**通讯录系统的设计与实现报告**

1. 设计目标

本题目要求用C语言实现通讯录管理系统的管理，包括联系人信息输入，查看，搜索，删除，保存等功能。

（二）功能要求：

1. 录入通信录信息；
2. 实现删除功能， 即输入姓名删除对应的记录；
3. 实现显示功能，即显示通讯录中所有的信息；
4. 实现 查找功能，即输入姓名显示对应信息；
5. 实现保存功能，即将输入的通讯录信息保存到指定的磁盘文件中；
6. 实现加载功能，即加载磁盘文件中保存的内容；
7. 退出通讯录管理系统。

（三）信息描述：

联系人姓名、电话、邮箱

（四）功能描述：

1，录入联系人信息：用户可以自由输入信息到通讯录系统中。

2，打印联系人信息：格式化展示系统中的联系人信息。

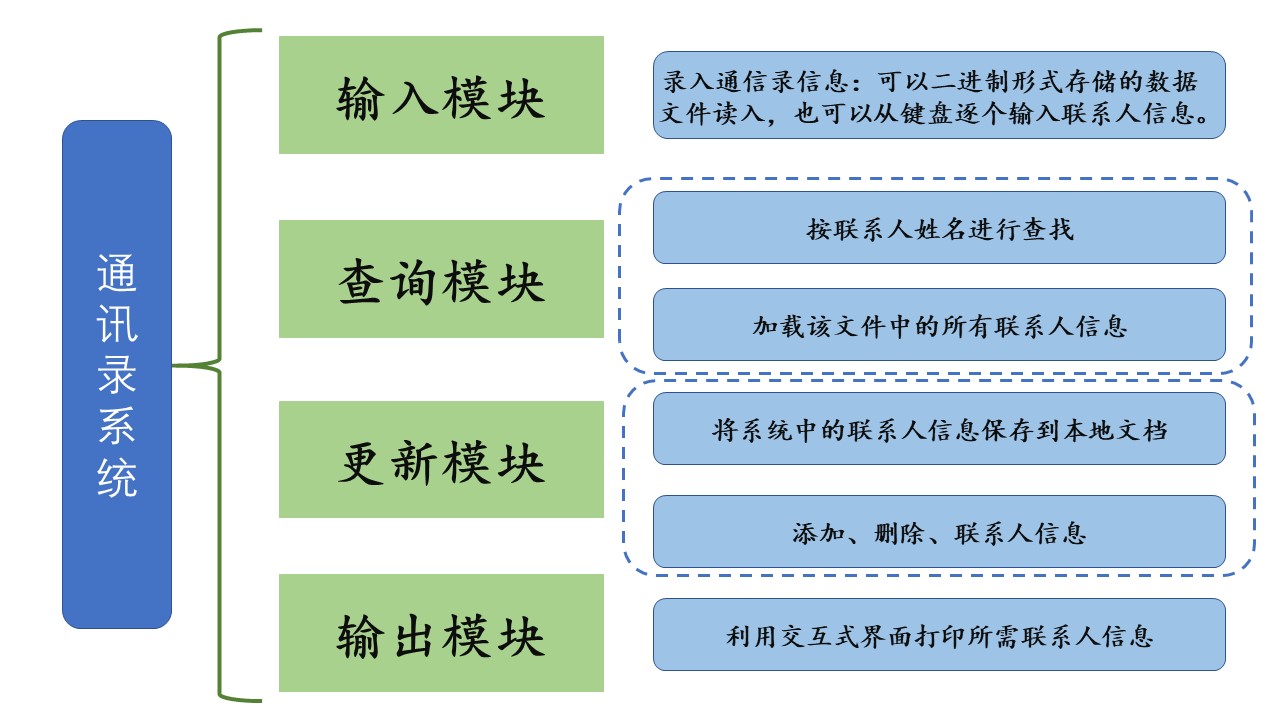
3，保存联系人信息：将系统中的联系人信息保存到本地文档。

4，读取联系人信息：加载本地文档中的联系人信息并显示。

5，查找联系人信息：根据姓名在系统中查找该联系人的信息。

6，删除联系人信息。

7，退出系统。

（五）总体设计：

（六）系统功能设计

1，输入模块：主要完成将数据存入一个数据文件中，在本管理系统中，记录可以以二进制形式存储的数据文件读入，也可以从键盘逐个输入联系人信息。

2，查询模块：主要完成查找满足相关条件的联系人记录，用户可以按联系人姓名进行查找，可以加载该文件中的所有联系人信息。

3，更新模块：主要完成对联系人记录的维护，实现对学生记录的删除、保存操作。

4，输出记录模块：主要完成两个任务：

i.实现对联系人记录的存盘操作，将联系人记录信息写入数据文件中。

ii.实现将联系人记录信息屏幕上输出。

（七）实现思路

1.数据结构：

通讯录条目使用结构体Entry表示，包含姓名、电话号码和电子邮件等字段。

通讯录使用结构体AddressBook表示，包含一个条目数组和当前条目数的计数。

功能函数：

add\_entry函数用于添加通讯录条目，通过输入获取姓名、电话号码和电子邮件，并将条目添加到通讯录中。

delete\_entry函数用于删除通讯录条目，通过输入要删除的姓名，在通讯录中查找匹配的条目并删除。

show\_all\_entries函数用于显示所有通讯录条目，遍历通讯录并逐个打印条目的姓名、电话号码和电子邮件。

find\_entry函数用于查找通讯录条目，通过输入要查找的姓名，在通讯录中查找匹配的条目并打印相关信息。

save\_address\_book函数用于将通讯录数据保存到文件中，打开指定的文件并将每个条目的信息格式化输出到文件中。

load\_address\_book函数用于从文件加载通讯录数据，读取文件中的每一行数据，并将姓名、电话号码和电子邮件解析后添加到通讯录中。

2.主函数：

主函数使用一个无限循环，通过输入选择要执行的操作。

根据用户的选择调用相应的功能函数来执行操作，如添加条目、删除条目、显示所有条目等。

提供选项编号为7时，退出循环，结束程序的执行。

3.文件操作：

通过使用标准C库中的fopen函数打开文件，可以选择以读取或写入的方式打开。

使用fprintf函数将数据格式化输出到文件中。

使用fgets函数逐行读取文件中的数据，并使用sscanf函数解析每一行数据中的字段。

（八）详细设计

1）程序采用简单菜单驱动方式，屏幕上显示菜单如下：

1—通讯录录入

2—通讯录查询

3—通讯录删除

4—通讯录列表显示

5—通讯录数据保存

6—通讯录数据加载

7— 退出系统

2）定义结构体以存储信息

1. *// 通讯录条目*
2. typedef struct {
3. char name[MAX\_NAME\_LENGTH];
4. char phone[MAX\_PHONE\_LENGTH];
5. char email[MAX\_EMAIL\_LENGTH];
6. } Entry;
7. *// 通讯录*
8. typedef struct {
9. Entry entries[MAX\_ENTRIES];
10. int num\_entries;
11. } AddressBook;

3）定义功能函数

1. 添加通讯录条目

1. 检查通讯录的条目数量是否已经达到了最大限制（MAX\_ENTRIES）。如果已经达到最大限制，即address\_book->num\_entries >= MAX\_ENTRIES，则打印出相应的提示信息"通讯录已满，无法添加条目。"，并使用return语句结束函数的执行。如果通讯录还有空间可以添加条目，则会继续执行下面的代码。
   1. 声明一个名为entry的指针，它指向了通讯录中的下一个可用条目。具体而言，entry指针指向address\_book->entries[address\_book->num\_entries]，即将要添加的条目的位置。
   2. 使用printf函数提示用户输入姓名，并通过scanf函数将用户输入的姓名存储到entry->name中，完成姓名信息的输入。同样的方式，依次输入电话号码和电子邮件信息。
   3. 递增通讯录的条目数量address\_book->num\_entries++，表示成功添加了一个条目。最后，打印添加成功的提示信息"添加成功。"。
2. *// 添加通讯录条目*
3. void add\_entry(AddressBook\* address\_book) {
4. if (address\_book->num\_entries >= MAX\_ENTRIES) {
5. printf("通讯录已满，无法添加条目。\n");
6. return;
7. }
8. Entry\* entry = &address\_book->entries[address\_book->num\_entries];
9. printf("请输入姓名：");*//使用指针指向结构体，完成信息输入*
10. scanf("%s", entry->name);
11. printf("请输入电话号码：");
12. scanf("%s", entry->phone);
13. printf("请输入电子邮件：");
14. scanf("%s", entry->email);
15. address\_book->num\_entries++;
16. printf("添加成功。\n");
17. }

2.删除通讯录条目

1. 声明一个名为name的字符数组，用于存储用户输入的要删除的姓名。然后通过printf函数提示用户输入要删除的姓名。
2. 使用scanf函数从用户输入中读取姓名，并将其存储在name数组中。
3. 声明两个变量found\_index和i，其中found\_index用于记录找到的条目在通讯录中的索引，i用于循环遍历通讯录中的条目。通过一个for循环，从0到address\_book->num\_entries - 1遍历通讯录中的条目。
4. 使用strcmp函数比较当前条目的姓名与用户输入的姓名是否匹配。如果匹配，将找到的索引保存在found\_index中，并使用break语句跳出循环。接下来，通过判断found\_index的值是否为-1，来确定是否找到了匹配的条目。
5. 如果found\_index为-1，表示未找到匹配的条目，将打印相应的提示信息，并使用return语句结束函数。如果found\_index不为-1，表示找到了匹配的条目。
6. 使用一个新的for循环，从found\_index开始遍历到address\_book->num\_entries - 1，将后面的条目依次往前移动一个位置，实现删除操作。
7. 更新通讯录的条目数量，即将address\_book->num\_entries减1，表示删除了一个条目，并打印删除成功的提示信息。
8. *// 删除通讯录条目*
9. void delete\_entry(AddressBook\* address\_book) {
10. char name[MAX\_NAME\_LENGTH];
11. printf("请输入要删除的姓名：");
12. scanf("%s", &name);
13. int found\_index = -1;
14. int i;
15. for ( i = 0; i < address\_book->num\_entries; i++) {
16. if (strcmp(address\_book->entries[i].name, name) == 0) {
17. found\_index = i;
18. break;
19. }
20. }
21. if (found\_index == -1) {
22. printf("未找到匹配条目。\n");
23. return;
24. }
25. for (i = found\_index; i < address\_book->num\_entries - 1; i++) {
26. address\_book->entries[i] = address\_book->entries[i + 1];
27. }
28. address\_book->num\_entries--;*//对应的计数也要减1*
29. printf("删除成功。\n");
30. }

3.显示所有通讯录条目

1. 使用printf函数打印表头，包括姓名、电话号码和电子邮件的标题。通过一个for循环，从0到address\_book->num\_entries - 1遍历通讯录中的条目。
2. 在循环中，使用指针entry指向当前遍历到的条目，通过&address\_book->entries[i]获取该条目的地址。
3. 在循环的每一次迭代中，使用printf函数按照格式"%s\t\t%s\t\t%s\n"打印出该条目的姓名、电话号码和电子邮件信息。
4. 通过entry->name、entry->phone和entry->email分别访问该条目的姓名、电话号码和电子邮件。
5. 依次打印出通讯录中所有条目的姓名、电话号码和电子邮件信息。
6. *// 显示所有通讯录条目*
7. void show\_all\_entries(AddressBook\* address\_book) {
8. printf("姓名\t\t电话号码\t\t电子邮件\n");
9. int i;
10. for ( i = 0; i < address\_book->num\_entries; i++) {
11. Entry\* entry = &address\_book->entries[i];
12. printf("%s\t\t%s\t\t%s\n", entry->name, entry->phone, entry->email);
13. }
14. }

4.查找通讯录条目

1. 声明一个名为name的字符数组，用于存储用户输入的要查找的姓名。然后通过printf函数提示用户输入要查找的姓名。
2. 使用scanf函数从用户输入中读取姓名，并将其存储在name数组中。
3. 声明两个变量found\_index和i，其中found\_index用于记录找到的条目在通讯录中的索引，i用于循环遍历通讯录中的条目。
4. 通过一个for循环，从0到address\_book->num\_entries - 1遍历通讯录中的条目。在循环中，使用strcmp函数比较当前条目的姓名与用户输入的姓名是否匹配。
5. 如果匹配，将找到的索引保存在found\_index中，并使用break语句跳出循环。接下来，通过判断found\_index的值是否为-1，来确定是否找到了匹配的条目。
6. 如果found\_index为-1，表示未找到匹配的条目，将打印相应的提示信息，并使用return语句结束函数。
7. 如果found\_index不为-1，表示找到了匹配的条目。通过将address\_book->entries[found\_index]赋值给指针entry，可以获得该条目的引用。
8. 使用printf函数打印该条目的姓名、电话号码和电子邮件信息。
9. *// 查找通讯录条目*
10. void find\_entry(AddressBook\* address\_book) {
11. char name[MAX\_NAME\_LENGTH];
12. printf("请输入要查找的姓名：");
13. scanf("%s", &name);
14. int found\_index = -1;
15. int i;
16. for (i = 0; i < address\_book->num\_entries; i++) {
17. if (strcmp(address\_book->entries[i].name, name) == 0) {
18. found\_index = i;
19. break;
20. }
21. }
22. if (found\_index == -1) {
23. printf("未找到匹配条目。\n");
24. return;
25. }
26. Entry\* entry = &address\_book->entries[found\_index];
27. printf("姓名：%s\n电话号码：%s\n电子邮件：%s\n", entry->name, entry->phone, entry->email);
28. }

5.保存通讯录到文件

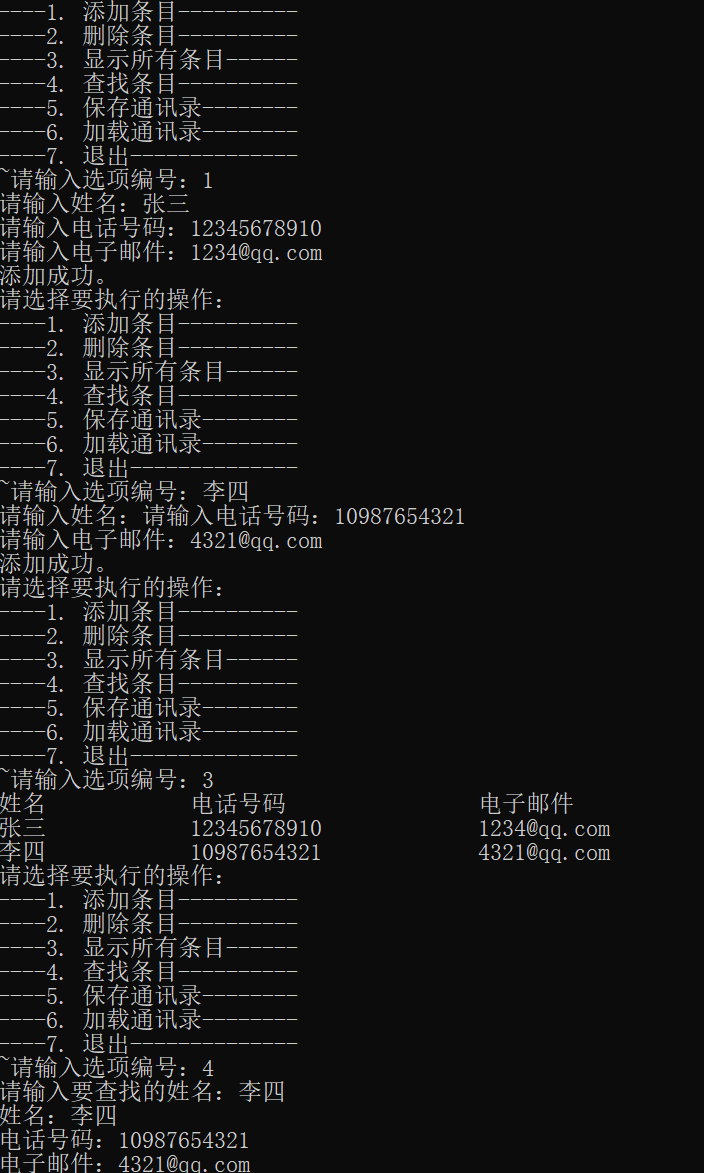
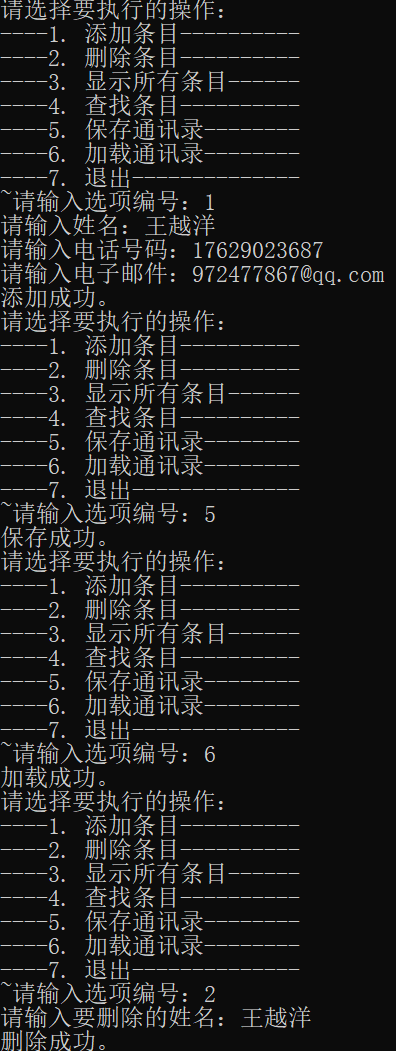
1. 函数通过fopen函数打开指定的文件，文件名由参数filename指定，并使用模式"w"表示以写入方式打开文件。
2. 如果文件打开失败，即返回的文件指针file为NULL，则打印相应的提示信息"无法打开文件。"，并使用return语句结束函数的执行。
3. 如果文件成功打开，则会继续执行下面的代码。接下来，函数使用一个循环遍历通讯录的条目，循环变量i从0递增到address\_book->num\_entries-1。
4. 通过指针获取当前条目的地址，即Entry\* entry = &address\_book->entries[i]。
5. 使用fprintf函数将条目中的姓名、电话号码和电子邮件格式化输出到文件中，格式为"%s %s %s\n"，即每个字段之间用空格分隔，每个条目占据一行。循环继续处理下一个条目，直到遍历完所有条目。
6. 使用fclose函数关闭文件，释放资源，并打印保存成功的提示信息"保存成功。"。
7. *// 保存通讯录到文件*
8. void save\_address\_book(AddressBook\* address\_book, const char\* filename) {
9. FILE\* file = fopen(filename, "w");
10. if (file == NULL) {
11. printf("无法打开文件。\n");
12. return;
13. }
14. int i;
15. for (i = 0; i < address\_book->num\_entries; i++) {
16. Entry\* entry = &address\_book->entries[i];
17. fprintf(file, "%s %s %s\n", entry->name, entry->phone, entry->email);
18. }
19. fclose(file);
20. printf("保存成功。\n");
21. }

6.从文件加载通讯录

1. 通过fopen函数打开指定的文件，文件名由参数filename指定，并使用模式"r"表示以只读方式打开文件。
2. 如果文件打开失败，即返回的文件指针file为NULL，则打印相应的提示信息"无法打开文件。"，并使用return语句结束函数的执行。
3. 如果文件成功打开，则会继续执行下面的代码。
4. 函数将通讯录的条目数量address\_book->num\_entries设置为0，以便加载数据时重新计数。然后，函数声明了一个字符数组line，用于存储从文件中读取的一行数据。这里假设每一行数据的长度不超过100个字符。
5. 使用fgets函数从文件中逐行读取数据，每次读取一行存储在line数组中，直到读取到文件末尾（fgets函数返回NULL）为止。读取的数据包含了姓名、电话号码和电子邮件，每个字段之间用空格分隔。
6. 函数声明了三个临时的字符数组name、phone和email，用于临时存储从line中解析出来的字段数据。
7. 使用sscanf函数将从line中解析出来的字段数据分别存储到对应的临时数组中，即将姓名存储到name数组，电话号码存储到phone数组，电子邮件存储到email数组。
8. 通过指针获取通讯录中下一个可用条目的地址，即Entry\* entry = &address\_book->entries[address\_book->num\_entries]。使用strncpy函数将临时数组中的数据复制到对应的条目的姓名、电话号码和电子邮件字段中，分别使用MAX\_NAME\_LENGTH、MAX\_PHONE\_LENGTH和MAX\_EMAIL\_LENGTH限制复制的最大长度。
9. 递增通讯录的条目数量address\_book->num\_entries++，表示成功加载了一个条目。
10. 循环继续读取文件中的下一行数据，直到文件末尾。最后，使用fclose函数关闭文件，释放资源，并打印加载成功的提示信息"加载成功。"。
11. *// 从文件加载通讯录*
12. void load\_address\_book(AddressBook\* address\_book, const char\* filename) {
13. FILE\* file = fopen(filename, "r");
14. if (file == NULL) {
15. printf("无法打开文件。\n");
16. return;
17. }
18. address\_book->num\_entries = 0;
19. char line[100];
20. while (fgets(line, sizeof(line), file) != NULL) {
21. char name[MAX\_NAME\_LENGTH];
22. char phone[MAX\_PHONE\_LENGTH];
23. char email[MAX\_EMAIL\_LENGTH];
24. sscanf(line, "%s %s %s", name, phone, email);
25. Entry\* entry = &address\_book->entries[address\_book->num\_entries];
26. strncpy(entry->name, name, MAX\_NAME\_LENGTH);
27. strncpy(entry->phone, phone, MAX\_PHONE\_LENGTH);
28. strncpy(entry->email, email, MAX\_EMAIL\_LENGTH);
29. address\_book->num\_entries++;
30. }
31. fclose(file);
32. printf("加载成功。\n");
33. }

7.main函数

1. 初始化了名为address\_book的结构变量，所有成员都设置为零。
2. 声明了一个字符数组filename来存储用于保存和加载地址簿的文件名。
3. 程序进入一个无限循环，用while(1)连续显示菜单并处理用户输入，直到用户选择退出。
4. 使用printf()语句显示菜单选项，并提示用户输入他们的选择。
5. 用户的输入用scanf()语句读取并存储在选择变量中。
6. 一个开关语句被用来根据用户的选择执行不同的动作。根据所选择的选项，用指向address\_book变量的指针调用相应的函数。
7. 函数add\_entry(), delete\_entry(), show\_all\_entries(), find\_entry(), save\_address\_book(), and load\_address\_book()被假定在程序的其他地方定义，以处理相应的操作。
8. 如果用户输入7作为他们的选择，程序将从main()函数返回0而退出。
9. 如果用户输入了一个无效的选项，就会显示一条信息，说明这个选择是无效的。
10. int main() {
11. AddressBook address\_book = {0};
12. char filename[] = "address\_book.txt";
13. int choice;
14. while (1) {
15. printf("请选择要执行的操作：\n");
16. printf("----1. 添加条目----------\n");
17. printf("----2. 删除条目----------\n");
18. printf("----3. 显示所有条目------\n");
19. printf("----4. 查找条目----------\n");
20. printf("----5. 保存通讯录--------\n");
21. printf("----6. 加载通讯录--------\n");
22. printf("----7. 退出--------------\n");
23. printf("~请输入选项编号：");
24. scanf("%d", &choice);
25. switch (choice) {
26. case 1:
27. add\_entry(&address\_book);
28. break;
29. case 2:
30. delete\_entry(&address\_book);
31. break;
32. case 3:
33. show\_all\_entries(&address\_book);
34. break;
35. case 4:
36. find\_entry(&address\_book);
37. break;
38. case 5:
39. save\_address\_book(&address\_book, filename);
40. break;
41. case 6:
42. load\_address\_book(&address\_book, filename);
43. break;
44. case 7:
45. return 0;
46. default:
47. printf("无效选项。\n");
48. }
49. }
50. }

（九）运行结果展示

（十）设计总结

该程序使用了适当的数据结构来组织通讯录数据，并提供了简单而实用的功能函数来管理通讯录。它还支持将通讯录数据保存到文件和从文件加载通讯录数据，以便数据的持久化和恢复。代码简洁明了，易于理解和维护。然而，该程序还可以进一步改进，例如加入输入验证、错误处理和更完善的用户界面，以提升用户体验和程序的可用性。

在设计和阅读该通讯录管理系统程序时，我从中获得了以下技能：

结构化编程：该程序采用了结构化编程的思想，将功能模块化、分解成不同的函数，使代码更加清晰和易于维护。这种模块化的设计风格有助于提高代码的可读性和可维护性。

数据结构的应用：通过使用结构体来表示通讯录条目和通讯录本身，我学会了如何将相关的数据字段组合成一个逻辑单元，并通过结构体的实例来操作和管理数据。这种数据结构的应用使得数据的组织和访问更加方便和灵活。

文件操作：程序中使用了文件操作函数，如打开文件、写入数据和读取数据等。这让我了解了如何使用文件来进行数据的持久化存储，使得通讯录数据可以在不同运行时的程序间进行传递和共享。

用户交互和控制流：程序提供了用户界面和菜单选项，通过输入选择执行不同的操作。这让我熟悉了如何与用户进行交互，并根据用户的选择来控制程序的流程和功能的执行。

错误处理和异常情况处理：在程序中，对于文件打开失败、查找不到条目等异常情况进行了处理，并给出了相应的提示信息。这让我认识到在实际开发中需要考虑异常情况，并采取适当的措施进行错误处理和用户提示。

阅读和理解他人代码：通过阅读该程序，我学会了如何理解他人编写的代码，从整体结构到具体实现逐步分析和理解每个函数的功能和作用。这对于提高自己的代码理解能力和阅读他人代码的能力非常有帮助。

总的来说，通过设计和阅读该通讯录管理系统程序，我巩固了自己的编程基础和知识，并学到了一些实用的编程技巧和设计原则。这让我对如何构建结构化、可维护和功能完善的程序有了更深入的认识和理解。