CHAPTER

第5章 MFC的通用类习题解答及上机实验

5.1 习题解答

5-1 解释下列语句的含义。

(1) CString s;

(2) CString s("Hello, Visual C++ 6.0");

(3) CString s('A',100);

(4) CString s(buffer,100);

(5) CString s(anotherCString)。

答:

(1) 构造一个长度为0的字符串对象。

(2) 构造一个名称为s的字符串对象,并把字符串初始化为Hello, Visual C++ 6.0。

(3) 构造一个名称为s的字符串对象,s字符串的内容是100个A。

(4) 构造一个名称为s的字符串对象,s字符串的内容是buffer的头100个字符,再加一个NULL。

(5) 构造一个名称为s的字符串对象,s字符串的内容和anotherCString字符串的内容相同。

5-2 执行

```
CString s(CString("Hello, world").Left(6)
      +CString("Visual C++").Right(3));
```

语句后,s字符串中的内容是什么?

答: Hello, C++。

5-3 现有语句CString s("My,name,is,C++");若想将s字符串中的“,”号全部更换成“ ”,将如何编写语句?

答：

```
s.Replace(',',' ');
pDC->TextOut(1,1,s);
```

5-4 CString 创建时只分配 128B 的缓冲区，如何分配更大的缓冲区？

答：使用 GetBuffer()函数。

例如：

```
CString s;
s.GetBuffer(1024);
```

5-5 编写一个满足下面要求的单文档界面应用程序。

(1) 单击左键显示“您已经单击左键了”。

(2) 单击右键显示“您已经单击右键了”。

答：

(1) 为添加的消息响应函数编写如下代码。

```
CString str1="您已经单击左键了";
AfxMessageBox(str1,MB_OK|MB_ICONINFORMATION);
```

(2) 用同样的方法添加右键消息响应函数，代码如下：

```
CString str1="您已经单击右键了";
AfxMessageBox(str1,MB_OK|MB_ICONINFORMATION);
```

5-6 编写一个单文档界面应用程序，该程序可以测试在鼠标左键按下时鼠标光标的位置是否处在某规定的矩形框内，如果不在该矩形内则计算机的扬声器会发出“叮”的声音，反之则会在用户区显示光标的位置。

答：

(1) 用 MFC AppWizard 创建一个名称为 MusInRec 的单文档应用程序框架。

(2) 在视图类的声明中定义一个 CRect 类的对象来描述矩形，再定义一个 POINT 结构来存储鼠标在按下时的位置，即在视图类的声明中添加如下代码。

```
public:
    POINT m_point;
    CRect m_rRct;
```

(3) 在视图类的构造函数中初始化数据成员：

```
CMusInRecView::CMusInRecView():m_rRct(50,50,250,200)
{
    m_point.x=0;m_point.y=0;
}
```

(4) 在视图类的 OnDraw 函数中写入如下代码。

```
void CMusInRecView::OnDraw(CDC * pDC)
```

```
{
    CMusInRecDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    char s[20];
    wsprintf(s, "X=%d Y=%d ", m_point.x, m_point.y);
    pDC->TextOut(5,5,s);
}
```

(5) 在视图类的鼠标左键按下消息响应函数 OnLButtonDown 中写入如下代码。

```
void CMusInRecView::OnLButtonDown(UINT nFlags, CPoint point)
{
    if(m_rRct.PtInRect(point))
    {
        m_point.x=point.x;
        m_point.y=point.y;
    }
    else
    {
        MessageBeep(0);
    }
    InvalidateRect(NULL);
    CView::OnLButtonDown(nFlags, point);
}
```

说明：本题用到了一个 wsprintf 函数，它的作用是把数字转换为字符，该函数的原型为：

```
int wsprintf(
  LPTSTR lpOut,              //字符串缓冲区的指针
  LPCTSTR lpFmt,             //格式字符串的指针
  ⋮                          //需要转换的参数
);
```

5-7　编写一个单文档界面应用程序，该程序在用户区能以在两个矩形的相交矩形为外接矩形画一个椭圆。

答：

(1) 用 MFC AppWizard 创建一个名称为 RecRec 的单文档应用程序框架。

(2) 在视图类的声明中声明两个描述矩形的成员变量：

```
CRect m_rRec1;
    CRect m_rRec2;
```

(3) 在视图类的构造函数初始化数据成员：

```
CRecRecView::CRecRecView()
```

```
        :m_rRec1(50,50,250,200),m_rRec2(100,120,300,400)
{
}
```

(4) 在视图类的 OnDraw 函数中写入如下代码。

```
void CRecRecView::OnDraw(CDC * pDC)
{
    CRecRecDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    int x1,y1;
    int x2,y2;
    //pDC->Rectangle(m_rRec1);
    //pDC->Rectangle(m_rRec2);
    if(m_rRec1.left<m_rRec2.left)
        x1=m_rRec2.left;
    else
        x1=m_rRec1.left;
    if(m_rRec1.top<m_rRec2.top)
        y1=m_rRec2.top;
    else
        y1=m_rRec1.top;
    if(m_rRec1.right<m_rRec2.right)
        x2=m_rRec1.right;
    else
        x2=m_rRec2.right;
    if(m_rRec1.bottom<m_rRec2.bottom)
        y2=m_rRec1.bottom;
    else
        y2=m_rRec2.bottom;
    pDC->Ellipse(x1,y1,x2,y2);
}
```

说明：该程序还有一些缺陷，它对两个矩形不相交的情况没有进行处理，请读者将应该有的代码补全。

5.2 上机实验

5.2.1 简单通用类的应用

实验内容：

通用类的应用。

实验目的：

(1) 通过对 CRect 类对象的应用，了解该类的成员函数。

(2) 通过对 CPoint 类对象的应用,了解该类的成员函数。

实验步骤:

(1) 用 MFC AppWizard 创建一个名称为 Cls 的单文档界面应用程序框架。

(2) 在视图类的声明中声明成员变量。

```
public:
    CPoint m_point,m_ofpoint;
    CRect m_rRct1,m_rRct2;
```

(3) 在视图类的构造函数中对成员变量进行初始化。

```
CClsView::CClsView()
            :m_rRct1(10,10,40,20),m_rRct2(10,30,40,40)
{
    // TODO: add construction code here
    m_point.x=10;
    m_point.y=0;
    m_ofpoint.x=0;
    m_ofpoint.y=10;
}
```

(4) 在视图类的鼠标按下消息响应函数中书写如下代码。

```
void CClsView::OnLButtonDown(UINT nFlags, CPoint point)
{
    if(m_rRct1.right<100)
    {
        m_rRct1.OffsetRect(10,0);
        m_rRct2.OffsetRect(m_point);
    }
    else
    {
        m_rRct1.OffsetRect(10,10);
        m_rRct2.OffsetRect(m_point+m_ofpoint);
    }
    InvalidateRect(NULL);
    CView::OnLButtonDown(nFlags, point);
}
```

(5) 在视图类的 OnDraw 函数中书写如下代码。

```
void CClsView::OnDraw(CDC * pDC)
{
    CClsDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    pDC->Rectangle(m_rRct1);
```

```
    pDC->Rectangle(m_rRct2);
}
```

(6) 按 Ctrl+F7 键编译并运行程序。

(7) 查阅 Visual C ++ 的帮助文件,了解 CPoint 和 CRect 类的成员函数,并在上面程序中用合适的方式运用这些成员函数。

5.2.2 群体类的应用

实验内容:

群体类 CArray 的应用。

实验目的:

通过对 CRect 类对象的应用,了解该类使用方法。

实验步骤:

(1) 用 MFC AppWzard 创建一个名称为 DrawLine 的单文档界面应用程序框架。

(2) 在应用程序的头文件 StdAfx.h 中包含头文件:

```
#include <afxtempl.h>
```

(3) 在工程中声明一个类:

```
class Element
{
public:
    CPoint m_pPoint;
    CSize m_scSize;
    Element();
    Element(CPoint&point);
    virtual ~ Element();

};
```

(4) 在类的构造函数中初始化成员。

```
Element::Element(CPoint&point):m_scSize(6,6)
{
    m_pPoint=point;
}
```

(5) 在视图类的声明文件中包含头文件。

```
#include"Element.h"
```

(6) 在视图类的声明中定义两个成员。

```
public:
    BOOL m_bDown;
    CArray<Element,Element&>m_aDraw;
```

(7) 在视图类的鼠标按下消息响应函数 OnLButtonDown 中写入如下代码。

```
void CDrawLineView::OnLButtonDown(UINT nFlags, CPoint point)
{
    Element e(point);
    m_aDraw.Add(e);
    m_bDown=TRUE;
    InvalidateRect(NULL);
    CView::OnLButtonDown(nFlags, point);
}
```

(8) 在视图类的 OnDraw 函数中写入如下代码。

```
void CDrawLineView::OnDraw(CDC * pDC)
{
    CDrawLineDoc * pDoc=GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    if(m_bDown)
    for(int i=0;i<m_aDraw.GetSize();++i)
    {
        if(i!=0)
        {
            pDC->MoveTo(m_aDraw[i-1].m_pPoint.x,
                        m_aDraw[i-1].m_pPoint.y);
            pDC->LineTo(m_aDraw[i].m_pPoint.x,
                        m_aDraw[i].m_pPoint.y);
        }
        pDC->Rectangle(
    m_aDraw[i].m_pPoint.x-m_aDraw[i].m_scSize.cx/2,
    m_aDraw[i].m_pPoint.y-m_aDraw[i].m_scSize.cy/2,
    m_aDraw[i].m_pPoint.x+m_aDraw[i].m_scSize.cx/2,
    m_aDraw[i].m_pPoint.y+m_aDraw[i].m_scSize.cy/2);
    }
}
```

(9) 输入代码无误后，按 Ctrl＋F5 键编译并运行程序，然后在用户区按下鼠标左键观察程序运行结果。

(10) 结合程序代码和程序运行结果，分析这个应用程序，体会群体类对象的使用。

(11) 把上面实验程序视图类中的数据定义都改在文档类，试编写程序使之仍然可以实现原程序的功能。