## 1.数据访问层(Data) 3个

#define MAX\_CONTACTS 100

### // 1.1结构体定义

struct ContactRecord {

char name[20];

char phone[12];

char address[50];

char postcode[8];

char email[20];

};

### 1.2读取联系人信息

int readContacts(struct ContactRecord contacts[], int \*num\_contacts);  
  
/\*  
负责人:  
功能: 从文件中读取联系人信息，并存储到 contacts 数组中。  
参数:   
    contacts[]: 结构体数组，用于存储读取的联系人信息。  
    num\_contacts: 指针，用于存储读取的联系人数量。  
返回值:   
    0: 读取成功  
    -1: 读取失败  
\*/

### 1.3写入联系人信息

int writeContacts(struct ContactRecord contacts[], int num\_contacts);  
  
/\*  
负责人:  
功能: 将 contacts 数组中的联系人信息写入文件。  
参数:   
    contacts[]: 结构体数组，包含要写入的联系人信息。  
    num\_contacts: 联系人数量。  
返回值:   
    0: 写入成功  
    -1: 写入失败  
\*/

### 1.4验证用户的账号和密码

bool authenticateUser(const char \*filename, const char \*username, const char \*password)

/\*  
负责人:  
功能: 验证用户身份，从指定文件中读取用户名和密码，并与用户输入的用户名和密码进行比较。  
参数:  
    filename: 用户信息文件的文件名。  
    username: 用户输入的用户名。  
    password: 用户输入的密码。  
返回值:  
    true: 验证成功  
    false: 验证失败  
\*/

# 2服务层(serve) 4个

// 2.1 添加联系人信息(已有)  
int addContact(struct ContactRecord contacts[], int \*num\_contacts);  
  
/\*  
负责人:  
功能: 添加新的联系人信息到 contacts 数组中。  
参数:   
    contacts[]: 结构体数组，用于存储联系人信息。  
    num\_contacts: 指针，用于存储联系人数量。  
返回值:   
    0: 添加成功  
    -1: 添加失败  
\*/

// 2.2 列出所有联系人信息(已有)  
void listContacts(struct ContactRecord contacts[], int num\_contacts);  
  
/\*  
负责人:  
功能: 打印所有联系人信息到控制台。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
返回值:   
    void  
\*/

// 2.3 按姓名查找联系人(已有)  
int searchContactByName(struct ContactRecord contacts[], int num\_contacts, char \*name);  
  
/\*  
负责人:  
功能: 在 contacts 数组中查找指定姓名的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
    name: 要查找的联系人姓名。  
返回值:   
    -1: 未找到联系人  
    >= 0: 找到联系人，返回联系人索引  
\*/

// 2.4 按电话号码查找联系人(已有)  
int searchContactByPhone(struct ContactRecord contacts[], int num\_contacts, char \*phone);  
  
/\*  
负责人:  
功能: 在 contacts 数组中查找指定电话号码的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
    phone: 要查找的联系人电话号码。  
返回值:   
    -1: 未找到联系人  
    >= 0: 找到联系人，返回联系人索引  
\*/

2.5 按地址查找联系人  
int searchContactByAddress(struct ContactRecord contacts[], int num\_contacts, char \*address);  
  
/\*  
负责人:  
功能: 在 contacts 数组中查找指定地址的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
    address: 要查找的联系人地址。  
返回值:   
    -1: 未找到联系人  
    >= 0: 找到联系人，返回联系人索引  
\*/

2.6 按邮编查找联系人  
int searchContactByPostcode(struct ContactRecord contacts[], int num\_contacts, char \*postcode);  
  
/\*  
负责人:  
功能: 在 contacts 数组中查找指定邮编的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
    postcode: 要查找的联系人邮编。  
返回值:   
    -1: 未找到联系人  
    >= 0: 找到联系人，返回联系人索引  
\*/

2.7 按邮箱查找联系人  
int searchContactByEmail(struct ContactRecord contacts[], int num\_contacts, char \*email);  
  
/\*  
负责人:  
功能: 在 contacts 数组中查找指定邮箱的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
    email: 要查找的联系人邮箱地址。  
返回值:   
    -1: 未找到联系人  
    >= 0: 找到联系人，返回联系人索引  
\*/

2.8 删除联系人信息  
int deleteContact(struct ContactRecord contacts[], int \*num\_contacts, char \*key, int delete\_type);  
  
/\*  
负责人:  
功能: 从 contacts 数组中删除指定条件的联系人。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 指针，用于存储联系人数量。  
    key: 要删除的联系人信息，例如姓名、电话号码或邮箱。  
    delete\_type: 删除类型，0 为按姓名删除，1 为按电话号码删除，2 为按邮箱删除。  
返回值:   
    0: 删除成功  
    -1: 删除失败  
\*/

//2.9 按姓名排序联系人信息(已有)  
int sortContactsByName(struct ContactRecord contacts[], int num\_contacts);  
  
/\*  
负责人:  
功能: 按姓名对 contacts 数组中的联系人信息进行排序。  
参数:   
    contacts[]: 结构体数组，包含联系人信息。  
    num\_contacts: 联系人数量。  
返回值:   
    0: 排序成功  
    -1: 排序失败  
\*/

// 以下均为拓展  
// 2.10 模糊搜索联系人  
int fuzzySearchContacts(struct ContactRecord contacts[], int num\_contacts, char \*keyword);  
  
// 2.11 多条件搜索联系人  
int multiConditionSearch(struct ContactRecord contacts[], int num\_contacts, char \*name, char \*phone, char \*address, char \*postcode, char \*email);  
  
// 2.12 按分组查找联系人  
int searchContactsByGroup(struct ContactRecord contacts[], int num\_contacts, char \*group);  
  
// 2.13 添加分组  
int addGroup(struct ContactRecord contacts[], int \*num\_contacts, char \*group);  
  
// 2.14 删除分组  
int deleteGroup(struct ContactRecord contacts[], int \*num\_contacts, char \*group);  
  
// 2.15 为联系人添加标签  
int addTagToContact(struct ContactRecord contacts[], int num\_contacts, char \*contactName, char \*tag);  
  
// 2.16 删除联系人标签  
int deleteTagFromContact(struct ContactRecord contacts[], int num\_contacts, char \*contactName, char \*tag);

3.表示层(View) 7个

// 3.1 显示菜单并获取用户选择(已有)  
int menuSelect();  
  
/\*  
负责人:  
功能: 显示菜单选项并获取用户的选择。  
参数: void  
返回值: 用户选择的值，例如 1、2、3 等。  
\*/

3.2 显示登录界面  
int showLoginInterface();  
  
/\*  
负责人:  
功能: 显示登录界面，并获取用户输入的用户名和密码。  
参数: void  
返回值:   
    0: 登录成功  
    -1: 登录失败  
\*/

3.3 显示主界面  
void showMainInterface();  
  
/\*  
负责人:  
功能: 显示主界面，包括各种功能选项，例如添加联系人、删除联系人等。  
参数: void  
返回值: void  
\*/

3.4 显示编辑记录界面  
void showEditRecordInterface(struct ContactRecord contacts[], int num\_contacts);  
/\*  
负责人:  
功能: 显示编辑记录界面，允许用户选择要编辑的联系人，并输入新的联系信息。  
参数:  
contacts: 存放联系人的数组。  
num\_contacts: 数组中联系人的数量。  
返回值: void  
\*/

// 3.5 显示删除记录界面  
void showDeleteRecordInterface(struct ContactRecord contacts[], int num\_contacts);  
/\*  
负责人:  
功能: 显示删除记录界面，允许用户选择要删除的联系人，并确认删除操作。  
参数:  
contacts: 存放联系人的数组。  
num\_contacts: 数组中联系人的数量。  
返回值: void  
\*/

3.6 显示插入记录界面  
void showInsertRecordInterface(struct ContactRecord contacts[], int \*num\_contacts);  
/\*  
负责人:  
功能: 显示插入记录界面，允许用户输入新的联系人信息，并将新联系人插入到数组中。  
参数:  
contacts: 存放联系人的数组。  
num\_contacts: 数组中联系人的数量，该参数为指针，用于更新联系人数量。  
返回值: void  
\*/

// 3.7 显示查找界面  
void showSearchInterface(struct ContactRecord contacts[], int num\_contacts);  
/\*  
负责人:  
功能: 显示查找界面，允许用户输入关键字，并根据关键字查找匹配的联系人。  
参数:  
contacts: 存放联系人的数组。  
num\_contacts: 数组中联系人的数量。  
返回值: void  
\*/

// 3.8 显示分组管理界面(拓展)  
void showGroupManagementInterface(struct ContactRecord contacts[], int num\_contacts);  
  
// 3.9 显示标签系统界面(拓展)  
void showTagSystemInterface(struct ContactRecord contacts[], int num\_contacts);

// 主函数(已有)  
int main() {  
    // ... 代码逻辑 ...  
    return 0;  
}