

信息科学与工程学院

《结构化程序设计》项目实践报告

学生信息系统的设计与实现

学科专业： 计算机科学与技术

小 组： SE-2

指导教师：  唐郑熠

成 员： 林俊海

吴喆涵

代子衡

曾麒麟

二零一七 年 一 月

**目 录**

[1 设计任务 2](#_Toc467316839)

[2 需求分析 3](#_Toc467316840)

[2.1 系统角色 3](#_Toc467316841)

[2.2 功能需求 3](#_Toc467316842)

[2.3 开发与运行环境 3](#_Toc467316843)

[2.3.1 开发环境 4](#_Toc467316844)

[2.3.2 运行环境 4](#_Toc467316845)

[3 系统概要设计 5](#_Toc467316846)

[3.1 数据结构设计 5](#_Toc467316847)

[3.2 系统总体设计 6](#_Toc467316848)

[4 系统详细设计 16](#_Toc467316849)

[4.1 模块设计 16](#_Toc467316850)

[4.2 模块结构 24](#_Toc467316851)

[5 系统测试 28](#_Toc467316852)

[6 总结与反思 41](#_Toc467316853)

[参考文献 4](#_Toc467316854)2

[致 谢 43](#_Toc467316855)

[附录 源代码 44](#_Toc467316856)

**1. 设计任务**

* 设计并实现一个学生管理系统，可以对高校学生的信息进行管理。
* 基本操作应至少包括：添加学生信息、删除学生信息、修改学生信息、浏览学生信息、查找学生信息。
* 学生属性应至少包括：学号、姓名、性别、籍贯、出生年月、专业、入学年份、所属年级、学籍状态（在读、休学、退学、服兵役）。
* 查找功能以学生的属性为查找关键字。
* 应实现根据所属年级进行批量删除的功能。
* 应以文件的形式对信息进行加密存储。

**2. 需求分析**

**2.1 系统角色**

* + 辅导员：管理学生成绩
  + 学生：修改管理自己的部分信息
  + 管理员：管理全部的辅导员和学生信息以及自己的部分信息

**2.2 功能需求**

学生管理系统，包括三个方面的需求，分别是学生的需求、辅导员的需求以及后台管理人员的需求。学生的需求是查询自己的信息，修改自己部分信息诸如联系方式、家庭住址、修改登录密码等。辅导员的需求是查询自己的信息，登记修改学生的成绩、修改登录密码以及修改自己部分信息诸如地址等。后台管理员人员的需求则是修改学生、辅导员除登录密码外所有信息以及自己的登录密码的修改。

基本功能：

1. 用户管理：能够对用户名、密码的简单验证；能够修改密码等操作。

2. 学生信息管理：能够对学生信息进行删除、添加、修改等操作。

3. 学生成绩管理：能够对学生成绩进行添加、修改、删除等操作。

4. 学生科目管理：能够对学生科目进行添加、修改、删除等操作。

5. 辅导员信息管理：能够对辅导员信息进行添加、修改、删除等操作。

6. 浏览查询：能够对学生信息、辅导员信息的查找、浏览等操作。

**2.3 开发与运行环境**

运行环境需求：电脑端：CPU 奔腾、运行内存 2G、硬盘空间等。

开发环境需求：操作系统 XP及以上、开发平台codeblocks、编译器GNU 、开发语言 C 、github等。

**2.3.1 开发环境**

开发环境统一为C语言。

**2.3.2 运行环境**

开发平台推荐Code::Blocks 16.01（或其他版本），GCC编译器。但也可选择其他熟悉的开发环境，如VC 6.0。

**3. 系统概要设计**

**3.1 数据结构设计**

账号的数据结构（struct \_ID）

1． id 字符串 账号

2. password 字符串 密码

3. flag 布尔值(char) 标记是否删除

科目信息的数据结构（struct \_subject）

1. name 字符串 学科名称
2. mark 浮点型数据 学科分数
3. flag 布尔值(char) 标记是否删除

班级的数据结构（struct \_class）

1. name 字符串 班级名称
2. Flag 布尔值(char) 标记是否删除

学生信息的数据结构（struct \_student）

1. struct \_ID 结构体 账号的数据结构
2. name 字符串 学生姓名
3. class 字符串 学生班级
4. address 字符串 学生地址
5. phoneNumber 字符串 学生电话号码
6. Struct\_subject 结构体 科目信息
7. SubjectNum 无符号常整型 科目数量
8. nativePlace 字符串 籍贯
9. birthday 字符串 生日
10. major 字符串 专业
11. YearOfEnrollment 字符串 入学年份
12. enum \_enum\_sex 枚举类型 性别
13. enum \_enum\_status 枚举类型 在读状态

辅导员信息的数据结构（struct \_counsellor）

1. struct \_ID 结构体 账号的数据结构
2. name 字符串 辅导员姓名
3. address 字符串 辅导员地址
4. struct\_class 结构体 辅导员管理班级
5. manageClassNum 无符号常整型 辅导员管理班级数目

管理员的数据结构（struct \_admin）

1. struct \_ID 结构体 账号的数据结构

文件系统的数据结构（struct \_fileStruct）

1. adminInfoFile 字符串 管理员信息的文件
2. studentInfoFile 字符串 学生信息的文件
3. counsellorInfoFile 字符串 辅导员信息的文件
4. profile 字符串 配置文件

学生信息管理系统的数据结构

1. FileStruct 结构体 文件信息
2. pStudent 结构体 学生数组信息
3. studentLen 无符号常整型 学生数组长度
4. Admin 结构体 管理员信息
5. pCounsellor 结构体 辅导员信息
6. counsellorLen 无符号常整型 辅导员数组长度

**3.2 系统总体设计**

[用例1]

绘制系统总体功能构成，如图一。



[用例1图]

[用例2]



[用例2图]

|  |  |
| --- | --- |
| **功能名** | 登陆 |
| **参与角色** | 老师、学生、管理员 |
| **事件流** | |
| 1. 输入用户名、密码  2. 登录成功  2.1 密码错误超过三次，退出程序 | |

|  |  |
| --- | --- |
| **功能名** | 修改密码 |
| **参与角色** | 老师、学生、管理员 |
| **事件流** | |
| 1. 输入原密码   1.1 密码错误三次自动退出修改密码界面  2. 输入两次新密码  2.1 两次新密码一致，修改成功  2.2 两次新密码不一致，选择继续修改或放弃 | |

[用例3]



[用例3图]

|  |  |
| --- | --- |
| **功能名** | 删除学生信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 是否删除    1. 删除信息    2. 不删除信息 | |

|  |  |
| --- | --- |
| **功能名** | 修改学生信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 修改学生信息（学号，姓名，班级，成绩，籍贯，手机号） 2. 确认修改信息    1. 修改信息    2. 不修改信息 | |

|  |  |
| --- | --- |
| **功能名** | 添加学生信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 添加学生信息（学号，姓名，班级，成绩，籍贯，手机号） 2. 确认添加    1. 添加    2. 不添加 | |

[用例4]



[用例4图]

|  |  |
| --- | --- |
| **功能名** | 添加科目信息 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1. 添加科目信息（科目名称、科目总分、绩点） 2. 确认添加   2.1 添加  2.2 不添加 | |

|  |  |
| --- | --- |
| **功能名** | 修改科目信息 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1. 修改科目信息 2. 确认修改信息   2.1 修改信息  2.2 不修改信息 | |

|  |  |
| --- | --- |
| **功能名** | 删除科目 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1．是否删除   * 1. 删除科目   2. 不删除信息 | |

[用例5]



[用例5图]

|  |  |
| --- | --- |
| **功能名** | 添加成绩信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 添加成绩 2. 确认添加   2.1 添加  2.2 不添加 | |

|  |  |
| --- | --- |
| **功能名** | 修改成绩信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 修改成绩 2. 确认修改信息   2.1 修改信息  2.2 不修改信息 | |

|  |  |
| --- | --- |
| **功能名** | 删除成绩信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1．是否删除   * 1. 删除科目   2. 不删除信息 | |

[用例6]



[用例6图]

|  |  |
| --- | --- |
| **功能名** | 按学号查找学生信息 |
| **参与角色** | 管理员、辅导员、学生 |
| **事件流** | |
| 1. 输入学号，查找    1. 查无此记录，请重新输入或选择退出 2. 输出学生信息 | |

|  |  |
| --- | --- |
| **功能名** | 按姓名查找学生信息 |
| **参与角色** | 管理员、辅导员、学生 |
| **事件流** | |
| 1. 输入姓名，查找   1.1 查无此记录，请重新输入或选择退出  2. 输出学生信息 | |

|  |  |
| --- | --- |
| **功能名** | 显示学生基本信息 |
| **参与角色** | 管理员、辅导员、学生 |
| **事件流** | |
| 1. 显示学号、姓名、班级 2. 显示成绩（各科目成绩） | |

|  |  |
| --- | --- |
| **功能名** | 显示学生详细信息 |
| **参与角色** | 管理员、辅导员、学生 |
| **事件流** | |
| 1. 显示学号、姓名、班级 2. 显示籍贯、手机号等 3. 显示成绩（各科目成绩） | |

|  |  |
| --- | --- |
| **功能名** | 按编号查找辅导员信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 输入编号，查找   1.1 查无此记录，请重新输入或选择退出  2. 输出辅导员信息 | |

|  |  |
| --- | --- |
| **功能名** | 按姓名查找辅导员信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1．输入姓名，查找  1.1 查无此记录，请重新输入或选择退出  2. 输出辅导员信息 | |

|  |  |
| --- | --- |
| **功能名** | 显示辅导员详细信息 |
| **参与角色** | 管理员、辅导员 |
| **事件流** | |
| 1. 显示编号、姓名 2. 显示籍贯、手机号等 3. 显示管理班级名称 | |

[用例7]



[用例7图]

|  |  |
| --- | --- |
| **功能名** | 删除辅导员信息 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1. 是否删除    1. 删除    2. 不删除 | |

|  |  |
| --- | --- |
| **功能名** | 添加辅导员信息 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1. 添加辅导员信息（姓名，编号，所管班级，手机号） 2. 确认是否添加    1. 确认添加    2. 不添加 | |

|  |  |
| --- | --- |
| **功能名** | 修改辅导员信息 |
| **参与角色** | 管理员 |
| **事件流** | |
| 1. 修改辅导员信息（姓名，编号，所管班级，手机号） 2. 确认是否修改    1. 确认修改    2. 不修改 | |

**4 系统详细设计**

* 1. **模块设计**

[用例1]

用户管理：登录、修改密码。

以学生身份登录认证

const student pStudentArray[] 参数指 学生数组

const size\_t len 参数指 学生数组的长度

size\_t \*const pIndex 参数指认证成功，获得该学生在学生数组中的下标

函数返回值 bool 型，若返回值为真，则表示认证成功，否则失败，pIndex 值无意义

bool LoginByStudent(const student pStudentArray[], const size\_t len, size\_t \*const pIndex);

以辅导员身份登录认证

const counsellor pCounsellorArray[] 参数指 辅导员数组

const size\_t len 参数指 辅导员数组的长度

size\_t \*const pIndex 参数指认证成功，获得该辅导员在辅导员数组中的下标

函数返回值 bool 型， 若返回值为真，则表示认证成功，否则失败，pIndex 值无意义

bool LoginByCounsellor(const counsellor pCounsellorArray[], const size\_t len, size\_t \*const pIndex);

以管理员身份登录认证

const admin \*const pAdmin 参数指 管理员指针

函数返回值 bool 型， 若返回值为真， 则表示认证成功， 否则失败

bool LoginByAdmin(const admin \*const pAdmin);

检查ID 是否一致

参数 const ID \*const pIDA 指 账号A

参数 const ID \*const pIDB 指 账号B

该账号A 和 账号B 相等， 则返回值为 真， 否则为假

bool CheckID(const ID \*const pIDA, const ID \*const pIDB);

修改密码

参数 ID \*const pID 为待修改的账号指针

void ChangePassword(ID \*const pID);

[用例2]

辅导员信息管理：添加辅导员信息、删除辅导员信息、修改辅导员信息。

删除辅导员信息， 通过标记 flag惰性删除

counsellor pcounsellorArray[ ] 参数指辅导员数组指针

const size\_t len 参数指 辅导员数组的长度

void DeleteCounsellorInfo(counsellor pCounsellorArray[ ], const size\_t len);

添加辅导员信息

counsellor \*\*pCounsellorArray 参数指辅导员数组指针的指针

size\_t \*const pConLen 参数指辅导员是数组的长度

void AddCounsellorInfo(counsellor \*\*pCounsellorArray, size\_t \*const pConLen);

修改辅导员信息

counsellor \*const pcounsellor 参数指修改当前辅导员信息

void ChangeCounsellorInfo(counsellor \*const pCounsellor);

删除 辅导员数组信息

counsellor \*\*pCounsellorArray 参数指指向辅导员数组指针的指针

size\_t \*pLen 参数指辅导员数组长度的指针

void FreeCounsellorArray(counsellor \*\*pCounsellorArray, size\_t \*pLen);

[用例3]

学生信息管理：删除学生信息、添加学生信息、修改学生信息。

删除学生信息，通过标记flag惰性删除

student \*const pStudentArray 参数指学生数组指针

const size\_t Len 参数指学生数组的长度

void DeleteStudentInfo(student \*const pStudentArray, const size\_t len);

添加学生信息

student \*\*pStudentArray 参数指学生数组指针的指针

size\_t \*const pConLen 参数指学生是数组的长度

void AddStudentInfo(student \*\*pStudentArray, size\_t \*const pConLen);

修改学生信息

student \*const pStudent 参数指修改当前学生信息

void ChangeStudentInfo(student \*const pStudent);

删除 学生数组信息

student \*\*pStudentArray 参数指指向学生数组指针的指针

size\_t \*pLen 参数指学生数组长度的指针

void FreeStudentArray(student \*\*pStudentArray, size\_t \*pLen);

[用例4]

浏览、查询：浏览、查询学生、辅导员信息。

查找学生信息

const student \*const pStudentArray 参数指学生数组指针

const size\_t len 参数指学生数组的长度

size\_t \*const locateIndex 参数指找到学生在数组中的下标

函数返回值 bool 值指是否在学生数组中找到该学生信息，若没有找到，locateIndex 值没有意义

bool LocateStudentInfo(const student \*const pStudentArray, const size\_t len, size\_t \*const pLocateIndex)

通过 ID 查找学生信息

const student \*const pStudentArray 参数指学生数组指针

const size\_t len 参数指学生数组的长度

const ID \*const id 参数指需要找到的学生的ID指针

size\_t \*const locateIndex 参数指找到学生在数组中的下标

函数返回值 bool 值指是否在学生数组中找到该学生信息，若没有找到，locateIndex 值没有意义

bool LocateStudentInfoByID(const student \*const pStudentArray, const size\_t len, const ID \*const pID, size\_t \*const pLocateIndex);

通过姓名查找学生信息

const student \*const pStudentArray 参数指学生数组指针

const size\_t len 参数指学生数组的长度

const char \*const name 参数指需要找到的学生的姓名

size\_t \*const pLocateIndex 参数指找到学生在数组中的下标

函数返回值 bool 值指是否在学生数组中找到该学生信息，若没有找到，locateIndex值没有意义

bool LocateStudentInfoByName(const student \*const pStudentArray, const size\_t len, const char \*const name, size\_t \*const pLocateIndex);

显示学生信息

const student \*pStudent 参数指需要显示的学生指针

void DispStudentInfo(const student \*const pStudent);

显示所有学生信息

const student \*pStudentArray 参数指需要显示的学生数组指针

const size\_t conLen 参数指需要显示的学生数组的长度

void DispAllStudentInfo(const student \*const pStudentArray, const size\_t conLen);

查找辅导员信息

const counsellor \*const pCounsellorArray 参数指辅导员数组指针

const size\_t len 参数指辅导员数组的长度

size\_t \*const locateIndex 参数指找到辅导员在数组中的下标

函数返回值bool值 指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex 值没有意义

bool LocateCounsellorInfo(const counsellor \*const pCounsellorArray, const size\_t len, size\_t \*const pLocateIndex);

通过 ID 查找辅导员信息

const counsellor \*const pCounsellorArray 参数指辅导员数组指针

const size\_t len 参数指辅导员数组的长度

const ID \*const id 参数指需要找到的辅导员的ID指针

size\_t \*const locateIndex 参数指找到辅导员在数组中的下标

函数返回bool值指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex值没有意义

bool LocateCounsellorInfoByID(const counsellor \*const pCounsellorArray, const size\_t len, const ID \*const pID, size\_t \*const pLocateIndex);

通过姓名查找辅导员信息

const counsellor \*const pCounsellorArray 参数指辅导员数组指针

const size\_t len 参数指辅导员数组的长度

const char \*const name 参数指需要找到的辅导员的姓名

size\_t \*const pLocateIndex 参数指找到辅导员在数组中的下标

函数返回值bool值指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex 值没有意义

bool LocateCounsellorInfoByName(const counsellor \*const pCounsellorArray, const size\_t len, const char \*const name, size\_t \*const pLocateIndex);

显示辅导员信息

const counsellor \*pcounsellor 参数指需要显示的辅导员指针

void DispCounsellorInfo(const counsellor \*const pCounsellor);

显示所有辅导员信息

const counsellor \*pcounsellorArray 参数指需要显示的辅导员数组指针

const size\_t conLen 参数指需要显示的辅导员数组的长度

void DispAllCounsellorInfo(const counsellor \*const pCounsellorArray, const size\_t conLen);

[用例5]

学生成绩管理

添加单个学生的成绩

const student \*const pStudent 参数指学生的指针

void AddMark(student \*const pStudent);

修改 单个学生的成绩

const student \*const pStudent 参数指学生的指针

void ChangeMark(student \*const pStudent);

[用例6]

学生科目管理

删除所有学生科目信息， 通过标记flag惰性删除

student pStudentArray[ ] 参数指学生数组指针

const size\_t Len 参数指 学生数组的长度

void DeleteAllStudentSubjectInfo(student pStudentArray[ ], const size\_t len);

添加所有学生科目信息，通过修改已标记删除的科目或者重新分配空间

student pStudentArray[ ] 参数指学生数组指针

const size\_t len 参数指学生是数组的长度

void AddAllStudentSubjectInfo(student pStudentArray[], const size\_t len);

修改所有学生科目信息，通过查找每个学生的科目信息，如若找到符合的科目信息，则修改。

student pStudentArray[ ] 参数指学生数组指针

const size\_t len 参数指学生是数组的长度

void ChangeAllStudentSubjectInfo(student pStudentArray[ ], const size\_t len);

删除单个学生科目信息，通过标记flag惰性删除

const Student \*const pStudent 参数指学生指针

void DeleteSingleStudentSubjectInfo(student \*const pStudent);

添加单个学生科目信息，通过修改已标记删除的科目或者重新分配空间

const student \*const pStudent 参数指 学生指针

void AddSingleStudentSubjectInfo(student \*const pStudent);

修改单个学生科目信息，通过查找该学生的科目信息，如若找到符合的科目信息，则修改。

student pStudent[ ] 参数指 学生指针

函数返回值 bool 若为真，则修改成功，否则修改失败或没有找到符合的科目

bool ChangeSingleStudentSubjectInfo(student \*const pStudent);

**4.2** **模块结构**

通过图形来描述函数模块之间的调用关系。

















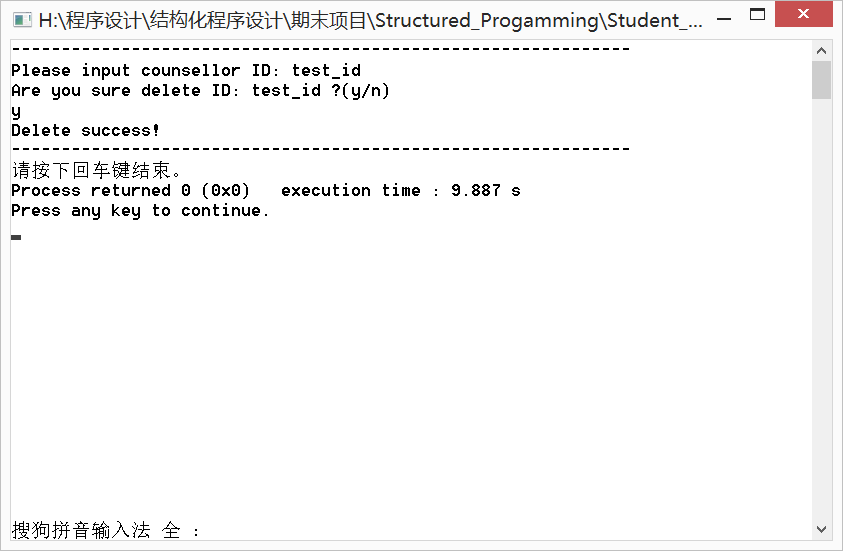


**5. 系统测试**

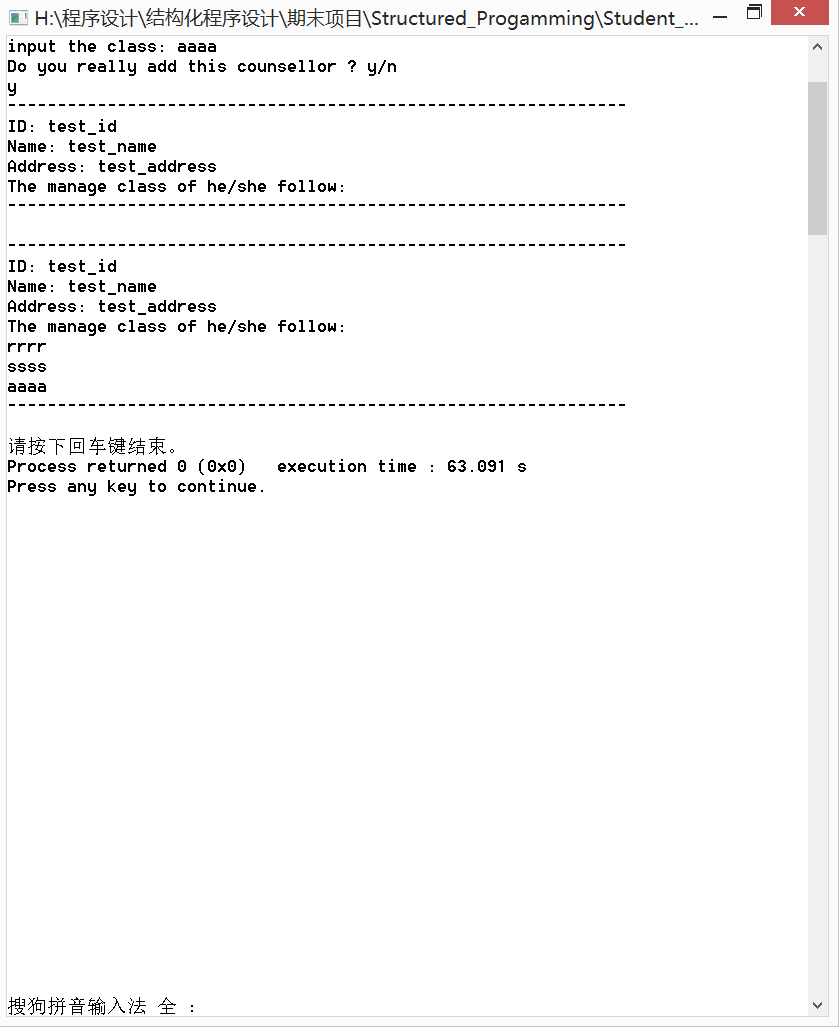
该部分结合插图演示的整个运行流程，要涉及到所实现的全部功能。

1.CounsellorInfoManage\_test():

DeleteCounsellorInfo\_test():

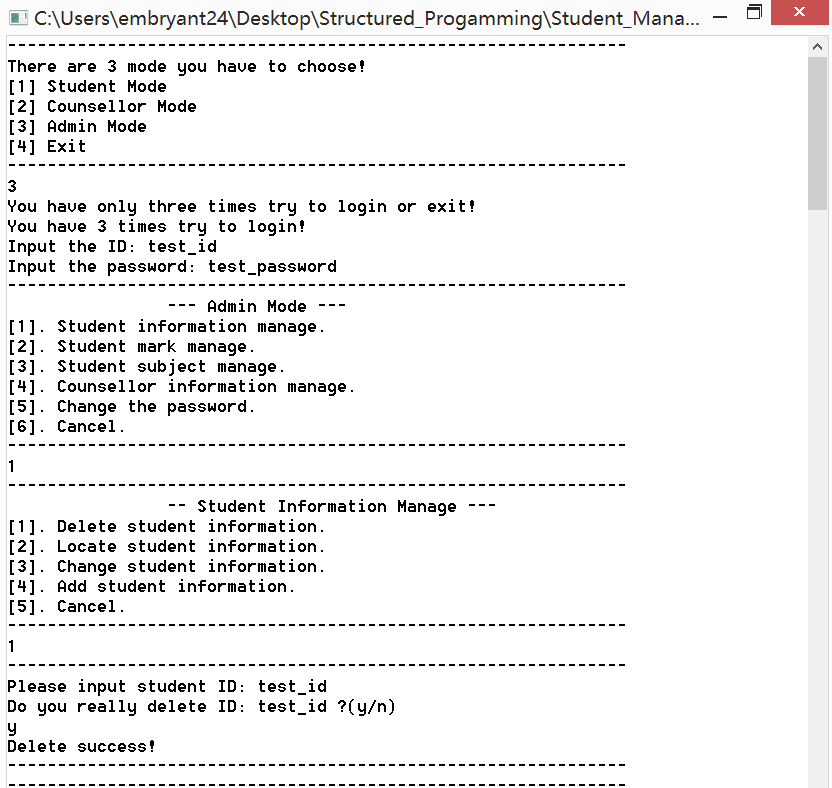


AddCounsellorInfo\_test():

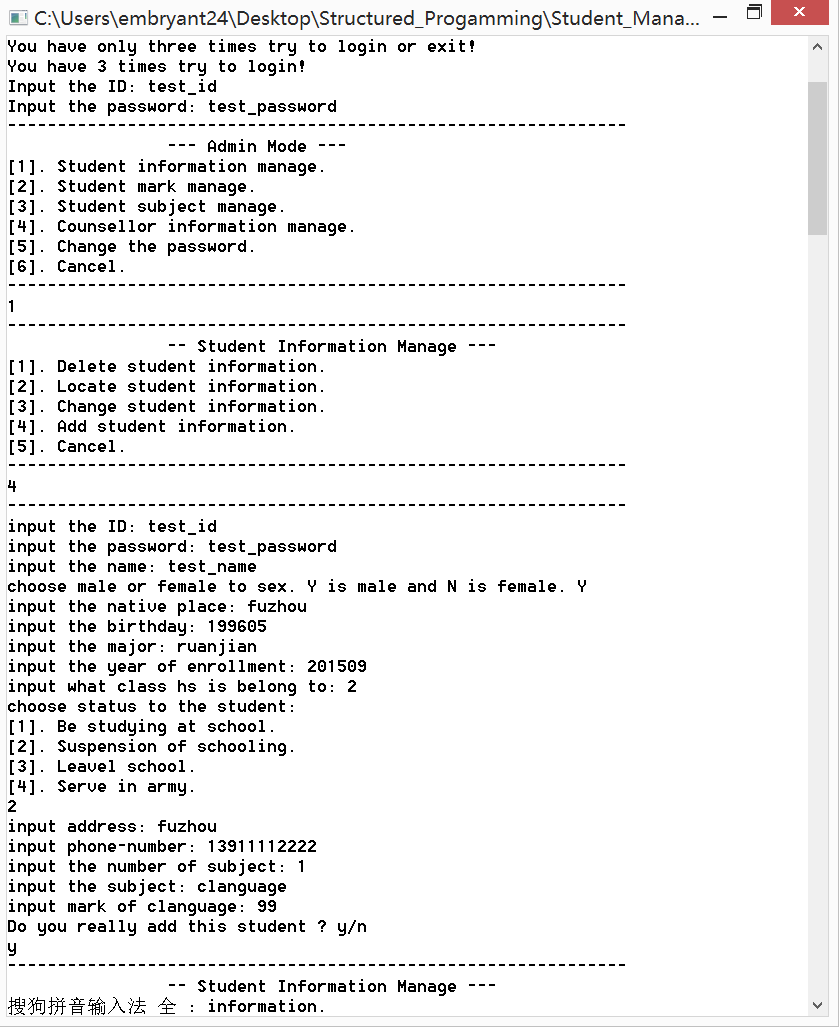


4. StudentInfoManage\_Test():

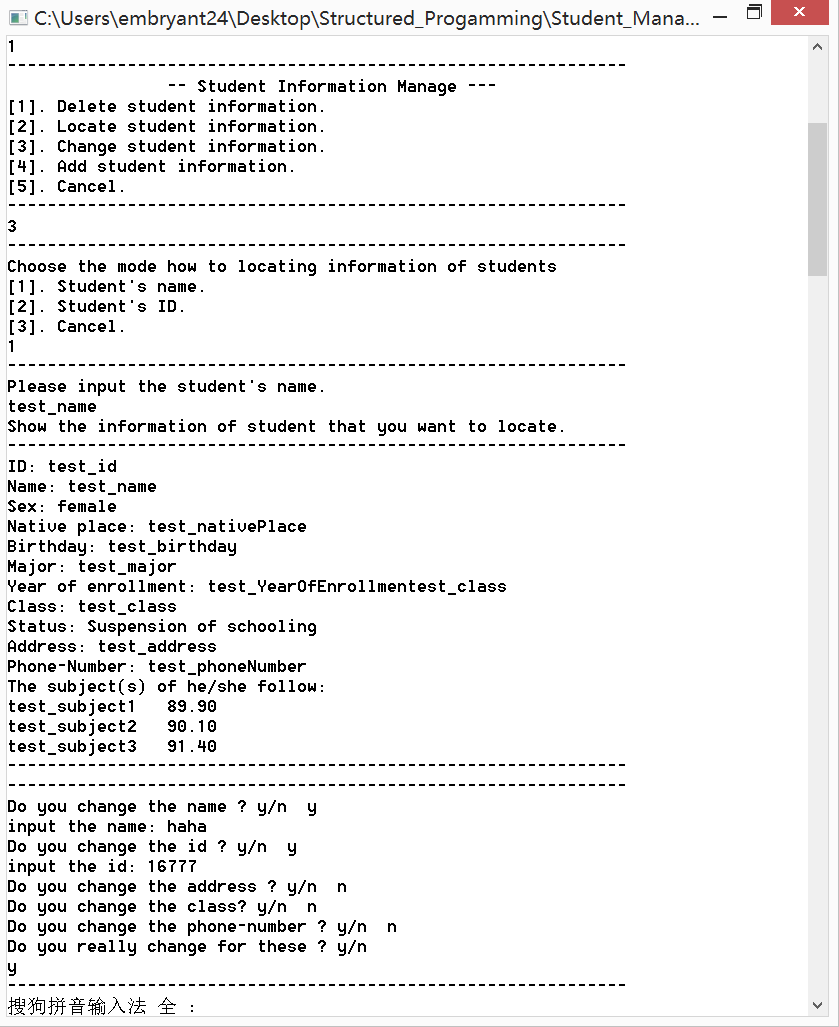
DeleteStudentInfo\_Test();



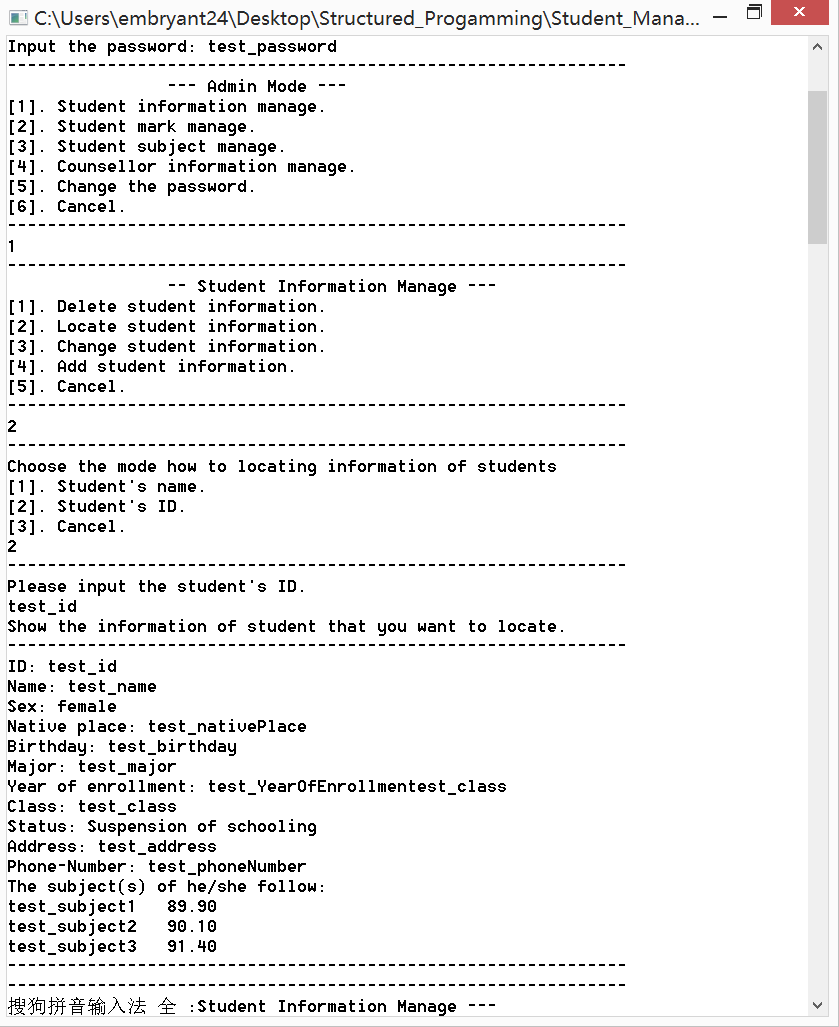
AddStudentInfo\_Test();



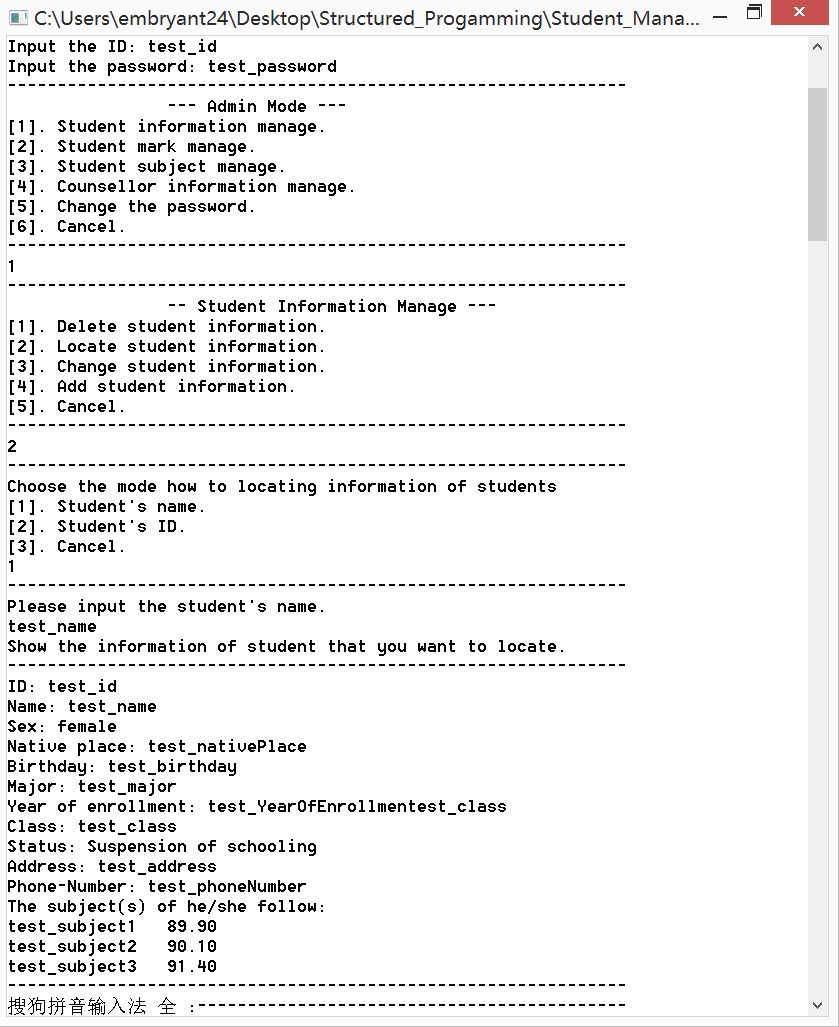
ChangeStudentInfo\_Test();



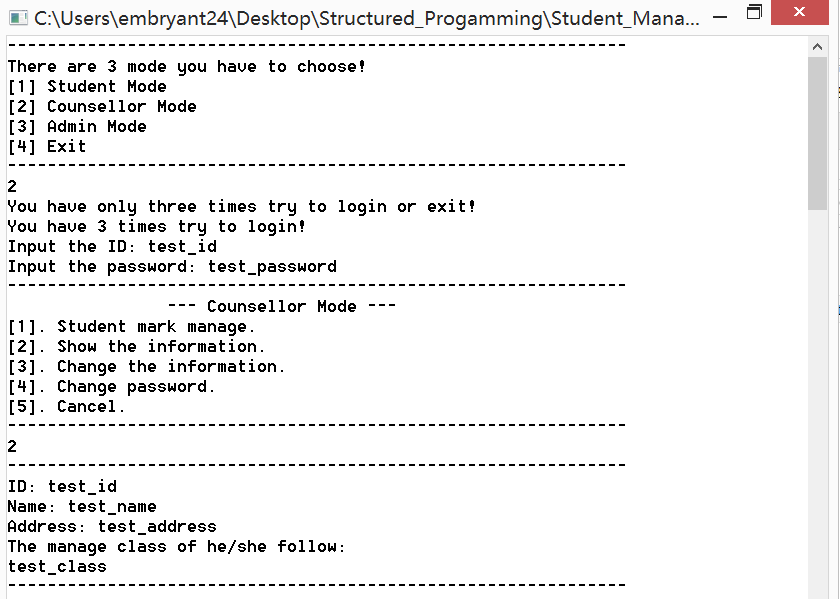
LocateStudentInfoByID\_Test();



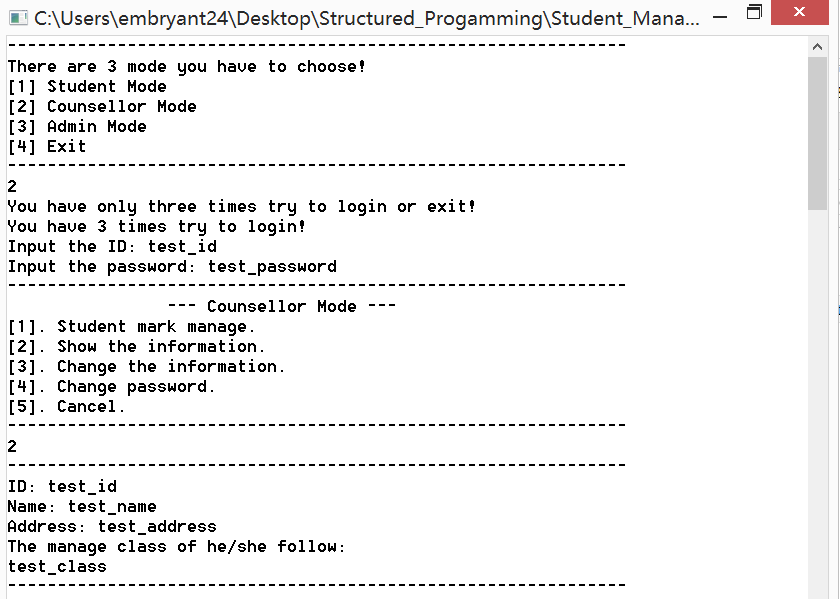
LocateStudentInfoByName\_Test();



DispStudentInfo;

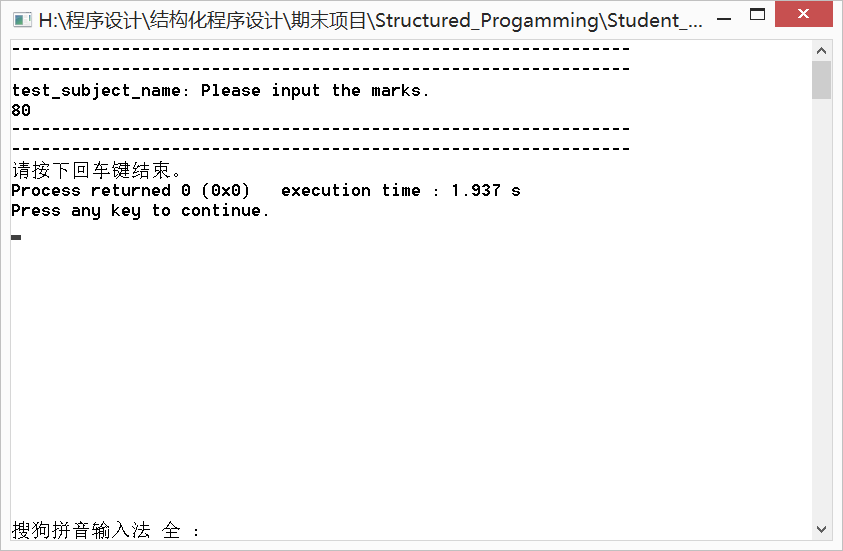


DispAllStudentInfo

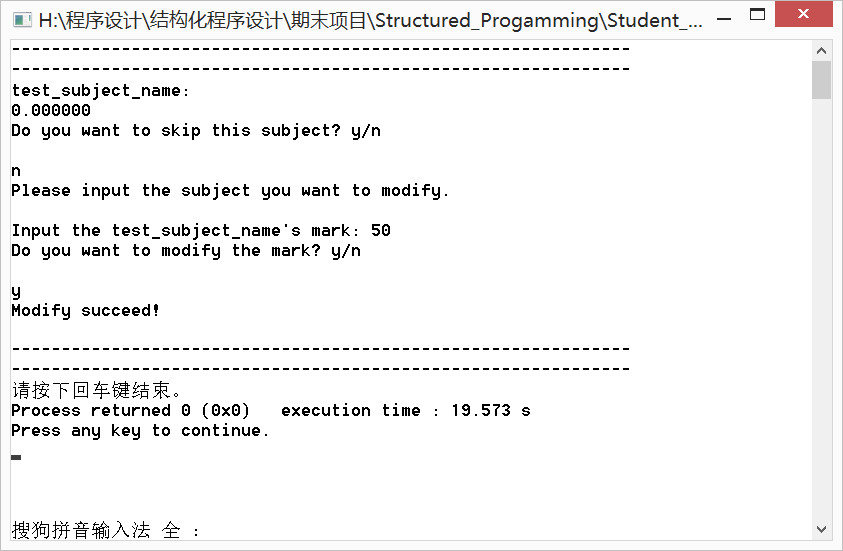


5. StudentMarkManage\_Test():

AddMark\_Test();

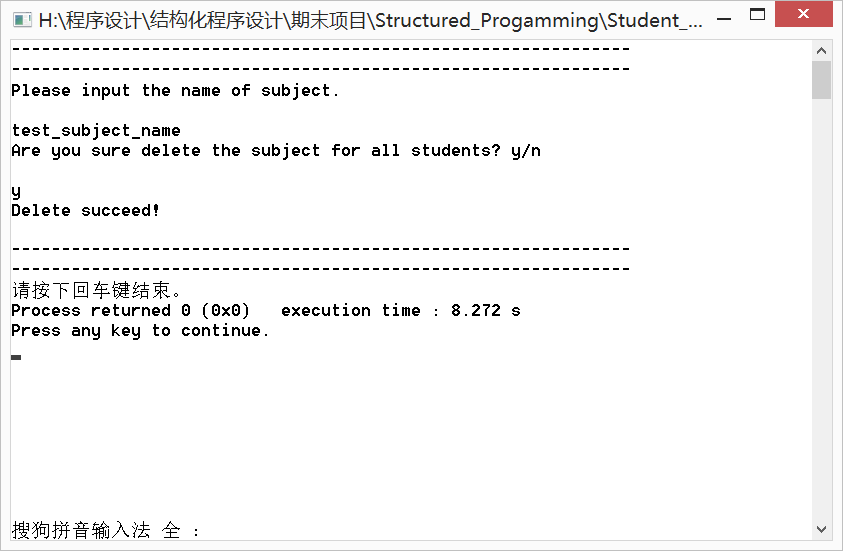


ChangeMark\_Test();

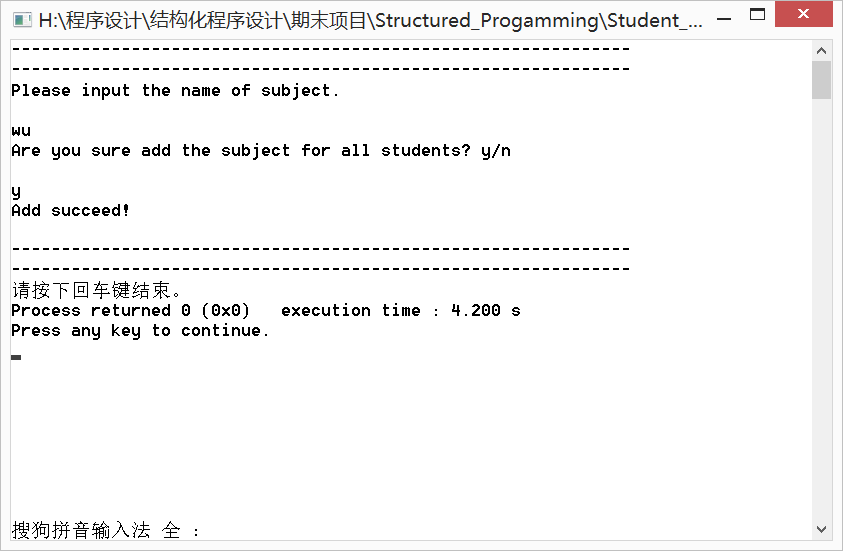


6. StudentSubjectManage\_Test():

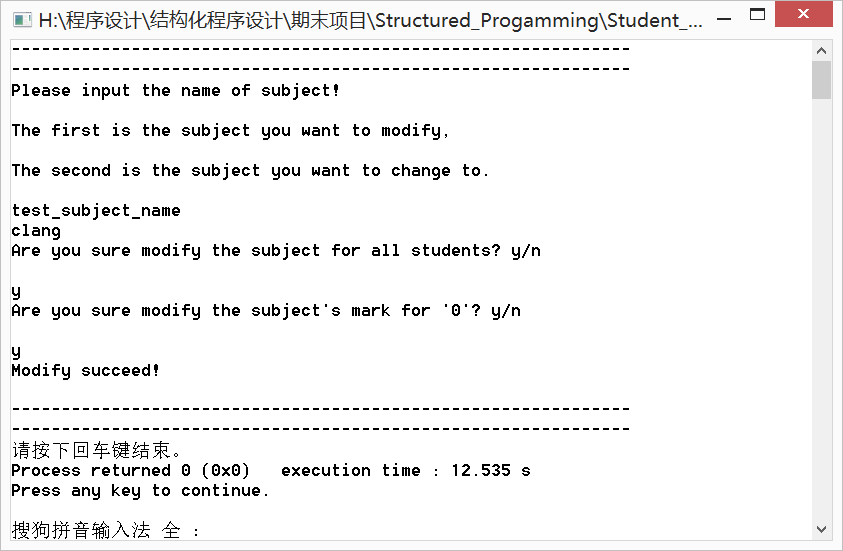
DeleteAllStudentSubjectInfo\_Test();



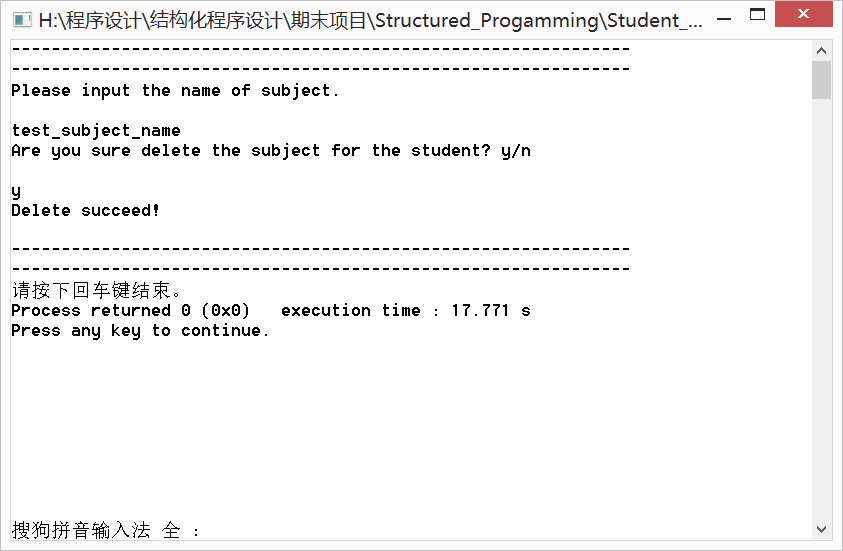
AddAllStudentSubjectInfo\_Test();



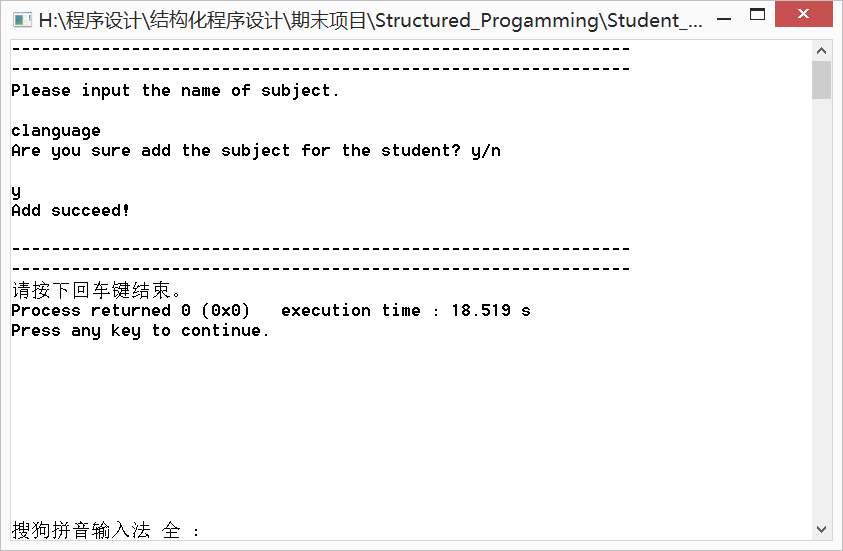
ChangeAllStudentSubjectInfo\_Test();



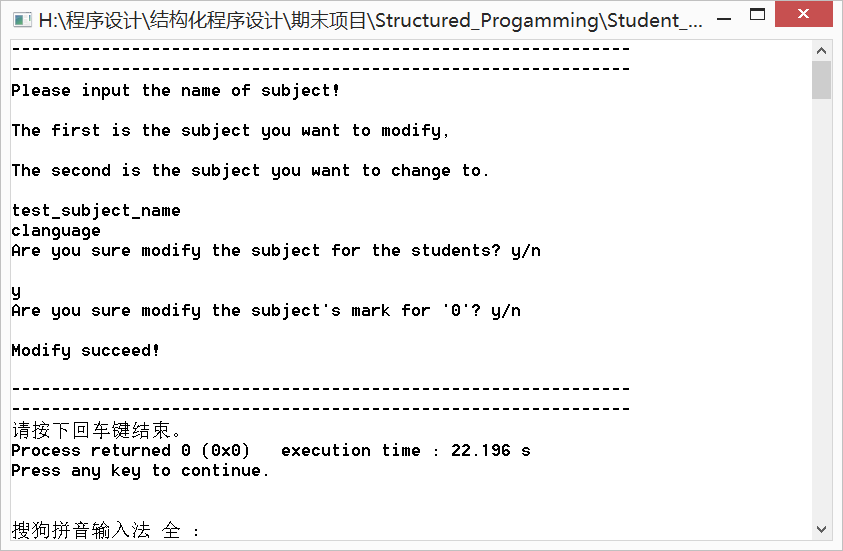
DeleteSingleStudentSubjectInfo\_Test();



AddSingleStudentSubjectInfo\_Test();



ChangeSingleStudentSubjectInfo\_Test();



**6. 总结与反思**

由于时间和技术水平的原因，在本系统设计上，还不能将全部功能设计得非常完善，还有一些功能模块有待于进一步的完善。同时，在本系统中还缺少了一部分必要的功能模块,例如系统的操作日志。这一系列工作都有待于今后在工作中逐步地加以补充和完善。

只要生命不停止，学习永不停止。有了知识才不会觉得虚拟，有了知识才会觉得充实。

在整个项目的建设与编程的过程中，我们深深地体会到了“学以致用”这句话的含义。过去，在课堂学习的过程，有很多知识我们错误地认为并不需要，甚至感到是浪费时间，但是在实践过程中，我们才发现课堂上学习的内容还远远不足，许多问题在编程的过程中，一一暴露出来。一些平时容易忽视的细节往往成为我们编程的一道障碍，百思不得其解。通过查阅了大量的资料和请教了指导老师以后我们才发现原来差错只是小小的一个，这更加促使我们经常提醒自己要做到事无巨细，处处留心。编程并不是一件很困难的事情，只要在实践中多加体会和运用，一定能越来越精。然而在编程前的系统分析和设计方面却需要我们们花费足够的耐心，考虑周详，从这次的学习和实践中，我们体会到了要能够适应这个突飞猛进的科技时代，真正困难的是如何能够不断地学习，提高自己的知识水平，来适应社会的需要。

相信在今后的学习过程中，我们一定要以仔细谨慎的思想来对待每一件事情，力争成为一个能为社会做出贡献的有用之人。

该部分对课程设计的全程进行总结，说明通过课程设计得到的锻炼、收获、感悟或经验。同时，也思考一下整个过程中可以改进的地方。

参考文献

1. 门鹏, 段振华. 一种基于Petri网的自动Web服务组合算法[J]. 西安电子科技大学学报. 2008, 35(4): 609-613.
2. 谭浩强. C程序设计[M]. 第3版. 北京: 清华大学出版社. 2005.
3. 孙昌爱, 尚岩, 李飞. 事务感知的Web服务开发框架[C]. 全国软件与应用学术会议. 2011: 6-11.
4. 邓水光. Web服务自动组合与形式化验证的研究[D]. 杭州: 浙江大学, 2007: 15-28.
5. W3C. OWL Web Ontology Language Overview. 2004: http://www.w3.org/TR/2004/REC-owl-

features-20040210.

1. 范玉顺, 倪悦. 语义Web服务组合的模型转换及形式化验证方法[P]. 中国: ZL200910236364.9, 2009-10-20.
2. G. Behrmann, A. David, and K. G. Larsen. A Tutorial on Uppaal4.0[R]. 2006.

# 致 谢

在报告完成之际，我们要特别感谢我们的指导老师唐郑熠老师的热情关怀和悉心指导。在我们撰写报告的过程中，唐郑熠老师倾注了大量的心血和汗水，无论是在报告的选题、构思和资料的收集方面，还是在报告的研究方法以及成文定稿方面，我们都得到了唐郑熠老师悉心细致的教诲和无私的帮助，特别是他广博的学识、深厚的学术素养、严谨的治学精神和一丝不苟的工作作风使我们终生受益，在此表示真诚地感谢和深深的谢意。

在报告的写作过程中，也得到了许多同学的宝贵建议，同时还到许多在工作过程中许多同事的支持和帮助，在此一并致以诚挚的谢意。

感谢所有关心、支持、帮助过我们的良师益友。

附录

程序源代码：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define NAME\_LEN 20

#define PASSWORD\_LEN 20

#define PHONENUMBER\_LEN 20

#define ADDRESS\_LEN 20

#define SUBJECT\_NAME\_LEN 20

#define ID\_LEN 20

#define CLASS\_NAME\_LEN 20

#define NATIVE\_PLACE\_LEN 20

#define BIRTHDAY\_LEN 20

#define MAJOR\_LEN 20

#define YEAR\_OF\_ENROLLMENT\_LEN 20

#define STRINGBUFFERLEN 256

#define true 1

#define false 0

typedef struct \_subject subject;

typedef struct \_student student;

typedef struct \_counsellor counsellor;

typedef struct \_ID ID;

typedef struct \_admin admin;

typedef struct \_class class;

typedef struct \_fileStruct fileStruct;

typedef struct \_SMS SMS;

typedef enum \_enum\_status enum\_status;

typedef enum \_enum\_sex enum\_sex;

typedef char bool;

enum \_enum\_sex

{

Male,

Female

};

enum \_enum\_status

{

Be\_Studying\_At\_School, // 在读

Suspension\_of\_Schooling, // 休学

Leavel\_School, // 退学

Serve\_in\_Army // 服兵役

};

// 该系统中所有人的账号信息

struct \_ID

{

char id[ID\_LEN];

char password[PASSWORD\_LEN];

char flag;

};

// 科目信息

struct \_subject

{

char name[NAME\_LEN];

float mark;

bool flag;

};

// 班级信息

struct \_class

{

char name[NAME\_LEN];

bool flag;

};

// 学生信息

struct \_student

{

struct \_ID id;

char name[NAME\_LEN];

enum\_sex sex;

char nativePlace[NATIVE\_PLACE\_LEN];

char birthday[BIRTHDAY\_LEN];

char major[MAJOR\_LEN];

char YearOfEnrollment[YEAR\_OF\_ENROLLMENT\_LEN];

char class[CLASS\_NAME\_LEN];

enum\_status status;

char address[ADDRESS\_LEN];

char phoneNumber[PHONENUMBER\_LEN];

struct \_subject \*pSubject;

size\_t subjectNum;

};

// 辅导员信息

struct \_counsellor

{

struct \_ID id;

char name[NAME\_LEN];

char address[ADDRESS\_LEN];

struct \_class \*pClass;

size\_t manageClassNum;

};

// 管理员信息

struct \_admin

{

struct \_ID id;

};

// 文件系统

struct \_fileStruct

{

char adminInfoFile[STRINGBUFFERLEN];

char studentInfoFile[STRINGBUFFERLEN];

char counsellorInfoFile[STRINGBUFFERLEN];

char profile[STRINGBUFFERLEN];

};

// 学生信息管理系统结构体

struct \_SMS

{

fileStruct FileStruct;

student \*pStudent;

size\_t studentLen;

admin Admin;

counsellor \*pCounsellor;

size\_t counsellorLen;

};

inline void PrintSplitLine()

{

puts("--------------------------------------------------------------");

}

inline bool YesOrNo()

{

char ch = '0';

while(ch != 'y' && ch != 'Y' && ch != 'N' && ch != 'n') {

ch = getchar();

}

if(ch == 'y' || ch == 'Y') {

return true;

}

return false;

}

// 删除辅导员信息， 通过 标记 flag 惰性删除

// counsellor pcounsellorArray[] 参数指辅导员数组指针

// const size\_t len 参数指 辅导员数组的长度

void DeleteCounsellorInfo(counsellor pCounsellorArray[], const size\_t len)

{

assert(pCounsellorArray && "the pointer is NULL!");

char ID[ID\_LEN];

bool f,k;

size\_t i;

PrintSplitLine();

printf("Please input counsellor ID: ");

scanf("%s", ID);

printf("Are you sure delete ID: %s ?(y/n)\n", ID);

f = YesOrNo();

if(f){

k = false;

for(i=0; i < len; ++ i) {

if(pCounsellorArray[i].id.flag && !strcmp(pCounsellorArray[i].id.id, ID)) {

k = true;

pCounsellorArray[i].id.flag = false;

}

}

if(k) {

puts("Delete success!");

}

else {

puts("Cann't find this ID, please check it out!");

}

}

PrintSplitLine();

}

// 添加辅导员信息

// counsellor \*\*pCounsellorArray 参数指 辅导员数组指针的指针

// size\_t \*const pConLen 参数指 辅导员是数组的长度

void AddCounsellorInfo(counsellor \*\*pCounsellorArray, size\_t \*const pConLen)

{

assert(pCounsellorArray && "The pointer is NULL");

assert(pConLen && "The pointer is NULL");

size\_t i;

counsellor inputCounsellor;

bool AddFlag, DelFlag;

PrintSplitLine();

printf("input the ID: ");

scanf("%s", inputCounsellor.id.id);

printf("input the password: ");

scanf("%s", inputCounsellor.id.password);

inputCounsellor.id.flag = true;

printf("input the name: ");

scanf("%s", inputCounsellor.name);

printf("input address: ");

scanf("%s", inputCounsellor.address);

printf("input the number of class: ");

scanf("%u", &inputCounsellor.manageClassNum);

inputCounsellor.pClass = (class \*)malloc(inputCounsellor.manageClassNum \* sizeof(class));

assert(!(inputCounsellor.manageClassNum != 0 && inputCounsellor.pClass == NULL)

&& "Cann't alloc the class memory in AddStudentInfo function!");

for(i=0; i < inputCounsellor.manageClassNum; ++i) {

printf("input the class: ");

scanf("%s", inputCounsellor.pClass[i].name);

inputCounsellor.pClass[i].flag = true;

}

puts("Do you really add this counsellor ? y/n");

AddFlag = YesOrNo();

if(AddFlag) {

DelFlag = false;

for(i=0; i < \*pConLen; ++i) {

if(!(\*pCounsellorArray[i]).id.flag) {

DelFlag = true;

free((\*pCounsellorArray)[i].pClass);

(\*pCounsellorArray)[i] = inputCounsellor;

break;

}

}

if(!DelFlag) {

\*pCounsellorArray = realloc(\*pCounsellorArray, (\*pConLen + 1) \* sizeof(counsellor));

assert(\*pCounsellorArray && "Cann't realloc the memory of Counsellor array");

(\*pCounsellorArray)[\*pConLen] = inputCounsellor;

\*pConLen += 1;

}

else {

free(inputCounsellor.pClass);

}

}

}

// 修改辅导员信息

// counsellor \*const pcounsellor 参数指修改当前辅导员信息

void ChangeCounsellorInfo(counsellor \*const pCounsellor)

{

assert(pCounsellor && "the pointer is NULL");

counsellor inputCounsellor;

bool changeFlag;

inputCounsellor = \*pCounsellor;

DispCounsellorInfo(pCounsellor);

PrintSplitLine();

printf("Do you change the name ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the name: ");

scanf("%s", inputCounsellor.name);

}

printf("Do you change the id ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the id: ");

scanf("%s", inputCounsellor.id.id);

}

printf("Do you change the address ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the address: ");

scanf("%s", inputCounsellor.address);

}

puts("Do you really change for these ? y/n");

changeFlag = YesOrNo();

if(changeFlag) {

\*pCounsellor = inputCounsellor;

}

}

// 通过 ID 查找辅导员信息

// const counsellor \*const pCounsellorArray 参数指辅导员数组指针

// const size\_t len 参数指 辅导员数组的长度

// const ID \*const id 参数指 需要找到的辅导员的ID指针

// size\_t \*const locateIndex 参数 指 找到辅导员在数组中的下标

// 函数返回值 bool 值 指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex 值没有意义

bool LocateCounsellorInfoByID(const counsellor \*const pCounsellorArray, const size\_t len, const ID \*const pID, size\_t \*const pLocateIndex)

{

assert(pCounsellorArray && "the pointer is NULL");

assert(pID && "the pointer is NULL");

size\_t i;

for(i=0; i < len; ++i) {

if(pCounsellorArray[i].id.flag && !strcmp(pID->id, pCounsellorArray[i].id.id)) {

\*pLocateIndex = i;

return true;

}

}

return false;

}

// 通过 姓名 查找辅导员信息

// const counsellor \*const pCounsellorArray 参数指辅导员数组指针

// const size\_t len 参数指 辅导员数组的长度

// const char \*const name 参数指 需要找到的辅导员的姓名

// size\_t \*const pLocateIndex 参数 指 找到辅导员在数组中的下标

// 函数返回值 bool 值 指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex 值没有意义

bool LocateCounsellorInfoByName(const counsellor \*const pCounsellorArray, const size\_t len, const char \*const name, size\_t \*const pLocateIndex)

{

assert(pCounsellorArray && "the pointer is NULL");

size\_t i;

for(i=0; i < len; ++i) {

if(pCounsellorArray[i].id.flag && !strcmp(name, pCounsellorArray[i].name)) {

\*pLocateIndex = i;

return true;

}

}

return false;

}

// 查找辅导员信息

// const counsellor \*const pCounsellorArray 参数指辅导员数组指针

// const size\_t len 参数指 辅导员数组的长度

// size\_t \*const locateIndex 参数 指 找到辅导员在数组中的下标

// 函数返回值 bool 值 指是否在辅导员数组中找到该辅导员信息，若没有找到，locateIndex 值没有意义

bool LocateCounsellorInfo(const counsellor \*const pCounsellorArray, const size\_t len, size\_t \*const pLocateIndex)

{

assert(pCounsellorArray && "the pointer is NULL");

assert(pLocateIndex && "the pointer is NULL");

size\_t chooseMode;

char inputBuffer[STRINGBUFFERLEN];

char inputFlag;

ID id;

PrintSplitLine();

puts("Choose the mode how to locating information of students");

puts("[1]. Student's name.");

puts("[2]. Student's ID.");

puts("[3]. Cancel.");

chooseMode = 0;

while(chooseMode != 1 && chooseMode != 2 && chooseMode != 3) {

inputFlag = scanf("%u", &chooseMode);

if(inputFlag != 1) {

getchar();

}

}

switch(chooseMode) {

case 1:

PrintSplitLine();

puts("Please input the student's name.");

scanf("%s", inputBuffer);

return LocateCounsellorInfoByName(pCounsellorArray, len, inputBuffer, pLocateIndex);

case 2:

PrintSplitLine();

puts("Please input the student's ID.");

scanf("%s", inputBuffer);

strcpy(id.id, inputBuffer);

return LocateCounsellorInfoByID(pCounsellorArray, len, &id, pLocateIndex);

case 3:

return false;

}

return false;

}

// 显示所有辅导员信息

// const counsellor \*pcounsellorArray 参数指需要显示的辅导员数组指针

// const size\_t conLen 参数指需要显示的辅导员数组的长度

void DispAllCounsellorInfo(const counsellor \*const pCounsellorArray, const size\_t conLen)

{

assert(pCounsellorArray &&

"the pointer is NULL in DispAllCounsellorInfo function, please check it out!");

size\_t i;

for(i=0; i < conLen; ++i) {

DispCounsellorInfo(pCounsellorArray + i);

}

}

// 显示辅导员信息

// const counsellor \*pcounsellor 参数指需要显示的辅导员指针

void DispCounsellorInfo(const counsellor \*const pCounsellor)

{

assert(pCounsellor &&

"the pointer is NULL in DispCounsellorInfo function, please check it out!");

size\_t i;

PrintSplitLine();

printf("ID: %s\n", pCounsellor->id.id);

printf("Name: %s\n", pCounsellor->name);

printf("Address: %s\n", pCounsellor->address);

puts("The manage class of he/she follow:");

for(i=0; i < pCounsellor->manageClassNum; ++i) {

if(pCounsellor->pClass[i].flag) {

printf("%s\n", pCounsellor->pClass[i].name);

}

}

PrintSplitLine();

putchar('\n');

}

// 删除 辅导员数组信息

// counsellor \*\*pCounsellorArray 参数指 指向辅导员数组指针的指针

// size\_t \*pLen 参数指 辅导员数组长度的指针

void FreeCounsellorArray(counsellor \*\*pCounsellorArray, size\_t \*pLen)

{

assert(pCounsellorArray && "the pointer is NULL");

assert(pLen && "The pointer is NULL");

int i;

if(\*pLen) {

for(i=0; i < \*pLen; ++i) {

if((\*pCounsellorArray)[i].manageClassNum) {

free((\*pCounsellorArray)[i].pClass);

}

}

free(\*pCounsellorArray);

}

\*pLen = 0;

\*pCounsellorArray = NULL;

}

// 初始化文件系统，读取配置文件

// const fileStruct \*pFileStruct 参数指 文件系统所必有信息的结构体

// 函数返回值 bool 若为真，则初始化成功，否则初始化失败

bool InitFileSystem(fileStruct \*const pFileStruct)

{

assert(pFileStruct && "pFileStruct is null pointer, Please check it out in InitFileSystem !");

assert(pFileStruct->profile && "profile is null");

FILE \*pProfile, \*pAdminInfoFile, \*pStudentInfoFile, \*pCounsellorInfoFile;

pProfile = fopen(pFileStruct->profile, "r");

if(!pProfile) {

return false;

}

fscanf(pProfile, "%s", pFileStruct->adminInfoFile);

fscanf(pProfile, "%s", pFileStruct->studentInfoFile);

fscanf(pProfile, "%s", pFileStruct->counsellorInfoFile);

pFileStruct->adminInfoFile[STRINGBUFFERLEN-1] = '\0';

pFileStruct->studentInfoFile[STRINGBUFFERLEN-1] = '\0';

pFileStruct->counsellorInfoFile[STRINGBUFFERLEN-1] = '\0';

pAdminInfoFile = fopen(pFileStruct->adminInfoFile, "r");

pStudentInfoFile = fopen(pFileStruct->studentInfoFile, "r");

pCounsellorInfoFile = fopen(pFileStruct->counsellorInfoFile, "r");

if(!(pAdminInfoFile && pStudentInfoFile && pCounsellorInfoFile)) {

return false;

}

assert(pAdminInfoFile && "Cann't open the file of user-manage, please check it out!");

assert(pStudentInfoFile && "Cann't open the file of student information, please check it out!");

assert(pCounsellorInfoFile && "Cann't open the file of counsellor information, please check it out!");

fclose(pProfile);

fclose(pAdminInfoFile);

fclose(pStudentInfoFile);

fclose(pCounsellorInfoFile);

return true;

}

// 读取学生信息

// const fileStruct \*const pFIleStruct 参数指 文件系统所必有信息的结构体

// student \*\*pStudentArray 参数指 学生数组指针的指针

// size\_t \*pLen 参数为 学生数组的长度指针

// 函数返回值 bool 若为真，则读取 学生信息 成功， 否则读取失败

bool ReadStudentFile(const fileStruct \*const pFileStruct, student \*\*pStudentArray, size\_t \*pLen)

{

FILE \*fp;

\*pLen = 0;

int flag = 0;

size\_t i, j;

fp = fopen(pFileStruct->studentInfoFile, "r");

assert(fp && "Cann't open the information of student file, please check it out!\n");

if(!fp) {

return false;

}

flag = fscanf(fp, "%u", pLen);

if(flag != 1) {

\*pLen = 0;

return false;

}

\*pStudentArray = (student \*)malloc(sizeof(student) \* \*pLen);

assert(pStudentArray && "Cann't mallocation the memory of student array!\n");

if(!\*pStudentArray) {

\*pLen = 0;

return false;

}

for(i=0; i < \*pLen; ++ i) {

(\*pStudentArray)[i].id.flag = true;

fscanf(fp, "%s %s", (\*pStudentArray)[i].id.id, (\*pStudentArray)[i].id.password);

fscanf(fp, "%s", (\*pStudentArray)[i].name);

fscanf(fp, "%u %s", &(\*pStudentArray)[i].sex, (\*pStudentArray)[i].nativePlace);

fscanf(fp, "%s %s", (\*pStudentArray)[i].birthday, (\*pStudentArray)[i].major);

fscanf(fp, "%s %s", (\*pStudentArray)[i].YearOfEnrollment, (\*pStudentArray)[i].class);

fscanf(fp, "%u", &(\*pStudentArray)[i].status);

fscanf(fp, "%s %s", (\*pStudentArray)[i].address, (\*pStudentArray)[i].phoneNumber);

fscanf(fp, "%d", &(\*pStudentArray)[i].subjectNum);

(\*pStudentArray)[i].pSubject = (subject \*)malloc(sizeof(subject) \* (\*pStudentArray)[i].subjectNum);

assert(!((\*pStudentArray)[i].pSubject == NULL && (\*pStudentArray)[i].subjectNum != 0) && "Cann't mallocating the memory of student subject!\n");

if((\*pStudentArray)[i].pSubject == NULL && (\*pStudentArray)[i].subjectNum != 0) {

return false;

}

for(j=0; j < (\*pStudentArray)[i].subjectNum; ++j) {

fscanf(fp, "%s", (\*pStudentArray)[i].pSubject[j].name);

fscanf(fp, "%f", &(\*pStudentArray)[i].pSubject[j].mark);

(\*pStudentArray)[i].pSubject[j].flag = true;

}

}

fclose(fp);

return true;

}

// 读取辅导员信息

// const fileStruct \*const pFileStruct 参数指 文件系统所必有信息的结构体

// conseller \*\*pCounsellorArray 参数指 辅导员数组指针的指针

// size\_t \*pLen 参数指 辅导员数组的长度指针

// 函数返回值 bool 若为真，则读取 辅导员信息 成功， 否则读取失败

bool ReadCounsellorFile(const fileStruct \*const pFileStruct, counsellor \*\*pCounsellorArray, size\_t \*pLen)

{

FILE \*fp;

size\_t i, j;

char flag;

assert(pFileStruct && "The pointer is NULL!");

assert(pCounsellorArray && "The pointer is NULL!");

assert(pLen && "The pointer is NULL!");

fp = fopen(pFileStruct->counsellorInfoFile, "r");

assert(fp && "Cann't open the counsellorInfoFile, please check it out!");

if(!fp) {

return false;

}

flag = fscanf(fp, "%d", pLen);

if(flag != 1) {

return false;

}

\*pCounsellorArray = (counsellor \*)malloc(\*pLen \* sizeof(counsellor));

assert(\*pCounsellorArray != NULL &&

"Cann't mallocating the memory in ReadCounsellorFile function, please check it out!");

if(\*pCounsellorArray == NULL) {

return false;

}

for(i=0; i < \*pLen; ++i) {

fscanf(fp, "%s %s", (\*pCounsellorArray)[i].id.id, (\*pCounsellorArray)[i].id.password);

fscanf(fp, "%s %s", (\*pCounsellorArray)[i].name, (\*pCounsellorArray)[i].address);

fscanf(fp, "%u", &(\*pCounsellorArray)[i].manageClassNum);

(\*pCounsellorArray)[i].id.flag = true;

(\*pCounsellorArray)[i].pClass = (class \*)malloc((\*pCounsellorArray)[i].manageClassNum \* sizeof(class));

assert(!((\*pCounsellorArray)[i].pClass == NULL && (\*pCounsellorArray)[i].manageClassNum != 0) &&

"Cann't mallocating the memory in ReadCounsellorFile function, please check it out!");

for(j=0; j < (\*pCounsellorArray)[i].manageClassNum; ++j) {

fscanf(fp, "%s", (\*pCounsellorArray)[i].pClass[j].name);

(\*pCounsellorArray)[i].pClass[j].flag = true;

}

}

fclose(fp);

return true;

}

// 读取管理员信息

// const fileStruct \*const pFileStruct 参数指 文件系统所必有信息的结构体

// admin \*const pAdmin 参数指 管理员

// 函数返回值 bool 若为真，则读取 管理员信息 成功， 否则读取失败

bool ReadAdminFile(const fileStruct \*const pFileStruct, admin \*const pAdmin)

{

FILE \*fp;

char flag;

assert(pFileStruct && "The pointer is NULL!");

assert(pAdmin && "The pointer is NULL!");

fp = fopen(pFileStruct->adminInfoFile, "r");

assert(fp && "Cann't open the adminInfoFile, please check it out!");

if(!fp) {

return false;

}

flag = fscanf(fp, "%s %s", pAdmin->id.id, pAdmin->id.password);

if(flag != 2) {

return false;

}

fclose(fp);

return true;

}

// 保存 学生信息

// const fileStruct \*const pFileStruct 参数指 文件系统所必有信息的结构体

// const student \*const pStudentArray 参数指 学生数组指针

// const size\_t len 指 学生数组的长度

// 函数返回值 bool 若为真，则保存学生信息成功，否则保存失败

bool SaveStudentFile(const fileStruct \*const pFileStruct, const student \*const pStudentArray, const size\_t len)

{

assert(pFileStruct && "the pointer is NULL in SaveStudentFile function, please check it out!");

assert(pStudentArray && "the pointer is NULL in SaveStudentFIle function, please check it out!");

FILE \*fp;

size\_t i, j, inFactLenOfStudentArray, inFactLenOfSubjectArray;

fp = fopen(pFileStruct->studentInfoFile, "w");

assert(fp && "Cann't open the studentInfoFile in SaveStudentFile function, please check it out!");

inFactLenOfStudentArray = 0;

for(i=0; i < len; ++i) {

if(pStudentArray[i].id.flag) {

++ inFactLenOfStudentArray;

}

}

fprintf(fp, "%u\n\n", inFactLenOfStudentArray);

for(i=0; i < len; ++i) {

if(pStudentArray[i].id.flag) {

fprintf(fp, "%s %s\n", pStudentArray[i].id.id, pStudentArray[i].id.password);

fprintf(fp, "%s %s\n", pStudentArray[i].name, pStudentArray[i].class);

fprintf(fp, "%s %s\n", pStudentArray[i].address, pStudentArray[i].phoneNumber);

inFactLenOfSubjectArray = 0;

for(j=0; j < pStudentArray[i].subjectNum; ++j) {

if(pStudentArray[i].pSubject[j].flag) {

++ inFactLenOfSubjectArray;

}

}

fprintf(fp, "%u\n", inFactLenOfSubjectArray);

for(j=0; j < pStudentArray[i].subjectNum; ++j) {

if(pStudentArray[i].pSubject[j].flag) {

fprintf(fp, "%s %.2f\n", pStudentArray[i].pSubject[j].name, pStudentArray[i].pSubject[j].mark);

}

}

fputc('\n', fp);

}

}

fclose(fp);

return true;

}

// 保存 辅导员信息

// const fileStruct \*const pFileStruct 参数指 文件系统所必有信息的结构体

// const conseller \*const pCounsellor 参数指 辅导员数组指针

// const size\_t len 指 辅导员数组的长度

// 函数返回值 bool 若为真，则保存辅导员信息成功，否则保存失败

bool SaveCounsellorFile(const fileStruct \*const pFileStruct, const counsellor \*const pCounsellorArray, const size\_t len)

{

assert(pFileStruct && "the pointer is NULL in SaveCounsellortFile function, please check it out!");

assert(pCounsellorArray && "the pointer is NULL in SaveCounsellor function, please check it out!");

FILE \*fp;

size\_t i, j, inFactLenOfCounsellorArray, inFactLenOfClassArray;

fp = fopen(pFileStruct->counsellorInfoFile, "w");

assert(fp && "Cann't open the counsellorInfoFile in SaveStudentFile function, please check it out!");

inFactLenOfCounsellorArray = 0;

for(i=0; i < len; ++i) {

if(pCounsellorArray[i].id.flag) {

++ inFactLenOfCounsellorArray;

}

}

fprintf(fp, "%u\n\n", inFactLenOfCounsellorArray);

for(i=0; i < len; ++i) {

if(pCounsellorArray[i].id.flag) {

fprintf(fp, "%s %s\n", pCounsellorArray[i].id.id, pCounsellorArray[i].id.password);

fprintf(fp, "%s %s\n", pCounsellorArray[i].name, pCounsellorArray[i].address);

inFactLenOfClassArray = 0;

for(j=0; j < pCounsellorArray[i].manageClassNum; ++j) {

if(pCounsellorArray[i].pClass[j].flag) {

++ inFactLenOfClassArray;

}

}

fprintf(fp, "%u\n", inFactLenOfClassArray);

for(j=0; j < pCounsellorArray[i].manageClassNum; ++j) {

if(pCounsellorArray[i].pClass[j].flag) {

fprintf(fp, "%s\n", pCounsellorArray[i].pClass[j].name);

}

}

fputc('\n', fp);

}

}

fclose(fp);

return true;

}

// 保存 管理员信息

// const fileStruct \*const pFileStruct 参数指 文件系统所必有信息的机构体

// const admin \*const pAdmin

// 函数返回值 bool 若为真，则保存管理员信息成功，否则保存失败

bool SaveAdminFile(const fileStruct \*const pFileStruct, const admin \*const pAdmin)

{

FILE \*fp;

assert(pFileStruct && "The pointer is NULL!");

assert(pAdmin && "The pointer is NULL!");

fp = fopen(pFileStruct->adminInfoFile, "w");

assert(fp && "Cann't open the adminInfoFile, please check it out!");

if(!fp) {

return false;

}

fprintf(fp, "%s\n%s\n", pAdmin->id.id, pAdmin->id.password);

fclose(fp);

return true;

}

bool InitSMS(SMS \*pSMS)

{

if(!InitFileSystem(&pSMS->FileStruct)) {

assert(!"InitFileSystem function is NULL, profile file or others is wrong!");

return false;

}

if(!ReadAdminFile(&pSMS->FileStruct, &pSMS->Admin)) {

assert(!"Cann't read ADMIN file, please check it out!");

return false;

}

if(!ReadCounsellorFile(&pSMS->FileStruct, &pSMS->pCounsellor, &pSMS->counsellorLen)) {

assert(!"Cann't read COUNSELLOR file, please check it out!");

return false;

}

if(!ReadStudentFile(&pSMS->FileStruct, &pSMS->pStudent, &pSMS->studentLen)) {

assert(!"Cann't read STUDENT file, please check it out!");

return false;

}

return true;

}

void RunSMS(SMS \*pSMS)

{

char ch = '0';

while(ch != '4') {

PrintSplitLine();

puts("There are 3 mode you have to choose!");

puts("[1] Student Mode");

puts("[2] Counsellor Mode");

puts("[3] Admin Mode");

puts("[4] Exit");

PrintSplitLine();

ch = '0';

while(ch < '1' || ch > '4') {

ch = getchar();

}

switch(ch) {

case '1': {

RunSMSByStudent(pSMS);

break;

}

case '2': {

RunSMSByCounsellor(pSMS);

break;

}

case '3': {

RunSMSByAdmin(pSMS);

break;

}

case '4': {

break;

}

default:

break;

}

}

}

void SaveSMS(SMS \*pSMS)

{

while(!SaveAdminFile(&pSMS->FileStruct, &pSMS->Admin)) {

assert("Cann't save ADMIN file, please check it out!");

puts("Cann't save ADMIN file, please check it out!");

getchar();

}

while(!SaveCounsellorFile(&pSMS->FileStruct, pSMS->pCounsellor, pSMS->counsellorLen)) {

assert("Cann't save COUNSELLOR file, please check it out!");

puts("Cann't save COUNSELLOR file, please check it out!");

getchar();

}

while(!SaveStudentFile(&pSMS->FileStruct, pSMS->pStudent, pSMS->studentLen)) {

assert("Cann't save STUDENT file, please check it out!");

puts("Cann't save STUDENT file, please check it out!");

getchar();

}

FreeCounsellorArray(&pSMS->pCounsellor, &pSMS->counsellorLen);

FreeStudentArray(&pSMS->pStudent, &pSMS->studentLen);

}

void RunSMSByAdmin(SMS \*pSMS)

{

bool loginFlag;

char inputFlag;

size\_t choice;

loginFlag = LoginByAdmin(&pSMS->Admin);

while(loginFlag) {

//system("cls");

PrintSplitLine();

puts("\t\t--- Admin Mode ---");

puts("[1]. Student information manage.");

puts("[2]. Student mark manage.");

puts("[3]. Student subject manage.");

puts("[4]. Counsellor information manage.");

puts("[5]. Change the password.");

puts("[6]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 6) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

StudentInfoManageModule(pSMS);

break;

case 2:

StudentMarkManageModule(pSMS);

break;

case 3:

StudentSubjectModule(pSMS);

break;

case 4:

CounsellorInfoManageModule(pSMS);

break;

case 5:

ChangePassword(&pSMS->Admin.id);

break;

case 6:

loginFlag = false;

break;

default:

break;

}

}

}

void RunSMSByCounsellor(SMS \*pSMS)

{

bool loginFlag;

char inputFlag;

size\_t choice;

size\_t index;

loginFlag = LoginByCounsellor(pSMS->pCounsellor, pSMS->counsellorLen, &index);

while(loginFlag) {

//system("cls");

PrintSplitLine();

puts("\t\t--- Counsellor Mode ---");

puts("[1]. Student mark manage.");

puts("[2]. Show the information.");

puts("[3]. Change the information.");

puts("[4]. Change password.");

puts("[5]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 5) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

StudentMarkManageModule(pSMS);

break;

case 2:

DispCounsellorInfo(pSMS->pCounsellor + index);

getchar();

getchar();

break;

case 3:

ChangeCounsellorInfo(pSMS->pCounsellor + index);

break;

case 4:

ChangePassword(&pSMS->pCounsellor[index].id);

break;

case 5:

loginFlag = false;

break;

default:

break;

}

}

}

void RunSMSByStudent(SMS \*pSMS)

{

bool loginFlag;

char inputFlag;

size\_t choice;

size\_t index;

loginFlag = LoginByCounsellor(pSMS->pCounsellor, pSMS->counsellorLen, &index);

while(loginFlag) {

//system("cls");

PrintSplitLine();

puts("\t\t--- Student Mode ---");

puts("[1]. Show the information.");

puts("[2]. Change the information.");

puts("[3]. Change the password.");

puts("[4]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 4) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

DispStudentInfo(pSMS->pStudent + index);

break;

case 2:

ChangeStudentInfo(pSMS->pStudent + index);

break;

case 3:

ChangePassword(&pSMS->pStudent[index].id);

break;

case 4:

loginFlag = false;

break;

default:

break;

}

}

}

void StudentInfoManageModule(SMS \*pSMS)

{

char inputFlag;

bool locateFlag;

size\_t choice;

size\_t index;

choice = 0;

while(choice != 5) {

//system("cls");

PrintSplitLine();

puts("\t\t-- Student Information Manage ---");

puts("[1]. Delete student information.");

puts("[2]. Locate student information.");

puts("[3]. Change student information.");

puts("[4]. Add student information.");

puts("[5]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 5) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

DeleteStudentInfo(pSMS->pStudent, pSMS->studentLen);

break;

case 2:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

DispStudentInfo(pSMS->pStudent + index);

getchar();

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 3:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

ChangeStudentInfo(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 4:

AddStudentInfo(&pSMS->pStudent, &pSMS->studentLen);

break;

case 5:

break;

default:

break;

}

}

}

void StudentMarkManageModule(SMS \*pSMS)

{

char inputFlag;

bool locateFlag;

size\_t index;

size\_t choice;

choice = 0;

while(choice != 3) {

PrintSplitLine();

puts("\t\t--- Student Mark Manage ---");

puts("[1]. Add mark to student.");

puts("[2]. Change mark to student.");

puts("[3]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 3) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

AddMark(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 2:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

ChangeMark(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 3:

break;

default:

break;

}

}

}

void StudentSubjectModule(SMS \*pSMS)

{

char inputFlag;

bool locateFlag;

size\_t index;

size\_t choice;

choice = 0;

while(choice != 7) {

PrintSplitLine();

puts("\t\t--- Student Mark Manage ---");

puts("[1]. Add same subject to all students.");

puts("[2]. Delete same subject to all students.");

puts("[3]. Change same subject to all students.");

puts("[4]. Delete subject to single student.");

puts("[5]. Add subject to single student.");

puts("[6]. Change subject to single student.");

puts("[7]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 7) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

AddAllStudentSubjectInfo(pSMS->pStudent, pSMS->studentLen);

break;

case 2:

DeleteAllStudentSubjectInfo(pSMS->pStudent, pSMS->studentLen);

break;

case 3:

DeleteAllStudentSubjectInfo(pSMS->pStudent, pSMS->studentLen);

break;

case 4:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

DeleteSingleStudentSubjectInfo(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 5:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

AddSingleStudentSubjectInfo(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 6:

locateFlag = LocateStudentInfo(pSMS->pStudent, pSMS->studentLen, &index);

if(locateFlag) {

puts("Show the information of student that you want to locate.");

ChangeSingleStudentSubjectInfo(pSMS->pStudent + index);

}

else {

puts("Cann't find the student, please check it out!");

}

break;

case 7:

break;

default:

break;

}

}

}

void CounsellorInfoManageModule(SMS \*pSMS)

{

char inputFlag;

bool locateFlag;

size\_t choice;

size\_t index;

choice = 0;

while(choice != 5) {

//system("cls");

PrintSplitLine();

puts("\t\t-- Counsellor Information Manage ---");

puts("[1]. Delete Counsellor information.");

puts("[2]. Locate Counsellor information.");

puts("[3]. Change Counsellor information.");

puts("[4]. Add Counsellor information.");

puts("[5]. Cancel.");

PrintSplitLine();

choice = 0;

while(choice == 0 || choice > 5) {

inputFlag = scanf("%u", &choice);

if(!inputFlag) {

getchar();

}

}

switch(choice) {

case 1:

DeleteCounsellorInfo(pSMS->pCounsellor, pSMS->counsellorLen);

break;

case 2:

locateFlag = LocateCounsellorInfo(pSMS->pCounsellor, pSMS->counsellorLen, &index);

if(locateFlag) {

puts("Show the information of Counsellor that you want to locate.");

DispCounsellorInfo(pSMS->pCounsellor + index);

getchar();

}

else {

puts("Cann't find the Counsellor, please check it out!");

}

break;

case 3:

locateFlag = LocateCounsellorInfo(pSMS->pCounsellor, pSMS->counsellorLen, &index);

if(locateFlag) {

puts("Show the information of Counsellor that you want to locate.");

ChangeCounsellorInfo(pSMS->pCounsellor + index);

}

else {

puts("Cann't find the Counsellor, please check it out!");

}

break;

case 4:

AddCounsellorInfo(&pSMS->pCounsellor, &pSMS->counsellorLen);

break;

case 5:

break;

default:

break;

}

}

}

// 显示学生信息

// const student \*pStudent 参数指需要显示的学生指针

void DispStudentInfo(const student \*const pStudent)

{

assert(pStudent && "The pointer is NULL in DispStudentInfo function, please check it out!");

size\_t i;

PrintSplitLine();

printf("ID: %s\n", pStudent->id.id);

printf("Name: %s\n", pStudent->name);

printf("Sex: %s\n", pStudent->sex == Male ? "male" : "female");

printf("Native place: %s\n", pStudent->nativePlace);

printf("Birthday: %s\n", pStudent->birthday);

printf("Major: %s\n", pStudent->major);

printf("Year of enrollment: %s\n", pStudent->YearOfEnrollment);

printf("Class: %s\n", pStudent->class);

printf("Status: ");

switch(pStudent->status) {

case Be\_Studying\_At\_School:

puts("Be studying at school");

break;

case Suspension\_of\_Schooling:

puts("Suspension of schooling");

break;

case Leavel\_School:

puts("Leavel school");

break;

case Serve\_in\_Army:

puts("Serve in army");

break;

default:

puts("none\n");

break;

}

printf("Address: %s\n", pStudent->address);

printf("Phone-Number: %s\n", pStudent->phoneNumber);

puts("The subject(s) of he/she follow:");

for(i=0; i < pStudent->subjectNum; ++i) {

if(pStudent->pSubject[i].flag) {

printf("%s\t%.2f\n", pStudent->pSubject[i].name, pStudent->pSubject[i].mark);

}

}

PrintSplitLine();

}

// 显示所有学生信息

// const student \*pStudentArray 参数指需要显示的学生数组指针

// const size\_t conLen 参数指需要显示的学生数组的长度

void DispAllStudentInfo(const student \*const pStudentArray, const size\_t conLen)

{

size\_t i;

for(i=0; i < conLen; ++i) {

PrintSplitLine();

//printf("The number of student array is: %u\n", i+1);

DispStudentInfo(pStudentArray + i);

putchar('\n');

}

}

// 删除 学生数组信息

// student \*\*pStudentArray 参数指 指向学生数组指针的指针

// size\_t \*pLen 参数指 学生数组长度的指针

void FreeStudentArray(student \*\*pStudentArray, size\_t \*pLen)

{

assert(pStudentArray && "the pointer is NULL");

assert(pLen && "The pointer is NULL");

int i;

if(\*pLen) {

for(i=0; i < \*pLen; ++i) {

if((\*pStudentArray)[i].subjectNum) {

free((\*pStudentArray)[i].pSubject);

}

}

free(\*pStudentArray);

}

\*pLen = 0;

\*pStudentArray = NULL;

}

// 删除学生信息， 通过 标记 flag 惰性删除

// student \*const pStudentArray 参数指学生数组指针

// const size\_t Len 参数指 学生数组的长度

void DeleteStudentInfo(student \*const pStudentArray, const size\_t len)

{

assert(pStudentArray && "the pointer is NULL!");

char findID[STRINGBUFFERLEN];

bool deleteFlag, findFlag;

size\_t i;

PrintSplitLine();

printf("Please input student ID: ");

scanf("%s", findID);

printf("Do you really delete ID: %s ?(y/n)\n", findID);

deleteFlag = YesOrNo();

if(deleteFlag) {

findFlag = false;

for(i=0; i < len; ++ i) {

if(pStudentArray[i].id.flag && !strcmp(pStudentArray[i].id.id, findID)) {

findFlag = true;

pStudentArray[i].id.flag = false;

}

}

if(findFlag) {

puts("Delete success!");

}

else {

puts("Cann't find this ID, please check it out!");

}

}

PrintSplitLine();

}

// 添加学生信息

// student \*\*pStudentArray 参数指 学生数组指针的指针

// size\_t \*const pConLen 参数指 学生是数组的长度

void AddStudentInfo(student \*\*pStudentArray, size\_t \*const pConLen)

{

assert(pStudentArray && "The pointer is NULL");

assert(pConLen && "The pointer is NULL");

size\_t i;

bool inputFlag;

student inputStudent;

bool addStudentFlag, findDeletedStudentFlag;

PrintSplitLine();

printf("input the ID: ");

scanf("%s", inputStudent.id.id);

printf("input the password: ");

scanf("%s", inputStudent.id.password);

inputStudent.id.flag = true;

printf("input the name: ");

scanf("%s", inputStudent.name);

printf("choose male or female to sex. Y is male and N is female. ");

if(YesOrNo()) {

inputStudent.sex = Male;

}

else {

inputStudent.sex = Female;

}

printf("input the native place: ");

scanf("%s", inputStudent.nativePlace);

printf("input the birthday: ");

scanf("%s", inputStudent.birthday);

printf("input the major: ");

scanf("%s", inputStudent.major);

printf("input the year of enrollment: ");

scanf("%s", inputStudent.YearOfEnrollment);

printf("input what class hs is belong to: ");

scanf("%s", inputStudent.class);

puts("choose status to the student:");

puts("[1]. Be studying at school.");

puts("[2]. Suspension of schooling.");

puts("[3]. Leavel school.");

puts("[4]. Serve in army.");

i=0;

while(i == 0 || i > 4) {

inputFlag = scanf("%u", &i);

if(inputFlag != 1) {

getchar();

}

}

switch(i) {

case 1:

inputStudent.status = Be\_Studying\_At\_School;

break;

case 2:

inputStudent.status = Suspension\_of\_Schooling;

break;

case 3:

inputStudent.status = Leavel\_School;

break;

case 4:

inputStudent.status = Serve\_in\_Army;

default:

break;

}

printf("input address: ");

scanf("%s", inputStudent.address);

printf("input phone-number: ");

scanf("%s", inputStudent.phoneNumber);

printf("input the number of subject: ");

scanf("%u", &inputStudent.subjectNum);

inputStudent.pSubject = (subject \*)malloc(inputStudent.subjectNum \* sizeof(subject));

assert(!(inputStudent.subjectNum != 0 && inputStudent.pSubject == NULL)

&& "Cann't alloc the class memory in AddStudentInfo function!");

for(i=0; i < inputStudent.subjectNum; ++i) {

printf("input the subject: ");

scanf("%s", inputStudent.pSubject[i].name);

printf("input mark of %s: ", inputStudent.pSubject[i].name);

scanf("%f", &inputStudent.pSubject[i].mark);

inputStudent.pSubject[i].flag = true;

}

puts("Do you really add this student ? y/n");

addStudentFlag = YesOrNo();

if(addStudentFlag) {

findDeletedStudentFlag = false;

for(i=0; i < \*pConLen; ++i) {

if(!(\*pStudentArray[i]).id.flag) {

findDeletedStudentFlag = true;

free((\*pStudentArray)[i].pSubject);

(\*pStudentArray)[i] = inputStudent;

break;

}

}

if(!findDeletedStudentFlag) {

\*pStudentArray = realloc(\*pStudentArray, (\*pConLen + 1) \* sizeof(student));

assert(\*pStudentArray && "Cann't realloc the memory of student array");

(\*pStudentArray)[\*pConLen] = inputStudent;

\*pConLen += 1;

}

}

else {

free(inputStudent.pSubject);

}

}

// 修改学生信息

// student \*const pStudent 参数指修改当前学生信息

void ChangeStudentInfo(student \*const pStudent)

{

assert(pStudent && "the pointer is NULL");

student inputStudent;

bool changeFlag;

inputStudent = \*pStudent;

DispStudentInfo(pStudent);

PrintSplitLine();

printf("Do you change the name ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the name: ");

scanf("%s", inputStudent.name);

}

printf("Do you change the id ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the id: ");

scanf("%s", inputStudent.id.id);

}

printf("Do you change the address ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the address: ");

scanf("%s", inputStudent.address);

}

printf("Do you change the class? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the class: ");

scanf("%s", inputStudent.class);

}

printf("Do you change the phone-number ? y/n ");

changeFlag = YesOrNo();

if(changeFlag) {

printf("input the phone-number: ");

scanf("%s", inputStudent.phoneNumber);

}

puts("Do you really change for these ? y/n");

changeFlag = YesOrNo();

if(changeFlag) {

\*pStudent = inputStudent;

}

}

// 通过 ID 查找学生信息

// const student \*const pStudentArray 参数指学生数组指针

// const size\_t len 参数指 学生数组的长度

// const ID \*const id 参数指 需要找到的学生的ID指针

// size\_t \*const locateIndex 参数 指 找到学生在数组中的下标

// 函数返回值 bool 值 指是否在学生数组中找到该学生信息，若没有找到，locateIndex 值没有意义

bool LocateStudentInfoByID(const student \*const pStudentArray, const size\_t len, const ID \*const pID, size\_t \*const pLocateIndex)

{

assert(pStudentArray && "the pointer is NULL");

assert(pID && "the pointer is NULL");

size\_t i;

for(i=0; i < len; ++i) {

if(pStudentArray[i].id.flag && !strcmp(pID->id, pStudentArray[i].id.id)) {

\*pLocateIndex = i;

return true;

}

}

return false;

}

// 通过 姓名 查找学生信息

// const student \*const pStudentArray 参数指学生数组指针

// const size\_t len 参数指 学生数组的长度

// const char \*const name 参数指 需要找到的学生的姓名

// size\_t \*const pLocateIndex 参数 指 找到学生在数组中的下标

// 函数返回值 bool 值 指是否在学生数组中找到该学生信息，若没有找到，locateIndex 值没有意义

bool LocateStudentInfoByName(const student \*const pStudentArray, const size\_t len, const char \*const name, size\_t \*const pLocateIndex)

{

assert(pStudentArray && "the pointer is NULL");

size\_t i;

for(i=0; i < len; ++i) {

if(pStudentArray[i].id.flag && !strcmp(name, pStudentArray[i].name)) {

\*pLocateIndex = i;

return true;

}

}

return false;

}

// 查找学生信息

// const student \*const pStudentArray 参数指学生数组指针

// const size\_t len 参数指 学生数组的长度

// size\_t \*const locateIndex 参数 指 找到学生在数组中的下标

// 函数返回值 bool 值 指是否在学生数组中找到该学生信息，若没有找到，locateIndex 值没有意义

bool LocateStudentInfo(const student \*const pStudentArray, const size\_t len, size\_t \*const pLocateIndex)

{

assert(pStudentArray && "the pointer is NULL");

assert(pLocateIndex && "the pointer is NULL");

size\_t chooseMode;

char inputBuffer[STRINGBUFFERLEN];

char inputFlag;

ID id;

PrintSplitLine();

puts("Choose the mode how to locating information of students");

puts("[1]. Student's name.");

puts("[2]. Student's ID.");

puts("[3]. Cancel.");

chooseMode = 0;

while(chooseMode != 1 && chooseMode != 2 && chooseMode != 3) {

inputFlag = scanf("%u", &chooseMode);

if(inputFlag != 1) {

getchar();

}

}

switch(chooseMode) {

case 1:

PrintSplitLine();

puts("Please input the student's name.");

scanf("%s", inputBuffer);

return LocateStudentInfoByName(pStudentArray, len, inputBuffer, pLocateIndex);

case 2:

PrintSplitLine();

puts("Please input the student's ID.");

scanf("%s", inputBuffer);

strcpy(id.id, inputBuffer);

return LocateStudentInfoByID(pStudentArray, len, &id, pLocateIndex);

case 3:

return false;

}

return false;

}

// 添加 单个学生的成绩

// const student \*const pStudent 参数指学生的指针

void AddMark(student \*const pStudent)

{

assert(pStudent && "The pointer is NULL!\n");

size\_t i;

bool flag;

PrintSplitLine();

PrintSplitLine();

//puts("Input error,pls input again!\n");

for(i=0; i < pStudent->subjectNum; ++i)

{

if(pStudent->pSubject[i].flag)

{

printf("%s: ", pStudent->pSubject[i].name);

puts("Please input the marks.");

flag = true;

while(flag)

{

if(scanf("%f", &pStudent->pSubject[i].mark) == 1)

{

flag = false;

}

else

{

getchar();

}

}

}

}

PrintSplitLine();

PrintSplitLine();

}

// 修改 单个学生的成绩

// const student \*const pStudent 参数指学生的指针

void ChangeMark(student \*const pStudent)

{

PrintSplitLine();

PrintSplitLine();

assert(pStudent && "The pointer is NULL!\n");

size\_t i;

bool Changeflag;

float mark;

for(i=0; i < pStudent->subjectNum; ++i)

{

//Changeflag = false;

if(pStudent->pSubject[i].flag)

{

printf("%s: ", pStudent -> pSubject[i].name);

printf("\n");

printf("%f ",pStudent -> pSubject[i].mark);

printf("\n");

puts("Do you want to skip this subject? y/n\n");

Changeflag = YesOrNo();

if(!Changeflag)

{

puts("Please input the subject you want to modify.\n");

printf("Input the %s's mark: ", pStudent->pSubject[i].name);

scanf("%f", &mark);

puts("Do you want to modify the mark? y/n\n");

Changeflag = YesOrNo();

if(Changeflag)

{

pStudent->pSubject[i].mark = mark;

}

}

}

}

puts("Modify succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 删除所有学生科目信息， 通过 标记 flag 惰性删除

// student pStudentArray[] 参数指学生数组指针

// const size\_t Len 参数指 学生数组的长度

void DeleteAllStudentSubjectInfo(student pStudentArray[], const size\_t len)

{

size\_t i,j,b;

char a[STRINGBUFFERLEN];

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject.\n");

scanf("%s", a);

puts("Are you sure delete the subject for all students? y/n\n");

b = YesOrNo();

if(b == true)

{

for(i=0; i < len; ++i)

{

for(j=0 ; j < pStudentArray[i].subjectNum; ++j)

{

if(pStudentArray[i].pSubject[j].flag && !strcmp(a , pStudentArray[i].pSubject[j].name))

{

pStudentArray[i].pSubject[j].flag = false;

}

}

}

}

else

getchar();

puts("Delete succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 添加所有学生科目信息，通过修改已标记删除的科目或者重新分配空间

// student pStudentArray[] 参数指 学生数组指针

// const size\_t len 参数指 学生是数组的长度

void AddAllStudentSubjectInfo(student pStudentArray[], const size\_t len)

{

size\_t i,j,b;

char \_subject\_[STRINGBUFFERLEN];

bool Addedflag;

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject.\n");

scanf("%s", \_subject\_);

puts("Are you sure add the subject for all students? y/n\n");

b = YesOrNo();

if(b == true)

{

//puts("Input the name of subject!\n");

//scanf("%s" , &a);

for(i = 0 ; i < len ; ++i)

{

Addedflag = false;

for(j = 0 ; j < pStudentArray[i].subjectNum; ++j)

{

if(!pStudentArray[i].pSubject[j].flag)

{

//strcmp()

Addedflag = true;

strcpy(pStudentArray[i].pSubject[j].name , \_subject\_);

pStudentArray[i].pSubject[j].mark = 0.0f;

pStudentArray[i].pSubject[j].flag = true;

break;

}

}

if(!Addedflag) {

pStudentArray[i].pSubject = realloc(pStudentArray[i].pSubject, (pStudentArray[i].subjectNum + 1) \* sizeof(subject));/\*malloc free copy\*/

assert(pStudentArray[i].pSubject && "the pointer is NULL!");

strcpy(pStudentArray[i].pSubject[pStudentArray[i].subjectNum].name, \_subject\_);

pStudentArray[i].pSubject[pStudentArray[i].subjectNum].flag = true;

pStudentArray[i].pSubject[pStudentArray[i].subjectNum].mark = 0.0f;

pStudentArray[i].subjectNum += 1;

}

}

}

puts("Add succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 修改所有学生科目信息，通过查找每个学生的科目信息，如若找到符合的科目信息，则修改。

// student pStudentArray[] 参数指 学生数组指针

// const size\_t len 参数指 学生是数组的长度

void ChangeAllStudentSubjectInfo(student pStudentArray[], const size\_t len)

{

size\_t i,j,b;

bool setZeroFlag;

char \_subject\_1[STRINGBUFFERLEN];

char \_subject\_2[STRINGBUFFERLEN];

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject!\n");

puts("The first is the subject you want to modify,\n");

puts("The second is the subject you want to change to.\n");

scanf("%s %s", \_subject\_1, \_subject\_2);

puts("Are you sure modify the subject for all students? y/n\n");

b = YesOrNo();

if(b == true)

{

puts("Are you sure modify the subject's mark for '0'? y/n\n");

setZeroFlag = YesOrNo();

for(i = 0; i < len; ++i)

{

for(j=0 ; j<pStudentArray[i].subjectNum ; ++j)

{

if(pStudentArray[i].id.flag && pStudentArray[i].pSubject[j].flag && !strcmp(pStudentArray[i].pSubject[j].name, \_subject\_1))

{

strcpy(pStudentArray[i].pSubject[j].name, \_subject\_2);

if(setZeroFlag)

{

pStudentArray[i].pSubject[j].mark = 0.0f;

}

}

}

}

}

puts("Modify succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 删除单个学生科目信息， 通过 标记 flag 惰性删除

// const Student \*const pStudent 参数指学生指针

void DeleteSingleStudentSubjectInfo(student \*const pStudent)

{

assert(pStudent && "Can't find the student!\n");

size\_t j,b;

char a[STRINGBUFFERLEN];

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject.\n");

scanf("%s" , a);

puts("Are you sure delete the subject for the student? y/n\n");

b = YesOrNo();

if(b == true)

{

for(j=0 ; j < pStudent->subjectNum; ++j)

{

if(pStudent->pSubject[j].flag && !strcmp(a , pStudent->pSubject[j].name))

{

pStudent->pSubject[j].flag = false;

}

}

}

puts("Delete succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 添加单个学生科目信息，通过修改已标记删除的科目或者重新分配空间

// const student \*const pStudent 参数指 学生指针

void AddSingleStudentSubjectInfo(student \*const pStudent)

{

assert(pStudent && "The pointer is NULL!\n");

size\_t j,b;

char \_subject\_[STRINGBUFFERLEN];

bool Addedflag;

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject.\n");

scanf("%s", \_subject\_);

puts("Are you sure add the subject for the student? y/n\n");

b = YesOrNo();

if(b == true)

{

//puts("Input the name of subject!\n");

//scanf("%s" , &a);

Addedflag = false;

for(j = 0 ; j < pStudent -> subjectNum; ++j)

{

if(!pStudent -> pSubject[j].flag)

{

//strcmp()

Addedflag = true;

strcpy(pStudent -> pSubject[j].name , \_subject\_);

pStudent -> pSubject[j].mark = 0.0f;

pStudent -> pSubject[j].flag = true;

break;

}

}

if(!Addedflag) {

pStudent -> pSubject = realloc(pStudent -> pSubject, (pStudent -> subjectNum + 1) \* sizeof(subject));/\*malloc free copy\*/

assert(pStudent -> pSubject && "the pointer is NULL!");

strcpy(pStudent -> pSubject[pStudent -> subjectNum].name, \_subject\_);

pStudent -> pSubject[pStudent -> subjectNum].flag = true;

pStudent -> pSubject[pStudent -> subjectNum].mark = 0.0f;

pStudent -> subjectNum += 1;

}

}

puts("Add succeed!\n");

PrintSplitLine();

PrintSplitLine();

}

// 修改单个学生科目信息，通过查找该学生的科目信息，如若找到符合的科目信息，则修改。

// student pStudent[] 参数指 学生指针

// 函数返回值 bool 若为真， 则修改成功，否则修改失败或没有找到符合的科目

bool ChangeSingleStudentSubjectInfo(student \*const pStudent)

{

assert(pStudent && "The pointer is NULL!\n");

size\_t j,b;

bool setZeroFlag;

bool changeFlag;

char \_subject\_1[STRINGBUFFERLEN];

char \_subject\_2[STRINGBUFFERLEN];

PrintSplitLine();

PrintSplitLine();

puts("Please input the name of subject!\n");

puts("The first is the subject you want to modify,\n");

puts("The second is the subject you want to change to.\n");

scanf("%s %s" , \_subject\_1 , \_subject\_2);

puts("Are you sure modify the subject for the students? y/n\n");

changeFlag = false;

b = YesOrNo();

if(b == true)

{

puts("Are you sure modify the subject's mark for '0'? y/n\n");

setZeroFlag = YesOrNo();

for(j=0 ; j < pStudent -> subjectNum ; ++j)

{

if(pStudent -> pSubject[j].flag && !strcmp(pStudent -> pSubject[j].name, \_subject\_1))

{

changeFlag = true;

strcpy(pStudent -> pSubject[j].name, \_subject\_2);

if(setZeroFlag)

{

pStudent -> pSubject[j].mark = 0.0f;

}

}

}

}

puts("Modify succeed!\n");

PrintSplitLine();

PrintSplitLine();

return changeFlag;

}

// 以学生身份登录认证

// const student pStudentArray[] 参数指 学生数组

// const size\_t len 参数指 学生数组的长度

// size\_t \*const pIndex 参数指认证成功，获得该学生在学生数组中的下标

// 函数返回值 bool 型，若返回值为真，则表示认证成功，否则失败，pIndex 值无意义

bool LoginByStudent(const student pStudentArray[], const size\_t len, size\_t \*const pIndex){

size\_t i,j;

ID inputID;

puts("You have only three times try to login or exit!");

for(i=0; i < 3; ++i) {

printf("You have %d times try to login!\n", 3-i);

printf("Input the ID: ");

scanf("%s", inputID.id);

printf("Input the password: ");

scanf("%s", inputID.password);

for(j=0; j < len; ++j) {

if(CheckID(&pStudentArray[j].id, &inputID)) {

\*pIndex = j;

return true;

}

}

}

return false;

}

// 以辅导员身份登录认证

// const counsellor pCounsellorArray[] 参数指 辅导员数组

// const size\_t len 参数指 辅导员数组的长度

// size\_t \*const pIndex 参数指认证成功，获得该辅导员在辅导员数组中的下标

// 函数返回值 bool 型， 若返回值为真，则表示认证成功，否则失败，pIndex 值无意义

bool LoginByCounsellor(const counsellor pCounsellorArray[], const size\_t len, size\_t \*const pIndex)

{

size\_t i,j;

ID inputID;

puts("You have only three times try to login or exit!");

for(i=0; i < 3; ++i) {

printf("You have %d times try to login!\n", 3-i);

printf("Input the ID: ");

scanf("%s", inputID.id);

printf("Input the password: ");

scanf("%s", inputID.password);

for(j=0; j < len; ++j) {

if(CheckID(&pCounsellorArray[j].id, &inputID)) {

\*pIndex = j;

return true;

}

}

}

return false;

}

// 以管理员身份登录认证

// const admin \*const pAdmin 参数指 管理员指针

// 函数返回值 bool 型， 若返回值为真， 则表示认证成功， 否则失败

bool LoginByAdmin(const admin \*const pAdmin)

{

size\_t i;

ID inputID;

puts("You have only three times try to login or exit!");

for(i=0; i < 3; ++i) {

printf("You have %d times try to login!\n", 3-i);

printf("Input the ID: ");

scanf("%s", inputID.id);

printf("Input the password: ");

scanf("%s", inputID.password);

if(CheckID(&pAdmin->id, &inputID)) {

return true;

}

}

return false;

}

bool CheckID(const ID \*const pIDA, const ID \*const pIDB)

{

assert(pIDA && "The pointer is NULL");

assert(pIDB && "The pointer is NULL");

return !strcmp(pIDA->id, pIDB->id) && !strcmp(pIDA->password, pIDB->password);

}

// 修改密码

// 参数 ID \*const pID 为待修改的账号指针

void ChangePassword(ID \*const pID)

{

assert(pID && "the pointer is NULL");

bool f;

ID inputID;

PrintSplitLine();

printf("Do you want to change the password? y/n");

f = YesOrNo();

if(f){

printf("Input the new password");

scanf("%s", inputID.password);

}

printf("Are you sure change the password? y/n");

if(f){

\*pID = inputID;

}

}

int main()

{

SMS \*pSMS;

char profilePath[] = "../../data/profile.txt";

pSMS = (SMS \*)malloc(sizeof(SMS));

strcpy(pSMS->FileStruct.profile, profilePath);

InitSMS(pSMS);

RunSMS(pSMS);

//freopen("1.txt", "rw+", stdin);

//UnitTest();

return 0;

}