“程序设计基础（C 语言）课程设计”报告

题目：　学生成绩管理系统

班级：　计算机 1808 班

组长：　学号：　20184444　　姓名：　刘添夷

组员：　学号：　20184539　　姓名：　刘荣江

2019 年 3 月

目　录

1. 概述

1.1 问题描述

制作一个学生成绩管理系统

1.2 基本要求

- 数据维护(数据初始化、录入、添加、修改、删除)
- 数据查询（可按学号、姓名、性别、民族、年龄、地址、各门课程成绩等进行查询，也可组合查询）
- 排序、统计、输出
- 系统维护（数据备份、数据恢复、口令维护）
- 帮助、退出等

本次程序的目标平台是 Windows，需要掌握基本的链表、向量等数据结构和排序算法，并且对大量数据的存储方式与存储结构有一定了解；为设计一个 Windows 窗体应用程序，也要求对 Windows API 和消息循环体系有一定了解。

1.3 人员及工作量占比

队长刘添夷，队员刘荣江。工作量主要为队长承担，大约占整个项目的 60％到 70％。

2. 需求分析

2.1 目标与功能

【目标】
完成一个基本的学生成绩管理系统，可以辅助教务处老师进行学生信息、成绩的管理工作，支持一些特殊需求（批量录入成绩等），使用友好的图形界面，并有密码保护功能。

【功能】
- 根据关键字查询成绩条目或学生信息，支持多关键词和不完整关键词匹配（模糊搜索）
- 初始化班级和新增学生信息，完成表结构初始化，为快速录入做准备
- 录入、修改成绩，可以只对一个学生操作，也可以对一个班级/学科批量操作
- 导出数据，将所有数据导出至文本文件备份
- 口令维护，修改进入密码

2.2 数据来源

外部数据由管理员手动录入，输入时有提示伴随

本地有一个数据文件和一个密码文件作为数据存储
程序启动时会把数据加载到内存，在完成对数据的修改后即时写入文件

2.3 输出

图形界面用于密码核对与功能选择，其他功能由命令行窗口与用户交互
“导出数据”功能可以把数据导出至文件

2.4 测试数据

在密码输入（错误会提示重新输入）、密码修改（两次输入密码不同会提示重输）、功能菜单选择等方面具有对非法输入的处理特性，不会产生异常。

3. 总体设计

3.1 总体功能模块

（此处为 SmartArt 图表）

3.2 总体数据结构

- 用于存储学生信息的结构体及其链表结构
- 用于存储成绩信息的结构体及其链表结构
- 另外还有一个单项插入的简单输出缓存队列 用于缓存筛选/排序后的条目指针

3.3 成员分工

队长刘添夷：

MainWindow.c 中的部分内容　　Encrypotor.c

FastIO.c　　QueueControl.c

队员刘荣江：

MainWindow.c 中部分关于绘制内容　　BackupIO.c

Sorting.c

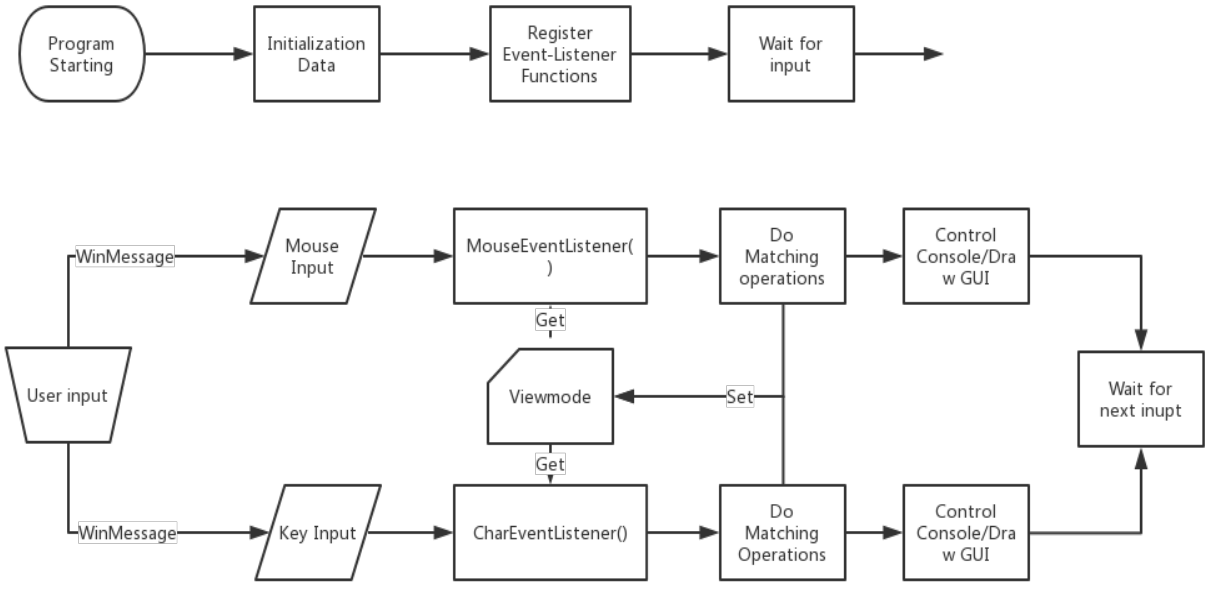
4. 详细设计

4.1 成员 1 详细设计

• IO 模块：使用 fread/fwrite 对整个结构体进行文件读写，在文件头标明记录数量以指定文件结尾
• 加解密模块：可以使用指定的秘钥对字符串进行 XOR 加解密
• 主窗口控制器模块：程序的控制核心，直接对用户交互并调用其他各模块，许多功能的实现函数也在其中，主要逻辑遵从 Windows 的消息循环体系
• 队列控制器模块：构建了一个简单的封装的队列控制器，外部可以使用 insert 函数在末尾插入值，也可以访问遍历元素

4.1.1 算法流程设计

下图是主窗口控制模块的消息循环主要流程图
涉及单线程下的 WinMessage 的接收反应机制



4.1.2 代表性数据结构

```
typedef struct {
        int id;
        char name[10];
        int sex;
        char nationality[10];
        int age;
        char address[100];
        char keyString[200];
} Student;

// 用于存储学生信息
```

```
typedef struct {
        int Typed;
        int StuID;
        char StuName[30];
        char Class[30];
        int Grade;
        char Subject[30];
        char keyString[200];
} Record;

// 用于存储成绩记录
```

```
typedef struct DataListNode {
        Record data;
        struct DataListNode * next;
}Node;

typedef struct StuListNode {
        Student data;
        struct StuListNode * next;
}StuNode;

// 链表结构
```

另外还有一个单项插入的简单的输出缓存队列 用于缓存筛选/排序后的条目地址：

int QueueCount;

void* list[1000];

4.2 成员 2 详细设计

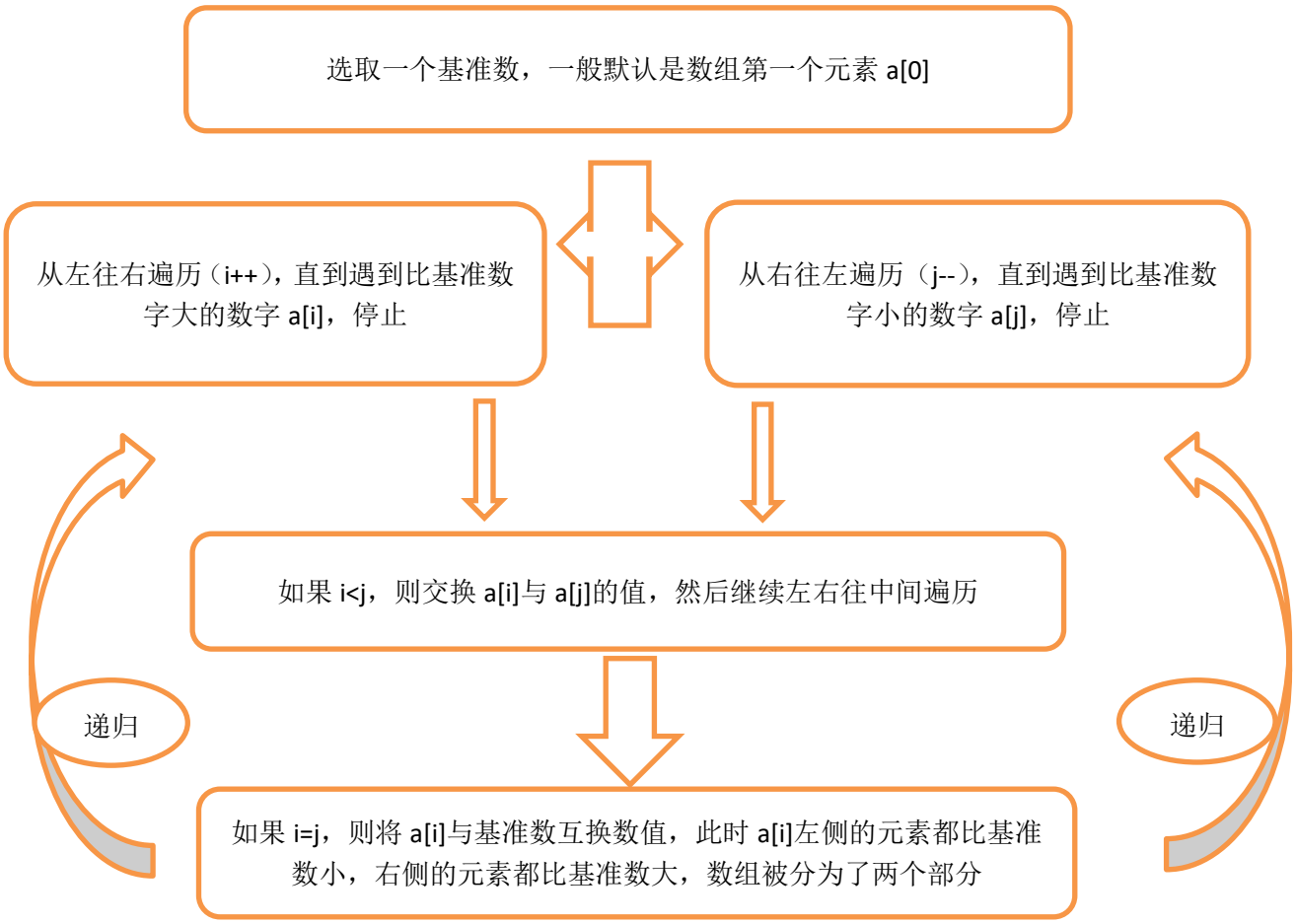
• 最初使用文本和光标绘制界面，后来改为使用一个纯 C 语言的函数库绘制图形界面，并且可以实现弹出提示窗口。

• 数据备份的相关功能：while 循环在没有读取到链表尾部之前把读取到的内容写入到另一个文本文件，从而实现数据的备份。

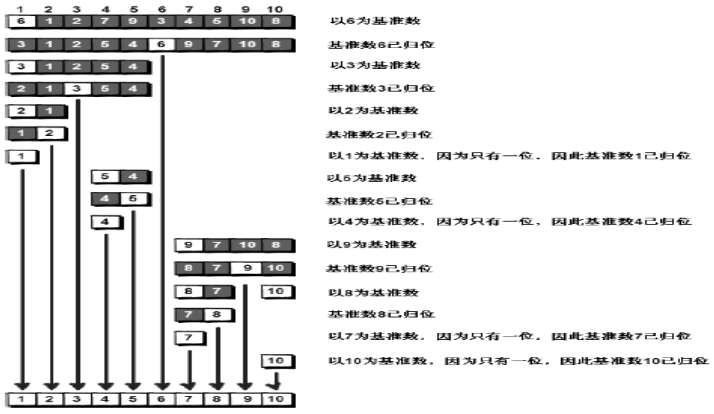
• sorting.h 中排序相关功能的相关函数：其中包含快速排序算法，冒泡排序算法，交换两个值的函数 swap，判断大小关系的函数 cmp（为了增加代码可重用性，将比较函数分离）。其中快速排序法的原理通过网页介绍和书籍学习。

4.2.1 算法流程设计

快速排序法的原理示意：

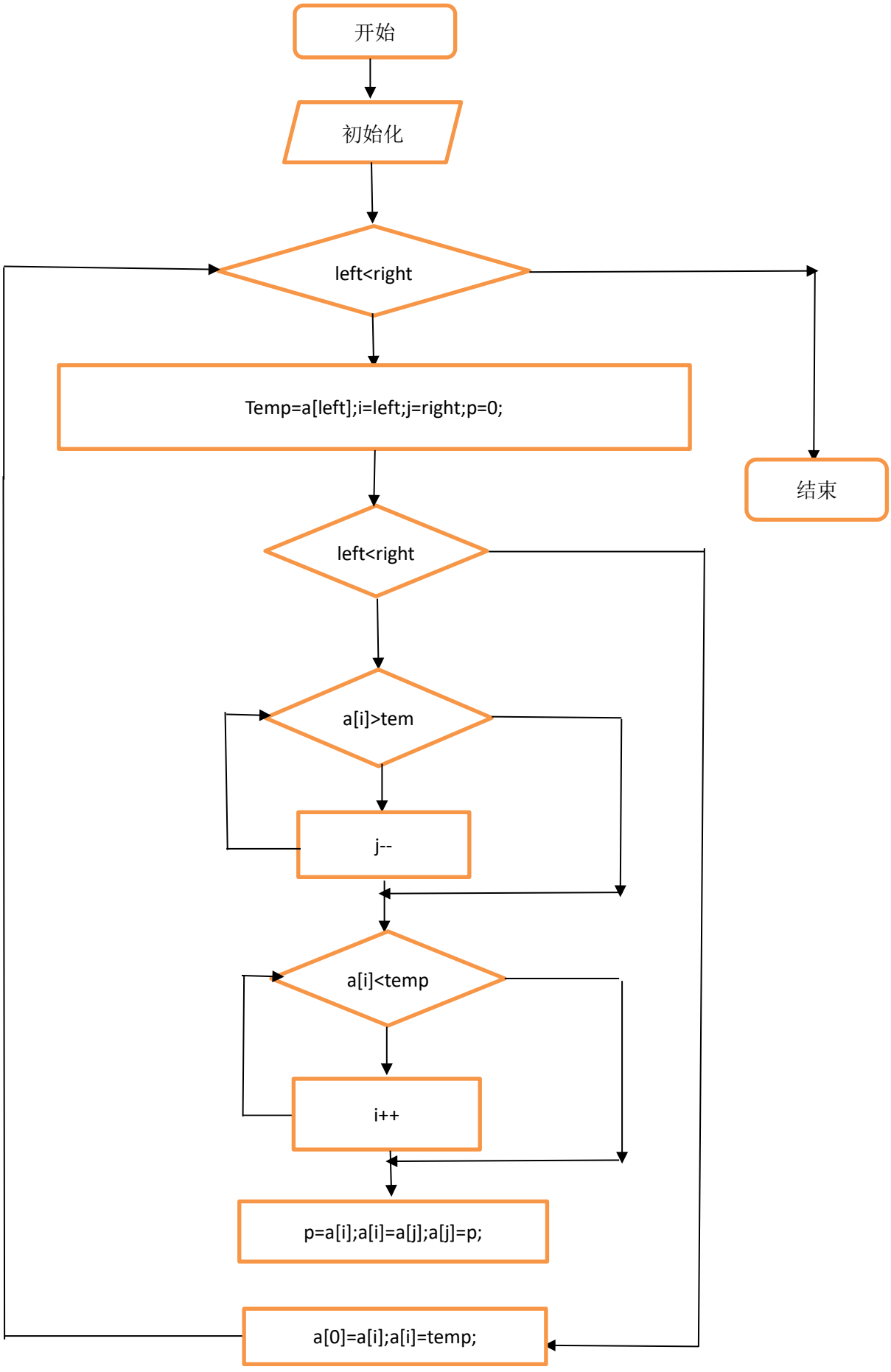


演示示意图：



图源：《啊哈！算法！》快速排序法运行示意图

算法流程图：


开始
初始化
left<right
Temp=a[left];i=left;j=right;p=0;
结束
left<right
a[i]>tem
j--
a[i]<temp
i++
p=a[i];a[i]=a[j];a[j]=p;
a[0]=a[i];a[i]=temp;

4.1.2 代表性数据结构

void SaveBackupFile(Node* head) 使用了链表备份数据，	static int cmp(Record* a, Record* b) void bubblesort(Record* arr, int left, int right) void quicksort(Record* arr,int left, int right) 指针数组的使用	typedef struct { int id; char name[10]; int sex; char nationality[10]; int age; char address[100]; char keyString[200]; } Student; 共同定义的结构体存贮信息

5. 调试分析

5.1 成员 1 调试分析

5.1.1 问题及解决方法

调试中遇到了许多错误，以下是部分遇到的主要问题：

- 文件 IO 模块用 EOF 判断文件尾不可靠，会导致多读出两行乱码
  解决方法：在文件头写明数据量，严格按数据量读入
- 单线程程序在等待用户 Console 输入时线程阻塞无法绘图的问题
  解决办法：改为异步架构，用变量存储操作状态，在短操作完成后释放线程使程序处于等待事件触发状态，同时进行绘图操作
- 有时程序运行会出现 Runtime Error 退出，或者数据结构损坏的问题
  解决方法：经过研究，是链表操作的部分函数出现了问题，在新增数据时有时会使链表断开，从而使链表指向未知地址

……

5.1.2 讨论和分析

大工程的编写和调试是一个很繁杂的过程，为了尽量顺利地完成，我采用了模块化的设计思想：编写一个模块，就立即开始试验，直到确认这一部分没有问题为止。

因为很多功能的改进是在调试中进行的，以下补充一些具体功能的思路来源：

【核心问题：数据存储结构】

• 学生信息+成绩信息

• 最初的想法：建立三个表 分别存储学生/课程/成绩（比较清晰）→ 需要全部遍历（没有明确的父子结构）

• 改进……

• 目前的方法：仅建立两个表 学生信息和成绩分开存储 必要时互相调用

【功能实现的简要思路——数据查询】

• 传统的数据查询——选项太多/多条件查询较麻烦

• 自研新思路：【使用关键词进行模糊搜索】

• Struct Record{char* name, int score, char* subject……}

• 构造特征字符串 KeyString：”name=LiHua score=90 subject=math……”

• 查询时将用户给定的关键词分割，依次筛选满足所有特征的放入缓存

• 对缓存进行排序（冒泡排序速度慢，使用 qsort 代替）

• 输出即可

【功能实现的简要思路——数据维护】

• 最初：一个一个维护，建立班级的结构体数组

• 问题：存在三个甚至更多个线性存储结构，存储和调用非常麻烦，且很多数据相互关联依赖，容易出现问题

• 思路：可否借用查询中一些比较方便的思想？

• 最终：使用搜索功能找出需要修改的位置，记录链表中该位置的指针，然后依次进行操作

5.1.3 经验与体会

• 在调试中，良好的模块化和函数化确实能大幅降低调试的压力，其中最明显的是代码量的大幅降低，比起未优化时的千余行代码，最终的 700 多行不论是在美观性上还是易调试性上都有了大的提升。

• 另外调试确实需要对写好的模块及时进行，如果是大量的模块堆叠在一起，想确定 bug 的位置就相当麻烦。

（关于整体项目的体会 请参见本文最后的总结）

5.2 成员 2 调试分析

5.2.1 问题及解决方法

输入数组发现排序不对，总是排序不对，甚至出现了没有任何返回的情况以及重复打印一个数字的情况。

界面对不齐，直线把文字给替代了，打印出来的界面和想象中的不完全一样，出现了界面文字和边框相互交错的情况。

递归之前那一步没有把 a[i]和基准数互换，通过对源代码的分析观察，以及和书籍之中存在的简单快速排序法的对比，发现了问题所在，直接修改代码就好了，学会了快速排序法的原理。

界面打印则是通过一次次的调整最后达到可以接受的情况，在这个过程之中团队想能不能有什么办法可以像数学的平面直角坐标系一样直接对绘制的区域做定位，网上寻找，最终发现了一个通过坐标直接打印直线的方法，并且升级了打印函数，使得可以通过直接输入横纵坐标进行绘制，清晰明了。

5.2.2 讨论和分析

讨论：为什么打印界面要花那么多时间精力还有代码来打印一些字符边框？怎么可以做到不用一次一次地调整位置或者怎么样可以做到更加快速简单地调整？如果换一个显示器怎么迅速地调整显示情况？

分析：可以使用平面直角坐标系，简单明了，而且以后调整也只需要修改横纵坐标。这样做到了代码量减少，内容简介易懂，修改方便。

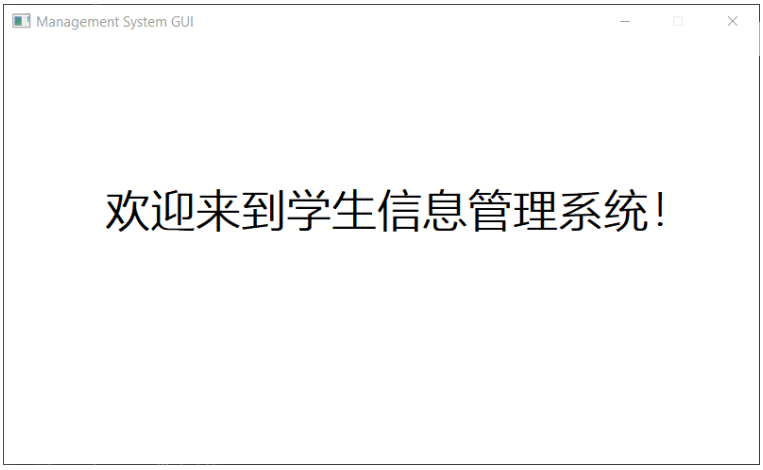
5.2.3 经验与体会

多使用百度谷歌等等工具，去看看有没有做好了的已经很方便的工具和算法并且利用好，好的工具可以节省大量的工作内容。

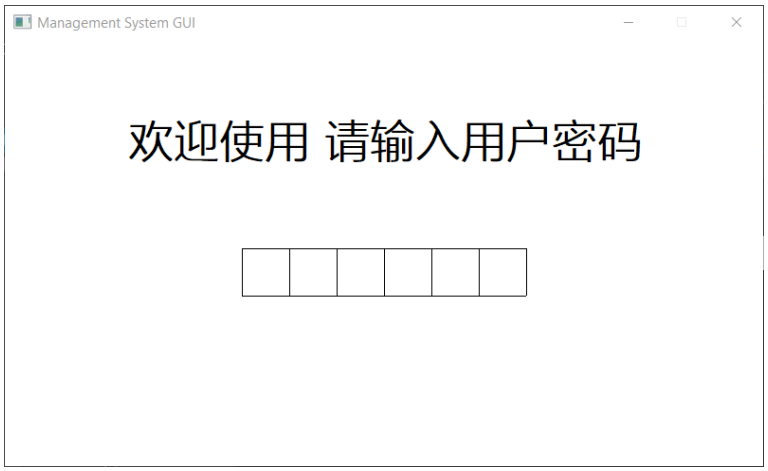
多去思考这些问题，从课堂上面脱离出来，课堂知识是告诉你思想，而商用的优秀算法一般在课堂上不会讲，需要自己去寻找。接下来通过自己的思考或者相关的讨论分析，书籍资料，网络工具等等方式进行学习。

6. 结果展示

欢迎界面（Author 刘荣江）

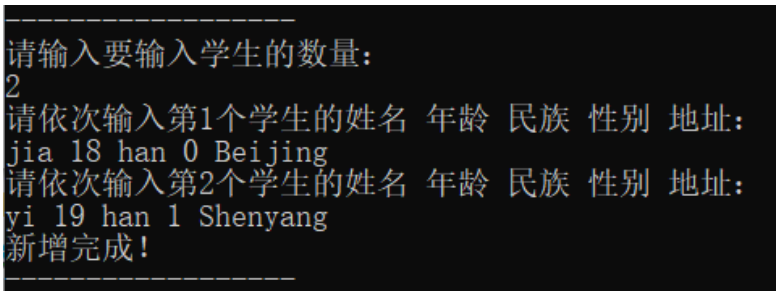


密码确认（Author 刘添夷）

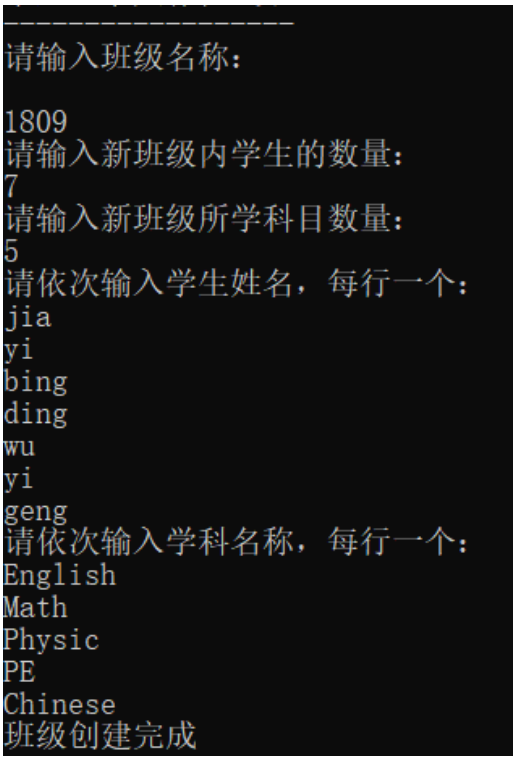


主菜单（Author 刘荣江）

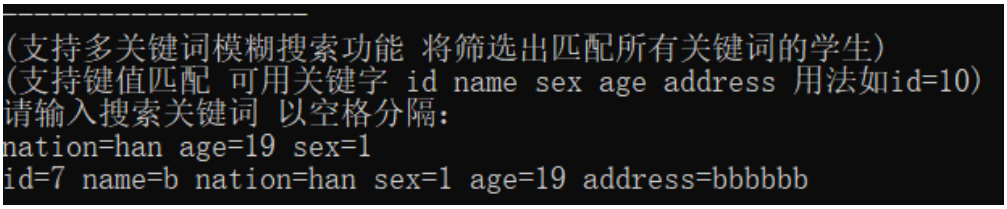
学生信息录入（Author 刘荣江）



班级创建（数据初始化） （Author 刘添夷）



学生查询（Author 刘添夷）



密码修改（Author 刘添夷）

成绩查询（Author 刘添夷）

```
------------------
(支持多关键词模糊搜索功能 将筛选出匹配所有关键词的记录)
(支持键值匹配 可用关键字 id name sex age address 用法如id=10)
请输入搜索关键词 以空格分隔:
class=1801
12  a  1801  sql  100
14  b  1801  sql  60
------------------
```

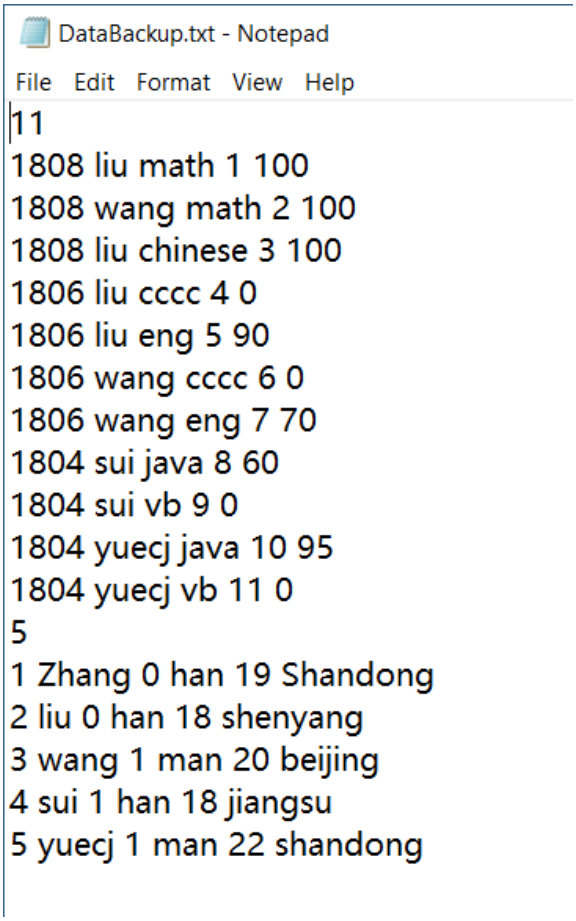
```
------------------
(支持多关键词模糊搜索功能 将筛选出匹配所有关键词的记录)
(支持键值匹配 可用关键字 id name sex age address 用法如id=10)
请输入搜索关键词 以空格分隔:
class=1808 subject=math
1  liu  1808  math  100
2  wang  1808  math  100
------------------
```

成绩录入/修改（Author 刘添夷）

```
(请输入修改成绩的范围)
(支持键值匹配 可用关键字 id name sex age address 用法如id=10)
请输入范围关键词 以空格分隔:
subject=java
--------------------
8  sui  1804  java :90

10  yuecj  1804  java :100
```

数据备份文件内容 （Author 刘荣江）



7. 总结与体会

7.1 成员 1 总结与体会

我对这次课程设计的体会：

- 总体上是一个从无到有，不断优化迭代的过程（推翻了好多次）
- 尝试使用 C 语言构建图形界面，并了解了 Windows 消息循环体系
- 认识到良好的可复用封装的重要性（同 GUI 一起有效缩减了重复代码行数），这是这次课程设计中最大的感触。观察别人的代码时，经常发现有大量语句使用“******”来绘制图形界面，函数的复用性也并不高，显然这样会使调试变得异常麻烦，也不利于代码的整洁。我们组在设计中努力贯彻简洁高效的思想，能不复制重复代码就不复制，数据结构能公用就公用（比如 Queue 中的缓存数据类型是 void* 可以承载任意类型的数据）。最后的感受是编写高度集成、简洁的代码不仅是一种好的能力，更是一种乐趣。

我认为目前存在的不足：

- 数据存储结构显然不是最优结构，线性结构之间的关系还需要进一步考虑
- 未能实现管理员和学生端权限分离
- 还应更多的学习算法和数据结构

7.2 成员 2 总结与体会

可以参考一些很好的开源项目，如何对于一个问题的解决要抱有做好而不仅仅是做完的态度，不断去优化，不断学习新东西，在我们的一生中都是很重要的。代码写注释，模块化也很重要，好看的代码可以在后期优化的时候更加方便，出现了新的需求修改的时候也很方便，debug 的时候有注释也可以更方便地去思考问题所在。