

浙江工业大学

大类基础课程大型实验

2016/2017(2)



实验题目 校友录管理系统

学生姓名 杨振华

学生学号 201626811225

学生班级 计算机类 1612

任课教师 周德龙

提交日期 2017 年 6 月 10 日

计算机科学与技术学院

校友录管理系统 实验报告

一、 大型实验的内容

校友录管理系统（ADMS: Alumni Directory Management System）用于进行管理校友信息，其中校友信息包括姓名、性别、年龄、届级、系、班级、通讯地址、电话、QQ、email等。题目要求完成的主要的功能包括数据的录入、查询（按姓名、届级、系、班级）、修改、删除、排序（按姓名、届级），以及校友信息读写文件。要求使用学习过的 C/C++ 程序设计的知识完成图书管理系统的设计与实现。

除此之外，为了该系统更易用，还增加了多用户（附带权限系统）以及扫二维码打电话和发邮件的功能。

二、 运行环境

图书管理系统（ADMS）在 Visual Studio 2017 平台下开发，同时，代码也能在 GCC 7.1.0 下编译通过并正常运行。操作系统：Windows 10。

硬件环境：

处理器：Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz 1.70GHz

内存：4.00GB

系统类型：64 位操作系统

三、 实验课题分析

3.1 校友录管理系统的主要功能

校友录管理系统（ADMS）主要功能为用户管理、校友信息维护、展示及存储等，详细的系统功能结构为图 1 所示。

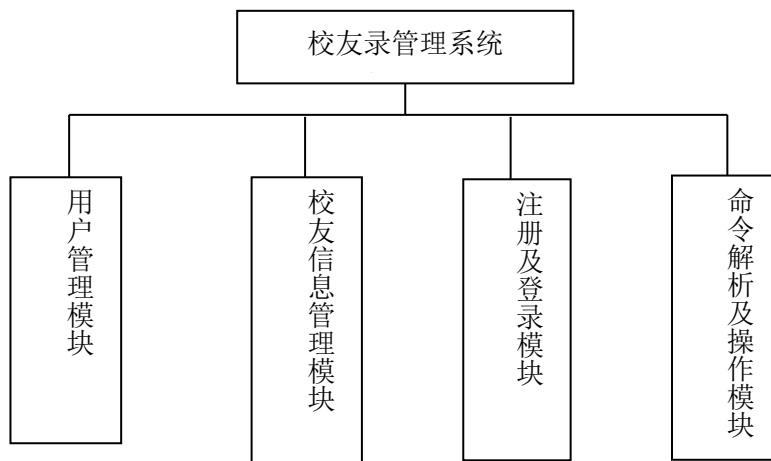


图 1 系统结构图

系统各模块的功能具体描述为：

（1）用户管理模块

主要功能接口包括：添加用户、删除用户、修改权限（普通用户或管理员）、修改密码、根据用户名查询密码、根据用户名查询权限、列出所有用户等。该模块具有自动保存功能，在进行修改用户数据的操作后会自动将数据保存到磁盘文件，防止数据丢失。

（2）校友信息管理模块

主要功能接口包括：添加校友信息、删除校友信息、根据多个键查询校友信息、列出所有校友信息、根据姓名对校友信息进行排序、根据毕业时间对校友信息进行排序等。与用户管理模块相同，该模块也具有自动保存功能，在进行修改校友信息的操作后数据会被自动保存到磁盘，防止数据丢失。

（3）注册及登录模块

由于本系统附带有用户管理功能，因此需要具备登录功能，登录模块将从用户管理模块获取用户信息，在确保用户名、密码正确的情况下自动判断用户类别（普通用户或超级管理员）从而构造具有正确权限的用户。

虽然之后的用户可由管理员添加，但第一个用户无法自动生成，而注册模块当检测到没有用户的时候会提示用户注册第一个管理员用户，确保系统功能逻辑正确。

（4）命令解析及操作模块

本模块主要用于解析用户输入的合法指令（当指令不正确时提示用户输入错误），并调用相关函数进行操作，最后返回并告知用户操作结果。

其中命令解析之后所调用的函数也是在本模块中实现的，是对用户管理模块和校友信息管理模块所提供接口的更深层次的包装和抽象并结合不同用户的权限提供了不同功能。

普通用户能够执行的功能有输出登录成功界面、输出帮助、列出所有校友信息、查询校友信息、输出电话号码二维码、输出电子邮箱二维码、按姓名对校友信息进行排序、按毕业时间对校友信息进行排序、修改密码。

除此之外，管理员用户多增加的功能有：添加校友信息、删除校友信息、列出所有用户、添加用户、删除用户、修改任意用户权限、修改任意用户密码。

3.2 系统分析及设计

本系统采用面向对象的程序设计方法设计而成，上文提到的大部分模块均由相应类组成，其中：

用户管理模块包含：UserInfo 结构、UserContainer 类，其中 UserInfo 结构包含单个用

户的密码和权限，而 `UserContainer` 类对一个用户名到 `UserInfo` 的映射（可使用 C++ STL 中的 `std::map` 实现）进行了包装并提供了相应的底层操作接口。

校友信息管理模块包含：`RecordType` 结构、`RecordContainer` 类，其中 `RecordType` 结构包含单条校友信息，`RecordContainer` 类对一个 `RecordType` 的变长数组（可使用 C++ STL 中的 `std::vector` 实现）进行了包装并提供了相应的底层操作接口。

命令解析及操作模块包含：`AbstractWorker` 抽象基类、`UserWorker` 类、`AdministratorWorker` 类，其继承关系见图 2。

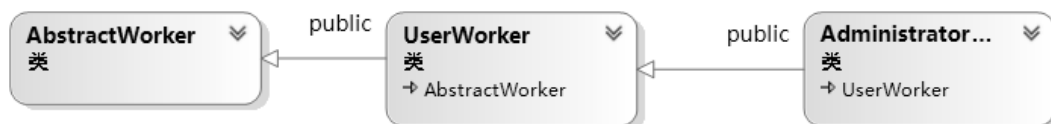


图 2 命令解析及操作模块中类的继承图

除此之外，还有一些异常处理类，例如 `LoginFailException`、`CommandLineSyntaxError`、`ValueExistException`、`KeyExistException` 等（均继承于 `std::runtime_error` 类），用于处理相关异常。

`AbstractWorker` 抽象基类中声明了相关虚函数，`UserWorker` 类中实现了普通用户可执行的相关操作，`Administrator` 类则增加了只有管理员能够执行的相关操作。这样采用继承的设计方法为实际操作时进行动态绑定（使用面向对象的多态性特性）提供了必要前提。

注册和登录模块由于没有很强的对象特征并未采用面向对象的程序设计方法设计，而是直接由相关函数组成，灵活机动。

3.3 系统的实现

(1) 类的编写

系统工程名为 ADMS，包含的主要类已在分析中列出，此处不再赘述。

具体类结构声明如下：

- **UserContainer 类**

```
class UserContainer {
public:
    //别名
    using UserName = std::string;
    using Iterator = std::map<UserName, UserInfo>::iterator;

    UserContainer();
    //禁止拷贝
    UserContainer(const UserContainer&) = delete;
    ~UserContainer() = default;
```

```
//禁止拷贝
UserContainer& operator=(const UserContainer&) = delete;

//添加用户
void AddUser(const std::string& username, const std::string& password,
const Privilege& privilege);

//删除用户
void DeleteUser(const std::string username);

//根据用户名查询密码
std::string QueryPassword(const std::string& username) const;

//根据用户名查询权限
Privilege QueryPrivilege(const std::string& username) const;

//改密码
void ChangePassword(const std::string& username, const std::string&
new_password);

//改权限
void ChangePrivilege(const std::string& username, const Privilege&
new_privilege);

//列出所有用户
std::set<std::string> ListUser();

//将容器内容保存到文件
void Save();

//登录（用于校验）
AbstractWorker* Login(const std::string& username, const std::string&
password);

//检测容器是否为空
bool Empty();

//注册第一个管理员用户
void RegisterFirstUser(const std::string& username, const
std::string& password);

private:
//核心容器，Key 为用户名，Value 为 UserInfo 的 map
std::map<UserName, UserInfo> CoreContainer;
};
```

- **RecordContainer 类**

```
class RecordContainer {
public:
    //别名
    using SizeType = std::vector<RecordType>::size_type;
    using Iterator = std::vector<RecordType>::iterator;

    RecordContainer();
    //禁止拷贝
    RecordContainer(const RecordContainer&) = delete;
    ~RecordContainer() = default;

    //禁止拷贝
    RecordContainer& operator=(const RecordContainer&) = delete;

    //添加记录
    void AddRecord(const RecordType& new_record);
    //删除记录
    void DeleteRecord(const Iterator& it);
    //查询记录
    std::vector<Iterator> QueryRecord(const RecordType& key);
    //按名字排序
    void SortedByName();
    //按毕业时间排序
    void SortedByGraduateYear();
    //将容器内容保存到文件
    void Save();

private:
    //核心容器，是 RecordType 类型 vector
    std::vector<RecordType> CoreContainer;
};
```

- **AbstractWorker 类**

```
class AbstractWorker {
public:
```

```
AbstractWorker() = default;
//每个程序仅存在一个工作对象，禁止拷贝
AbstractWorker(const AbstractWorker&) = delete;
AbstractWorker(const std::string& username, const std::string& pwd,
UserContainer& uc):
    username(username), password(pwd), users(uc){}
//虚析构函数，防止基类部分不被析构
virtual ~AbstractWorker() = default;

//同拷贝构造函数，禁止拷贝
AbstractWorker& operator=(const AbstractWorker&) = delete;

//输出主提示符 >>>
void PrintMainPrompt() const;
//输出次提示符 ...
void PrintSecondaryPrompt() const;
//清屏
void Clear() const;
//退出程序
void Exit() const;
//将多行输入处理、整合成单个命令字符串
std::string GetCommandString();

//解析命令字符串，调用类中相关函数
virtual void Parse() = 0;
//输出登录成功界面
virtual void PrintHello() = 0;
//输出帮助
virtual void PrintHelp() = 0;
//列出所有记录
virtual void PrintListRecord() = 0;
//查询记录
virtual void PrintQueryRecord(const RecordType& query_key) = 0;
//输出电话号码二维码
virtual void ShowTelephoneQRCode(const
std::vector<RecordContainer::Iterator>::size_type& index) = 0;
```

```
        //输出电子邮箱二维码
        virtual void ShowEmailQRCode(const
std::vector<RecordContainer::Iterator>::size_type& index) = 0;

        //按名字排序
        virtual void SortedByName() = 0;

        //按毕业时间排序
        virtual void SortedByGraduateYear() = 0;

protected:
        //用户名
        std::string username;

        //密码
        std::string password;

        //用户信息容器
        UserContainer& users;

        //记录容器
        RecordContainer records;

        //记录查询缓存容器
        std::vector<RecordContainer::Iterator> record_query;

};
```

● UserWorker 类

```
class UserWorker: public AbstractWorker {
public:
    UserWorker() = default;

    //禁止拷贝
    UserWorker(const UserWorker&) = delete;

    UserWorker(const std::string& usrname, const std::string& pwd,
UserContainer& uc):
        AbstractWorker(usrname, pwd, uc) {}

    ~UserWorker() = default;

    //禁止拷贝
    UserWorker& operator=(const UserWorker&) = default;

    //解析命令字符串，调用类中相关函数
    void Parse() override;
```



```

        //输出登录成功界面
        void PrintHello() override;

        //输出帮助
        void PrintHelp() override;

        //列出所有记录
        void PrintListRecord() override;

        //查询记录
        void PrintQueryRecord(const RecordType& query_key) override;

        //输出电话号码二维码
        void ShowTelephoneQRCode(const
std::vector<RecordContainer::Iterator>::size_type& index) override;

        //输出电子邮箱二维码
        void ShowEmailQRCode(const
std::vector<RecordContainer::Iterator>::size_type& index) override;

        //按名字排序
        void SortedByName() override;

        //按毕业时间排序
        void SortedByGraduateYear() override;

        //改密码
        void ChangePassword(std::string& new_password) const;

};

```

● Administrator 类

```

class AdministratorWorker: public UserWorker {
public:
    //阻止默认构造
    AdministratorWorker() = delete;

    //每个程序仅存在一个工作对象，禁止拷贝
    AdministratorWorker(const AdministratorWorker&) = delete;

    AdministratorWorker(const std::string& username, const std::string&
pwd,
        UserContainer& uc): UserWorker(username, pwd, uc) {}

    ~AdministratorWorker() = default;

    //同拷贝构造函数，禁止拷贝
    AdministratorWorker& operator=(const AdministratorWorker&) = default;

```

```
//解析命令字符串，调用类中相关函数
void Parse() override;

//输出登录成功界面
void PrintHello() override;

//输出帮助
void PrintHelp() override;

//添加记录
void AddRecord(const RecordType& new_record);

//删除记录
void DeleteRecord(const
std::vector<RecordContainer::Iterator>::size_type& index);

//列出用户
void ListUser();

//添加用户
void AddUser(const std::string& new_username, const std::string&
new_password,
const Privilege& new_privilege);

//删除用户
void DeleteUser(const std::string& username);

//改任意用户的密码
void ChangePassword(std::string& username, std::string& new_password)
const;

//改任意用户权限
void ChangePrivilege(const std::string& username, const Privilege&
new_privilege);

private:
//用户查询缓存容器
std::set<std::string> user_query;
};
```

(2) 注册及登录模块的实现

● 密码获取函数

如下图（图3），在使用密码输入函数获取密码时会将密码显示*，防止他人偷窥。

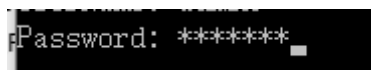


图3 密码输入图

实现原理：采用 Windows 支持的_getch()函数获取字符，使用_putch()函数输出单个字符。源码如下：

```
std::string GetPassword()
{
    std::string password;
    int cnt = 0;
    char tmp;
    while((tmp = _getch()) != '\r') {
        if(tmp == '\b') {
            if(cnt > 0) {
                //删除屏幕上一个字符
                --cnt;
                _putch('\b');
                _putch(' ');
                _putch('\b');
                password = password.substr(0, password.size() - 1);
            }
        }
        else {
            //输出 *
            _putch('*');
            password += tmp;
            ++cnt;
        }
    }
    _putch('\n');
    return password;
}
```

- 登录逻辑

以下为登录过程的流程图（图4）：

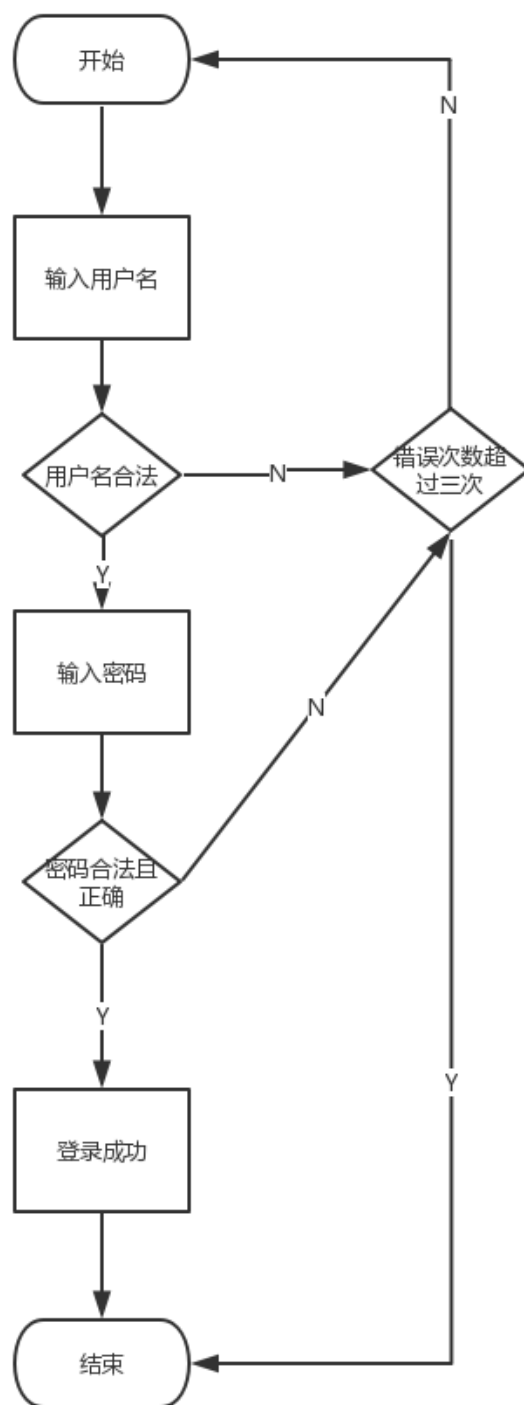


图4 登录流程图

(3) 命令行多行命令输入的实现（在 AbstractWorker 类中实现）

为了提供更好的用户体验，本程序支持将命令代码分成多行输入，在命令未完成的行输出提示符“...”，在命令已完成的行输出提示符“>>>”，由用户输入的“;”判断输入结束。

其大致流程是：读取一行到字符串，判断其中是否存在“;”，若不存在，则继续读取

下一行；若存在，判断“;”后是否存在非空字符，存在则报错，不存在就可以将其与之前的字符串连接起来，即得原始的命令字符串。

由于查询功能用到了“=”语法，即类似“query record email=xxx@qq.com;”的命令语法，原始命令字符串还需进行进一步处理。这里的做法是去除所有“=”并在第一个“=”之前的单词前放置“EQUAL_FORM_MARK”，用于进一步处理。

实现代码如下：

```
std::string AbstractWorker::GetCommandString()
{
    //最后返回的字符串
    std::string ret;
    //临时字符串
    std::string temp;
    bool firstline = true;
    PrintMainPrompt();
    while(1) {
        //读取一行
        std::getline(std::cin, temp);
        if(temp == "" && firstline == true) {
            PrintMainPrompt();
            continue;
        }
        firstline = false;
        //提取：之前的字符串
        std::size_t semicolon_pos = temp.find(';');
        if(semicolon_pos < temp.size()) {
            //之后若还有非空格字符，直接抛出语法错误
            for(std::size_t i = semicolon_pos + 1; i < temp.size(); ++i)
            {
                if(temp[i] != ' ') {
                    throw CommandLineSyntaxError(std::string("ERROR:
Syntax Error !"));
                }
            }
            ret += (" " + temp.substr(0, semicolon_pos));
            std::size_t equal_pos;
            bool first_equal = true;
```

```
//去=并在第一个等号前放置 EQUAL_FORM_MARK
while((equal_pos = ret.find('=')) < ret.size()) {
    ret.replace(equal_pos, 1, " ");
    if(first_equal) {
        while(--equal_pos > 0 && ret[equal_pos] == ' ');
        while(--equal_pos > 0 && ret[equal_pos] != ' ');
        ret.replace(equal_pos, 1, " EQUAL_FORM_MARK ");
        first_equal = false;
    }
}

return ret;
}

ret += (" " + temp);
PrintSecondaryPrompt();
}
}
```

(3) 命令字符串解析并调取相关函数的功能实现

采用 `std::stringstream` 读取命令字符串，分次赋值给 `std::string` 对象，运用 `if` 语句逐个判断，最后解析得语意，调用相关函数，因代码过长，不在此赘述。

其中用到了一个自行设计的判断 `std::stringstream` 是否为空的函数，在此列出代码：

```
bool CheckEmpty(std::stringstream& ss)
{
    std::string temp, templine;
    ss >> temp;
    if(ss) {
        //先取出来，再放进去
        std::getline(ss, templine);
        ss.clear();
        ss << temp << ' ' << templine;
        return false;
    }
    else {
        return true;
    }
}
```

四、 实验调试、测试、运行记录及分析

正确的运行结果

首次运行程序会提示注册第一个管理员账号（图 5）

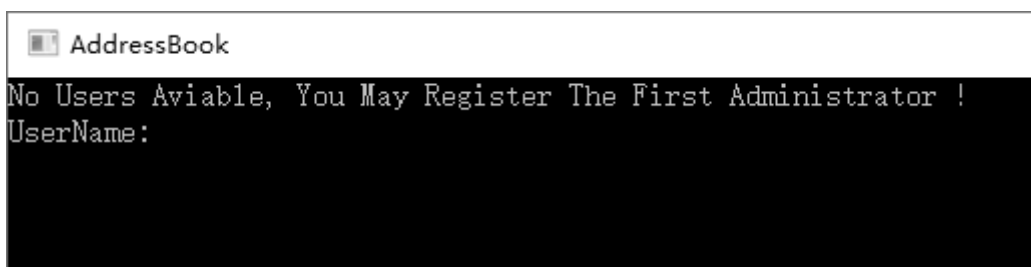


图 5

注册完成（图 6）



图 6

以管理员账号登录（图 7）

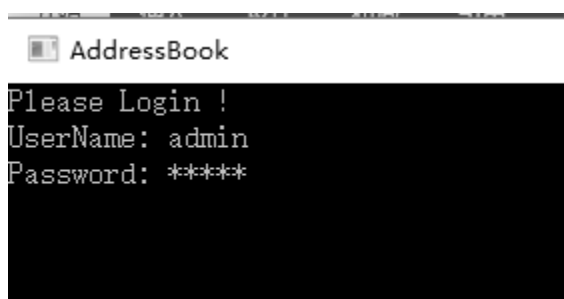


图 7

登录成功界面（图 8）

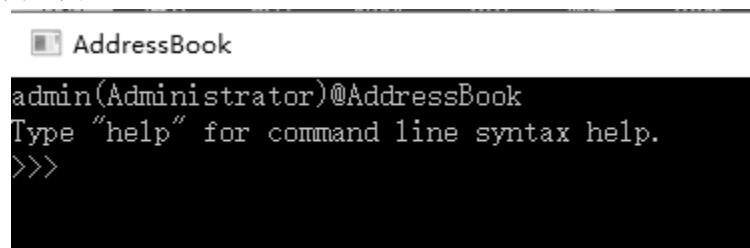


图 8

输入“help;”查看管理员支持的命令（图 9）

```

AddressBook
admin(Administrator)@AddressBook
Type "help" for command line syntax help.
>>>help;
Commands Available:
clear;                                清屏
help;                                输出帮助
add record <[record1]...>;           增加记录
add user <username> <User|Administrator>; 增加用户
list record;                          列出记录
list user;                            列出用户
delete record <index>;               删除记录
delete user <username>;              删除用户
showqr phone <index>;               在查询记录后指定以二维码形式输出某记录的电话
showqr email <index>;               在查询记录后指定以二维码形式输出某记录的电子邮箱
sort <name|graduate_year>;          按 姓名|毕业年份 对记录进行排序
change password <username>;         修改某帐户密码
change privilege <username> <User|Administrator>; 修改某帐户权限
query record <[key]=[value]>;       查询记录
>>>

```

图 9

录入校友信息（图 10）

```

>>>add record
...张三 男 45 1996 计算机 9603 浙江省杭州市拱墅区湖州街51号 15785569625 766524565 zhangsan@qq.com
...李四 女 46 1995 化工 9501 浙江省杭州市西湖区文二路328号b座2楼 13864475268 65871265 lisi@zjut.edu.cn
...王五 男 58 1984 生物技术 8402 山东省济南市山大南路27号 15789542145 5374158 wangfive@foxmail.com
...吴依洁 女 34 2008 艺术 0806 湖北省武汉市武昌东湖路147号 13866547852 7755862 wuuu@outlook.com
...余迎秋 女 43 1999 环境 9903 山西省太原市坞城路92号 15877525423 865956963 yinqiu@gamil.com
...佟云霞 女 47 1995 机械 9502 天津市卫津路94号 18755254585 875624863 xia95@163.com
...;
6 Record Added Successfully !

```

图 10

列出所有校友信息（图 11）

```
>>>list record;
```

Index	Name	Sex	Age	Graduate_Year	Department	Class	Address	Phone	QQ	Email
1	张三	男	45	1996	计算机	9603	浙江省杭州市拱墅区湖州街51号	15785569625	766524565	zhangsan@qq.com
2	李四	女	46	1995	化工	9501	浙江省杭州市西湖区文二路328号b座2楼	13864475268	65871265	lisi@zjut.edu.cn
3	王五	男	58	1984	生物技术	8402	山东省济南市山大南路27号	15789542145	5374158	wangfive@foxmail.com
4	吴依洁	女	34	2008	艺术	0806	湖北省武汉市武昌东湖路147号	13866547852	7755862	wuuu@outlook.com
5	余迎秋	女	43	1999	环境	9903	山西省太原市坞城路92号	15877525423	865956963	yinqiu@gamil.com
6	佟云霞	女	47	1995	机械	9502	天津市卫津路94号	18755254585	875624863	xia95@163.com

图 11

删除校友信息（图 12）

```

>>>delete record 1;
Record Deleted Successfully !
>>>list record;

```

Index	Name	Sex	Age	Graduate_Year	Department	Class	Address	Phone	QQ	Email
1	李四	女	46	1995	化工	9501	浙江省杭州市西湖区文二路328号b座2楼	13864475268	65871265	lisi@zjut.edu.cn
2	王五	男	58	1984	生物技术	8402	山东省济南市山大南路27号	15789542145	5374158	wangfive@foxmail.com
3	吴依洁	女	34	2008	艺术	0806	湖北省武汉市武昌东湖路147号	13866547852	7755862	wuuu@outlook.com
4	余迎秋	女	43	1999	环境	9903	山西省太原市坞城路92号	15877525423	865956963	yinqiu@gamil.com
5	佟云霞	女	47	1995	机械	9502	天津市卫津路94号	18755254585	875624863	xia95@163.com

图 12

查询校友信息（图 13）

```
>>>query record name=李四 sex=女;
```

Index	Name	Sex	Age	Graduate_Year	Department	Class	Address	Phone	QQ	Email
1	李四	女	46	1995	化工	9501	浙江省杭州市西湖区文二路328号b座2楼	13864475268	65871265	lisi@zjut.edu.cn

图 13

展示校友邮箱二维码（图 14）

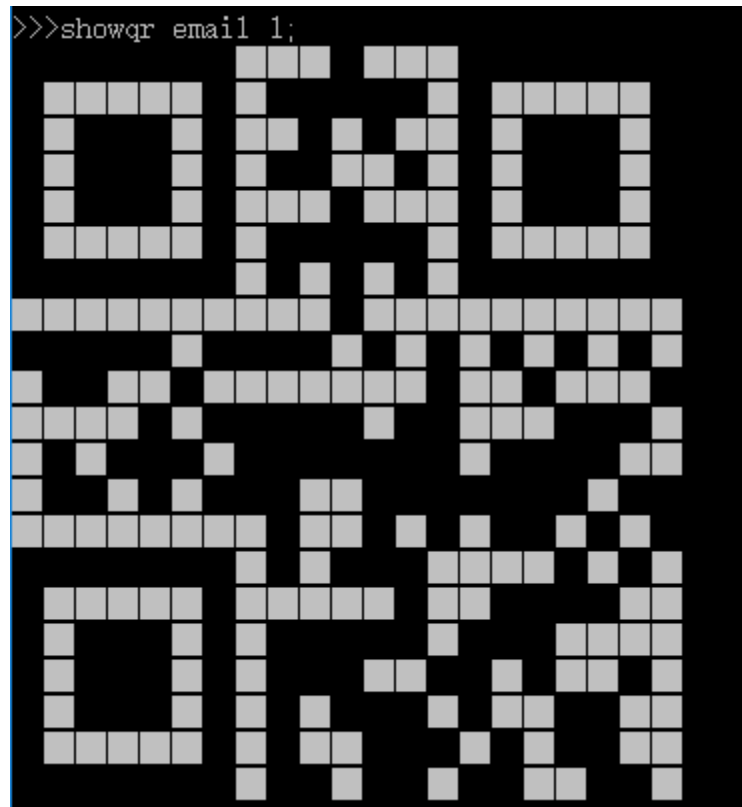


图 14

展示校友电话二维码（图 15）

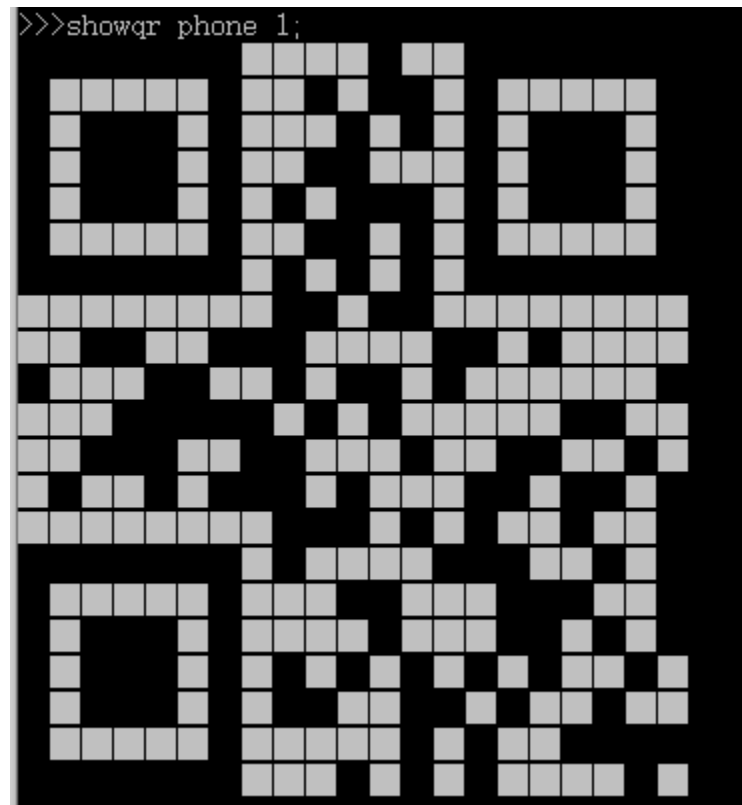


图 15

按姓名排序（图 16）

```
>>>list record;
```

Index	Name	Sex	Age	Graduate_Year	Department	Class	Address	Phone	QQ	Email
1	李四	女	46	1995	化工	9501	浙江省杭州市西湖区文二路328号b座2楼	13864475268	65871265	lisi@zjut.edu.cn
2	王五	男	58	1984	生物技术	8402	山东省济南市山大南路27号	15789542145	5374158	wangfive@foxmail.com
3	吴依洁	女	34	2008	艺术	0806	湖北省武汉市武昌东湖路147号	13866547852	7755862	wuuu@outlook.com
4	余迎秋	女	43	1999	环境	9903	山西省太原市坞城路92号	15877525423	865956963	yinqiu@gamil.com

图 16

按毕业时间排序（图 17）

```
>>>sort graduate_year;
Sorted by Graduate_Year Successfully !
>>>list record;
```

Index	Name	Sex	Age	Graduate_Year	Department	Class	Address	Phone	QQ	Email
1	王五	男	58	1984	生物技术	8402	山东省济南市山大南路27号	15789542145	5374158	wangfive@foxmail.com
2	李四	女	46	1995	化工	9501	浙江省杭州市西湖区文二路328号b座2楼	13864475268	65871265	lisi@zjut.edu.cn
3	余迎秋	女	43	1999	环境	9903	山西省太原市坞城路92号	15877525423	865956963	yinqiu@gamil.com
4	吴依洁	女	34	2008	艺术	0806	湖北省武汉市武昌东湖路147号	13866547852	7755862	wuuu@outlook.com

图 17

增加用户、查询用户（图 18）

```
>>>add user U1 User;
New Password: *
New Password Again: *
User Add Successfully !
>>>list user;
Here Are The Users:
UserName | Password
      U1 |      User
  admin | Administrator
```

图 18

删除用户（图 19）

```
>>>delete user U1;
User Deleted Successfully !
>>>list user;
Here Are The Users:
UserName | Password
  admin | Administrator
```

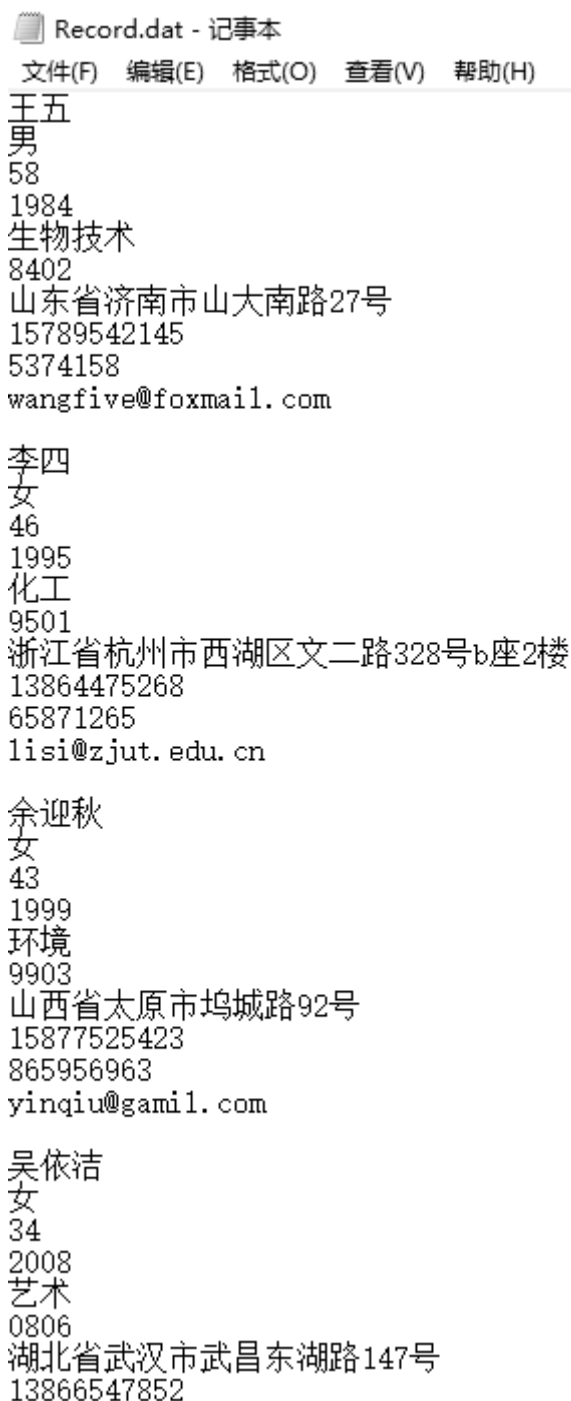
图 19

修改密码（图 20）

```
>>>change password admin;  
New Password: *  
New Password Again: *  
Password of admin Changed Successfully !
```

图 20

操作结束之后文件内的信息，可见信息被正确保存（图 21）



Record.dat - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

王五
男
58
1984
生物技术
8402
山东省济南市山大南路27号
15789542145
5374158
wangfive@foxmail.com

李四
女
46
1995
化工
9501
浙江省杭州市西湖区文二路328号b座2楼
13864475268
65871265
lisi@zjut.edu.cn

余迎秋
女
43
1999
环境
9903
山西省太原市坞城路92号
15877525423
865956963
yinqiu@gamil.com

吴依洁
女
34
2008
艺术
0806
湖北省武汉市武昌东湖路147号
13866547852

图 21-1

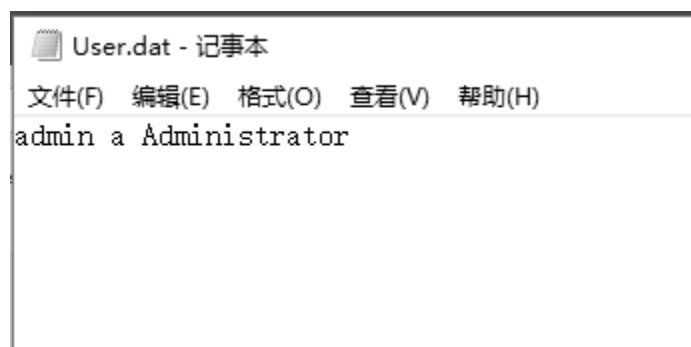


图 21-2

当使用非管理员账号登录时，支持的命令是管理员时的子集，如下，不再赘述（图 22）

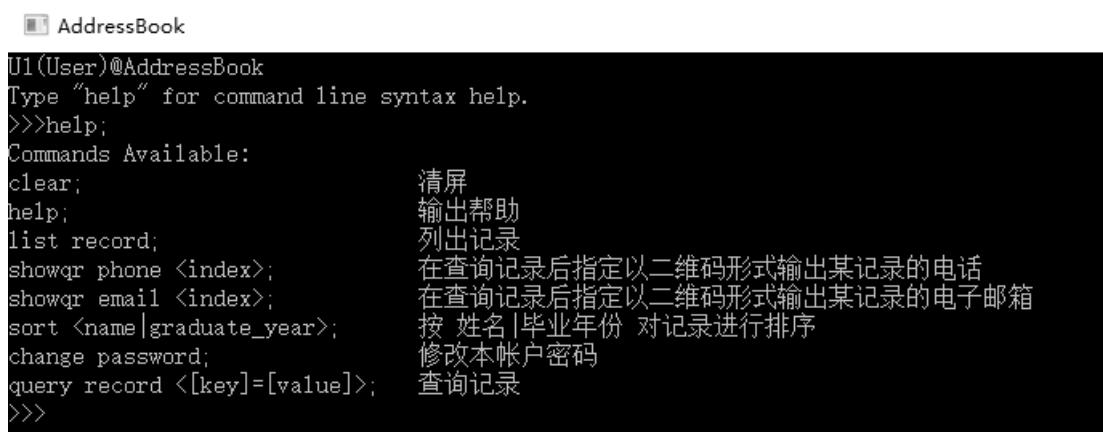


图 22

遇到的问题及解决方法如下：

● 问题 1:

问题描述：在一次编译过程中，发生奇怪错误，Visual Studio 2017 提示已定义的类型未定义，G++ 7.1.0 提示缺少分号，在反复核实后发现错误提示中所说的问题并不存在。

解决方法：后请教同学和查阅网上资料得知此问题是由于头文件循环包含而导致的，可以使用类似下图（图 23）的前置声明方法解决循环依赖问题，或重新组织头文件，消除循环包含。

```

//前置类声明
class AbstractWorker;
class UserContainer;
  
```

图 23 前置声明

● 问题 2:

问题描述：在一次运行过程中 Visual Studio 2017 监测到内存占用急速升高，如下图（图 24）。

解决方法：经过断点调试发现原因是出现了一个死循环不断构造文件流对象，修改跳出循环条件，使循环能够正确终止，并正确析构文件流对象，即解决问题。

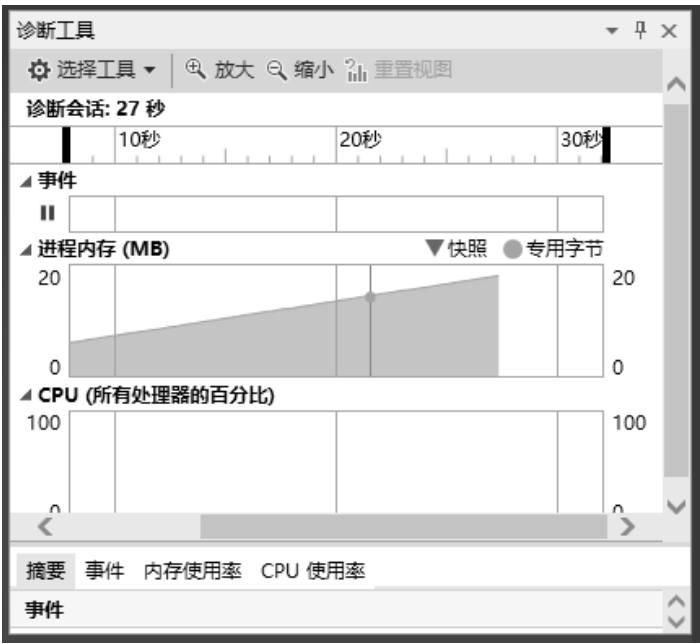


图 24 内存占用

● 问题 3:

问题描述: 由于窗口太窄, 无法正常显示表格, 如图 (图 25)。

解决方法: 后查阅资料, 咨询同学得知可以使用代码控制窗口列数, 如图, 解决。



图 25-1 表格显示错位

```
//将窗口列数调大, 以正确显示表格
std::system("mode con cols=150");
```

图 25-2 设置窗口列数

● 问题 5:

问题描述: 原以为采用如图的“头文件卫士”就不用担心重复定义的问题,但还是产生了如图(图 26)“找到了一个或多个多重定义的符号”错误。

解决方法: 思考之后,发现由于 C++ 以一个 .cpp 文件作为编译单位,当同一个头文件被不同 .cpp 文件包含之后就相当于同一个变量或函数被定义了很多遍,会导致链接错误。

```
4
5  #ifndef ADDRESSBOOK_USERCONTAINER_H
6  #define ADDRESSBOOK_USERCONTAINER_H
7
8  #include <set>
9  #include <map>
10 #include <vector>
11 #include <string>
12 #include <stdexcept>
13 #include "GlobalDefine.h"
14
15 namespace hua { ... }
92
93 #endif
94
```

图 26-1 “头文件卫士”

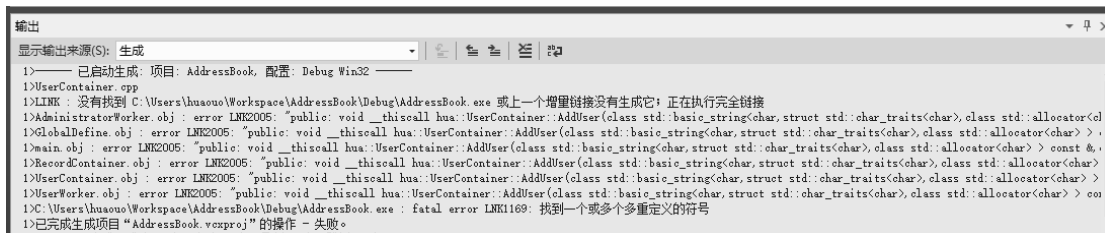


图 26-2 报错

五、实验总结

我设计的校友录管理系统基本满足任务书的功能要求,还增加了用户管理功能(带权限)、二维码显示等特色、创新功能,其中用户管理功能为使用者和管理者带来了切实的便利之处,在防止数据被误删等方面起到了重要的作用。在数据存储上充分使用了具有工业强度的 STL 容器(`std::vector` 和 `std::map`)保证了数据存储、查询等操作的高效性。在面向对象的基础上,我对裸露的 STL 容器设计了两层包装,每层提供了友好的接口,在减轻编码难度、提高了程序的可读性的同时减少了出错的机会。

存在的缺点是,对于命令的解析,没有找到好方法,而采用传统的 if-else 方法逐词解析命令。这种做法不仅编码效率低下,更容易导致判断的疏忽和纰漏。在完成编码之后,通过在线查询资料,我了解到现代编译器往往采用所谓的“递归下降法”解析命令,如果本实验采取这种方法,程序的简洁性一定会得到大幅提升。

通过这次 C++ 的大型实验,我深刻的明白到:在开始编码之前必须经过深刻思考,而不

应该想到哪，就直接开始编码充分的思考可以避免很多的设计错误、缩短编码时间、提高工作效率。其次，当调试遇到错误的时候应该冷静分析，设置断点进行调试，从而改正错误，而不应该失去信心、半途而废。

六、 附录：源代码