**程序设计课程设计实验报告**

****

**学院： 计算机科学与教育软件学院**

**专业班级： 软件工程 174**

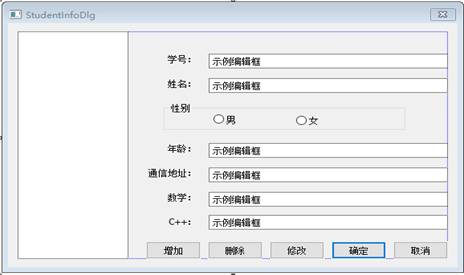
**姓名： 黎永杰**

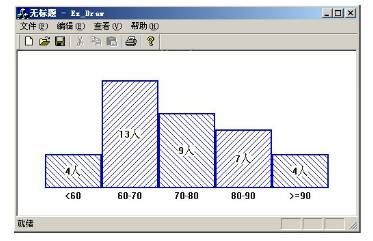
**学号： 1706300154**

**2018.12.24**

**一、课程设计题目**

应用MFC应用程序实现学生信息管理系统

（1）、设计一个学生类Student,包括数据成员：姓名、学号、二门课程(面向对象程序设计、高等数学)的成绩。（2）、创建一个管理学生的类Management，包括实现学生的数据的增加、删除、修改、按课程成绩排序、保存学生数据到文件及加载文件中的数据等功能。（3）、创建一个基于对话框的MFC应用程序，程序窗口的标题上有你姓名、学号和应用程序名称。使用（1）和（2）中的类，实现对学生信息和成绩的输入和管理。

（4）、创建一个单文档的MFC应用程序，读取（3）中保存的文件中的学生成绩，分别用直方图和折线方式显示所有学生某课程的成绩分布图。

**二、程序设计基本步骤**

1.分析需求，了解程序所需的功能；

2.根据所需功能构思、设计相应算法；

3.创建工程，添加所需类、变量、函数等；

4.编写程序；

5.调试、测试；

6.生成、导出程序；

**三、自定义类及函数功能概括**

//学生管理系统

//StuMana.exe

Class Student{}; //储存学生信息

Class Management{}; //管理学生信息

bool Management::AddManaAdd(CString str\_Name, CString str\_Num, CString str\_Sex, CString str\_Math, CString str\_Cpp, int i\_KeyInd); //添加学生信息

int Management::FloorFull(Tree\* p\_Full);

//当节点key已满，执行该函数以分裂节点

Tree\* Management::Search(Tree\* p\_Root, int i\_fl, int key);

//查找该学生信息所在叶节点

Student\* ManaSearch(int i\_key); //查找并获取该学生地址

bool ManaDel(int i\_DelKey, int i\_DelInd, int KeyInd);

//删除学生信息

bool Unite(Tree\* p\_Unite);

//当删除操作后若节点、及其兄弟节点需要合并，执行该函数

Class About{}; //About窗口类

Class AddInfoDlg{}; //添加信息窗口类

afx\_msg void AddInfoDlg ::OnBnClickedExit();

//添加并关闭的消息处理程序

afx\_msg void AddInfoDlg ::OnBnClickedCancel();

//取消按钮的消息处理程序

afx\_msg void AddInfoDlg ::OnBnClickedEmpty();

//清空按钮的消息处理程序

afx\_msg void AddInfoDlg ::OnBnClickedAdd();

//添加按钮的消息处理程序

bool AddInfoDlg ::CheckInfo(); //检查信息是否填写规范

Class AddWaring{}; //添加时警告对话框类

afx\_msg void AddWaring ::OnBnClickedCancel(); //取消按钮消息处理程序

afx\_msg void AddWaring OnBnClickedContinue(); //继续按钮消息处理程序

Class AltDlg{}; //修改对话框类

afx\_msg void AltDlg ::OnBnClickedAlter(); //修改按钮消息处理程序

afx\_msg void AltDlg ::OnBnClickedCamcel(); //取消按钮消息处理程序

virtual BOOL AltDlg ::OnInitDialog(); //在此函数添加额外初始化

Class RankDlg{}; //排序对话框类

virtual BOOL RankDlg ::OnInitDialog(); //此函数添加额外初始化

afx\_msg void RankDlg ::OnBnClickedRank(); //排序按钮消息处理程序

afx\_msg void RankDlg ::OnBnClickedCancel(); //取消按钮消息处理程序

Class SearchDlg{}; //查找对话框类

afx\_msg void SearchDlg ::OnBnClickedOnSearch(); //查找按钮消息处理

afx\_msg void SearchDlg ::OnBnClickedSearchNext(); //查找下一个按钮

virtual BOOL SearchDlg ::OnInitDialog(); //添加额外的初始化

Class StuManaDlg{}; //主窗口

afx\_msg void StuManaDlg ::OnBnClickedSearch(); //主对话框查找按钮

afx\_msg void StuManaDlg ::OnBnClickedDel(); //主对话框删除按钮

afx\_msg void StuManaDlg ::OnBnClickedRank(); //主对话框排序按钮

afx\_msg void StuManaDlg ::OnBnClickedAlter(); //主对话框修改按钮

afx\_msg void StuManaDlg ::OnBnClickedPic(); //主对话框绘图按钮

afx\_msg void StuManaDlg ::OnBnClickedSave(); //主对话框保存按钮

afx\_msg void StuManaDlg ::OnBnClickedLoad(); //主对话框打开按钮

afx\_msg void StuManaDlg ::OnBnClickedAbout(); //主对话框作者按钮

afx\_msg void StuManaDlg ::OnBnClickedOk(); //保存并关闭按钮

void CStuManaDlg::OnBnClickedReStart() //重置

INT\_PTR StuMana ::OnSave(); //储存信息到文件

Class Tree{}; //B+树类，以B+树储存学生信息

int Tree ::AddKey(int i\_AddKey); //添加关键字

void Tree ::Write(int index, int key, Tree\* node, Tree\* former = NULL);

//写入信息到叶节点对应位置

void Tree ::Del(int index); //删除某节点内某组数据

//绘图程序

//StuPic.exe

Class MainFrm{}; //框架

afx\_msg void OnFileOpen(); //打开文件并读取

afx\_msg void OnMathHis(); //绘制数学成绩直方图

afx\_msg void OnMathLine(); //绘制数学成绩折线图

afx\_msg void OnCppHis(); //绘制程设成绩直方图

afx\_msg void OnCppLine(); //绘制程设成绩折线图

afx\_msg void OnUpdateMathHis(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateMathLine(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateCppHis(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateCppLine(CCmdUI \*pCmdUI);

//以上四个函数均为 禁用/启用 菜单内选项

void GetCount(int i\_Sub); //获取各分段人数

void DrawHis(bool b\_Sub); //绘制直方图

void DrawLine(bool b\_Sub); //绘制折线图

void ClearDraw(); //清除

afx\_msg void OnViewOMH();

afx\_msg void OnViewOCH();

afx\_msg void OnViewOCL();

afx\_msg void OnViewOML();

//以上四个函数均为工具栏中四个绘图按钮所对应函数

afx\_msg void OnUpdateViewMl(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewMh(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewCl(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewCh(CCmdUI \*pCmdUI);

//以上四个函数均为 禁用/启用 工具栏选项

**四、实验代码**

//Student.h

#pragma once

#include "Tree.h"

class Student

:public Tree

{

public:

Student();

~Student();

CString str\_Name; //姓名

CString str\_Math; //数学

CString str\_Sex; //性别

CString str\_Num; //学号

CString str\_Cpp; //程设

Student\* p\_Same; //用于指向相同数据特征的学生

int i\_Rank;

};

//Student.cpp

#include "stdafx.h"

#include "Student.h"

Student::Student()

: str\_Name(\_T(""))

, str\_Math(\_T(""))

, str\_Sex(\_T(""))

, str\_Num(\_T(""))

, str\_Cpp(\_T(""))

, i\_Rank(0)

, p\_Same(NULL)

{

}

Student::~Student()

{

}

//Management.h

#pragma once

#include "Student.h"

#include "Tree.h"

class Management

{

public:

Management();

~Management();

int i\_Total; //记录总数

int i\_floor; //记录树的层数

Tree m\_TreeHead; //头指针

Tree \*p\_TreeThis; //操作指针

Tree \*p\_TreeRoot; //树根指针

bool ManaAdd(CString str\_Name, CString str\_Num, CString str\_Sex, CString str\_Math, CString str\_Cpp, int i\_KeyInd); //向链表添加学生信息

int FloorFull(Tree\* p\_Full); //节点已满时的处理函数

Tree\* Search(Tree\* p\_Root, int i\_fl, int key); //查找叶节点函数

Student\* ManaSearch(int i\_key); //查找学生函数

bool ManaDel(int i\_DelKey, int i\_DelInd, int KeyInd); //删除学生函数

//bool Unite(Tree\* p\_Unite); //节点合并函数（功能未通过测试）

};

//Management.cpp

#include "stdafx.h"

#include "Management.h"

Management::Management() //初始化各个变量

: p\_TreeThis(&m\_TreeHead)

, p\_TreeRoot(&m\_TreeHead)

, i\_floor(0)

, i\_Total(0)

{

m\_TreeHead.p\_Node[0] = &m\_TreeHead;

}

Management::~Management()

{

}

bool Management::ManaAdd(CString str\_Name, CString str\_Num, CString str\_Sex, CString str\_Math, CString str\_Cpp, int i\_KeyInd)

{

//本函数用于向树添加学生信息

i\_Total++;

int i\_AddKey;

Tree\* p\_ThisBefo = NULL;

switch (i\_KeyInd) //将CString转为int

{

case 0:

{

int i\_Len = str\_Name.GetLength();

if (i\_Len > 0)

{

i\_AddKey = str\_Name[0] - 19000;

i\_AddKey = i\_AddKey \* 100000;

if (i\_Len > 1)

i\_AddKey = i\_AddKey + str\_Name[1] - 19000;

}

else return 0;

}; break;

case 1:i\_AddKey = \_ttoi(str\_Num); break;

case 2:i\_AddKey = \_ttoi(str\_Math); break;

case 3:i\_AddKey = \_ttoi(str\_Cpp); break;

}

int i\_Index = 0;

bool b\_same = 1;

//查找叶节点

p\_TreeThis = Search(p\_TreeRoot, i\_floor, i\_AddKey);

for (; i\_Index < p\_TreeThis->i\_count; i\_Index++)

{

//查找叶节点内学生信息

int i\_same;

switch (i\_KeyInd)

{

case 0:

{

CString Name = ((Student\*)(p\_TreeThis->p\_Node[i\_Index]))->str\_Name;

int i\_Len = Name.GetLength();

if (i\_Len > 0)

{

i\_same = Name[0] - 19000;

i\_same = i\_same \* 100000;

if (i\_Len > 1)

i\_same = i\_same + Name[1] - 19000;

}

}; break;

case 1:i\_same = \_ttoi(((Student\*)(p\_TreeThis->p\_Node[i\_Index]))->str\_Num); break;

case 2:i\_same = \_ttoi(((Student\*)(p\_TreeThis->p\_Node[i\_Index]))->str\_Math); break;

case 3:i\_same = \_ttoi(((Student\*)(p\_TreeThis->p\_Node[i\_Index]))->str\_Cpp); break;

}

if (i\_same == i\_AddKey)

{

//若找到相同的key，则判断是否有多个

//该key对应的信息，并获取正确的存放位置

p\_ThisBefo = (Student\*)(p\_TreeThis->p\_Node[i\_Index]);

p\_TreeThis = ((Student\*)(p\_TreeThis->p\_Node[i\_Index]))->p\_Same;

while (p\_TreeThis != NULL)

{

p\_ThisBefo = p\_TreeThis;

p\_TreeThis = ((Student\*)p\_TreeThis)->p\_Same;

}

b\_same = 0;

break;

}

}

if (b\_same)

{ //判断叶节点是否已满，已满则分裂节点

if (p\_TreeThis->i\_count == 4)

{

FloorFull(p\_TreeThis);

p\_TreeThis = p\_TreeThis->p\_Former;

p\_TreeThis = Search(p\_TreeThis, 1, i\_AddKey);

if (p\_TreeThis == NULL)

p\_TreeThis = new Tree;

}

}

//添加学生信息至叶节点内

if (b\_same)

{

i\_Index = p\_TreeThis->AddKey(i\_AddKey);

p\_TreeThis->p\_Node[i\_Index] = new Student;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->str\_Name = str\_Name;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->str\_Math = str\_Math;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->str\_Sex = str\_Sex;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->str\_Num = str\_Num;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->str\_Cpp = str\_Cpp;

((Student\*)p\_TreeThis->p\_Node[i\_Index])->i\_Rank = i\_Total;

}

else

{

p\_TreeThis = new Student;

((Student\*)p\_ThisBefo)->p\_Same = (Student\*)p\_TreeThis;

((Student\*)p\_TreeThis)->str\_Name = str\_Name;

((Student\*)p\_TreeThis)->str\_Math = str\_Math;

((Student\*)p\_TreeThis)->str\_Sex = str\_Sex;

((Student\*)p\_TreeThis)->str\_Num = str\_Num;

((Student\*)p\_TreeThis)->str\_Cpp = str\_Cpp;

((Student\*)p\_TreeThis)->i\_Rank = i\_Total;

}

return 1;

}

int Management::FloorFull(Tree\* p\_Full)

{ //本函数用于节点已满时进行分裂操作

Tree\* p\_This; //操作指针

int i\_Index = 0; //索引值

if (p\_Full->p\_Former != NULL) //若上一节点不为空

{

p\_This = p\_Full->p\_Former; //操作指针移至上一节点

if (p\_This->i\_count == 4) //若上一节点已满

{

int i = 0, n = 0;

for (; p\_Full != p\_This->p\_Node[i]; i++);

n = FloorFull(p\_This); //则递归本函数

if (i > 3)

p\_This = p\_This->p\_Former->p\_Node[n];

p\_Full->p\_Former = p\_This;

}

i\_Index = p\_This->AddKey(p\_Full->i\_key[2]) + 1; //向操作节点添加新关键字并获取索引值

p\_This->p\_Node[i\_Index - 1] = p\_This->p\_Node[i\_Index];

}

else

{

i\_floor++;

p\_This = new Tree; //建立新节点作为根节点

p\_This->i\_onfl = i\_floor;

p\_TreeRoot = p\_This; //根节点指针指向新的根节点

p\_Full->p\_Former = p\_This; //已满节点的上一节点指针指向新节点

i\_Index = p\_This->AddKey(p\_Full->i\_key[2]) + 1; //添加新关键字

p\_This->p\_Node[0] = p\_Full; //将新节点首位指针指向原节点

}

p\_This->p\_Node[i\_Index] = new Tree; //添加新子节点

p\_This = p\_This->p\_Node[i\_Index]; //操作指针指向新节点

p\_This->p\_Former = p\_Full->p\_Former; //写入上一节点

p\_This->i\_onfl = p\_Full->i\_onfl;

if (p\_Full->i\_onfl)

{

p\_This->Write(0, p\_Full->i\_key[3], p\_Full->p\_Node[3], p\_Full->p\_Former); //将已满节点的后两组数据写入新节点

p\_This->p\_Node[1] = p\_Full->p\_Node[4];

Tree\* p\_Stay = p\_Full->p\_Node[2];

p\_Full->Del(2);

p\_Full->p\_Node[2] = p\_Stay;

}

else

{

p\_This->p\_Node[2] = p\_Full->p\_Node[4]; //将指向下一节点指针写入新节点

p\_This->Write(0, p\_Full->i\_key[2], p\_Full->p\_Node[2]);

p\_This->Write(1, p\_Full->i\_key[3], p\_Full->p\_Node[3]);

p\_Full->Del(2);

p\_Full->p\_Node[2] = p\_This;

}

p\_Full->Del(3); //删除已满节点后两组数据

p\_Full->Del(4);

return i\_Index;

}

Tree\* Management::Search(Tree\* p\_Root, int i\_fl, int key)

{ //本函数用于查找key所在叶节点

//判断是否已在叶节点

if (i\_fl == 0)

return p\_Root;

Tree\* p\_This;

p\_This = p\_Root;

int i = 0;

int i\_flag = 1;

for (int fl = i\_fl; fl > 0; fl--)

{ //逐层查找

i\_flag = 1;

for (i = 0; i < p\_This->i\_count; i++)

{

if (key < p\_This->i\_key[i])

{

p\_This = p\_This->p\_Node[i];

i\_flag = 0;

break;

}

}

if (i\_flag)

p\_This = p\_This->p\_Node[p\_This->i\_count];

}

return p\_This;

}

Student\* Management::ManaSearch(int i\_key)

{ //查找key所对应的具体地址

Tree\* p\_Search;

//先查找key所在叶节点

p\_Search = Search(p\_TreeRoot, i\_floor, i\_key);

if (p\_Search != NULL)

{ //若已找到叶节点，则查找具体地址

for (int i = 0; i < p\_Search->i\_count; i++)

{

if (p\_Search->i\_key[i] == i\_key)

return (Student\*)p\_Search->p\_Node[i];

}

return NULL;

}

return NULL;

}

bool Management::ManaDel(int i\_DelKey, int i\_DelInd, int i\_KedInd)

{ //本函数用于删除key及其对应信息

Tree\* p\_Del;

int index = 0;

//先查找key所在叶节点

p\_Del = Search(p\_TreeRoot, i\_floor, i\_DelKey);

if (p\_Del != NULL)

{

bool flag = 1;

for(; index < p\_Del->i\_count; index++) //查找该叶节点

{

int i\_ThisDel;

switch (i\_KedInd)

{

case 0:

{

CString str\_Del = ((Student\*)(p\_Del->p\_Node[index]))->str\_Name;

int i\_Len = str\_Del.GetLength();

if (i\_Len > 0)

{

i\_ThisDel = str\_Del[0] - 19000;

i\_ThisDel = i\_ThisDel \* 100000;

if (i\_Len > 1)

i\_ThisDel = i\_ThisDel + str\_Del[1] - 19000;

}

}; break;

case 1:i\_ThisDel = \_ttoi(((Student\*)(p\_Del->p\_Node[index]))->str\_Num); break;

case 2:i\_ThisDel = \_ttoi(((Student\*)(p\_Del->p\_Node[index]))->str\_Math); break;

case 3:i\_ThisDel = \_ttoi(((Student\*)(p\_Del->p\_Node[index]))->str\_Cpp); break;

}

if (i\_ThisDel == i\_DelKey) //若已经找到相同的key

{

flag = 0;

if (((Student\*)(p\_Del->p\_Node[index]))->i\_Rank - 1 == i\_DelInd) //判断是否为选中行所对应信息

{

if (((Student\*)(p\_Del->p\_Node[index]))->p\_Same) //判断该key是否有多个信息

{

Tree\* p\_DelSame= p\_Del->p\_Node[index];

p\_Del->p\_Node[index] = ((Student\*)(p\_Del->p\_Node[index]))->p\_Same;

delete p\_DelSame;

}

else delete p\_Del->p\_Node[index]; //没有多个信息则直接删除

}

else

{ //不是选中行所对应信息

Tree\* p\_DelSame = p\_Del->p\_Node[index];

Tree\* p\_DelBefo = NULL;

while (p\_DelSame)

{ //查找相同key的其它信息

p\_DelBefo = p\_DelSame;

if (((Student\*)(p\_DelSame))->i\_Rank - 1 != i\_DelInd)

p\_DelSame = ((Student\*)p\_DelSame)->p\_Same;

else break;

}

if (p\_DelSame != NULL)

{

((Student\*)p\_DelBefo)->p\_Same = ((Student\*)p\_DelSame)->p\_Same;

delete p\_DelSame;

}

else return 0;

}

break;

}

}

if (flag)

return 0;

}

for (; index < p\_Del->i\_count; index++)

{ //整理节点内信息

p\_Del->i\_key[index] = p\_Del->i\_key[index + 1];

p\_Del->p\_Node[index] = p\_Del->p\_Node[index + 1];

}

p\_Del->p\_Node[p\_Del->i\_count] = p\_Del->p\_Node[p\_Del->i\_count + 1];

p\_Del->p\_Node[p\_Del->i\_count + 1] = NULL;

p\_Del->i\_count--;

if (p\_Del->i\_count >= 2 || p\_Del->p\_Former == NULL)

return 1;

////如果该节点剩余key的个数大于等于2，或无父节点则返回1，否则继续执行以下代码

//Unite(p\_Del);

////此处未通过测试

return 1;

}

/\*bool Management::Unite(Tree\* p\_Unite)

{

int index = 0;

Tree\* p\_Brother;

if (p\_Unite->p\_Former == NULL)

return 0;

Tree\* p\_Former = p\_Unite->p\_Former;

for (; index < p\_Unite->p\_Former->i\_count; index++)

{

if (p\_Unite->i\_key[0] < p\_Former->i\_key[index])

break;

}

index--;

if (index < 0)

return 0;

p\_Brother = p\_Former->p\_Node[index];

if (p\_Brother->i\_count > 2)

{

//如果兄弟节点的key数量富余，则将最后一个key转移

int key = p\_Brother->i\_key[p\_Brother->i\_count - 1];

p\_Former->i\_key[index] = key;

p\_Unite->AddKey(key);

if (p\_Brother->i\_onfl)

p\_Unite->p\_Node[0] = p\_Brother->p\_Node[p\_Brother->i\_count];

else

{

p\_Unite->p\_Node[0] = p\_Brother->p\_Node[p\_Brother->i\_count - 1];

p\_Brother->p\_Node[p\_Brother->i\_count - 1] = p\_Brother->p\_Node[p\_Brother->i\_count];

}

p\_Brother->p\_Node[p\_Brother->i\_count] = NULL;

p\_Brother->i\_key[p\_Brother->i\_count - 1] = 0;

p\_Brother->i\_count--;

}

else

{

//如果兄弟节点的key不足，则合并

int sum = p\_Brother->i\_count + p\_Unite->i\_count;

p\_Brother->p\_Node[sum] = p\_Unite->p\_Node[p\_Unite->i\_count];

for (int i = p\_Brother->i\_count; i < sum; i++)

p\_Brother->Write(i, p\_Unite->i\_key[i - p\_Brother->i\_count], p\_Unite->p\_Node[i - p\_Brother->i\_count]);

p\_Former->i\_count--;

for (int i = index; i < p\_Former->i\_count - 1; i++)

{

p\_Former->p\_Node[i] = p\_Former->p\_Node[i + 1];

p\_Former->i\_key[i] = p\_Former->i\_key[i + 1];

}

p\_Former->p\_Node[p\_Former->i\_count] = p\_Former->p\_Node[p\_Former->i\_count + 1];

p\_Former->p\_Node[p\_Former->i\_count + 1] = NULL;

p\_Former->i\_key[p\_Former->i\_count] = 0;

if (p\_Former->i\_count < 2)

Unite(p\_Former);

delete p\_Unite;

}

return 1;

//本函数用于节点、及其兄弟节点内均key的数量不足时的合并处理，但由于考虑欠佳，也未能及时排除问题，限于时间精力暂时停用该函数；

}\*/

//About.cpp

//此类用于弹出“关于”窗口，该窗口展示作者信息(static)及程序图标;

//AddInfoDlg.h

#pragma once

#include "AddWarning.h"

#define WM\_ADDINFO WM\_USER+1

// AddInfoDlg 对话框

class AddInfoDlg : public CDialogEx

{

DECLARE\_DYNAMIC(AddInfoDlg)

public:

AddInfoDlg(CWnd\* pParent = nullptr); // 标准构造函数

virtual ~AddInfoDlg();

// 对话框数据

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_DIALOG1 };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

DECLARE\_MESSAGE\_MAP()

public:

CString str\_Name; //姓名

CString str\_Num; //学号

CString str\_Math; //数学

CString str\_Sex; //性别

CString str\_Cpp; //程序设计

virtual BOOL OnInitDialog(); //重写初始化函数，以初始化控件

CComboBox m\_Sex; //控制性别的ComboBox控件的变量

afx\_msg void OnBnClickedExit(); //添加并关闭的消息处理程序

afx\_msg void OnBnClickedCancel(); //取消按钮的消息处理程序

afx\_msg void OnBnClickedEmpty(); //清空按钮的消息处理程序

afx\_msg void OnBnClickedAdd(); //添加按钮的消息处理程序

bool CheckInfo(); //检查信息是否已填写完整

CEdit m\_Edit;

};

//AddInfoDlg.cpp

BOOL AddInfoDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

//初始化ComboBox，默认初始选择第一项

m\_Sex.SetCurSel(0);

return TRUE;

}

void AddInfoDlg::OnBnClickedCancel()

{

//取消按钮的消息处理程序

CDialogEx::OnCancel();

}

void AddInfoDlg::OnBnClickedExit()

{

//添加并关闭的消息处理程序

UpdateData();

if (CheckInfo())

{

m\_Sex.GetLBText(m\_Sex.GetCurSel(), str\_Sex);

CDialogEx::OnOK();

}

}

void AddInfoDlg::OnBnClickedEmpty()

{

//清空按钮的消息处理程序，清空编辑框内信息

str\_Name.Empty();

str\_Num.Empty();

str\_Sex.Empty();

str\_Cpp.Empty();

str\_Math.Empty();

m\_Edit.SetFocus();

UpdateData(0);

}

void AddInfoDlg::OnBnClickedAdd()

{

//添加按钮的消息处理程序

UpdateData();

if (CheckInfo())

{

m\_Sex.GetLBText(m\_Sex.GetCurSel(), str\_Sex);

SendMessageA(AfxGetMainWnd()->m\_hWnd, WM\_ADDINFO, 0, 0);

OnBnClickedEmpty();

m\_Edit.SetFocus();

}

}

bool AddInfoDlg::CheckInfo()

{

//检查信息是否填写完整、规范

if (str\_Name == "")

{

MessageBox(\_T("请填写姓名！"), \_T("Error"));

return false;

}

if (str\_Num.GetLength() > 10)

{

MessageBox(\_T("学号的长度不能大于10！"), \_T("Error"));

return false;

}

if (\_ttoi(str\_Math)>100)

{

MessageBox(\_T("数学成绩不能大于100！"), \_T("Error"));

return false;

}

if (\_ttoi(str\_Cpp) > 100)

{

MessageBox(\_T("程序设计成绩不能大于100！"), \_T("Error"));

return false;

}

if (str\_Num == "" ||

str\_Cpp == "" ||

str\_Math == "")

{

//若信息不完整则弹出警告窗口，询问是否继续

AddWarning dlg;

if (dlg.DoModal() == IDOK)

return true;

else return false;

}

return true;

}

//AddWaring.cpp

void AddWarning::OnBnClickedCancel()

{

//取消按钮

CDialogEx::OnCancel();

}

void AddWarning::OnBnClickedContinue()

{

//继续按钮

CDialogEx::OnOK();

}

//AltDlg.h

#pragma once

class AltDlg : public CDialogEx

{

DECLARE\_DYNAMIC(AltDlg)

public:

AltDlg(CWnd\* pParent = nullptr); // 标准构造函数

virtual ~AltDlg();

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_DIALOG5 };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

DECLARE\_MESSAGE\_MAP()

public:

CComboBox m\_AltSex; //性别

CString str\_AltName; //名字

CString str\_AltNum; //学号

CString str\_AltMath; //数学

CString str\_AltCpp; //程设

afx\_msg void OnBnClickedAlter(); //修改按钮

afx\_msg void OnBnClickedCamcel(); //取消按钮

virtual BOOL OnInitDialog(); //初始化按钮

bool b\_Sex;

};

//AlgDlg.cpp

void AltDlg::OnBnClickedAlter()

{

UpdateData();

if (str\_AltName == "")

MessageBox(\_T("名字不能为空！"), \_T("Error"));

else

{

if (str\_AltNum.GetLength() > 10)

MessageBox(\_T("学号不能大于10位！"), \_T("Error"));

else

{

if (\_ttoi(str\_AltMath) > 100 || \_ttoi(str\_AltCpp) > 100)

MessageBox(\_T("成绩不能大于100！"), \_T("Error"));

else

{

b\_Sex = m\_AltSex.GetCurSel();

CDialogEx::OnOK();

}

}

}

}

void AltDlg::OnBnClickedCamcel()

{ //取消按钮

CDialogEx::OnCancel();

}

BOOL AltDlg::OnInitDialog()

{ //初始化

CDialogEx::OnInitDialog();

m\_AltSex.SetCurSel(b\_Sex);

UpdateData(0);

return TRUE;

}

//RankDlg.h

#pragma once

class RankDlg : public CDialogEx

{

DECLARE\_DYNAMIC(RankDlg)

public:

RankDlg(CWnd\* pParent = nullptr); // 标准构造函数

virtual ~RankDlg();

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_DIALOG4 };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

DECLARE\_MESSAGE\_MAP()

public:

CComboBox m\_Accord; //排序依据

CComboBox m\_UpDowm; //升倒序

virtual BOOL OnInitDialog();

afx\_msg void OnBnClickedRank(); //排序按钮

afx\_msg void OnBnClickedCancel(); //取消按钮

int i\_Accord; //排序依据

bool b\_UpDown; //升倒序

};

//RankDlg.cpp

BOOL RankDlg::OnInitDialog()

{ //初始化

CDialogEx::OnInitDialog();

m\_Accord.SetCurSel(0);

m\_UpDowm.SetCurSel(0);

return TRUE;

}

void RankDlg::OnBnClickedRank()

{ //排序按钮

i\_Accord = m\_Accord.GetCurSel();

b\_UpDown = m\_UpDowm.GetCurSel();

CDialog::OnOK();

}

void RankDlg::OnBnClickedCancel()

{ //取消按钮

CDialog::OnCancel();

}

//SearchDlg.h

#pragma once

#include "Tree.h"

class SearchDlg : public CDialogEx

{

DECLARE\_DYNAMIC(SearchDlg)

public:

SearchDlg(CWnd\* pParent = nullptr); // 标准构造函数

virtual ~SearchDlg();

#ifdef AFX\_DESIGN\_TIME

enum { IDD = IDD\_DIALOG3 };

#endif

protected:

virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持

DECLARE\_MESSAGE\_MAP()

public:

CString str\_Search; //查找关键字

afx\_msg void OnBnClickedOnSearch(); //查找按钮

afx\_msg void OnBnClickedSearchNext(); //查找下一个按钮

CComboBox m\_SearchKey; //查找依据

Tree\* p\_Search; //接收查找结果的指针

virtual BOOL OnInitDialog();

int i\_key; //记录查找依据

CButton m\_Next; //查找下一个按钮关联变量

bool Buttom; //判断查找还是查找下一个

};

//SearchDlg.cpp

void SearchDlg::OnBnClickedOnSearch()

{ //查找按钮

UpdateData(1);

Buttom = 1;

i\_key = m\_SearchKey.GetCurSel();

if (str\_Search != "") //判断查找关键字是否为空

CDialogEx::OnOK();

else MessageBox(\_T("关键字不能为空！"),\_T("Error"));

}

void SearchDlg::OnBnClickedSearchNext()

{ //查找下一个按钮

UpdateData(1);

Buttom = 0;

i\_key = m\_SearchKey.GetCurSel();

if (str\_Search != "")

CDialogEx::OnOK();

else MessageBox(\_T("关键字不能为空！"), \_T("Error"));

}

BOOL SearchDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

m\_SearchKey.SetCurSel(0);

return TRUE;

}

//StuMana.h

public:

afx\_msg void OnBnClickedAdd(); //该函数用于弹出添加信息窗口

void AddInfo(AddInfoDlg &dlg); //将信息写入表格并储存在链表中

SearchDlg m\_SchDlg;

AddInfoDlg m\_AddDlg; //“添加学生信息”窗口类

CListCtrl List\_Ctrl; //该表格用于显示学生信息

Management m\_MathStu;

Management m\_NameStu;

Management m\_NumStu;

Management m\_CppStu;

protected:

afx\_msg LRESULT OnAddinfo(WPARAM wParam, LPARAM lParam);

//处理来自“添加信息”子窗口的消息

public:

afx\_msg void OnBnClickedSearch();

afx\_msg void OnBnClickedDel();

afx\_msg void OnBnClickedRank();

afx\_msg void OnBnClickedAlter();

afx\_msg void OnBnClickedPic();

afx\_msg void OnBnClickedSave();

afx\_msg void OnBnClickedLoad();

afx\_msg void OnBnClickedAbout();

afx\_msg void OnBnClickedOk();

INT\_PTR OnSave();

};

//STuManaDlg.cpp

BOOL CStuManaDlg::OnInitDialog()

{

CDialogEx::OnInitDialog();

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

CMenu\* pSysMenu = GetSystemMenu(FALSE);

if (pSysMenu != nullptr)

{

BOOL bNameValid;

CString strAboutMenu;

bNameValid = strAboutMenu.LoadString(IDS\_ABOUTBOX);

ASSERT(bNameValid);

if (!strAboutMenu.IsEmpty())

{

pSysMenu->AppendMenu(MF\_SEPARATOR);

pSysMenu->AppendMenu(MF\_STRING, IDM\_ABOUTBOX, strAboutMenu);

}

}

SetIcon(m\_hIcon, TRUE); // 设置大图标

SetIcon(m\_hIcon, FALSE); // 设置小图标

//初始化ListControl控件

DWORD dwListCtrl = List\_Ctrl.GetExtendedStyle();

dwListCtrl |= LVS\_EX\_FULLROWSELECT;

dwListCtrl |= LVS\_EX\_GRIDLINES;

List\_Ctrl.SetExtendedStyle(dwListCtrl);

List\_Ctrl.InsertColumn(0, \_T("姓名"), LVCFMT\_CENTER, 70);

List\_Ctrl.InsertColumn(1, \_T("学号"), LVCFMT\_CENTER, 100);

List\_Ctrl.InsertColumn(2, \_T("性别"), LVCFMT\_CENTER, 70);

List\_Ctrl.InsertColumn(3, \_T("数学"), LVCFMT\_CENTER, 70);

List\_Ctrl.InsertColumn(4, \_T("面向对象程序设计"), LVCFMT\_CENTER, 120);

List\_Ctrl.InsertColumn(5, \_T("状态"), LVCFMT\_CENTER, 70);

return TRUE; // 除非将焦点设置到控件，否则返回 TRUE

}

void CStuManaDlg::OnBnClickedAdd()

{

//该函数用于弹出添加信息窗口

if (IDOK == m\_AddDlg.DoModal())

AddInfo(m\_AddDlg);

}

void CStuManaDlg::AddInfo(AddInfoDlg &dlg)

{

//将信息写入表格并储存在链表中

int i\_Count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_Count, dlg.str\_Name);

List\_Ctrl.SetItemText(i\_Count, 1, dlg.str\_Num);

List\_Ctrl.SetItemText(i\_Count, 2, dlg.str\_Sex);

List\_Ctrl.SetItemText(i\_Count, 3, dlg.str\_Math);

List\_Ctrl.SetItemText(i\_Count, 4, dlg.str\_Cpp);

m\_MathStu.ManaAdd(dlg.str\_Name, dlg.str\_Num, dlg.str\_Sex, dlg.str\_Math, dlg.str\_Cpp, 2);

m\_NameStu.ManaAdd(dlg.str\_Name, dlg.str\_Num, dlg.str\_Sex, dlg.str\_Math, dlg.str\_Cpp, 0);

m\_NumStu.ManaAdd(dlg.str\_Name, dlg.str\_Num, dlg.str\_Sex, dlg.str\_Math, dlg.str\_Cpp, 1);

m\_CppStu.ManaAdd(dlg.str\_Name, dlg.str\_Num, dlg.str\_Sex, dlg.str\_Math, dlg.str\_Cpp, 3);

List\_Ctrl.EnsureVisible(i\_Count, FALSE);

}

afx\_msg LRESULT CStuManaDlg::OnAddinfo(WPARAM wParam, LPARAM lParam)

{

//本窗口接收 WM\_ADDINFO 消息的处理函数

//用于不关闭子窗口的情况下添加学生信息

AddInfo(m\_AddDlg);

return 0;

}

void CStuManaDlg::OnBnClickedSearch()

{

CString str\_ForSearch = m\_SchDlg.str\_Search;

if (IDOK == m\_SchDlg.DoModal())

{

int i\_ForKey = m\_SchDlg.i\_key;

if (m\_SchDlg.Buttom)

{

switch (i\_ForKey)

{

case 0:

int i\_SchKey;

{

int i\_Len = m\_SchDlg.str\_Search.GetLength();

if (i\_Len > 0)

{

i\_SchKey = m\_SchDlg.str\_Search[0] - 19000;

i\_SchKey = i\_SchKey \* 100000;

if (i\_Len > 1)

i\_SchKey = i\_SchKey + m\_SchDlg.str\_Search[1] - 19000;

}

}; m\_SchDlg.p\_Search = m\_NameStu.ManaSearch(i\_SchKey); break;

case 1:m\_SchDlg.p\_Search = m\_NumStu.ManaSearch(\_ttoi(m\_SchDlg.str\_Search)); break;

case 2:m\_SchDlg.p\_Search = m\_MathStu.ManaSearch(\_ttoi(m\_SchDlg.str\_Search)); break;

case 3:m\_SchDlg.p\_Search = m\_CppStu.ManaSearch(\_ttoi(m\_SchDlg.str\_Search)); break;

}

CString str\_Check;

if (m\_SchDlg.p\_Search != NULL)

{

while (((Student\*)m\_SchDlg.p\_Search)->p\_Same != NULL)

{

switch (i\_ForKey)

{

case 0:str\_Check = ((Student\*)m\_SchDlg.p\_Search)->str\_Name; break;

case 1:str\_Check = ((Student\*)m\_SchDlg.p\_Search)->str\_Num; break;

case 2:str\_Check = ((Student\*)m\_SchDlg.p\_Search)->str\_Math; break;

case 3:str\_Check = ((Student\*)m\_SchDlg.p\_Search)->str\_Cpp; break;

}

if (str\_Check == m\_SchDlg.str\_Search)

break;

else m\_SchDlg.p\_Search = ((Student\*)m\_SchDlg.p\_Search)->p\_Same;

}

}

}

else

{

if (m\_SchDlg.i\_key == i\_ForKey && m\_SchDlg.str\_Search == str\_ForSearch)

{

if (m\_SchDlg.p\_Search)

m\_SchDlg.p\_Search = ((Student\*)m\_SchDlg.p\_Search)->p\_Same;

else MessageBox(\_T("已是最后一个"), NULL);

}

}

if (m\_SchDlg.p\_Search)

{ //若找到则选中该行并滚动列表至该行

List\_Ctrl.SetFocus();

List\_Ctrl.SetItemState(((Student\*)m\_SchDlg.p\_Search)->i\_Rank - 1, LVIS\_SELECTED | LVIS\_FOCUSED, LVIS\_SELECTED | LVIS\_FOCUSED);

List\_Ctrl.EnsureVisible(((Student\*)m\_SchDlg.p\_Search)->i\_Rank - 1, FALSE);

} //没找到则提示

else MessageBox(\_T("没有该记录或已删除"));

}

}

void CStuManaDlg::OnBnClickedDel()

{ //删除信息按钮

int i\_select;

POSITION pos\_sel;

pos\_sel = List\_Ctrl.GetFirstSelectedItemPosition();

if (pos\_sel != NULL) //若有选中某行

{

i\_select = List\_Ctrl.GetNextSelectedItem(pos\_sel);

while (1)

{

CString Name = List\_Ctrl.GetItemText(i\_select, 0);

int i\_Len = Name.GetLength();

int i\_DelKey = Name[0] - 19000;

i\_DelKey = i\_DelKey \* 100000;

i\_DelKey = i\_DelKey + Name[1] - 19000;

m\_NameStu.ManaDel(i\_DelKey, i\_select, 0);

m\_MathStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 3)), i\_select, 2);

m\_CppStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 4)), i\_select, 3);

m\_NumStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 1)), i\_select, 1);

List\_Ctrl.SetItemText(i\_select, 5, \_T("已删除"));

if (pos\_sel == NULL)

break;

pos\_sel = List\_Ctrl.GetFirstSelectedItemPosition();

i\_select = List\_Ctrl.GetNextSelectedItem(pos\_sel);

}

}

else MessageBox(\_T("未选中任何信息！"), \_T("Error"));

}

void CStuManaDlg::OnBnClickedRank()

{ //排序按钮

RankDlg m\_RankDlg;

if (IDOK == m\_RankDlg.DoModal())

{

Tree\* p\_Rank = NULL;

Tree\* p\_Ergodic = NULL;

Student\* p\_StuErg = NULL;

List\_Ctrl.DeleteAllItems();

switch (m\_RankDlg.i\_Accord)

{

case 0:p\_Rank = &m\_NameStu.m\_TreeHead; break;

case 1:p\_Rank = &m\_NumStu.m\_TreeHead; break;

case 2:p\_Rank = &m\_MathStu.m\_TreeHead; break;

case 3:p\_Rank = &m\_CppStu.m\_TreeHead; break;

}

int i\_Count = 0;

//重新将信息写入列表中

for (int i = 0; i < p\_Rank->i\_count; i++)

{

p\_StuErg = (Student\*)p\_Rank->p\_Node[i];

i\_Count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_Count, p\_StuErg->str\_Name);

List\_Ctrl.SetItemText(i\_Count, 1, p\_StuErg->str\_Num);

List\_Ctrl.SetItemText(i\_Count, 2, p\_StuErg->str\_Sex);

List\_Ctrl.SetItemText(i\_Count, 3, p\_StuErg->str\_Math);

List\_Ctrl.SetItemText(i\_Count, 4, p\_StuErg->str\_Cpp);

p\_StuErg->i\_Rank = i\_Count + 1;

while (p\_StuErg->p\_Same != NULL)

{

p\_StuErg = p\_StuErg->p\_Same;

i\_Count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_Count, p\_StuErg->str\_Name);

List\_Ctrl.SetItemText(i\_Count, 1, p\_StuErg->str\_Num);

List\_Ctrl.SetItemText(i\_Count, 2, p\_StuErg->str\_Sex);

List\_Ctrl.SetItemText(i\_Count, 3, p\_StuErg->str\_Math);

List\_Ctrl.SetItemText(i\_Count, 4, p\_StuErg->str\_Cpp);

p\_StuErg->i\_Rank = i\_Count + 1;

}

}

p\_Ergodic = p\_Rank->p\_Node[p\_Rank->i\_count];

for (; p\_Ergodic != p\_Rank; p\_Ergodic = p\_Ergodic->p\_Node[p\_Ergodic->i\_count])

{

for (int i = 0; i < p\_Ergodic->i\_count; i++)

{

p\_StuErg = (Student\*)p\_Ergodic->p\_Node[i];

i\_Count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_Count, p\_StuErg->str\_Name);

List\_Ctrl.SetItemText(i\_Count, 1, p\_StuErg->str\_Num);

List\_Ctrl.SetItemText(i\_Count, 2, p\_StuErg->str\_Sex);

List\_Ctrl.SetItemText(i\_Count, 3, p\_StuErg->str\_Math);

List\_Ctrl.SetItemText(i\_Count, 4, p\_StuErg->str\_Cpp);

p\_StuErg->i\_Rank = i\_Count + 1;

while (p\_StuErg->p\_Same != NULL)

{

p\_StuErg = p\_StuErg->p\_Same;

i\_Count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_Count, p\_StuErg->str\_Name);

List\_Ctrl.SetItemText(i\_Count, 1, p\_StuErg->str\_Num);

List\_Ctrl.SetItemText(i\_Count, 2, p\_StuErg->str\_Sex);

List\_Ctrl.SetItemText(i\_Count, 3, p\_StuErg->str\_Math);

List\_Ctrl.SetItemText(i\_Count, 4, p\_StuErg->str\_Cpp);

p\_StuErg->i\_Rank = i\_Count + 1;

}

}

}

}

if (m\_RankDlg.b\_UpDown)

{ //判断升降序

int i\_Up = 0;

int i\_Down = List\_Ctrl.GetItemCount() - 1;

CString str\_Info[5];

while (i\_Up < i\_Down)

{

for (int i = 0; i < 5; i++)

str\_Info[i] = List\_Ctrl.GetItemText(i\_Up, i);

for (int i = 0; i < 5; i++)

List\_Ctrl.SetItemText(i\_Up, i, List\_Ctrl.GetItemText(i\_Down, i));

for (int i = 0; i < 5; i++)

List\_Ctrl.SetItemText(i\_Down, i, str\_Info[i]);

i\_Up++;

i\_Down--;

}

}

}

void CStuManaDlg::OnBnClickedAlter()

{ //修改按钮

int i\_select;

POSITION pos\_sel;

pos\_sel = List\_Ctrl.GetFirstSelectedItemPosition();

if (pos\_sel != NULL)

{

i\_select = List\_Ctrl.GetNextSelectedItem(pos\_sel);

Student\* p\_Alt[4];

AltDlg dlg;

while (1)

{

CString Name = List\_Ctrl.GetItemText(i\_select, 0);

int i\_Len = Name.GetLength();

int i\_AltKey = 0;

if (i\_Len > 0)

{

i\_AltKey = Name[0] - 19000;

i\_AltKey = i\_AltKey \* 100000;

if (i\_Len > 1)

i\_AltKey = i\_AltKey + Name[1] - 19000;

} //查找信息

p\_Alt[0] = m\_NameStu.ManaSearch(i\_AltKey);

p\_Alt[1] = m\_NumStu.ManaSearch(\_ttoi(List\_Ctrl.GetItemText(i\_select, 1)));

p\_Alt[2] = m\_MathStu.ManaSearch(\_ttoi(List\_Ctrl.GetItemText(i\_select, 3)));

p\_Alt[3] = m\_CppStu.ManaSearch(\_ttoi(List\_Ctrl.GetItemText(i\_select, 4)));

dlg.str\_AltName = Name;

dlg.str\_AltNum = List\_Ctrl.GetItemText(i\_select, 1);

dlg.str\_AltMath = List\_Ctrl.GetItemText(i\_select, 3);

dlg.str\_AltCpp = List\_Ctrl.GetItemText(i\_select, 4);

if (List\_Ctrl.GetItemText(i\_select, 2) == "男")

dlg.b\_Sex = 0;

else dlg.b\_Sex = 1;

if (IDOK == dlg.DoModal())

{ //先删除信息

m\_NameStu.ManaDel(i\_AltKey, i\_select, 0);

m\_NumStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 1)), i\_select, 1);

m\_MathStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 3)), i\_select, 2);

m\_CppStu.ManaDel(\_ttoi(List\_Ctrl.GetItemText(i\_select, 4)), i\_select, 3);

CString str\_AltSex;

if (dlg.b\_Sex)

str\_AltSex = "女";

else str\_AltSex = "男";

//重新添加信息至树

m\_NameStu.ManaAdd(dlg.str\_AltName, dlg.str\_AltNum, str\_AltSex, dlg.str\_AltMath, dlg.str\_AltCpp, 0);

m\_NumStu.ManaAdd(dlg.str\_AltName, dlg.str\_AltNum, str\_AltSex, dlg.str\_AltMath, dlg.str\_AltCpp, 1);

m\_MathStu.ManaAdd(dlg.str\_AltName, dlg.str\_AltNum, str\_AltSex, dlg.str\_AltMath, dlg.str\_AltCpp, 2);

m\_CppStu.ManaAdd(dlg.str\_AltName, dlg.str\_AltNum, str\_AltSex, dlg.str\_AltMath, dlg.str\_AltCpp, 3);

List\_Ctrl.SetItemText(i\_select, 0, dlg.str\_AltName);

List\_Ctrl.SetItemText(i\_select, 1, dlg.str\_AltNum);

List\_Ctrl.SetItemText(i\_select, 2, str\_AltSex);

List\_Ctrl.SetItemText(i\_select, 3, dlg.str\_AltMath);

List\_Ctrl.SetItemText(i\_select, 4, dlg.str\_AltCpp);

List\_Ctrl.SetItemText(i\_select, 5, \_T("已修改"));

}

else

{

MessageBox(\_T("已取消"), \_T("Exit"));

break;

}

if (pos\_sel == NULL)

break;

pos\_sel = List\_Ctrl.GetFirstSelectedItemPosition();

i\_select = List\_Ctrl.GetNextSelectedItem(pos\_sel);

}

}

else MessageBox(\_T("未选中任何信息！"), \_T("Error"));

}

void CStuManaDlg::OnBnClickedPic()

{ //绘图按钮

CString str\_Path = L"";

int nPos;

GetModuleFileName(NULL, str\_Path.GetBufferSetLength(MAX\_PATH + 1), MAX\_PATH);

str\_Path.ReleaseBuffer();

nPos = str\_Path.ReverseFind(\_T('\\'));

str\_Path = str\_Path.Left(nPos);

str\_Path = str\_Path + \_T("\\StuPic.exe");

//获取路径

USES\_CONVERSION;

LPCSTR lpstr = (LPCSTR)T2A(str\_Path);

if (WinExec(lpstr, SW\_SHOWNORMAL) < 32) //打开程序

MessageBox(\_T("未能找到 StuPic.exe 请检查该文件是否与本程序在相同目录中！\n（注：暂不支持32位系统下运行！）"), \_T("Error"));

}

void CStuManaDlg::OnBnClickedSave()

{ //保存按钮

OnSave();

}

void CStuManaDlg::OnBnClickedLoad()

{ //打开按钮

CStdioFile m\_File;

BOOL isOpen = TRUE; //打开

CString str\_Dir = L"";

CString str\_FName = L"";

CString str\_Filter = L"文件 (\*.txt)|\*.txt||";

CFileDialog m\_LoadDlg(isOpen, str\_Dir, str\_FName, OFN\_HIDEREADONLY | OFN\_OVERWRITEPROMPT, str\_Filter, NULL);

INT\_PTR result = m\_LoadDlg.DoModal();

CString str\_FPath;

if (result == IDOK)

{

setlocale(LC\_CTYPE, "chs"); //转换字符模式

str\_FPath = m\_LoadDlg.GetPathName();

m\_File.Open(str\_FPath, CFile::modeRead);

int i\_count = List\_Ctrl.GetItemCount();

CString str\_Read;

CString str\_Info[5];

int n = 0;

while (m\_File.ReadString(str\_Read))

{ //读入信息

n = 0;

for (int i = 0; i < 5; i++)

{

for (; n < str\_Read.GetLength(); n++)

{

if (str\_Read.GetAt(n) != '\t')

str\_Info[i] = str\_Info[i] + str\_Read.GetAt(n);

else

{

n = n + 2;

break;

}

}

}

m\_NameStu.ManaAdd(str\_Info[0], str\_Info[1], str\_Info[2], str\_Info[3], str\_Info[4], 0);

m\_NumStu.ManaAdd(str\_Info[0], str\_Info[1], str\_Info[2], str\_Info[3], str\_Info[4], 1);

m\_MathStu.ManaAdd(str\_Info[0], str\_Info[1], str\_Info[2], str\_Info[3], str\_Info[4], 2);

m\_CppStu.ManaAdd(str\_Info[0], str\_Info[1], str\_Info[2], str\_Info[3], str\_Info[4], 3);

i\_count = List\_Ctrl.GetItemCount();

List\_Ctrl.InsertItem(i\_count, str\_Info[0]);

List\_Ctrl.SetItemText(i\_count, 1, str\_Info[1]);

List\_Ctrl.SetItemText(i\_count, 2, str\_Info[2]);

List\_Ctrl.SetItemText(i\_count, 3, str\_Info[3]);

List\_Ctrl.SetItemText(i\_count, 4, str\_Info[4]);

for (int i = 0; i < 5; i++)

str\_Info[i] = "";

}

}

}

void CStuManaDlg::OnBnClickedAbout()

{

About dlg;

dlg.DoModal();

}

void CStuManaDlg::OnBnClickedOk()

{ //保存并关闭

if (OnSave() == IDOK)

CDialogEx::OnOK();

}

INT\_PTR CStuManaDlg::OnSave()

{ //保存文件处理函数

CStdioFile m\_File;

BOOL isOpen = FALSE; //另存为

CString str\_Dir = L""; //默认打开的文件路径

CString str\_FName = L"Save"; //默认打开的文件名

CString str\_Filter = L"文件 (\*.txt)|\*.txt||"; //文件类型

CFileDialog m\_SaveDlg(isOpen, str\_Dir, str\_FName, OFN\_HIDEREADONLY | OFN\_OVERWRITEPROMPT, str\_Filter, NULL);

INT\_PTR result = m\_SaveDlg.DoModal(); //弹出保存对话框

CString str\_FPath;

if (result == IDOK)

{

setlocale(LC\_CTYPE, "chs");

str\_FPath = m\_SaveDlg.GetPathName();

m\_File.Open(str\_FPath, CFile::modeCreate | CFile::modeWrite);

int i\_count = List\_Ctrl.GetItemCount();

CString str\_Write;

CString str\_Tab;

str\_Tab = "\t\t";

for (int i = 0; i < i\_count; i++)

{

for (int n = 0; n < 5; n++)

{

str\_Write = List\_Ctrl.GetItemText(i, n) + str\_Tab;

m\_File.WriteString(str\_Write);

}

m\_File.WriteString(\_T("\n"));

}

}

return result;

}

void CStuManaDlg::OnBnClickedReStart()

{

//重置

::PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_SYSCOMMAND, SC\_CLOSE, NULL);

//获取exe程序当前路径

extern CStuManaApp theApp;

TCHAR szAppName[MAX\_PATH];

::GetModuleFileName(theApp.m\_hInstance, szAppName, MAX\_PATH);

CString strAppFullName;

strAppFullName.Format(\_T("%s"), szAppName);

//重启程序

STARTUPINFO StartInfo;

PROCESS\_INFORMATION procStruct;

memset(&StartInfo, 0, sizeof(STARTUPINFO));

StartInfo.cb = sizeof(STARTUPINFO);

::CreateProcess(

(LPCTSTR)strAppFullName,

NULL,

NULL,

NULL,

FALSE,

NORMAL\_PRIORITY\_CLASS,

NULL,

NULL,

&StartInfo,

&procStruct);

}

//Tree.h

#pragma once

class Tree

{

public:

Tree();

~Tree();

Tree\* p\_Node[5]; //结点指针

Tree\* p\_Former; //上层指针

int i\_key[4]; //关键字

int i\_count; //计数

int i\_onfl; //所在层数

int AddKey(int i\_AddKey); //添加关键字

void Write(int index, int key, Tree\* node, Tree\* former = NULL);

void Del(int index);

};

//Tree.cpp

#include "stdafx.h"

#include "Tree.h"

Tree::Tree()

: i\_count(0)

, p\_Former(NULL)

, i\_onfl(0)

{

for (int i = 0; i < 4; i++)

{

p\_Node[i] = NULL;

i\_key[i] = 0;

}

p\_Node[4] = NULL;

}

int Tree::AddKey(int i\_AddKey)

{

//添加关键字

int index = 0; //索引值

for (; index < i\_count; index++) //遍历该节点关键字

{

if (i\_AddKey < i\_key[index]) //若参数小于该关键字

break; //则退出循环

}

p\_Node[i\_count + 1] = p\_Node[i\_count]; //将最后一位指针向后移动

for (int n = i\_count; n >= index; n--) //将索引后所有指针与关键字后移

{

i\_key[n] = i\_key[n - 1];

p\_Node[n] = p\_Node[n - 1];

}

i\_key[index] = i\_AddKey; //写入至索引处

i\_count++; //计数

return index;

}

void Tree::Write(int index, int key, Tree\* node, Tree\* former)

{

i\_key[index] = key;

p\_Node[index] = node;

if (former != NULL)

p\_Former = former;

i\_count++;

}

void Tree::Del(int index)

{

p\_Node[index] = NULL;

if (index != 4)

{

i\_key[index] = 0;

i\_count--;

}

}

//MainFrm.h

public:

afx\_msg void OnFileOpen();

CStdioFile m\_File; //用于打开文件

CString str\_FPath; //储存路径

afx\_msg void OnMathHis();

afx\_msg void OnMathLine();

afx\_msg void OnCppHis();

afx\_msg void OnCppLine();

afx\_msg void OnUpdateMathHis(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateMathLine(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateCppHis(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateCppLine(CCmdUI \*pCmdUI);

void GetCount(int i\_Sub);

void DrawHis(bool b\_Sub);

void DrawLine(bool b\_Sub);

void ClearDraw();

int i\_Score[6]; //统计分段人数

afx\_msg void OnViewOMH();

afx\_msg void OnViewOCH();

afx\_msg void OnViewOCL();

afx\_msg void OnViewOML();

afx\_msg void OnUpdateViewMl(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewMh(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewCl(CCmdUI \*pCmdUI);

afx\_msg void OnUpdateViewCh(CCmdUI \*pCmdUI);

//MainFrm.cpp

void CMainFrame::OnFileOpen()

{

BOOL isOpen = TRUE; //打开

CString str\_Dir = L"";

CString str\_FName = L"";

CString str\_Filter = L"文件 (\*.txt)|\*.txt||";

CFileDialog m\_LoadDlg(isOpen, str\_Dir, str\_FName, OFN\_HIDEREADONLY | OFN\_OVERWRITEPROMPT, str\_Filter, NULL);

INT\_PTR result = m\_LoadDlg.DoModal();

if (result == IDOK)

{

setlocale(LC\_CTYPE, "chs");

str\_FPath = m\_LoadDlg.GetPathName();

GetCount(0);

DrawHis(1);

}

}

void CMainFrame::OnMathHis()

{ //绘制数学直方图

ClearDraw();

GetCount(0);

DrawHis(1);

}

void CMainFrame::OnMathLine()

{ //绘制数学折线图

ClearDraw();

GetCount(0);

DrawLine(1);

}

void CMainFrame::OnCppHis()

{ //绘制程设直方图

ClearDraw();

GetCount(2);

DrawHis(0);

}

void CMainFrame::OnCppLine()

{ //绘制程设折线图

ClearDraw();

GetCount(2);

DrawLine(0);

}

void CMainFrame::OnUpdateMathHis(CCmdUI \*pCmdUI)

{ //判断是否启用该控件

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateMathLine(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateCppHis(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateCppLine(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::GetCount(int i\_Sub)

{ //获取各分段人数

CString str\_Score;

CString str\_Read;

int i\_count = 0;

int n = 0;

m\_File.Open(str\_FPath, CFile::modeRead); //打开文件

for (int i = 0; i < 6; i++)

i\_Score[i] = 0;

while (m\_File.ReadString(str\_Read))

{ //按行读取

for (n = 0; ; n++)

{

if (i\_count == 6 + i\_Sub)

break;

if (str\_Read.GetAt(n) == '\t')

i\_count++;

}

for (; str\_Read.GetAt(n) != '\t'; n++)

str\_Score = str\_Score + str\_Read.GetAt(n);

switch (\_ttoi(str\_Score) / 10)

{

case 10:

case 9:i\_Score[0]++; break;

case 8:i\_Score[1]++; break;

case 7:i\_Score[2]++; break;

case 6:i\_Score[3]++; break;

default:i\_Score[4]++;

}

i\_Score[5]++;

str\_Score = "";

i\_count = 0;

}

m\_File.Close();

}

void CMainFrame::DrawHis(bool b\_Sub)

{ //绘制直方图

CRect rectClient;

CBrush m\_Brush;

CPen m\_Pen;

CClientDC dc(this);

CString str\_OutW;

if (b\_Sub) //按科目设定画刷画笔

{

m\_Brush.CreateHatchBrush(HS\_BDIAGONAL, RGB(255, 100, 100));

m\_Pen.CreatePen(PS\_SOLID, 1, RGB(255, 100, 100));

str\_OutW = "数学成绩直方图";

}

else

{

m\_Brush.CreateHatchBrush(HS\_FDIAGONAL, RGB(100, 100, 255));

m\_Pen.CreatePen(PS\_SOLID, 1, RGB(100, 100, 255));

str\_OutW = "程序设计成绩直方图";

}

dc.SelectObject(&m\_Brush);

dc.SelectObject(&m\_Pen);

GetClientRect(rectClient); //获取客户区坐标

rectClient.DeflateRect(25, 25);

int i\_wide = (rectClient.right - rectClient.left) / 7;

int i\_totalH = (rectClient.bottom - rectClient.top) - 40;

int i\_higt[5];

int i\_xy[4];

for (int i = 0; i < 5; i++)

{

i\_higt[i] = i\_Score[i] \* i\_totalH \* 6;

i\_higt[i] = i\_higt[i] / i\_Score[5];

i\_higt[i] = i\_higt[i] / 7;

}

dc.TextOutW(rectClient.top + 10,rectClient.left + 40, str\_OutW);

for (int i = 0; i < 5; i++)

{ //计算合理的绘图位置并绘图

i\_xy[0] = i\_wide \* (i + 1);

i\_xy[1] = rectClient.bottom - i\_higt[i] - (i\_totalH / 10) + 1;

i\_xy[2] = i\_xy[0] + i\_wide - 5;

i\_xy[3] = rectClient.bottom - (i\_totalH / 10);

dc.Rectangle(i\_xy[0], i\_xy[1], i\_xy[2], i\_xy[3]);

str\_OutW.Format(\_T("%d人"), i\_Score[i]);

dc.TextOutW(i\_xy[0] + i\_wide / 2.5, i\_xy[1] - 25, str\_OutW);

switch (i)

{

case 0:str\_OutW = \_T("0 - 59 分"); break;

case 1:str\_OutW = \_T("60 - 69 分"); break;

case 2:str\_OutW = \_T("70 - 79 分"); break;

case 3:str\_OutW = \_T("80 - 89 分"); break;

case 4:str\_OutW = \_T("90 - 100 分"); break;

} //输出相对应的文字

dc.TextOutW(i\_xy[0] + i\_wide / 4, i\_xy[3] + 5, str\_OutW);

}

}

void CMainFrame::DrawLine(bool b\_Sub)

{

CRect rectClient;

CBrush m\_Brush;

CPen m\_Pen;

CClientDC dc(this);

CString str\_OutW;

POINT m\_Point[2];

if (b\_Sub)

{

m\_Brush.CreateSolidBrush(RGB(255, 100, 100));

m\_Pen.CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

str\_OutW = "数学成绩折线图";

}

else

{

m\_Brush.CreateSolidBrush(RGB(100, 100, 255));

m\_Pen.CreatePen(PS\_SOLID, 1, RGB(0, 0, 255));

str\_OutW = "程序设计成绩折线图";

}

dc.SelectObject(&m\_Brush);

dc.SelectObject(&m\_Pen);

GetClientRect(rectClient); //获取客户区坐标

rectClient.DeflateRect(25, 25);

int i\_wide = (rectClient.right - rectClient.left) / 7;

int i\_totalH = (rectClient.bottom - rectClient.top);

int i\_higt[5];

int i\_xy[4];

for (int i = 0; i < 5; i++)

{

i\_higt[i] = i\_Score[i] \* i\_totalH \* 6;

i\_higt[i] = i\_higt[i] / i\_Score[5];

i\_higt[i] = i\_higt[i] / 7;

}

dc.TextOutW(rectClient.top + 10, rectClient.left + 40, str\_OutW);

for (int i = 0; i < 5; i++)

{

i\_xy[0] = i\_wide \* (i + 1);

i\_xy[1] = rectClient.bottom - i\_higt[i] - (i\_totalH / 10) + 1;

dc.Rectangle(i\_xy[0], i\_xy[1], i\_xy[0] + 5, i\_xy[1] + 5);

str\_OutW.Format(\_T("%d人"), i\_Score[i]);

dc.TextOutW(i\_xy[0] - 5, i\_xy[1] - 20, str\_OutW);

switch (i)

{

case 0:str\_OutW = \_T("0 - 59 分"); break;

case 1:str\_OutW = \_T("60 - 69 分"); break;

case 2:str\_OutW = \_T("70 - 79 分"); break;

case 3:str\_OutW = \_T("80 - 89 分"); break;

case 4:str\_OutW = \_T("90 - 100 分"); break;

}

dc.TextOutW(i\_xy[0] - 25, i\_xy[1] + 10, str\_OutW);

if (i > 0)

dc.LineTo(i\_xy[0], i\_xy[1] + 2);

dc.MoveTo(i\_xy[0] + 5, i\_xy[1] + 2);

}

}

void CMainFrame::ClearDraw()

{ //清除已绘制的图

CRect rectClient;

GetClientRect(rectClient);

CPen m\_Pen(PS\_SOLID, 1, RGB(255, 255, 255));

CClientDC dc(this);

dc.SelectObject(m\_Pen);

int left = rectClient.top + 10;

int right = rectClient.right - rectClient.left;

int top = rectClient.left + 60;

int bottom = rectClient.bottom - 20;

dc.Rectangle(left, top, right, bottom);

}

void CMainFrame::OnViewOMH()

{

OnMathHis();

}

void CMainFrame::OnViewOCH()

{

OnCppHis();

}

void CMainFrame::OnViewOCL()

{

OnCppLine();

}

void CMainFrame::OnViewOML()

{

OnMathLine();

}

void CMainFrame::OnUpdateViewMl(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateViewMh(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateViewCl(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

pCmdUI->Enable();

else pCmdUI->Enable(0);

}

void CMainFrame::OnUpdateViewCh(CCmdUI \*pCmdUI)

{

if (!str\_FPath.IsEmpty())

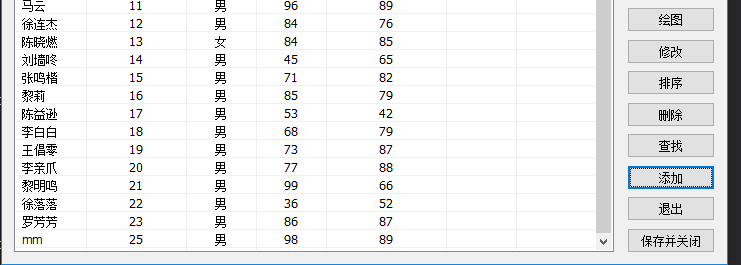
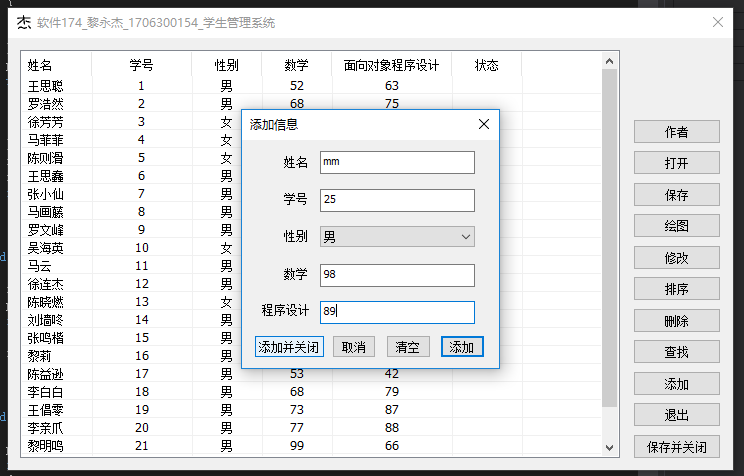
pCmdUI->Enable();

else pCmdUI->Enable(0);

}

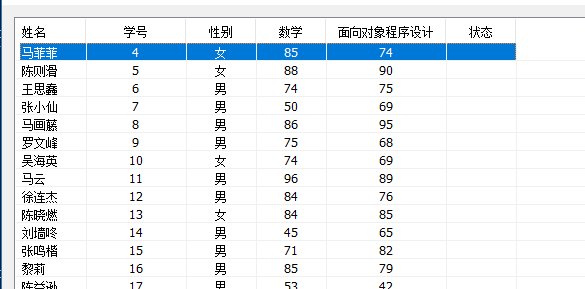
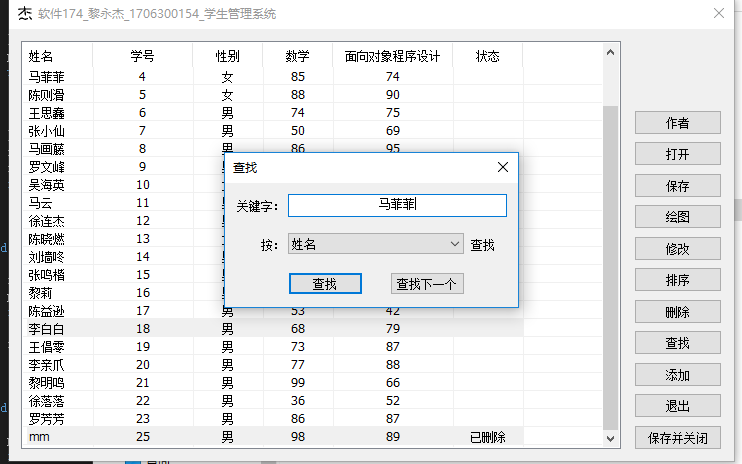
**五、实验结果**

1.添加信息功能



添加信息窗口中，点击“添加”键，可以将信息添加至列表但不关闭该窗口，并清空EDIT控件内的内容，使“姓名”的编辑框获得焦点；“清空”键可以清空所有编辑框的内容。

2.查找功能



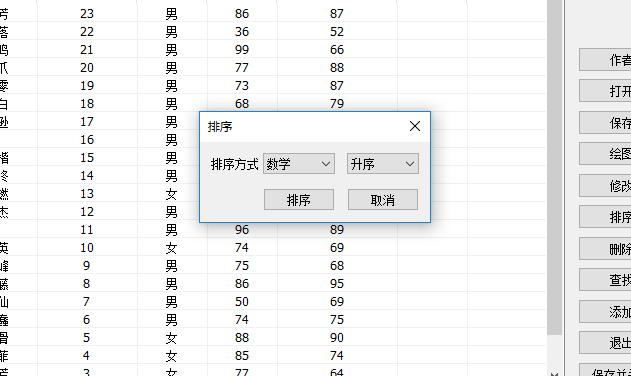
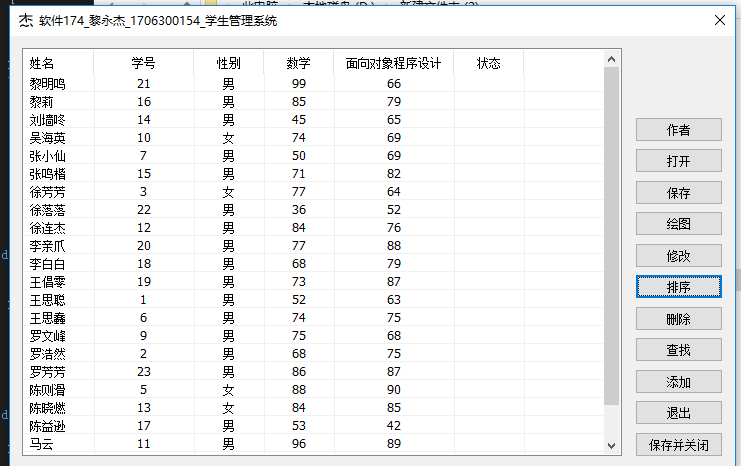
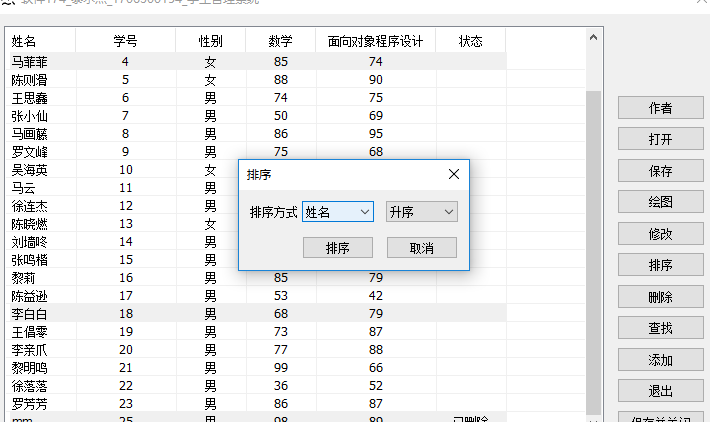
“查找”对话框内可选择以依据“姓名”、“学号”、“数学成绩”、“程序设计成绩”之一来查找信息；“查找下一个”可当在查找关键字不变的情况下，查找当前选中行的下一个与它关键字相同的信息；

3.删除



选中某一行后点击“删除”，可从内存删除该条信息，但不会从列表消失，取而代之的是“状态”出现“已删除”，以帮助误删恢复；删除后的信息经“排序”功能从而从列表消失；删除的信息不会被保存到文件。

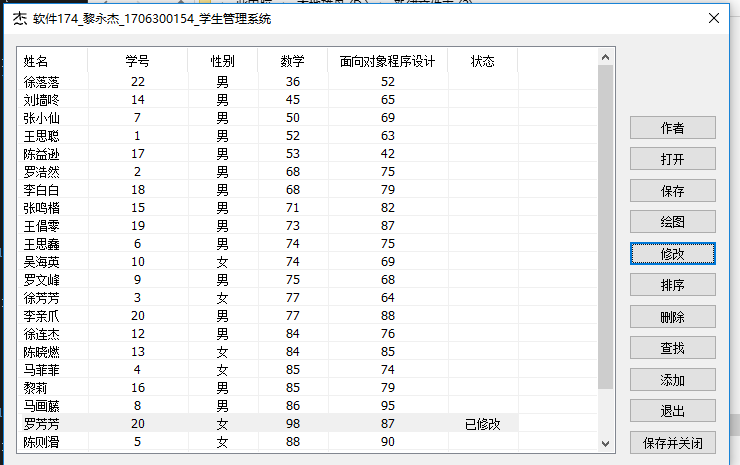
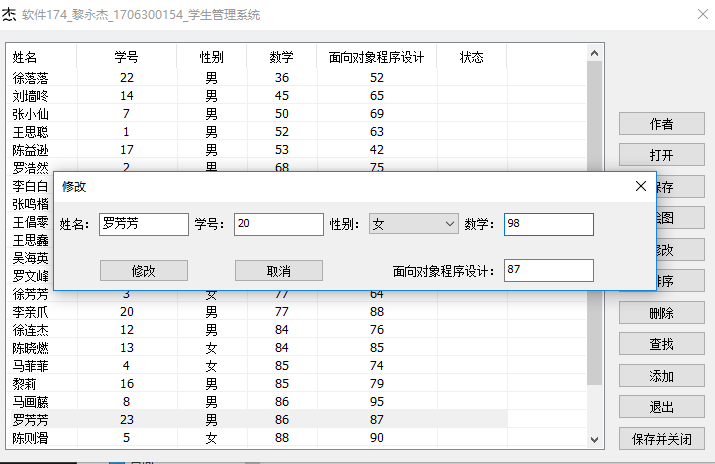
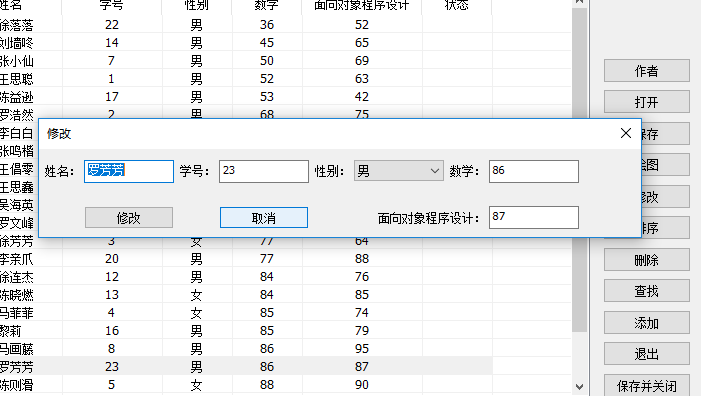
4.排序



排序可将“姓名”、“学号”、“数学”、“程序设计”的其中之一项作为排序依据，

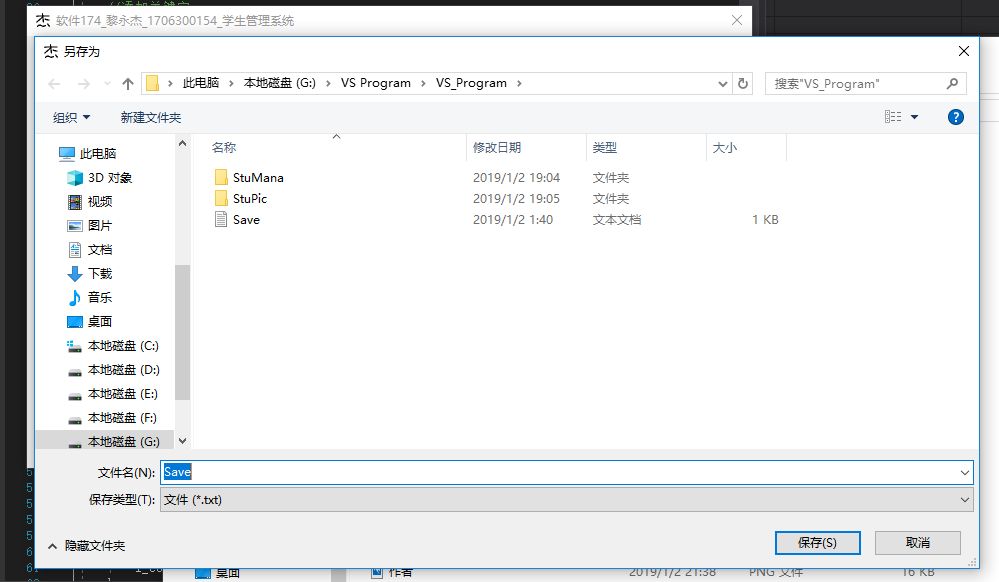
并可选择升序或降序，排序后“已删除”的信息将不会被显示

5.修改



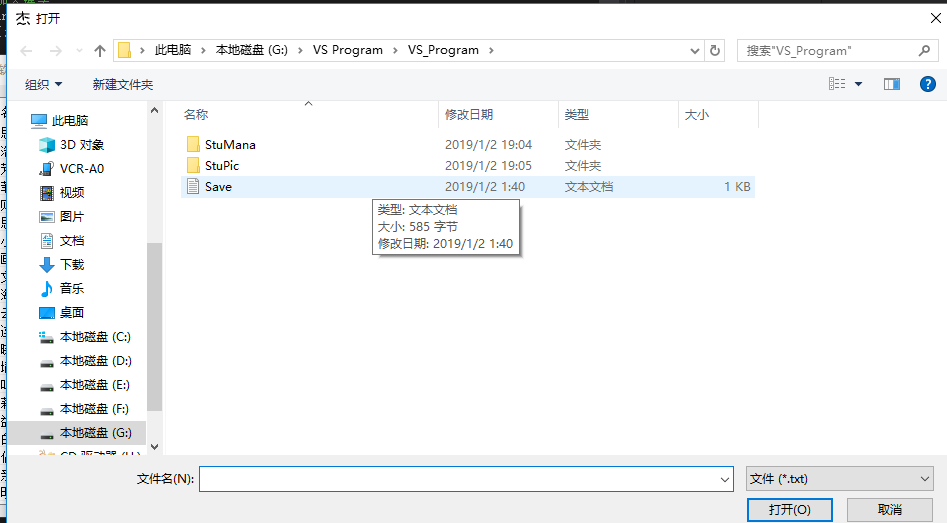
选中某一行后，点击修改按钮可以修改弹出的对话框内的信息，然后点击修改便可修改该行内容并刷新；修改功能支持多选并依次修改。

6.保存



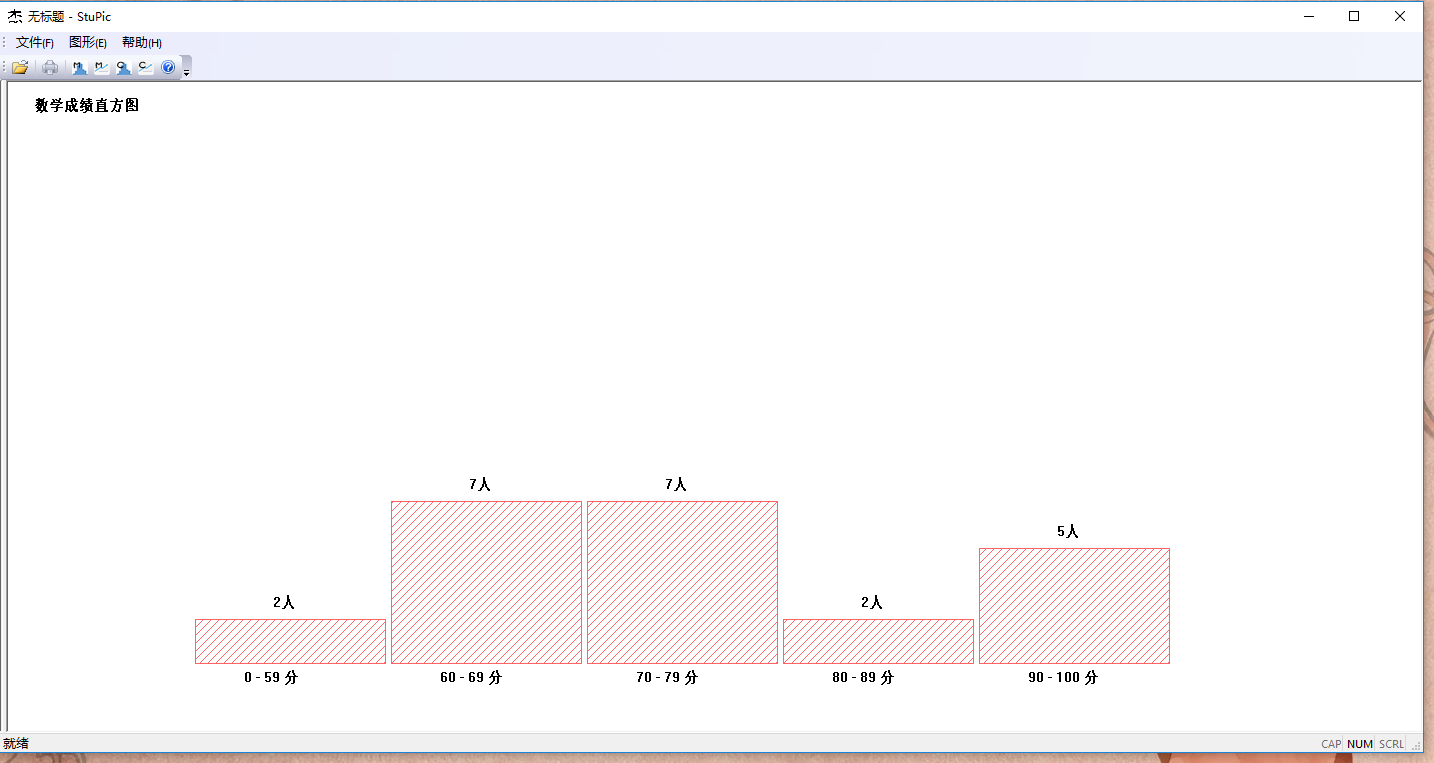
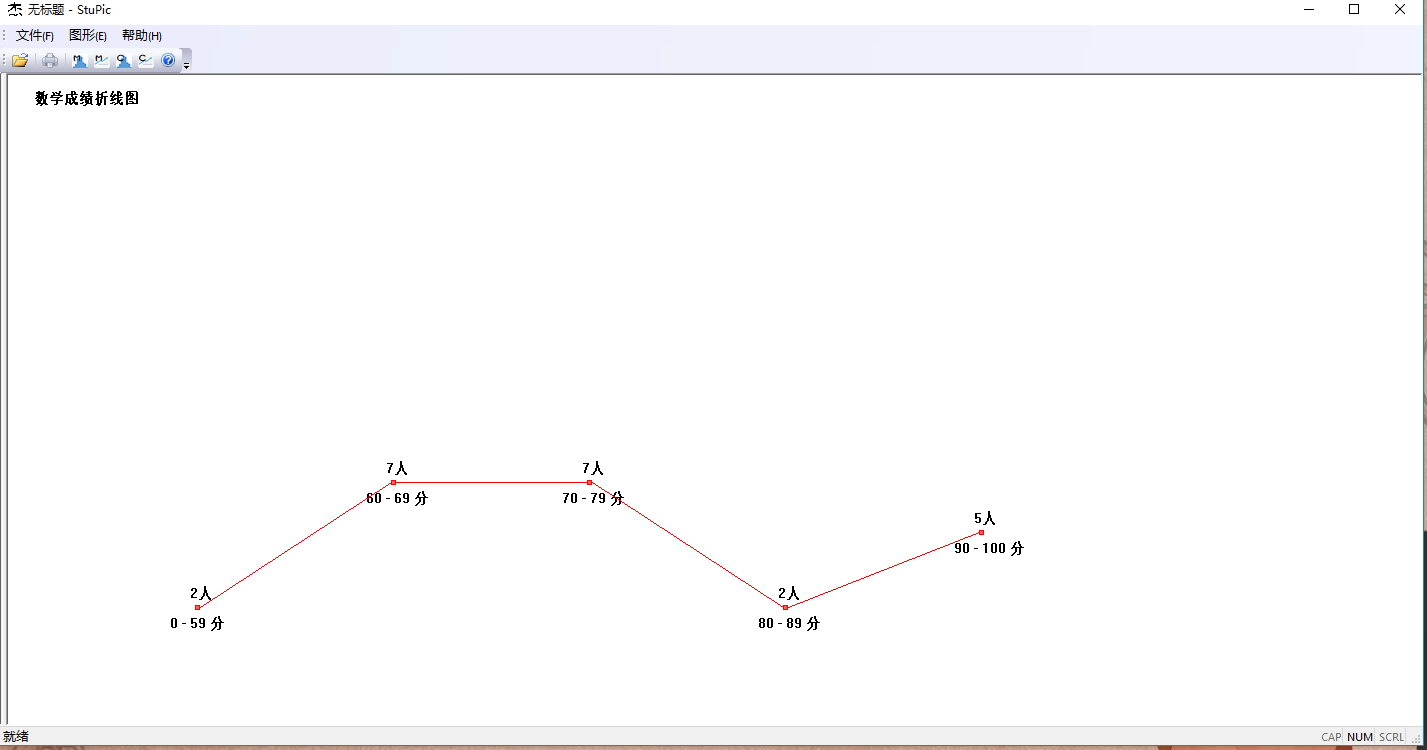
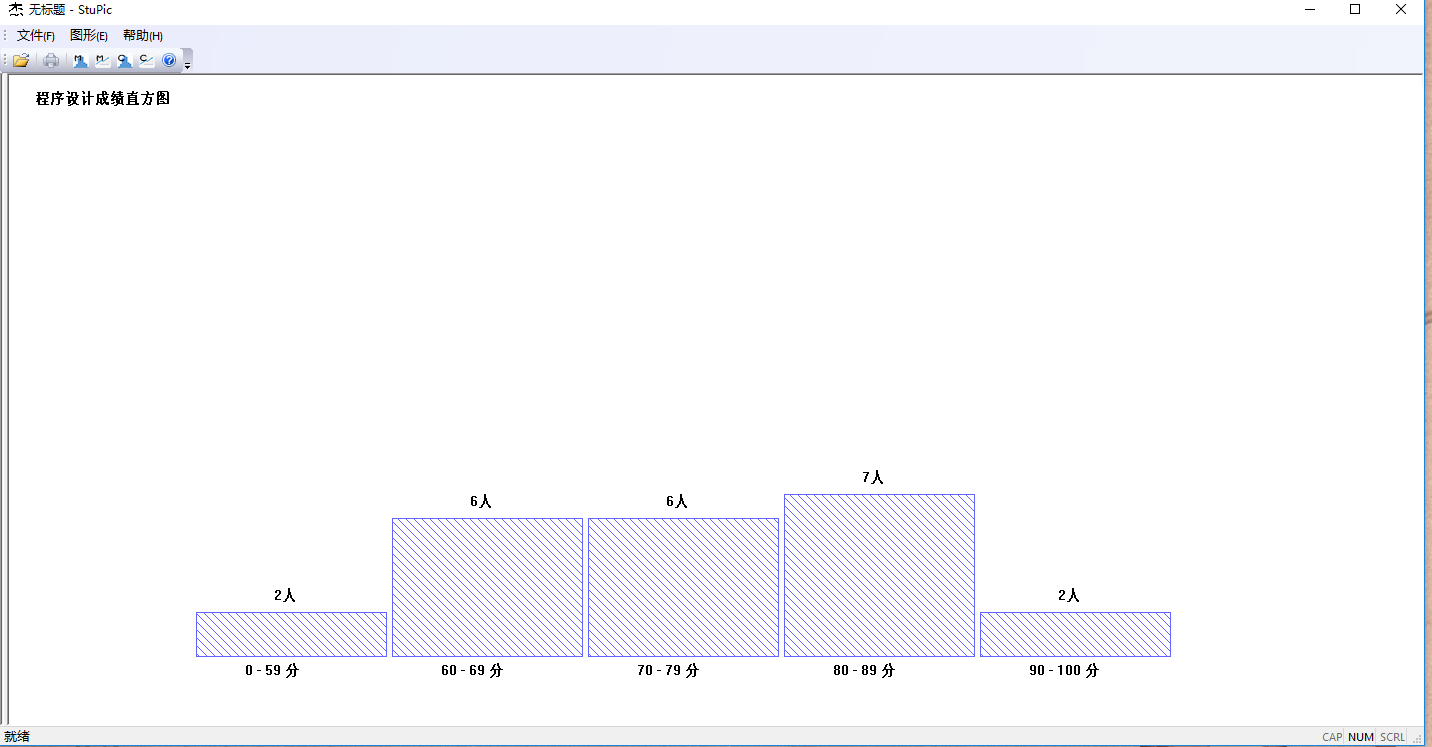
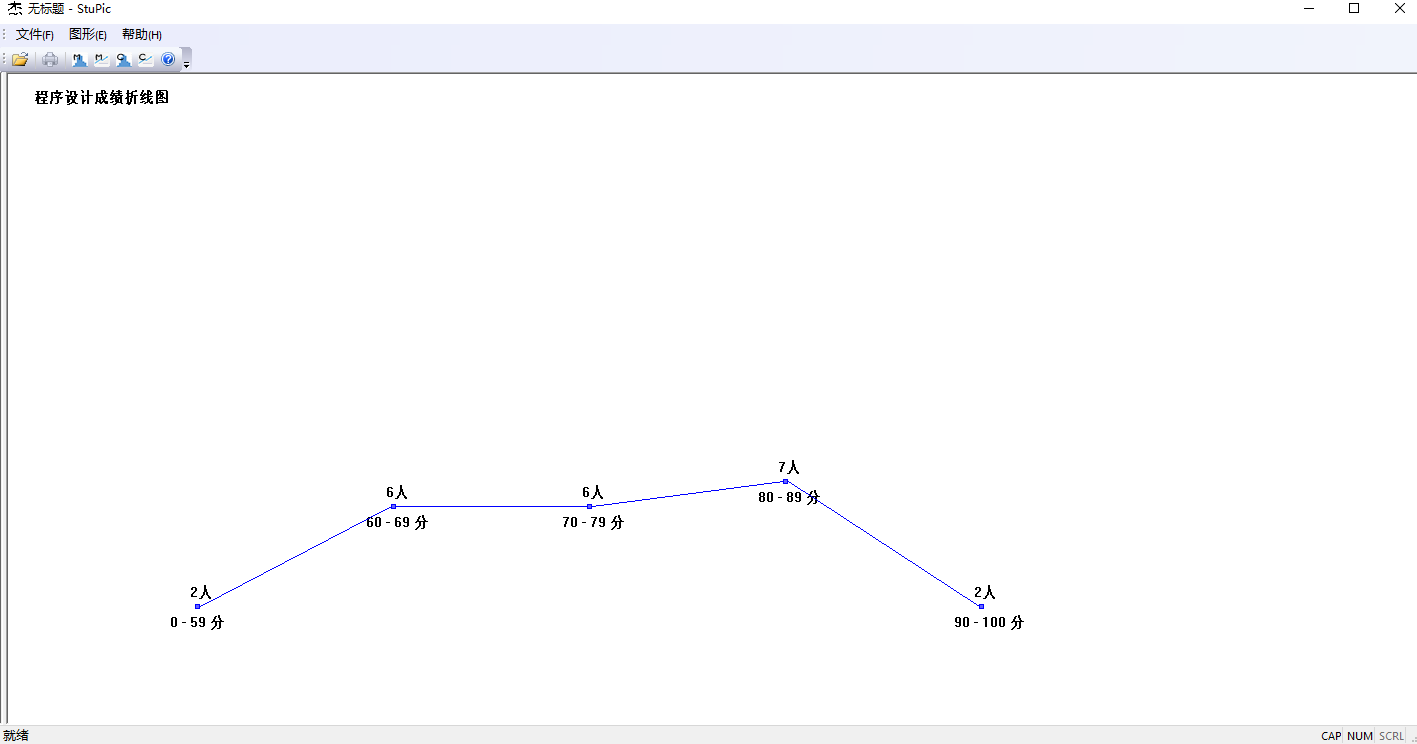
点击保存后可以将文件保存在自选的路径中，只能保存为.txt文件，文件内存有表格内“有效”的信息

7.打开



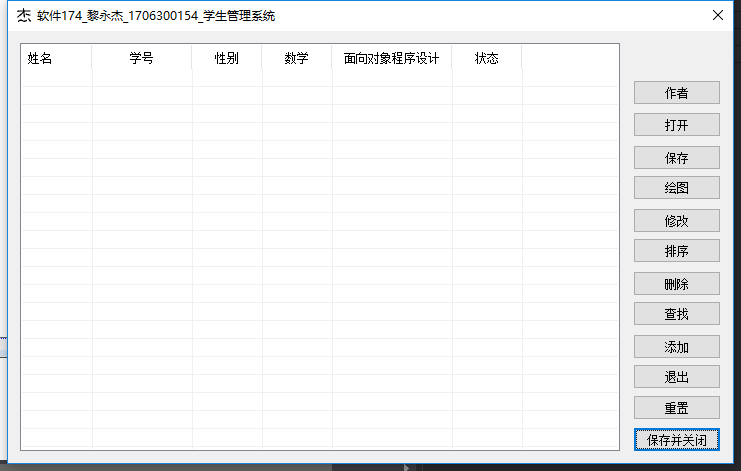
可打开以本程序储存的.txt文件，并录入到表格内，但若是打开未知的.txt可能会出问题。

8.绘图



绘图的程序可由“学生管理系统”的“绘图”按钮自动搜索同目录下“StuPic.exe”打开，或者自行打开“StuPic.exe”；打开.txt文件将数据录入后，自动绘制“数学成绩直方图”，可以从“图形”内选择所需要的统计图，也可以从工具栏的四个按钮快速的切换统计图。

9.重置



清空所有内容，并恢复至刚打开程序的状态

**六、实验疑难点分析**

1.本次实验操作数据的方式本应使用链表，但近期学习了B+树算法而且没有尝试过实现该算法，于是在本程序中首次使用了B+树 算法用以进行数据操作，但首次试验总是崎岖的，表面上该算法不难理解，但背地里却有许多小细节需要在实现的过程中注意，因此调试该算法耗费了大量的时间，同样也因为自身的考虑不周，导致了后续的其它函数与树的对接也出现了一些问题。

2.在代码展示中有一函数被注释了，该函数因考虑时不完善和未能及时分析出问题原因，从而不能通过测试，但由于函数其具有一定的独立性，可以暂时屏蔽掉该函数而不影响正常使用。

3.多个功能看似关联不大，但在实际的实现的过程中，功能间互相影响，互相制约，例如“添加”功能，若是添加信息时出现了错误，则“查找”、“删除”、“排序”等功能将会受到影响，若其它功能出现错误，错误本身未必在该功能上，也可能错误出现在“添加”功能上，这给测试调试添加了不少困难。

4.秉着“以人为本”的原则，总是心念着能有更人性化的设计，但这也增加了不少工作量和调试难度。

5.以本人的能力，考虑问题时不能面面俱到，大致的实现过程编写完成以后，调试的过程中会时不时遇到意想不到的情况，甚至有的情况出现的条件较为困难、复杂，给工作增加了不少难度。

6.合理的使用数组能有效的加强效率，但编写之初只考虑到数据并不多，故没有较多的使用数组，当发现数组能带来不少方便时，为时已晚。

7.编写“绘图”程序时，对界面的编辑比较困难，即使找到相关的初始化函数时，对其进行屏蔽、修改等操作时也会容易导致程序执行错误，对于“单文档”程序的学习有待加强。