山东大学计算机科学与技术学院学院

计算导论与程序设计课程实验报告

学号：201705130120	姓名： 苑宗鹤	班级： 1 班
实验题目：学生成绩管理系统设计与实现		
实验学时：20	实验日期： 11.8-12.20	
实验目的：编程实现学生成绩管理系统 1、每个学生的记录包括：学号、姓名、成绩 2、功能：菜单提示：成绩录入、修改、添加、删除、查询统计、退出等功能。 如按学号、姓名、各科成绩查询（包括成绩段查询），按分数段统计人数等 3、录入修改后保存到文件中，以后从文件中读取进行相应的操作。		
硬件环境：Microsoft New Surface pro , Intel Core i7-7660U , 16G Ram , Intel Iris Plus Graphics 640		
软件环境：Windows10 64bit , Visual Studio2017 , Dev-cpp		

学生成绩管理系统开发文档

目录

一. 引言 2

1.软件设计功能 2

2.可行性分析/设计理念 3

二.模块功能实现思路 3

1.模块简介 3

2.数据库模块 3

3.添加单条信息模块(UI) 4

4.修改单条信息模块(UI) 4

5 统计功能模块(UI)..... 4
6.主界面(UI)..... 5
7 菜单界面..... 5
8 有关窗口类的封装..... 5
三.程序使用方法及注意事项..... 6
1.数据储存文件的要求..... 6
2.有关于数据的读入和输出..... 6
3.程序的容错功能..... 6
四.与开发文档关系不大的东西→开发过程的收获..... 7
1.有关 win32api 开发..... 7
(1).消息处理机制..... 7
(2).编码模式..... 7
(3).文件路径..... 7
(4).宏定义..... 7
(5).开发文档..... 8
(6).指针..... 8
2.C++/C 语言更多的神奇操作..... 8
(1).std:bind..... 8
(2).stable_sort()..... 9
(3).string 的内存管理机制..... 9
(4).跨文件全局变量..... 9
(5).c++文件流..... 9
(6).vector 的删除元素..... 9
3.程序设计思路..... 10
(1).抽象/封装/模块化/接口..... 10
(2).注释..... 10
(3).分层..... 10
(4).善用轮子..... 10

一.引言

1.软件设计功能

实现学生成绩管理系统 支持对一组学生成绩的管理操作如

查询功能:关键字查询(模糊搜索) 条件查询(如查找大于 50 的)

筛选功能:筛选出符合条件的供用户操作 条件与查询相似 即在查询后可以选择将查询结果加入筛选列表开启筛选模式 并提供追加筛选 重置筛选等功能

修改:提供修改单人单项 和经过筛选后批量修改的功能

插入/添加 添加单条信息 并插入制定位置 /从文件/窗口 批量录入多条信息 并插入到指定位置

文件功能 从文件读入数据库 新建一个学生成绩表单 保存到文件 另存为 打印当前筛选

统计功能 按科目统计分数 按分数统计人 计算各科排名 和上线人数/区间人数 统计单人

删除 批量删除与单个删除

排序功能 多关键字排序升序降序

UI 功能 多选/单选 快捷排序

2.可行性分析/设计理念

底层数据库使用 stl 的 vector string 和 map 通过 vector<map<string,string> >来完成对 n 个人 n 条数据的储存

Ui 界面使用 win32sdk 完成 开发工具使用 vs2017

封装:将底层数据和数据操作封装为一个类

将每一个窗口封装为一个窗口类

二.模块功能实现思路

1.模块简介

数据库模块/排序底层: Stugrade.h 提供数据的储存 文件输入输出 底层的排序修改删除操作

主界面:从零开始的 win32sdk 开发.h 基本的界面展示当前的数据并提供一些交互的操作

主界面附加操作函数 solve.h

预编译头文件 stdafx.h

添加单条信息模块 Addwindow.h 当点击快速添加按钮 弹出这个界面

编辑单条信息 Editwindow.h 当右键信息 并选择编辑 弹出这个界面

实现筛选功能的 bool 判断 filter.h 提供底层的筛选 bool 判断

统计功能 Statistics.h 接受多种条件统计并返回结果

2.数据库模块

主要成员

```
vector<map<string,string> >datas 每个人的数据是一个 map vector 中存储 n 个人的信息
vector<string> keyord 存着 n 个表头的 string
vector<int>displayord 当前窗口显示的数据集 用于筛选模式(使一些人的数据不被显示)
string filestring 保存着当前打开的文件的路径
```

主要 方法

```
int readfile() 从 filestring 指向的文件中读入全部的数据集到内存
int deldata(set <int> x) 传入一个集合 从数据中删去集合中的所有标号
```

int deldisplay(set<int>x) 从显示的列表中隐藏一些数据

int savedata() 将内存中的数据写入 filestring 指向的文件

int sortviakey(int x) 按照传进来的表头标号对数据进行排序

完全使用动态结构的数据层 使得数据结构能根据文件弹性变化

如在文件中敲入 5 个科目 则可以显示 5 个科目 敲入 10 个科目 可以显示 10 个科目

3.添加单条信息模块(UI)

一个 listbox 盛放表头的各项

Edit ctrl 用来输入增加的数据

右侧的插入位 默认插入到表的最后

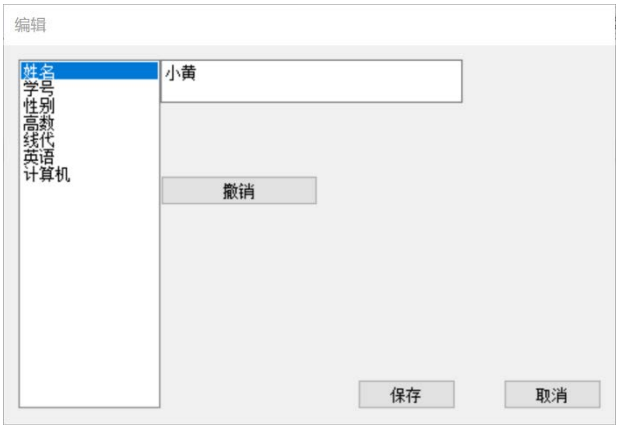
可以自定义插入的位置

撤销键通过一个栈结构维护历史操作

按回车等效于按下下一步

点击保存将数据写入数据系统

4.修改单条信息模块(UI)

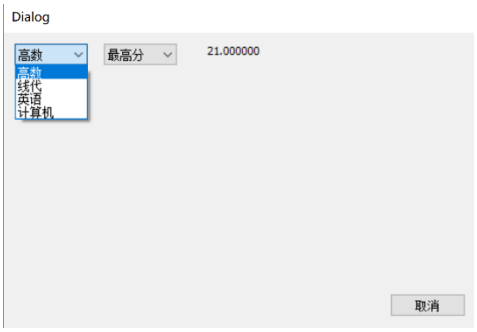


左侧一个 listbox 列出表头的索引

右侧的 editctrl 显示当前的表头对应的数据并提供修改功能

保存键将修改保存至数据系统

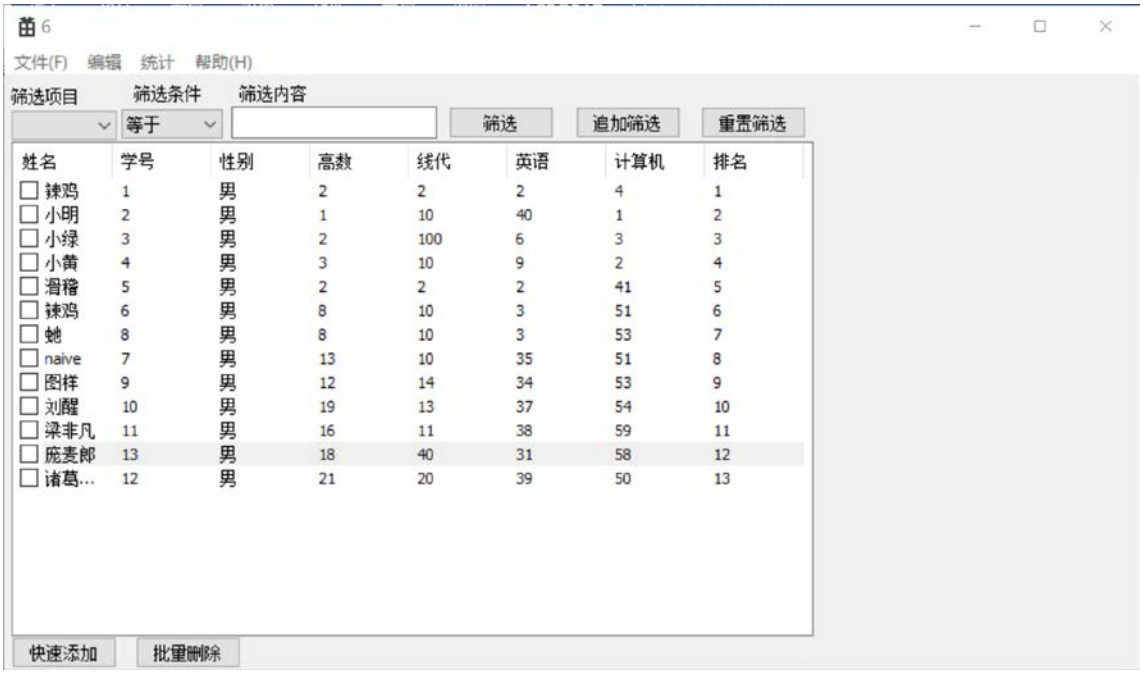
5 统计功能模块(UI)



两个下拉菜单提供任意项目和统计条件的组合

然后算出结果

6.主界面(UI)



点击 姓名 学号 性别 高数 那些表头元素 可以实现排序 类似于 windows 的文件管理器的排序

左下角的快速添加 可以单条添加信息

左侧的复选框选中后可以实现批量删除

右键列表中的单个项目 可以展开一个菜单提供了删除和编辑的功能

上方的筛选可按照任意表头为关键字 使用小于小于等于 等于不等于 大于大于等于六种筛选方式 可单次筛选 也可多次筛选

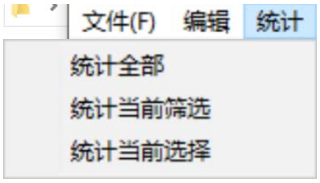
如筛选出 60~80 分的 可以筛选两次 一次条件为大于等于 60 另一次条件为小于等于 80

7 菜单界面

文件菜单 提供了从文件打开 保存到文件 另存为其他文件

退出(自动保存) 不保存就退出的功能

其中从文件打开和另存为都调用了 win 自带的文件选择器保证了交互体验的友好



统计菜单

可以从复选框选中一些进行统计 也可以在筛选之后统计筛选过的 也可以统计当前读入的全部数据

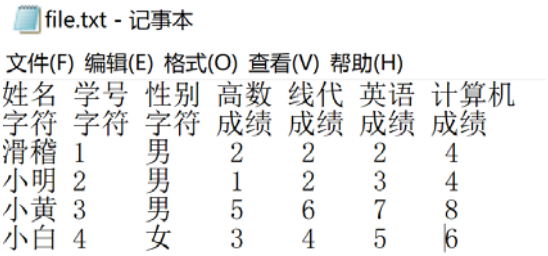
8 有关窗口类的封装

对于每个独立的窗口 都实现一个 class 来封装 这里直接使用类的静态成员和静态函数来作

为每个窗口的回调函数和内部变量 免去了实例化窗口类的过程(其实应该实例化的 但是由于非静态函数作为回调函数会有 this 指针的问题 暂时使用了静态方法)

三.程序使用方法及注意事项

1.数据储存文件的要求



如图 文件分为 3 部分

1 第一行 为表头(数据的关键字) 需要空格隔开

2 第二行 与第一行对应 描述的是对应表头的数据类型 比如姓名 学号 都是字符类型的 而各科成绩都是成绩类型的 对这些做区分一是为了排序 对于字符类型将使用字典序排序 而对于成绩 将使用数值大小进行排序 二是提供给用户更多的自定义空间 比如用户可以加入 联系电话-字符 这样就能动态的定义整个数据结构

而从第三行开始每一行为一个人的数据按照表头的顺序每个元素之间需隔开一个空格

理论上数据允许有多余空行

2.有关于数据的读入和输出

程序在启动时会自动读取同目录下的 file.txt 文件 如找不到文件或者文件打开失败又或者格式不对 程序会提示使用者 这时需要从左上角的文件菜单里点打开 通过文件选择器来选择可以使用的文件

程序在点击文件菜单里的退出的时候会自动保存文件 点击右上角的关闭时不会自动保存文件

可以在程序运行时点击菜单里的保存或者另存为来保存当前的文件

在文件中***表示当前表头对应的值为空

3.程序的容错功能

你可以使用各种奇怪的方式来使这个程序崩溃 比如强行让程序读入错误的程序或者添加一些奇怪的数据导致程序崩溃

一个良好的程序需要大量的机制来防止各种不合适的输入(但是由于时间原因我并没有写的很多)

如果你对程序进行最小化 在恢复时窗口不会重绘 需要拖动窗口来刷新界面

程序有可能在读取文件失败后不做出任何提示

一个完美的程序需要大量容错的机制与异常处理

而这需要耗费太多精力 很可惜 我没有

四.与开发文档关系不大的东西→开发过程的收获

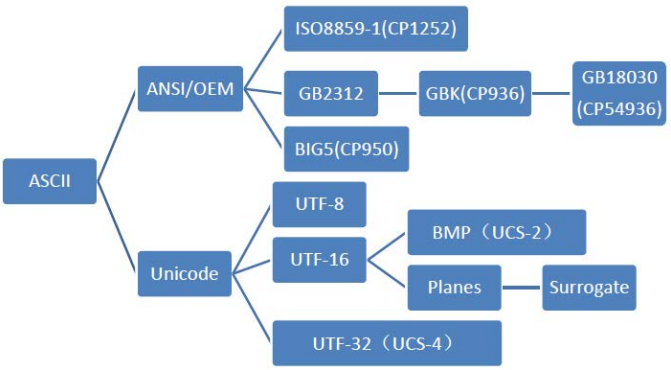
1.有关 win32api 开发

(1).消息处理机制

Win32 程序在运行后会进入一个消息循环 一个死循环
在接受到消息之后循环便执行一次 在处理消息结束后循环又进入等待的状态 等待下一个消息被出发
消息有多种多样 如窗口大小的改变 窗口控件的点击 鼠标在窗口上的操作 等等 都会让操作系统给程序发送消息 而这些消息有着不同的标识码 让用户能够自行处理这些消息
对于一个窗口 对应的拥有一个回调函数 用来处理发生在这个窗口上的消息

(2).编码模式

Vs 提供了多字节码和 unicode 字符集的编码模式 多字节码



Unicode 对应着 wchar_t 和 wstring
而多字节码对应着 char 和 string
使用 unicode 编码会一定程度上提高编程的投入 但是能够兼容多国语言

(3).文件路径

在 windows 下 .\\file.txt 表示与程序同目录下的 file.txt 双斜杠是转义符
而使用两个句号 如 ..\\file.txt 表示当前的目录向上一级父目录中的 file.txt
这些都是相对路径 相对路径取决于程序运行的位置和运行环境
如使用 visual studio 打开程序时 相对路径的默认路径将被修改至工程文件夹的路径 而不是 exe 的路径
而 d:\\qq295\\Desktop\\file.txt 这种 加上盘符 或者写着全部地址的 为绝对路径
不随着程序运行的位置改变而改变

(4).宏定义

从编写 cpp 源代码转向 win32api 开发最不适应的就是其宏定义

比如 int* 定义为 PINT

char 定义为 CHAR

其他的还有很多

而且这些宏定义不是一对一的而是多对一的

比如 int 类型 用 INT HWND BOOL 都表示 int

而如此定义的原因是让代码的编写者注重这些变量的用途而不是数据类型

比如 HWND 表示句柄 虽然是一个 int 类型 但把他想想为句柄 似乎更为直观

又或者 c 语言中没有布尔值 所以用 BOOL 代替 int 让程序员明白此处的 int 是用作 bool 值的 只有 10

再如 win32sdk 采用大量的宏定义来简化开发过程

如

```
#define ListView_GetCheckState(hwndLV,i) ((((UINT)(SNDMSG((hwndLV), LVM_GETITEMSTATE, (WPARAM)(i), LVIS_STATEIMAGEMASK))) >> 12) -1)
```

这些宏定义某种意义上是一种封装 提高了代码的简洁程度 降低了无用元素的出现 使得开发过程更为直观

以前从来没有考虑过使用宏定义 现在看开需要仔细考虑考虑了

(5).开发文档

微软给开发者提供了开发文档 在入门时需要跟着网上的教程来一步一步走

但当你需要弄懂一些技术细节时 官方的开发文档成为了唯一的救命稻草 那里有所有函数的声明 用法 功能 等等 虽然看起来十分冗长 但想要知道的一些 都来源于开发文档

(6).指针

在 win32 的开发中 指针使用的更加频繁 为了防止全局变量满天飞 需要大量的使用指针

同时在系统层面也有很多涉及到指针的操作

包括结构体 函数等 都和指针操作紧密相关

同时使用指针也体现了 c 语言的灵活性

2.C++/C 语言更多的神奇操作

(1).std:bind

头文件`#include <functional>`

在工程中使用 stable_sort 传入 cmp 函数时 其使用的 cmp 函数无法正确的获得当前实例的 this 指针 这时需要使用 bind 传入一个 this 指针

Bind 可以看作是创建了一个函数的副本

而在创建这个副本的过程中对传参的过程进行了修改

比如传入一个自定义的 this 指针 或者使某个参数在调用时为固定值

改变参数的顺序等等 相当于在一个函数的基础上自造了一个

基于之前的函数但是传参机制不同的函数副本

(2).stable_sort()

Stable_sort 稳定排序 使得相同(重载小于号相同)的元素在经过排序后顺序不变
比如一组 pair 值 1-1 2-2 2-1 1-4 如果按第一关键字进行 stable-sort
结果为 1-1 1-4 2-2 2-1 可见 2-2 和 2-1 的元素相对顺序没有改变因为按照第一关键字 2-2 和 2-3 是相等的 而如果使用 sort 则可能改变这些元素的相对顺序
通过这种特性便可以不用重载函数来实现对多关键字数据的多级排序
通过向 stable_sort 传入不同的 cmp 函数来进行多次排序 这样每次排序只改变对应关键字不同元素的顺序 而之前一次的排序顺序某种程度上不会被改变

(3).string 的内存管理机制

String 实际上是一个指针 系统在程序运行时维护者对这个对这个 string 数据的引用数量
当引用量为 0 时 string 对象被释放(有待考究)
而这就发生了奇怪的事情
在一个函数中返回一个 string 对象的.c_str()指针时 程序并不能在函数返回后接受到正确的 string 对象 而如果用字符数组来完成这个操作则可以
可以猜测 string 的.c_str()并不是对 string 对象的引用 在返回它时 函数中对 string 的引用被释放 紧接着 string 类型也被释放 在函数返回后便无法获得正确的结果

(4).跨文件全局变量

在一个 cpp 中声明的变量的作用域只存在于这个 cpp 文件中 从其他 cpp 文件无法访问这个变量
而使用如 extern int a ; 在 xxx.h 中
其他 cpp 文件 include 了 xxx.h
就可以使用这个 a 变量了

(5).c++文件流

Fstream 文件读写流 istream 文件读入流 ostream 文件输出流
通过一个文件地址和一些开关就可以创建一个文件对象
如 fstream file; file.open("233.txt",ios::in);
就创建了一个读写流并以只读模式打开了一个 233.txt 的文件
而通过类似于标准流的操作便可以从流中读取数据
如 int x; file>>x;

(6).vector 的删除元素

通过迭代器 it 遍历 vector 数组时 vector.erase(it)可以删除这个迭代器指向的元素
但是这时这个 it 迭代器就失效了
所以正确的做法是用一个迭代器去接收 vector.erase()的返回值→指向下一个元素的迭代器

在循环判断时通过 it!=vector.end()来判断迭代器是否完成了对数组的遍历

3.程序设计思路

(1).抽象/封装/模块化/接口

抽象就是把一些固定的模式拿出来 然后封装/打包
这样既能减少重复的操作 有能降低程序各部分的聚合度
比如生成一个窗口 那么这个窗口内部就有一些变量一些属性一些方法
如果这些都写在全局变量里 整个程序的可读性和可编辑程度都会大大降低
这时候就要把这些东西抽象出来 再打包成一个类 或者一个函数
而在调用时则简洁明了
或者说提供了一种接口 你在编写主程序时无需知道这个接口是怎么实现的 你只需要知道使用这个接口 输入一些数据 你就能得到一些数据 这就足够了
这样让整体程序的开发变得思路清楚
所有的代码都模块化以后修改起来也变得简单 只需要修改对应函数的实现即可

(2).注释

一个良好的变量名能够提高代码的可读性 但是只靠变量名是不够的如果所有的变量名都起的很长 编写起来仍然十分痛苦
最好的方式是加上注释 给函数加注释 给变量加注释 给一些代码块加注释 这样别人在阅读你的程序时就会有一个清晰的思路了

(3).分层

Ui 界面和底层的数据 就应该被分为两层
比如在 ui 界面上改变了元素的显示顺序
不意味着应该在底层数据库中改变元素的存贮顺序 否则将带来混乱和浪费
这也是一种抽象 显示层和数据层之间不应该有过多的代码层面的交织
应该使用一些接口和函数将两层联系起来
Ui 层和数据层互相调用各自的一些函数
这样在修改程序结构的时候才不至于发生改一处 处处改的尴尬情况

(4).善用轮子

前人铺好的路就要走 不然就浪费了 这次程序设计最失败的一点就是既没有用 qt 也没有用 mfc 只是在用 win32api 和 win32sdk 硬写 虽然说提升了自己对底层的了解 但是从软件开发上来讲降低了开发的效率并且自己实现的大部分没有已经造好的轮子好
如果使用 mfc 或者 qt 可能整个开发时长会大大缩短 或许时间能用在更有意义的事情上而不是学习如何自己写 ui 界面
比如从字符转换到 int 类型 一个 atio()就搞定 类似的还有 atof() itoa()等等

而数据结构的实现 也直接使用了 stl 的 map vector set

这样大大减少了程序编写的复杂度 提高了编码的效率 各种各样的库函数都可以拿来使用

细枝末节的地方也无需自己注意 而性能上甚至还有提升

或许进一步 可以直接使用现成的数据库比如 mysql 能让代码更加简洁