# 实验十 面向对象程序设计实践

## 题目1 堆栈与简单语法检查

### 实验描述：

编写一个程序，利用堆栈

来检查一个C/C++语言程序中的花括号、方括号和圆括号是否配对，若能够完全配对返回1，

否则返回0

### 实验代码：

#### 1stack.cpp

/\*

@Author: 王贤义

 \* @FilePath: \code\C+++C\c++\10\1stack.cpp

@Description: 堆栈实现及括号匹配检查

\*/

#include <iostream>

#include <vector>

using namespace std;

const int MaxSize = 100; // 定义堆栈最大容量

typedef char ElemType;

class Stack

{

private:

    ElemType stack[MaxSize]; // 堆栈数组

    int top;                 // 堆栈指针，指向栈顶元素

public:

    Stack()

    { // 构造函数，初始化堆栈指针

        top = -1;

    }

    void InitStack()

    { // 初始化堆栈

        top = -1;

    }

    bool IsEmpty()

    { // 判断堆栈是否为空

        return top == -1;

    }

    bool IsFull()

    { // 判断堆栈是否已满

        return top == MaxSize - 1;

    }

    void push(ElemType item)

    { // 元素进栈

        if (IsFull())

        {

            cout << "Stack is full" << endl;

            return;

        }

        top++;

        stack[top] = item;

    }

    ElemType pop()

    { // 弹出栈顶元素并返回其值

        if (IsEmpty())

        {

            cout << "Stack is empty" << endl;

            return -1;

        }

        ElemType item = stack[top];

        top--;

        return item;

    }

    ElemType peek()

    { // 返回栈顶元素的值，但不移动栈顶指针

        if (IsEmpty())

        {

            cout << "Stack is empty" << endl;

            return -1;

        }

        return stack[top];

    }

    void ClearStack()

    { // 清空堆栈

        top = -1;

    }

};

/\*\*

@description: 检查括号匹配

@param {string} exp 待检查的表达式

@return {bool}

\*/

bool checkBalance(string exp)

{

    Stack s;

    for (int i = 0; i < exp.size(); i++)

    {

        char c = exp[i];

        if (c == '(' || c == '{' || c == '[')

        {

            s.push(c);

        }

        else if (c == ')' || c == '}' || c == ']')

        {

            if (s.IsEmpty())

            {

                return false;

            }

            char top = s.pop();

            if ((c == ')' && top != '(') || (c == '}' && top != '{') || (c == ']' && top != '['))

            {

                return false;

            }

        }

    }

    return s.IsEmpty();

}

int main()

{

    string exp;

    while (true)

    {

        cout << "Please input an expression (input 'exit' to exit): ";

        cin >> exp;

        if (exp == "exit")

        {

            break;

        }

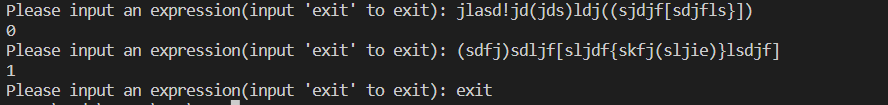
        cout << checkBalance(exp) << endl;

    }

    return 0;

}

### 实验结果：



### 思考与扩展：

无

## 题目2 成绩管理系统

### 实验描述：

现有学生成绩信息，内容如下

姓名 学号 语文数学 英语

张明明 01 67 78 82

李成友 02 78 91 88

张辉灿 03 68 82 56

王露 04 56 45 77

陈东明 05 67 38 47

…. .. .. .. …

请用C/C++编写一系统，实现学生信息管理，软件的入口界面应包括如下几个方面

的

1、功能要求：

（1）信息维护：

要求：学生信息数据要以文件的形式保存，能实现学生信息数据的维护。此模块包括子模块有：增加学生信息、删除学生信息、修改学生信息

（2）信息查询：

要求：查询时可实现按姓名查询、按学号查询

（3）成绩统计：

要求：A 输入任意的一个课程名（如数学）和一个分数段（如60--70），统计出在此分数段的学生情况。

（4）排序：能对用户指定的任意课程名，按成绩升序或降序排列学生数据并显示排序结果（使用表格的形式显示排序后的输出结果）

### 实验代码：

#### Student.h

#pragma once

#define MAX\_LOADSTRING 100

#define ID\_CHINESE\_RADIOBUTTON 101

#define ID\_MATH\_RADIOBUTTON 102

#define ID\_ENGLISH\_RADIOBUTTON 103

#define ID\_ASCENDING\_RADIOBUTTON 104

#define ID\_DESCENDING\_RADIOBUTTON 105

#include <Commdlg.h>

#include "resource.h"

// 全局变量:

HINSTANCE hInst; // 当前实例

WCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本

WCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名

// 页面句柄和按钮句柄

HWND hMainPage;

HWND hSecondPage1, hSecondPage2, hSecondPage3, hSecondPage4, hSecondPage5, hSecondPage6, hSecondPage7, hSecondPage8, hSecondPage9;

HWND hCurrentPage;

HWND hReturnButton;

HWND hButton1, hButton2, hButton3, hButton4, hButton5, hButton6, hButton7, hButton8, hButton9;

HWND hTextBoxa1, hTextBoxa2, hTextBoxa3, hTextBoxa4, hTextBoxa5;

HWND hTextBoxb1, hTextBoxb2, hTextBoxb3, hTextBoxb4, hTextBoxb5;

HWND hSubmitButton, hSubmitButton1, hSubmitButton2;

HWND hListView;

HWND hChineseRadioButton, hMathRadioButton, hEnglishRadioButton, hAscendingRadioButton, hDescendingRadioButton, hChineseRadioButton1, hMathRadioButton1, hEnglishRadioButton1;

const int COLUMN\_COUNT = 5;

const wchar\_t\* columnHeaders[COLUMN\_COUNT] = { L"Name", L"ID", L"Chinese", L"Math", L"English" };

#### Student.cpp

#include "framework.h"

#include "Student.h"

#include <bits/stdc++.h>

#include <CommCtrl.h>

using namespace std;

// 将wstring转换成string

int choice\_class, choice\_sort;

void fresh\_list();

string wstring2string(wstring wstr)

{

string result;

// 获取缓冲区大小，并申请空间，缓冲区大小事按字节计算的

int len = WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), NULL, 0, NULL, NULL);

char\* buffer = new char[len + 1];

// 宽字节编码转换成多字节编码

WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), buffer, len, NULL, NULL);

buffer[len] = '\0';

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

wstring string2wstring(string str)

{

wstring result;

// 获取缓冲区大小，并申请空间，缓冲区大小按字符计算

int len = MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), NULL, 0);

TCHAR\* buffer = new TCHAR[len + 1];

// 多字节编码转换成宽字节编码

MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), buffer, len);

buffer[len] = '\0'; // 添加字符串结尾

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

// 学生信息结构体

struct Student

{

string name;

string id;

int chinese;

int math;

int english;

};

vector<Student> students;

vector<Student> read\_student\_file(vector<Student>& students)

{

OPENFILENAMEA ofn;

CHAR szFile[MAX\_PATH];

ZeroMemory(&ofn, sizeof(ofn));

ZeroMemory(szFile, sizeof(szFile));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFilter = "Text Files (\*.txt)\0\*.txt\0All Files (\*.\*)\0\*.\*\0";

ofn.lpstrFile = szFile;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_EXPLORER | OFN\_FILEMUSTEXIST | OFN\_HIDEREADONLY;

ofn.lpstrDefExt = "txt";

if (GetOpenFileNameA(&ofn))

{

std::ifstream infile(ofn.lpstrFile);

if (!infile.is\_open())

{

std::cerr << "Error: can't open file " << ofn.lpstrFile << std::endl;

return students;

}

std::string line;

while (std::getline(infile, line))

{

std::istringstream iss(line);

std::string name, id;

std::string dummy;

int chinese, math, english;

if (!(iss >> name >> dummy >> id >> dummy >> chinese >> dummy >> math >> dummy >> english))

{

std::cerr << "Error: invalid student file format" << std::endl;

return students;

}

students.push\_back({ name, id, chinese, math, english });

}

infile.close();

}

return students;

}

// 写入学生信息文件

void write\_student\_file(string filename, vector<Student>& students)

{

ofstream outfile(filename);

if (!outfile.is\_open())

{

cerr << "Error: can't open file " << filename << endl;

return;

}

for (const auto& s : students)

{

outfile << s.name << " , " << s.id << " , " << s.chinese << " , " << s.math << " , " << s.english << endl;

}

outfile.close();

}

// 根据姓名查找学生

vector<Student> find\_student\_by\_name(vector<Student>& students, string name)

{

vector<Student> result;

for (const auto& s : students)

{

if (s.name == name)

{

result.push\_back(s);

}

}

return result;

}

// 根据学号查找学生

Student\* find\_student\_by\_id(vector<Student>& students, string id)

{

for (const auto& s : students)

{

if (s.id == id)

{

return (Student\*)&s;

}

}

return NULL;

}

// 添加学生信息

void add\_student(vector<Student>& students)

{

string name, id;

int chinese, math, english;

cout << "Enter student name: ";

cin >> name;

cout << "Enter student id: ";

cin >> id;

cout << "Enter Chinese score: ";

cin >> chinese;

cout << "Enter Math score: ";

cin >> math;

cout << "Enter English score: ";

cin >> english;

students.push\_back({ name, id, chinese, math, english });

cout << "Student " << name << " added." << endl;

}

// 删除学生信息

void delete\_student(vector<Student>& students, string id)

{

cout << "Enter student id: ";

// cin >> id;

std::vector<Student>::iterator it = remove\_if(students.begin(), students.end(), [id](const Student& s)

{ return s.id == id; });

if (it != students.end())

{

students.erase(it, students.end());

cout << "Student " << id << " deleted." << endl;

}

else

{

cout << "Error: student " << id << " not found." << endl;

}

}

// 修改学生信息

void modify\_student(vector<Student>& students)

{

string id;

cout << "Enter student id: ";

cin >> id;

auto s = find\_student\_by\_id(students, id);

if (s == nullptr)

{

cout << "Error: student " << id << " not found." << endl;

return;

}

// 修改学生信息

int chinese, math, english;

cout << "Enter Chinese score [" << s->chinese << "]: ";

cin >> chinese;

cout << "Enter Math score [" << s->math << "]: ";

cin >> math;

cout << "Enter English score [" << s->english << "]: ";

cin >> english;

if (chinese != -1)

{

s->chinese = chinese;

}

if (math != -1)

{

s->math = math;

}

if (english != -1)

{

s->english = english;

}

cout << "Student " << s->name << " modified." << endl;

}

void score\_statistics(vector<Student>& students)

{

string course;

int low, high;

cout << "Enter course name: ";

cin >> course;

cout << "Enter score range [low high]: ";

cin >> low >> high;

int count = 0;

for (const Student& s : students)

{

int score;

if (course == "Chinese")

{

score = s.chinese;

}

else if (course == "Math")

{

score = s.math;

}

else if (course == "English")

{

score = s.english;

}

else

{

cout << "Error: invalid course name." << endl;

return;

}

if (score >= low && score <= high)

{

count++;

}

}

cout << "There are " << count << " students with " << course << " scores between " << low << " and " << high << "." << endl;

}

void sort\_students(vector<Student>& students)

{

string course;

int order;

cout << "Enter course name: ";

cin >> course;

cout << "Enter sort order (0 for ascending, 1 for descending): ";

cin >> order; // 按成绩排序

if (course == "Chinese")

{

if (order == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.chinese < b.chinese; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.chinese > b.chinese; });

}

}

else if (course == "Math")

{

if (order == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.math < b.math; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.math > b.math; });

}

}

else if (course == "English")

{

if (order == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.english < b.english; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.english > b.english; });

}

}

else

{

cout << "Error: invalid course name." << endl;

return;

}

// 输出排序结果

cout << "Name\tID\tChinese\tMath\tEnglish" << endl;

for (const Student& s : students)

{

// 输出学生信息

cout << s.name << "\t" << s.id << "\t" << s.chinese << "\t" << s.math << "\t" << s.english << endl;

}

}

// 计算平均分

void average\_score(vector<Student>& students)

{

string course;

cout << "Enter course name: ";

cin >> course;

double sum = 0;

for (const Student& s : students)

{

int score;

if (course == "Chinese")

{

score = s.chinese;

}

else if (course == "Math")

{

score = s.math;

}

else if (course == "English")

{

score = s.english;

}

else

{

cout << "Error: invalid course name." << endl;

return;

}

sum += score;

}

cout << "The average score of " << course << " is " << sum / students.size() << "." << endl;

}

void search\_students(vector<Student>& students)

{

int choice = 0;

cout << "1. Search by name" << endl;

cout << "2. Search by id" << endl;

cout << "Enter your choice [1-2]: ";

cin >> choice;

if (choice == 1)

{

string name;

cout << "Enter student name: ";

cin >> name;

vector<Student> result = find\_student\_by\_name(students, name);

if (result.empty())

{

cout << "Error: student " << name << " not found." << endl;

return;

}

// 输出学生信息

cout << "Name\tID\tChinese\tMath\tEnglish" << endl;

for (const auto& s : result)

{

cout << s.name << "\t" << s.id << "\t" << s.chinese << "\t" << s.math << "\t" << s.english << endl;

}

}

else if (choice == 2)

{

string id;

cout << "Enter student id: ";

cin >> id;

Student\* s = find\_student\_by\_id(students, id);

if (s == NULL)

{

cout << "Error: student " << id << " not found." << endl;

return;

}

// 输出学生信息

cout << "Name\tID\tChinese\tMath\tEnglish" << endl;

cout << s->name << "\t" << s->id << "\t" << s->chinese << "\t" << s->math << "\t" << s->english << endl;

}

else

{

cout << "Error: invalid choice." << endl;

}

}

void show\_all\_students(vector<Student>& students)

{

// 输出学生信息

cout << "Name\tID\tChinese\tMath\tEnglish" << endl;

for (const auto& s : students)

{

cout << s.name << "\t" << s.id << "\t" << s.chinese << "\t" << s.math << "\t" << s.english << endl;

}

}

// 此代码模块中包含的函数的前向声明:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: 在此处放置代码。

//read\_student\_file(students);

// 初始化全局字符串

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_STUDENT, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// 执行应用程序初始化:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_STUDENT));

MSG msg;

// 主消息循环:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_STUDENT));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_STUDENT);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hMainPage = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hMainPage)

{

return FALSE;

}

hListView = CreateWindowW(WC\_LISTVIEW, L"", WS\_VISIBLE | WS\_CHILD | LVS\_REPORT, 10, 10, 500,500, hMainPage, nullptr, hInstance, nullptr);

// 设置列表视图控件样式

ListView\_SetExtendedListViewStyle(hListView, LVS\_EX\_FULLROWSELECT);

// 添加列表视图控件的列

LVCOLUMN lvColumn;

lvColumn.mask = LVCF\_TEXT | LVCF\_WIDTH;

lvColumn.cx = 100;

for (int i = 0; i < COLUMN\_COUNT; i++)

{

lvColumn.pszText = const\_cast<wchar\_t\*>(columnHeaders[i]);

ListView\_InsertColumn(hListView, i, &lvColumn);

}

fresh\_list();

// 创建9个按钮

hButton1 = CreateWindowW(L"BUTTON", L"添加学生", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

600, 50, 100, 30, hMainPage, (HMENU)1, hInstance, NULL);

hButton2 = CreateWindowW(L"BUTTON", L"删除学生", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

700, 50, 100, 30, hMainPage, (HMENU)2, hInstance, NULL);

hButton3 = CreateWindowW(L"BUTTON", L"修改信息", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

800, 50, 100, 30, hMainPage, (HMENU)3, hInstance, NULL);

hButton4 = CreateWindowW(L"BUTTON", L"搜索信息", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

600, 100, 100, 30, hMainPage, (HMENU)4, hInstance, NULL);

hButton5 = CreateWindowW(L"BUTTON", L"成绩统计", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

700, 100, 100, 30, hMainPage, (HMENU)5, hInstance, NULL);

hButton6 = CreateWindowW(L"BUTTON", L"学生排序", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

800, 100, 100, 30, hMainPage, (HMENU)6, hInstance, NULL);

hButton7 = CreateWindowW(L"BUTTON", L"打开文件", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

600, 150, 100, 30, hMainPage, (HMENU)7, hInstance, NULL);

hButton8 = CreateWindowW(L"BUTTON", L"统计均值", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

700, 150, 100, 30, hMainPage, (HMENU)8, hInstance, NULL);

hButton9 = CreateWindowW(L"BUTTON", L"保存退出", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

800, 150, 100, 30, hMainPage, (HMENU)9, hInstance, NULL);

hCurrentPage = hMainPage; // 设置当前页面为主窗口

ShowWindow(hMainPage, nCmdShow);

UpdateWindow(hMainPage);

return TRUE;

}

void fresh\_list()

{

ListView\_DeleteAllItems(hListView);

for (int i = 0; i < students.size(); i++)

{

LVITEM lvItem;

lvItem.mask = LVIF\_TEXT;

lvItem.iItem = i;

lvItem.iSubItem = 0;

wstring name = string2wstring(students[i].name);

lvItem.pszText = &name[0]; // 插入姓名

// 插入第一列的数据

ListView\_InsertItem(hListView, &lvItem);

// 插入其他列的数据

std::wstring chinese = std::to\_wstring(students[i].chinese);

std::wstring math = std::to\_wstring(students[i].math);

std::wstring english = std::to\_wstring(students[i].english);

lvItem.iSubItem = 1;

std::wstring id = string2wstring(students[i].id);

lvItem.pszText = &id[0]; // 插入学号

ListView\_SetItemText(hListView, i, 1, lvItem.pszText);

lvItem.iSubItem = 2;

lvItem.pszText = (LPWSTR)chinese.c\_str(); // 插入语文成绩

ListView\_SetItemText(hListView, i, 2, lvItem.pszText);

lvItem.iSubItem = 3;

lvItem.pszText = (LPWSTR)math.c\_str(); // 插入数学成绩

ListView\_SetItemText(hListView, i, 3, lvItem.pszText);

lvItem.iSubItem = 4;

lvItem.pszText = (LPWSTR)english.c\_str(); // 插入英语成绩

ListView\_SetItemText(hListView, i, 4, lvItem.pszText);

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// 分析菜单选择:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case 1: // 处理按钮1的点击事件

{

// 创建第二个页面窗口

hSecondPage1 = CreateWindowW(szWindowClass, L"添加学生", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage1, SW\_SHOW);

HWND hLabel1 = CreateWindowW(L"STATIC", L"姓名:", WS\_CHILD | WS\_VISIBLE, 150, 50, 100, 20, hSecondPage1, NULL, hInst, NULL);

HWND hLabel2 = CreateWindowW(L"STATIC", L"ID:", WS\_CHILD | WS\_VISIBLE, 150, 80, 100, 20, hSecondPage1, NULL, hInst, NULL);

HWND hLabel3 = CreateWindowW(L"STATIC", L"语文:", WS\_CHILD | WS\_VISIBLE, 150, 110, 100, 20, hSecondPage1, NULL, hInst, NULL);

HWND hLabel4 = CreateWindowW(L"STATIC", L"数学:", WS\_CHILD | WS\_VISIBLE, 150, 140, 100, 20, hSecondPage1, NULL, hInst, NULL);

HWND hLabel5 = CreateWindowW(L"STATIC", L"英语:", WS\_CHILD | WS\_VISIBLE, 150, 170, 100, 20, hSecondPage1, NULL, hInst, NULL);

hTextBoxa1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 50, 200, 20, hSecondPage1, NULL, hInst, NULL);

hTextBoxa2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 80, 200, 20, hSecondPage1, NULL, hInst, NULL);

hTextBoxa3 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 110, 200, 20, hSecondPage1, NULL, hInst, NULL);

hTextBoxa4 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 140, 200, 20, hSecondPage1, NULL, hInst, NULL);

hTextBoxa5 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 170, 200, 20, hSecondPage1, NULL, hInst, NULL);

// 创建提交按钮

hSubmitButton = CreateWindowW(L"BUTTON", L"确认", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage1, (HMENU)11, hInst, NULL);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage1, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage1;

}

break;

case 2:

{

// 创建第二个页面窗口

hSecondPage2 = CreateWindowW(szWindowClass, L"删除学生", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

HWND hLabel1 = CreateWindowW(L"STATIC", L"ID：", WS\_CHILD | WS\_VISIBLE, 50, 90, 20, 20, hSecondPage2, NULL, hInst, NULL);

hTextBoxa1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

70, 90, 80, 20, hSecondPage2, NULL, hInst, NULL);

hSubmitButton = CreateWindowW(L"BUTTON", L"确认", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage2, (HMENU)12, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage2, SW\_SHOW);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage2, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage2;

}

break;

case 3:

{

// 创建第二个页面窗口

hSecondPage3 = CreateWindowW(szWindowClass, L"修改信息", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

hTextBoxa1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 50, 200, 20, hSecondPage3, NULL, hInst, NULL);

HWND hLabel1 = CreateWindowW(L"STATIC", L"ID：", WS\_CHILD | WS\_VISIBLE, 150, 50, 50, 20, hSecondPage3, NULL, hInst, NULL);

hTextBoxa2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 80, 200, 20, hSecondPage3, NULL, hInst, NULL);

HWND hLabel2 = CreateWindowW(L"STATIC", L"语文：", WS\_CHILD | WS\_VISIBLE, 150, 80, 50, 20, hSecondPage3, NULL, hInst, NULL);

hTextBoxa3 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 110, 200, 20, hSecondPage3, NULL, hInst, NULL);

HWND hLabel3 = CreateWindowW(L"STATIC", L"数学：", WS\_CHILD | WS\_VISIBLE, 150, 110, 50, 20, hSecondPage3, NULL, hInst, NULL);

hTextBoxa4 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

200, 140, 200, 20, hSecondPage3, NULL, hInst, NULL);

HWND hLabel4 = CreateWindowW(L"STATIC", L"英语：", WS\_CHILD | WS\_VISIBLE, 150, 140, 50, 20, hSecondPage3, NULL, hInst, NULL);

hSubmitButton = CreateWindowW(L"BUTTON", L"确认", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage3, (HMENU)13, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage3, SW\_SHOW);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage3, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage3;

}

break;

case 4:

{

// 创建第二个页面窗口

hSecondPage4 = CreateWindowW(szWindowClass, L"Second Page", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

HWND hLabel1 = CreateWindowW(L"STATIC", L"ID/姓名：", WS\_CHILD | WS\_VISIBLE, 50, 80, 100, 20, hSecondPage4, NULL, hInst, NULL);

hTextBoxa1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

50, 110, 100, 20, hSecondPage4, NULL, hInst, NULL);

hSubmitButton1 = CreateWindowW(L"BUTTON", L"姓名查询", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage4, (HMENU)14, hInst, NULL);

hSubmitButton2 = CreateWindowW(L"BUTTON", L"ID查询", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 200, 100, 30, hSecondPage4, (HMENU)15, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage4, SW\_SHOW);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage4, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage4;

}

break;

case 5:

{

// 创建第二个页面窗口

hSecondPage5 = CreateWindowW(szWindowClass, L"Second Page", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

hButton1 = CreateWindowW(L"BUTTON", L"语文", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage5, (HMENU)16, hInst, NULL);

hButton2 = CreateWindowW(L"BUTTON", L"数学", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 200, 100, 30, hSecondPage5, (HMENU)17, hInst, NULL);

hButton3 = CreateWindowW(L"BUTTON", L"英语", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 250, 100, 30, hSecondPage5, (HMENU)18, hInst, NULL);

HWND hLabel1 = CreateWindowW(L"STATIC", L"最小", WS\_CHILD | WS\_VISIBLE, 50, 90, 50, 20, hSecondPage5, NULL, hInst, NULL);

HWND hLabel2 = CreateWindowW(L"STATIC", L"最大", WS\_CHILD | WS\_VISIBLE, 100, 90, 50, 20, hSecondPage5, NULL, hInst, NULL);

hTextBoxa3 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

50, 110, 50, 20, hSecondPage5, NULL, hInst, NULL);

hTextBoxa4 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

100, 110, 50, 20, hSecondPage5, NULL, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage5, SW\_SHOW);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage5, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage5;

}

break;

case 6:

{

// 创建第二个页面窗口

hSecondPage6 = CreateWindowW(szWindowClass, L"学生排序", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

// 创建 Chinese 单选按钮

hChineseRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"语文",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

200, 50, 100, 30,

hSecondPage6,

(HMENU)30, // 修改为不同的标识符

hInst,

NULL);

// 创建 Math 单选按钮

hMathRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"数学",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

200, 80, 100, 30,

hSecondPage6,

(HMENU)31, // 修改为不同的标识符

hInst,

NULL);

// 创建 English 单选按钮

hEnglishRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"英语",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

200, 110, 100, 30,

hSecondPage6,

(HMENU)32, // 修改为不同的标识符

hInst,

NULL);

// 创建 Ascending 单选按钮

hAscendingRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"升序",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

200, 150, 100, 30,

hSecondPage6,

(HMENU)40, // 修改为不同的标识符

hInst,

NULL);

// 创建 Descending 单选按钮

hDescendingRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"降序",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

200, 180, 100, 30,

hSecondPage6,

(HMENU)41, // 修改为不同的标识符

hInst,

NULL);

hSubmitButton = CreateWindowW(L"BUTTON", L"确认", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage6, (HMENU)19, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage6, SW\_SHOW);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage6, (HMENU)10, hInst, NULL);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage6;

}

break;

case 7:

{

read\_student\_file(students);

fresh\_list();

}

break;

case 8:

{

// 创建第二个页面窗口

hSecondPage8 = CreateWindowW(szWindowClass, L"Second Page", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

// 创建 Chinese 单选按钮

hChineseRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"语文",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

200, 50, 100, 30,

hSecondPage8,

(HMENU)33, // 修改为不同的标识符

hInst,

NULL);

// 创建 Math 单选按钮

hMathRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"数学",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

200, 80, 100, 30,

hSecondPage8,

(HMENU)34, // 修改为不同的标识符

hInst,

NULL);

// 创建 English 单选按钮

hEnglishRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"英语",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

200, 110, 100, 30,

hSecondPage8,

(HMENU)35, // 修改为不同的标识符

hInst,

NULL);

// 创建返回按钮

hReturnButton = CreateWindowW(L"BUTTON", L"返回", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 50, 100, 30, hSecondPage8, (HMENU)10, hInst, NULL);

hSubmitButton = CreateWindowW(L"BUTTON", L"确认", WS\_TABSTOP | WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

50, 150, 100, 30, hSecondPage8, (HMENU)20, hInst, NULL);

// 隐藏当前页面并显示第二个页面

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hSecondPage8, SW\_SHOW);

// 更新当前页面变量为第二个页面

hCurrentPage = hSecondPage8;

}

break;

case 9:

{

write\_student\_file("students.txt", students);

MessageBox(hSecondPage8, L"修改已保存，点击确认后退出", L"提示", MB\_OK);

exit(0);

}

break;

case 10:

{

// 隐藏当前页面并显示主窗口

ShowWindow(hCurrentPage, SW\_HIDE);

ShowWindow(hMainPage, SW\_SHOW);

// 更新当前页面变量为主窗口

hCurrentPage = hMainPage;

fresh\_list();

// 销毁返回按钮

DestroyWindow(hReturnButton);

}

break;

case 11: // 处理sub按钮的点击事件

{

// 获取第一个输入框的文本

int textLength = GetWindowTextLengthW(hTextBoxa1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa1, buffer, textLength + 1);

std::wstring inputText1 = buffer;

delete[] buffer;

// 获取第二个输入框的文本

textLength = GetWindowTextLengthW(hTextBoxa2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa2, buffer, textLength + 1);

std::wstring inputText2 = buffer;

delete[] buffer;

// 获取第三个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa3);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa3, buffer, textLength + 1);

int inputValue3 = \_wtoi(buffer);

delete[] buffer;

// 获取第四个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa4, buffer, textLength + 1);

int inputValue4 = \_wtoi(buffer);

delete[] buffer;

// 获取第五个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa5);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa5, buffer, textLength + 1);

int inputValue5 = \_wtoi(buffer);

delete[] buffer;

// 创建保存的消息文本

std::wstring messageText = L"信息已保存：\n";

messageText += L"姓名: " + inputText1 + L"\n";

messageText += L"ID: " + inputText2 + L"\n";

messageText += L"语文: " + std::to\_wstring(inputValue3) + L"\n";

messageText += L"数学: " + std::to\_wstring(inputValue4) + L"\n";

messageText += L"英语: " + std::to\_wstring(inputValue5);

// 显示消息框

MessageBoxW(hCurrentPage, messageText.c\_str(), L"Saved Data", MB\_OK);

students.push\_back({ wstring2string(inputText1), wstring2string(inputText2), inputValue3, inputValue4, inputValue5 });

fresh\_list();

}

break;

case 12:

{

int textLength = GetWindowTextLengthW(hTextBoxa1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

textLength = GetWindowTextLengthW(hTextBoxa1);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa1, buffer, textLength + 1);

std::wstring inputText2 = buffer;

delete[] buffer;

delete\_student(students, wstring2string(inputText2));

fresh\_list();

}

break;

case 13:

{

int textLength = GetWindowTextLengthW(hTextBoxa1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

textLength = GetWindowTextLengthW(hTextBoxa1);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa1, buffer, textLength + 1);

std::wstring id = buffer;

delete[] buffer;

auto s = find\_student\_by\_id(students, wstring2string(id));

if (s == nullptr)

{

cout << "Error: student " << wstring2string(id) << " not found." << endl;

break;

}

// 获取第二个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa2, buffer, textLength + 1);

int chinese = \_wtoi(buffer);

delete[] buffer;

// 获取第三个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa3);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa3, buffer, textLength + 1);

int math = \_wtoi(buffer);

delete[] buffer;

// 获取第四个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa4, buffer, textLength + 1);

int english = \_wtoi(buffer);

delete[] buffer;

if (chinese != -1)

{

s->chinese = chinese;

}

if (math != -1)

{

s->math = math;

}

if (english != -1)

{

s->english = english;

}

cout << "Student " << s->name << " modified." << endl;

fresh\_list();

}

break;

case 14:

{

int textLength = GetWindowTextLengthW(hTextBoxa1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa1, buffer, textLength + 1);

std::wstring inputText1 = buffer;

delete[] buffer;

string name = wstring2string(inputText1);

cout << "Enter student name: ";

vector<Student> result = find\_student\_by\_name(students, name);

if (result.empty())

{

cout << "Error: student " << name << " not found." << endl;

break;

}

// 输出学生信息

// 创建保存的消息文本

string messageText = "Name\tID\tChinese\tMath\tEnglish\n";

for (const auto& s : result)

{

messageText += s.name + "\t" + s.id + "\t" + to\_string(s.chinese) + "\t" + to\_string(s.math) + "\t" + to\_string(s.english) + "\n";

}

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage4,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

cout << "Name\tID\tChinese\tMath\tEnglish" << endl;

}

break;

case 15:

{

int textLength = GetWindowTextLengthW(hTextBoxa1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa1, buffer, textLength + 1);

std::wstring inputText1 = buffer;

delete[] buffer;

string id = wstring2string(inputText1);

cout << "Enter student name: ";

Student\* result = find\_student\_by\_id(students, id);

if (result == NULL)

{

cout << "Error: student " << id << " not found." << endl;

break;

}

// 输出学生信息

// 创建保存的消息文本

string messageText = "Name\tID\tChinese\tMath\tEnglish\n";

messageText += result->name + "\t" + result->id + "\t" + to\_string(result->chinese) + "\t" + to\_string(result->math) + "\t" + to\_string(result->english) + "\n";

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage4,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

}

break;

case 16:

{

int count = 0;

// 获取第三个输入框的整数

int textLength = GetWindowTextLengthW(hTextBoxa3);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa3, buffer, textLength + 1);

int low = \_wtoi(buffer);

delete[] buffer;

// 获取第四个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa4, buffer, textLength + 1);

int high = \_wtoi(buffer);

delete[] buffer;

for (const Student& s : students)

{

int score = s.chinese;

if (score >= low && score <= high)

{

count++;

}

}

string messageText = "总共有" + to\_string(count) + "个学生的语文成绩在" + to\_string(low) + "和" + to\_string(high) + "之间";

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage5,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

}

break;

case 17:

{

int count = 0;

// 获取第三个输入框的整数

int textLength = GetWindowTextLengthW(hTextBoxa3);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa3, buffer, textLength + 1);

int low = \_wtoi(buffer);

delete[] buffer;

// 获取第四个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa4, buffer, textLength + 1);

int high = \_wtoi(buffer);

delete[] buffer;

for (const Student& s : students)

{

int score = s.math;

if (score >= low && score <= high)

{

count++;

}

}

string messageText = "总共有" + to\_string(count) + "个学生的数学成绩在" + to\_string(low) + "和" + to\_string(high) + "之间";

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage5,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

}

break;

case 18:

{

int count = 0;

// 获取第三个输入框的整数

int textLength = GetWindowTextLengthW(hTextBoxa3);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa3, buffer, textLength + 1);

int low = \_wtoi(buffer);

delete[] buffer;

// 获取第四个输入框的整数

textLength = GetWindowTextLengthW(hTextBoxa4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBoxa4, buffer, textLength + 1);

int high = \_wtoi(buffer);

delete[] buffer;

for (const Student& s : students)

{

int score = s.english;

if (score >= low && score <= high)

{

count++;

}

}

string messageText = "总共有" + to\_string(count) + "个学生的英语成绩在" + to\_string(low) + "和" + to\_string(high) + "之间";

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage5,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

}

break;

case 19:

{

if (choice\_class == 0)

{

if (choice\_sort == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.chinese < b.chinese; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.chinese > b.chinese; });

}

}

else if (choice\_class == 1)

{

if (choice\_sort == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.math < b.math; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.math > b.math; });

}

}

else

{

if (choice\_sort == 0)

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.english < b.english; });

}

else

{

sort(students.begin(), students.end(), [](const Student& a, const Student& b)

{ return a.english > b.english; });

}

}

}

break;

case 20:

{

string course = choice\_class == 0 ? "语文" : choice\_class == 1 ? "数学" : "英语";

cout << "Enter course name: ";

//cin >> course;

double sum = 0;

for (const Student& s : students)

{

int score;

if (choice\_class == 0)

{

score = s.chinese;

}

else if (choice\_class == 1)

{

score = s.math;

}

else if (choice\_class == 2)

{

score = s.english;

}

else

{

cout << "Error: invalid course name." << endl;

break;

}

sum += score;

}

cout << "The average score of " << course << " ： " << sum / students.size() << "." << endl;

string messageText = "平均分 " + course + " ： " + to\_string(sum / students.size()) + ".";

// 显示消息框

// 创建标签

HWND hLabel = CreateWindowEx(

0,

L"STATIC",

L"",

WS\_CHILD | WS\_VISIBLE,

500, 50, 2000, 3000,

hSecondPage8,

NULL,

hInst,

NULL);

SetWindowText(hLabel, string2wstring(messageText).c\_str());

}

break;

case 30:

{

choice\_class = 0;

}

break;

case 31:

{

choice\_class = 1;

}

break;

case 32:

{

choice\_class = 2;

}

break;

case 33:

{

choice\_class = 0;

}

break;

case 34:

{

choice\_class = 1;

}

break;

case 35:

{

choice\_class = 2;

}

break;

case 40:

{

choice\_sort = 0;

}

break;

case 41:

{

choice\_sort = 1;

}

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 在此处添加使用 hdc 的任何绘图代码...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

### 实验结果：

#### 打开文件：

图形用户界面, 表格

描述已自动生成

表格

描述已自动生成

#### 添加学生：

电脑萤幕的截图

描述已自动生成表格

描述已自动生成

#### 删除学生：

形状

描述已自动生成

表格

描述已自动生成

#### 修改信息：

图片包含 图形用户界面

描述已自动生成表格

描述已自动生成

#### 搜索信息：

姓名查询：

图形用户界面, 应用程序

描述已自动生成

ID查询：

图形用户界面, 应用程序

描述已自动生成

#### 成绩统计（数学90-100）：

图形用户界面, 应用程序, Word

描述已自动生成

#### 学生排序（数学，降序）：

形状

低可信度描述已自动生成表格

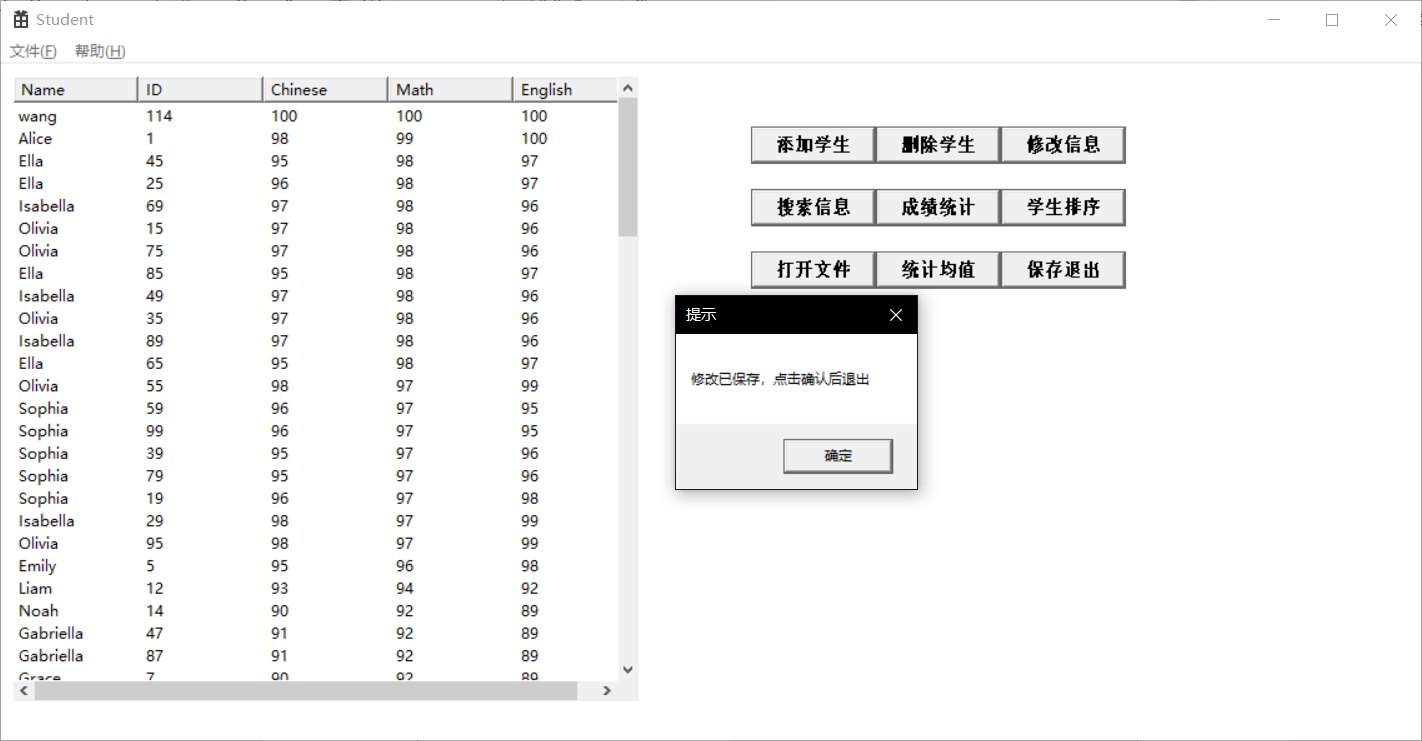
描述已自动生成

#### 统计均值（数学）：

图形用户界面, 应用程序

描述已自动生成

#### 保存退出：



### 思考与扩展：

## 题目3 通信录管理系统

### 实验描述：

用 C/C++设计出模拟手机通信录管理系统，实现对手机中的通信录进行管理。

### 实验代码：

#### AddressBook.h

#pragma once

#include <CommCtrl.h>

#include <iostream>

#include <string>

// 在头文件中包含Windows API

#include <Windows.h>

#include <bits/stdc++.h>

#include "resource.h"

#include <Commdlg.h>

#define IDM\_VIEW 1

#define IDM\_ADD 2

#define IDM\_DIAL 3

#define IDM\_MODIFY 4

#define IDM\_DELETE 5

#define IDM\_SAVE 7

#define IDM\_READ 8

#define IDM\_CHANGE 9

#define IDM\_FIND 10

#define MAX\_LOADSTRING 100

#define IDM\_CONFIRM 11

HWND hWnd;

HINSTANCE hInst; // 当前实例

WCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本

WCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

const int COLUMN\_COUNT = 4;

const wchar\_t\* columnHeaders[COLUMN\_COUNT] = { L"姓名", L"号码", L"类别", L"邮箱"};

HWND hListView;

struct Contact

{

std::string name, phoneNumber, category,email;

Contact\* next;

};

HWND hOfficialRadioButton, hIndividualRadioButton, hBusinessRadioButton;

HWND hTextBox1, hTextBox2, hTextBox3, hTextBox4, hButton1;

int mode = 0;

//HWND hCurrentPage;

//HWND hSecondPage;

#### AddressBook.cpp

// AddressBook.cpp : 定义应用程序的入口点。

//

#include "framework.h"

#include "AddressBook.h"

using namespace std;

string wstring2string(wstring wstr)

{

string result;

// 获取缓冲区大小，并申请空间，缓冲区大小事按字节计算的

int len = WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), NULL, 0, NULL, NULL);

char\* buffer = new char[len + 1];

// 宽字节编码转换成多字节编码

WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), buffer, len, NULL, NULL);

buffer[len] = '\0';

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

wstring string2wstring(string str)

{

wstring result;

// 获取缓冲区大小，并申请空间，缓冲区大小按字符计算

int len = MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), NULL, 0);

TCHAR\* buffer = new TCHAR[len + 1];

// 多字节编码转换成宽字节编码

MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), buffer, len);

buffer[len] = '\0'; // 添加字符串结尾

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

// 通信录管理系统

//void write\_student\_file(string filename)

//{

// ofstream outfile(filename);

// if (!outfile.is\_open())

// {

// cerr << "Error: can't open file " << filename << endl;

// return;

// }

//

// for (const auto& s : students)

// {

// outfile << s.name << " , " << s.id << " , " << s.chinese << " , " << s.math << " , " << s.english << endl;

// }

//

// outfile.close();

//}

class ContactManager

{

public:

Contact\* head;

int itemCount;

ContactManager()

{

head = nullptr;

itemCount = 0;

}

// 查看通信录

void viewContacts()

{

if (head == nullptr)

{

cout << "通信录为空\n";

return;

}

Contact\* current = head;

while (current != nullptr)

{

cout << "姓名: " << current->name << endl;

cout << "电话号码: " << current->phoneNumber << endl;

cout << "分类: " << current->category << endl;

cout << "电子邮件: " << current->email << endl;

cout << endl;

current = current->next;

}

}

// 增加联系人

void addContact(string name, string phoneNumber, string category, string email)

{

if (itemCount >= 15)

{

cout << "存储空间已满，不能再录入新数据\n";

return;

}

/\*cout << "请输入姓名：";

string name;

cin >> name;

cout << "请输入电话号码：";

string phoneNumber;

cin >> phoneNumber;

cout << "请选择分类（A-办公类，B-个人类，C-商务类）：";

char categoryChoice;

cin >> categoryChoice;

string category;

switch (categoryChoice)

{

case 'A':

category = "办公类";

break;

case 'B':

category = "个人类";

break;

case 'C':

category = "商务类";

break;

default:

cout << "无效的选择\n";

return;

}

cout << "请输入电子邮件：";

string email;

cin >> email;\*/

// 检查是否已存在相同姓名和电话号码的联系人

Contact\* current = head;

while (current != nullptr)

{

if (current->name == name && current->phoneNumber == phoneNumber)

{

cout << "数据录入重复\n";

return;

}

current = current->next;

}

// 创建新的联系人节点

Contact\* newContact = new Contact;

newContact->name = name;

newContact->phoneNumber = phoneNumber;

newContact->category = category;

newContact->email = email;

newContact->next = nullptr;

// 按递增顺序插入联系人

if (head == nullptr || name < head->name)

{

newContact->next = head;

head = newContact;

}

else

{

Contact\* prev = nullptr;

current = head;

while (current != nullptr && name > current->name)

{

prev = current;

current = current->next;

}

prev->next = newContact;

newContact->next = current;

}

cout << "联系人已添加\n";

itemCount++;

}

// 拔号

string dialContact(string name)

{

if (head == nullptr)

{

cout << "通信录为空\n";

return "";

}

cout << "通信录中的联系人：\n";

Contact\* current = head;

while (current != nullptr)

{

cout << current->name << endl;

current = current->next;

}

cout << "请选择联系人姓名：";

//string name;

//cin >> name;

current = head;

while (current != nullptr)

{

if (current->name == name)

{

//cout << "拨号中...\n";

//for (char c : current->phoneNumber)

//{

// cout << c;

// // 在代码中播放滴滴声音

// MessageBeep(MB\_ICONASTERISK);

// Sleep(500); // 延迟显示每个数字的效果

//}

//cout << endl;

return current->phoneNumber;

}

current = current->next;

}

cout << "未找到联系人\n";

return "";

}

// 修改联系人

void modifyContact(string name,string phoneNumber)

{

if (head == nullptr)

{

cout << "通信录为空\n";

return;

}

cout << "通信录中的联系人：\n";

Contact\* current = head;

while (current != nullptr)

{

cout << current->name << endl;

current = current->next;

}

cout << "请选择要修改的联系人姓名：";

current = head;

while (current != nullptr)

{

if (current->name == name)

{

cout << "请输入新的电话号码：";

current->phoneNumber = phoneNumber;

cout << "联系人已修改\n";

return;

}

current = current->next;

}

cout << "未找到联系人\n";

}

// 删除联系人

void deleteContact(string name)

{

if (head == nullptr)

{

cout << "通信录为空\n";

return;

}

cout << "通信录中的联系人：\n";

Contact\* current = head;

while (current != nullptr)

{

cout << current->name << endl;

current = current->next;

}

cout << "请选择要删除的联系人姓名：";

//string name;

//cin >> name;

if (head->name == name)

{

Contact\* temp = head;

head = head->next;

delete temp;

cout << "联系人已删除\n";

itemCount--;

return;

}

Contact\* prev = head;

current = head->next;

while (current != nullptr)

{

if (current->name == name)

{

prev->next = current->next;

delete current;

cout << "联系人已删除\n";

itemCount--;

return;

}

prev = current;

current = current->next;

}

cout << "未找到联系人\n";

}

};

// 全局变量:

// 此代码模块中包含的函数的前向声明:

// 通信录管理系统对象

ContactManager manager;

void read\_contact\_file()

{

OPENFILENAMEA ofn;

CHAR szFile[MAX\_PATH];

ZeroMemory(&ofn, sizeof(ofn));

ZeroMemory(szFile, sizeof(szFile));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFilter = "Text Files (\*.txt)\0\*.txt\0All Files (\*.\*)\0\*.\*\0";

ofn.lpstrFile = szFile;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_EXPLORER | OFN\_FILEMUSTEXIST | OFN\_HIDEREADONLY;

ofn.lpstrDefExt = "txt";

if (GetOpenFileNameA(&ofn))

{

std::ifstream infile(ofn.lpstrFile);

if (!infile.is\_open())

{

std::cerr << "Error: can't open file " << ofn.lpstrFile << std::endl;

return;

}

std::string line;

while (std::getline(infile, line))

{

std::istringstream iss(line);

std::string name, phoneNumber;

string category;

std::string email;

std::string dummy;

if (!(iss >> name >> dummy >> phoneNumber >> dummy >> category >> dummy >> email))

{

std::cerr << "Error: invalid student file format" << std::endl;

return;

}

manager.addContact(name, phoneNumber, category, email);

}

infile.close();

}

return;

}

void SaveContactsToFile(HWND hWnd, Contact\* head)

{

OPENFILENAME ofn = { 0 };

wchar\_t filePath[MAX\_PATH] = { 0 };

// 初始化 OPENFILENAME 结构体

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hWnd;

ofn.lpstrFile = filePath;

ofn.nMaxFile = MAX\_PATH;

ofn.lpstrDefExt = L"txt";

ofn.lpstrFilter = L"Text Files (\*.txt)\0\*.txt\0All Files (\*.\*)\0\*.\*\0";

ofn.lpstrTitle = L"Save As";

ofn.Flags = OFN\_OVERWRITEPROMPT | OFN\_PATHMUSTEXIST;

// 打开文件选择框

if (GetSaveFileName(&ofn))

{

std::ofstream file(ofn.lpstrFile);

if (file.is\_open())

{

Contact\* current = head;

while (current != nullptr)

{

file << current->name << " , " << current->phoneNumber << " , " << current->category <<" , " << current->email << endl;

current = current->next;

}

file.close();

}

else

{

// 文件无法打开，处理错误

}

}

}

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// 初始化全局字符串

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_ADDRESSBOOK, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// 执行应用程序初始化:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_ADDRESSBOOK));

MSG msg;

// 主消息循环:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// 函数: MyRegisterClass()

//

// 目标: 注册窗口类。

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_ADDRESSBOOK));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_ADDRESSBOOK);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// 函数: InitInstance(HINSTANCE, int)

//

// 目标: 保存实例句柄并创建主窗口

//

// 注释:

//

// 在此函数中，我们在全局变量中保存实例句柄并

// 创建和显示主程序窗口。

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // 将实例句柄存储在全局变量中

hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

// 创建一个按钮

HWND hButton = CreateWindow(TEXT("button"), TEXT("添加联系人"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 10, 100, 30, hWnd, (HMENU)IDM\_ADD, hInstance, NULL);

HWND hButton2 = CreateWindow(TEXT("button"), TEXT("导入联系人"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 50, 100, 30, hWnd, (HMENU)IDM\_VIEW, hInstance, NULL);

HWND hButton3 = CreateWindow(TEXT("button"), TEXT("删除联系人"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 90, 100, 30, hWnd, (HMENU)IDM\_DELETE, hInstance, NULL);

HWND hButton4 = CreateWindow(TEXT("button"), TEXT("修改联系人"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 130, 100, 30, hWnd, (HMENU)IDM\_MODIFY, hInstance, NULL);

HWND hButton5 = CreateWindow(TEXT("button"), TEXT("保存并退出"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 170, 100, 30, hWnd, (HMENU)IDM\_EXIT, hInstance, NULL);

HWND hButton6 = CreateWindow(TEXT("button"), TEXT("拨号联系人"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

600, 210, 100, 30, hWnd, (HMENU)IDM\_DIAL, hInstance, NULL);

hListView = CreateWindowW(WC\_LISTVIEW, L"", WS\_VISIBLE | WS\_CHILD | LVS\_REPORT, 10, 10, 550, 500, hWnd, nullptr, hInstance, nullptr);

// 设置列表视图控件样式

ListView\_SetExtendedListViewStyle(hListView, LVS\_EX\_FULLROWSELECT);

// 添加列表视图控件的列

LVCOLUMN lvColumn;

lvColumn.mask = LVCF\_TEXT | LVCF\_WIDTH;

lvColumn.cx = 150;

for (int i = 0; i < COLUMN\_COUNT; i++)

{

lvColumn.pszText = const\_cast<wchar\_t\*>(columnHeaders[i]);

ListView\_InsertColumn(hListView, i, &lvColumn);

}

if (!hWnd)

{

return FALSE;

}

// hCurrentPage = hWnd; // 设置当前页面为主窗口

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// 函数: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// 目标: 处理主窗口的消息。

//

// WM\_COMMAND - 处理应用程序菜单

// WM\_PAINT - 绘制主窗口

// WM\_DESTROY - 发送退出消息并返回

//

//

void fresh\_list() {

// 清空列表视图控件

ListView\_DeleteAllItems(hListView);

Contact\* item = manager.head;

for (int i = 0; i < manager.itemCount; i++)

{

LVITEM lvItem;

lvItem.mask = LVIF\_TEXT;

lvItem.iItem = i;

lvItem.iSubItem = 0;

wstring name = string2wstring(item->name);

lvItem.pszText = &name[0]; // 插入姓名

// 插入第一列的数据

ListView\_InsertItem(hListView, &lvItem);

lvItem.iSubItem = 1;

std::wstring phoneNumber = string2wstring(item->phoneNumber);

lvItem.pszText = &phoneNumber[0]; // 插入电话

ListView\_SetItemText(hListView, i, 1, lvItem.pszText);

lvItem.iSubItem = 2;

std::wstring category = string2wstring(item->category);

lvItem.pszText = &category[0]; // 插入学号

ListView\_SetItemText(hListView, i, 2, lvItem.pszText);

lvItem.iSubItem = 3;

std::wstring email = string2wstring(item->email);

lvItem.pszText = &email[0]; // 插入学号

ListView\_SetItemText(hListView, i, 3, lvItem.pszText);

item = item->next;

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// 分析菜单选择:

switch (wmId)

{

case IDM\_VIEW:

{

// manager.viewContacts();

read\_contact\_file();

// 将manager中的联系人信息显示到列表视图控件中

fresh\_list();

}

break;

case IDM\_ADD:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 110, 50, 20, hWnd, NULL, hInst, NULL);

hTextBox2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 130, 50, 20, hWnd, NULL, hInst, NULL);

hTextBox3 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 150, 50, 20, hWnd, NULL, hInst, NULL);

hOfficialRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"办公类",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

800, 200, 100, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hIndividualRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"个人类",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

800, 230, 100, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hBusinessRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"商务类",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

800, 260, 100, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

// 创建确认按钮

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_ADD;

}

break;

case IDM\_DIAL:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 110, 50, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_DIAL;

}

break;

case IDM\_MODIFY:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 110, 50, 20, hWnd, NULL, hInst, NULL);

hTextBox2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 130, 50, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

//manager.modifyContact();

mode = IDM\_MODIFY;

}

break;

case IDM\_DELETE:

{

// manager.deleteContact();

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 110, 50, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_DELETE;

}

break;

case IDM\_EXIT:

{

SaveContactsToFile(hWnd, manager.head);

DestroyWindow(hWnd);

}

break;

case IDM\_CONFIRM:

{

switch (mode)

{

case IDM\_ADD:

{

string name, phoneNumber, category, email;

int officialCheckState = SendMessage(hOfficialRadioButton, BM\_GETCHECK, 0, 0);

int individualCheckState = SendMessage(hIndividualRadioButton, BM\_GETCHECK, 0, 0);

int businessCheckState = SendMessage(hBusinessRadioButton, BM\_GETCHECK, 0, 0);

if (officialCheckState == BST\_CHECKED)

{

category = "Official";

}

else if (individualCheckState == BST\_CHECKED)

{

category = "Individual";

}

else if (businessCheckState == BST\_CHECKED)

{

category = "Business";

}

// 获取第一个输入框的文本

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

name = wstring2string(buffer);

delete[] buffer;

// 获取第二个输入框的文本

textLength = GetWindowTextLengthW(hTextBox2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox2, buffer, textLength + 1);

phoneNumber = wstring2string(buffer);

delete[] buffer;

// 获取第三个输入框的文本

textLength = GetWindowTextLengthW(hTextBox3);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox3, buffer, textLength + 1);

email = wstring2string(buffer);

delete[] buffer;

manager.addContact(name, phoneNumber, category, email);

}

break;

case IDM\_DELETE:

{

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string name = wstring2string(buffer);

delete[] buffer;

DestroyWindow(hTextBox1);

DestroyWindow(hButton1);

manager.deleteContact(name);

}

break;

case IDM\_MODIFY: {

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string name = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox2, buffer, textLength + 1);

string phoneNumber = wstring2string(buffer);

delete[] buffer;

manager.modifyContact(name,phoneNumber);

}break;

case IDM\_DIAL:

{

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string name = wstring2string(buffer);

delete[] buffer;

string phoneNumber=manager.dialContact(name);

HWND hSecondPage1 = CreateWindowW(szWindowClass, L"拨号中", WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInst, nullptr);

ShowWindow(hSecondPage1, SW\_SHOW);

HWND label = CreateWindowW(L"STATIC", L"拨号中...\n", WS\_CHILD | WS\_VISIBLE,

100, 100, 500, 500, hSecondPage1, NULL, hInst, NULL);

//以打字机的方式逐个将phoneNumber中的字符追加到label中

for (int i = 0; i < phoneNumber.length(); i++) {

Sleep(1000);

wchar\_t temp[2] = { phoneNumber[i], '\0' };

MessageBeep(MB\_OK);

// 获取当前标签控件的文本长度

int textLength = SendMessage(label, WM\_GETTEXTLENGTH, 0, 0);

// 为新文本分配内存，并将原有文本复制到新内存中

wchar\_t\* newText = new wchar\_t[textLength + 2];

SendMessage(label, WM\_GETTEXT, textLength + 1, (LPARAM)newText);

// 追加新字符到新文本末尾

newText[textLength] = temp[0];

newText[textLength + 1] = '\0';

// 设置标签控件的新文本

SendMessage(label, WM\_SETTEXT, 0, (LPARAM)newText);

// 释放内存

delete[] newText;

}

}break;

}

DestroyWindow(hTextBox1);

DestroyWindow(hTextBox2);

DestroyWindow(hTextBox3);

DestroyWindow(hOfficialRadioButton);

DestroyWindow(hIndividualRadioButton);

DestroyWindow(hBusinessRadioButton);

DestroyWindow(hButton1);

fresh\_list();

}

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 在此处添加使用 hdc 的任何绘图代码...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

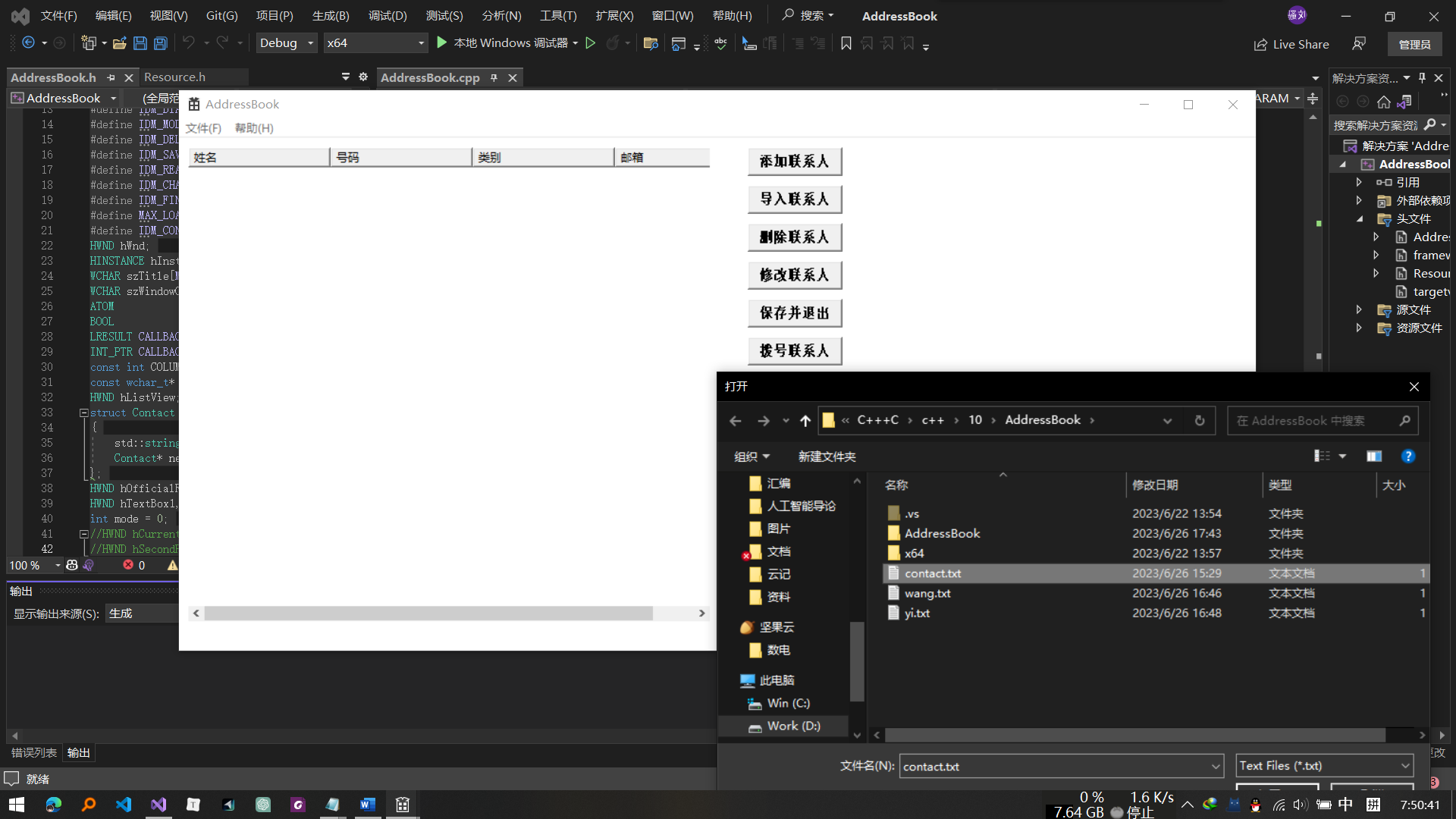
}

return 0;

}

### 实验结果：

#### 导入联系人：



表格

描述已自动生成

#### 添加联系人：

表格

描述已自动生成

表格

描述已自动生成

#### 删除联系人：

图形用户界面, 应用程序, 表格

描述已自动生成表格

描述已自动生成

#### 修改联系人：

图形用户界面, 应用程序, 表格

描述已自动生成表格

描述已自动生成

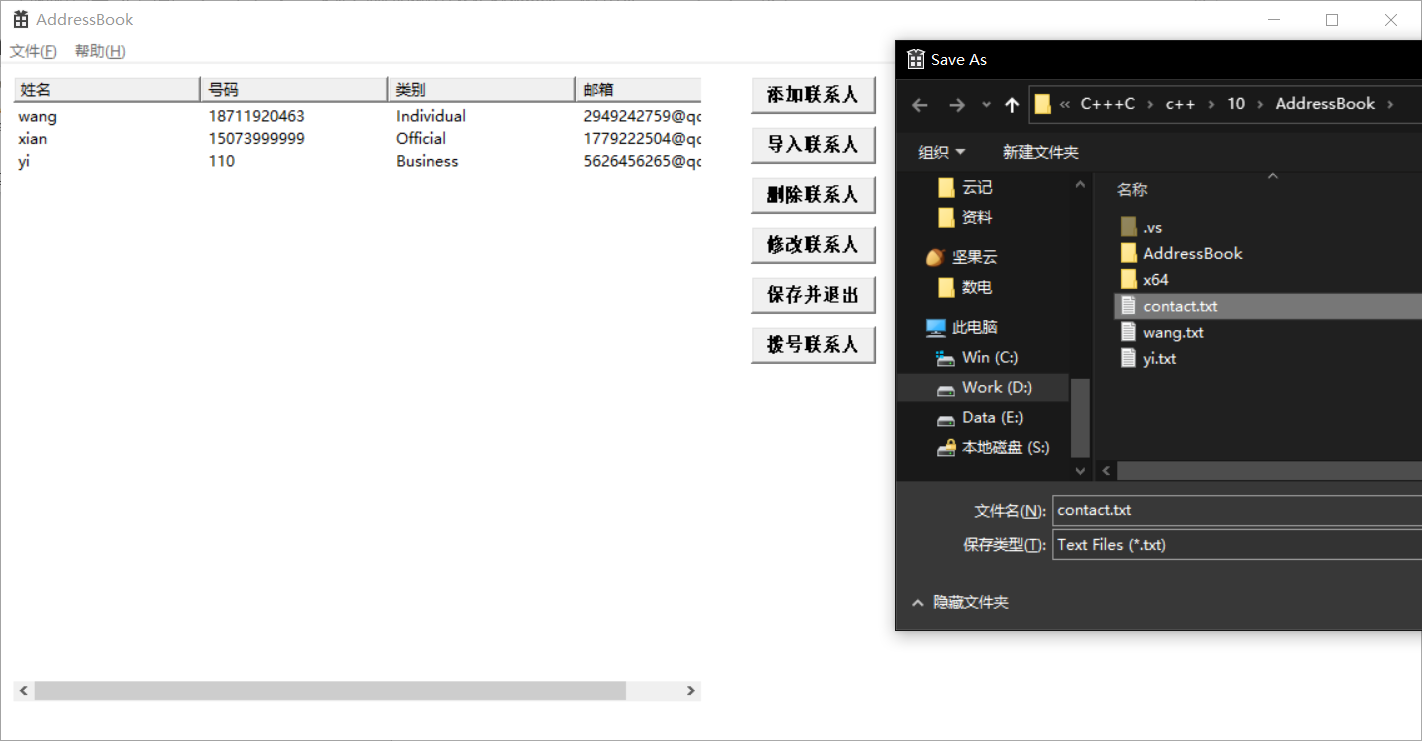
拨号联系人：

图形用户界面, 应用程序, 表格

描述已自动生成图形用户界面, 应用程序, Word

描述已自动生成

#### 保存并退出：

文本

描述已自动生成

### 思考与扩展：

## 题目4 学生管理系统

### 实验描述：

使用下面的数据，用 C/C++设计一个简单的学籍管理系统，实现出最基本的功能。

学生基本信息文件(A.TXT)及其内容：A.TXT 文件不需要编程录入数据，可用文本编辑工具

直接生成

学号 姓名 性别 宿舍号码电话号码

01 张成成男 501 87732111

02 李成华女 101 87723112

03 王成凤女 101 87723112

04 张明明男 502 87734333

05 陈东 男 501 87732111

06 李果 男 502 87734333

07 张园园女 102 87756122

… …. .. … ………..

学生成绩基本信息文件(B.TXT)及其内容：

学号 课程编号课程名称 学分 平时成绩实验成绩 卷面成绩 综合成绩

01 A01 大学物理 3 66 78 82

02 B03 高等数学 4 78 -1 90

01 B03 高等数学 4 45 -1 88

02 C01 VF 3 65 76 66

… …. ………. .. .. …

### 实验代码：

StudentScore.h

#pragma once

#include <CommCtrl.h>

#include <iostream>

#include <string>

// 在头文件中包含Windows API

#include <Windows.h>

#include <bits/stdc++.h>

#include "resource.h"

#include <Commdlg.h>

using namespace std;

#define IDM\_STUSEARCH 1

#define IDM\_ADD 2

#define IDM\_EXIT 3

#define IDM\_MODIFY 4

#define IDM\_DELETE 5

#define IDM\_SAVE 7

#define IDM\_SORT 8

#define IDM\_CHANGE 9

#define IDM\_SCOSEARCH 10

#define IDM\_LOAD 12

#define MAX\_LOADSTRING 100

#define IDM\_CONFIRM 11

HWND hWnd;

HINSTANCE hInst; // 当前实例

WCHAR szTitle[MAX\_LOADSTRING]; // 标题栏文本

WCHAR szWindowClass[MAX\_LOADSTRING]; // 主窗口类名

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

const int COLUMN\_COUNT1 = 5;

const int COLUMN\_COUNT2 = 8;

const wchar\_t\* columnHeaders1[COLUMN\_COUNT1] = { L"学号", L"姓名", L"性别", L"宿舍号码" ,L"电话号码" };

const wchar\_t\* columnHeaders2[COLUMN\_COUNT2] = { L"学号", L"课程编号", L"课程名称", L"学分",L"平时成绩",L"实验成绩",L"卷面成绩",L"综合成绩" };

HWND hListView1, hListView2;

string f1, f2;

// 学生基本信息结构体

struct StudentInfo {

std::string studentID;

std::string name;

std::string gender;

std::string dormitory;

std::string phoneNumber;

};

// 学生成绩结构体

struct StudentScore {

std::string studentID;

std::string courseCode;

std::string courseName;

int credit;

int regularScore;

int labScore;

int examScore;

double compositeScore;

};

// 函数声明

void loadData(string f1, string f2);

void saveData();

void inputData(string studentID, string courseCode, string courseName, int credit, int regularScore, int labScore, int examScore);

string printStudentInfo(const StudentInfo& student);

string printScoreInfo(const StudentScore& score);

string searchStudentInfo(int choice, string StuID, string StuName);

string searchScoreInfo(string stuID);

void deleteStudent(string studentID);

bool compareByCompositeScore(const StudentScore& score1, const StudentScore& score2);

bool compareByCredit(const StudentScore& score1, const StudentScore& score2);

void sortScores(int choice);

vector<StudentInfo> students;

vector<StudentScore> scores;

HWND hStuIDRadioButton, hStuNameRadioButton, hBusinessRadioButton;

HWND hTextBox1, hTextBox2, hTextBox3, hTextBox4, hTextBox5, hTextBox6, hTextBox7, hButton1;

HWND hlabel1, hlabel2, hlabel3, hlabel4, hlabel5, hlabel6, hlabel7;

int mode = 0;

StudentScore.cpp

// AddressBook.cpp : 定义应用程序的入口点。

//

#include "framework.h"

#include "StudentScore.h"

using namespace std;

// 加载数据

void loadData(string f1,string f2) {

ifstream studentFile(f1);

ifstream scoreFile(f2);

if (!studentFile.is\_open() || !scoreFile.is\_open()) {

cout << "无法打开文件。" << endl;

return;

}

// 加载学生基本信息

while (!studentFile.eof()) {

StudentInfo student;

studentFile >> student.studentID >> student.name >> student.gender >> student.dormitory >> student.phoneNumber;

if (student.studentID == "")break;

students.push\_back(student);

}

// 加载学生成绩信息

while (!scoreFile.eof()) {

StudentScore score;

scoreFile >> score.studentID >> score.courseCode >> score.courseName >> score.credit >> score.regularScore >> score.labScore >> score.examScore;

if (score.studentID == "")break;

if (score.labScore == -1) {

score.compositeScore = score.regularScore \* 0.3 + score.examScore \* 0.7;

}

else {

score.compositeScore = score.regularScore \* 0.15 + score.labScore \* 0.15 + score.examScore \* 0.7;

}

scores.push\_back(score);

}

studentFile.close();

scoreFile.close();

}

// 保存数据

void saveData() {

ofstream studentFile(f1);

ofstream scoreFile(f2);

if (!studentFile.is\_open() || !scoreFile.is\_open()) {

cout << "无法保存数据到文件。" << endl;

return;

}

// 保存学生基本信息

for (const auto& student : students) {

studentFile << student.studentID << " " << student.name << " " << student.gender << " " << student.dormitory << " " << student.phoneNumber << endl;

}

// 保存学生成绩信息

for (const auto& score : scores) {

scoreFile << score.studentID << " " << score.courseCode << " " << score.courseName << " " << score.credit << " " << score.regularScore << " " << score.labScore << " " << score.examScore << endl;

}

studentFile.close();

scoreFile.close();

}

// 录入数据

void inputData(string studentID, string courseCode, string courseName, int credit, int regularScore, int labScore, int examScore) {

// 检查学号是否存在

auto it = find\_if(scores.begin(), scores.end(), [&studentID](const StudentScore& score) {

return score.studentID == studentID;

});

//if (it != scores.end()) {

// cout << "该学生已存在，请输入其他学号。" << endl;

// return;

//}

StudentScore score;

score.studentID = studentID;

score.courseCode = courseCode;

score.courseName = courseName;

score.credit = credit;

score.regularScore = regularScore;

score.labScore = labScore;

score.examScore = examScore;

if (labScore == -1) {

score.compositeScore = regularScore \* 0.3 + examScore \* 0.7;

}

else {

score.compositeScore = regularScore \* 0.15 + labScore \* 0.15 + examScore \* 0.7;

}

scores.push\_back(score);

}

// 打印学生基本信息

string printStudentInfo(const StudentInfo& student) {

string res = "";

res += "学号：" + student.studentID + " 姓名：" + student.name + " 性别：" + student.gender + " 宿舍号码：" + student.dormitory + " 电话号码：" + student.phoneNumber + "\n";

//cout << "学号：" << student.studentID << " 姓名：" << student.name << " 性别：" << student.gender << " 宿舍号码：" << student.dormitory << " 电话号码：" << student.phoneNumber << endl;

return res;

}

// 打印学生成绩信息

string printScoreInfo(const StudentScore& score) {

//cout << "学号：" << score.studentID << " 课程编号：" << score.courseCode << " 课程名称：" << score.courseName << " 综合成绩：" << score.compositeScore << " 实得学分：" << score.credit << endl;

string res = "";

res += "学号：" + score.studentID + " 课程编号：" + score.courseCode + " 课程名称：" + score.courseName + " 综合成绩：" + to\_string(score.compositeScore) + " 实得学分：" + to\_string(score.credit) + "\n";

return res;

}

// 学生基本情况查询

string searchStudentInfo(int choice, string StuID, string StuName) {

string key;

if (choice == 1) {

//cout << "请输入学号：";

key = StuID;

}

else if (choice == 2) {

//cout << "请输入姓名：";

key = StuName;

}

else {

cout << "无效的选项。" << endl;

return "";

}

bool found = false;

string message = "";

for (const auto& student : students) {

if (choice == 1 && student.studentID == key) {

message += printStudentInfo(student);

found = true;

break;

}

else if (choice == 2 && student.name == key) {

message += printStudentInfo(student);

found = true;

}

}

if (!found) {

cout << "未找到相关学生信息。" << endl;

}

return message;

}

// 成绩查询

string searchScoreInfo(string studentID) {

//string studentID;

//cout << "请输入学号：";

//cin >> studentID;

bool found = false;

double totalCredits = 0.0;

double totalCompositeScore = 0.0;

string res = "";

for (const auto& score : scores) {

if (score.studentID == studentID) {

res+=printScoreInfo(score);

totalCredits += score.credit;

totalCompositeScore += score.compositeScore;

found = true;

}

}

if (!found) {

//cout << "未找到该学生的成绩信息。" << endl;

res = "未找到该学生的成绩信息。";

}

else {

//cout << "共修：" << scores.size() << " 科，实得总学分为：" << totalCredits << endl;

res += "共修：" + to\_string(scores.size()) + " 科，实得总学分为：" + to\_string(totalCredits) + "\n";

}

return res;

}

// 删除学生信息

void deleteStudent(string studentID) {

//string studentID;

//cout << "请输入学号：";

//cin >> studentID;

// 删除学生基本信息

students.erase(remove\_if(students.begin(), students.end(), [&studentID](const StudentInfo& student) {

return student.studentID == studentID;

}), students.end());

// 删除学生成绩信息

scores.erase(remove\_if(scores.begin(), scores.end(), [&studentID](const StudentScore& score) {

return score.studentID == studentID;

}), scores.end());

}

// 按综合成绩升序排序

bool compareByCompositeScore(const StudentScore& score1, const StudentScore& score2) {

return score1.compositeScore < score2.compositeScore;

}

// 按实得学分升序排序

bool compareByCredit(const StudentScore& score1, const StudentScore& score2) {

return score1.credit < score2.credit;

}

// 排序功能

void sortScores(int choice) {

//int choice;

//cout << "请选择排序方式：" << endl;

//cout << "1. 按综合成绩升序排序" << endl;

//cout << "2. 按实得学分升序排序" << endl;

//cin >> choice;

if (choice == 1) {

sort(scores.begin(), scores.end(), compareByCompositeScore);

}

else if (choice == 2) {

sort(scores.begin(), scores.end(), compareByCredit);

}

else {

cout << "无效的选项。" << endl;

return;

}

cout << "排序完成。" << endl;

}

string wstring2string(wstring wstr)

{

string result;

// 获取缓冲区大小，并申请空间，缓冲区大小事按字节计算的

int len = WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), NULL, 0, NULL, NULL);

char\* buffer = new char[len + 1];

// 宽字节编码转换成多字节编码

WideCharToMultiByte(CP\_ACP, 0, wstr.c\_str(), wstr.size(), buffer, len, NULL, NULL);

buffer[len] = '\0';

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

wstring string2wstring(string str)

{

wstring result;

// 获取缓冲区大小，并申请空间，缓冲区大小按字符计算

int len = MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), NULL, 0);

TCHAR\* buffer = new TCHAR[len + 1];

// 多字节编码转换成宽字节编码

MultiByteToWideChar(CP\_ACP, 0, str.c\_str(), str.size(), buffer, len);

buffer[len] = '\0'; // 添加字符串结尾

// 删除缓冲区并返回值

result.append(buffer);

delete[] buffer;

return result;

}

// 通信录管理系统

// 全局变量:

// 此代码模块中包含的函数的前向声明:

string read\_contact\_file()

{

OPENFILENAMEA ofn;

CHAR szFile[MAX\_PATH];

ZeroMemory(&ofn, sizeof(ofn));

ZeroMemory(szFile, sizeof(szFile));

ofn.lStructSize = sizeof(ofn);

ofn.hwndOwner = NULL;

ofn.lpstrFilter = "Text Files (\*.txt)\0\*.txt\0All Files (\*.\*)\0\*.\*\0";

ofn.lpstrFile = szFile;

ofn.nMaxFile = MAX\_PATH;

ofn.Flags = OFN\_EXPLORER | OFN\_FILEMUSTEXIST | OFN\_HIDEREADONLY;

ofn.lpstrDefExt = "txt";

GetOpenFileNameA(&ofn);

string fname = ofn.lpstrFile;

return fname;

}

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// 初始化全局字符串

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_STUDENTSCORE, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// 执行应用程序初始化:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_STUDENTSCORE));

MSG msg;

// 主消息循环:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// 函数: MyRegisterClass()

//

// 目标: 注册窗口类。

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_STUDENTSCORE));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_STUDENTSCORE);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// 函数: InitInstance(HINSTANCE, int)

//

// 目标: 保存实例句柄并创建主窗口

//

// 注释:

//

// 在此函数中，我们在全局变量中保存实例句柄并

// 创建和显示主程序窗口。

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // 将实例句柄存储在全局变量中

hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

// 创建一个按钮

SetWindowLongPtr(hWnd, GWL\_STYLE, WS\_POPUP | WS\_VISIBLE);

SetWindowPos(hWnd, HWND\_TOP, 0, 0, GetSystemMetrics(SM\_CXSCREEN), GetSystemMetrics(SM\_CYSCREEN), SWP\_FRAMECHANGED);

HWND hButton = CreateWindow(TEXT("button"), TEXT("数据录入"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 10, 100, 30, hWnd, (HMENU)IDM\_ADD, hInstance, NULL);

HWND hButton2 = CreateWindow(TEXT("button"), TEXT("信息查询"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 50, 100, 30, hWnd, (HMENU)IDM\_STUSEARCH, hInstance, NULL);

HWND hButton3 = CreateWindow(TEXT("button"), TEXT("成绩查询"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 90, 100, 30, hWnd, (HMENU)IDM\_SCOSEARCH, hInstance, NULL);

HWND hButton4 = CreateWindow(TEXT("button"), TEXT("删除信息"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 130, 100, 30, hWnd, (HMENU)IDM\_DELETE, hInstance, NULL);

HWND hButton5 = CreateWindow(TEXT("button"), TEXT("排序功能"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 170, 100, 30, hWnd, (HMENU)IDM\_SORT, hInstance, NULL);

HWND hButton6 = CreateWindow(TEXT("button"), TEXT("保存并退出"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 210, 100, 30, hWnd, (HMENU)IDM\_EXIT, hInstance, NULL);

HWND hButton7 = CreateWindow(TEXT("button"), TEXT("导入信息"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 250, 100, 30, hWnd, (HMENU)IDM\_LOAD, hInstance, NULL);

hListView1 = CreateWindowW(WC\_LISTVIEW, L"", WS\_VISIBLE | WS\_CHILD | LVS\_REPORT, 10, 10, 750, 400, hWnd, nullptr, hInstance, nullptr);

// 设置列表视图控件样式

ListView\_SetExtendedListViewStyle(hListView1, LVS\_EX\_FULLROWSELECT);

// 添加列表视图控件的列

LVCOLUMN lvColumn;

lvColumn.mask = LVCF\_TEXT | LVCF\_WIDTH;

lvColumn.cx = 150;

for (int i = 0; i < COLUMN\_COUNT1; i++)

{

lvColumn.pszText = const\_cast<wchar\_t\*>(columnHeaders1[i]);

ListView\_InsertColumn(hListView1, i, &lvColumn);

}

hListView2 = CreateWindowW(WC\_LISTVIEW, L"", WS\_VISIBLE | WS\_CHILD | LVS\_REPORT, 10, 420, 1200, 400, hWnd, nullptr, hInstance, nullptr);

// 设置列表视图控件样式

ListView\_SetExtendedListViewStyle(hListView2, LVS\_EX\_FULLROWSELECT);

// 添加列表视图控件的列

lvColumn;

lvColumn.mask = LVCF\_TEXT | LVCF\_WIDTH;

lvColumn.cx = 150;

for (int i = 0; i < COLUMN\_COUNT2; i++)

{

lvColumn.pszText = const\_cast<wchar\_t\*>(columnHeaders2[i]);

ListView\_InsertColumn(hListView2, i, &lvColumn);

}

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// 函数: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// 目标: 处理主窗口的消息。

//

// WM\_COMMAND - 处理应用程序菜单

// WM\_PAINT - 绘制主窗口

// WM\_DESTROY - 发送退出消息并返回

//

//

void fresh\_list() {

// 清空列表视图控件

ListView\_DeleteAllItems(hListView1);

ListView\_DeleteAllItems(hListView2);

for (int i = 0; i < students.size(); i++)

{

LVITEM lvItem;

lvItem.mask = LVIF\_TEXT;

lvItem.iItem = i;

lvItem.iSubItem = 0;

wstring studentID = string2wstring(students[i].studentID);

lvItem.pszText = &studentID[0]; // 插入姓名

// 插入第一列的数据

ListView\_InsertItem(hListView1, &lvItem);

lvItem.iSubItem = 1;

std::wstring name = string2wstring(students[i].name);

lvItem.pszText = &name[0]; // 插入电话

ListView\_SetItemText(hListView1, i, 1, lvItem.pszText);

lvItem.iSubItem = 2;

std::wstring gender = string2wstring(students[i].gender);

lvItem.pszText = &gender[0]; // 插入学号

ListView\_SetItemText(hListView1, i, 2, lvItem.pszText);

lvItem.iSubItem = 3;

std::wstring dormitory = string2wstring(students[i].dormitory);

lvItem.pszText = &dormitory[0]; // 插入学号

ListView\_SetItemText(hListView1, i, 3, lvItem.pszText);

lvItem.iSubItem = 4;

std::wstring phoneNumber = string2wstring(students[i].phoneNumber);

lvItem.pszText = &phoneNumber[0]; // 插入学号

ListView\_SetItemText(hListView1, i, 4, lvItem.pszText);

}

for (int i = 0; i < scores.size(); i++) {

LVITEM lvItem;

lvItem.mask = LVIF\_TEXT;

lvItem.iItem = i;

lvItem.iSubItem = 0;

wstring studentID = string2wstring(scores[i].studentID);

lvItem.pszText = &studentID[0]; // 插入姓名

// 插入第一列的数据

ListView\_InsertItem(hListView2, &lvItem);

lvItem.iSubItem = 1;

std::wstring courseCode = string2wstring(scores[i].courseCode);

lvItem.pszText = &courseCode[0]; // 插入电话

ListView\_SetItemText(hListView2, i, 1, lvItem.pszText);

lvItem.iSubItem = 2;

std::wstring courseName = string2wstring(scores[i].courseName);

lvItem.pszText = &courseName[0]; // 插入学号

ListView\_SetItemText(hListView2, i, 2, lvItem.pszText);

// 插入其他列的数据

std::wstring credit = std::to\_wstring(scores[i].credit);

std::wstring regularScore = std::to\_wstring(scores[i].regularScore);

std::wstring labScore = std::to\_wstring(scores[i].labScore);

std::wstring examScore = std::to\_wstring(scores[i].examScore);

std::wstring compositeScore = std::to\_wstring(scores[i].compositeScore);

lvItem.iSubItem = 3;

lvItem.pszText = (LPWSTR)credit.c\_str(); // 插入数学成绩

ListView\_SetItemText(hListView2, i, 3, lvItem.pszText);

lvItem.iSubItem = 4;

lvItem.pszText = (LPWSTR)regularScore.c\_str(); // 插入英语成绩

ListView\_SetItemText(hListView2, i, 4, lvItem.pszText);

lvItem.iSubItem = 5;

lvItem.pszText = (LPWSTR)labScore.c\_str(); // 插入数学成绩

ListView\_SetItemText(hListView2, i, 5, lvItem.pszText);

lvItem.iSubItem = 6;

lvItem.pszText = (LPWSTR)examScore.c\_str(); // 插入英语成绩

ListView\_SetItemText(hListView2, i, 6, lvItem.pszText);

lvItem.iSubItem = 7;

lvItem.pszText = (LPWSTR)compositeScore.c\_str(); // 插入英语成绩

ListView\_SetItemText(hListView2, i, 7, lvItem.pszText);

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// 分析菜单选择:

switch (wmId)

{

case IDM\_LOAD:

{

f1 = read\_contact\_file();

f2 = read\_contact\_file();

loadData(f1, f2);

// 将manager中的联系人信息显示到列表视图控件中

fresh\_list();

}

break;

case IDM\_ADD:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 110, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 130, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox3 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 150, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox4 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 170, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox5 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 190, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox6 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 210, 100, 20, hWnd, NULL, hInst, NULL);

hTextBox7 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 230, 100, 20, hWnd, NULL, hInst, NULL);

hlabel1 = CreateWindowW(L"STATIC", L"学号", WS\_CHILD | WS\_VISIBLE,

920, 110, 80, 20, hWnd, NULL, hInst, NULL);

hlabel2 = CreateWindowW(L"STATIC", L"课程号", WS\_CHILD | WS\_VISIBLE,

920, 130, 80, 20, hWnd, NULL, hInst, NULL);

hlabel3 = CreateWindowW(L"STATIC", L"课程名", WS\_CHILD | WS\_VISIBLE,

920, 150, 80, 20, hWnd, NULL, hInst, NULL);

hlabel4 = CreateWindowW(L"STATIC", L"学分", WS\_CHILD | WS\_VISIBLE,

920, 170, 80, 20, hWnd, NULL, hInst, NULL);

hlabel5 = CreateWindowW(L"STATIC", L"平时成绩", WS\_CHILD | WS\_VISIBLE,

920, 190, 80, 20, hWnd, NULL, hInst, NULL);

hlabel6 = CreateWindowW(L"STATIC", L"实验成绩", WS\_CHILD | WS\_VISIBLE,

920, 210, 80, 20, hWnd, NULL, hInst, NULL);

hlabel7 = CreateWindowW(L"STATIC", L"考试成绩", WS\_CHILD | WS\_VISIBLE,

920, 230, 80, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_ADD;

}

break;

case IDM\_MODIFY:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 110, 50, 20, hWnd, NULL, hInst, NULL);

hTextBox2 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

800, 130, 50, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

//manager.modifyContact();

mode = IDM\_MODIFY;

}

break;

case IDM\_DELETE:

{

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 110, 100, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_DELETE;

}

break;

case IDM\_EXIT:

{

saveData();

DestroyWindow(hWnd);

}

break;

case IDM\_STUSEARCH: {

hStuIDRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"ID",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

1000, 240, 100, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hStuNameRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"姓名",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

1000, 270, 100, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 110, 100, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_STUSEARCH;

}break;

case IDM\_SCOSEARCH: {

hTextBox1 = CreateWindowW(L"EDIT", L"", WS\_BORDER | WS\_CHILD | WS\_VISIBLE,

1000, 110, 100, 20, hWnd, NULL, hInst, NULL);

hButton1 = CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_SCOSEARCH;

}break;

case IDM\_SORT: {

hStuIDRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"综合成绩升序",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON | WS\_GROUP,

1000, 240, 200, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hStuNameRadioButton = CreateWindowEx(

0,

L"BUTTON",

L"实得学分升序",

WS\_CHILD | WS\_VISIBLE | BS\_AUTORADIOBUTTON,

1000, 270, 200, 30,

hWnd,

NULL, // 修改为不同的标识符

hInst,

NULL);

hButton1= CreateWindow(TEXT("button"), TEXT("确认"), WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

800, 300, 100, 30, hWnd, (HMENU)IDM\_CONFIRM, hInst, NULL);

mode = IDM\_SORT;

}break;

case IDM\_CONFIRM:

{

switch (mode)

{

case IDM\_ADD:

{

string stuID, courseID, courseName, credit, usualGrade, experimentGrade, examGrade;

// 获取第一个输入框的文本

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

stuID = wstring2string(buffer);

delete[] buffer;

// 获取第二个输入框的文本

textLength = GetWindowTextLengthW(hTextBox2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox2, buffer, textLength + 1);

courseID = wstring2string(buffer);

delete[] buffer;

// 获取第三个输入框的文本

textLength = GetWindowTextLengthW(hTextBox3);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox3, buffer, textLength + 1);

courseName = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox4);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox4, buffer, textLength + 1);

credit = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox5);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox5, buffer, textLength + 1);

usualGrade = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox6);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox6, buffer, textLength + 1);

experimentGrade = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox7);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox7, buffer, textLength + 1);

examGrade = wstring2string(buffer);

delete[] buffer;

inputData(stuID, courseID, courseName, stoi(credit), stoi(usualGrade), stoi(experimentGrade), stoi(examGrade));

}

break;

case IDM\_DELETE:

{

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string stuID = wstring2string(buffer);

delete[] buffer;

deleteStudent(stuID);

}

break;

case IDM\_MODIFY: {

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string name = wstring2string(buffer);

delete[] buffer;

textLength = GetWindowTextLengthW(hTextBox2);

buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox2, buffer, textLength + 1);

string phoneNumber = wstring2string(buffer);

delete[] buffer;

//manager.modifyContact(name, phoneNumber);

}break;

case IDM\_STUSEARCH:

{

int StuIDCheckState = SendMessage(hStuIDRadioButton, BM\_GETCHECK, 0, 0);

int StuNameCheckState = SendMessage(hStuNameRadioButton, BM\_GETCHECK, 0, 0);

int choice = 0;

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string stuID, stuName;

if (StuIDCheckState == BST\_CHECKED)

{

choice = 1;

stuID = wstring2string(buffer);

}

else if (StuNameCheckState == BST\_CHECKED)

{

choice = 2;

stuName = wstring2string(buffer);

}

delete[] buffer;

wstring messageText = string2wstring(searchStudentInfo(choice, stuID, stuName));

//用messagebox显示searchStudentInfo的信息

MessageBoxW(hWnd, messageText.c\_str(), L"", MB\_OK);

}

break;

case IDM\_SCOSEARCH: {

int textLength = GetWindowTextLengthW(hTextBox1);

wchar\_t\* buffer = new wchar\_t[textLength + 1];

GetWindowTextW(hTextBox1, buffer, textLength + 1);

string stuID = wstring2string(buffer);

delete[] buffer;

wstring messageText = string2wstring(searchScoreInfo(stuID));

//用messagebox显示searchStudentInfo的信息

MessageBoxW(hWnd, messageText.c\_str(), L"", MB\_OK);

}break;

case IDM\_SORT: {

int StuIDCheckState = SendMessage(hStuIDRadioButton, BM\_GETCHECK, 0, 0);

int StuNameCheckState = SendMessage(hStuNameRadioButton, BM\_GETCHECK, 0, 0);

int choice = 0;

if (StuIDCheckState == BST\_CHECKED)

{

choice = 1;

}

else if (StuNameCheckState == BST\_CHECKED)

{

choice = 2;

}

sortScores(choice);

}break;

}

DestroyWindow(hTextBox1);

DestroyWindow(hTextBox2);

DestroyWindow(hTextBox3);

DestroyWindow(hTextBox4);

DestroyWindow(hTextBox5);

DestroyWindow(hTextBox6);

DestroyWindow(hTextBox7);

DestroyWindow(hlabel1);

DestroyWindow(hlabel2);

DestroyWindow(hlabel3);

DestroyWindow(hlabel4);

DestroyWindow(hlabel5);

DestroyWindow(hlabel6);

DestroyWindow(hlabel7);

DestroyWindow(hStuNameRadioButton);

DestroyWindow(hStuIDRadioButton);

DestroyWindow(hBusinessRadioButton);

DestroyWindow(hButton1);

fresh\_list();

}

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: 在此处添加使用 hdc 的任何绘图代码...

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

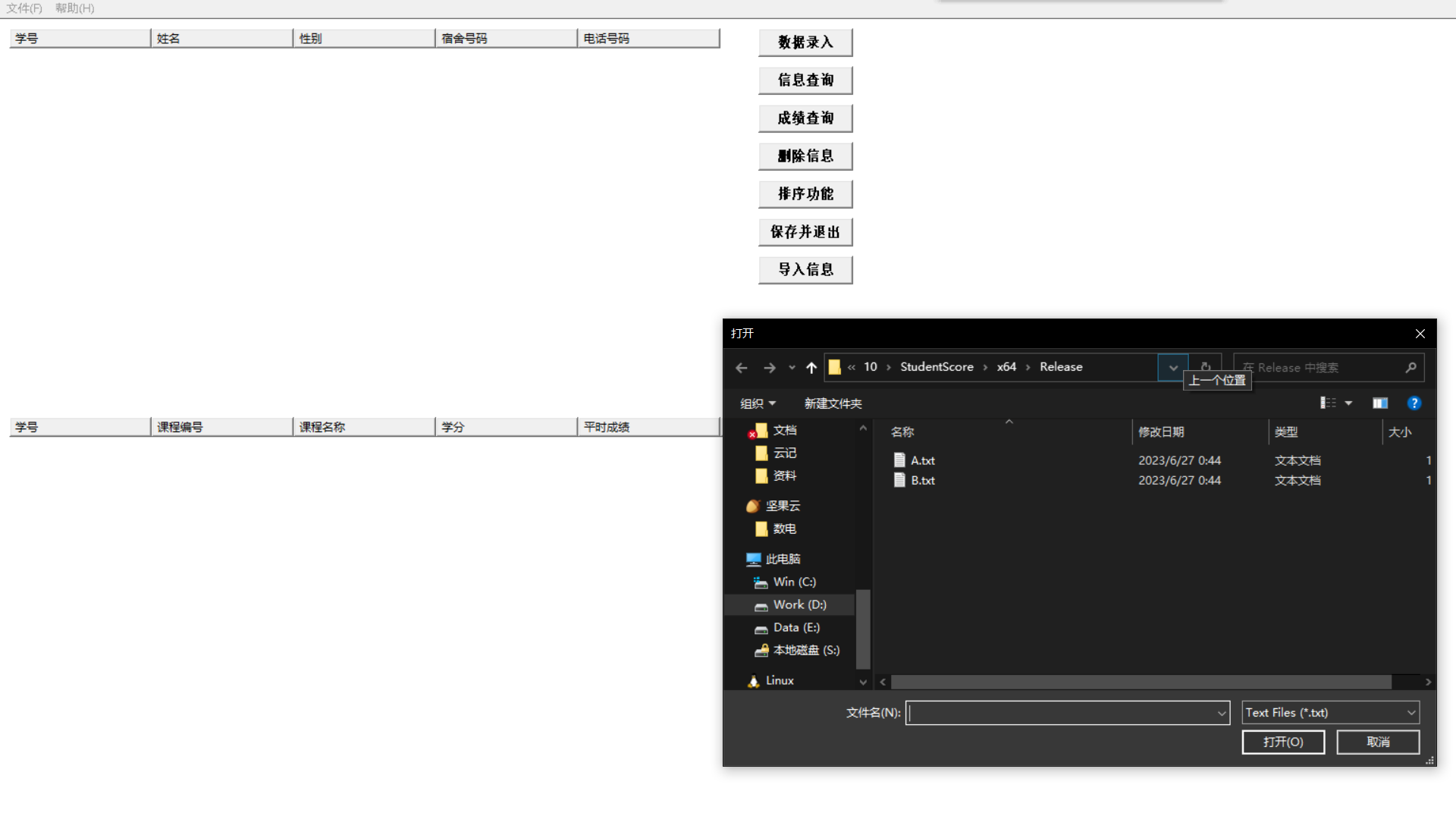
}

return 0;

}

### 实验结果：

点击“导入信息”后选择“A.txt”和”B.txt“文件，分别存储学生信息和成绩信息。

表格, 箱线图

中度可信度描述已自动生成

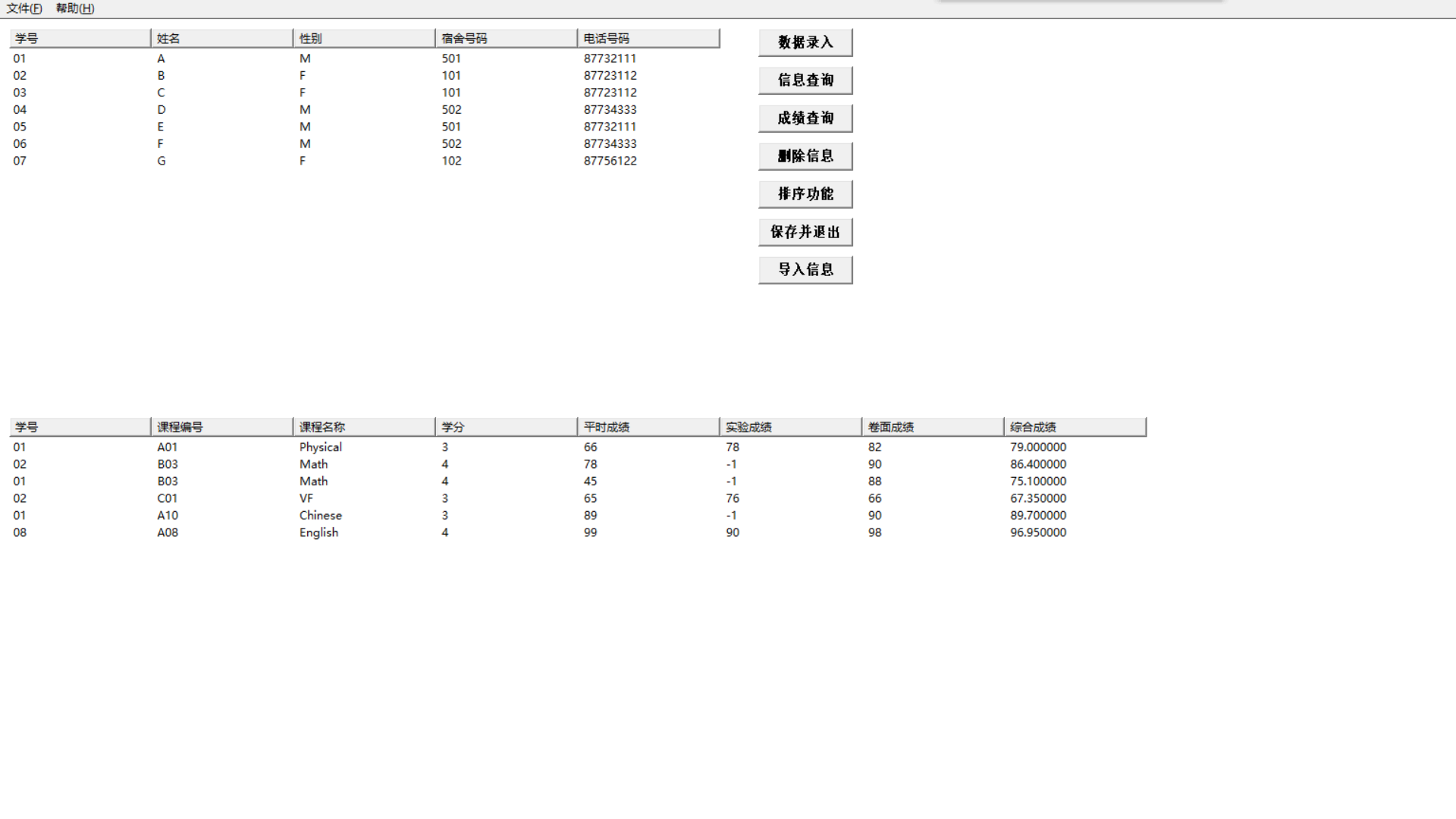
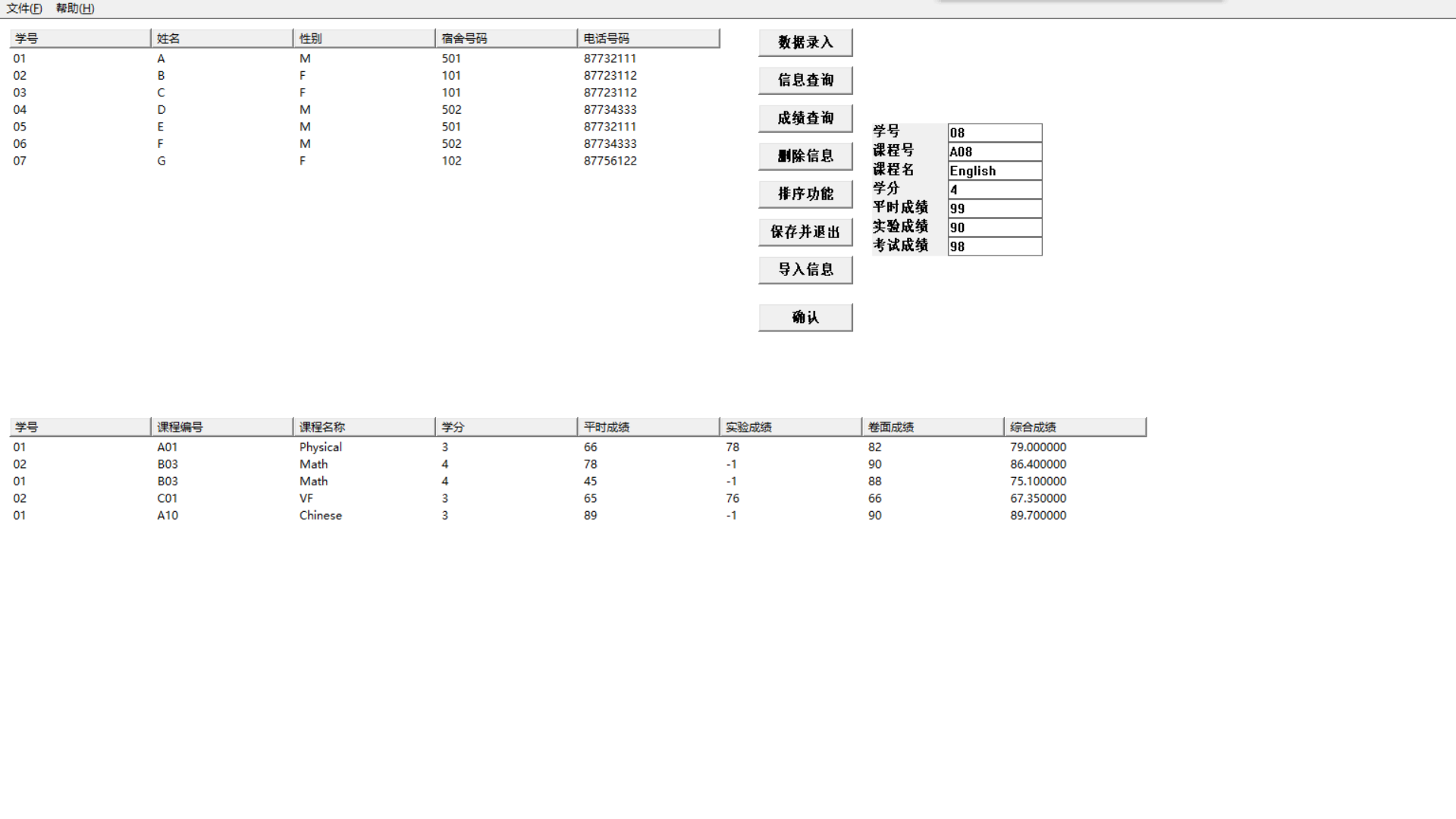
在txt文件中存储的信息和列表中展示的一致

电脑萤幕的截图

描述已自动生成电脑萤幕的截图

描述已自动生成

选择“数据录入“后填写要录入的信息后信息被添加到score列表中



信息查询模块测试

图形用户界面

描述已自动生成图形用户界面

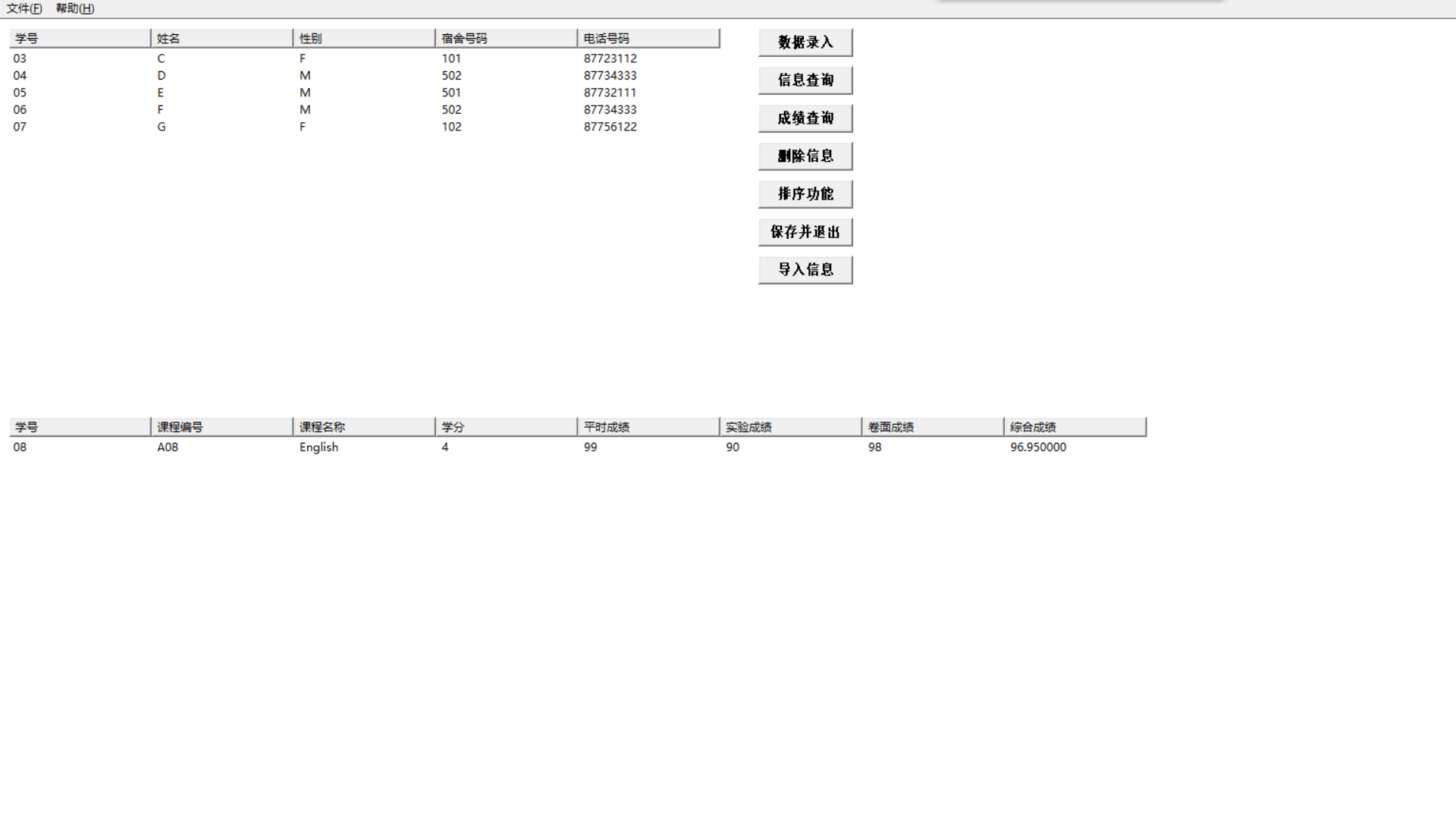
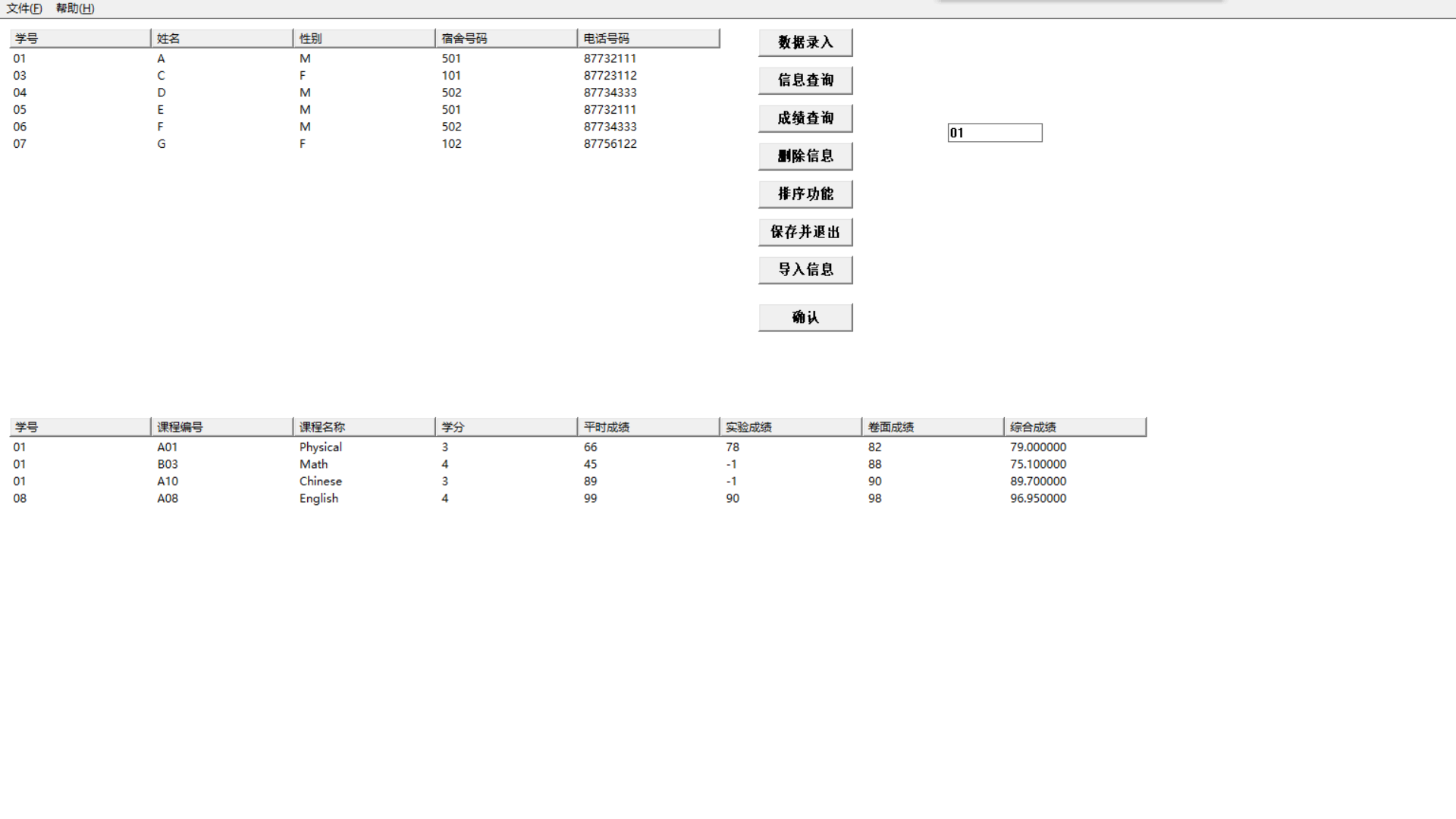
描述已自动生成

成绩查询模块测试

图形用户界面, 应用程序

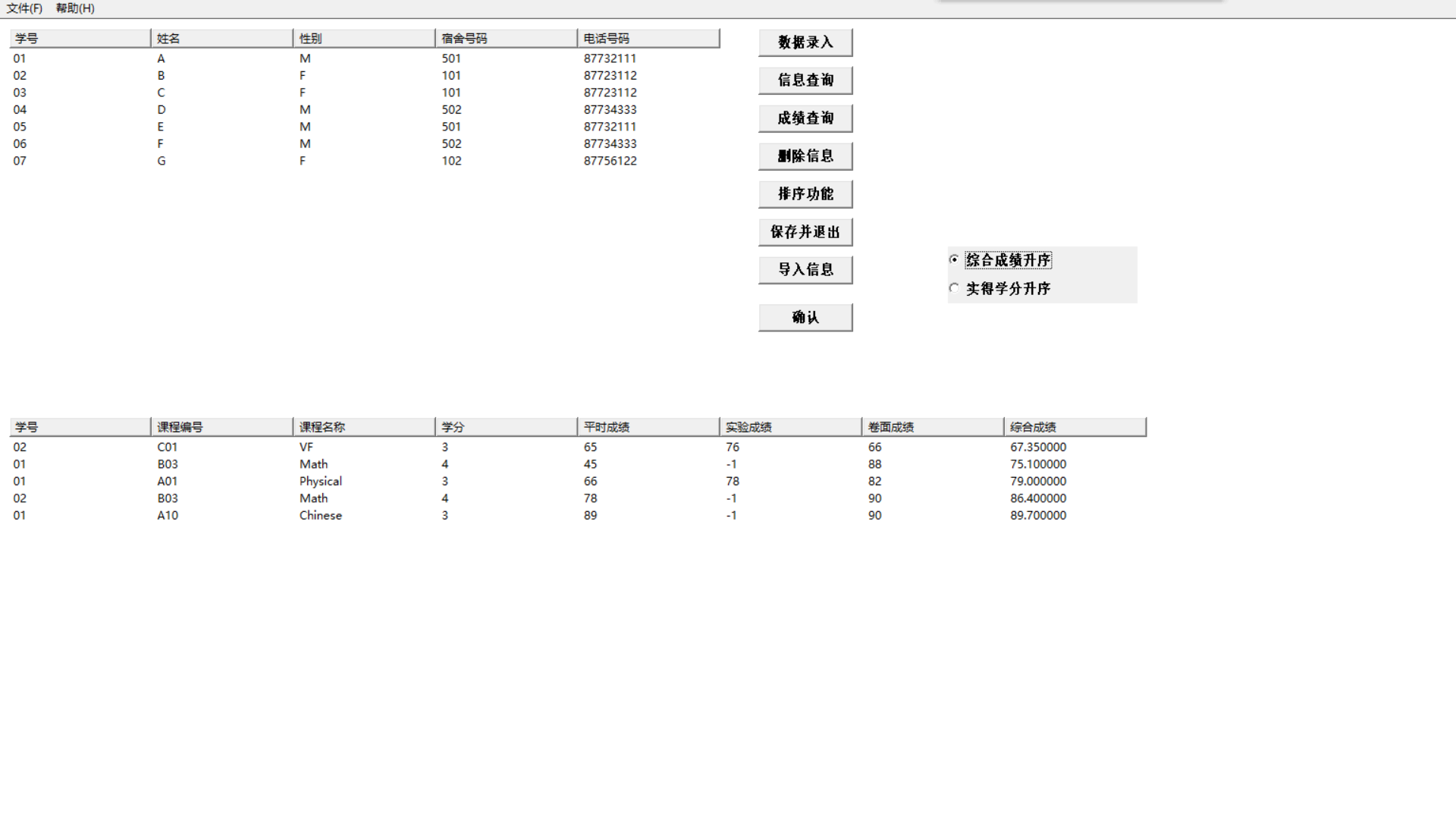
描述已自动生成

删除信息模块测试



排序模块测试

综合成绩升序

表格, 箱线图

描述已自动生成

实际学分升序

图表, 表格, 箱线图

描述已自动生成表格, 箱线图

中度可信度描述已自动生成

### 思考与扩展：