverArray 继承类 MyArray, 在该类中定义一个函数, 具有求输入的整数平均值的功能。

[提示]

在第 (1) 步的基础上可增加下面的参考框架:

```
class AverArray:public MyArray
{
public:
    AverArray(int leng);
    ~AverArray();
    double Aver();
};
```

请读者自行编写构造函数、析构函数和求平均值函数 Aver (求解整数的平均值), 以及修改主函数。

实验三 多态性

一、实验目的

- 1) 了解多态性的概念。
- 2) 掌握虚函数的定义和使用方法。
- 3) 掌握纯虚函数和抽象类的概念和用法。

二、实验内容和步骤

1、分析并调试下列程序，写出程序的输出结果，并解释输出结果。

```
#include<iostream>
using namespace std;
class B
{
public:
    virtual void f1(double x)
    {
        cout<<"B::f1(double)"<<x<<endl;
    }
    void f2(double x)
    {
        cout<<"B::f2(double)"<<2*x<<endl;
    }
    void f3(double x)
    {
        cout<<"B::f3(double)"<<3*x<<endl;
    }
};
class D: public B
{
public:
    virtual void f1(double x)
    {
        cout<<"D::f1(double)"<<x<<endl;
    }
    void f2(double x)
    {
        cout<<"D::f2(double)"<<2*x<<endl;
    }
    void f3(double x)
    {
        cout<<"D::f3(double)"<<3*x<<endl;
    }
};
```

```
int main()
{
    D d;
    B *pb = &d;
    D *pd = &d;
    pb->f1(1.23);
    pd->f1(3.21);
    pb->f2(1.23);
    pd->f2(3.21);
    pb->f3(1.23);
    pd->f3(3.21);
    return 0;
}
```

2、给出下面的抽象基类 container;

```
class container
{
protected:
    double radius;
public:
    container(double radius1);
    virtual double surface_area() = 0;
    virtual double volume() = 0;
};
```

要求建立 3 个继承 container 的派生类 cube、sphere 与 cylinder，让每一个派生类都包含虚函数 surface_area() 和 volume()，分别用来计算正方体、球体和圆柱体的表面积及体积。要求写出主程序，应用 C++ 的多态性，分别计算边长为 6.0 的正方体、半径为 5.0 的球体，以及半径为 5.0 和高为 6.0 的圆柱体的表面积和体积。

实验四 运算符重载

一、实验目的

- 1) 了解多态性的概念。
- 2) 掌握运算符重载的基本方法。

二、实验内容和步骤

1、编写一个程序，其中设计一个时间类 Time, 用来保存时、分、秒等私有数据成员，通过重载操作符“+”实现两个时间的相加。要求将小时范围限制在大于等于 0，分钟范围限制在 0~59，秒钟范围限制在 0~59 秒。

[提示] 时间类

```
class Time
{
public:
    Time(int h=0,int m=0,int s=0);
    Time operator+ (Time &);
    void dispTime(string);
private:
    int hourse;
    int minutes;
    int seconds;
};
```

2、编写一个程序，用于进行集合的并、差和交运算。例如输入整数集合 {9 5 4 3 6 7} 和 {2 4 6 9 }，计算出它们进行集合的并、差和交运算后的结果。

[提示]

- ① 可用以下表达式实现整数集合的基本运算：

s1 + s2 两个整数集合的并运算
s1 - s2 两个整数集合的差运算
s1 * s2 两个整数集合的交运算

- ② 参考以下 Set 类的框架，用于完成集合基本运算所需的各项功能。

```
class set
{
public:
    set();
    void input(int d);
    int length();
    int getd(int i);
    void disp();
    set operator+ (set s1);
    set operator- (set s1);
    set operator* (set s1);
    set operator= (set s1);
protected:
    int len;
    int s[MAX];
};
```

3、对于下面的类 MyString，要求重载一些运算符后可以计算表达式： $a=b+c$ ；，其中 a, b, c 都是类 MyString 的对象。请重载相应的运算符并编写程序测试。

```
class
```

```
MyString{ public:
```

```
    MyString(char *s){
```

```
        str=new char[strlen(s)+1];
```

```
        strcpy(str,s);
```

```
}
```

```
    ~MyString(){ delete[]
```

```
        str;
```

```
}
```

```
private:
```

```
    char *str;
```

```
};
```

4、定义一个名为 Month 的类，它是用于表示月份的一个抽象数据类型。这个类要用 int 类型的一个变量来表示一个月（1 表 1 月，2 表示 2 月，依次类推）。请包括以下所有成员函数：一个构造函数使用英文月份名称的前 3 个字母来设置月份名称，这 3 个字母要通过 3 个参数来接收；一个构造函数只接收一个 int 类型的参数，并用这个整数来设置月份；一个默认构造

函数；一个输入函数，它将月份作为整数来读取；一个输入函数，它将月份作为月份名称的前 3 个字母来读取；一个输出函数，它将月份作为一个整数来输出；一个输出函数，它将月份作为月份名称的前 3 个字母来输出；以及一个成员函数，它将下一个月作为 Month 类型的一个值来返回。输入和输出函数分别有一个用于流的形参。将类定义嵌入一个测试程序。

5、重新定义编程项目 2（上一个 Month 类）所描述的 Month 类的实现（如果是首次定义 它，一定要像下面描述的那样实验）。这一次，请将月份作为 char 类型的 3 个成员变量来 实现；这些成员变量分别存储了月份名称的前 3 个字母。将定义嵌入一个测试程序。

6、定义一个复数类（复数由实部和虚部构成），编写程序重载四则运算符和 ++、-- 运算符， 实现复数的相关运算。

实验五 模板与异常处理

一、实验目的

- 1) 正确理解模板的概念。
- 2) 掌握函数模板和类模板的声明和使用方法。
- 3) 学习简单的异常处理方法。

二、实验内容和步骤

1、分析并调试下面两个程序，写出运行结果并分析原因。

```
//程序5-1
#include<iostream>
using namespace std;
template<class T>
T max(T x, T y)
{
    return x>y?x:y;
}
int max(int a, int b)
{
    return a>b?a:b;
}
double max(double a, double b)
{
    return a>b?a:b;
}
int main()
{
    cout<<"max(3, 7) is: "<<max(3, 7)<<endl;
    cout<<"max(5.6, 9.7) is: "<<max(5.6, 9.7)<<endl;
    return 0;
}

//程序5-2
#include<iostream>
using namespace std;
int max(int a, int b)
{
    return a>b?a:b;
}
double max(double a, double b)
{
    return a>b?a:b;
}
int main()
{
    cout<<"max(3, 7) is: "<<max(3, 7)<<endl;
    return 0;
}
```

2、编写一个求任意类型数组中最大元素和最小元素的程序，要求将求最大元素和最小元素的函数设计成函数模板。

3、编写一个程序，使用类模板对数组元素进行排序、倒置、查找和求和。

[提示]

设计一个类模板

```
template<class Type>
class Array
{
    .....
};
```

具有对数组元素进行排序、倒置、查找和求和功能，然后产生类型实参分别为 int 型和 double 型的两个模板类，分别对整型数组与双精度数组完成所要求的操作。

4、编写一个程序，求输入数的平方根。设置异常处理，对输入负数的情况给出提示。

实验六 标准库和输入/输出流

一、实验目的

- 1) 掌握 C++ 格式化的输入输出方法。
- 2) 掌握重载运算符 “<<” 和 “>>” 的方法。
- 3) 掌握磁盘文件的输入输出方法。

二、实验内容和步骤

1、下面给出的 test 程序用于打印九九乘法表，但程序中存在错误。请上机调试，使得此程序运行后，能够输出如下所示的九九乘法表。

*	1	2	3	4	5	6	7	8	9
1	1								
2	2	4							
3	3	6	9						
4	4	8	12	16					
5	5	10	15	20	25				
6	6	12	18	24	30	36			
7	7	14	21	28	35	42	49		
8	8	16	24	32	40	48	56	64	
9	9	18	27	36	45	54	63	72	81

```
//test.cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int i,j;
    cout<<"*";
    for(i=1;i<=9;i++)
        cout<<i<<" ";
    cout<<endl;
    for(i=1;i<=9;i++)
    {
        cout<<i;
        for(j=1;j<=i;j++)
            cout<<i*j;
    }
    return 0;
}
```

2、下面的程序用于统计文件 xyz.txt 中的字符个数，请填空完成程序。

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    char ch;
    int i=0;
    ifstream file;
    file.open("xyz.txt",ios::in);
    if( ① )
    {
        cout<<"xyz.txt cannot open" << endl;
        abort();
    }
    while (!file.eof())
    {
        ②
        i++;
    }
    cout<<"文件字符个数: "<<i<< endl;
    ③
    return 0;
}
```

3、重载运算符“<<”和“>>”，使其能够输入一件商品的信息和输出这件商品的信息。
商品的信息由编号、商品名和价格。假如商品类 Merchandise 的框架如下：

```
class merchandise
{
public:
    Merchandise();
    ~Merchandise();
    friend istream& operator>>(istream& in,Merchandise& s);
    friend ostream& operator<<(ostream& out,Merchandise& s);
private:
    int no;
    char *name;
    double price;
};
```

要求实现该类，并编写以下的 main 函数对该类进行操作。

```
int main()
{
    Merchandise mer;
    cin>>mer;
    cout<<mer;
    return 0;
}
```

4、编写一个程序，将两个文本文件连接成一个文件，然后将此文件中所有小写字母转换成大写字母，并打印出来。

5、写一个程序来计算一门课的成绩（以满分 100 计）。这门课程的记录包含在一个文件中，该文件将学生的名字、又一个空格，最后是学生 10 次测验的分数（全部包含在一行中）。测验分数全是整数，而且每个分数都以空格分隔。程序将从这个文件读取输入，并将输出发送到另一个文件。输出文件中的数据与输入文件中的数据几乎一样，唯一的区别在于输出文件中各行末尾多一个 double 类型的数。这个数是该学生 10 次测验的平均分。

6、这个程序用于为一个文本文件中的各行编号。请写一个程序，要求它从一个文件中读取文本，并将读取的每一行同时输出到屏幕和另一个文件，并为其附加一个行号。要在每行的起始处打印行号。行号占 3 个字符位置的一个域，并在这个域中右对齐。在行号之后，请添加一个冒号，再添加一个空格，最后附上原来的文本。每次应该读取一个字符，而且要忽略每一行原来在开头位置的任何空格。你可假定这些行都非常短，能在屏幕上的一行之内完整地显示。否则，如果一行文本太长，你就要允许默认地打印机或屏幕输出行为（也就是说，需要允许自动换行或截尾）。一个比较难的版本是事先统计号行数，然后据此判断行号需要占据多大的一个域。这个工作是在处理文件中的各个行之前完成的。这个较难的版本应该确保每一行的长度不超过 72 个字符；如果某一行太长，就需要在刚好能满足这个要求的最后一个完整单词之后插入一个换行符。

7、写一个程序，用它计算一个文件的以下各项统计信息：文件中的字符总数、非空白字符的总数和文件中的字母数。并将统计结果输出到屏幕和另一个文件。

实验七 综合应用——利用指针、模板设计通用链表以及类

1、写一个名为 `merge_lists` 的函数，它获取两个传引用调用参数。两个参数都是指针变量，分别指向两个链表的表头，而且两个链表都由 `int` 类型的值构成。假定两个链表已经排序，位于表头的数字是最小的数字，然后从小到大一次排列。函数返回指向一个新链表的表头的指针。新链表包含原来两个链表的所有节点，而且同样按从小到大的顺序排列。注意，你的函数不能创建或销毁任何节点。函数调用结束之后，两个指针变量参数的值应该等于 `NULL`。

2、本项目要求你使用链表（而不是数组）重做第 7 章的编程项目 10。由于要生成的是一个 `double` 类型的链表，所以需要对部分成员函数进行修改。它的成员包括：一个默认构造函数；一个名为 `add_item` 的成员函数，它在列表中添加一个 `double` 值；一个 `full()` 函数，它检测列表是否已满，返回一个布尔值；以及一个友元函数，它重载了插入操作符 `<<`。

3、通讯簿类管理软件设计。按照下面的要求编写程序：某公司财务部需要开发一个计算雇员工资的程序。

该公司有 3 类雇员，工人的工资为每小时工资额乘当月工作时数再加上工龄工资；销售员工资为每小时工资额乘当月工作时数加上销售额提成再加上工龄工资，其中销售额提成等于该销售员当月售出商品金额的 1%；管理人员工资为基本工资再加上工龄工资。工龄工资就是雇员在该公司工作的工龄每增加 1 年，月工资就增加 35 元。

请用面向对象方法分析、设计这个程序，并用 C++ 语言写出完整的程序。任选如下题目之一：

- (1) 个人信息管理软件；
- (2) 通讯录管理软件二； 要求：

- (a) 使用面向对象方法进行课程设计。要求使用对象/类，继承，多态性等技术；
- (b) 程序的界面、逻辑功能和数据的存储尽可能的分开；
- (c) 软件的主要结构包括：用户登录，数据的录入、查询、删除、修改、统计等功能。

实验八 用 VC制作自己的屏幕保护

结合参考文献《用 VC制作屏保程序》，我们给出了每一步的 VC操作提示，具体编写代码的时候，先阅读参考文献，然后，根据每一步的图形提示，完成参考文献中屏保的制作。

一、程序代码编写提示

结合参考文献《用 VC制作屏保程序》，我们给出了每一步的 VC操作提示，具体编写代码的时候，先阅读参考文献，然后，根据每一步的图形提示，完成参考文献中屏保的制作。

1、建立 MFCAppWizard (exe) 应用工程 MyScreenSave

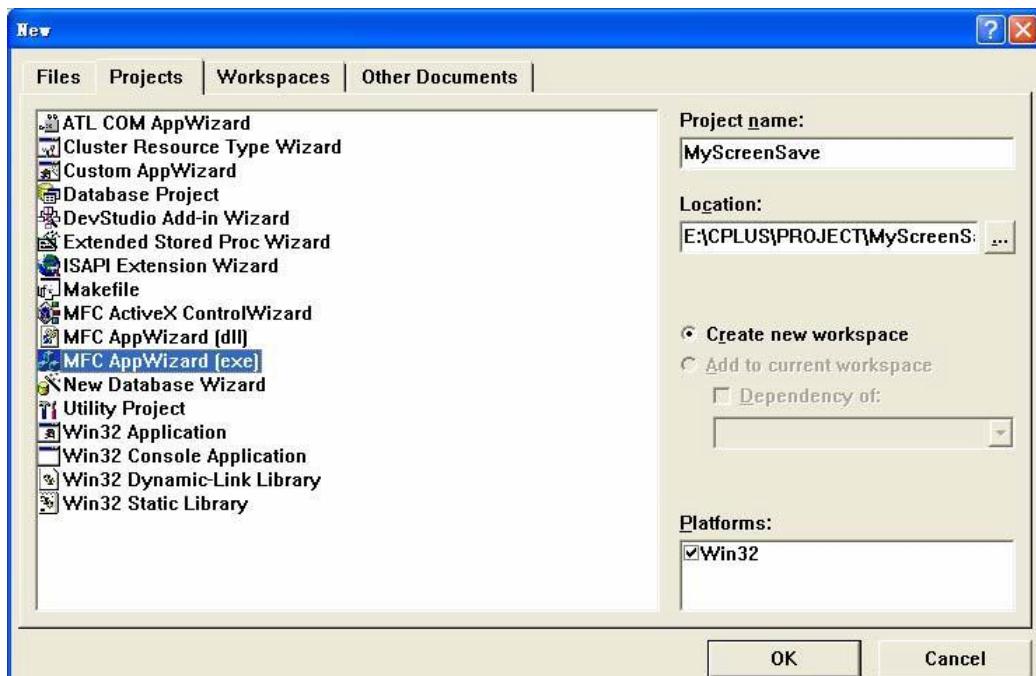


图 1-1

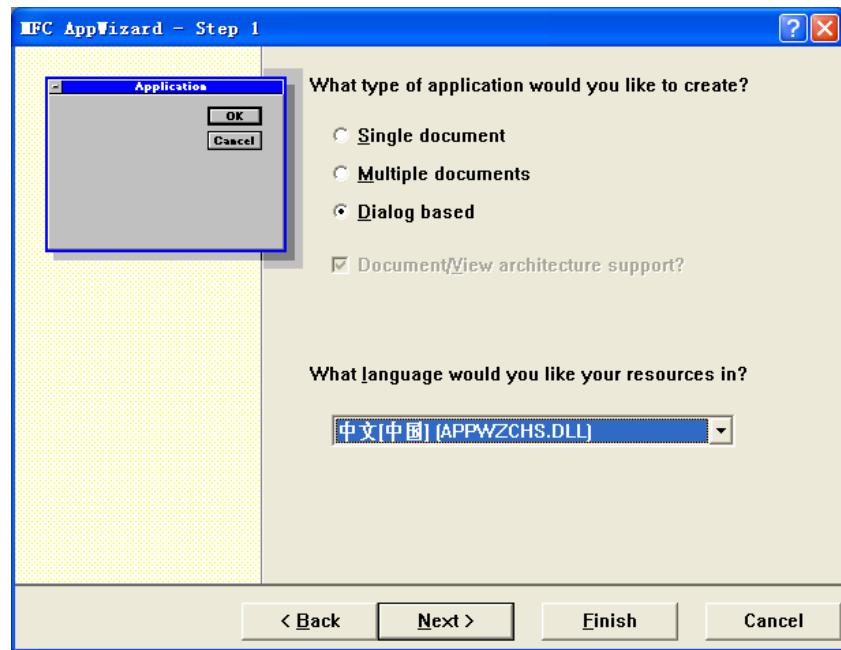


图 1-2

2、插入 Cursor 的光标，重新绘制该光标。

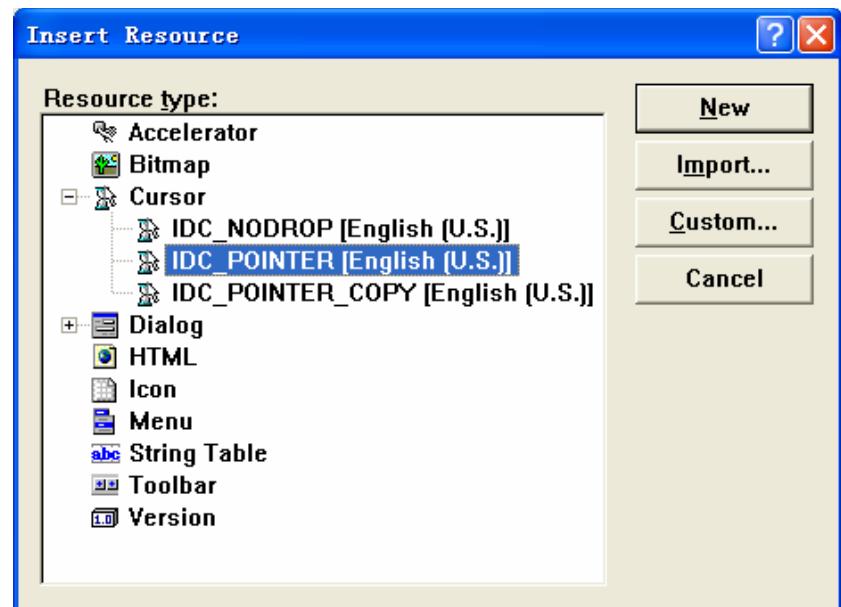


图 1-3

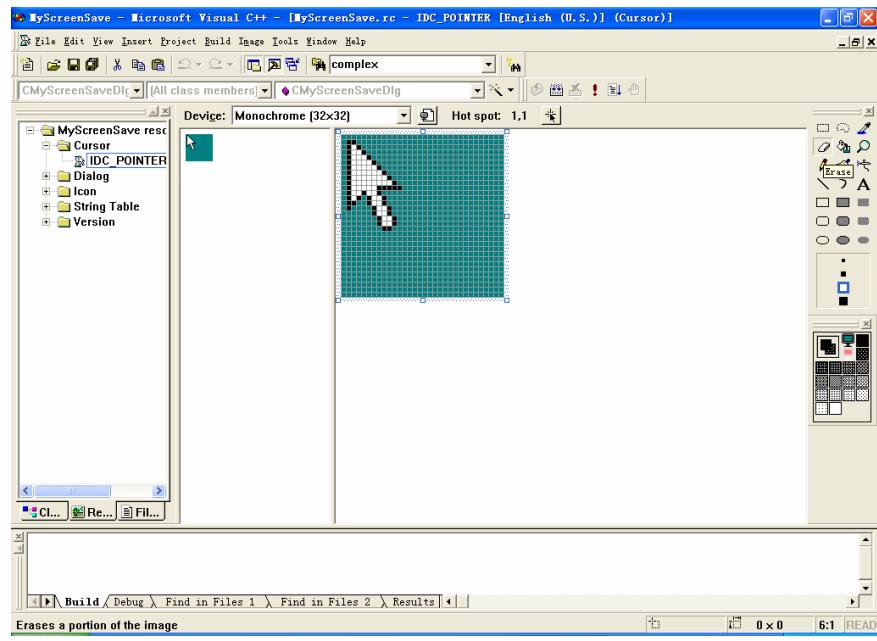


图 1-4

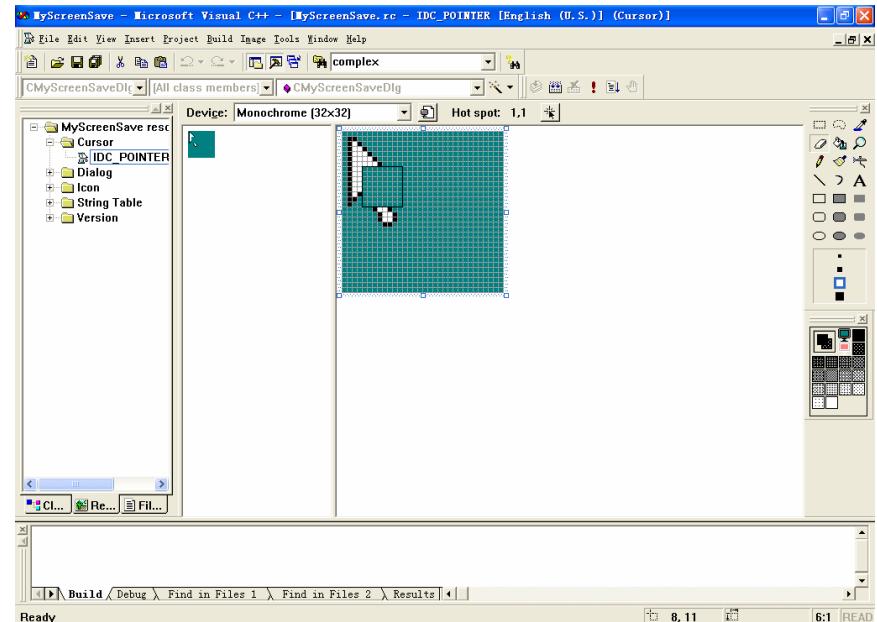


图 1-5

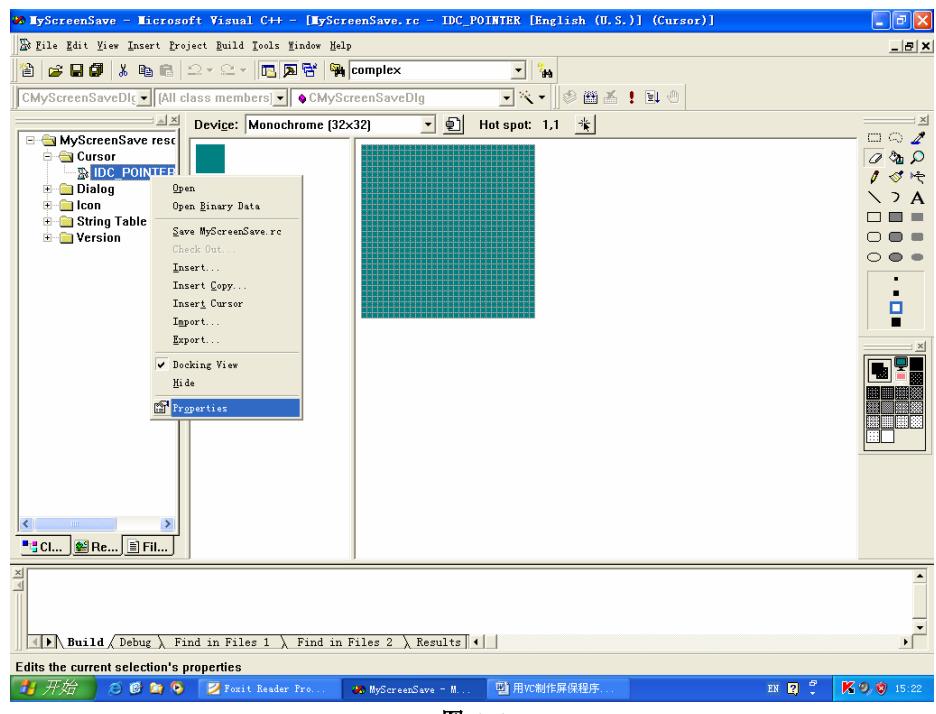


图 1-6

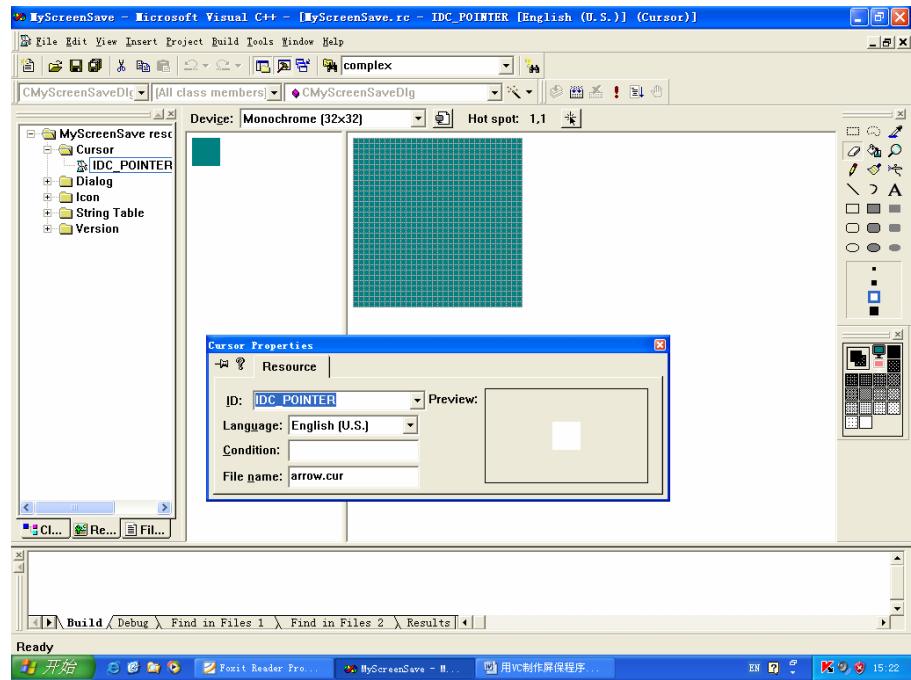


图 1-7

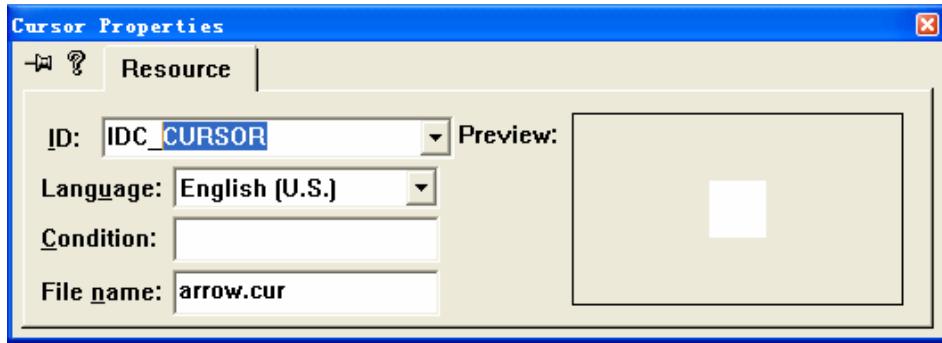


图 1-8

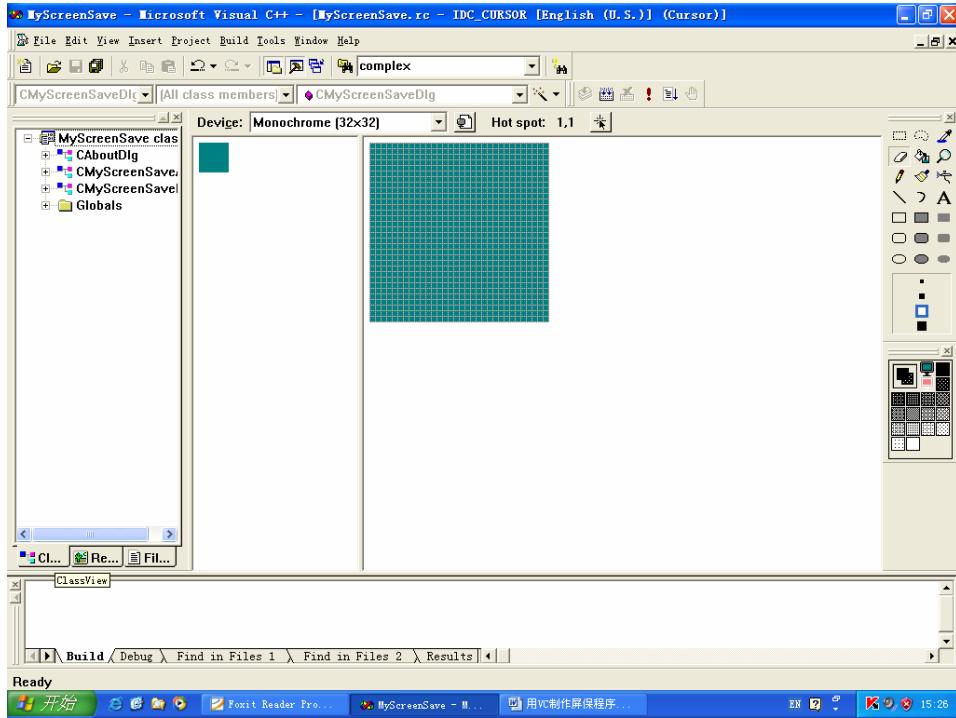


图 1-9

3、建立以 CWnd 为基类的窗口类 CScreenWnd，并且添加一个私有变量。

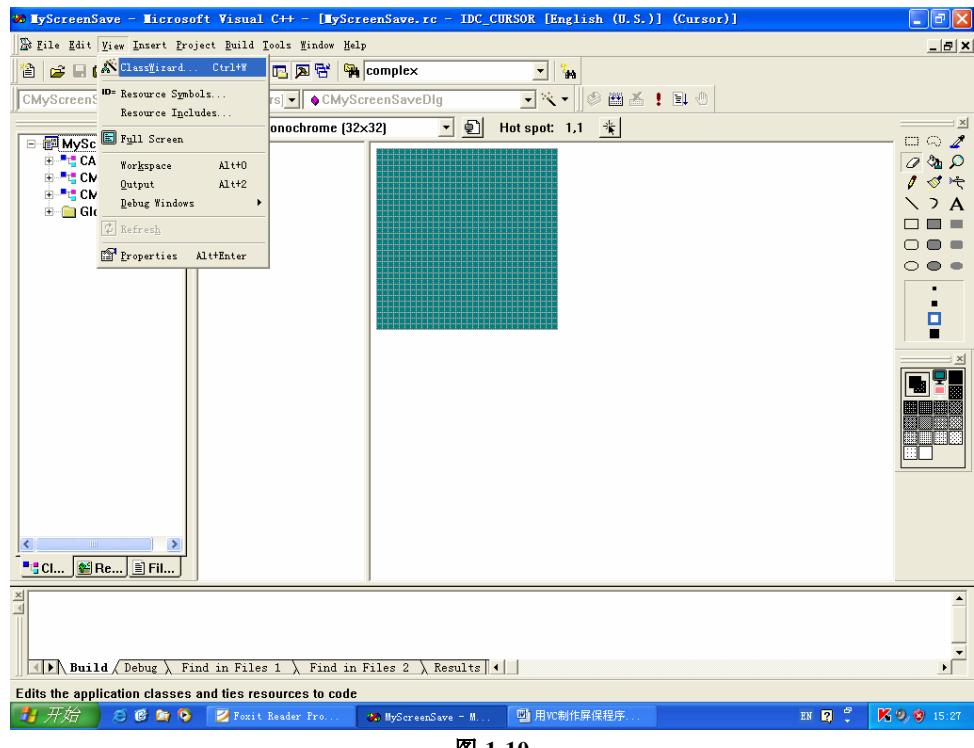


图 1-10

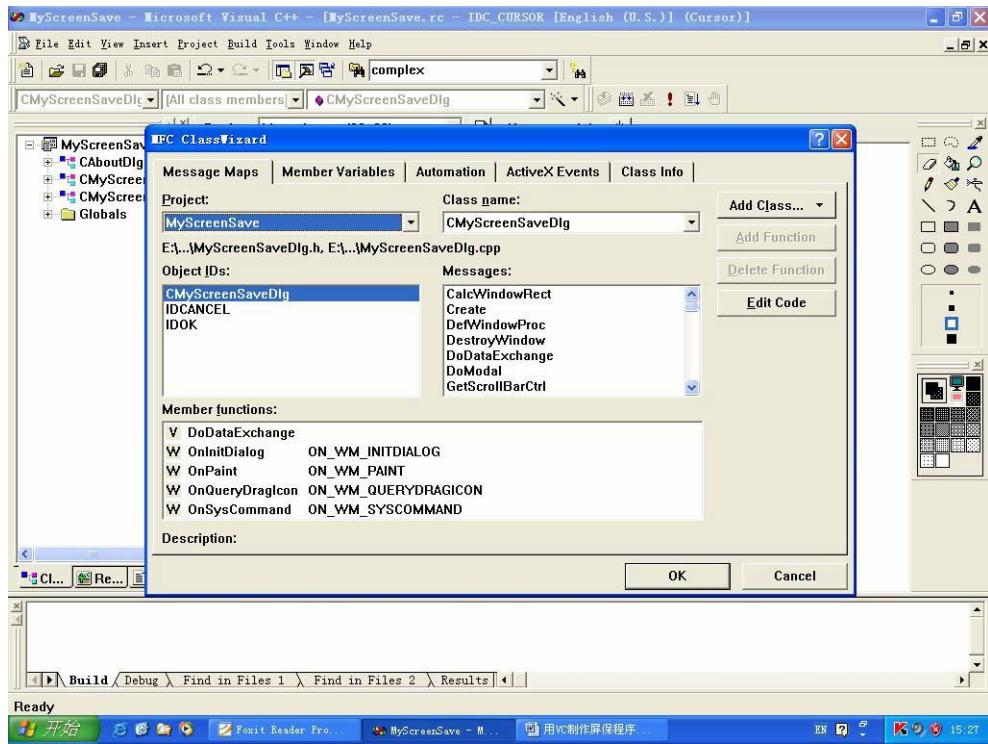


图 1-11

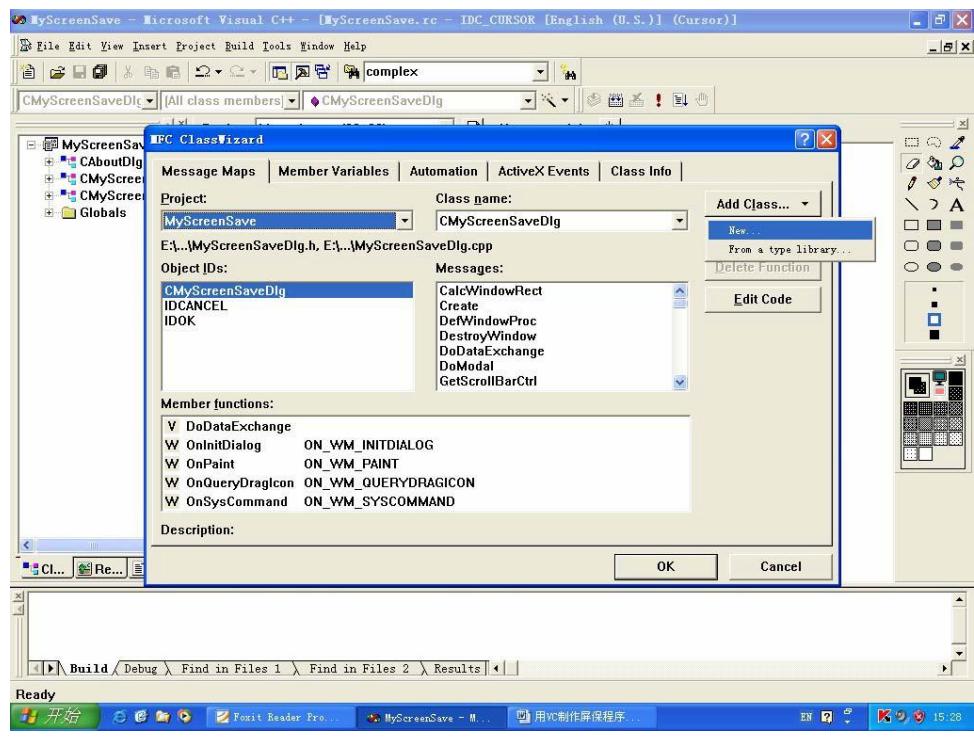


图 1-12

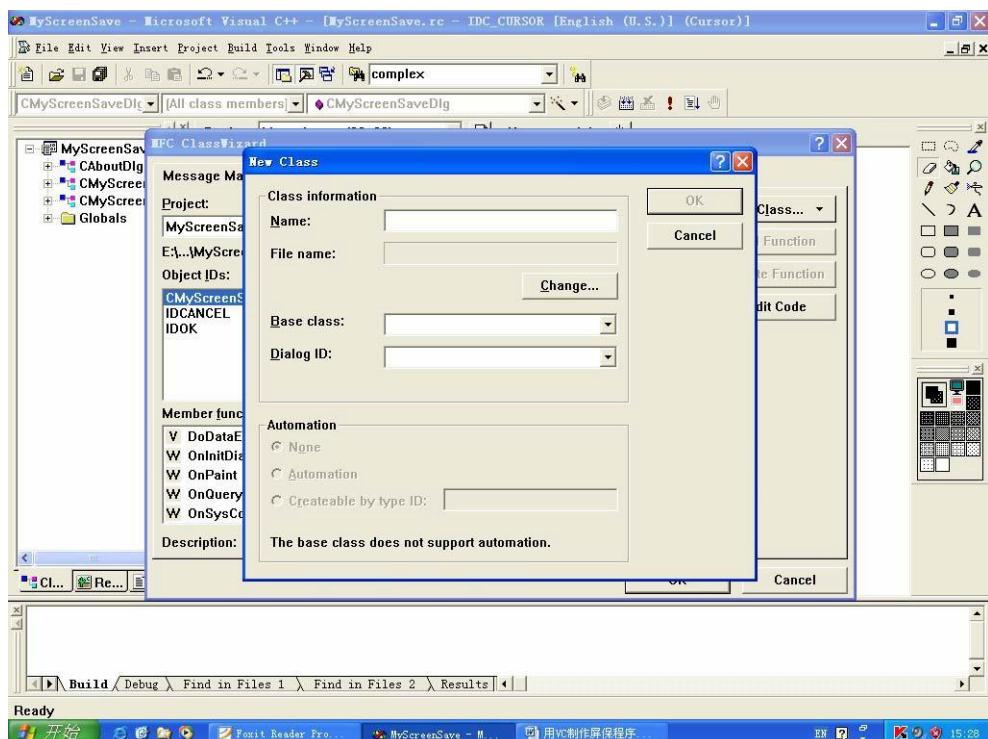


图 1-13

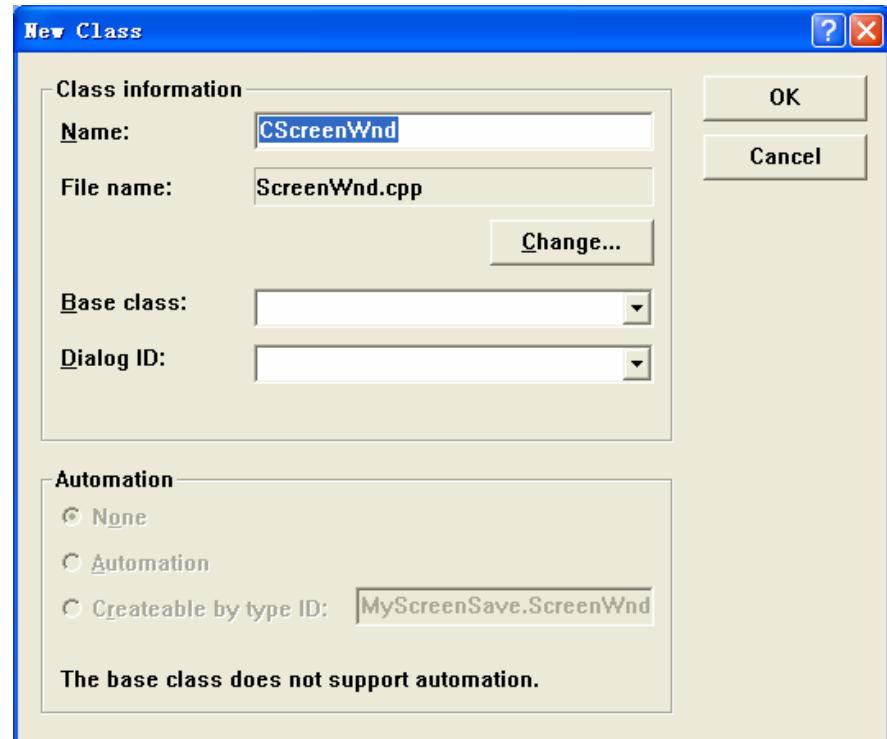


图 1-14

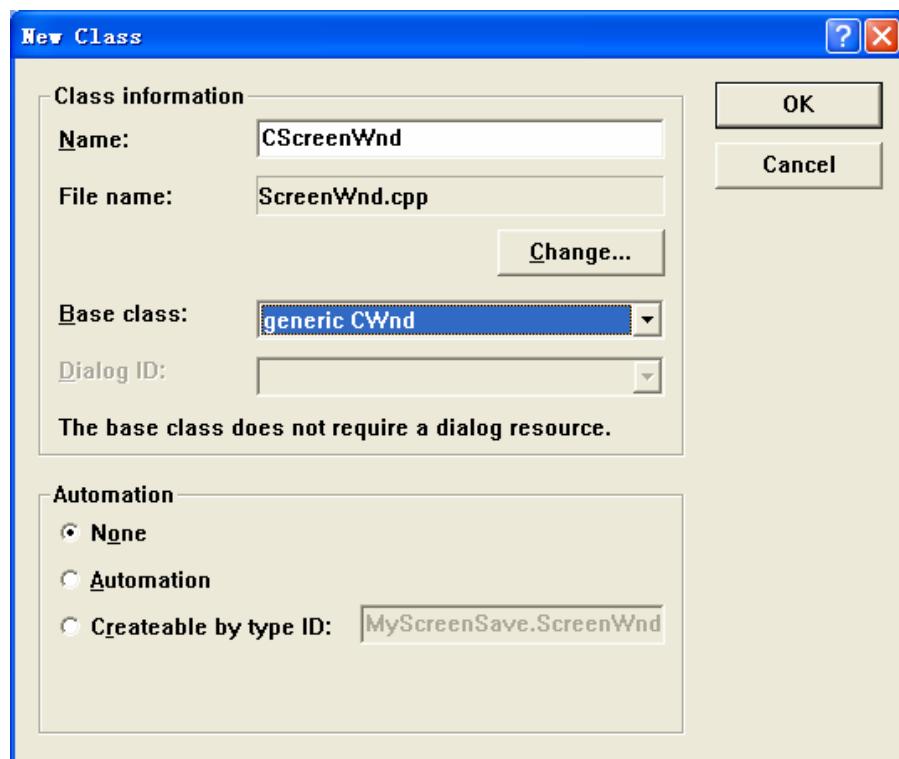


图 1-15

点击 OK 后返回到如下界面：

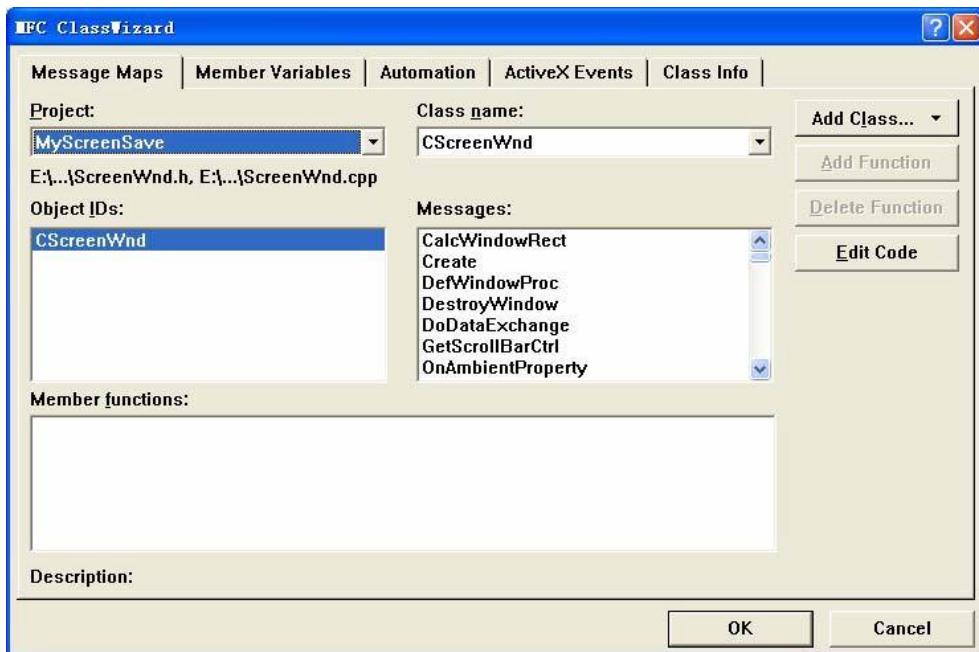


图 1-16

再次点击 OK 返回到如下界面：

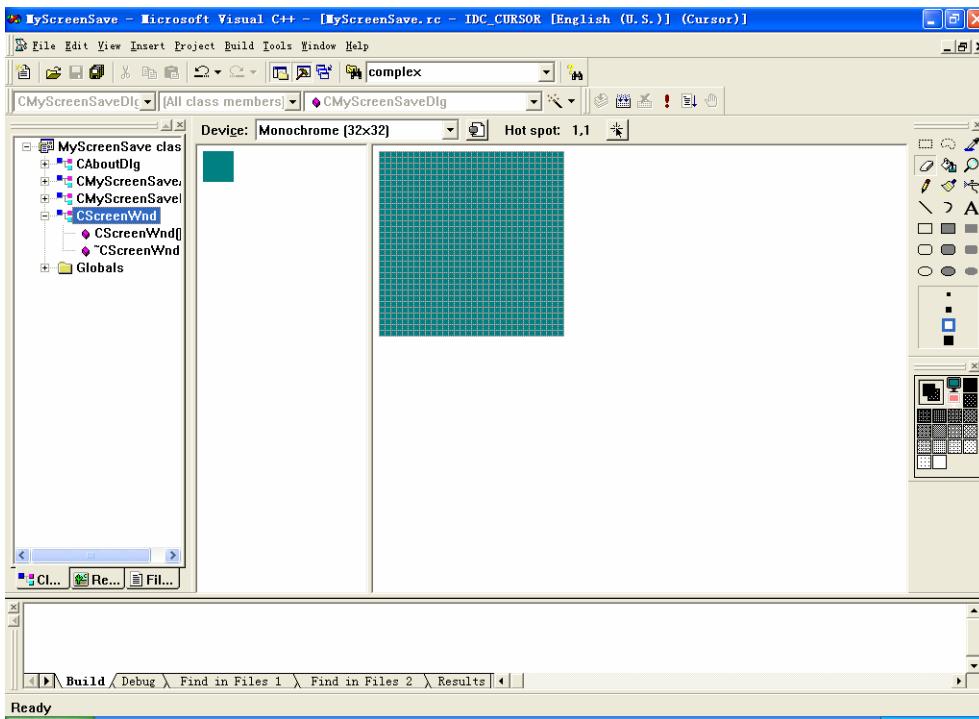


图 1-17

在 CScreenWnd 类中添加私有变量

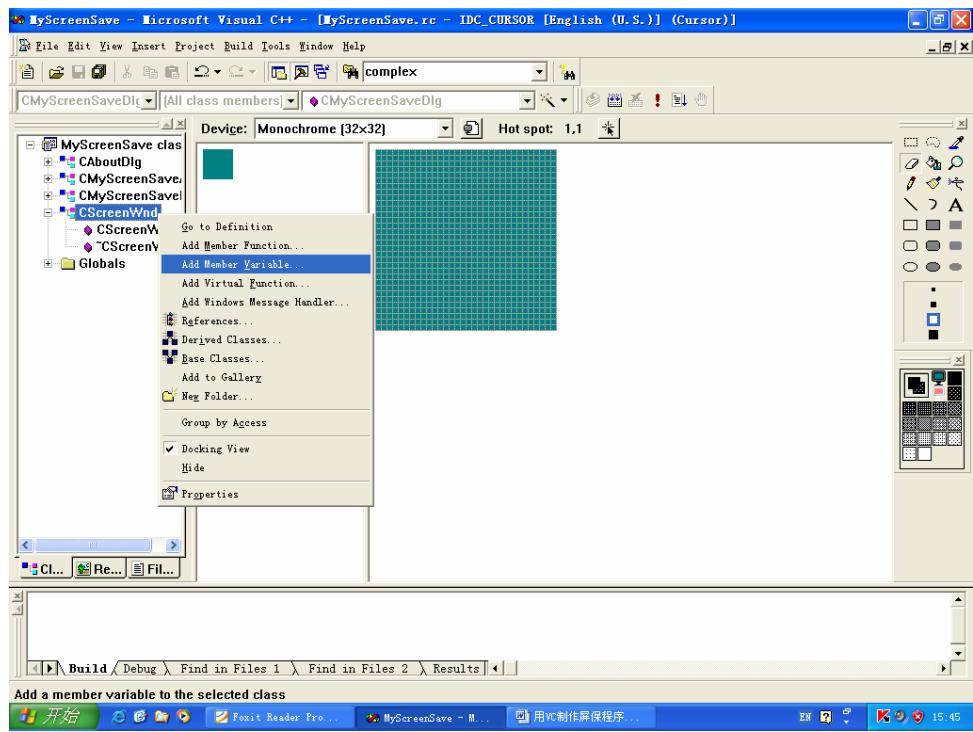


图 1-18

私有变量 lpszClassName，类型 (Variable Type) : LPCTSTR，权限 Access: Private

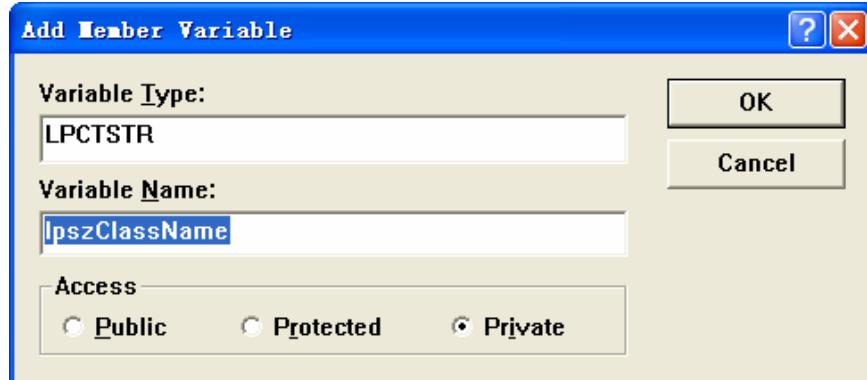


图 1-19

再次点击 OK 返回到如下界面：

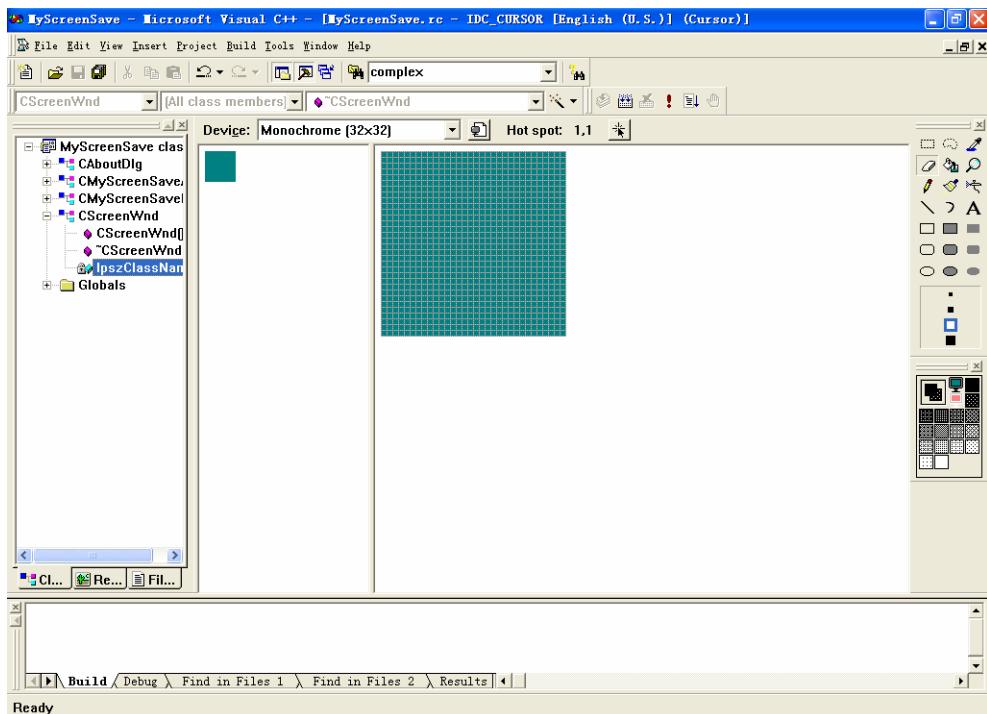


图 1-20

双击 lpszClassName，得到它的定义部分如下图：

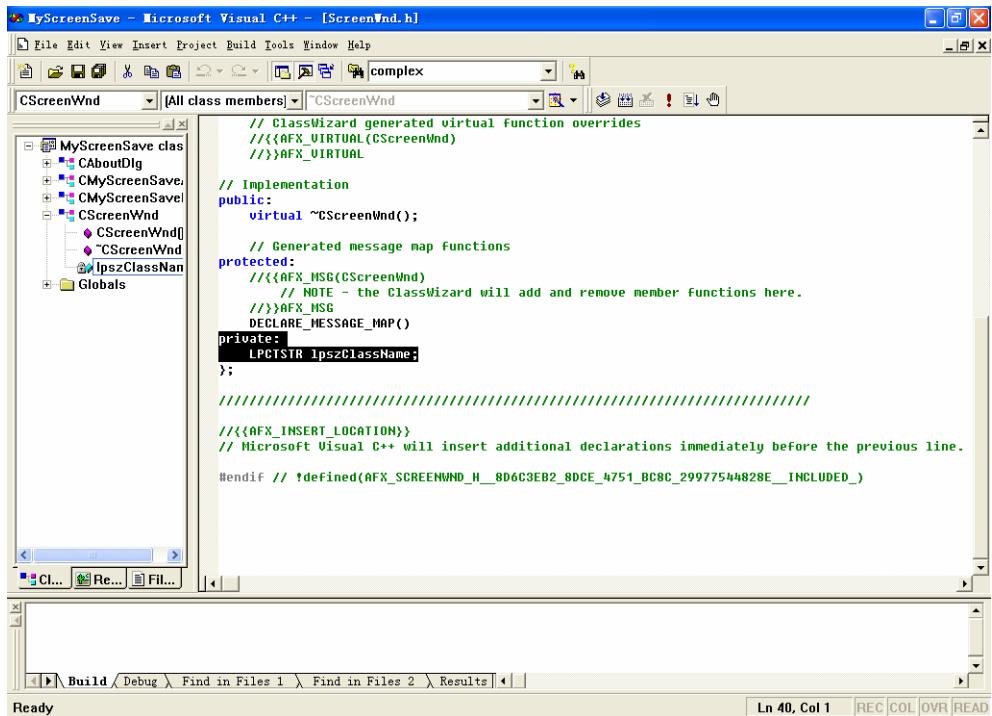


图 1-21

双击 CScreenWnd 类的构造函数，得到如下图所示部分：

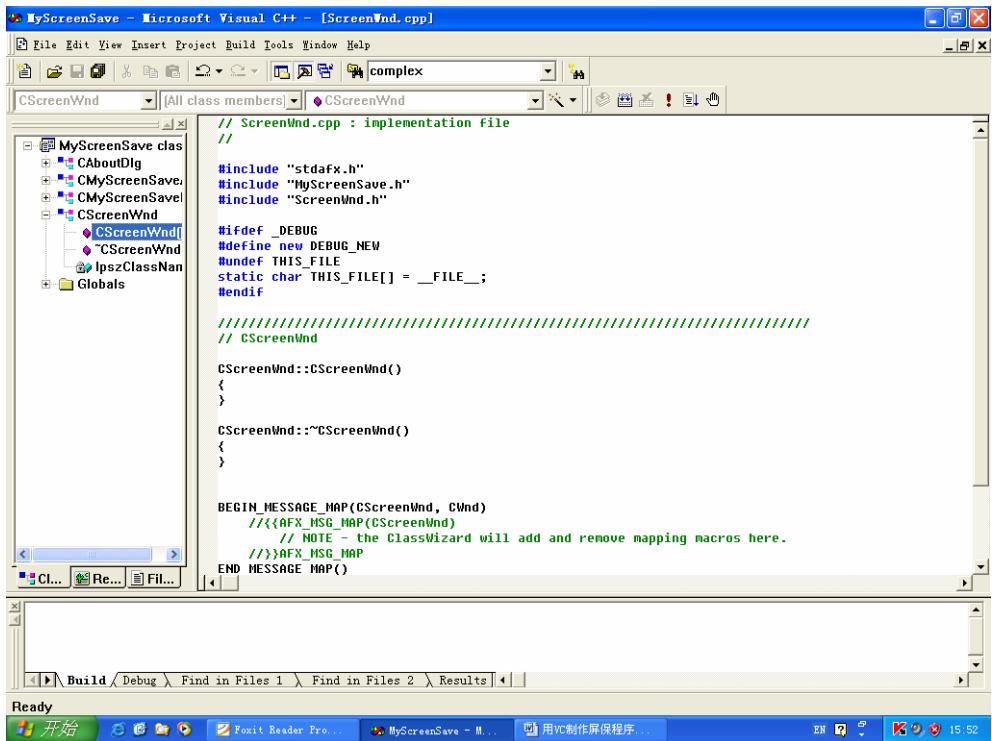


图 1-22

在构造函数体内部，添加代码：

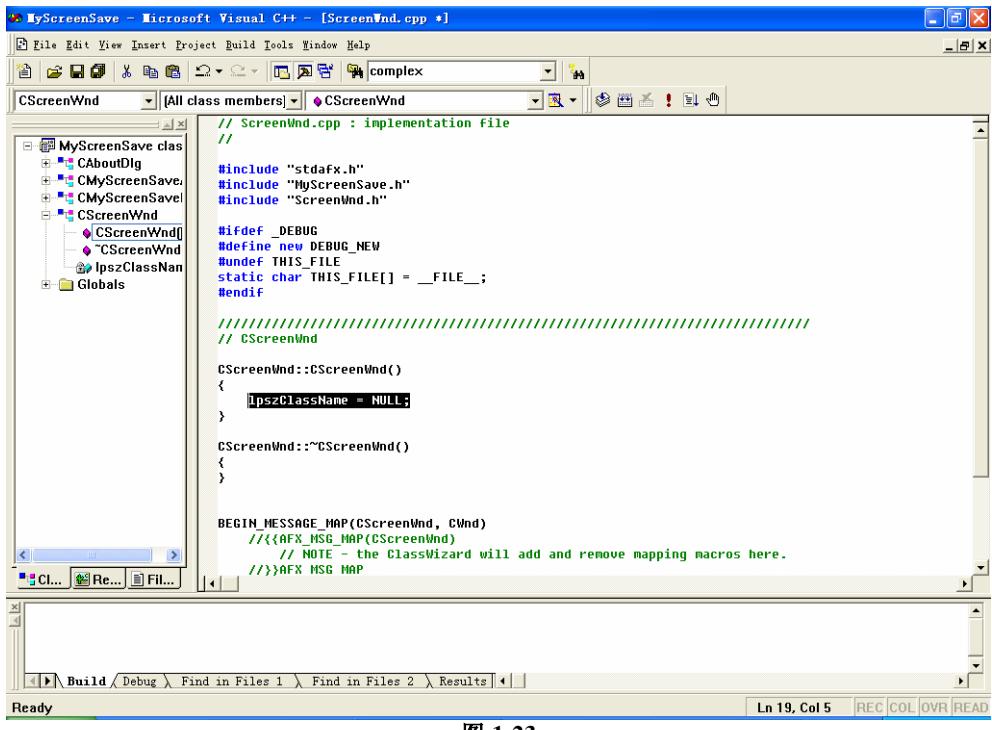


图 1-23

4、在 CScreenWnd 类中添加 CreateScreenWnd 函数

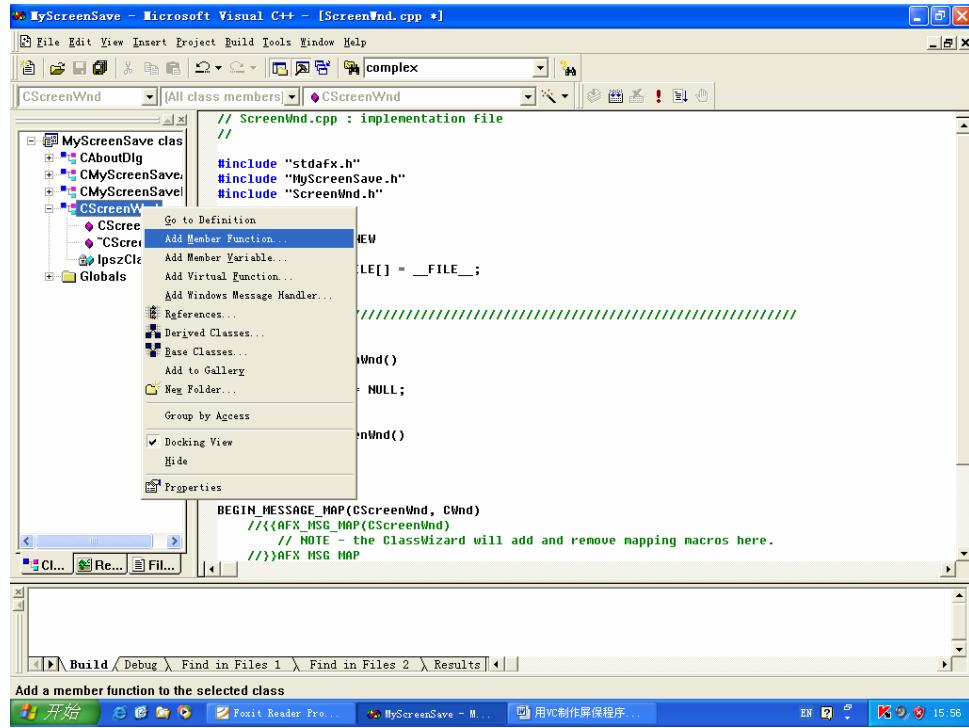


图 1-24

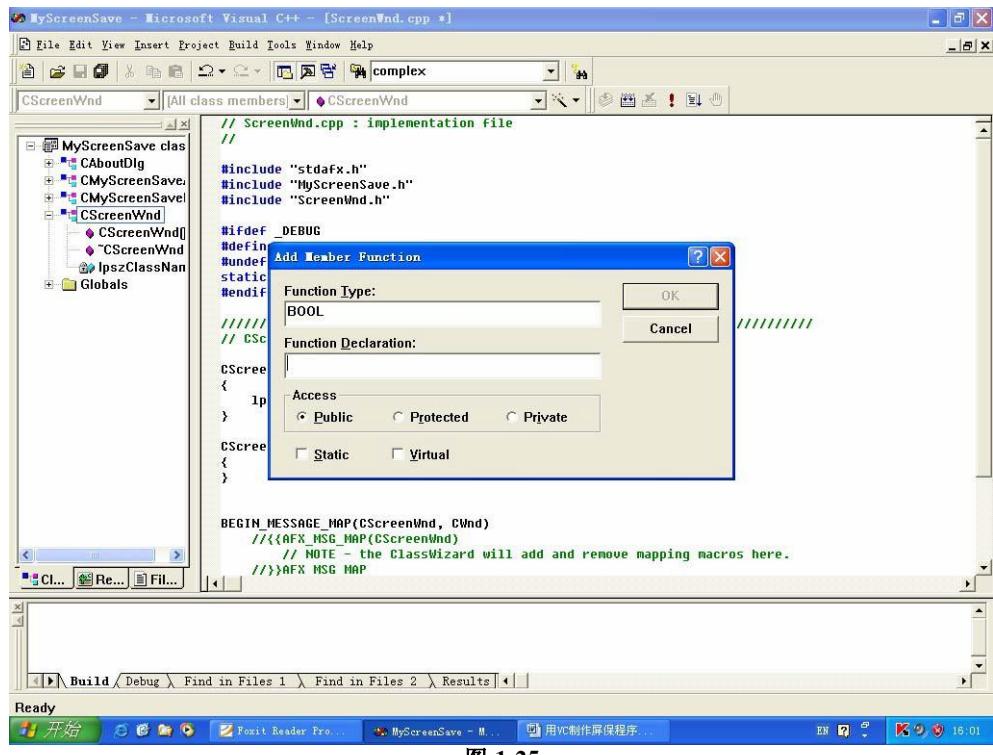


图 1-25



图 1-26

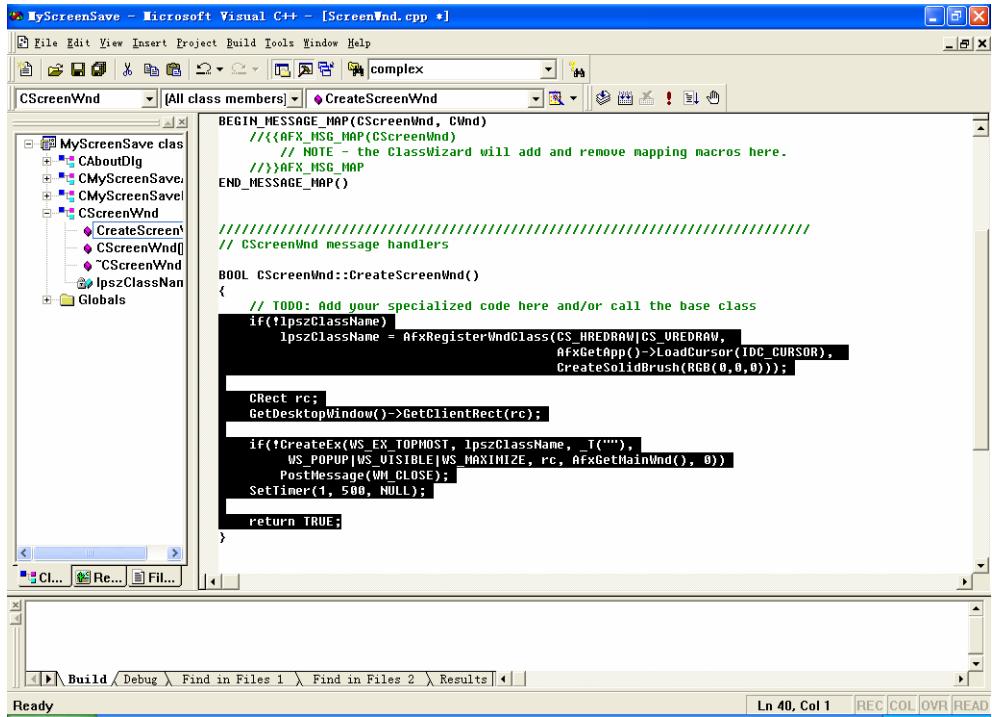


图 1-27

5、选中 CScreenWnd 类，然后调出 ClassWizard 菜单（或者 Ctrl + W）

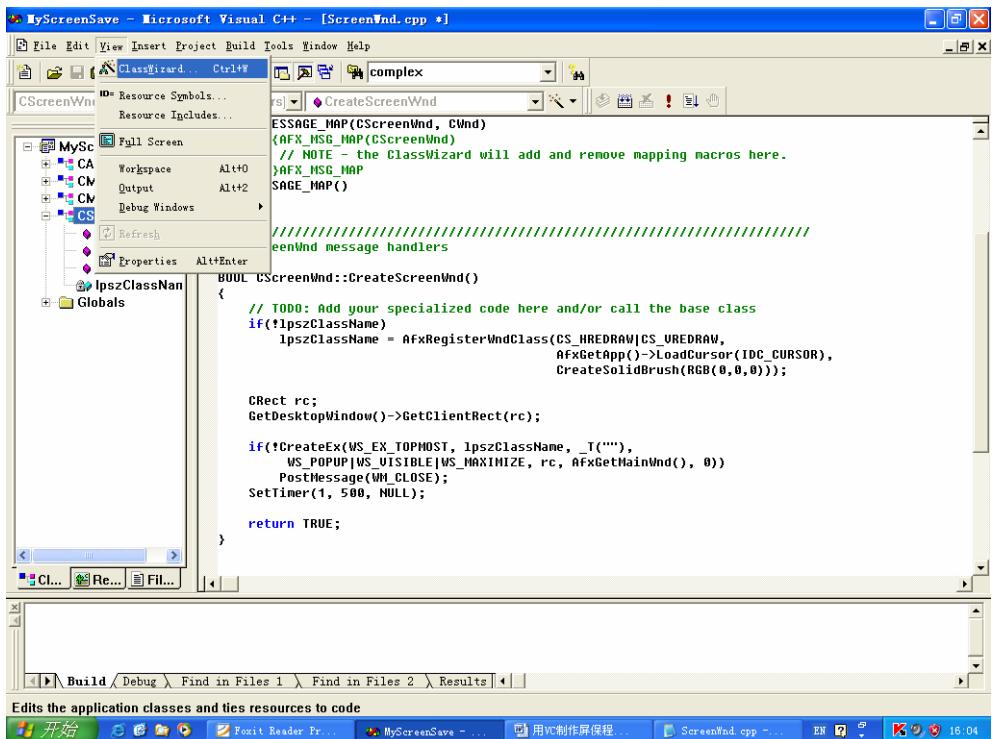


图 1-28

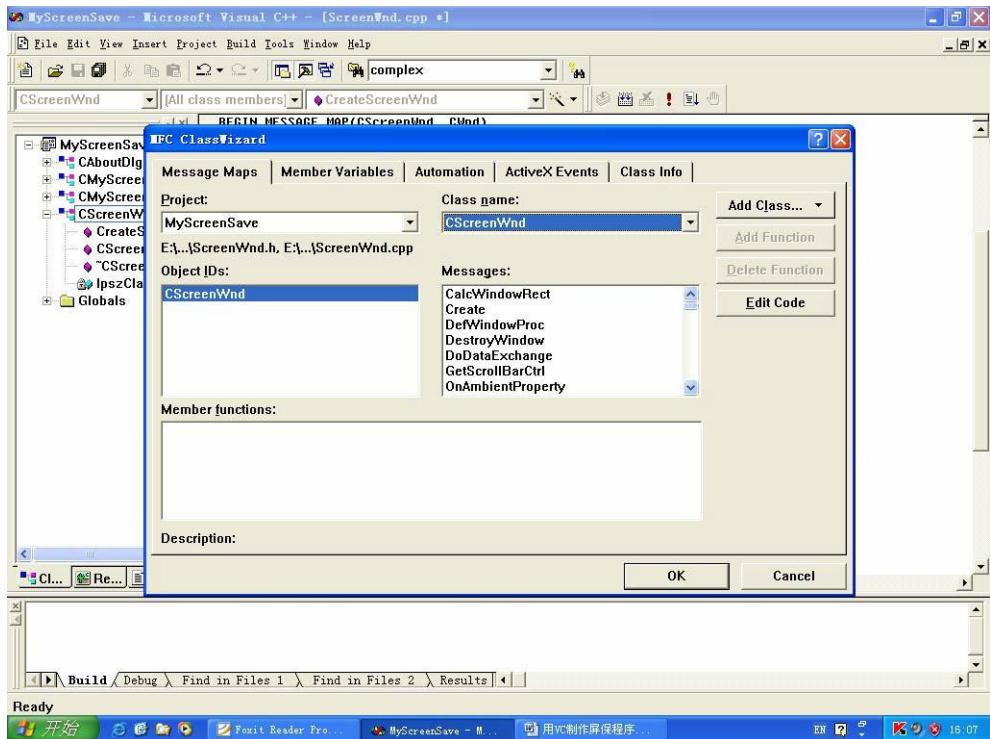


图 1-29

选中 Class name : CScreenWnd 类后，在 Messages 下拉框中选中 WM_TIMER 消息事件，后面的 WM_ 的消息事情，都是如此选择，图示如下：

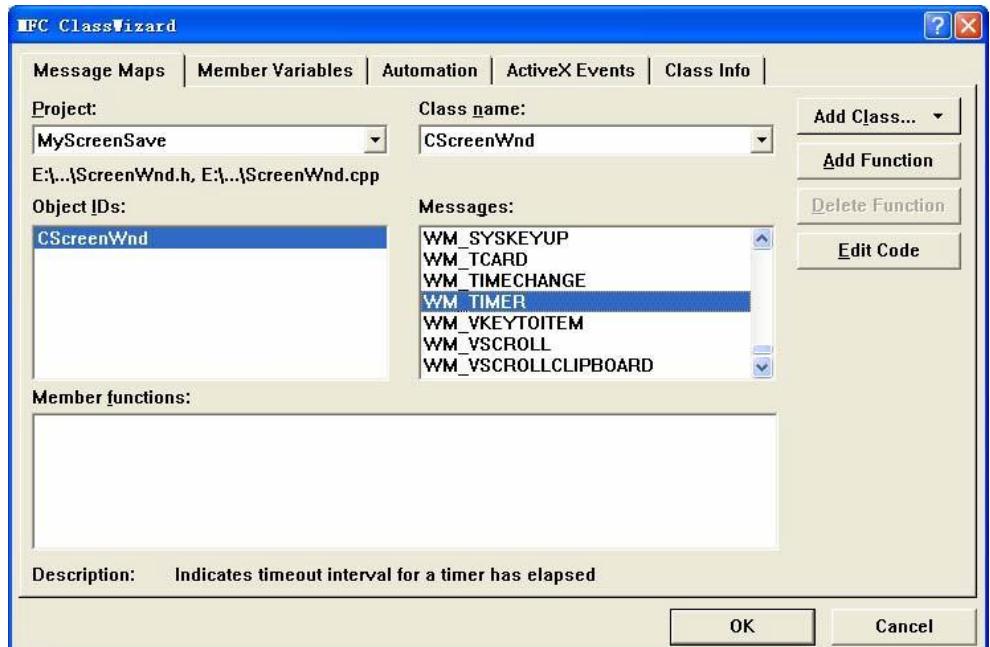


图 1-30

点击右侧的 Add Function 按钮，在 Member functions 框中，可以看到增添了一个消息函数 OnTimer，如下图所示：

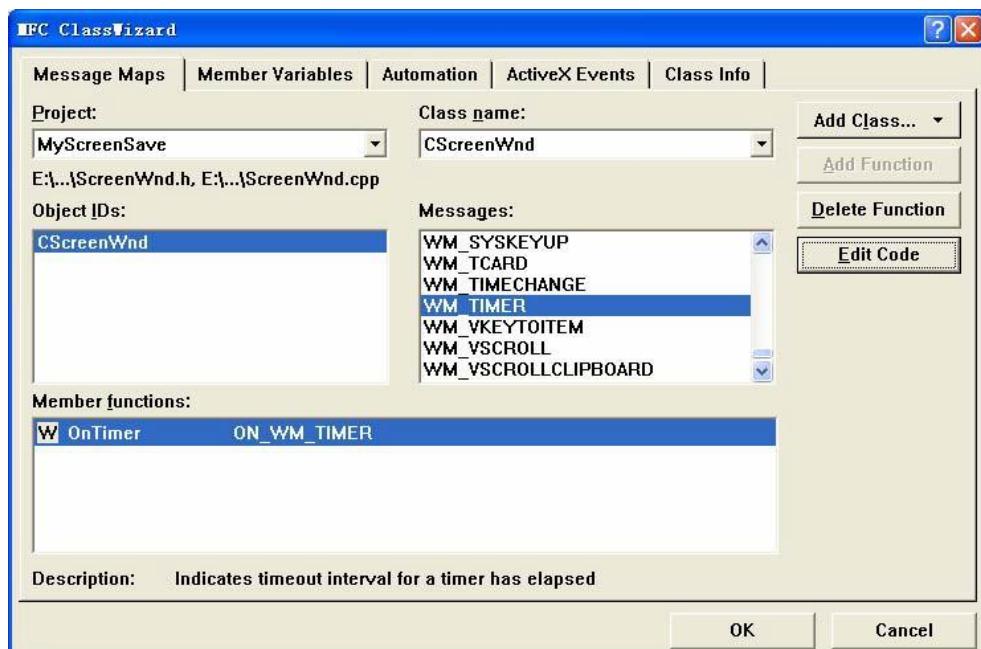


图 1-31

然后点击右侧的 Edit Code 按钮，出现如下窗口，在相关位置添加代码即可。

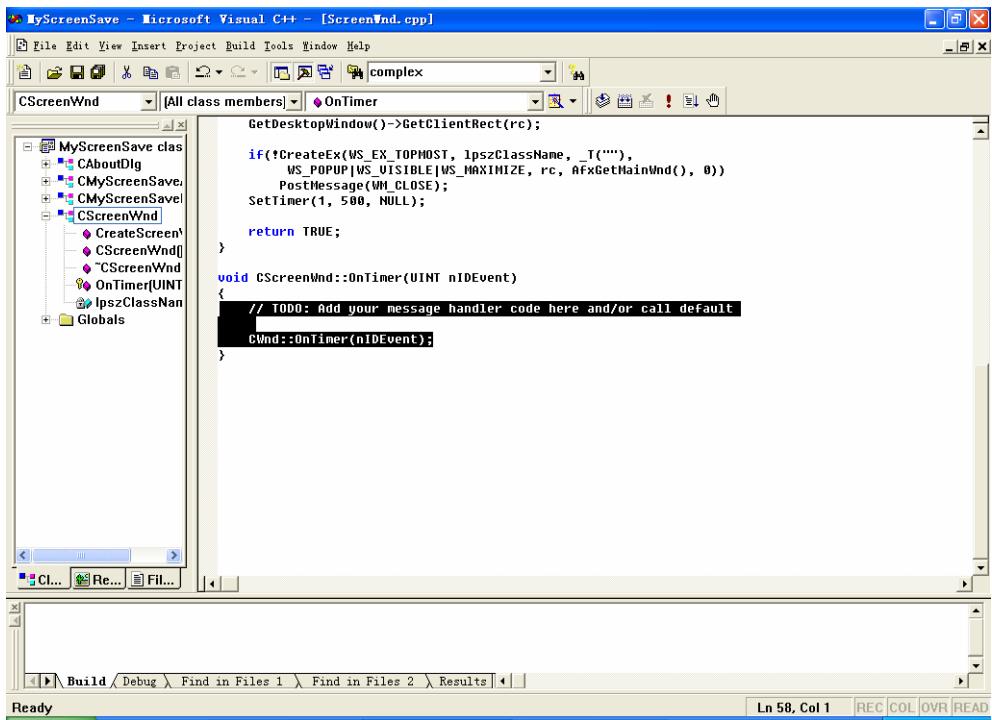


图 1-32

```

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]
File Edit View Insert Project Build Tools Window Help
complex
CScreenWnd [All class members] OnTimer
GetDesktopWindow()->GetClientRect(rc);
if(!CreateEx(WS_EX_TOPMOST, lpszClassName, _T(""),
    WS_POPUP|WS_VISIBLE|WS_MAXIMIZE, rc, AfxGetMainWnd(), 0))
    PostMessage(WM_CLOSE);
SetTimer(1, 500, NULL);

return TRUE;
}

void CScreenWnd::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    Invalidate();
    CWnd::OnTimer(nIDEvent);
}

```

图 1-33

6、同 5 一样的方式增加 CScreenWnd 类的消息 WM_PAINT，并且在消息映射函数中添加相应代码，图示如下：

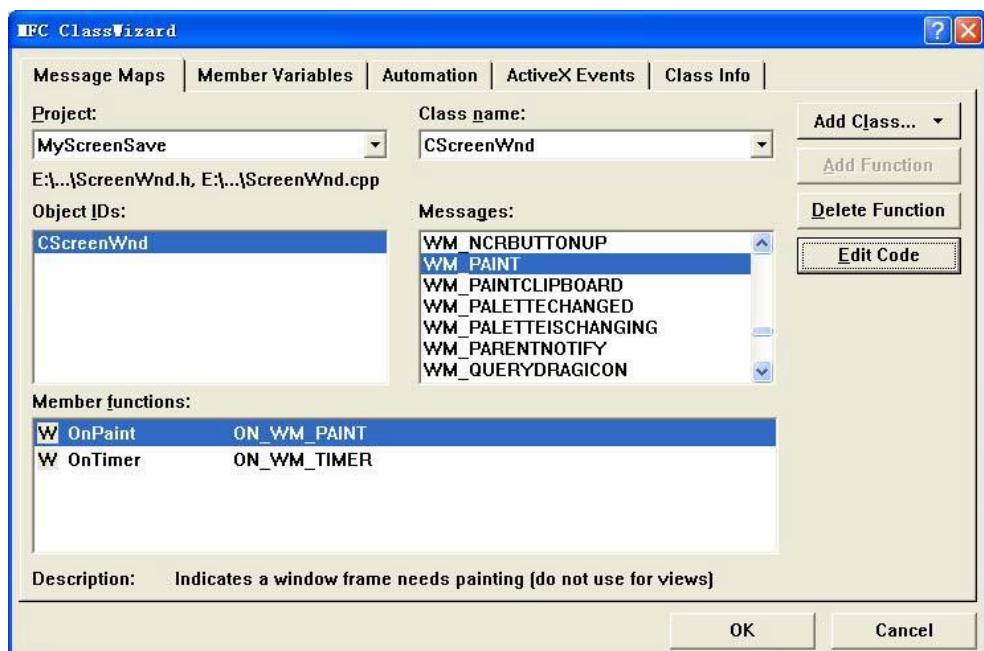


图 1-34

添加完消息 WM_PAINT 函数后，然后点击右侧的 Edit Code 按钮，出现如下窗口，在相关位置添加代码即可，图示如下：

```

void CScreenWnd::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    // TODO: Add your message handler code here

    Rect rect;
    GetClientRect(rect);
    CFont myFont;

    myFont.CreateFont(56, 20, 0, 0, FW_BOLD, FALSE, FALSE, FALSE, DEFAULT_CHARSET,
                      OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
                      DEFAULT_PITCH, "Verdana");

    CFont *pOldFont = dc.SelectObject(&myFont);

    dc.SetTextColor(RGB(abs(rand())%256, abs(rand())%256, abs(rand())%256));
    dc.SetBkColor(RGB(0,0,0));

    CString str("My Screen Savers ! ");

    dc.ExtTextOut(rect.Width()/2 - str.GetLength() * 10, rect.Height()/8 * 3,
                  ETO_OPAQUE, NULL, "My Screen Savers ! ", NULL);
    dc.SelectObject(pOldFont);

    // Do not call CWnd::OnPaint() for painting messages
}

```

图 1-35

7、同 5 一样的方式增加 CScreenWnd 类的消息 WM_CLOSE，并且在消息映射函数中添加相应代码，图示如下：

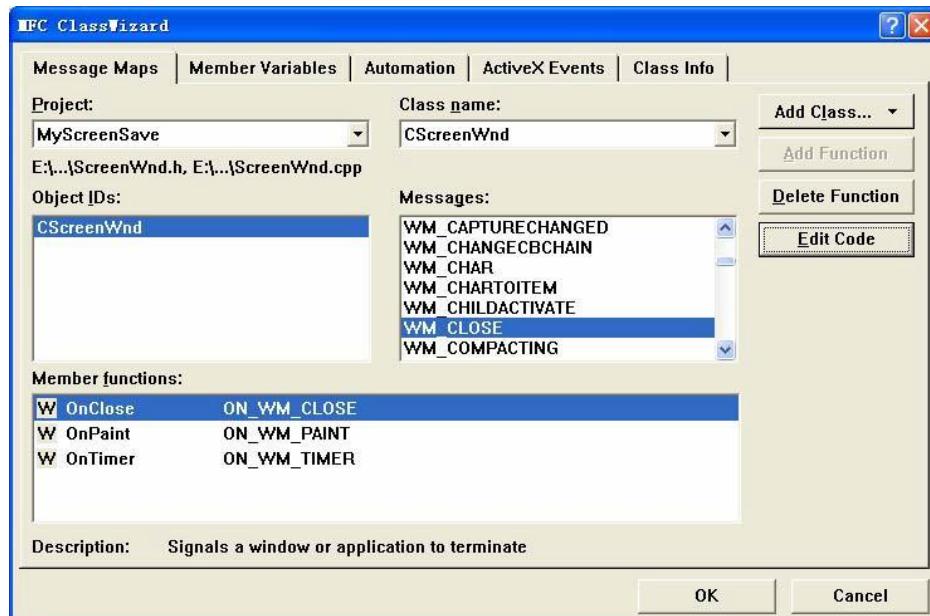


图 1-36

添加完消息 WM_CLOSE 函数后，然后点击右侧的 Edit Code 按钮，出现如下窗口，在相关位置添加代码即可，图示如下：

```

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]
File Edit View Insert Project Build Tools Window Help
complex
CScreenWnd [All class members] [OnClose]
CString str("My Screen Savers ! ");
dc.ExtTextOut(rect.Width()/2 - str.GetLength()*10, rect.Height()/8 * 3,
ETO_OPAQUE, NULL, "My Screen Savers ! ", NULL);
dc.SelectObject(pOldFont);

// Do not call CWnd::OnPaint() for painting messages
}

void CScreenWnd::OnClose()
{
    // TODO: Add your message handler code here and/or call default
    KillTimer(1);

    CWnd::OnClose();
}

```

图 1-37

8、同 5 一样的方式增加 CScreenWnd 类的消息 WM_ACTIVATE 和 WM_ACTIVATEAPP，并且在消息映射函数中添加相应代码，图示如下：

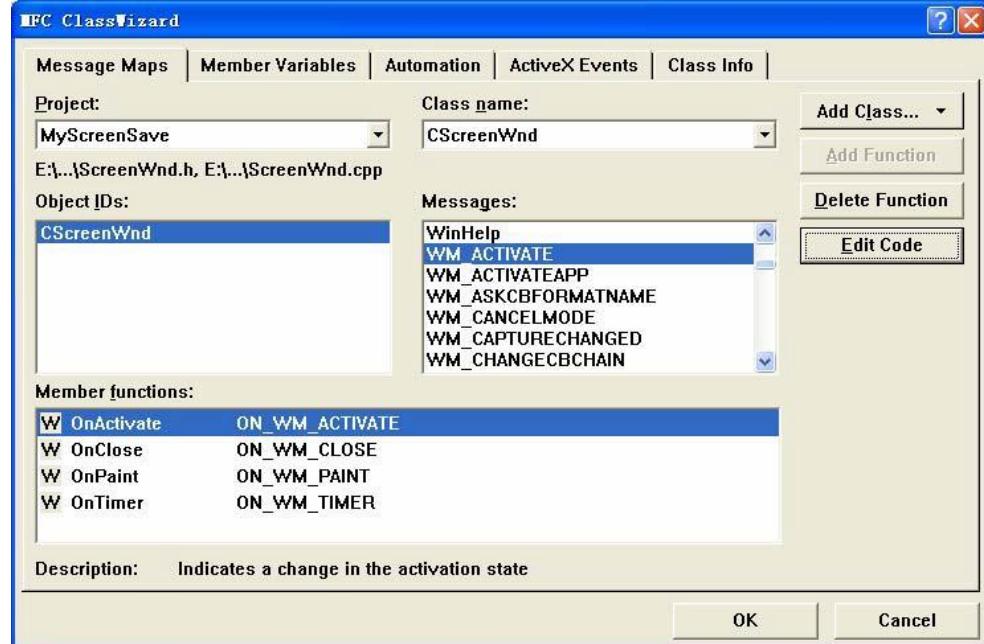


图 1-38

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

```

File Edit View Insert Project Build Tools Window Help
File View Insert Project Build Tools Window Help
CScreenWnd [All class members] OnActivate
CScreenWnd
MyScreenSave class
+ CAboutDlg
+ CMYScreensave
+ CMYScreensave
+ CScreenWnd
+ CreateScreen
+ CScreenWnd()
+ CScreenWnd
+ OnActivate(UINT)
+ OnClose()
+ OnPaint()
+ OnTimer(UINT)
+ IpszClassNames
+ Globals

CString str("My Screen Savers ! ");
dc.ExtTextOut(rect.Width()/2 - str.GetLength() *10, rect.Height()/8 *3,
ETO_OPAQUE, NULL, "My Screen Savers ! ", NULL);
dc.SelectObject(pOldFont);

// Do not call CWnd::OnPaint() for painting messages
}

void CScreenWnd::OnClose()
{
// TODO: Add your message handler code here and/or call default
KillTimer(1);
CWnd::OnClose();
}

void CScreenWnd::OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized)
{
CWnd::OnActivate(nState, pWndOther, bMinimized);

// TODO: Add your message handler code here

if(nState == WA_INACTIVE)
PostMessage(WM_CLOSE);
}

```

Build / Debug / Find in Files 1 / Find in Files 2 / Results | 4 |

Ready Ln 110, Col 1 REC COL OVR READ

图 1-39

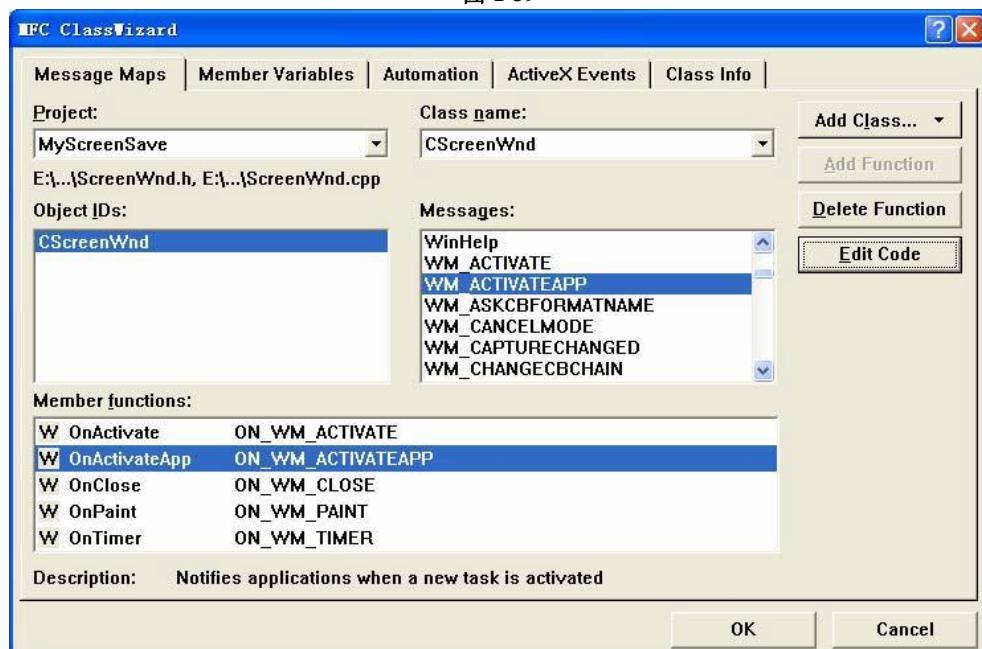


图 1-40

```

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]
File Edit View Insert Project Build Tools Window Help
CScreenWnd [All class members] OnActivateApp
{
    CWnd::OnActivate(nState, pWndOther, bMinimized);

    // TODO: Add your message handler code here

    if(nState == WA_INACTIVE)
        PostMessage(WM_CLOSE);
}

void CScreenWnd::OnActivateApp(BOOL bActive, HTASK hTask)
{
    CWnd::OnActivateApp(bActive, hTask);

    // TODO: Add your message handler code here

    if(! bActive)
        PostMessage(WM_CLOSE);
}

```

图 1-41

9、同 5 一 样的方 式增加 CScreenWnd 类 的消息 WM_LBUTTONDOWN、WM_RBUTTONDOWN、WM_MBUTTONDOWN、WM_MOUSEMOVE、WM_KEYDOWN 和 WM_SYSKEYDOWN 等消息，并且在消息映射函数中添加相应代码，图示如下：

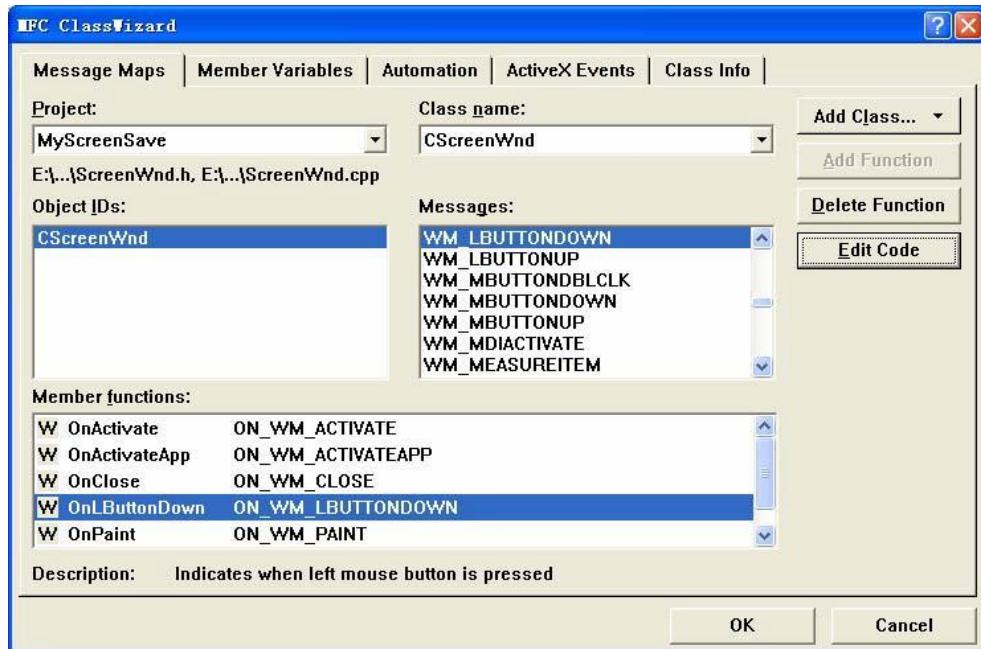


图 1-42

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

```

File Edit View Insert Project Build Tools Window Help
[All class members] OnLButtonDown

CScreenWnd
+ MyScreenSave class
  + CABoutDlg
  + CMyScreenSave
  + CMYScreensave
  + CScreenWnd
    + CreateScreen
    + CScreenWnd()
    + ~CScreenWnd
    + OnActivate(UINT)
    + OnActivateApp()
    + OnClose()
    + OnLButtonDown(UINT nFlags, CPoint point)
    + OnPaint()
    + OnTimer(UINT nIDEvent)
    + IpszClassNan
  + Globals

void CScreenWnd::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here

    if(nFlags & MK_RBUTTON)
        PostMessage(WM_RBUTTONDOWN);
}

void CScreenWnd::OnPaint()
{
    // TODO: Add your message handler code here and/or call default

    CPaintDC dc(this);
    dc.FillSolidColor(RGB(255, 255, 255));
}

```

Build | Debug | Find in Files 1 | Find in Files 2 | Results | Ln 131, Col 5 | REC | COL | OVR | READ

图 1-43

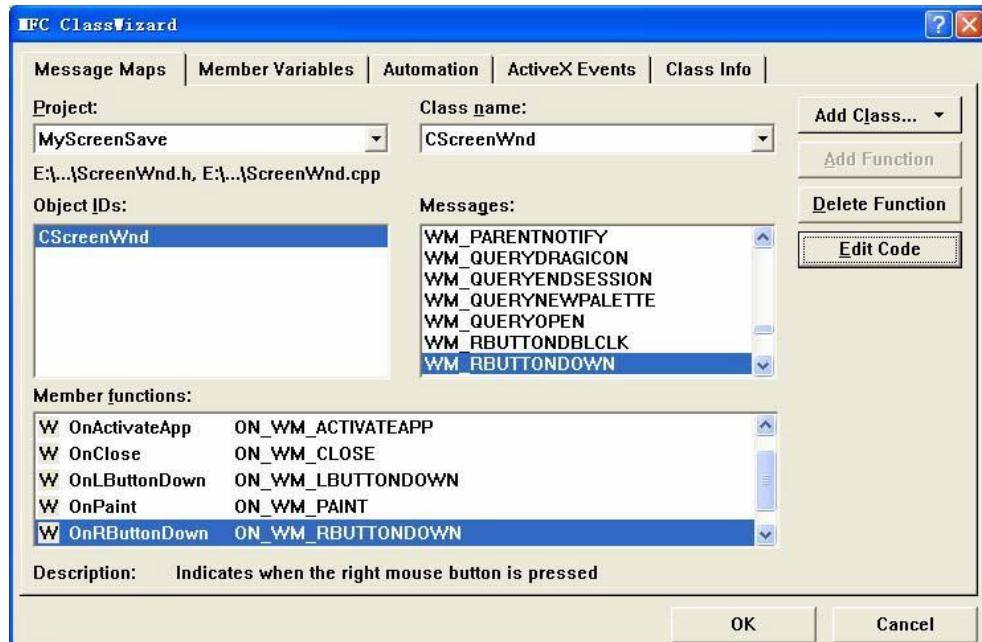


图 1-44

The screenshot shows the Microsoft Visual C++ IDE interface. The title bar reads "MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]". The menu bar includes File, Edit, View, Insert, Project, Build, Tools, Window, Help. The toolbar has icons for New, Open, Save, Print, and others. The status bar at the bottom shows "Ready" and "Ln 141, Col 5 REC COL OVR READ".

The left pane displays the class hierarchy:

- MyScreenSave class
 - CAboutDlg
 - CMyScreenSave
 - CMyScreenSave1
 - CScreenWnd
 - CreateScreen
 - CScreenWnd()
 - OnActivate
 - OnActivateApp
 - OnClose()
 - OnPaint()
 - OnRButtonDown
 - OnTimer(UINT)
 - IpszClassNan

The right pane shows the code for the `CScreenWnd` class:

```
void CScreenWnd::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    PostMessage(WM_CLOSE);

    CWnd::OnLButtonDown(nFlags, point);
}

void CScreenWnd::OnRButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    PostMessage(WM_CLOSE);

    CWnd::OnRButtonDown(nFlags, point);
}
```

图 1-45

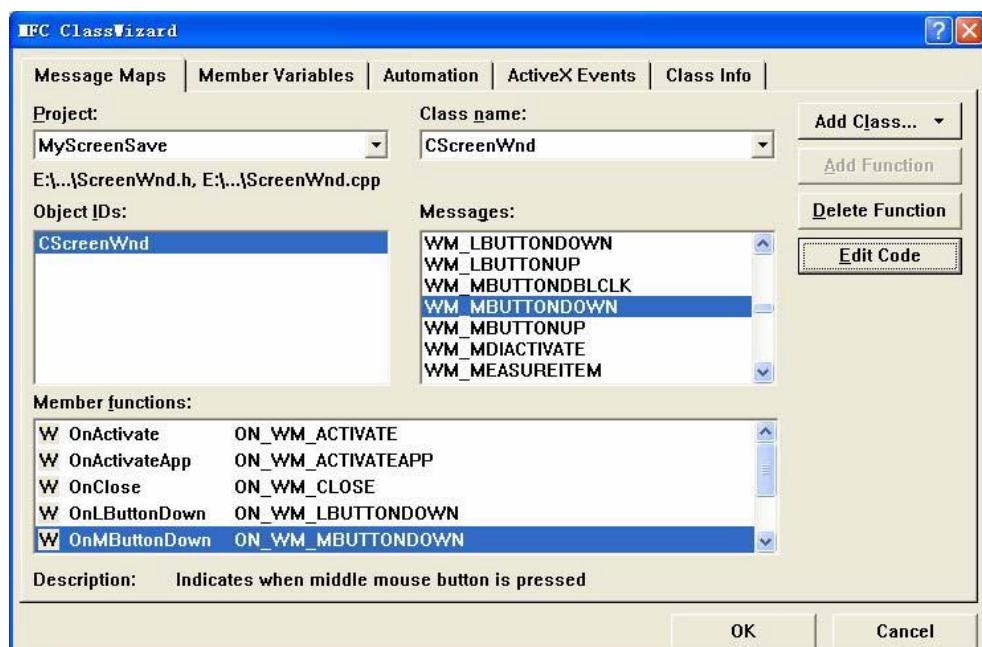


图 1-46

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

```

File Edit View Insert Project Build Tools Window Help
[All class members] OnLButtonDown
CScreenWnd

void CScreenWnd::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    PostMessage(WM_CLOSE);
    CWnd::OnLButtonDown(nFlags, point);
}

void CScreenWnd::OnRButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    PostMessage(WM_CLOSE);
    CWnd::OnRButtonDown(nFlags, point);
}

void CScreenWnd::OnNButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    PostMessage(WM_CLOSE);
    CWnd::OnNButtonDown(nFlags, point);
}

```

Ready Ln 151, Col 5 REC COL OVR READ

图 1-47

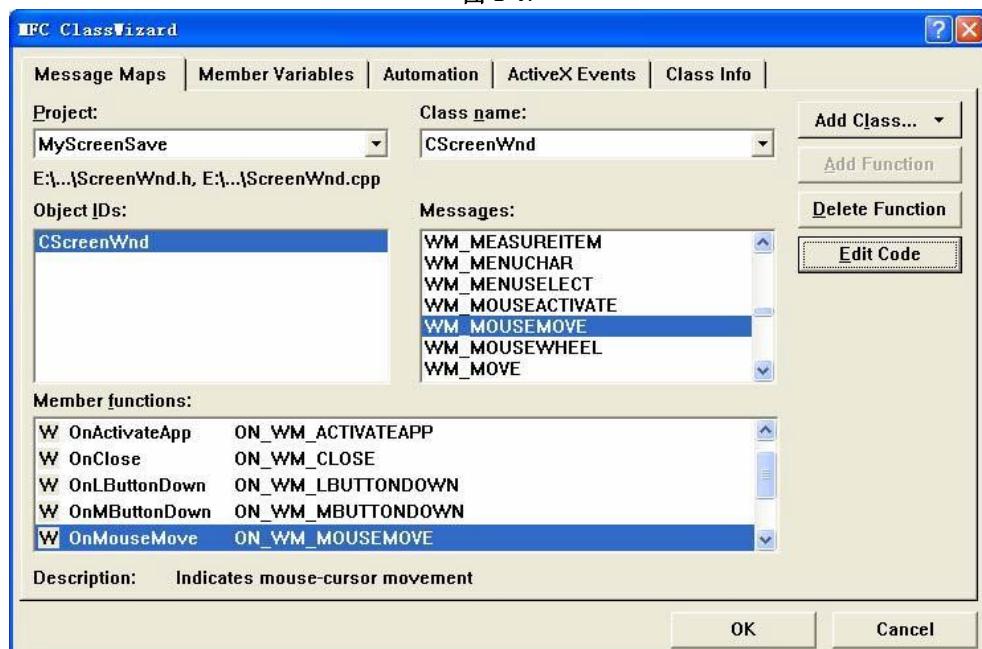


图 1-48

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

File Edit View Insert Project Build Tools Window Help

complex

CScreenWnd [All class members] OnMouseMove

MyScreenSave class

+ CABoutDlg

+ CMysScreenSave

+ CMysScreenSave

+ CScreenWnd

+ CreateScreen

+ CScreenWnd()

+ CScreenWnd

+ OnActivate(U32)

+ OnActivateApp

+ OnClose()

+ OnLButtonDown

+ OnLButtonDo

+ OnMouseMov

+ OnPaint()

+ OnRButtonDown

+ OnTimer(UINT)

+ IpszClassNam

+ Globals

```
    CWnd::OnRButtonDown(nFlags, point);  
}  
  
void CScreenWnd::OnMButtonDown(UINT nFlags, CPoint point)  
{  
    // TODO: Add your message handler code here and/or call default  
  
    PostMessage(WM_CLOSE);  
  
    CWnd::OnMButtonDown(nFlags, point);  
}  
  
void CScreenWnd::OnMouseMove(UINT nFlags, CPoint point)  
{  
    // TODO: Add your message handler code here and/or call default  
  
    static CPoint staticPoint(-1, -1);  
    CPoint temp(-1, -1); [ ]  
    if(temp!=staticPoint)  
        PostMessage(WM_CLOSE);  
    else staticPoint = point;  
  
    CWnd::OnMouseMove(nFlags, point);  
}
```

Cl... Re... Fil... |

Build Debug Find in Files 1 Find in Files 2 Results |

Ready

Ln 161, Col 1 REC COL OVR READ

图 1-49

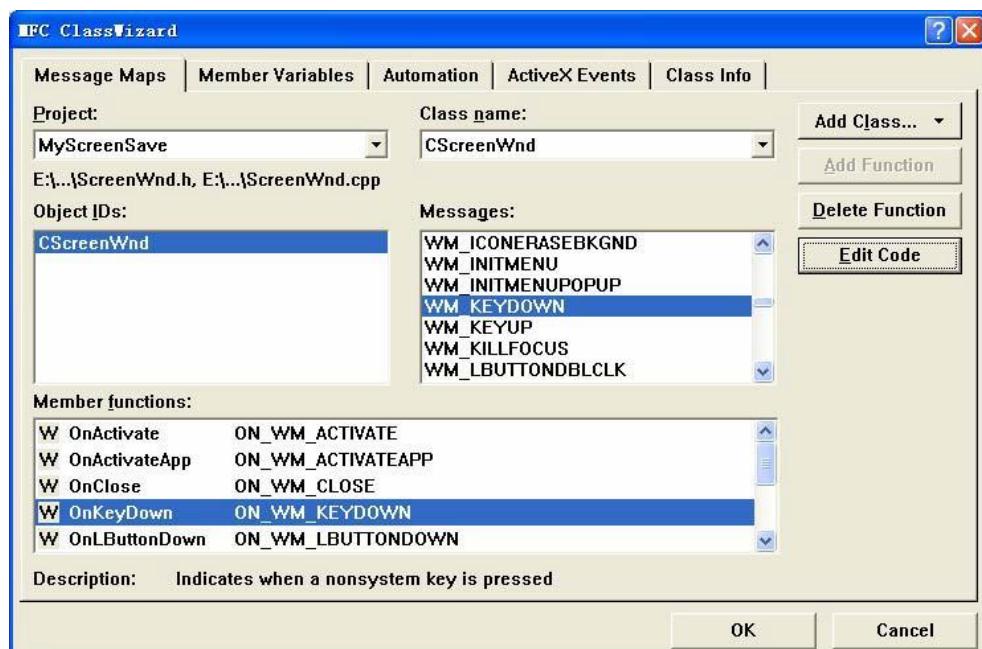


图 1-50

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

File Edit View Insert Project Build Tools Window Help

complex

CScreenWnd [All class members] OnKeyDown

MyScreenSave class

CAboutDlg

CMyScreenSave

CMyScreenSave

CScreenWnd

CreateScreen

CScreenWnd()

CScreenWnd

OnActivate(UIL)

OnActivateApp

OnClose()

OnKeyDown(L

OnLButtonDo

OnMButtonDo

OnMouseMov

OnPaint()

OnRButtonDo

OnTimer(UINT

IpszClassNam

Globals

```
    CWnd::OnMButtonDown(nFlags, point);

}

void CScreenWnd::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    static CPoint staticPoint(-1, -1);
    CPoint temp(-1, -1);
    if(temp!=staticPoint)
        PostMessage(WM_CLOSE);
    else staticPoint = point;

    CWnd::OnMouseMove(nFlags, point);
}

void CScreenWnd::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    // TODO: Add your message handler code here and/or call default

    PostMessage(WM_CLOSE);

    CWnd::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

Cl... Re... Fil...

Build Debug Find in Files 1 Find in Files 2 Results

Ready

Ln 175, Col 5 REC COL OVR READ

图 1-51

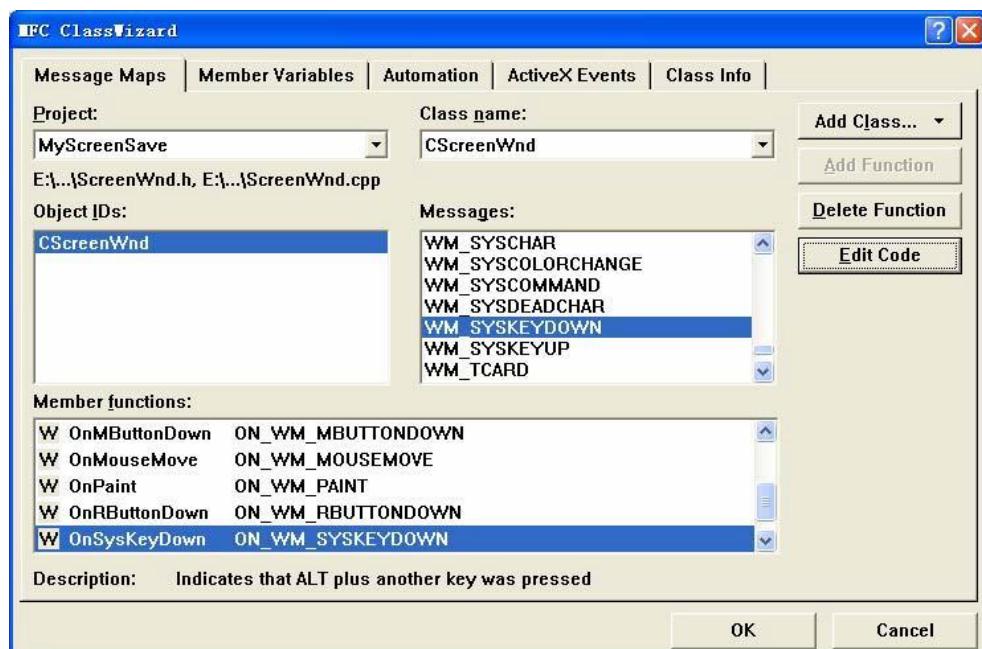


图 1-52

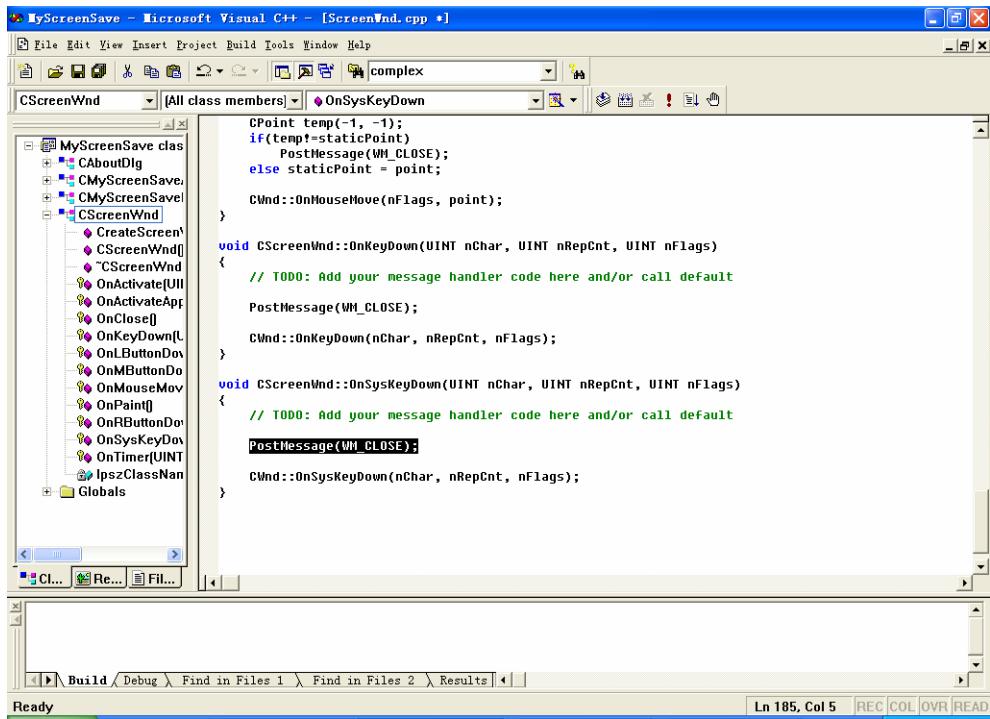


图 1-53

10、同 3 中添加私有成员变量的方式，在 CMYScreenApp 类中添加私有的 CScreenWnd*型成员变量 m_pScrWnd，如下图所示：

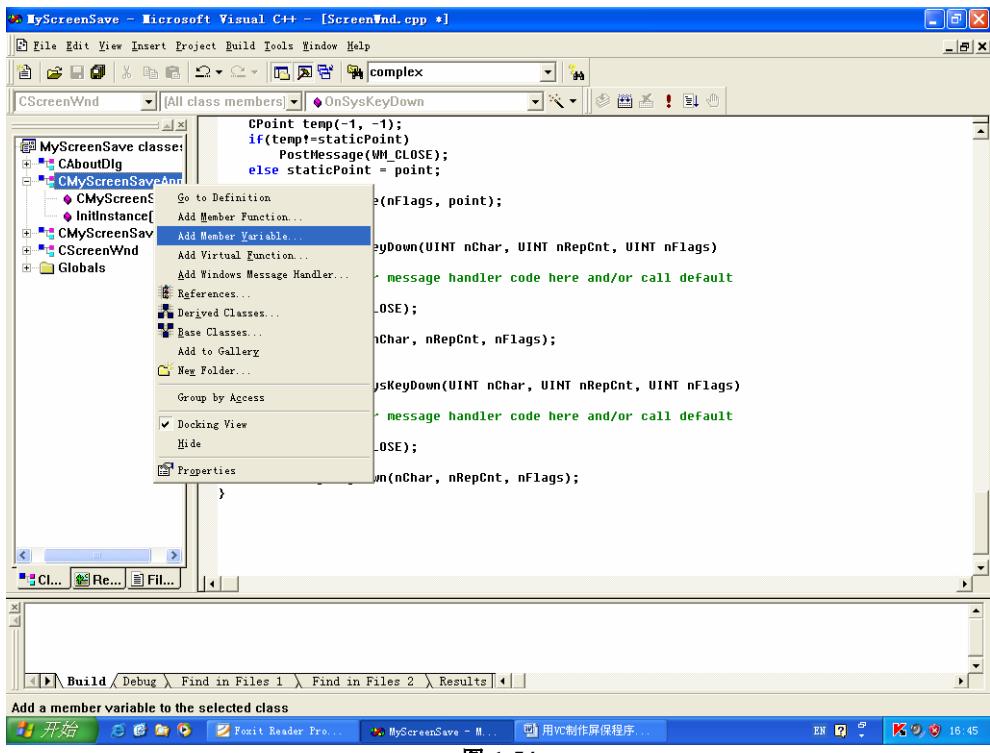


图 1-54

私有变量 m_pScrWnd, 类型 (Variable Type) : CScreenWnd*, 权限 Access: Private



图 1-55

点击 OK 按钮返回，如下图所示：

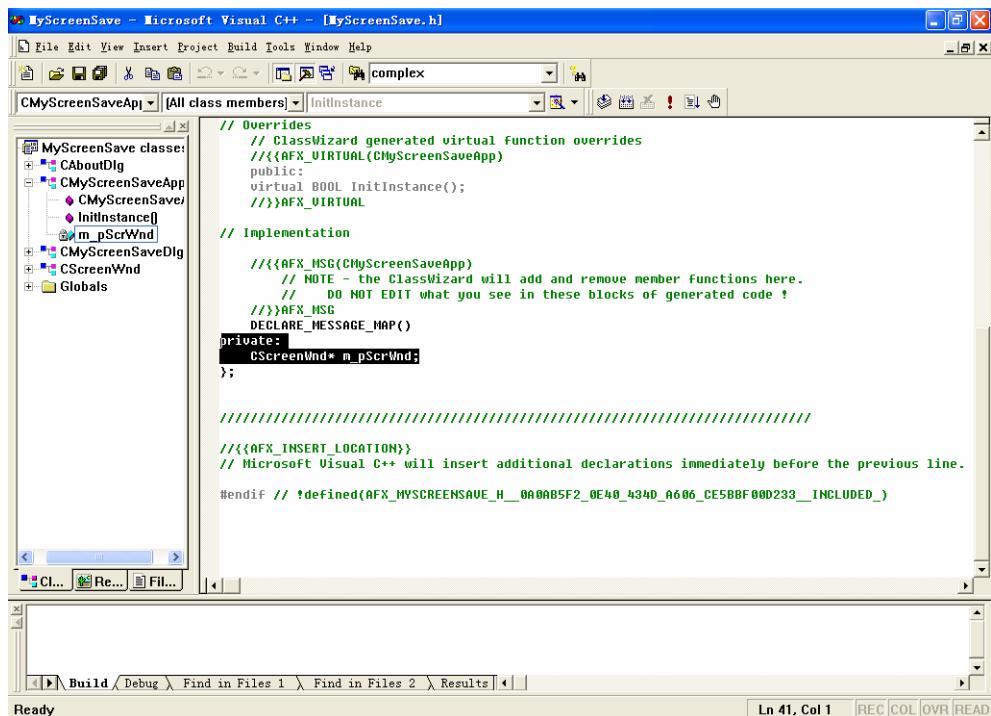


图 1-56

并将 CMyScreenApp 类中的 InitInstance 函数如下代码进行替换，如图所示：

```
// Standard initialization
// If you are not using these features and wish to reduce the size
// of your final executable, you should remove from the following
// the specific initialization routines you do not need.

#ifndef _AFXDLL
    Enable3dControls();           // Call this when using MFC in a shared DLL
#else
    Enable3dControlsStatic();     // Call this when linking to MFC statically
#endif

    CMyScreenSaveDlg dlg;
    m_mainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with Cancel
    }
}

// Since the dialog has been closed, return FALSE so that we exit the
// application, rather than start the application's message pump.
return FALSE;
}
```

图 1-57

替换成如下代码，如图所示：

```
////////////////////////////////////////////////////////////////////////
// CMyScreenSaveApp initialization

BOOL CMyScreenSaveApp::InitInstance()
{
    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

    CScreenWnd* pWnd = new CScreenWnd();
    m_mainWnd = pWnd;
    m_pScrWnd = pWnd;
    pWnd->CreateScreenWnd();

    return TRUE;
}
```

图 1-58

11、同 5 一样的方式增加 CScreenWnd 类的消息方式，为 CMyScreenApp 类添加函数 ExitInstance，（特别注意：Class name 应选择 CMyScreenApp 类，而不是 CScreenWnd 类，因为前面都是为 CScreenWnd 类添加消息的原因，这里打开后的默认初始是 CScreenWnd 类，切忌不要搞错！！！）图示如下：

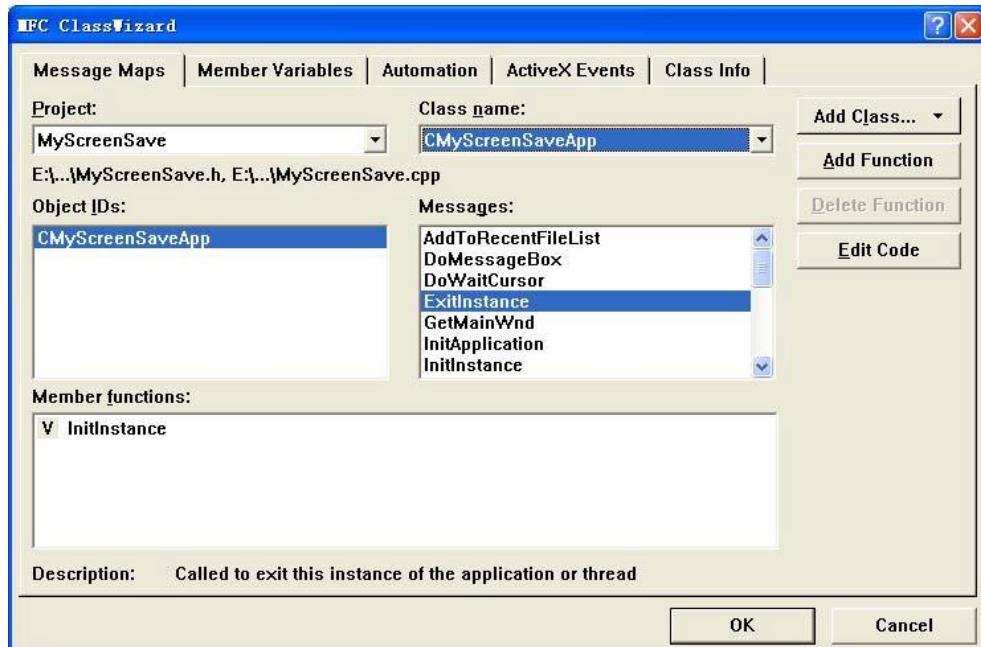


图 1-59

点击右侧的 Add Function 按钮，在 Member functions 框中，可以看到增添了一个函数 ExitInstance，在其中添加相应代码，如下图所示：

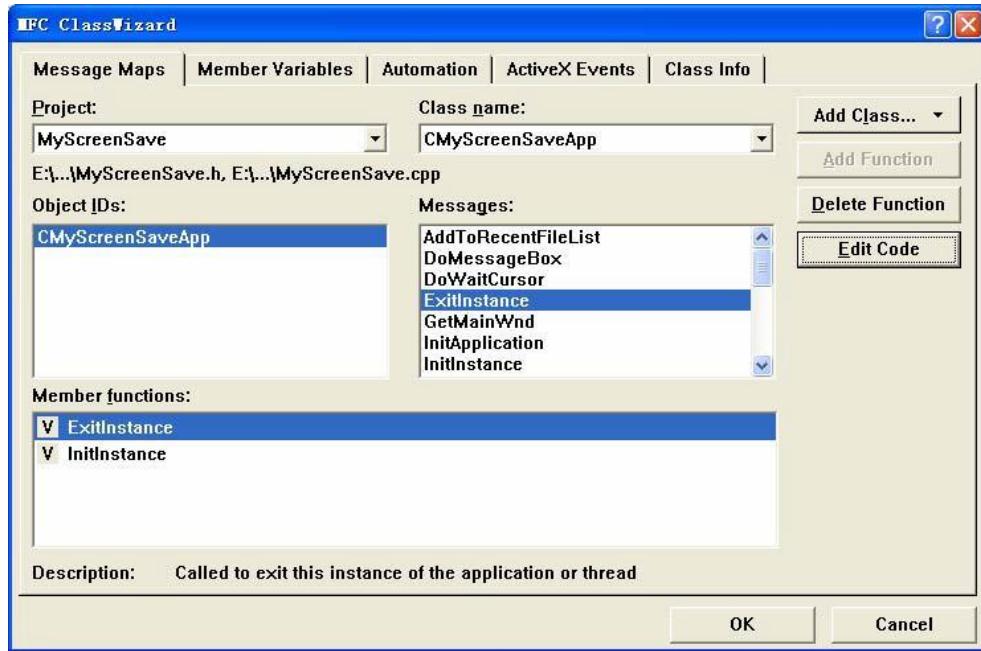
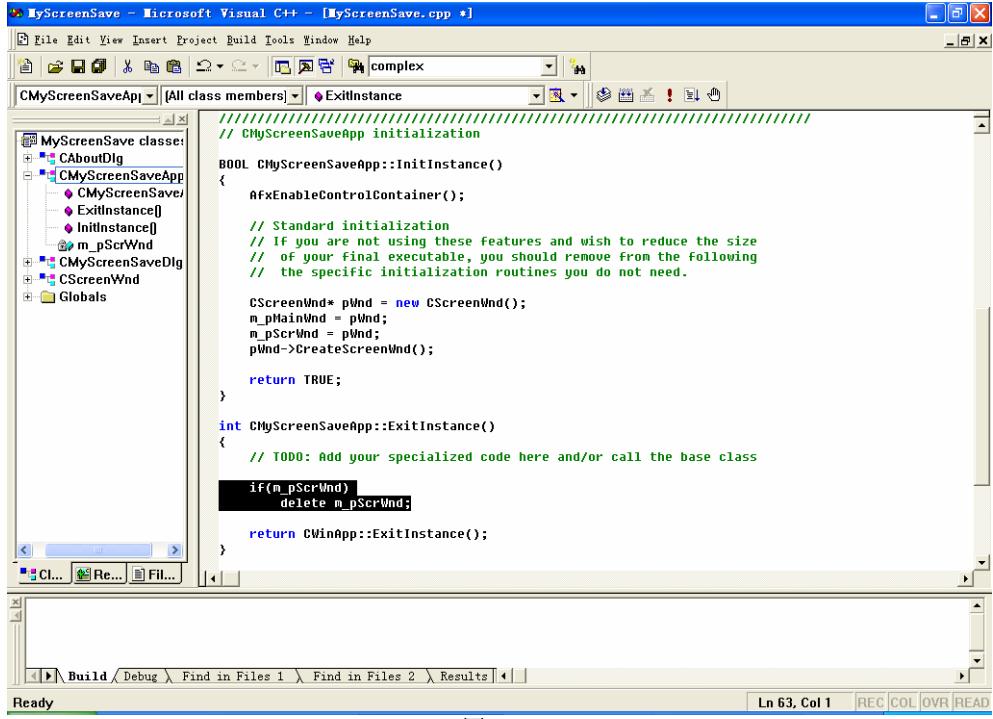


图 1-60

点击 Edit Code 按钮，添加相应代码：如下图所示：



```
MyScreenSave - Microsoft Visual C++ - [MyScreenSave.cpp]
File Edit View Insert Project Build Tools Window Help
complex
CMyScreenSaveApp
All class members | ExitInstance
CMyScreenSave classe:
+ CAboutDlg
+ CMyscreenSaveApp
  + CMyscreenSave
  + ExitInstance()
  + InitInstance()
  m_pScrWnd
+ CMyscreenSaveDlg
+ CScreenWnd
+ Globals
CMyscreenSaveApp initialization
BOOL CMyscreenSaveApp::InitInstance()
{
    AfxEnableControlContainer();
    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

    CScreenWnd* pWnd = new CScreenWnd();
    m_pMainWnd = pWnd;
    m_pScrWnd = pWnd;
    pWnd->CreateScreenWnd();

    return TRUE;
}

int CMyscreenSaveApp::ExitInstance()
{
    // TODO: Add your specialized code here and/or call the base class

    if(m_pScrWnd)
        delete m_pScrWnd;

    return CWinApp::ExitInstance();
}
```

图 1-61

二、代码编译调试

在完成上面代码的编写后，接着要对代码进行调试编译，首先转到 FileView 浏览模式，如下图 2-1 所示，编译文件的步骤如下：

- (1) 先 Compile 编译 stdafx.cpp，编译通过，如图 2-2；
- (2) 接着 Compile 编译 ScreenWnd.cpp，出现了三个错误，如图 2-3，第一个错误指向

CScreenWnd* m_pScrWnd;

语法没有任何问题，那么可能的原因是，CMyscreenSaveApp 类的头文件 MyScreenSave.h 中，类的定义部分用到了 CScreenWnd 类，而在 MyScreenSave.h 没有包含 CScreenWnd 类定义的头文件，在该文件的开头部分补上包括 ScreenWnd.h 头文件：

#include "ScreenWnd.h"

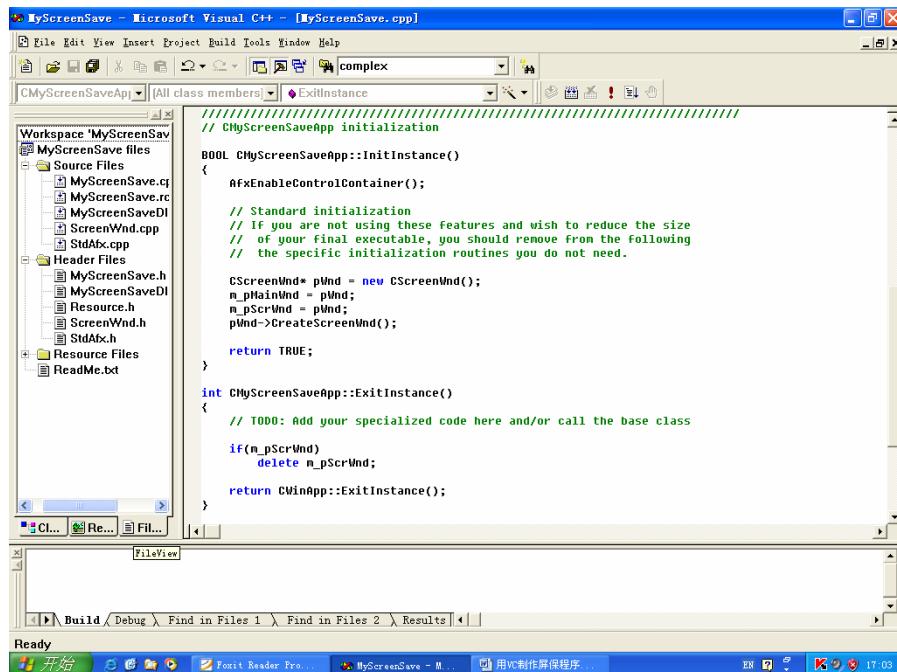
(3) 增加后的情况，如图 2-4 所示：

(4) 修改后，Compile 编译 ScreenWnd.cpp，通过 Compile 编译，如图 2-5 所示；

(5) 然后 Compile 编译 MyScreenSave.cpp，通过 Compile 编译，如图 2-6 所示；

(6) 接下来 Build 所有文件，如图 2-7 所示；

(7) 最后一步，Execute Program，如图 2-8 所示，回到一个 MyScreenSave.exe 的文件，点击运行就可以看到最后生成屏保的效果，如图 2-9 示。



```
// CMyScreenSaveApp initialization
BOOL CMyScreenSaveApp::InitInstance()
{
    AfxEnableControlContainer();

    // Standard initialization
    // If you are not using these features and wish to reduce the size
    // of your final executable, you should remove from the following
    // the specific initialization routines you do not need.

    CScreenWnd* pWnd = new CScreenWnd();
    m_pMainWnd = pWnd;
    m_pScrWnd = pWnd;
    pWnd->CreateScreenWnd();

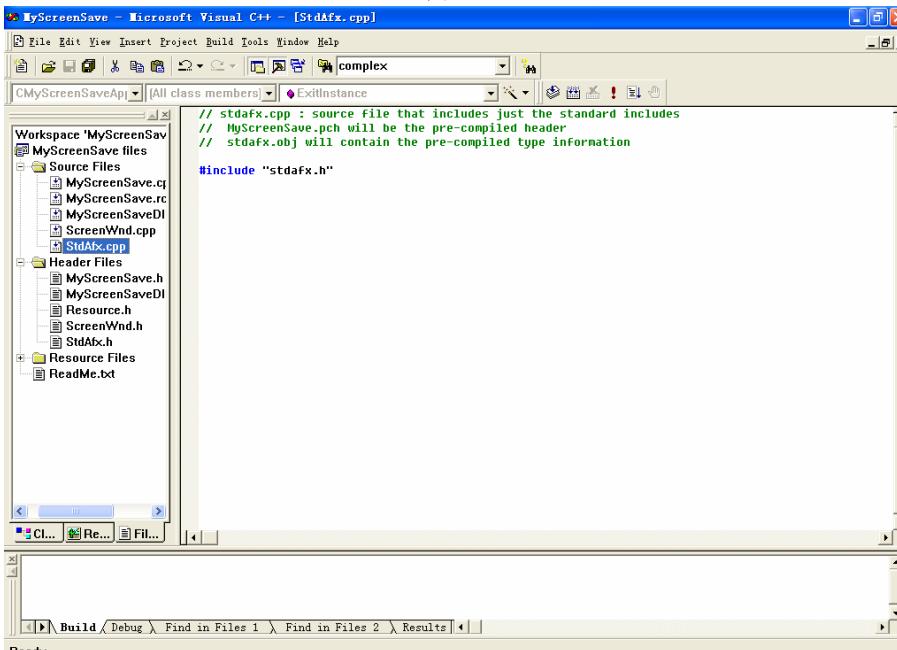
    return TRUE;
}

int CMyScreenSaveApp::ExitInstance()
{
    // TODO: Add your specialized code here and/or call the base class

    if(m_pScrWnd)
        delete m_pScrWnd;

    return CWinApp::ExitInstance();
}
```

图 2-1



```
// stdafx.cpp : source file that includes just the standard includes
// MyScreenSave.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

图 2-2

图 2-3

```

MyScreenSave - Microsoft Visual C++ - [CMyScreenSave.h]
File Edit View Insert Project Build Tools Window Help
complex
CMyScreenSaveApp [All class members] CMyScreenSaveApp
Workspace 'MyScreenSave'
Source Files
MyScreenSave.cpp
MyScreenSave.rc
MyScreenSaveDlg.cpp
StdAfx.cpp
Header Files
MyScreenSave.h
MyScreenSaveDlg.h
Resource.h
ScreenWnd.h
StdAfx.h
Resource Files
ReadMe.txt
External Dependencies

class CMyScreenSaveApp : public CWinApp
{
public:
    CMyScreenSaveApp();

    // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyScreenSaveApp)
    public:
        virtual BOOL InitInstance();
        virtual int ExitInstance();
    }}AFX_VIRTUAL

    // Implementation

    //{{AFX_MSG(CMyScreenSaveApp)
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
private:
    CScreenWnd* m_pScrWnd;
};

//{{AFX_INSERT_LOCATION}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

ScreenWnd.cpp
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2143: syntax error : missing ';' before '*'
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2501: 'CScreenWnd' : missing storage-class or type specifiers
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2501: 'm_pScrWnd' : missing storage-class or type specifiers
Error executing cl.exe
Build|Debug|Find in Files 1|Find in Files 2|Results|
```

图 2-3

图 2-4

```

MyScreenSave - Microsoft Visual C++ - [CMyScreenSave.h]
File Edit View Insert Project Build Tools Window Help
complex
CMyScreenSaveApp [All class members] CMyScreenSaveApp
Workspace 'MyScreenSave'
Source Files
MyScreenSave.cpp
MyScreenSave.rc
MyScreenSaveDlg.cpp
ScreenWnd.cpp
StdAfx.cpp
Header Files
MyScreenSave.h
MyScreenSaveDlg.h
Resource.h
ScreenWnd.h
StdAfx.h
Resource Files
ReadMe.txt
External Dependencies

// MyScreenSave.h : main header file for the MYSCREENSAVE application
//
#ifndef _AFXWIN_H_
#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"           // main symbols
#include "ScreenWnd.h"

// CMyScreenSaveApp:
// See MyScreenSave.cpp for the implementation of this class
//

class CMyScreenSaveApp : public CWinApp
{
public:
    CMyScreenSaveApp();

    // Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyScreenSaveApp)
    public:
        virtual BOOL InitInstance();
        virtual int ExitInstance();
    }}AFX_VIRTUAL

    // Implementation

    //{{AFX_MSG(CMyScreenSaveApp)
    // NOTE - the ClassWizard will add and remove member functions here.
    // DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
private:
    CScreenWnd* m_pScrWnd;
};

//{{AFX_INSERT_LOCATION}
// Microsoft Visual C++ will insert additional declarations immediately before the previous line.

ScreenWnd.cpp
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2143: syntax error : missing ';' before '*'
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2501: 'CScreenWnd' : missing storage-class or type specifiers
e:\cplus\project\myscreensave\myscreensave.h(43) : error C2501: 'm_pScrWnd' : missing storage-class or type specifiers
Error executing cl.exe
Build|Debug|Find in Files 1|Find in Files 2|Results|
```

图 2-4

MyScreenSave - Microsoft Visual C++ - [ScreenWnd.cpp]

```

// ScreenWnd.cpp : implementation file
//

#include "stdafx.h"
#include "MyScreenSave.h"
#include "ScreenWnd.h"

#ifndef _DEBUG
#define new DEBUG_NEW
#endif
static char THIS_FILE[] = __FILE__;
#endif

// CScreenWnd

CScreenWnd::CScreenWnd()
{
    lpszClassName = NULL;
}

CScreenWnd::~CScreenWnd()
{
}

BEGIN_MESSAGE_MAP(CScreenWnd, CWnd)
    //{{AFX_MSG_MAP(CScreenWnd)
    ON_WM_TIMER()
    ON_WM_PAINT()
    //}}AFX_MSG
END_MESSAGE_MAP()

ScreenWnd.obj - 0 error(s), 0 warning(s)

Build\Debug\Find in Files 1\Find in Files 2\Results\|_|

Ln 4, Col 20 REC COL OVR READ

```

图 2-5

MyScreenSave - Microsoft Visual C++ - [MyScreenSave.cpp]

```

// MyScreenSave.cpp : Defines the class behaviors for application.

//

#include "stdafx.h"
#include "MyScreenSave.h"
#include "MyScreenSaveDlg.h"

#ifndef _DEBUG
#define new DEBUG_NEW
#endif
static char THIS_FILE[] = __FILE__;
#endif

// CMyScreenSaveApp

BEGIN_MESSAGE_MAP(CMyScreenSaveApp, CWinApp)
    //{{AFX_MSG_MAP(CMyScreenSaveApp)
    // NOTE: the ClassWizard will add and remove mapping macros here.
    // DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

// CMyScreenSaveApp construction

CMyScreenSaveApp::CMyScreenSaveApp()
{
    // TODO: add construction code here,
}

MyScreenSave.obj - 0 error(s), 0 warning(s)

Build\Debug\Find in Files 1\Find in Files 2\Results\|_|

Builds the project Ln 1, Col 4 REC COL OVR READ

```

图 2-6

The screenshot shows the Microsoft Visual Studio IDE interface. The title bar reads "MyScreenSave - Microsoft Visual C++ - [MyScreenSave.cpp]". The menu bar includes File, Edit, View, Insert, Project, Build, Tools, Window, Help. The toolbar has standard icons for file operations. The status bar at the bottom says "Ready".

The code editor displays the source file `MyScreenSave.cpp`. The code defines a class `CMyScreenSaveApp` that inherits from `CWinApp`. It includes headers for `stdafx.h`, `MyScreenSave.h`, and `MyScreenSaveDlg.h`. The code uses preprocessor directives to define symbols and map message handlers. A note in the code states: "NOTE - the ClassWizard will add and remove mapping macros here. DO NOT EDIT what you see in these blocks of generated code!"

```
// MyScreenSave.cpp : Defines the class behaviors for the application.
//
#include "stdafx.h"
#include "MyScreenSave.h"
#include "MyScreenSaveDlg.h"

#define _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;

/////////////////////////////////////////////////////////////////////////////
// CMyScreenSaveApp construction

CMyScreenSaveApp::CMyScreenSaveApp()
{
    // TODO: add construction code here,
}

BEGIN_MESSAGE_MAP(CMyScreenSaveApp, CWinApp)
    //{{AFX_MSG_MAP(CMyScreenSaveApp)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        // DO NOT EDIT what you see in these blocks of generated code!
    }}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMyScreenSaveApp message handlers
```

The status bar at the bottom of the code editor window shows "MyScreenSave.exe - 0 error(s), 0 warning(s)".

图 2-7

This screenshot is identical to Figure 2-7, showing the Microsoft Visual Studio IDE with the `MyScreenSave.cpp` file open. A tooltip appears over the F5 key on the keyboard, which is labeled "Execute Program (Ctrl+F5)".

The code editor and status bar are the same as in Figure 2-7.

图 2-8

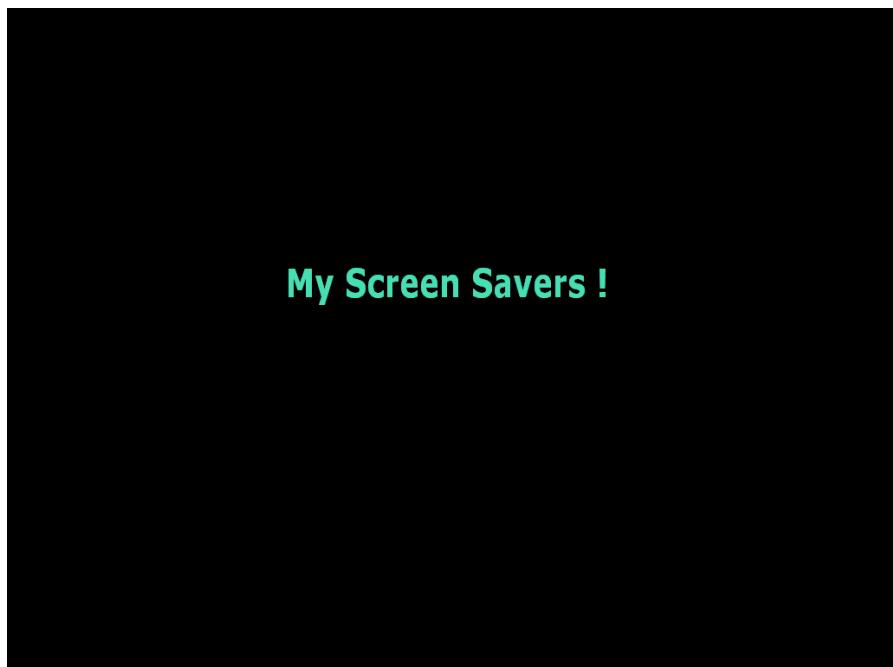


图 2

三、制作你自己的屏保

当生成了可以执行的 MyScreenSave.exe 的文件后，要让 Windows 系统能够调用你的屏保，还必须生成 MyScreenSave.scr 文件，即参考文献的 12 部分，选定整个工程（不然没有你需要的设定菜单），如图 3-1 所示，然后设定输出的格式，如图 3-2 和 3-3 所示，如果第一次设定会出现 3-2 的情况，这个时候要把 Category 选择为 Output，然后将 MyScreenSave.exe 改名为 MyScreenSave.scr，因为 Windows 的屏保格式为*.scr 形式，改后的情况如图 3-4 所示。

通过上面的方式，你就可以得到了能够然 Windows 系统运行屏保了，加载你的屏保，把生成 MyScreenSave.scr 如图 3-5，拷贝到系统的 system32 目录下面，如图 3-6 所示，然后在桌面的屏幕保护设置中，就可以看到刚才的屏保，如图 3-7 所示。

上面只是实现了我们给定参考文献上的屏保，如何生成自己的漂亮屏保了，你只需要更改 CScreenWnd 类的 OnPaint() 函数的相关部分，如图所示 3-8，就可以制作自己的屏保，你可以生成除了文字以外的其他图案，例如图形等等，参照附带的文件 BGIDEMO，这个是 TC 里面的演示 Demo，里面有很多漂亮的图形的生成代码。

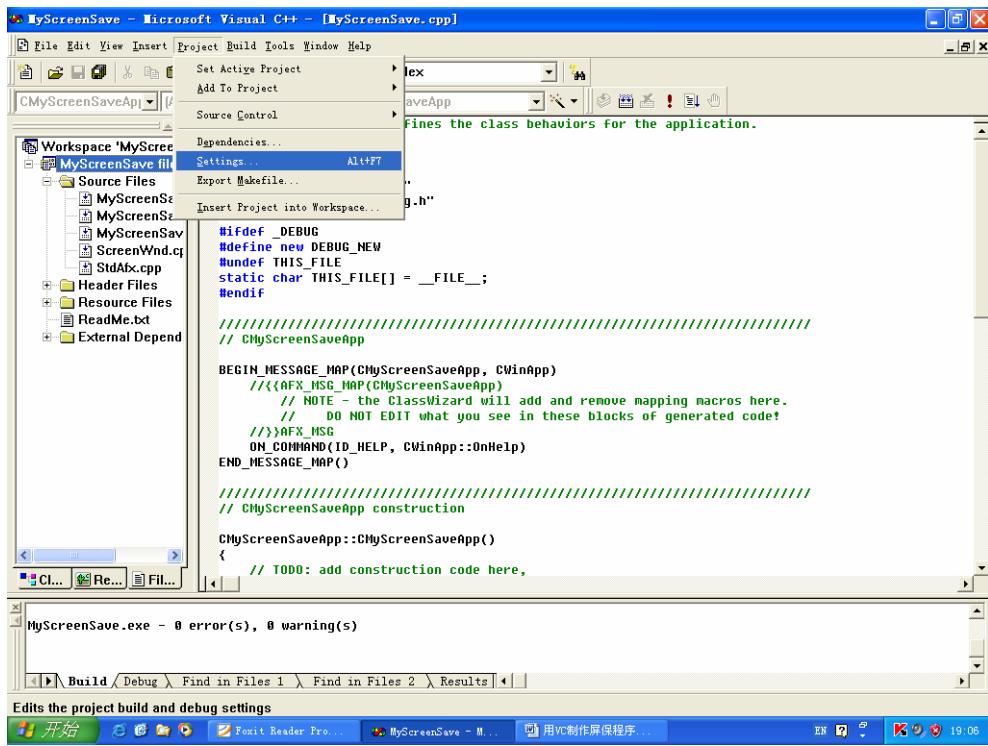


图 3-1

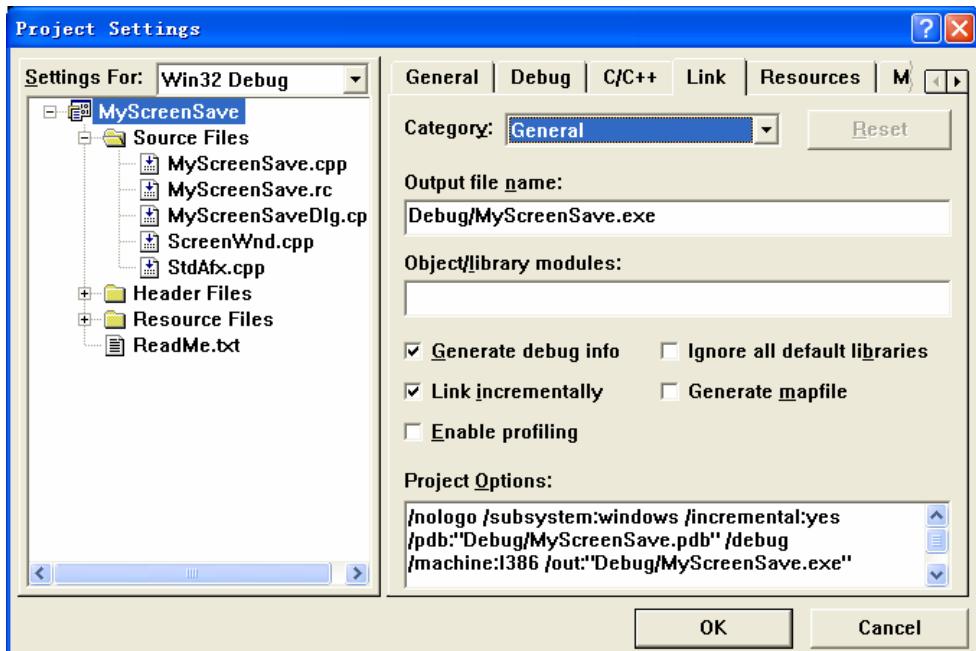


图 3-2

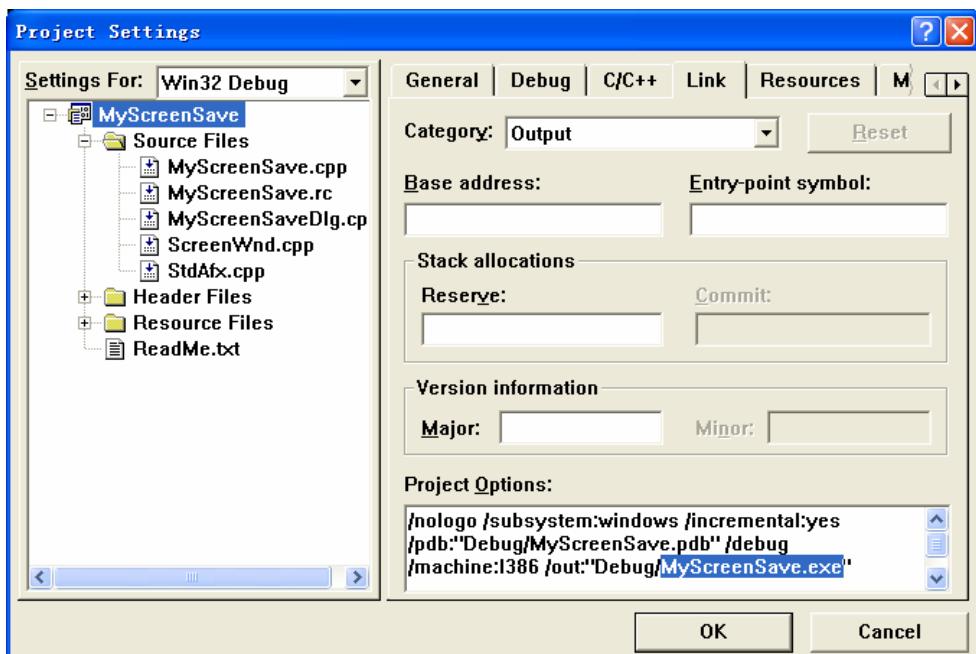


图 3-3

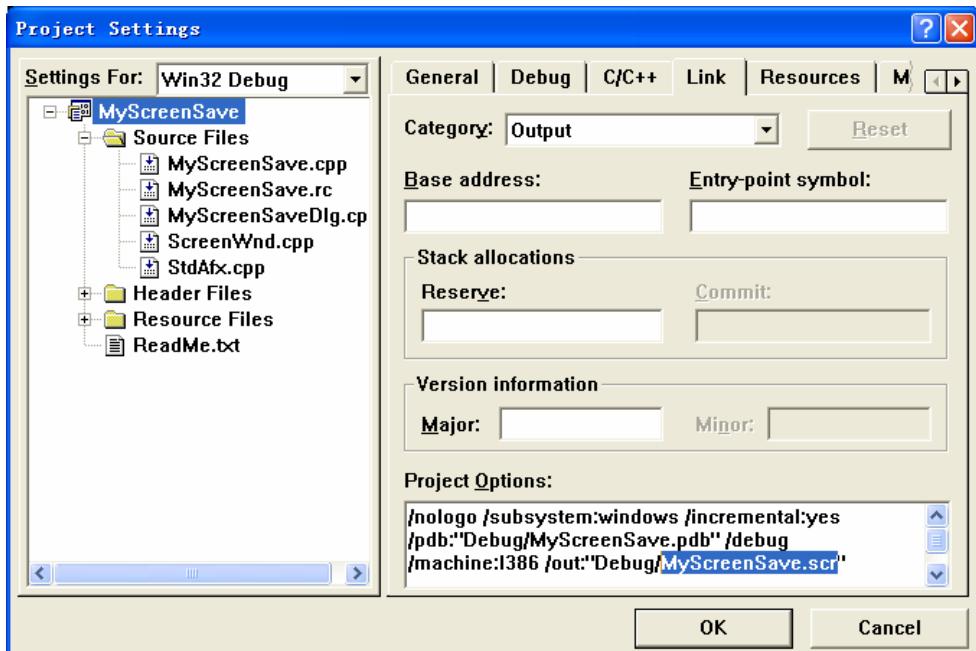


图 3-4

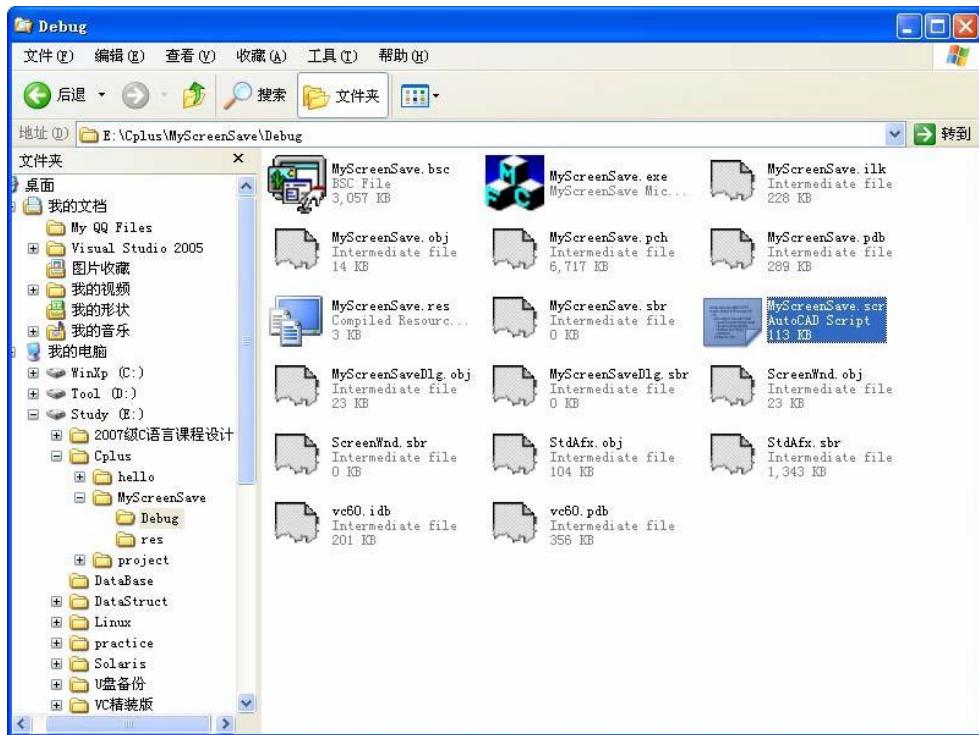


图 3-5

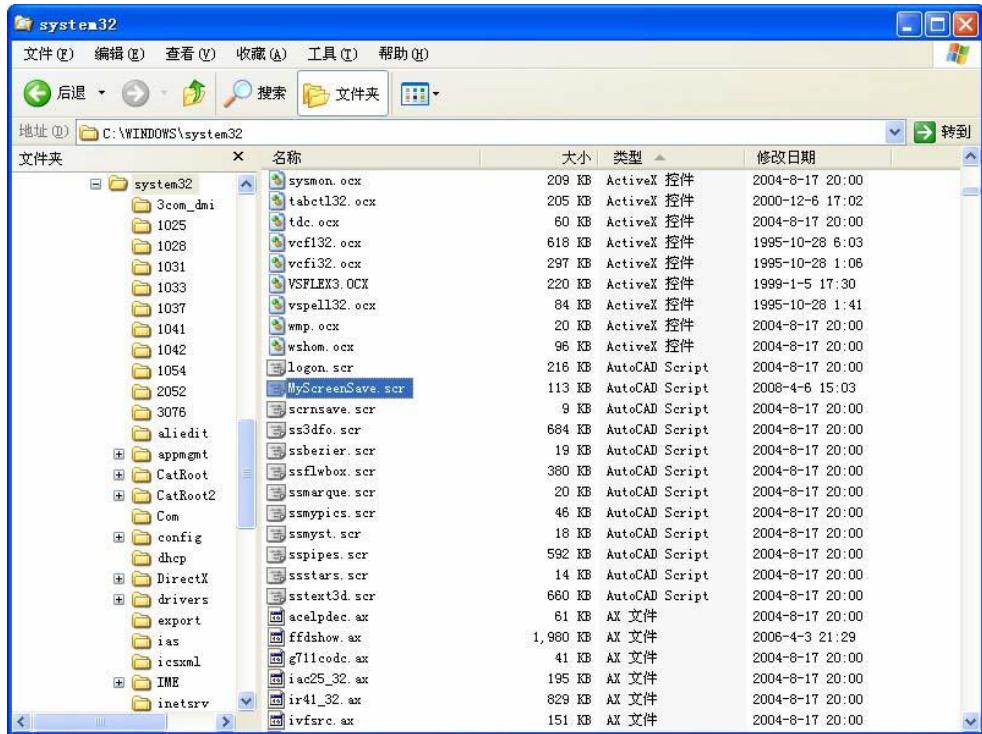


图 3-6



图 3-7

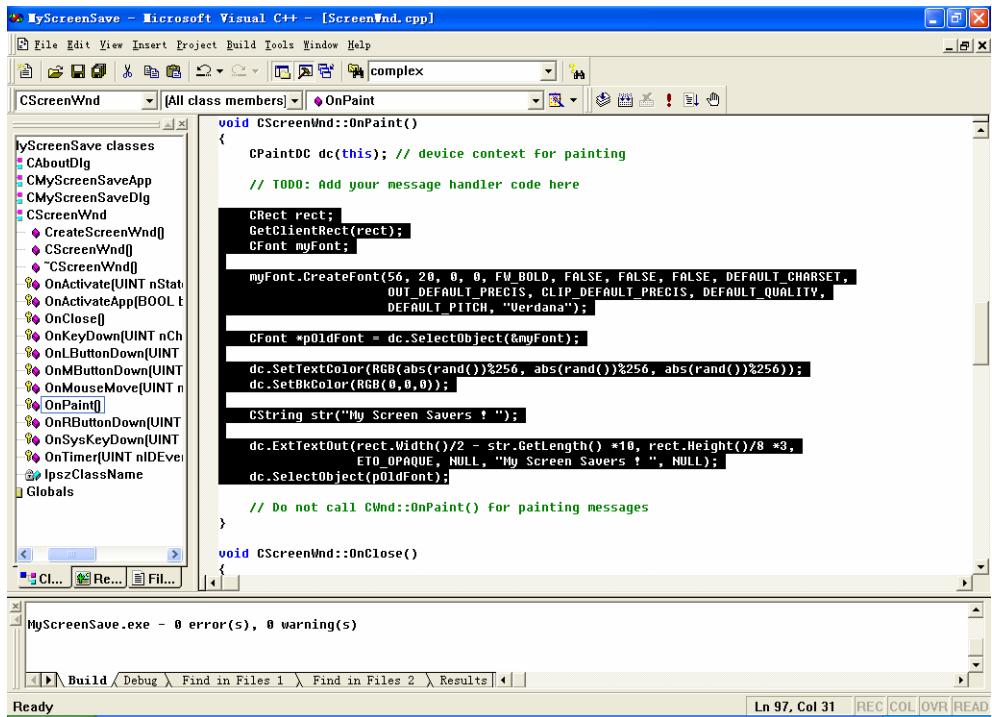


图 3-8

参考文献：

- [1] 王卫动.《用 VC 制作屏保程序》，编程天地，<http://www.cnki.cn>
- [2] 石祥生,石秋云等译.《Visual C++ 6 使用指南》，北京：电子工业出版社，2000，4.

附录 程序编写风格规范参考

下面的程序，主要实现比较三种函数调用方式对两个数字的交换情况，通过程序运行 结果，可以清晰的看到，传值调用，指针做参数调用，传引用调用对两个数交换能否实现 的情况。

常用注释的两种注释方法：

“//” 只是注释本行；

“/* */” 注释从 “/*” 开始到 “*/” 结尾。 程序的注释，尽量使用简单英文进行描述，主要分为三个部分：

(1) 工程的主题介绍部分，程序的开头部分，主要描述了，改工程的名称以及实现什么功能，程序代码作者，联系方式，最后修改时间等；

(2) 每个函数主题的介绍，主要包括，函数实现的功能，前备条件（Precondition）和后备条件（Postcondition），以及最后输出或者要得到什么结果；

(3) 具体某个语句的说明，某些语句比较重要或者不易理解，进行适当注释增强了程序的可读性。

两种注释的使用：

在程序开头和函数注释部分，建议使用“`/* */`”，因为基本调整较少，不用使用太多的“`//`”，能够正段落的注释；而在具体某一句语句的注释时，可以使用“`//`”，方便随时修改，每一次只是注释一行。

在编写程序代码的时候，良好的编程风格有利于程序代码的编写和维护，提高程序的可读性，在开始编写代码的时候，养成良好的编程风格，在以后的程序员生涯中，一定会受益匪浅！

```
*****  
/*      Function Parameter project      */  
*****  
/* Task: Compare three swaping function */  
*****  
/* TeacherID: 0001292      */  
/* Author: Cole xu        */  
/* Email: hustxuke@163.com */  
/* Datatime: 28 Feb,2008   */  
*****  
  
#include <iostream>  
using namespace std;  
  
void mymenu(); // print the prompts  
void swap_by_value(int first, int second); // swap two number using Call-by-Value  
void swap_by_pointer(int * first, int * second); // swap two number using Call-by-Pointer  
void swap_by_reference(int& first, int& second); // swap two number using Call-by-Reference  
  
void main()  
{  
    int front, back; // the two number to swap  
    char choose, ans; // choose for choosing, ans for yes or no  
  
    do  
    {  
        // input two number to swap  
        cout << "Please Input two number!" << endl;  
        cout << "the front number: ";  
        cin >> front;  
        cout << "the back number: ";  
        cin >> back;  
        cout << "the swap result is: ";  
        cout << swap_by_value(front, back);  
    } while (choose != 'n');  
}
```

批注 [hust1]: 程序实现什么功能，作者信息，修改时间

批注 [hust2]: 函数声明，描述
函数实现什么功能

```

cout<<"the back number:";  

cin>>back;

mymenu();  

cout<<"Please enter the choose: "; //input using which function to swap  

cin>>choose;

switch(choose)  

{  
  

    case '1':           // calling swap_by_value function  

        cout<<"The two number before calling swap_by_value as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        swap_by_value(front,back);  

        cout<<"The two number after calling swap_by_value as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        break;  
  

    case '2':           // calling swap_by_pointer function  

        cout<<"The two number before calling swap_by_pointer as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        swap_by_pointer(&front,&back);  

        cout<<"The two number after calling swap_by_pointer as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        break;  
  

    case '3':           // calling swap_by_reference function  

        cout<<"The two number before calling swap_by_reference as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        swap_by_reference(front,back);  

        cout<<"The two number after calling swap_by_reference as follow !\n"  

        <<"the front number: "<<front<<" and its address is "<<&front  

        <<"\nthe back number: "<<back<<" and its address is  

        "=<&back<<endl;  

        break;
}

```

批注 [hust3]: 具体一程序语句说明或者提示

```

        }
        // asking for do once more
        cout<<"Do you want do once more ?\n"
        <<"Press y for yes, n for no, \n"
        <<"and then press return: ";
        cin>>ans;
    } while(ans=='y' || ans=='Y');

}

```

/* print the prompts menu, no parameter to input */

```
void mymenu()
```

```
{
```

cout<<"Please choose which means to swap !\n"

```

        <<*****|n"
        <<"*** 1: Call-by-value ***|n"
        <<"*** 2: Call-by-pointer ***|n"
        <<"*** 3: Call-by-reference ***|n"
        <<*****|n";
}

```

/*

swap two number calling by value

Precondition: two number taken for keyboard.

Postcondition: two number calling function by to show.

field: average of scores.

***/**

```
void swap_by_value(int first, int second)
```

```
{
```

```
    cout<<"*** Begin Calling swap_by_value ***|n";
```

cout<<"The two form parameter and those address before swap as follow !\n"

```

        <<"the first number: "<<first<<" and its address is "<<&first
        <<"\nthe second number: "<<second<<" and its address is "<<&second<<endl;
```

int temp;

temp=first;

first=second;

second=temp;

cout<<"The two form parameter and those address after swap as follow !\n"

```

        <<"the first number: "<<first<<" and its address is "<<&first
        <<"\nthe second number: "<<second<<" and its address is "<<&second<<endl;
```

批注 [hust4]: 菜单打印提示 函数，以后会经常用到

```

    cout<<"*** End Calling swap_by_value ***\n";
}

/*
swap two number calling by pointer
Precondition: two number taken for keyboard.

Postcondition: two number calling function by to show.

field: average of scores.
*/
void swap_by_pointer(int * first, int * second)
{
    cout<<"*** Begin Calling swap_by_pointer ***\n";

    cout<<"The two form parameter and those address before swap as follow !\n"
        <<"the *first number: "<<*first<<" and its address is "<<first
        <<"\nthe *second number: "<<*second<<" and its address is "<<second<<endl;

    int temp;
    temp=*first;
    *first=*second;
    *second=temp;

    cout<<"The two form parameter and those address after swap as follow !\n"
        <<"the *first number: "<<*first<<" and its address is "<<first
        <<"\nthe *second number: "<<*second<<" and its address is "<<second<<endl;

    cout<<"*** End Calling swap_by_pointer ***\n";
}

```

```

/*
swap two number calling by reference
Precondition: two number taken for keyboard.

Postcondition: two number calling function by to show.

field: average of scores.
*/
void swap_by_reference(int& first, int& second)
{
    cout<<"*** Begin Calling swap_by_reference ***\n";

```

```

    cout<<"The two form parameter and those address before swap as follow !\n"
        <<"the first number: "<<first<<" and its address is "<<&first
        <<"\nthe second number: "<<second<<" and its address is "<<&second<<endl;

```

批注 [hust5]: 函数定义时候的注释说明，包括前备条件和后备条件，输出或者换回结果

```
int temp;
temp=first;
first=second;
second=temp;
cout<<"The two form parameter and those address after swap as follow !\n"
     <<"the first number: "<<first<<" and its address is "<<&first
     <<"\nthe second number: "<<second<<" and its address is "<<&second<<endl;

cout<<"* * * End Calling swap_by_reference * * *\n";
}
```

参考文献

- [1] (美) Walter Savitch 著. 周靖 译. 《C++面向对象程序设计》, 清华大学出版社, 2007
- [2] 王卫动. 《用 VC 制作屏保程序》, 编程天地, <http://www.cnki.cn>
- [3] 石祥生, 石秋云等译. 《Visual C++ 6 使用指南》, 北京: 电子工业出版社, 2000, 4