

PART 1 部分

长安大学

面向对象程序设计实验指导书

任课教师：胡笑钊

2018 年 3 月

目 录

实验 1	常用控件（1）	3
实验 2	常用控件（2）	7
实验 3	列表控件和树控件.....	12
实验 4	菜单、工具栏和状态栏.....	17
实验 5	框架窗口和文档.....	21
实验 6	切分窗口.....	31
实验 7	图形、文本和打印.....	38

实验 1 常用控件（1）

实验目的和要求

- (1) 创建一个默认的对话框应用程序 Ex_Ctrls，如图 2.1 所示。
- (2) 设计一个如图 2.2 所示的“课程信息”对话框。

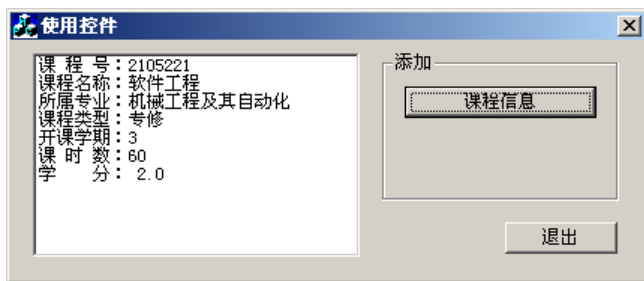


图 2.1 Ex_Ctrls 对话框

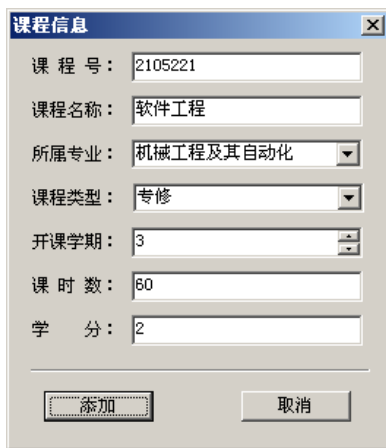


图 2.2 “课程信息”对话框

(3) 实现功能：单击图 2.1 中的“课程信息”按钮，弹出“学生基本信息”对话框，单击“添加”按钮，对话框内容显示在图 2.1 中的列表框中。

实验准备和说明

- (1) 具备知识：静态控件、按钮、编辑框、旋转按钮控件、列表框和组合框控件。
- (2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验 2”。

实验内容和步骤


1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 创建一个默认的对话框应用程序 Ex_Ctrls

① 选择“文件”→“新建”菜单，在弹出的“新建”对话框中选择“工程”页面，选择 MFC AppWizard (exe)，在工程框中输入 Ex_Ctrls，并将工程文件夹定位到“...\Visual C++程序\实验\实验 2”。

② 单击“确定”按钮，在出现的 Step 1 对话框中选择“基本对话框”应用程序类型，单击“完成”按钮。

③ 在对话框编辑器中，单击对话框工具栏上的切换网格按钮 ，显示对话框网格，将对话框标题改为“使用控件”。

④ 调整对话框的大小, 删除对话框中间的“TODO: 在这里设置对话控制。”静态文本控件和“确定”按钮控件, 将“取消”按钮标题改为“退出”, 并移至对话框的下方。

⑤ 向对话框中添加组框(Group)控件, 标题设为“添加”, 然后调整其大小和位置。

⑥ 添加一个按钮, 标题设为“课程信息”, ID 设为 IDC_BUTTON_COURSE。

⑦ 添加一个列表框, 取其默认 ID 号, 去掉 Sort 风格属性。

3. 添加并设计“课程信息”对话框

① 按 Ctrl+R 快捷键, 弹出“插入资源”对话框, 在资源类型列表中选择 Dialog, 单击“新建”按钮。

② 将该对话框资源的 ID 设为 IDD_COURSE, 标题设为“课程信息”, 字体设为“宋体, 9 号”。

③ 将 OK 和 Cancel 按钮的标题改为“添加”和“取消”。

④ 打开对话框网格, 参看图 2.2 的控件布局, 为对话框添加如表 2.1 所示的一些控件。

表 2.1 课程信息对话框添加的控件

添加的控件	ID 号	标题	其他属性
编辑框(课程号)	IDC_EDIT_COURSENO	——	默认
编辑框(课程名称)	IDC_EDIT_COURSENAME	——	默认
组合框(所属专业)	IDC_COMBO_SPECIAL	——	默认
组合框(课程类型)	IDC_COMBO_TYPE	——	默认
编辑框(开课学期)	IDC_EDIT_OPEN	——	默认
旋转按钮	IDC_SPIN1	——	Auto buddy、Set buddy integer、Right 对齐, 其余默认
编辑框(课时数)	IDC_EDIT_COURSEHOURS	——	默认
编辑框(学分)	IDC_EDIT_CREDIT	——	默认

⑤ 右击添加的课程类型的组合框控件, 从弹出的快捷菜单中选择“属性”命令, 将其属性对话框切换到 Data 页面, 直接输入内容, 输入一行后按 Ctrl+Return 键添加另一行。结果如图 2.3 所示。

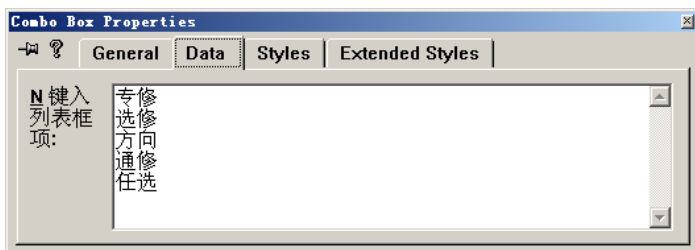


图 2.3 课程类型组合框内容输入

⑥ 按 Ctrl+W 快捷键或双击对话框资源模板的空白处, 为 IDD_COURSE 创建一个对话框 CCourseDlg。

⑦ 打开 ClassWizard 的 Member Variables 页面, 看 Class name 是否是 CCourseDlg, 选中所需的控件 ID 号, 双击鼠标。依次为表 2.2 控件增加成员变量。

表 2.2 控件变量

控件 ID 号	变量类别	变量类型	变量名	范围和大小
IDC_EDIT_COURSENO	Value	CString	m_strNO	
IDC_EDIT_COURSENAME	Value	CString	m_strName	
IDC_COMBO_SPECIAL	Value	CString	m_strSpecial	
IDC_COMBO_SPECIAL	Control	CComboBox	m_comboSpecial	——
IDC_COMBO_TYPE	Value	CString	m_strType	
IDC_EDIT_OPEN	Value	BYTE	m_nOpen	
IDC_SPIN1	Control	CSpinButtonCtrl	m_spinOpen	——
IDC_EDIT_COURSEHOURS	Value	int	m_nHours	
IDC_EDIT_CREDIT	Value	float	m_fCredit	

4. 添加 CCourseDlg 类代码

① 用 MFC ClassWizard 为 CCourseDlg 类添加 WM_INITDIALOG 消息映射，并添加下列初始化代码：

```
BOOL CCourseDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_spinOpen.SetRange( 1, 8 );
    m_nHours = 60;
    m_fCredit = 2.0;
    m_nOpen = 1;
    m_strType = "专修";
    // 这里对专业组合框进行初如化
    m_comboSpecial.AddString( "机械工程及其自动化" );
    m_comboSpecial.AddString( "电气工程及其自动化" );
    m_strSpecial = "机械工程及其自动化";
    UpdateData(FALSE);

    return TRUE; // return TRUE unless you set the focus to a control
}
```

② 用 MFC ClassWizard 为按钮 IDOK 添加 BN_CLICKED 的消息映射，并增加下列代码：

```
void CCourseDlg::OnOK()
{
    UpdateData();
    m_strNO.TrimLeft();
    if (m_strNO.IsEmpty()) {
        MessageBox("课程号不能为空！"); return;
    }
    m_strName.TrimLeft();
    if (m_strName.IsEmpty()) {
        MessageBox("课程名称不能为空！"); return;
    }
}
```

```

        CDialog::OnOK();
    }

```

5. 添加 CEx_CtrlsDlg 程序代码

① 按 Ctrl+W 快捷键，打开 MFC ClassWizard 对话框，为列表框控件 IDC_LIST1 添加控件变量 m_List，类型为 CListBox。

② 用 MFC ClassWizard 为按钮 IDC_BUTTON_COURSE 添加 BN_CLICKED 消息映射，并添加下列代码：

```

void CEx_CtrlsDlg::OnButtonCourse()
{
    CCourseDlg dlg;
    if (IDOK != dlg.DoModal()) return;
    // 清除列表框原来的显示内容
    while(m_List.GetCount() != 0) m_List.DeleteString(0);
    m_List.AddString( "课 程 号: "+dlg.m_strNO);
    m_List.AddString( "课程名称: "+dlg.m_strName);
    m_List.AddString( "所属专业: "+dlg.m_strSpecial);
    m_List.AddString( "课程类型: "+dlg.m_strType);
    CString str;
    str.Format("开课学期: %d", dlg.m_nOpen );
    m_List.AddString( str );
    str.Format("课 时 数: %d", dlg.m_nHours );
    m_List.AddString( str );
    str.Format("学    分: %4.1f", dlg.m_fCredit);
    m_List.AddString( str );
}

```

③ 在 Ex_CtrlsDlg.cpp 文件的前面添加 CInputDialog 的头文件包含：

```

#include "Ex_CtrlsDlg.h"
#include "CourseDlg.h"

```

④ 编译运行并测试。

6. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

(1) 为 CCourseDlg 类添加一个公有型 CString 成员变量 m_strOKText，当通过 CCourseDlg 类对象将 m_strOKText 设为“修改”，CCourseDlg 对话框中的“添加”按钮标题变成“修改”，试编程实现。

(2) 由于每学期的一门课程学分一般不超过 6 个学分，因此若将学分编辑框改为组合框，应如何修改和编程？

实验 2 常用控件（2）

实验目的和要求

（1）设计一个如图 3.1 所示的“学生基本信息”对话框。

（2）在实验 2 的基础上，实现功能：在图 2.1 中的“课程信息”按钮下方添加一个“学生基本信息”按钮，单击该按钮，弹出“学生基本信息”对话框，单击“添加”按钮，学生基本信息显示在列表框中。

（3）在 CEx_CtrlsDlg 对话框中添加一个滚动条和两个滑动条来调整对话框的背景颜色的 3 个分量：R（红色分量）、G（绿色分量）和 B（蓝色分量），结果如图 3.2 所示。

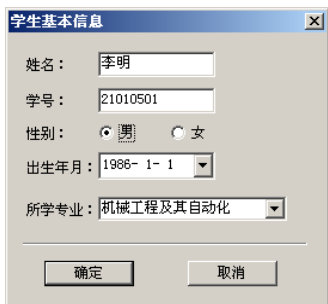


图 3.1 “学生基本信息”对话框

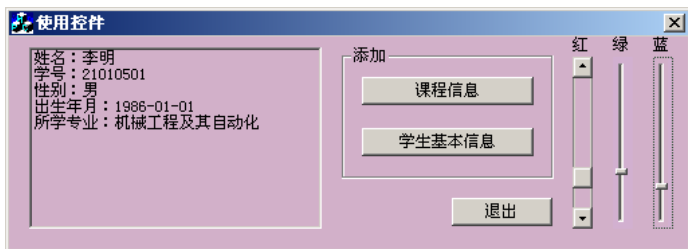


图 3.2 设置对话框背景颜色

实验准备和说明

- （1）具备知识：滚动条、进展条、滑动条、时间和日历控件。
- （2）创建本次实验工作文件夹“...\Visual C++程序\实验\实验 3”。

实验内容和步骤

1. 在实验 3 的工作文件夹中复制 Ex_Ctrls 文件夹

将本书实验 2 中的 Ex_Ctrls 文件夹复制到实验 3 的工作文件夹中。

2. 启动 Visual C++ 6.0

启动 Visual C++ 6.0 系统。

3. 创建并添加“学生基本信息”对话框

① 选择“文件”→“打开工作区”菜单，将“...\Visual C++程序\实验\实验 3\Ex_Ctrls”文件夹中的 Ex_Ctrls 项目打开。

② 添加一个新的对话框资源，将 ID 号改为 IDD_INPUT，标题为“学生成绩输入”，将对话框字体改为“宋体，9 号”。将 OK 和 Cancel 按钮标题改为“确定”和“取消”。

③ 调整对话框的大小，将“确定”和“取消”按钮移至对话框的下方，然后显示对话框网格。

④ 参看图 3.1，向对话框添加如表 3.1 所示的控件。

表 3.1 学生基本信息对话框添加的控件

添加的控件	ID 号	标 题	其 他 属 性
编辑框	IDC_EDIT_NAME	——	默认
编辑框	IDC_EDIT_NO	——	默认
编辑框	IDC_EDIT_S1	——	默认
旋转按钮控件	IDC_SPIN_S1	——	Auto buddy、Right 对齐
编辑框	IDC_EDIT_S2	——	默认
旋转按钮控件	IDC_SPIN_S2	——	Auto buddy、Set buddy integer、Right 对齐
编辑框	IDC_EDIT_S3	——	默认
旋转按钮控件	IDC_SPIN_S3	——	Auto bud0dy、Set buddy integer、Right 对齐

⑤ 双击对话框模板空白处，为该对话框模板创建一个对话框类 CInputDialog。

⑥ 在 MFC ClassWizard 的 Member Variables 页面中，确定 Class name 中是否已选择了 CInputDialog，选中所需的控件 ID 号，双击鼠标或单击 Add Variables 按钮。依次为表 3.2 控件增加成员变量。

表 3.2 控件变量

控件 ID 号	变 量 类 别	变 量 类 型	变 量 名	范围和大小
IDC_EDIT_NAME	Value	CString	m_strName	20
IDC_EDIT_NO	Value	CString	m_strNO	20
IDC_EDIT_S1	Value	float	m_fScore1	0.0 ~ 100.0
IDC_SPIN_S1	Control	CSpinButtonCtrl	m_spinScore1	——
IDC_EDIT_S2	Value	float	m_fScore2	0.0 ~ 100.0
IDC_SPIN_S2	Control	CSpinButtonCtrl	m_spinScore2	——
IDC_EDIT_S3	Value	float	m_fScore3	0.0 ~ 100.0
IDC_SPIN_S3	Control	CSpinButtonCtrl	m_spin Score3	——

⑦ 在 MFC ClassWizard 的 Message Maps 页面中，为 CInputDialog 添加 WM_INITDIALOG 消息映射，并添加下列代码：

```

BOOL CInputDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    m_spinScore1.SetRange( 0, 100 );    // 设置旋转按钮控件范围
    m_spinScore2.SetRange( 0, 100 );
    m_spinScore3.SetRange( 0, 100 );

    return TRUE; // return TRUE unless you set the focus to a control
}

```

⑧ 用 MFC ClassWizard 为 CInputDialog 增加 IDC_SPIN_S1 控件的 UDN_DELTAPOS 消息映射，并添加下列代码：

```

void CInputDialog::OnDeltaposSpinS1(NMHDR* pNMHDR, LRESULT* pResult)
{

```



```

NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;
UpdateData(TRUE);           // 将控件的内容保存到变量中
m_fScore1 += (float)pNMUpDown->iDelta * 0.5f;
if (m_fScore1<0.0) m_fScore1 = 0.0f;
if (m_fScore1>100.0) m_fScore1 = 100.0f;
UpdateData(FALSE);         // 将变量的内容显示在控件中
*pResult = 0;
}

```

⑨ 打开 IDD_EX_CTRLIS_DIALOG 对话框资源，在 “课程信息” 按钮下方添加一个 “学生基本信息” 按钮，并将 ID 号设为 IDC_BUTTON_STUINFO。

⑩ 用 MFC ClassWizard 为按钮 IDC_BUTTON_STUINFO 添加 BN_CLICKED 消息映射，并添加下列代码：

```

void CEx_CtrlsDlg::OnButtonStuinfo()
{
    CStuInfoDlg dlg;
    if (IDOK != dlg.DoModal()) return;
    while(m_List.GetCount() != 0)
        m_List.DeleteString(0);
    CString strSex("女");
    if (dlg.m_bMale) strSex = "男";
    m_List.AddString("姓名: "+dlg.m_strName);
    m_List.AddString("学号: "+dlg.m_strNo);
    m_List.AddString("性别: "+strSex);
    m_List.AddString("出生年月: "+dlg.m_tBirth.Format("%Y-%m-%d"));
    m_List.AddString("所学专业: "+dlg.m_strSpecial);
}

```

⑪ 在 Ex_CtrlsDlg.cpp 文件的前面添加 CStuInfoDlg 类的头文件包含：

```

#include "CourseDlg.h"
#include "StuInfoDlg.h"

```

⑫ 编译运行并测试。结果如图 3.3 所示。

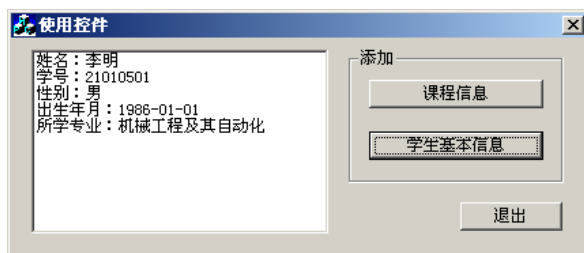


图 3.3 学生基本信息添加后的结果

4. 改变对话框背景颜色

① 将项目工作区窗口切换到 ResourceView 页面，打开 IDD_EX_CTRLIS_DIALOG 对

话框资源。

② 打开对话框网格，参看图 3.2 的控件布局，为对话框添加如表 3.3 所示的一些控件。

表 3.3 添加的控件

添加的控件	ID 号	标 题	其 他 属 性
垂直滚动条	IDC_SCROLLBAR_RED	——	默认
滑动条(绿色)	IDC_SLIDER_GREEN	——	方位为 Vertical, 其他默认
滑动条(蓝色)	IDC_SLIDER_BLUE	——	方位为 Vertical, 其他默认

③ 用 MFC ClassWizard 为表 3.4 控件添加成员变量。

表 3.4 控件变量

控件 ID 号	变 量 类 别	变 量 类 型	变 量 名	范围和大小
IDC_SCROLLBAR_RED	Control	CScrollBar	m_scrollRed	——
IDC_SLIDER_GREEN	Control	CSliderCtrl	m_sliderGreen	——
IDC_SLIDER_GREEN	Value	int	m_nBlue	
IDC_SLIDER_BLUE	Control	CSliderCtrl	m_sliderBlue	——
IDC_SLIDER_BLUE	Value	int	m_nBlue	——

④ 为 CEx_CtrlsDlg 类添加两个成员变量，一个是 int 型 m_nRed，用来设置颜色 R、G、B 中的红色分量，另一个是画刷 CBrush 类对象 m_Brush，用来设置对话框背景所需要的画刷。

⑤ 在 CEx_CtrlsDlg::OnInitDialog 函数中添加下列代码：

```

BOOL CEx_CtrlsDlg::OnInitDialog()
{
    ...
    m_scrollRed.SetScrollRange(0, 255);
    m_sliderBlue.SetRange(0, 255);
    m_sliderGreen.SetRange(0, 255);
    m_nBlue = m_nGreen = m_nRed = 192;
    UpdateData( FALSE );
    m_scrollRed.SetScrollPos(m_nRed);
    return TRUE; // return TRUE unless you set the focus to a control
}

```

⑥ 用 MFC ClassWizard 为 CEx_CtrlsDlg 类映射 WM_VSCROLL 消息，并添加下列代码：

```

void CEx_CtrlsDlg::OnVScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    int nID = pScrollBar->GetDlgCtrlID();           // 获取对话框中控件ID号
    if (nID == IDC_SCROLLBAR_RED) {                 // 或是滚动条产生的水平滚动消息
        switch(nSBCode){
            case SB_LINEUP:      m_nRed--;          // 单击滚动条向上箭头
                                break;

```

```

        case SB_LINEDOWN: m_nRed++;    // 单击滚动条向下箭头
                        break;
        case SB_PAGEUP:   m_nRed -= 10;
                        break;
        case SB_PAGEDOWN: m_nRed += 10;
                        break;
        case SB_THUMBTRACK: m_nRed = nPos;
                        break;
    }
    if (m_nRed < 0) m_nRed = 0;
    if (m_nRed > 255) m_nRed = 255;
    m_scrollRed.SetScrollPos(m_nRed);
}
Invalidate();    // 使对话框无效，强迫系统重绘对话框

CDialog::OnVScroll(nSBCode, nPos, pScrollBar);
}

```

⑦ 用 MFC ClassWizard 为 CEx_CtrlsDlg 类映射 WM_CTLCOLOR 消息，并添加下列代码：

```

HBRUSH CEx_CtrlsDlg::OnCtlColor(CDC* pDC, CWnd* pWnd, UINT nCtlColor)
{
    //HBRUSH hbr = CDialog::OnCtlColor(pDC, pWnd, nCtlColor);
    UpdateData(TRUE);
    COLORREF color = RGB(m_nRed, m_nGreen, m_nBlue);
    m_Brush.Detach();    // 使画刷和对象分离
    m_Brush.CreateSolidBrush(color);    // 创建颜色画刷
    pDC->SetBkColor( color );    // 设置背景颜色
    return (HBRUSH)m_Brush;    // 返回画刷句柄，以便系统使此画刷绘制对话框
}

```

⑧ 编译运行并测试。

5. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

- (1) 若将控制绿色和蓝色颜色分量的滑动条全部换成滚动条，则代码应如何修改？
- (2) 若将 CEx_CtrlsDlg 对话框中的列表框换成静态文本，并用于显示信息，则应如何实现？

实验 3 列表控件和树控件

实验目的要求

- (1) 创建一个对话框应用程序 Ex_List，其主界面如图 4.1 所示。
- (2) 设计一个如图 4.2 所示的“学生课程成绩”对话框，创建并完善该对话框类 CScoreDlg。

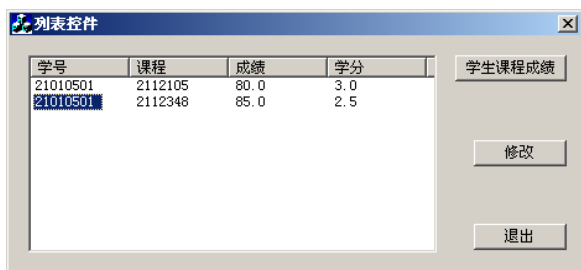


图 4.1 Ex_List 界面

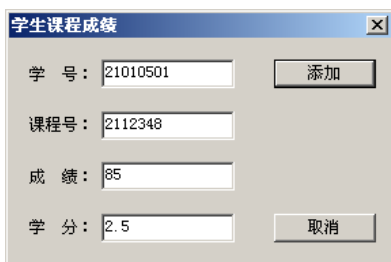


图 4.2 “学生课程成绩”对话框

(3) 实现功能：单击图 4.1 中的“学生课程成绩”按钮，弹出“学生课程成绩”对话框，单击“添加”按钮，学生课程成绩添加到列表控件。若选中列表项，“修改”按钮由原来的禁用变成可用，单击“修改”按钮，则弹出的“学生课程成绩”对话框中的“添加”按钮标题变成“修改”，单击“学生课程成绩”对话框中的“修改”按钮，该列表项的内容被修改。

(4) 按照教材“树控件”中的示例进行实验。

实验准备和说明

- (1) 具备知识：图像列表、列表控件和树控件。
- (2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验 4”。

实验内容和步骤


1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 创建对话框应用程序 Ex_List，并设计其界面

① 选择“文件”→“新建”菜单，在弹出的“新建”对话框中选择“工程”页面，选择 MFC AppWizard (exe)，在工程框中输入 Ex_List，并将工程文件夹定位到“...\Visual C++程序\实验\实验 4”。

② 单击“确定”按钮，在出现的 Step 1 对话框中选择“基本对话框”应用程序类型，单击“完成”按钮。

③ 在对话框编辑器中，单击对话框工具栏上的切换网格按钮 ，显示对话框网格，

将对话框标题改为“列表控件”。

④ 调整对话框的大小，删除对话框中间的“TODO: 在这里设置对话控制。”静态文本控件和“确定”按钮控件，将“取消”按钮标题改为“退出”，并移至对话框的下方。

⑤ 添加两个按钮，一个是“学生课程成绩]按钮，ID 为 IDC_BUTTON_SCORE，另一个是“修改”按钮，ID 为 IDC_BUTTON_CHANGE。

⑥ 添加一个列表控件，取其默认 ID 号，将“查看”风格设为 Report，如图 4.3 所示。



图 4.3 设置列表控件的“查看”风格

3. 添加并设计“学生课程成绩”对话框

① 按 Ctrl+R 快捷键，弹出“插入资源”对话框，在资源类型列表中选择 Dialog，单击“新建”按钮。

② 将该对话框资源的 ID 设为 IDD_SCORE，标题设为“学生课程成绩”，字体设为“宋体，9 号”。

③ 将 OK 和 Cancel 按钮的标题改为“添加”和“取消”。

④ 打开对话框网格，参看图 4.2 的控件布局，为对话框添加如表所示的一些控件。

表 4.1 学生课程成绩对话框添加的控件

添加的控件	ID 号	标 题	其 他 属 性
编辑框(学号)	IDC_EDIT_STUNO	——	默认
编辑框(课程号)	IDC_EDIT_COURSENO	——	默认
编辑框(成绩)	IDC_EDIT_SCORE	——	默认
编辑框(学分)	IDC_EDIT_CREDIT	——	默认

⑤ 按 Ctrl+W 快捷键或双击对话框资源模板的空白处，为 IDD_SCORE 创建一个对话框类 CScoreDlg。

⑥ 打开 ClassWizard 的 Member Variables 页面，看 Class name 是否是 CScoreDlg，选中所需的控件 ID 号，双击鼠标或单击 Add Variables 按钮。依次为表 4.2 控件增加成员变量。

表 4.2 控件变量

控件 ID 号	变 量 类 别	变 量 类 型	变 量 名	范围和大小
IDC_EDIT_STUNO	Value	CString	m_strStuNo	
IDC_EDIT_COURSENO	Value	CString	m_strCourseNo	
IDC_EDIT_SCORE	Value	float	m_fScore	
IDC_EDIT_CREDIT	Value	float	m_fCredit	

⑦ 用 MFC ClassWizard 为按钮 IDOK 添加 BN_CLICKED 消息映射,并增加下列代码:

```
void CScoreDlg::OnOK()
{
    UpdateData();
    m_strStuNo.TrimLeft();
    if (m_strStuNo.IsEmpty()) {
        MessageBox("学号不能为空!");    return;
    }
    m_strCourseNo.TrimLeft();
    if (m_strCourseNo.IsEmpty()) {
        MessageBox("课程号不能为空!");    return;
    }
    CDialog::OnOK();
}
```

⑧ 为 CScoreDlg 类添加一个公有型 CString 类型成员变量 m_strOKText, 用来设置 IDOK 按钮的标题, 并在 CScoreDlg 类构造函数中, 将 m_strOKText 设为空, 如下面的代码:

```
CScoreDlg::CScoreDlg(CWnd* pParent /*=NULL*/)
: CDialog(CScoreDlg::IDD, pParent)
{
    m_strOKText.Empty();
   //{{AFX_DATA_INIT(CScoreDlg)
    ...
   //}}AFX_DATA_INIT
}
```

⑨ 用 MFC ClassWizard 为 CScoreDlg 类映射 WM_INITDIALOG 消息, 并添加下列代码:

```
BOOL CScoreDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    if (!m_strOKText.IsEmpty())
        GetDlgItem( IDOK )->SetWindowText( m_strOKText );
    return TRUE; // return TRUE unless you set the focus to a control
}
```

4. 完善 CEx_ListDlg 类代码

① 用 MFC ClassWizard 为 CEx_ListDlg 类添加列表控件(IDC_LIST1)变量 m_ListCtrl, 变量类型为 CListCtrl。

② 在 CEx_ListDlg::OnInitDialog 函数中添加设置列表控件标题头代码:

```
BOOL CEx_ListDlg::OnInitDialog()
{
```

```

CDialog::OnInitDialog();
...
// 创建列表控件的标题头
CString strHeader[4]={ "学号", "课程", "成绩", "学分"};
for (int nCol=0; nCol<4; nCol++)
    m_ListCtrl.InsertColumn(nCol,strHeader[nCol],LVCFMT_LEFT,80);
GetDlgItem( IDC_BUTTON_CHANGE )->EnableWindow(FALSE);
return TRUE; // return TRUE unless you set the focus to a control
}

```

③ 用 MFC ClassWizard 映射按钮 IDC_BUTTON_SCORE 的 BN_CLICKED 消息，并添加下列代码：

```

void CEx_ListDlg::OnButtonScore()
{
    CScoreDlg dlg;
    if (IDOK != dlg.DoModal()) return;
    int nItem = m_ListCtrl.GetItemCount();
    m_ListCtrl.InsertItem( nItem, dlg.m_strStuNo );
    m_ListCtrl.SetItemText( nItem, 1, dlg.m_strCourseNo );
    CString str;
    str.Format("%4.1f", dlg.m_fScore );
    m_ListCtrl.SetItemText( nItem, 2, str );
    str.Format("%3.1f", dlg.m_fCredit );
    m_ListCtrl.SetItemText( nItem, 3, str );
}

```

④ 用 MFC ClassWizard 映射按钮 IDC_BUTTON_CHANGE 的 BN_CLICKED 消息，并添加下列代码：

```

void CEx_ListDlg::OnButtonChange()
{
    // 获取被选择的列表项索引号
    POSITION pos;
    pos = m_ListCtrl.GetFirstSelectedItemPosition();
    if (pos == NULL){
        MessageBox("你还没有选中列表项！");        return;
    }
    int nItem = m_ListCtrl.GetNextSelectedItem( pos );
    CScoreDlg dlg;
    dlg.m_strOKText = "修改";
    dlg.m_strStuNo = m_ListCtrl.GetItemText( nItem, 0 );
    dlg.m_strCourseNo = m_ListCtrl.GetItemText( nItem, 1 );
    CString str = m_ListCtrl.GetItemText( nItem, 2 );
    dlg.m_fScore = (float)atof( str );
    str = m_ListCtrl.GetItemText( nItem, 3 );
}

```

```

    dlg.m_fCredit = (float)atof( str );
    if (IDOK != dlg.DoModal()) return;
    m_ListCtrl.SetItemText( nItem, 0, dlg.m_strStuNo );
    m_ListCtrl.SetItemText( nItem, 1, dlg.m_strCourseNo );
    str.Format("%4.1f", dlg.m_fScore );
    m_ListCtrl.SetItemText( nItem, 2, str );
    str.Format("%3.1f", dlg.m_fCredit );
    m_ListCtrl.SetItemText( nItem, 3, str );

```

```

}

```

⑤ 用 MFC ClassWizard 映射列表控件 IDC_LIST1 的 LVN_ITEMCHANGED 消息，并添加下列代码：

```

void CEx_ListDlg::OnItemchangedList1(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;
    GetDlgItem( IDC_BUTTON_CHANGE )->EnableWindow(TRUE);
    *pResult = 0;
}

```

⑥ 在 Ex_ListDlg.cpp 文件的前面添加 CScoreDlg 类的头文件包含：

```

#include "Ex_ListDlg.h"
#include "ScoreDlg.h"

```

⑦ 编译运行并测试。

5. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

(1) 在图 4.1 中再添加一个“删除”按钮，若选中列表项，“删除”按钮由原来的禁用变成可用，单击“删除”按钮，删除选中的列表项。

(2) 若将学生课程成绩按“学号”、“课程号”和“成绩”的层次关系显示在一个树控件中，则应如何实现？

实验 4 菜单、工具栏和状态栏

实验目的和内容

- (1) 创建一个单文档应用程序 Ex_SDI。在“查看”菜单下添加一个子菜单“鼠标位置”，ID 为 ID_VIEW_MOUSE，当选择该菜单命令后，鼠标当前的位置显示到状态栏上，同时该菜单项呈选中状态。再次选择该菜单命令，状态栏不再显示当前鼠标位置，同时该菜单项的选中状态被去除。
- (2) 在工具栏上添加并设计一个工具按钮图标，使该按钮和 ID_VIEW_MOUSE 菜单命令联动，并添加一个快捷键 Ctrl+M 和该菜单命令联动。
- (3) 在窗口客户区中右击鼠标，弹出快捷菜单，显示主菜单“查看”中的菜单命令。结果如图 5.1 所示。

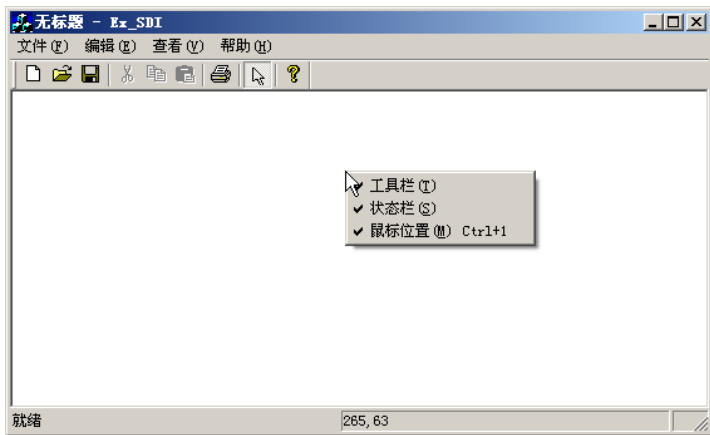


图 5.1 Ex_SDI 运行结果

实验准备和说明

- (1) 具备知识：菜单、工具条和状态栏。
- (2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验 5”。

实验内容和步骤

1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 用 MFC AppWizard (exe) 创建一个默认的单文档应用程序 Ex_SDI

① 选择“文件”→“新建”菜单，在弹出的“新建”对话框中选择“工程”标签，在应用程序项目类型列表框中选择 MFC AppWizard (exe) 的项目类型，将工程文件夹定位到“...\Visual C++程序\实验\实验 5”，并在工程框中输入项目名 Ex_SDI。

② 单击“确定”按钮，从出现的对话框中，选择单个文档（Single Document，SDI）应用程序类型。

③ 保留其他的默认选项，单击“完成”按钮出现一个对话框，显示出用户在步骤中作出的选择，单击“确定”按钮，系统开始创建。

3. 添加菜单

① 在项目工作区窗口中选择 ResourceView 页面，双击资源 Menu 项中的 IDR_MAINFRAME，则菜单编辑器窗口出现在主界面的右边，相应的 Ex_SDI 项目的菜单资源被显示出来。

② 单击“查看”菜单，则在该菜单的最后一项，Visual C++为用户留出了一个空位置，用来输入新的菜单项。

③ 在菜单的空位置上双击鼠标左键，则出现它的属性对话框，如图 5.2 所示，在标题框中输入“鼠标位置(&M)\tCtrl+1”，在 ID 框输入该菜单项的资源标识：ID_VIEW_MOUSE，在提示框中输入“在状态栏上显示当前鼠标位置\n 鼠标位置”，其中\n 前一部分的文本是显示在状态栏上的，后一部分是联动的工具图标按钮的提示文本。



图 5.2 菜单项属性设置

4. 添加并设计一个工具图标按钮

① 在项目工作区窗口的 ResourceView 页面中，双击 Toolbar 中的 IDR_MAINFRAME，打开工具栏资源。

② 单击工具栏最右端的空白按钮，在资源编辑器的按钮设计窗口中绘制一个“箭头”，颜色为黑色，然后将其拖动到“帮助”按钮的前面，并使该按钮的前后均有半个空格，结果如图 5.3 所示。

③ 双击刚才设计的工具按钮，在弹出的属性对话框中将其 ID 设为 ID_VIEW_MOUSE。

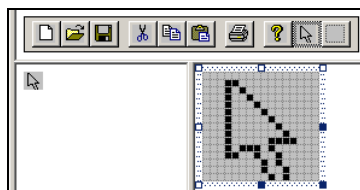


图 5.3 设计的工具按钮

5. 菜单命令和工具按钮的更新

① 为 CMainFrame 类添加一个 BOOL 型的成员变量 m_bIsMouse，在 CMainFrame 类构造函数中将 m_bIsMouse 的初值设为 FALSE。

② 用 MFC ClassWizard 在 CMainFrame 类中添加工具按钮 ID_VIEW_MOUSE 的 COMMAND 和 UPDATE_COMMAND_UI 消息映射函数，并添加下列代码：

```
void CMainFrame::OnViewMouse()
{
```

```

        m_bIsMouse = !m_bIsMouse;
    }
    void CMainFrame::OnUpdateViewMouse(CCmdUI* pCmdUI)
    {
        pCmdUI->SetCheck(m_bIsMouse);
    }

```

③ 编译运行并测试。

6. 设置快捷键

① 在项目工作区窗口的 ResourceView 页面中，双击 Accelerator 中的 IDR_MAINFRAME，打开快捷键资源。

② 双击加速键列表的最下端的空行，弹出如图 5.4 所示的 Accel Properties 对话框，选择菜单项 ID_VIEW_MOUSE 作为要联动的快捷键的 ID 号，然后单击“下一键”按钮，并按下 Ctrl+1 作为此加速键的键值。

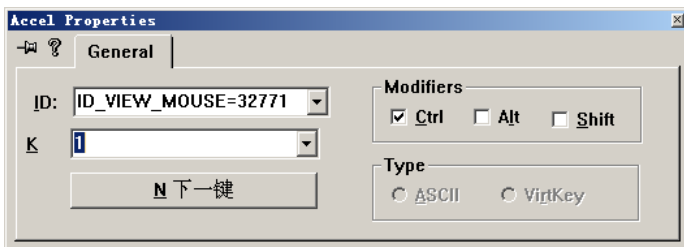


图 5.4 设置快捷键

③ 编译运行并测试。

7. 添加状态栏窗格并显示鼠标当前位置

① 打开 MainFrm.cpp 文件，将原先的 indicators 数组修改如下：

```

static UINT indicators[] =
{
    ID_SEPARATOR,           // status line indicator
    ID_SEPARATOR,
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};

```

② 用 MFC ClassWizard 在 CEx_SDIView 类中映射 WM_MOUSEMOVE（移动鼠标）消息，并在映射函数中添加下列代码：

```

void CEx_SDIView::OnMouseMove(UINT nFlags, CPoint point)
{
    CMainFrame* pFrame=(CMainFrame*)AfxGetApp()->m_pMainWnd;    // 获得主窗口指针
    CStatusBar* pStatus=&pFrame->m_wndStatusBar; // 获得主窗口中的状态栏指针
    CString str;

```

```

        if (pFrame->m_bIsMouse)
            str.Format("%d,%d",point.x, point.y);           // 格式化文本
        else
            str.Empty();           // 为空字符
        if (pStatus)
            pStatus->SetPaneText(1,str);                     // 更新第二个窗格的文本
        CView::OnMouseMove(nFlags, point);
    }

```

③ 将 MainFrm.h 文件中的受保护变量 m_wndStatusBar 变成公共变量。

④ 在 Ex_SDIView.cpp 文件的开始处增加下列语句：

```

#include "Ex_SDIView.h"
#include "MainFrm.h"

```

⑤ 编译运行并测试。

8. 实现快捷菜单

① 用 MFC ClassWizard 在 CMainFrame 类添加 WM_CONTEXTMENU 消息映射，并在映射函数添加下列代码：

```

void CMainFrame::OnContextMenu(CWnd* pWnd, CPoint point)
{
    CMenu* pSysMenu = GetMenu();           // 获得程序菜单指针
    pSysMenu->GetSubMenu(2)
        ->TrackPopupMenu(TPM_LEFTALIGN|TPM_RIGHTBUTTON, point.x, point.y, this);
}

```

② 编译运行并测试。

9. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

若状态栏只有一个用户定义的指示器窗格（其 ID 号为 ID_TEXT_PANE），应如何定义？若当用户在客户区双击鼠标时，在该窗格中显示“双击鼠标”字样，则应如何编程？

实验 5 框架窗口和文档

实验目的和要求

(1) 创建一个多文档应用程序 Ex_MDI，具有两种类型的文档模板，一类是用来操作“课程信息”文档，另一类是用来操作“学生基本信息”文档。其中，用于操作“学生基本信息”的文档和视图类分别为 CEx_StudentDoc 和 CEx_StudentView。

(2) 创建一个可序列化类 CStudentInfo，用于“学生基本信息”的序列化操作。

(3) 创建一个可序列化类 CCourseInfo，用于“课程信息”的序列化操作。

(4) 建立用于两种类型文档的菜单系统，其基本菜单命令包括“添加”、“打开”和“保存”。选择“添加”菜单命令，弹出相应的信息对话框，添加后信息显示在视图中，并保存在相应的 CObArray 类对象。选择“保存”菜单命令，将添加的信息保存在指定的文件中。选择“打开”菜单命令，将保存的文件内容读取并显示在视图中。图 6.1 和 6.2 分别是“课程信息”和“学生基本信息”文档添加时的界面。

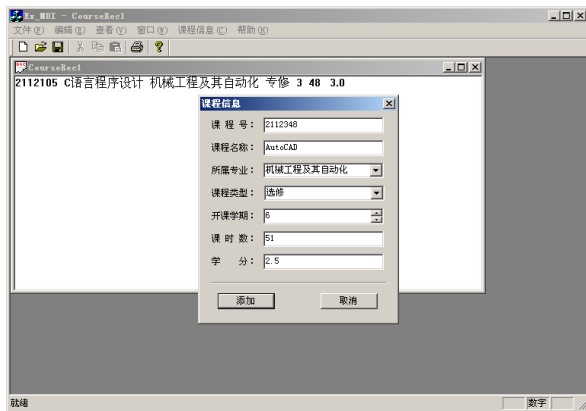


图 6.1 课程信息添加

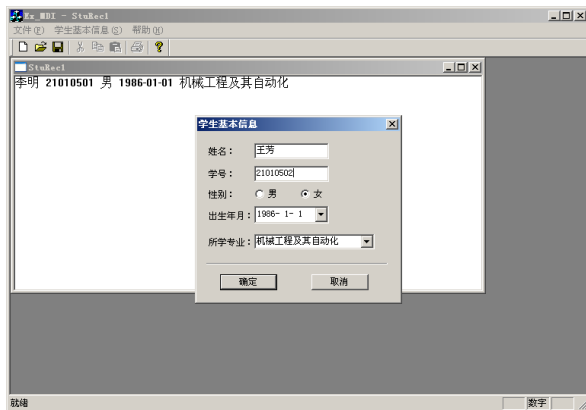


图 6.2 学生基本信息添加

实验准备和说明

- (1) 具备知识：框架窗口、文档模板和文档的读写。
- (2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验6”。
- (3) 建议本实验分两次进行。

实验内容和步骤


1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 创建多文档应用程序 Ex_MDI

用 MFC AppWizard 创建一个默认的多文档应用程序 Ex_MDI。

3. 复制“学生基本信息”和“课程信息”对话框

- ① 将 Ex_Student 项目工作区切换到 ResourceView 页面，展开 Dialog 资源的所有项。
- ② 单击开发环境标准工具栏上的“打开”按钮，打开实验3中 Ex_Ctrls 的资源文件 Ex_Ctrl.rc，展开 Dialog 资源的所有项，选定 IDD_STUINFO 对话框资源项，按住 Ctrl 键，将其拖放到 Ex_Student 项目的 Dialog 资源处。同样，再将 IDD_COURSE 对话框资源击中。

4. 复制对话框类 CStuInfoDlg 和 CCourseDlg

① 单击标准工具栏上的“打开”按钮，在“打开”文件对话框中，定位到 Ex_Ctrls 项目文件夹，选中 StuInfoDlg.h 和 StuInfoDlg.cpp 以及 CourseDlg.h 和 CourseDlg.cpp 文件，按快捷键 Ctrl+C，然后将“打开”文件对话框的文件“查找范围”定位到本项目 Ex_MDI 的文件夹中，按快捷键 Ctrl+V，CStuInfoDlg 和 CCourseDlg 类的源代码文件就复制过来了。

② 关闭“打开”文件对话框，选择“工程”→“添加工程”→Files 菜单，在弹出的 Insert Files Into Project 对话框中选中刚才复制的源文件，单击“确定”按钮，CStuInfoDlg 和 CCourseDlg 类就添加到 Ex_MDI 项目中。

③ 打开 StuInfoDlg.cpp 文件，将文件前面的头文件包含进行修改，如下所示：

```
#include "Ex_MDI.h"           // 修改原来的#include "Ex_Ctrls.h"
#include "StuInfoDlg.h"
```

④ 打开 CourseDlg.cpp 文件，将文件前面的头文件包含进行修改，如下所示：

```
#include "Ex_MDI.h"           // 修改原来的#include "Ex_Ctrls.h"
#include "CourseDlg.h"
```

5. 实现 MFC ClassWizard 对 CStuInfoDlg 和 CCourse 类的支持

若此时打开 MFC ClassWizard 对话框，在 Class name 组合框中有时是找不到刚添加的 CStuInfoDlg 和 CCourse 类。为此需要进行下列操作：

① 单击标准工具栏上的“打开”按钮，在“打开”文件对话框中，将文件类型选择为“所有文件 (*.*)”，在文件列表框中，右击文件 Ex_MDI.clw，从弹出的快捷菜单中选择“删除”命令。这里的 Ex_MDI.clw 是项目 Ex_MDI 的类向导文件。

② 关闭“打开”文件对话框后，按快捷键 **Ctrl+W**，出现一个消息对话框，询问是否从资源中重新创建一个类向导文件，单击“是(Y)”按钮，出现如图 6.3 所示的对话框。

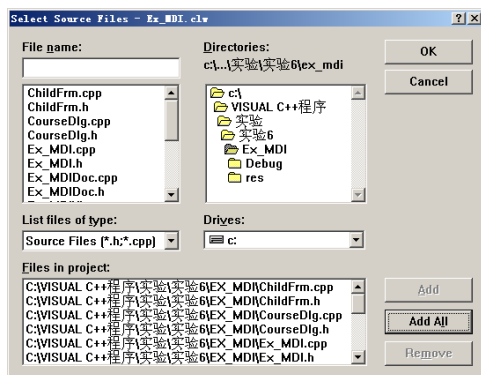


图 6.3 重新创建类向导文件

③ 单击 **Add All** 按钮，再单击 **OK** 按钮。

6. 添加一个 **CStudentInfo** 类并使该类可序列化

① 选择“文件”→“新建”菜单命令，显示出“新建”对话框。单击“文件”标签，在左边的列表框中选择 **C/C++ Header File** 项，在右边的“文件”下的编辑框中输入 **StudentInfo.h**，单击“确定”按钮。

② 在文档窗口中输入下面的代码：

```
class CStudentInfo : public CObject
{
    CString strName;           // 姓名
    CString strNO;             // 学号
    BOOL    bMale;             // 性别，是否为男
    CTime   tBirth;            // 出生年月
    CString strSpecial;         // 专业
    DECLARE_SERIAL(CStudentInfo) // 序列化声明
public:
    CStudentInfo() {};
    CStudentInfo(CString name, CString id, BOOL male, CTime birth, CString special);
    void Serialize(CArchive &ar);
    void Display(int y, CDC *pDC); // 在坐标为(0,y)处显示数据
};
```

③ 再次选择“文件”→“新建”菜单命令，显示出“新建”对话框。单击“文件”标签，在左边的列表框中选择 **C++ Source File** 项，在右边的“文件”下的编辑框中输入 **StudentInfo.cpp**，单击“确定”按钮。

④ 在文档窗口中输入下面的代码：

```
#include "stdafx.h"
#include "StudentInfo.h"
```

```

CStudentInfo::CStudentInfo(CString name, CString id, BOOL male, CTime birth, CString special)
{
    strName      = name;
    strNO        = id;
    bMale        = male;
    tBirth        = birth;
    strSpecial    = special;
}
void CStudentInfo::Display(int y, CDC *pDC)
{
    CString str, strSex("女");
    if (bMale) strSex = "男";
    str.Format("%s %s %s %s %s", strName, strNO,
               strSex, tBirth.Format("%Y-%m-%d"), strSpecial);
    pDC->TextOut(0, y, str);
}
IMPLEMENT_SERIAL(CStudentInfo, CObject, 1) // 序列化实现
void CStudentInfo::Serialize(CArchive &ar)
{
    if (ar.IsStoring())
        ar<<strName<<strNO<<bMale<<tBirth<<strSpecial;
    else
        ar>>strName>>strNO>>bMale>>tBirth>>strSpecial;
}

```

⑤ 编译。

7. 添加一个 CCourseInfo 类并使该类可序列化

① 选择“文件”→“新建”菜单命令，显示出“新建”对话框。单击“文件”标签，在左边的列表框中选择 C/C++ Header File 项，在右边的“文件”下的编辑框中输入 CourseInfo.h，单击“确定”按钮。

② 在文档窗口中输入下面的代码：

```

class CCourseInfo : public CObject
{
    CString strNO;                // 课程号
    CString strName;              // 课程名称
    CString strSpecial;           // 所属专业
    CString strType;              // 课程类型
    BYTE nOpen;                   // 开课学期
    BYTE nHours;                  // 课时数
    float fCredit;                // 学分
    DECLARE_SERIAL(CCourseInfo)   // 序列化声明
public:
    CCourseInfo() {};
    CCourseInfo(CString id, CString name, CString special, CString type,

```



```

        BYTE term, BYTE hours, float credit);
void Serialize(CArchive &ar);
void Display(int y, CDC *pDC);        // 在坐标为(0,y)处显示数据
};

```

③ 再次选择“文件”→“新建”菜单命令，显示出“新建”对话框。单击“文件”标签，在左边的列表框中选择 C++ Source File 项，在右边的“文件”下的编辑框中输入 CourseInfo.cpp，单击“确定”按钮。

④ 在文档窗口中输入下面的代码：

```

#include "stdafx.h"
#include "CourseInfo.h"
CCourseInfo::CCourseInfo(CString id, CString name, CString special, CString type,
    BYTE term, BYTE hours, float credit)
{
    strNO        = id;
    strName       = name;
    strSpecial    = special;
    strType       = type;
    nOpen        = term;
    nHours        = hours;
    fCredit       = credit;
}
void CCourseInfo::Display(int y, CDC *pDC)
{
    CString str;
    str.Format("%s  %s  %s  %s  %d  %d  %4.1f", strNO, strName,
        strSpecial, strType, nOpen, nHours, fCredit);
    pDC->TextOut(0, y, str);
}
IMPLEMENT_SERIAL(CCourseInfo, CObject, 1) // 序列化实现
void CCourseInfo::Serialize(CArchive &ar)
{
    if (ar.IsStoring())
        ar<<strNO<<strName<<strSpecial<<strType<<nOpen<<nHours<<fCredit;
    else
        ar>>strNO>>strName>>strSpecial>>strType>>nOpen>>nHours>>fCredit;
}

```

⑤ 编译。

8. 实现课程信息文档序列化

① 在 Ex_MDIDoc.h 文件的 class CEx_MDIDoc 前面，添加包含 CCourseInfo 类的头文件。

```

#include "CourseInfo.h"

```

② 为 CEx_MDIDoc 类添加下列成员变量，用来保存添加的 CCourseInfo 类对象数据：

public:

```
CObArray m_courseObArray;           // 对象集合类对象
```

③ 为 CEx_MDIDoc 类添加成员函数 CCourseInfo* GetCourseInfoAt (int nIndex)，用来获取 m_courseObArray 中指定索引号的 CCourseInfo 类指针，其代码如下：

```
CCourseInfo* CEx_MDIDoc::GetCourseInfoAt(int nIndex)
{
    if ((nIndex < 0) || nIndex > m_courseObArray.GetUpperBound())
        return NULL;           // 超界处理
    return (CCourseInfo *)m_courseObArray.GetAt(nIndex);
}
```

④ 为 CEx_MDIDoc 类添加成员函数 int GetCourseRecNum (void)，用于获取集合类中对象的个数其代码如下：

```
int CEx_MDIDoc::GetCourseRecNum()
{
    return m_courseObArray.GetSize();
}
```

⑤ 在 CEx_MDIDoc 类析构函数~CEx_MDIDoc 添加下列删除并释放对象的代码：

```
CEx_MDIDoc::~CEx_MDIDoc()
{
    int nIndex = GetCourseRecNum();
    while (nIndex--)
        delete m_courseObArray.GetAt(nIndex); // 删除并释放对象的内存空间
    m_courseObArray.RemoveAll();
}
```

⑥ 在 CEx_StudentDoc::Serialize 函数中添加下列代码：

```
void CEx_MDIDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring()) {
        m_courseObArray.Serialize(ar);
    } else {
        m_courseObArray.Serialize(ar);
    }
}
```

⑦ 在 Ex_MDIDoc.cpp 文件的开始处，添加包含 CCourseDlg 类的头文件包含：

```
#include "Ex_MDIDoc.h"
#include "CourseDlg.h"
```

⑧ 在菜单资源 IDR_EX_MDITYPE 中添加顶层菜单项“课程信息 (&S)”，在该顶层菜单项中添加子菜单“添加 (&A)” (ID_COURSEINFO_ADD)。

⑨ 用 MFC ClassWizard 为 CEx_MDIDoc 类添加处理菜单项 ID_COURSEINFO_ADD 的 COMMAND 消息，并添加下列代码：

```
void CEx_MDIDoc::OnCourseinfoAdd()
{
    CCourseDlg dlg;
    if (dlg.DoModal() != IDOK) return;
    // 添加记录
    CCourseInfo *pCourse = new CCourseInfo(dlg.m_strNO, dlg.m_strName,
        dlg.m_strSpecial, dlg.m_strType, dlg.m_nOpen, dlg.m_nHours, dlg.m_fCredit);
    m_courseObArray.Add(pCourse);
    SetModifiedFlag();          // 设置文档更改标志
    UpdateAllViews(NULL);      // 更新视图
}
```

⑩ 修改 CEx_MDIView::OnDraw 代码，用来将所有的课程信息记录显示在视图中：

```
void CEx_MDIView::OnDraw(CDC* pDC)
{
    CEx_MDIDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    int y = 0;
    for (int nIndex = 0; nIndex < pDoc->GetCourseRecNum(); nIndex++) {
        pDoc->GetCourseInfoAt(nIndex)->Display(y, pDC);
        y += 16;
    }
}
```

⑪ 打开文档的字串资源 IDR_EX_MDITYPE，将其内容修改为：

```
\nCourseRec\nEx_MDI\n课程信息文件(*.cou)\n.cou\nExMDI.Document\nEx_MDI Document
```

⑫ 编译并运行。

9. 第一次测试

① 运行后，选择“课程基本信息”→“添加”菜单命令，弹出“课程信息”对话框，输入相关信息后，单击“确定”按钮。重复刚才的操作，添加两个课程信息记录，结果如图 6.1 所示。

② 选择“文件”→“保存”菜单命令，弹出保存文件对话框，指定要保存的文件名 1.cou，单击“保存”按钮，这样就将添加的记录保存到 1.cou 文件中。

③ 关闭应用程序，然后重新运行。选择“文件”→“打开”菜单命令，从弹出的打开文件对话框中指定刚才保存的 1.cou 文件，单击“打开”按钮，该文件中的课程信息记录被保存到 m_courseObArray 中，并在视图中显示出来。

④ 再添加一个课程信息记录，保存后再打开，结果如何？

10. 添加另一个文档模板类型

① 打开项目工作区窗口中 String Table 的资源项，双击该项下的 String Table，打开字符串表资源，拖动字符串表编辑器右边的滚动块，直到出现最后一个字符串项，双击最后的空行，在字符串属性对话框中将 ID 设为 IDR_STUDENT，值为 130，标题内容设为：

\nStuRec\nEx_Student\n学生基本信息文件(*.stu)\n.stu\nExMDI.Document\nEx_MDI Document

② 按快捷键 Ctrl+W，打开 MFC ClassWizard，单击 Add Class 按钮，从弹出的菜单中选择 New，出现 New Class 对话框，在 Name 框中输入类名 CEx_StudentDoc，在 Base class 组合框中选择基类 CDocument。单击 OK 按钮，新的文档类 COtherDoc 就添加到 Ex_MDI 项目中。

③ 类似地再添加一个新的视图类 CEx_StudentView，基类为 CView。单击“确定”按钮，关闭 MFC ClassWizard 对话框。

④ 修改 CEx_MDIApp::InitInstance 函数代码，如下所示：

```
BOOL CEx_MDIApp::InitInstance()
{
    ...

    CMultiDocTemplate* pDocTemplate;
    pDocTemplate = new CMultiDocTemplate(
        IDR_EX_MDITYPE,
        RUNTIME_CLASS(CEx_MDIDoc),
        RUNTIME_CLASS(CChildFrame),    // custom MDI child frame
        RUNTIME_CLASS(CEx_MDIView));
    AddDocTemplate(pDocTemplate);

    pDocTemplate = new CMultiDocTemplate(
        IDR_STUDENT,                    // 指定新的资源
        RUNTIME_CLASS(CEx_StudentDoc), // 指定新的文档类
        RUNTIME_CLASS(CChildFrame),
        RUNTIME_CLASS(CEx_StudentView)); // 指定新的视图类
    AddDocTemplate(pDocTemplate);

    ...
}
```

⑤ 在文件 Ex_MDI.cpp 的开始处，添加包含前面创建的两个派生类的头文件包含：

```
#include "Ex_MDIDoc.h"
#include "Ex_MDIView.h"
#include "Ex_StudentDoc.h"
#include "Ex_StudentView.h"
```

⑥ 编译并运行。

11. 实现学生基本信息文档序列化

① 在 Ex_StudentDoc.cpp 文件的开始处，添加包含 CStuInfoDlg 类的头文件包含。

```
#include "Ex_StudentDoc.h"
```

```
#include "StuInfoDlg.h"
```

② 将项目工作区切换到 ResourceView 页面，展开 Menu 所有节点，选中 IDR_EX_MDITYPE，然后按住 Ctrl 键，将其拖放到 Menu 节点上，将 IDR_EX_MDITYPE1 标识改为 IDR_STUDENT。

③ 保留“文件”和“帮助”菜单，其他删除。添加顶层菜单项“学生基本信息(&S)”，在该顶层菜单项中添加子菜单“添加(&A)”(ID_STUINFO_ADD)。

④ 用 MFC ClassWizard 为 CEx_StudentDoc 类添加处理菜单项 ID_STUINFO_ADD 的 COMMAND 消息，并添加下列代码：

```
void CEx_StudentDoc::OnStuinfoAdd()
{
    CStuInfoDlg dlg;
    if (dlg.DoModal() != IDOK) return;
    // 添加记录
    CStudentInfo *pStudent = new CStudentInfo(dlg.m_strName,
        dlg.m_strNo, dlg.m_bMale, dlg.m_tBirth, dlg.m_strSpecial);
    m_stuObArray.Add(pStudent);
    SetModifiedFlag();           // 设置文档更改标志
    UpdateAllViews(NULL);       // 更新视图
}
```

⑤ 修改 CEx_StudentDoc 类代码（与教材文档序列化示例 Ex_Student 相同）。

⑥ 在 CEx_StudentView::OnDraw 中添加下列代码：

```
void CEx_StudentView::OnDraw(CDC* pDC)
{
    CEx_StudentDoc* pDoc = (CEx_StudentDoc*)GetDocument();
    int y = 0;
    for (int nIndex = 0; nIndex < pDoc->GetAllRecNum(); nIndex++) {
        pDoc->GetStudentInfoAt(nIndex)->Display(y, pDC);
        y += 16;
    }
}
```

⑦ 在 Ex_StudentView.cpp 文件的前面添加 CEx_StudentDoc 类的头文件包含：

```
#include "Ex_StudentView.h"
#include "Ex_StudentDoc.h"
```

⑧ 编译运行。在文档类型“新建”对话框中，选中 Ex_Student，单击“确定”按钮，出现程序主界面，然后进行测试，测试的内容与教材文档序列化示例 Ex_Student 相同。结果如图 6.2 所示。

12. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

- (1) 经过上述实验后，简述对类的序列化和文档序列化的理解。
- (2) 文档字符串资源中，哪一段内容影响文档类型“新建”对话框的内容？将“新建”对话框的文档类型显示为“课程信息文档”和“学生基本信息文档”。

实验 6 切分窗口

实验目的和要求

(1) 如图 7.1 所示是一个有切分窗口的单文档应用程序。左边是树视图，用于显示“学生信息”、“专业”和“班号”3 层结构信息。右边是列表视图，以报表的形式显示学生基本信息。

(2) 单击图 7.1 右边视图，选择“学生基本信息”→“添加”菜单命令，弹出“学生基本信息”对话框，单击“添加”按钮，学生基本信息被保存在 CObArray 类对象 m_stuObArray 中，同时显示在列表视图中，并自动更新左边树视图的节点内容。

(3) 单击左边树视图“学生信息”节点，则在列表视图中全部显示所有学生的基本信息，若单击“专业”节点，则显示所有该专业的学生基本信息，若单击“班号”节点，则显示所有该班级的学生基本信息。

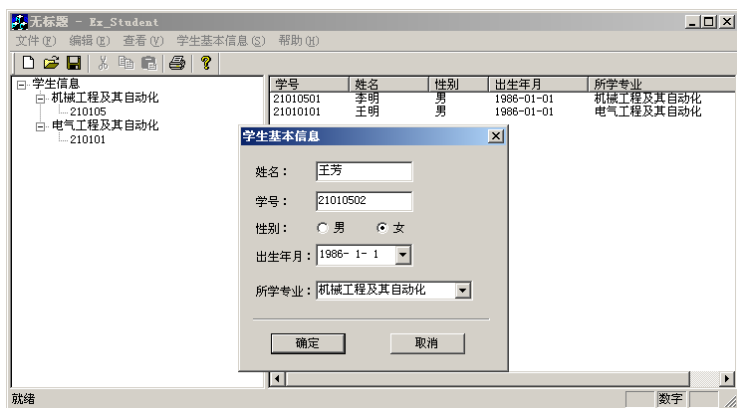


图 7.1 实验 7 运行结果

实验准备和说明

- (1) 具备知识：列表视图、树视图、切分窗口。
- (2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验 7”。

实验内容和步骤

1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 创建一个单文档应用程序 Ex_Student

① 用 MFC AppWizard 创建一个单文档应用程序 Ex_Student，在向导的第六步将 CEx_StudentView 的基类由 CView 改为 CListView。

② 参看上一个实验，将“学生基本信息”对话框及类 CStuInfoDlg 复制过来。

③ 按照上一个实验的代码, 添加 `CStudentInfo` 类以及在 `CEx_StudentDoc` 类中添加 `CObArray` 类对象 `m_stuObArray` 和 `GetStudentInfoAt` 及 `GetAllRecNum` 成员函数。

④ 将 `CStudentInfo` 类的所有成员变量变成 `public`。

3. 创建切分窗口

① 用 MFC ClassWizard 添加一个新类 `CStudentTreeView`, 基类为 `CTreeView`。

② 打开 `MainFrm.h` 文件, 为 `CMainFrame` 类添加下列成员变量:

`public:`

```
CSplitterWnd m_wndSplitter;
```

③ 用 MFC ClassWizard 为 `CMainFrame` 类添加 `OnCreateClient` 函数的重载, 并添加下列代码:

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
    CRect rc;
    GetClientRect(rc);                // 获取客户区大小
    CSize paneSize(rc.Width()/3,rc.Height()); // 计算每个窗格的平均尺寸
    m_wndSplitter.CreateStatic(this,1,2);
    m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CStudentTreeView),
        paneSize,pContext);           // 为相应的窗格指定视图类
    m_wndSplitter.CreateView(0,1,RUNTIME_CLASS(CEx_StudentView),
        CSize(0,0),pContext);
    m_wndSplitter.SetActivePane( 0, 1 ); // 设置初始活动窗格
    return TRUE;
}
```

④ 在 `MainFrm.cpp` 的前面添加下列语句:

```
#include "MainFrm.h"
#include "StudentTreeView.h"
#include "Ex_StudentView.h"
```

⑤ 打开 `Ex_StudentView.h` 文件, 在 `class CEx_StudentView : public CListView` 语句前面添加下列代码:

```
class CEx_StudentDoc;
class CEx_StudentView : public CListView
{
...
}
```

⑥ 编译运行。

4. 添加 `CStudentTreeView` 类代码

① 用 MFC ClassWizard 为 `CStudentTreeView` 类添加 `PreCreateWindow` 函数的重载, 并在该函数中添加下列代码, 用来设置树视图内嵌树控件的风格:

```
BOOL CStudentTreeView::PreCreateWindow(CREATESTRUCT& cs)
```



```

{
    cs.style |= (TVS_HASLINES | TVS_LINESATROOT | TVS_HASBUTTONS );
    return CTreeView::PreCreateWindow(cs);
}

```

② 为 CStudentTreeView 类添加一个成员函数 FindTreeItem，用来查找指定节点下是否有指定节点文本的子节点，该函数的代码如下：

```

HTREEITEM CStudentTreeView::FindTreeItem(CTreeCtrl &treeCtrl, HTREEITEM hParent, CString str)
{
    HTREEITEM hNext;
    CString strItem;
    hNext = treeCtrl.GetChildItem( hParent);
    while (hNext != NULL) {
        strItem = treeCtrl.GetItemText( hNext );
        if ( strItem == str ) {
            return hNext;
        } else {
            hNext = treeCtrl.GetNextItem( hNext, TVGN_NEXT );
        }
    }
    return NULL;
}

```

③ 为 CStudentTreeView 类添加一个成员函数 ResetTreeItem，用来根据 m_stuObArray 中的内容更新树节点，该函数的代码如下：

```

void CStudentTreeView::ResetTreeItem()
{
    CTreeCtrl& m_TreeCtrl = GetTreeCtrl();
    if (m_TreeCtrl.GetCount()>0) m_TreeCtrl.DeleteAllItems();    // 删除原来的所有节点
    CEx_StudentDoc* pDoc = (CEx_StudentDoc*)GetDocument();
    HTREEITEM hRoot, hSpec, hClass;
    CString strSpecial, strClass;
    hRoot = m_TreeCtrl.InsertItem("学生信息",0,1);
    for (int nIndex = 0; nIndex < pDoc->GetAllRecNum(); nIndex++) {
        strSpecial = pDoc->GetStudentInfoAt(nIndex)->strSpecial;
        strClass = pDoc->GetStudentInfoAt(nIndex)->strNO.Left(6);
        // 学号的前6位是班级号
        hSpec = FindTreeItem( m_TreeCtrl, hRoot, strSpecial);
        // 查找是否有重复的专业节点
        if (hSpec == NULL)                // 若没有重复的专业节点
            hSpec = m_TreeCtrl.InsertItem( strSpecial, 0, 1, hRoot);
        hClass = FindTreeItem( m_TreeCtrl, hSpec, strClass);
        // 查找是否有重复的班级节点
        if (hClass == NULL)                // 若没有重复的班级节点

```

```

        hClass = m_TreeCtrl.InsertItem(strClass, 0, 1, hSpec);
    }
}

```

④ 在 StudentTreeView.cpp 文件的前面添加 CEx_StudentDoc 类的头文件包含：

```

#include "StudentTreeView.h"
#include "Ex_StudentDoc.h"

```

⑤ 为 CStudentTreeView 类添加一个 CImageList 成员变量 m_ImageList。

⑥ 用 MFC ClassWizard 为 CStudentTreeView 类添加 OnInitialUpdate 函数的重载，并添加下列代码：

```

void CStudentTreeView::OnInitialUpdate()
{
    CTreeView::OnInitialUpdate();
    CTreeCtrl& m_TreeCtrl = GetTreeCtrl();
    m_ImageList.Create(16, 16, ILC_COLOR8 | ILC_MASK, 2, 1);
    m_TreeCtrl.SetImageList( &m_ImageList, TVSIL_NORMAL );
    SHFILEINFO fi; // 定义一个文件信息结构变量
    SHGetFileInfo("C:\\Windows", 0, &fi, sizeof(SHFILEINFO),
        SHGFI_ICON | SHGFI_SMALLICON); // 获取文件夹图标
    m_ImageList.Add( fi.hIcon );
    SHGetFileInfo("C:\\Windows", 0, &fi, sizeof(SHFILEINFO),
        SHGFI_ICON | SHGFI_SMALLICON | SHGFI_OPENICON); // 获取打开文件夹图标
    m_ImageList.Add( fi.hIcon );
}

```

5. 添加 CEx_StudentView 类代码

① 在 CEx_StudentView::PreCreateWindow 函数添加下列代码，用来设置列表视图内嵌列表控件的风格：

```

BOOL CEx_StudentView::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.style |= LVS_REPORT; // 报表风格
    return CListView::PreCreateWindow(cs);
}

```

② 在 CEx_StudentView::OnInitialUpdate 函数中添加下列代码，用来创建列表标题头：

```

void CEx_StudentView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    CListCtrl& m_ListCtrl = GetListCtrl();
    CString strHeader[]={ "学号", "姓名", "性别", "出生年月", "所学专业" };
    int nLong[] = { 80, 80, 60, 100, 180 };
    for (int nCol=0; nCol<sizeof(strHeader)/sizeof(CString); nCol++)

```

```

        m_ListCtrl.InsertColumn(nCol,strHeader[nCol],LVCFMT_LEFT,nLong[nCol]);
    }

```

③ 为 CEx_StudentView 类添加一个成员函数 DispStudentInfo，用来根据指定的条件在列表视图中用报表形式显示学生基本信息，该函数的代码如下：

```

void CEx_StudentView::DispStudentInfo(CString strSpecial, CString strClass)
{
    CListCtrl& m_ListCtrl = GetListCtrl();
    m_ListCtrl.DeleteAllItems();    // 删除所有的列表项
    CEx_StudentDoc* pDoc = (CEx_StudentDoc*)GetDocument();
    CString strName, strNO, strSex, strBirth, strStuSpecial;
    int nItem = 0;
    for (int nIndex = 0; nIndex < pDoc->GetAllRecNum(); nIndex++) {
        strName      = pDoc->GetStudentInfoAt(nIndex)->strName;
        strNO        = pDoc->GetStudentInfoAt(nIndex)->strNO;
        if (pDoc->GetStudentInfoAt(nIndex)->bMale)      strSex = "男";
        else          strSex = "女";
        strBirth      = pDoc->GetStudentInfoAt(nIndex)->tBirth.Format("%Y-%m-%d");
        strStuSpecial  = pDoc->GetStudentInfoAt(nIndex)->strSpecial;
        BOOL bAdd = FALSE;
        if (!strSpecial.IsEmpty()){          // 指定专业条件
            if (strStuSpecial == strSpecial)    bAdd = TRUE;
        } else if (!strClass.IsEmpty()) {      // 指定班级条件
            if (strNO.Left(6) == strClass)bAdd = TRUE;
        } else
            bAdd = TRUE;
        if (bAdd) {
            m_ListCtrl.InsertItem( nItem, strNO );
            m_ListCtrl.SetItemText( nItem, 1, strName );
            m_ListCtrl.SetItemText( nItem, 2, strSex );
            m_ListCtrl.SetItemText( nItem, 3, strBirth );
            m_ListCtrl.SetItemText( nItem, 4, strStuSpecial );
            nItem++;
        }
    }
}

```

6. 完善代码

① 在 Ex_StudentView.cpp 文件的开始处，添加包含 CStuInfoDlg 类的头文件包含：

```

#include "Ex_StudentView.h"
#include "StuInfoDlg.h"

```

② 在菜单资源的主菜单中添加顶层菜单项“学生基本信息 (&S)”，在该顶层菜单项中添加子菜单“添加 (&A)” (ID_STUINFO_ADD)。

③ 用 MFC ClassWizard 为 CEx_StudentView 类添加处理菜单项 ID_STUINFO_ADD 的 COMMAND 消息，并添加下列代码：

```
void CEx_StudentView::OnStuinfoAdd()
{
    CStuInfoDlg dlg;
    if (dlg.DoModal() != IDOK) return;
    CEx_StudentDoc* pDoc = (CEx_StudentDoc*)GetDocument();
    // 添加记录
    CStudentInfo *pStudent = new CStudentInfo(dlg.m_strName,
        dlg.m_strNo, dlg.m_bMale, dlg.m_tBirth, dlg.m_strSpecial);
    pDoc->m_stuObArray.Add(pStudent);
    DispStudentInfo( "", dlg.m_strNo.Left(6));    // 按班级显示学生基本信息
    pDoc->UpdateAllViews(NULL, 1);                // 更新视图
}
```

④ 用 MFC ClassWizard 为 CStudentTreeView 类添加 OnUpdate 重载，并添加下列代码：

```
void CStudentTreeView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    if (lHint == 1)                                // 是从左边列表视图传递过来的消息
        ResetTreeItem();
}
```

⑤ 用 MFC ClassWizard 为 CStudentTreeView 类添加 WM_LBUTTONDOWN 的消息映射，并添加下列代码：

```
void CStudentTreeView::OnLButtonDown(UINT nFlags, CPoint point)
{
    UINT uFlags;
    CTreeCtrl& m_TreeCtrl = GetTreeCtrl();
    HTREEITEM hSel = m_TreeCtrl.HitTest(point, &uFlags);
    // 测试鼠标点是否一个节点项，若是，选中该节点
    if ((hSel != NULL) && (TVHT_ONITEM & uFlags))
    {
        m_TreeCtrl.SelectItem(hSel);
    } else {
        CTreeView::OnLButtonDown(nFlags, point);
        return;
    }
    CString strSelItem;
    strSelItem = m_TreeCtrl.GetItemText( hSel );
    int nHint;    // 1表示“学生信息”，2表示“专业”，3表示班级
    // 如果击中的节点没有子节点，那说明该节点是班级号节点
    if (m_TreeCtrl.GetChildItem(hSel) == NULL)
        nHint = 3;
```

```

else
    if (strSelItem == "学生信息")
        nHint = 1;
    else
        nHint = 2;
    GetDocument()->UpdateAllViews( NULL, nHint, (CObject *)new CString(strSelItem));
}

```

⑥ 用 MFC ClassWizard 为 CEx_StudentView 添加 OnUpdate 函数的重载，并添加下列代码，用来根据左边树视图发送过来的数据显示学生基本信息：

```

void CEx_StudentView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{

```

```

    CString strHint;
    if (pHint) {
        strHint = *((CString *)pHint);
        delete (CString *)pHint;
    }
    switch ((int)lHint) {
        case 1:        DispStudentInfo("", "");           // 全部学生信息
                        break;
        case 2:        DispStudentInfo(strHint, "");      // 按专业
                        break;
        case 3:        DispStudentInfo("", strHint);      // 按班号
                        break;
    }
}

```

⑦ 编译运行并测试。

7. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

(1) 在本实验中，在“学生基本信息”菜单添加一个“修改”菜单，并在 CEx_StudentView 类中映射该菜单命令，使得能修改当前列表视图被选中列表项的学生信息，试完善其代码。

(2) 若添加的学生基本信息能保存到文档，则应如何实现？

实验 7 图形、文本和打印

实验目的和要求

(1) 创建一个基于 CListView 的单文档应用程序 Ex_Student，用于以报表方式来显示学生基本信息。

(2) 修改“学生基本信息”对话框，使其能显示学生 BMP 照片文件。如图 8.1 所示。

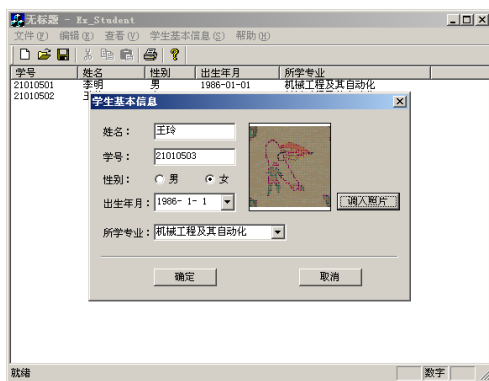


图 8.1 “学生基本信息”对话框

(3) 选择“学生基本信息”→“添加”菜单命令，弹出“学生基本信息”对话框，单击“添加”按钮，学生基本信息添加到列表视图中，并且图片文件被复制到指定文件夹中。

(4) 打印并能预览列表视图中所有的列表项内容。如图 8.2 所示。

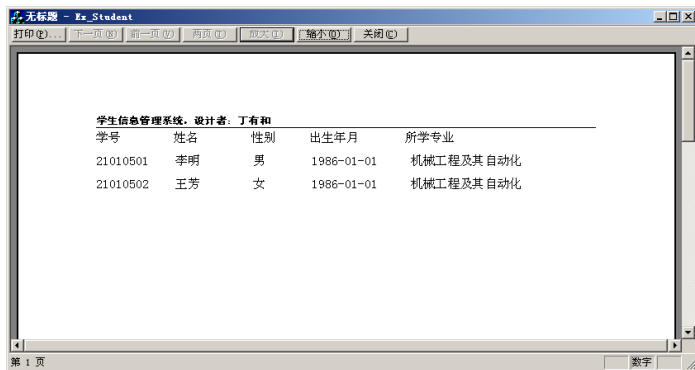


图 8.2 打印预览结果

实验准备和说明

(1) 具备知识：图形、文本、BMP 位图、打印和打印预览。

(2) 创建本次实验工作文件夹“...\Visual C++程序\实验\实验 8”。

实验内容和步骤

1. 启动 Visual C++ 6.0

打开计算机，启动 Visual C++ 6.0 系统。

2. 创建一个单文档应用程序 Ex_Student

① 用 MFC AppWizard 创建一个单文档应用程序 Ex_Student，在向导的第六步将 CEx_StudentView 的基类由 CView 改为 CListView。

② 在 CEx_StudentView::PreCreateWindow 函数添加下列代码，用来设置列表视图内嵌列表控件的风格：

```
BOOL CEx_StudentView::PreCreateWindow(CREATESTRUCT& cs)
{
    cs.style |= LVS_REPORT;           // 报表风格
    return CListView::PreCreateWindow(cs);
}
```

③ 在 CEx_StudentView::OnInitialUpdate 函数中添加下列代码，用来创建列表标题头：

```
void CEx_StudentView::OnInitialUpdate()
{
    CListView::OnInitialUpdate();
    CListCtrl& m_ListCtrl = GetListCtrl();
    CString strHeader[]={"学号","姓名","性别","出生年月","所学专业"};
    int nLong[]={80,80,60,100,180};
    for (int nCol=0; nCol<sizeof(strHeader)/sizeof(CString); nCol++)
        m_ListCtrl.InsertColumn(nCol,strHeader[nCol],LVCFMT_LEFT,nLong[nCol]);
}
```

④ 编译运行。

3. 添加并修改“学生基本信息”对话框及其代码

① 将实验 6 中的“学生基本信息”对话框及类 CStuInfoDlg 复制过来。

② 打开 StuInfoDlg.cpp 文件，将前面的头文件进行下列修改：

```
#include "stdafx.h"
#include "Ex_Student.h"           // 将Ex_MDI.h修改成Ex_Student.h
#include "StuInfoDlg.h"
```

③ 打开 IDD_STUINFO 对话框资源，单击控件布局工具栏上的网格按钮，如图 8.1 所示，调整对话框的大小，添加两个控件：一个是静态文本控件，ID 设为 IDC_DRAW，选中“下沉”风格；另一个是按钮控件，标题为“调入照片”，ID 为 ID_BUTTON_PHOTO。

④ 为 CStuInfoDlg 类添加下列成员变量：

```
public:
    BOOL        m_bPhotoChange;    // 照片文件是否重新指定
    HBITMAP     m_hBitmap;         // 位图句柄
```

```
CString      m_strPhotoFilePath;      // 照片文件全路径名
```

⑤ 在 CStuInfoDlg::OnInitDialog 中添加下列代码:

```
BOOL CStuInfoDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    ...
    if (!m_strPhotoFilePath.IsEmpty()) {
        m_hBitmap = (HBITMAP)::LoadImage(AfxGetInstanceHandle(), m_strPhotoFilePath,
        IMAGE_BITMAP, 0,0,LR_LOADFROMFILE|LR_CREATEDIBSECTION);
        // 将外部BMP文件装载并返回HBITMAP句柄
    }
    m_bPhotoChange = FALSE;
    return TRUE; // return TRUE unless you set the focus to a control
}
```

⑥ 用 MFC ClassWizard 为 CStuInfoDlg 类映射 ID_BUTTON_PHOTO 按钮的 BN_CLICKED 消息, 并添加下列代码:

```
void CStuInfoDlg::OnButtonPhoto()
{
    static char BASED_CODE szFilter[] = "图片文件(*.bmp)|*.bmp|";
    CFileDialog dlg( TRUE,NULL, NULL, OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,
    szFilter);
    if (IDOK != dlg.DoModal()) return;
    m_bPhotoChange = TRUE;
    m_strPhotoFilePath = dlg.GetPathName();
    m_hBitmap = (HBITMAP)::LoadImage(AfxGetInstanceHandle(), m_strPhotoFilePath,
    IMAGE_BITMAP, 0,0,LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    Invalidate(); // 强制对话框无效, 调用OnPaint
}
```

⑦ 为 CStuInfoDlg 类添加成员函数 DrawPhoto, 用于在静态文本控件 IDC_DRAW 中绘制照片。

```
void CStuInfoDlg::DrawPhoto()
{
    if(!m_hBitmap) return;
    CBitmap m_bmp;
    m_bmp.Attach(m_hBitmap);
    BITMAP bm; // 定义一个BITMAP结构变量, 以便获取位图参数
    m_bmp.GetObject(sizeof(BITMAP),&bm);
    float fScale; // 高/宽
    fScale = (float)bm.bmHeight / (float)bm.bmWidth;
    CWnd* pWnd = GetDlgItem(IDC_DRAW); // 获得IDC_DRAW控件窗口指针
    CDC* pDC = pWnd->GetDC(); // 获得窗口当前的设备环境指针
```



```

// 调整可以显示的矩形大小
CRect rcClient;
int nWidth, nHeight, nX = 0, nY = 0;
pWnd->GetClientRect( rcClient );
nWidth = rcClient.Width();
nHeight = (int)((float)nWidth * fScale);
nX = 0;    nY = ( rcClient.Height() - nHeight ) / 2;
if ( nHeight > rcClient.Height() ) {
    nHeight = rcClient.Height();    nWidth = (int)((float)nHeight/fScale);
    nX = ( rcClient.Width() - nWidth )/2;    nY = 0;
}
CDC dcMem;                                // 定义并创建一个内存设备环境
dcMem.CreateCompatibleDC(pDC);
CBitmap *pOldbmp = dcMem.SelectObject(&m_bmp); // 将位图选入内存设备环境中
rcClient.DeflateRect(-1, -1);
pDC->Rectangle( rcClient );
pDC->StretchBlt(nX, nY, nWidth, nHeight,
    &dcMem,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
// 将位图复制到实际的设备环境中
dcMem.SelectObject(pOldbmp);                // 恢复原来的内存设备环境
}

```

⑧ 用 MFC ClassWizard 为 CStuInfoDlg 类映射 WM_PAINT 消息，并添加下列代码：

```

void CStuInfoDlg::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    CWnd* pWnd = GetDlgItem(IDC_DRAW);
    pWnd->UpdateWindow();
    DrawPhoto();
}

```

4. 测试“学生基本信息”对话框

① 在 Ex_StudentView.cpp 文件的开始处，添加包含 CStuInfoDlg 类的头文件包含：

```

#include "Ex_StudentView.h"
#include "StuInfoDlg.h"

```

② 在菜单资源的主菜单中添加顶层菜单项“学生基本信息 (&S)”，在该顶层菜单项中添加子菜单“添加 (&A)” (ID_STUINFO_ADD)。

③ 用 MFC ClassWizard 为 CEx_StudentView 类添加处理菜单项 ID_STUINFO_ADD 的 COMMAND 消息，并添加下列代码：

```

void CEx_StudentView::OnStuinfoAdd()
{
    CStuInfoDlg dlg;
    if (dlg.DoModal() != IDOK) return;
}

```

```
}
```

④ 编译运行并测试。

5. 完善 OnStuinfoAdd 代码

① 为 CEx_StudentView 类添加成员函数 DoCopyFile，用于复制文件，其代码如下：

```
void CEx_StudentView::DoCopyFile(CString strFileTo, CString strFileFrom)
```

```
{
    CFile fileFrom, fileTo;
    if (!fileFrom.Open( strFileFrom, CFile::modeRead)) {
        AfxMessageBox("源文件无法打开，复制失败！");          return;
    }
    fileTo.Open( strFileTo, CFile::modeCreate | CFile::modeWrite );
    BYTE buffer[4096];
    DWORD dwRead;
    do{
        dwRead = fileFrom.Read(buffer, 4096);
        fileTo.Write(buffer, dwRead);
    } while (dwRead > 0);
    fileFrom.Close();
    fileTo.Close();
}
```

② 在 CEx_StudentView::OnStuinfoAdd 函数中添加下列代码：

```
void CEx_StudentView::OnStuinfoAdd()
```

```
{
    CStuInfoDlg dlg;
    if (dlg.DoModal() != IDOK) return;
    // 复制照片文件
    CString strFileTO;
    strFileTO.Format("c://%s.bmp", dlg.m_strNo);
    if (!dlg.m_strPhotoFilePath.IsEmpty())
        DoCopyFile(strFileTO, dlg.m_strPhotoFilePath);
    // 添加列表项
    CListCtrl& m_ListCtrl = GetListCtrl();
    int nItem = m_ListCtrl.GetItemCount();
    CString strSex("女");
    if (dlg.m_bMale) strSex = "男";
    m_ListCtrl.InsertItem( nItem, dlg.m_strNo );
    m_ListCtrl.SetItemText( nItem, 1, dlg.m_strName );
    m_ListCtrl.SetItemText( nItem, 2, strSex );
    m_ListCtrl.SetItemText( nItem, 3, dlg.m_tBirth.Format("%Y-%m-%d") );
    m_ListCtrl.SetItemText( nItem, 4, dlg.m_strSpecial);
}
```

③ 编译运行并测试。

6. 实现打印和打印预览

① 为 CEx_StudentView 类添加一个成员变量 m_strContents, 类型为 CStringArray。用来保存读取列表项的数据。

② 为 CEx_StudentView 类添加一个成员函数 ReadListViewData, 用于读取列表项的数据, 其代码如下:

```
void CEx_StudentView::ReadListViewData()
{
    m_strContents.RemoveAll();
    CListCtrl& m_ListCtrl = GetListCtrl();
    int nCount = 0;
    CHeaderCtrl* pHeaderCtrl = m_ListCtrl.GetHeaderCtrl();
    if ( pHeaderCtrl!= NULL)    nCount = pHeaderCtrl->GetItemCount();
    // 读取表头信息作为第一行
    LVCOLUMN    column;
    TCHAR        lpBuffer[256];
    int          nWidthCharNum[100];
    CString str, strTemp;
    for (int nCol = 0; nCol < nCount; nCol++)    {
        column.mask            = LVCF_TEXT;
        column.pszText         = lpBuffer;
        column.cchTextMax      = 256;
        m_ListCtrl.GetColumn( nCol, &column );
        nWidthCharNum[nCol] = (int)(m_ListCtrl.GetColumnWidth( nCol )/6.5);
        strTemp.Format("%s", lpBuffer);
        strTemp.TrimRight();
        int nDelta = nWidthCharNum[nCol] - strTemp.GetLength();
        if (nDelta>0)    strTemp = AppendStringSpace( strTemp, nDelta);
        else    strTemp = strTemp.Left(nWidthCharNum[nCol]);
        str = str + strTemp;
    }
    str.TrimRight();
    m_strContents.Add( str );
    // 读取各个列表项信息
    int nItemCount = m_ListCtrl.GetItemCount();
    for ( int nItem=0; nItem<nItemCount; nItem++)    {
        str.Empty();
        strTemp.Empty();
        for (int i=0; i<nCount; i++)    {
            strTemp = m_ListCtrl.GetItemText(nItem, i);
            strTemp.TrimRight();
            int nDelta = nWidthCharNum[i] - strTemp.GetLength();
            if (nDelta>0)    strTemp = AppendStringSpace( strTemp, nDelta);
            else    strTemp = strTemp.Left(nWidthCharNum[i]);
            str = str + strTemp;
        }
    }
}
```

```

    }
    str.TrimRight();
    m_strContents.Add( str );
}
}

```

③ 为 CEx_StudentView 类添加一个成员函数 AppendStringSpace，用来在一个字符串后添加空格，其代码如下：

```

CString CEx_StudentView::AppendStringSpace(CString str, int nNum)
{
    for (int i=0; i<nNum; i++)    str.Insert( 500, ' ');
    // 通常500大于一个字符串长度，当大于字符串长度，则Insert在字符串末尾添加
    return str;
}

```

④ 打开 Ex_StudentView.h 文件，在 class CEx_StudentView : public CListView 语句前添加 PAGEINFO 结构的定义，具体代码如下：

```

typedef struct {
    CSize sizePage;           // 页面/纸大小
    CSize sizeLine;           // 每行的大小
    CSize sizeChar;           // 每个字符的平均大小
    int nLMargin;             // 左边距
    int nRMargin;             // 右边距
    int nTMargin;             // 上边距
    int nBMargin;             // 下边距
    int nPhyLeft;             // 物理左边距
    int nPhyRight;            // 物理右边距
    int nPhyTop;              // 物理上边距
    int nPhyBottom;           // 物理下边距
    LOGFONT lfHead;           // 页眉字体
    LOGFONT lfFoot;           // 页脚字体
    LOGFONT lfText;           // 正文字体
} PAGEINFO;                  // 页面信息结构

```

⑤ 为 CEx_StudentView 类添加下列成员变量：

public:

```

PAGEINFO    m_pageInfo;
CStringArray m_arrText;           // 处理后的文档内容
CUIntArray  m_nLineArray;        // 记录每页的开始行号

```

⑥ 为 CEx_StudentView 类添加成员函数 SetPageInfo、PrintHead、PrintFoot 和 AdjustAllLine，具体代码如下：

```

void CEx_StudentView::SetPageInfo(CDC *pDC, CPrintInfo *pInfo, PAGEINFO *pPage,

```

```

        int l, int t, int r, int b, int nLineSpace)
// nLineSpace为行间距, l,t,r,b分别表示左上右下的页边距
{
    // 计算一个设备单位等于多少0.1mm
    double scaleX = 254.0/(double)GetDeviceCaps(pDC->m_hAttribDC, LOGPIXELSX);
    double scaleY = 254.0/(double)GetDeviceCaps(pDC->m_hAttribDC, LOGPIXELSY);
    int x = GetDeviceCaps(pDC->m_hAttribDC, PHYSICALOFFSETX);
    int y = GetDeviceCaps(pDC->m_hAttribDC, PHYSICALOFFSETY);
    int w = GetDeviceCaps(pDC->m_hAttribDC, PHYSICALWIDTH);
    int h = GetDeviceCaps(pDC->m_hAttribDC, PHYSICALHEIGHT);
    int nPageWidth = (int)((double)w*scaleX + 0.5);           // 纸宽, 单位0.1mm
    int nPageHeight = (int)((double)h*scaleY + 0.5);          // 纸高, 单位0.1mm
    int nPhyLeft = (int)((double)x*scaleX + 0.5);             // 物理左边距, 单位0.1mm
    int nPhyTop = (int)((double)y*scaleY + 0.5);              // 物理上边距, 单位0.1mm
    CRect rcTemp = pInfo->m_rectDraw;
    rcTemp.NormalizeRect();
    int nPhyRight = nPageWidth - rcTemp.Width() - nPhyLeft;   // 物理右边距, 单位0.1mm
    int nPhyBottom = nPageHeight - rcTemp.Height() - nPhyTop; // 物理下边距, 单位0.1mm
    // 若边距小于物理边距, 则调整它们
    if (l < nPhyLeft) l = nPhyLeft;    if (t < nPhyTop) t = nPhyTop;
    if (r < nPhyRight) r = nPhyRight;  if (b < nPhyBottom) b = nPhyBottom;
    pPage->nLMargin = l; pPage->nRMargin = r;
    pPage->nTMargin = t; pPage->nBMargin = b;
    pPage->nPhyLeft = nPhyLeft;
    pPage->nPhyRight = nPhyRight;
    pPage->nPhyTop = nPhyTop;
    pPage->nPhyBottom = nPhyBottom;
    pPage->sizePage = CSize(nPageWidth, nPageHeight);
    // 计算并调整pInfo->m_rectDraw的大小
    pInfo->m_rectDraw.left = l - nPhyLeft;
    pInfo->m_rectDraw.top = - t + nPhyTop;
    pInfo->m_rectDraw.right -= r - nPhyRight;
    pInfo->m_rectDraw.bottom += b - nPhyBottom;
    // 计算字符的大小
    pPage->sizeChar = pDC->GetTextExtent("G");
    // 计算行的大小
    pPage->sizeLine = CSize(pInfo->m_rectDraw.Width(), pPage->sizeChar.cy + nLineSpace);
}

void CEx_StudentView::PrintHead(CDC* pDC, CPrintInfo* pInfo, PAGEINFO page,
                                CString title, int margin, int mode)
// mode表示页眉文本对齐模式, 0为居中, >0表示右对齐, <0表示左对齐
// title表示页眉内容, margin为页眉与顶边的距离
{
    CFont font;
    font.CreateFontIndirect(&page.lfHead);

```

```

CFont* oldFont = pDC->SelectObject(&font);
CSize strSize = pDC->GetTextExtent(title);
CRect rc = pInfo->m_rectDraw;
CPoint pt;
margin = margin - page.nPhyTop;
if (margin < 0) margin = 0;
// 根据mode计算绘制页眉文本的起点
if (mode < 0) pt = CPoint(rc.left, -margin);
if (mode == 0) pt = CPoint(rc.CenterPoint().x - strSize.cx/2, -margin);
if (mode > 0) pt = CPoint(rc.right - strSize.cx, -margin);
pDC->TextOut(pt.x, pt.y, title);           // 绘制页眉文本
pt.y -= strSize.cy + 5;
pDC->MoveTo(rc.left, pt.y);               // 画线
pDC->LineTo(rc.right, pt.y);
pt.y -= 10;
int absY = pt.y > 0 ? pt.y : -pt.y;
if (absY > page.nTMargin) pInfo->m_rectDraw.top = pt.y;
pDC->SelectObject(oldFont);
font.DeleteObject();

```

```

}
void CEx_StudentView::PrintFoot(CDC* pDC, CPrintInfo* pInfo, PAGEINFO page,
                                CString title, int margin, int mode)
// mode表示页脚文本对齐模式，0为居中，>0表示右对齐，<0表示左对齐
// title表示页脚内容， margin为页脚与底边的距离

```

```

{
    CFont font;
    font.CreateFontIndirect(&page.lfFoot);
    CFont* oldFont = pDC->SelectObject(&font);
    CSize strSize = pDC->GetTextExtent(title);
    CRect rc = pInfo->m_rectDraw;
    CPoint pt;
    margin = page.nBMargin - margin - strSize.cy;
    // 根据mode计算绘制页眉文本的起点
    if (mode < 0) pt = CPoint(rc.left, rc.bottom - margin);
    if (mode == 0) pt = CPoint(rc.CenterPoint().x - strSize.cx/2, rc.bottom - margin);
    if (mode > 0) pt = CPoint(rc.right - strSize.cx, rc.bottom - margin);
    pDC->TextOut(pt.x, pt.y, title);           // 绘制页脚文本
    if (margin < 0)
        pInfo->m_rectDraw.bottom -= margin;
    pDC->SelectObject(oldFont);

```

```

}
void CEx_StudentView::AdjustAllLine(CDC *pDC)

```

```

{
    CEx_PrintDoc* pDoc = GetDocument();
    int nLineNums = pDoc->m_strContents.GetSize();    // 文档总行数

```

```

int tab = m_pageInfo.sizeChar.cx * 4;           // 为一个TAB设置4个字符
// 将文档的每一行作换行处理，只处理一次，以提高预览速度
// 处理的结果保存在arrText中
CString str;
if (pDoc->m_bNewDocument) {
    m_arrText.RemoveAll();
    CSize strSize;
    for (int i=0; i<nLineNums; i++) {
        str = pDoc->m_strContents.GetAt(i);
        strSize = pDC->GetTabbedTextExtent(str, 1, &tab);
        CString strTemp = str;
        while ( strSize.cx > m_pageInfo.sizeLine.cx) {
            unsigned int pos = 0;
            for (pos = 0; pos<strlen(strTemp); pos++) {
                CSize size = pDC->GetTabbedTextExtent(strTemp, pos+1, 1, &tab);
                if (size.cx >= m_pageInfo.sizeLine.cx) break;
            }
            // 判断汉字双字符是否被分开
            int nCharHZ = 0;
            for (unsigned int chIndex = 0; chIndex <= pos; chIndex++)
                if (strTemp.GetAt(chIndex) < 0) nCharHZ++;
            if (nCharHZ % 2) pos = pos - 1;
            m_arrText.Add(strTemp.Left(pos+1));
            strTemp = strTemp.Mid(pos+1);
            strSize = pDC->GetTabbedTextExtent(strTemp, 1, &tab);
        }
        m_arrText.Add(strTemp);
    }
    pDoc->m_bNewDocument = FALSE;
    m_nLineArray.RemoveAll();
    m_nLineArray.Add(0);
}
}

```

⑦ 在 CEx_StudentView 类的构造函数处，添加下列初始化代码：

```

CEx_StudentView::CEx_StudentView()
{
    memset(&m_pageInfo, 0, sizeof(m_pageInfo));           // 所有成员置为0
    double fontScale = 254.0/72.0;    // 一个点相当于多少0.1mm
    // 页眉字体
    m_pageInfo.lfHead.lfHeight = - (int)(9 * fontScale + 0.5);    // 9号字
    m_pageInfo.lfHead.lfWeight = FW_BOLD;
    m_pageInfo.lfHead.lfCharSet = GB2312_CHARSET;
    strcpy((LPSTR)&(m_pageInfo.lfHead.lfFaceName), "黑体");
    // 页脚字体
}

```

```

        m_pageInfo.lfFoot.lfHeight = - (int)(9 * fontScale + 0.5);    // 9号字
        m_pageInfo.lfFoot.lfWeight = FW_NORMAL;
        m_pageInfo.lfFoot.lfCharSet = GB2312_CHARSET;
        strcpy((LPSTR)&(m_pageInfo.lfFoot.lfFaceName), "楷体_GB2312");
        // 正文字体
        m_pageInfo.lfText.lfHeight = - (int)(11 * fontScale + 0.5);    // 11号字
        m_pageInfo.lfText.lfWeight = FW_NORMAL;
        m_pageInfo.lfText.lfCharSet = GB2312_CHARSET;
        strcpy((LPSTR)&(m_pageInfo.lfText.lfFaceName), "宋体");
    }

```

⑧ 在 CEx_StudentView::OnPreparePrinting 中设置当内容为空时的最大打印页数:

```

BOOL CEx_StudentView::OnPreparePrinting(CPrintInfo* pInfo)

```

```

{
    ReadListViewData();
    int nSize = m_strContents.GetSize();
    if (nSize < 1) pInfo->SetMaxPage(1);
    return DoPreparePrinting(pInfo);
}

```

- 用 MFC ClassWizard 为 CEx_StudentView 类 OnPrepareDC 函数的重载, 并添加设置映射模式和多页打印的代码:

```

void CEx_StudentView::OnPrepareDC(CDC* pDC, CPrintInfo* pInfo)

```

```

{
    ReadListViewData();
    pDC->SetMapMode(MM_LOMETRIC);    // 单位0.1mm
    CListView::OnPrepareDC(pDC, pInfo);
    int nSize = m_strContents.GetSize();
    if ((pInfo)&&(nSize)){
        if (pInfo->m_nCurPage <= pInfo->GetToPage())
            pInfo->m_bContinuePrinting = TRUE;
        else
            pInfo->m_bContinuePrinting = FALSE;
    }
}

```

- 用 MFC ClassWizard 为 CEx_StudentView 类增加 OnPrint 函数的重载, 并添加下列代码:

```

void CEx_StudentView::OnPrint(CDC* pDC, CPrintInfo* pInfo)

```

```

{
    CFont font;
    font.CreateFontIndirect(&m_pageInfo.lfText);
    CFont* oldFont = pDC->SelectObject(&font);    // 设置正文字体
    SetPageInfo(pDC, pInfo, &m_pageInfo, 250, 250, 250, 250, 35); // 设置页边距和行距
}

```



```

PrintHead(pDC, pInfo, m_pageInfo, "学生信息管理系统, 设计者: 丁有和", 200, -1); // 打印页眉
CString str;
str.Format("- %d -", pInfo->m_nCurPage);
PrintFoot(pDC, pInfo, m_pageInfo, str, 200, 0); // 打印页脚
if (m_strContents.GetSize() < 1) return; // 没有文档内容则返回
AdjustAllLine(pDC); // 调整每行的文本
int nIndex = pInfo->m_nCurPage - 1;
int nStartLine = m_nLineArray.GetAt(nIndex);
CRect rc = pInfo->m_rectDraw;
int y = rc.top;
int nHeight = m_pageInfo.sizeLine.cy;
int tab = m_pageInfo.sizeChar.cx * 4; // 为一个TAB设置4个字符
while (y >= pInfo->m_rectDraw.bottom)
{
    str = m_arrText.GetAt(nStartLine);
    rc.top = y;
    pDC->TabbedTextOut(rc.left, y, str, 1, &tab, rc.left);
    nStartLine++;
    if (nStartLine >= m_arrText.GetSize()) {
        pInfo->SetMaxPage(pInfo->m_nCurPage);
        pInfo->m_pPD->m_pd.nToPage = pInfo->m_nCurPage;
        break;
    }
    y -= nHeight;
}
if (nIndex >= (m_nLineArray.GetSize() - 1))
    m_nLineArray.Add(nStartLine); // 保存下一页的起始行号
pDC->SelectObject(oldFont);
CListView::OnPrint(pDC, pInfo);
}

```

⑨ 编译运行并测试。

7. 写出实验报告

分析上述运行结果以及思考与练习，写出实验报告。

思考与练习

创建一个对话框用于显示一个班级某门课程成绩的分布直方图。