****

《数据结构与算法课程设计》

课程报告

|  |  |
| --- | --- |
| 学 院： | 信息工程学院 |
| 专 业： | 网络工程 |
| 完成人员： | 胡智伟、黄进、胡井龙 |
| 题 目： | 基于顺序表的通讯录的设计与实现 |
| 指导教师： | 李小荣 |
| 完成时间： | 2021年4月25日 |

**《数据结构与算法课程设计》任务书**

1、每位同学限选1题，并在选题表中登记，同一题目团队成员（含组长）控制在3-5人，课程设计题目不得重复选用。

2、编程可以选择不同的方式实现功能模块，但**禁止调用**非C语言或C++语言的程序代码，**严禁在网上直接抄袭代码**。

3、程序运行时，要有友好的**说明界面**和**操作提示菜单**，严禁出现“**一运行屏幕一片黑**”的情形；程序要有良好的**容错性**，当输入数据不合理或非法，程序必须能处理之并显示友好的提示信息而不能崩溃（DOS界面关闭或者无法运行）。

4、课程设计报告必须包括以下内容：

①**封面：**请按照制式模板规范填写，学院、专业、班级等名称应当按照标准名称正确表达，

②**任务书：**将本页插入报告。

③**目录：**按照报告标题内容生成目录，有分级，有页码。

④**绪论：**包括但不限于选题背景意义、编译平台等；

⑤**分析与设计：**包括但不限于分析选题的功能性需求分析、概要设计（数据结构的选择、系统结构设计及各模块划分等）、详细设计（程序流程图、算法设计等）。

⑥**团队协作**：围绕选题的功能模块，明确团队成员个体的分工任务，以及成员之间的协作内容（协作部分要与团队交流会议相一致）。

⑦**实现**：给出各功能模块的实现源代码（注意只写与本功能紧密相关的**核心代码**，完整代码应放在附录中）。源代码必须**正确缩进**，关键性代码（各函数及参数的说明、关键变量及语句的意义等）必须辅以清晰的**中文注释**；以截图和语言描述的形式描述程序输入的结果，对于需要输入数据的，必须给出覆盖功能实现的测试数据（尽可能有异常输入数据的检测和相应处理）。

⑧**总结**：课程设计的收获、遇到的具体问题及解决过程的思考（**严禁套话**）、不足与展望、对C语言程序设计课程的知识应用到相关工程实践的思考等内容；既要体现成员独立的思考，也要由团队整体的反思。

⑨**参考文献**：不少于5篇（含教材、期刊等）。

⑩**附录**：本选题的完整源代码（代码必须正确缩进）、操作说明（函数功能描述）等。

5、为做好存档工作，课程设计报告（含附录代码），一律以电子档案形式保存，其中，报告以WORD文档或者PDF文档，并以选题名称作为文档名（需要签名、盖章的扫描插入对应位置）；源代码需要给出.cpp或者.c文件，以便指导教师验证，文件名同报告文档名；每个团队的上述两类材料打包成压缩文件，连同其他过程材料在答辩、修改完成后由班级学委统一收集，并向指导教师提交。

**指导教师（签名）**

**年 月 日**

**通讯录的制作**

**摘 要**

随着社会的开展,人们生活的步调日益加快,越来越多的人参加了全球化的世界。人们不再拘泥于自己的一块小天地,随时随地能沟通是生活中非常重要的一件事,于是通讯录成为每个人必不可少的工具。所以编写通讯录管理系统,旨在帮助人们更方便联系自己的朋友。

此通讯录管理系统具有通讯录的录入、通讯者结点的插入、通讯者的查询、通讯者的删除、通讯录链表的输出、退出通讯录管理系统等功能。使用数据结构编写,运用大量的函数实现通讯录的功能。让使用者进行简单的操作即可以实现通讯录管理系统的根本功能,丰富并方便了人们的生活。

通过这次课程设计使我们懂得了理论与实际相结合是很重要的,只有理论知识是远远不够的,只有把所学到的理论知识与实践相结合起来,从理论中得出结论,将结论辅助与理论,才能真正学到知识并写出有用的程序来为他人效劳,更为社会效劳,所以提高自己的实际动手能力和独立思考的能力是十分重要的。同时通过这次设计,也让我体验到了编程的的乐趣。

**关键词：**数据结构；通讯录管理系统；函数

**目录**

[第1章 绪 论 1](#_Toc17413)

[1.1 项目背景 1](#_Toc1049)

[1.2 开发目的和意义 1](#_Toc21542)

[1.3 开发技术 2](#_Toc21971)

[第2章 团队协作 3](#_Toc21310)

[2.1 团队组成 3](#_Toc3036)

[2.2 团队分工 3](#_Toc26325)

[第3章 系统分析与设计 4](#_Toc16673)

[3.1 系统需求分析 4](#_Toc17283)

[3.2 系统概要设计 4](#_Toc20406)

[3.2.1 系统结构图 4](#_Toc23431)

[3.3 系统详细设计 5](#_Toc10121)

[3.3.1 菜单界面 5](#_Toc13071)

[3.3.2 录入联系人信息 5](#_Toc18442)

[3.3.3 查找联系人信息 6](#_Toc4067)

[第4章 系统实现 8](#_Toc23465)

[4.1 录入（完成人：胡井龙） 8](#_Toc10068)

[4.2 显示（完成人：胡井龙） 10](#_Toc11409)

[4.3 查找（完成人：黄进） 12](#_Toc17554)

[4.4 删除（完成人：黄进） 14](#_Toc9238)

[4.5 排序（完成人：胡智伟） 16](#_Toc29409)

[4.6 统计（完成人：胡智伟） 18](#_Toc21161)

[第5章 系统测试 20](#_Toc21059)

[5.1 录入功能测试 20](#_Toc11344)

[5.2 查询功能测试 21](#_Toc12937)

[5.3 删除功能测试 22](#_Toc17904)

[5.4 修改功能测试 23](#_Toc12275)

[5.5 排序功能测试 24](#_Toc501)

[5.6 统计功能测试 25](#_Toc20242)

[第6章 总结 26](#_Toc29236)

[6.1 团队完成情况总结 26](#_Toc26187)

[6.2 个人完成情况总结 26](#_Toc23775)

[6.3 不足与下一步改进 26](#_Toc3338)

[6.4 团队评价 27](#_Toc23429)

[参考文献 28](#_Toc17170)

[附 录 29](#_Toc929)

# 第1章 绪 论

## 1.1 项目背景

当今时代是飞速发展的信息时代。在各行各业中离不开信息处理，这正是计算机被广泛应用于信息管理系统的环境。计算机的最大好处在于利用它能够进行信息管理。使用计算机进行信息控制，不仅提高了工作效率，而且大大的提高了其安全性。

尤其对于复杂的信息管理，计算机能够充分发挥它的优越性。计算机进行信息管理与信息管理系统的开发密切相关，系统的开发是系统管理的前提。本系统就是为了管理而设计的。

## 1.2 开发目的和意义

通讯录管理系统的建立是可以有效节省同学、朋友、客户间日常交流的投资(指时间、精力)，其设计理念很简单，变传统的纸张与笔录方式为计算机自动化进行通讯录信息管理系统，既省时间又提高效率。因此在系统构建时，只要我们本着丰富通讯录的信息资源，为用户提供最为方便的快捷的信息阅览途径和管理手段，去吸引用户，那么通讯录管理系统是完全可行的。

通讯录软件设计的灵感来源于生活和工作中的需要。如今，随着社会的飞速发展，信息时代改变着人们的各种生活方式。人们的联系信息，联系方式变得复杂而多样化，以前所使用的各种电话簿、通讯本等小册子由于查找不方便、功能单一等缺陷已经无法胜任它的“时代使命”，而现在各种手机，商务通内设的电话簿尽管携带方便却又挥之不去其“记录量少，界面小，浏览不方便”的缺点。工作中看到有些人巧妙地利用excell或者word制表格来建立通讯录，每逢用时再打开，可是查找极其不便，维护起来也麻烦。

通讯录系统设计，它的内容对于计算机来说是至关重要的，通讯录系统为计算机的使用者提供充足的信息和快捷的查询手段。用c语言构建的通讯录系统设计，通过课上学到的和查阅一些文献的一些关于结构体、数组、指针、函数以及循环函数的运用和字符串的处理等基本知识可以初步的实现通讯录的输入、显示、查找、删除、快速查询等通讯录简单而用的一些实用的功能，给人们带来更多的方便。本管理系统设计合理、操作方便、运行稳定、功能完备，具有较高的实用价值。

## 1.3 开发技术

本课程设计主要运用C语言中的结构体、数组、链表、等数据结构，设计一个简单的管理系统应用程序。本程序设计出了通讯录管理系统的基本功能，并设计了简单的界面，主要考察对自定义函数的熟悉程度，使用的是数组的相关操作，包括结构体数组的输入、输出、查找、删除等。通过完成本课题进一步熟悉C语言基本知识，并掌握数据结构的一些基本算法思想，进一步熟悉指针的用法，数组的建立运用和函数调用，加深对数据结构的理解，提高算法设计的能力，锻炼编程的能力。用C语言编程的通讯录管理系统，要求实现通讯录系统中，添加新记录、删除记录、显示记录信息、按名字查找信息和退出系统等功能。

# **第2章 团队协作**

## 2.1 团队组成

本团队共由三名成员组成，分别是：

组长：胡智伟

组员：胡井龙

组员：黄进

## 2.2 团队分工

（1）插入，排序，统计和清空（完成人：胡智伟）

功能模块简介：实现联系人的插入操作；实现联系人的插入功能，插入联系人在想放入的位置；实现联系人的排序统计操作，便于观察。实现联系人清空操作。

（2）查找，删除和修改（完成人：黄进）

功能模块简介：实现联系人的信息查找，根据输入联系人姓名来输出联系人的信息；实现联系人的删除操作，并实现对联系人的信息进行修改。

（3）菜单，录入和显示（完成人：胡井龙）

功能模块简介：实现菜单功能，可了解本系统所拥有的全部功能；实现联系人的信息录入，可根据编入信息的相关情况进行相对应的录入操作；实现联系人的显示功能。

# 第3章 系统分析与设计

## 3.1 系统需求分析

1、录入人员的编号、姓名、个人邮箱、地址、联系电话。

2、显示所有人员的信息。

3、通过输入姓名查找人员信息。

4、通过输入姓名查找到要删除的人员信息，然后可以进行删除，回显当前通讯录信息。

5、通过输入姓名查找到要修改的人员信息，然后依次进行信息修改。

6、添加人员信息。

7、对通讯录进行排序

8、对通讯录进行统计

## 3.2 系统概要设计

### 3.2.1 系统结构图

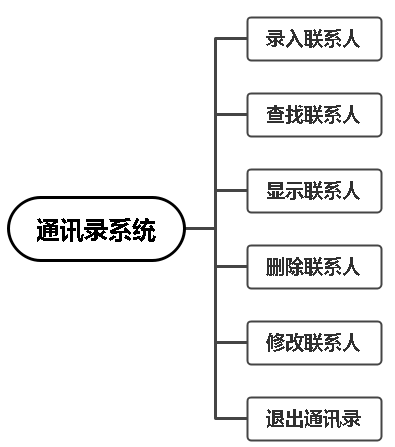


图 1 系统结构图

## 3.3 系统详细设计

### 3.3.1 菜单界面

这是通讯录系统的菜单界面，通过switch语句选择想要使用的功能，输入功能前面的序号，进入对应功能界面

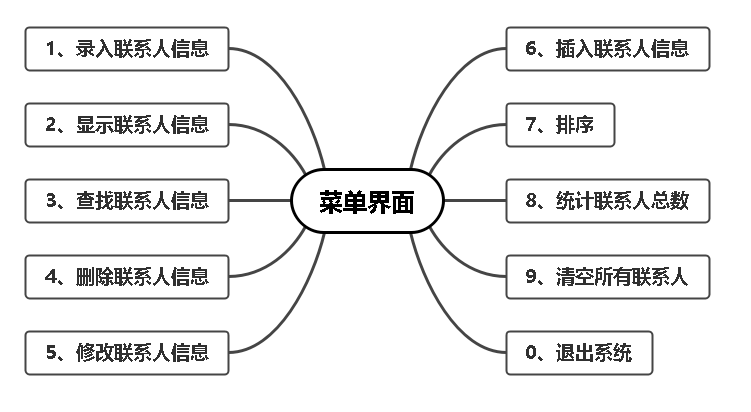


图 2 菜单界面

### 3.3.2 录入联系人信息

由菜单界面进入录入录入联系人信息的界面，依次输入联系人的姓名、性别、年龄、电话号码、家庭住址和工作单位

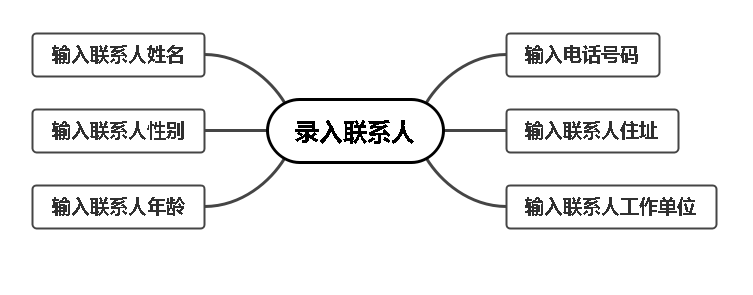


图 3 录入联系人结构

### 3.3.3 查找联系人信息

通过输入联系人的姓名进入系统进行查找联系人的信息，若系统中有该联系人，则显示该联系人的信息；若系统中无该联系人，则显示查无此人。流程图如图4所示

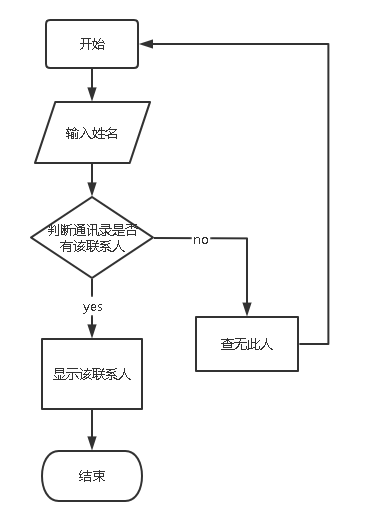


图 4 插入联系人流程图

**3.3.4 修改联系人信息**

通过输入联系人的姓名进入系统进行查找联系人的信息，若系统中有该联系人，则删除该联系人的信息，并显示删除成功；若系统中无该联系人，则显示查无此人。流程图如图5所示

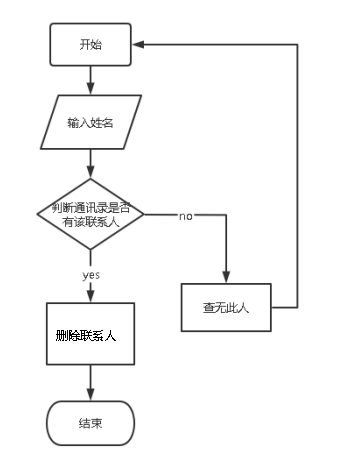


图 5 删除联系人流程图

# 第4章 系统实现

## 4.1 录入（完成人：胡井龙）

功能模块简介：由菜单界面进入录入录入联系人信息的界面，依次输入联系人的姓名、性别、年龄、电话号码、家庭住址和工作单位

核心程序：

void addPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

if (abs->m\_Size == MAX)

{

printf("通讯录已满，无法添加！\n");

return;

}

else

{

char name[20];

printf("请输入姓名：");

scanf("%s", name);

strcpy(abs->personArray[abs->m\_Size].m\_name, name);

printf("\n");

printf("请输入性别：\n");

printf("1---男\n");

printf("2---女\n");

int sex = 0;

while (1)

{

scanf("%d", &sex);

if (sex == 1 || sex == 2)

{

abs->personArray[abs->m\_Size].m\_sex = sex;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

printf("请输入年龄：");

int age = 0;

while (1)

{

scanf("%d", &age);

if (age < 120 && age>3)

{

abs->personArray[abs->m\_Size].m\_age = age;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

printf("请输入联系电话：");

char phone[20];

scanf("%s", phone);

strcpy(abs->personArray[abs->m\_Size].m\_phone, phone);

printf("\n");

printf("请输入住址：");

char address[50];

scanf("%s", address);

strcpy(abs->personArray[abs->m\_Size].m\_addr, address);

printf("\n");

printf("请输入工作单位：");

char job[30];

scanf("%s", job);

strcpy(abs->personArray[abs->m\_Size].m\_job, job);

printf("\n");

abs->m\_Size++;

printf("添加成功\n");

system("pause");

system("cls");//清屏

}

**界面展示**



图 6 录入联系人

## 4.2 显示（完成人：胡井龙）

功能模块简介：实现联系人信息的显示，将结构体中的联系人通过遍历依次显示到控制台

核心程序：

void showPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

if (abs->m\_Size == 0)

{

printf("当前记录为空\n");

}

else

{

for (int i = 0; i < abs->m\_Size; i++)

{

printf("姓名：");

printf("%s", abs->personArray[i].m\_name);

printf("\t性别：");

printf("%s", abs->personArray[i].m\_sex == 1 ? "男" : "女");

printf("\t年龄：");

printf("%d", abs->personArray[i].m\_age);

printf("\t电话：");

printf("%s", abs->personArray[i].m\_phone);

printf("\t住址：");

printf("%s", abs->personArray[i].m\_addr);

printf("\t工作地址：");

printf("%s", abs->personArray[i].m\_job);

printf("\n");

}

}

system("pause");

system("cls");

}

**界面展示**



图 7 显示联系人

## 4.3 查找（完成人：黄进）

功能模块简介：查询联系人信息，通过输入联系人的姓名，进入系统进行查找联系人，若系统存有该联系人，则显示该联系人的信息，若系统中无该联系人，这显示查无此人并重新输入姓名

核心程序：

void findPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

printf("请输入您要查找的联系人：");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name,name) == 0)

{

ret = i;

}

}

if (ret != -1)

{

printf("姓名：%s", abs->personArray[ret].m\_name);

printf("\t性别：%s", abs->personArray[ret].m\_sex == 1 ? "男" : "女");

printf("\t年龄：%d", abs->personArray[ret].m\_age);

printf("\t电话：%s", abs->personArray[ret].m\_phone);

printf("\t住址：%s", abs->personArray[ret].m\_addr);

printf("\t工作地址：%s", abs->personArray[ret].m\_job);

printf("\n");

}

else

{

printf("查无此人\n");

}

system("pause");

system("cls");

**界面展示**



图 8 查找联系人

**4.4 删除（完成人：黄进）**

功能简介：实现了联系人信息的删除，通过输入联系人的姓名，进入系统进行查找联系人，若系统存有该联系人，则删除该联系人的信息，若系统中无该联系人，这显示查无此人并重新输入姓名

核心代码：

void deletePerson(struct Addressbooks\* abs){

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

printf("请输入删除的联系人\n");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name, name) == 0)

{

ret = i;

}

}

if (ret != -1)

{

for (int i = ret; i < abs->m\_Size; i++)

{

abs->personArray[i] = abs->personArray[i + 1];

}

abs->m\_Size--;//更新通讯录中的人员数

printf("删除成功\n");

abs->m\_Size--;

}

else

{

printf("查无此人\n");

}

system("pause");

system("cls");

}

**界面显示：**



图 9 删除联系人

**4.5 排序（完成人：胡智伟）**

功能模块简介：实现对联系人信息的排序，进入功能有两种排序可供选择，输入1为性别排序，男生在前面女生在后面，输入2为年龄排序，按照年龄的大小由小到大排序，代码方面用冒泡排序进行排序

核心代码**：**

void sortPerson(struct Addressbooks\* abs)

{

THER:

printf("请选择如何排序：\n");

printf("1---性别\n");

printf("2---年龄\n");

int ret = 0;

scanf("%d", &ret);

if (ret = 1)

{

//冒泡排序

for (int i = 0; i < abs->m\_Size - 1; i++)

{

for (int j = 0; j < abs->m\_Size - i - 1; j++)

{

if (abs->personArray[j].m\_sex > abs->personArray[j + 1].m\_sex)

{

struct Person temp = abs->personArray[j];

abs->personArray[j] = abs->personArray[j + 1];

abs->personArray[j + 1] = temp;

}

}

}

//排序完成后显示所有联系人

showPerson(abs);

}

else if (ret == 2)

{

//冒泡排序

for (int i = 0; i < abs->m\_Size - 1; i++)

{

for (int j = 0; j < abs->m\_Size - i - 1; j++)

{

if (abs->personArray[j].m\_age > abs->personArray[j + 1].m\_age)

{

struct Person temp = abs->personArray[j];

abs->personArray[j] = abs->personArray[j + 1];

abs->personArray[j + 1] = temp;

}

}

}

//排序完成后显示所有联系人

showPerson(abs);

}

else

{

printf("输入错误，请重新输入！\n");

goto THER;

}

system("pause");

system("cls");

}

**界面设计**



图 10 排序

**4.6 统计（完成人：胡智伟）**

功能模块简介：实现对联系人信息的统计，统计系统中联系人的总数，并选择是否显示所有联系人，y即yes打印所有联系人信息，n即no返回主界面

核心代码：

void totalPerson(struct Addressbooks\* abs)

{

if (abs->m\_Size == 0)

{

printf("通讯录无联系人\n");

}

else

{

printf("您一个有%d位联系人\n", abs->m\_Size);

THERE:

printf("是否打印所有人：(y/n)\n");

printf("y---Yes\n");

printf("n---No\n");

char yn[10];

scanf("%s", yn);

if (strcmp(yn, "y") == 0)

{

showPerson(abs); //打印所有联系人

}

else if (strcmp(yn, "n") == 0)

{

system("pause");

}

else

{

printf("输入错误，请重新输入\n");

goto THERE;

}

}

system("pause");

system("cls");

}

**界面设计**



图 11 统计

# 第5章 系统测试

## 5.1 录入功能测试

测试场景：联系人的录入

测试用例：依次输入联系人的姓名、性别、年龄、联系电话、家庭住址和工作单位

测试结果：联系人添加成功，返回主界面



图 12 录入功能测试

## 

## 5.2 查询功能测试

测试场景：联系人的查询

测试用例：输入想要查询联系人的姓名，先输入一个通讯录不存在的姓名，再输入一个通讯录中存在的姓名

测试结果：先输入的姓名系统显示查无此人，第二次输入姓名后系统显示该联系人的信息



图 13 查询功能测试

## 5.3 删除功能测试

测试场景：删除联系人

测试用例：输入删除联系人的姓名，先输入一个通讯录不存在的姓名，再输入一个通讯录中存在的姓名

测试结果：先输入的姓名系统显示查无此人，第二次输入姓名后系统显示联系人删除成功



图 14 删除功能测试

## 5.4 修改功能测试

测试场景：修改联系人

测试用例：输入联系人姓名对联系人的信息进行修改，先输入一个通讯录不存在的姓名，再输入一个通讯录中存在的姓名

测试结果：先输入的姓名系统显示查无此人，第二次输入姓名后系统显示请输入联系人的信息，依次输入联系人的姓名、性别、年龄、电话号码、家庭住址和工作单位



图 15 修改功能测试

## 5.5 排序功能测试

测试场景：对联系信息进行排序

测试用例：排序功能有两种排序方式，一种是按照性别排序，一种是按年龄排序，分别测试两种排序方式

测试结果：性别排序按男女依次排序，年龄排序按幼长依次排序

# 

图 16 排序功能测试

## 5.6 统计功能测试

测试场景：对通讯录中的联系人进行统计

测试用例：进入统计功能，依次选择打印联系人的选项和返回主界面的选项

测试结果：系统显示通讯录中的联系人个数，选择打印联系人系统显示所有联系人信息，选择退出系统返回系统主界面

# 

图 17 统计功能测试

# 第6章 总结

## 6.1 团队完成情况总结

握了一些基本的C语言进行程序设计的技巧，更深的理解和运用结构化程序设计的思想和方法，掌握开发一个小型实用系统的基本方法，同时学会了一些调试一个较长程序的基本方法，提高了书写程序设计开发文档的能力（书写课程设计报告）。

## 6.2 个人完成情况总结

每个人都尽力完成了自己所需要的完成的模块，对功能的使用正常运行等提供的完整的思路，遇到的问题大多是知识储备不足，对问题了解不够深刻，比如在需求分析上面，对系统本身要完成的功能没有深刻的认识，导致系统做出来没有连贯性与实用性。同时，通过这个系统的设计，我们也都学习到很多，比如对工程的流程的深刻理解，对需求分析的重要性的重新认识，对实施前的准备工作的了解是非常重要的，不然就会导致做出来的系统与实际不相符合。不足之处：大家对数据结构的理解不够深刻，很多知识都是通过复习巩固实现的。

## 6.3 不足与下一步改进

当前系统并不是很完善，只能实现基本的功能，还需要根据实际情况进行修改，例如：对不同的用户设置不同的管理权限，使系统更加完善，使用户的使用体验感更好。

虽然完成了这次课程设计，我们对C语言以及数据结构的认识、对整体项目的了解也只是在一个基础层面上，以后我们还需要深入学习数据结构。

## 6.4 团队评价

表 1 胡智伟团队评价

|  |  |  |
| --- | --- | --- |
| 组员姓名 | 评价理由 | 评价结果 |
| 黄进 | 积极和组员交流并解决问题 | 良 |
| 胡智伟 | 工作责任强，工作时细心积极 | 良 |
| 胡井龙 | 学习能力强，基础知识扎实 | 良 |

表 2 黄进团队评价

|  |  |  |
| --- | --- | --- |
| 组员姓名 | 评价理由 | 评价结果 |
| 黄进 | 积极主动，配合组长完成任务 | 良 |
| 胡智伟 | 工作责任心强，积极组织团队完成设计 | 良 |
| 胡井龙 | 工作认真，有团队奉献精神 | 良 |

表 3 胡井龙团队评价

|  |  |  |
| --- | --- | --- |
| 组员姓名 | 评价理由 | 评价结果 |
| 黄进 | 做事有条理，责任感强 | 良 |
| 胡井龙 | 积极配合并完成组长分配的任务 | 良 |
| 胡智伟 | 有较强的组织能力和动手能力，做事认真 | 良 |

# 

# 参考文献

[1] 严蔚敏,吴伟民.数据结构C语言版[M].清华大学出版社，2019年.

[2] 范德宝，于晓聪，丁伟祥.提高数据结构课程教学效果的探讨[J].黑龙江科技信息，2020年.

[3]严太山，郭观七，李文彬.课堂设问的技巧及其在《数据结构》课程教学中的应用[J].湖南理工学院学报：自然科学版，2020(1):81-83.

[4]托马斯·科尔曼、查尔斯·雷瑟尔森、[罗纳德·李维斯特](https://baike.baidu.com/item/%E7%BD%97%E7%BA%B3%E5%BE%B7%C2%B7%E6%9D%8E%E7%BB%B4%E6%96%AF%E7%89%B9/700199" \t "https://baike.baidu.com/item/%E7%AE%97%E6%B3%95%E5%AF%BC%E8%AE%BA/_blank)、克利福德·斯坦.算法导论.机器工业出版社，2018年.

[5]Stephen Prata.C Primer Plus（第6版）,人民邮电出版社,2019年.

[6]逯鹏，张赞.数据结构课程教学方法的研究和实践[J].教育教学论坛，2018(18):121-123.

# 附 录

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#include<assert.h>

#define MAX 100//通讯录最大人数

//菜单

void menu();

//添加联系人

void addPerson(struct Addressbooks\* abs);

//显示联系人

void showPerson(struct Addressbooks\* abs);

//查找联系人

void findPerson(struct Addressbooks\* abs);

//删除联系人

void deletePerson(struct Addressbooks\* abs);

//修改联系人

void modifyPerson(struct Addressbooks\* abs);

//统计联系人

void totalPerson(struct Addressbooks\* abs);

//排序

void sortPerson(struct Addressbooks\* abs);

//插入联系人

void inSertPerson(struct Addressbooks\* abs);

//清空联系人

void cleanPerson(struct Addressbooks\* abs);

//联系人结构体

struct Person

{

//姓名

char m\_name[20];

//性别

int m\_sex;//1、男 2、女

//年龄

int m\_age;

//电话

char m\_phone[20];

//住址

char m\_addr[50];

//工作单位

char m\_job[30];

};

//设计通讯录结构体

struct Addressbooks

{

//通讯录中保存的联系人数组

struct Person personArray[MAX];

//通讯录中当前记录联系人个数

int m\_Size;

};

//菜单

void menu()

{

system("cls");

printf("\n\n\n");

printf("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*通讯录管理系统\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\t\t======================================\n");

printf("\t\t\t\t\t\* 1.录入联系人信息 \*\n");

printf("\t\t\t\t\t\* 2.显示所有联系人 \*\n");

printf("\t\t\t\t\t\* 3.查找联系人信息 \*\n");

printf("\t\t\t\t\t\* 4.删除联系人信息 \*\n");

printf("\t\t\t\t\t\* 5.修改联系人信息 \*\n");

printf("\t\t\t\t\t\* 6.插入联系人信息 \*\n");

printf("\t\t\t\t\t\* 7.排序 \*\n");

printf("\t\t\t\t\t\* 8.统计联系人总数 \*\n");

printf("\t\t\t\t\t\* 9.清空所有联系人 \*\n");

printf("\t\t\t\t\t\* 0.退出系统 \*\n");

printf("\t\t\t\t\t======================================\n");

printf("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t\t\t 请选择(0-9):");

}

//添加联系人

void addPerson(struct Addressbooks\* abs)

{

char ch[2];

FILE\* fp;

if ((fp = fopen("data.txt", "a+")) == NULL)

{

printf("\t\t 文件不存在! \n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t 文件中没有记录!\n");

}

else

{

show();

}

if ((fp = fopen("data.txt", "a+")) == NULL)

{

printf("\t\t 文件不存在!\n");

return;

}

//判断通讯录是否已满，如果满了就不在添加

if (abs->m\_Size == MAX)

{

printf("通讯录已满，无法添加！\n");

return;

}

else

{

//添加具体的联系人

//姓名

char name[20];

printf("请输入姓名：");

scanf("%s", name);

strcpy(abs->personArray[abs->m\_Size].m\_name, name);

//abs->personArray[abs->m\_Size].m\_name = name;

printf("\n");

//性别

printf("请输入性别：\n");

printf("1---男\n");

printf("2---女\n");

int sex = 0;

while (1)

{

//如果输入的是1或是2可以退出循环，因为输入的是正确值

//如果输入有误，重新输入

scanf("%d", &sex);

if (sex == 1 || sex == 2)

{

abs->personArray[abs->m\_Size].m\_sex = sex;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//年龄

printf("请输入年龄：");

int age = 0;

while (1)

{

scanf("%d", &age);

if (age < 120 && age>3)

{

abs->personArray[abs->m\_Size].m\_age = age;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//电话

printf("请输入联系电话：");

char phone[20];

scanf("%s", phone);

strcpy(abs->personArray[abs->m\_Size].m\_phone, phone);

//abs->personArray[abs->m\_Size].m\_phone = phone;

printf("\n");

//住址

printf("请输入住址：");

char address[50];

scanf("%s", address);

strcpy(abs->personArray[abs->m\_Size].m\_addr, address);

//abs->personArray[abs->m\_Size].m\_addr = address;

printf("\n");

//工作单位

printf("请输入工作单位：");

char job[30];

scanf("%s", job);

strcpy(abs->personArray[abs->m\_Size].m\_job, job);

//abs->personArray[abs->m\_Size].m\_job = job;

printf("\n");

//更新通讯录人数

abs->m\_Size++;

printf("添加成功\n");

system("pause");

system("cls");//清屏

}

}

//显示联系人

void showPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

//判断通讯录中的人数是否为0，如果为0，提示记录为空

//如果不为0，显示记录的联系人信息

if (abs->m\_Size == 0)

{

printf("当前记录为空\n");

}

else

{

for (int i = 0; i < abs->m\_Size; i++)

{

printf("姓名：");

printf("%s", abs->personArray[i].m\_name);

printf("\t性别：");

printf("%s", abs->personArray[i].m\_sex == 1 ? "男" : "女");

printf("\t年龄：");

printf("%d", abs->personArray[i].m\_age);

printf("\t电话：");

printf("%s", abs->personArray[i].m\_phone);

printf("\t住址：");

printf("%s", abs->personArray[i].m\_addr);

printf("\t工作地址：");

printf("%s", abs->personArray[i].m\_job);

printf("\n");

}

}

system("pause");

system("cls");

}

//查找联系人

void findPerson(struct Addressbooks\* abs)

{

printf("请输入您要查找的联系人：");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name,name) == 0)

{

ret = i;

}

}

//判断

if (ret != -1)

{

printf("姓名：%s", abs->personArray[ret].m\_name);

printf("\t性别：%s", abs->personArray[ret].m\_sex == 1 ? "男" : "女");

printf("\t年龄：%d", abs->personArray[ret].m\_age);

printf("\t电话：%s", abs->personArray[ret].m\_phone);

printf("\t住址：%s", abs->personArray[ret].m\_addr);

printf("\t工作地址：%s", abs->personArray[ret].m\_job);

printf("\n");

}

else

{

printf("查无此人\n");

}

system("pause");

system("cls");

}

//删除联系人

void deletePerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

printf("请输入删除的联系人\n");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name, name) == 0)

{

ret = i;

}

}

if (ret != -1)

{

//查找到人，要进行删除操作

for (int i = ret; i < abs->m\_Size; i++)

{

//数据前移

abs->personArray[i] = abs->personArray[i + 1];

}

abs->m\_Size--;//更新通讯录中的人员数

printf("删除成功\n");

//更新联系人

abs->m\_Size--;

}

else

{

printf("查无此人\n");

}

system("pause");

system("cls");

}

//修改联系人

void modifyPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

printf("请输入您要修改的联系人：");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name, name) == 0)

{

ret = i;

}

}

if (ret != -1)

{

//输入修改联系人的信息

//姓名

char name[20];

printf("请输入姓名：");

scanf("%s", name);

strcpy(abs->personArray[ret].m\_name, name);

//abs->personArray[abs->m\_Size].m\_name = name;

printf("\n");

//性别

printf("请输入性别：\n");

printf("1---男\n");

printf("2---女\n");

int sex = 0;

while (1)

{

//如果输入的是1或是2可以退出循环，因为输入的是正确值

//如果输入有误，重新输入

scanf("%d", &sex);

if (sex == 1 || sex == 2)

{

abs->personArray[ret].m\_sex = sex;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//年龄

printf("请输入年龄：");

int age = 0;

while (1)

{

scanf("%d", &age);

if (age < 120 && age>3)

{

abs->personArray[ret].m\_age = age;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//电话

printf("请输入联系电话：");

char phone[20];

scanf("%s", phone);

strcpy(abs->personArray[ret].m\_phone, phone);

//abs->personArray[abs->m\_Size].m\_phone = phone;

printf("\n");

//住址

printf("请输入住址：");

char address[50];

scanf("%s", address);

strcpy(abs->personArray[ret].m\_addr, address);

//abs->personArray[abs->m\_Size].m\_addr = address;

printf("\n");

//工作单位

printf("请输入工作单位：");

char job[30];

scanf("%s", job);

strcpy(abs->personArray[ret].m\_job, job);

//abs->personArray[abs->m\_Size].m\_job = job;

printf("\n");

printf("修改成功\n");

}

else

{

printf("查无此人\n");

}

system("pause");

system("cls");//清屏

}

//统计联系人

void totalPerson(struct Addressbooks\* abs)

{

if (abs->m\_Size == 0)

{

printf("通讯录无联系人\n");

}

else

{

printf("您一个有%d位联系人\n", abs->m\_Size);

THERE:

printf("是否打印所有人：(y/n)\n");

printf("y---Yes\n");

printf("n---No\n");

char yn[10];

scanf("%s", yn);

if (strcmp(yn, "y") == 0)

{

showPerson(abs); //打印所有联系人

}

else if (strcmp(yn, "n") == 0)

{

system("pause");

}

else

{

printf("输入错误，请重新输入\n");

goto THERE;

}

}

system("pause");

system("cls");

}

//插入联系人

void inSertPerson(struct Addressbooks\* abs)

{

FILE\* fp;

if ((fp = fopen("data.txt", "r+")) == NULL)

{

printf("\t\t \n文件不存在\n");

return;

}

while (!feof(fp))

{

if (fread(&a[m], LEN, 1, fp) == 1)

m++;

}

fclose(fp);

if (m == 0)

{

printf("\t\t \n文件没有记录!\n");

return;

}

printf("请输入您想添加的联系人在另一联系人前面的名字：");

char name[20];

scanf("%s", name);

int ret = -1;

//查询联系人的位置

for (int i = 0; i < abs->m\_Size; i++)

{

if (strcmp(abs->personArray[i].m\_name, name) == 0)

{

ret = i;

}

}

if (ret != -1)

{

for (int j = abs->m\_Size - 1; j >= ret; j--)

{

//元素后移

abs->personArray[j + 1] = abs->personArray[j];

}

//输入所要插入联系人的信息

//姓名

char name[20];

printf("请输入姓名：");

scanf("%s", name);

strcpy(abs->personArray[ret].m\_name, name);

//abs->personArray[abs->m\_Size].m\_name = name;

printf("\n");

//性别

printf("请输入性别：\n");

printf("1---男\n");

printf("2---女\n");

int sex = 0;

while (1)

{

//如果输入的是1或是2可以退出循环，因为输入的是正确值

//如果输入有误，重新输入

scanf("%d", &sex);

if (sex == 1 || sex == 2)

{

abs->personArray[ret].m\_sex = sex;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//年龄

printf("请输入年龄：");

int age = 0;

while (1)

{

scanf("%d", &age);

//判断年龄是否合法

if (age < 120 && age>3)

{

abs->personArray[ret].m\_age = age;

break;

}

printf("输入有误，请重新输入\n");

}

printf("\n");

//电话

printf("请输入联系电话：");

char phone[20];

scanf("%s", phone);

strcpy(abs->personArray[ret].m\_phone, phone);

//abs->personArray[abs->m\_Size].m\_phone = phone;

printf("\n");

//住址

printf("请输入住址：");

char address[50];

scanf("%s", address);

strcpy(abs->personArray[ret].m\_addr, address);

//abs->personArray[abs->m\_Size].m\_addr = address;

printf("\n");

//工作单位

printf("请输入工作单位：");

char job[30];

scanf("%s", job);

strcpy(abs->personArray[ret].m\_job, job);

//abs->personArray[abs->m\_Size].m\_job = job;

printf("\n");

abs->m\_Size++;

printf("插入成功\n");

}

else

{

printf("查无此人！\n");

}

system("pause");

system("cls");

}

//排序

void sortPerson(struct Addressbooks\* abs)

{

THER:

printf("请选择如何排序：\n");

printf("1---性别\n");

printf("2---年龄\n");

int ret = 0;

scanf("%d", &ret);

if (ret = 1)

{

//冒泡排序

for (int i = 0; i < abs->m\_Size - 1; i++)

{

for (int j = 0; j < abs->m\_Size - i - 1; j++)

{

if (abs->personArray[j].m\_sex > abs->personArray[j + 1].m\_sex)

{

struct Person temp = abs->personArray[j];

abs->personArray[j] = abs->personArray[j + 1];

abs->personArray[j + 1] = temp;

}

}

}

//排序完成后显示所有联系人

showPerson(abs);

}

else if (ret == 2)

{

//冒泡排序

for (int i = 0; i < abs->m\_Size - 1; i++)

{

for (int j = 0; j < abs->m\_Size - i - 1; j++)

{

if (abs->personArray[j].m\_age > abs->personArray[j + 1].m\_age)

{

struct Person temp = abs->personArray[j];

abs->personArray[j] = abs->personArray[j + 1];

abs->personArray[j + 1] = temp;

}

}

}

//排序完成后显示所有联系人

showPerson(abs);

}

else

{

printf("输入错误，请重新输入！\n");

goto THER;

}

system("pause");

system("cls");

}

//清空联系人

void cleanPerson(struct Addressbooks\* abs)

{

int a = 0;

printf("是否清空通讯录\n");

printf("1--确认\n");

printf("2--取消\n");

while (1)

{

scanf("%d", &a);

if (a == 1)

{

//逻辑清零

abs->m\_Size = 0;

printf("通讯录已清空\n");

break;

}

else if (a == 2)

{

break;

}

else

{

printf("选择错误，请重新选择\n");

}

}

system("pause");

system("cls");

}

int main()

{

//创建通讯录结构体变量

struct Addressbooks abs;

//初始化通讯录中当前人员个数

abs.m\_Size = 0;

//选择

int select = 0;

while (1)

{

//菜单调用

menu();

scanf\_s("%d", &select);

switch (select)

{

case 1:

addPerson(&abs); //添加联系人

break;

case 2:

showPerson(&abs); //显示联系人

break;

case 3:

findPerson(&abs); //查找联系人

break;

case 4:

deletePerson(&abs); //删除联系人

break;

case 5:

modifyPerson(&abs); //修改联系人

break;

case 6:

inSertPerson(&abs); //插入联系人

break;

case 7:

sortPerson(&abs); //排序

break;

case 8:

totalPerson(&abs); //统计联系人

break;

case 9:

cleanPerson(&abs); //清空联系人

break;

case 0: //输入的是0，退出系统

printf("欢迎下次使用！\n");

system("pause");

return 0;

break;

default:

printf("选择错误，请重新选择\n");

break;

}

}

system("pause");

return 0;

}