

**《程序设计基础课程设计》**

**项目报告**

|  |  |
| --- | --- |
| 项目名称： | 动态内存通讯录系统 |
| 专业名称： | 计算机科学与技术 |
| 班级名称： | 24计算机科学与技术二 |
| 组 长： | 陆佳涛24219111219 |
| 组 员： | 王子乔24219111224 |
| 指导教师： | 许淑华 |
|  |  |

2025年06月

**1 任务描述**

本项目使用 C 语言实现一个动态扩容的通讯录管理系统，主要完成以下功能：

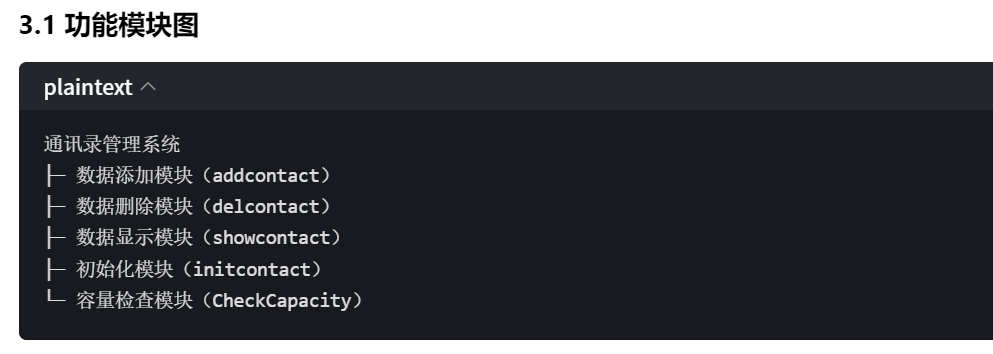
* 联系人信息录入（姓名、年龄、性别、电话、地址）
* 联系人信息浏览
* 联系人信息删除
* 通讯录动态扩容（初始容量 3 人，满员后每次增加 2 人容量）

**2 需求分析**

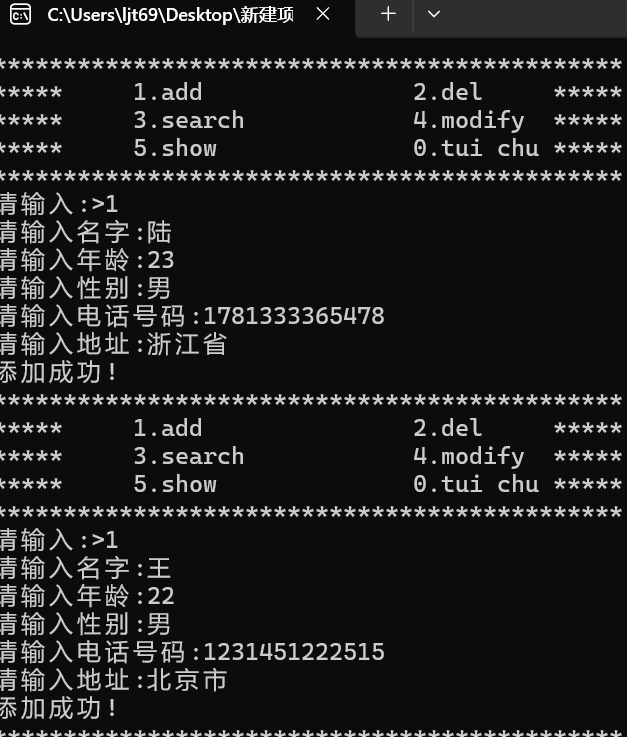
通过对通讯录系统的分析，确定以下功能需求：

* ****数据存储****：使用动态数组存储联系人信息，通过malloc和realloc实现内存动态管理
* ****核心操作****：
  + 添加联系人（含容量检查与自动扩容）
  + 删除联系人（按姓名查找并删除）
  + 显示所有联系人信息
* ****交互界面****：键盘式菜单驱动（0-6 数字选项）
* ****扩展性****：预留查询、修改、排序等功能接口（暂未实现）

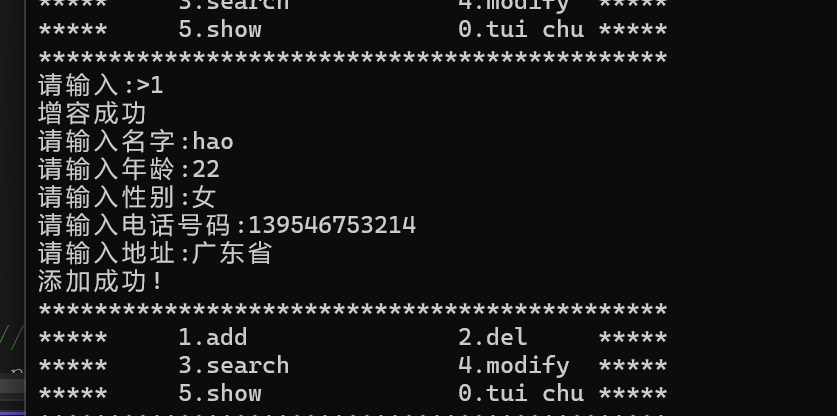
**3 总体设计**



1.添加add模块



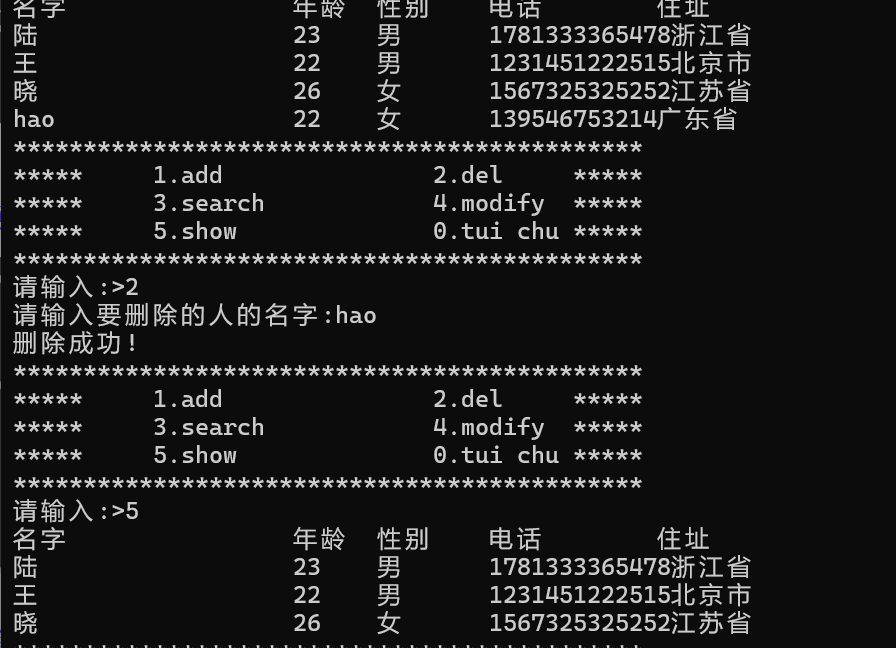
1.add模块的动态扩容



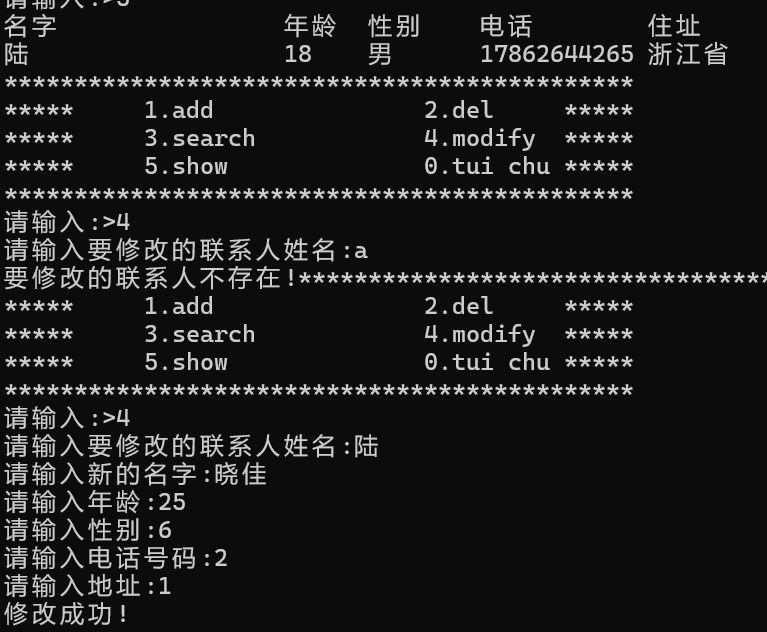
1. show展示模块



3.del删除模块



4.modify修改模块



**4 详细设计**

**4.1 拟采用解决问题的方法**

// 联系人信息结构体 typedef struct peopleinformation { char name[20]; // 姓名 int age; // 年龄 char sex[5]; // 性别 char tele[12]; // 电话 char addr[30]; // 地址 } PeoInfo; // 通讯录结构体（包含动态数组） typedef struct contact { PeoInfo\* data; // 动态数组指针 int size; // 当前数据量 int capacity; // 当前容量 } Contact;

**4.2 主要算法和处理流程图**

#### **4.2.1 动态扩容算法**

* ****功能****：当通讯录满员时自动扩展内存
* ****逻辑****：
  1. 检查size == capacity
  2. 使用realloc扩展内存（每次增加 2 个元素空间）
  3. 成功则更新data指针和capacity，失败则报错退出

#### **4.2.2 删除联系人算法**

* ****功能****：根据姓名删除指定联系人
* ****逻辑****：
  1. 遍历数组查找匹配姓名
  2. 找到后将后续元素前移覆盖目标元素
  3. 减少size计数

4.2.3 修改联系人算法

功能：根据查找人的姓名来修改已经保存信息

逻辑:

1. . 先通过遍历数组查找匹配姓名（如果没找到就说明联系人不存在）
2. . 然后找到后将新的元素前移覆盖目标元素
3. .退回menu菜单界面

**5 实验遇到的语法问题和逻辑问题**

### **5.1 语法问题**

1. ****结构体指针操作****：初期误用ps.data访问成员，后修正为ps->data
2. ****动态内存分配****：忘记在initcontact中初始化size=0，导致显示空数据
3. ****scanf 安全警告****：通过#pragma warning(disable:4996)规避scanf不安全警告

### **5.2 逻辑问题**

1. ****扩容时机错误****：初始版本在添加数据后检查容量，导致首次添加第 4 个元素时未触发扩容，修正为添加前检查
2. ****数组越界风险****：删除函数中未判断i == size，导致访问无效下标，增加条件判断解决

**6 程序实现**

**总代码**

**contact.h**

#pragma once

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

enum xinxi {

MAX\_NAME = 20, //枚举是逗号，

MAX\_SEX = 5,

MAX\_TELE = 12,

MAX\_ADDE = 30,

MAX\_PEOPLE =1000,

DEFAULT\_SZ = 3

};

struct peopleinformation { //结构体是分号

char name[MAX\_NAME];

int age;

char sex[MAX\_SEX];

int tele[MAX\_TELE];

char addr[MAX\_ADDE];

};

struct contact{

//初始化通讯录

struct peopleinformation \*data;//存放信息

//查看已经存放的数量

int size;

int capacity;

};

//声明函数

void initcontact(struct contact\* ps);//类型是strcut con\* ps是我自己创造的变量

void addcontact(struct contact\* ps);

void showcontact(const struct contact\* ps);

void delcontact( struct contact\* ps);

void modifycontact(struct contact\* ps);

**contact.c**

//#define \_CRT\_SECURE\_NO\_WARNINGS 1 作用与下面一致

#pragma warning(disable:4996)//规避scanf函数不安全的问题

#include "contact.h"//本地头文件要""来引用

void initcontact(struct contact\* ps) { //注意a是结构体指针

ps->data = (struct peopleinformation\*)malloc(DEFAULT\_SZ \* sizeof(struct peopleinformation));

if (ps->data == NULL)

{

return;

}

ps->size = 0;

ps->capacity = DEFAULT\_SZ;

}

void CheckCapacity(struct contact\* ps ){

if (ps->size == ps->capacity) {

//增容

struct peopleinformation\* ptr = realloc(ps->data, (ps->capacity + 2) \* sizeof(struct peopleinformation));

if (ptr != NULL) {

ps->data = ptr;

ps->capacity += 2;

printf("增容成功\n");

}

else { printf("增容失败\n"); }

}

}

void addcontact(struct contact\* ps) {

//检测当前通讯录的容量

//1.如果满了，就增加空间

//2.如果不满，就啥事不干

CheckCapacity(ps);

printf("请输入名字:");

//scanf("%s", ps->data->name);//以下两者都可以

scanf("%s", ps->data[ps->size].name);

printf("请输入年龄:");

scanf("%d", &(ps->data[ps->size].age));

printf("请输入性别:");

scanf("%s", ps->data[ps->size].sex);

printf("请输入电话号码:");

scanf("%s", ps->data[ps->size].tele);

printf("请输入地址:");

scanf("%s", ps->data[ps->size].addr);

ps->size++;

printf("添加成功!\n");

}

void showcontact(const struct contact\* ps) {

if (ps->size == 0)

printf("通讯录为空\n");

else {

int i = 0;

printf("%-20s%-6s%-8s%-12s%-2s\n", "名字", "年龄", "性别", "电话", "住址");

for (i = 0;i < ps->size;i++) {

printf("%-20s%-6d%-8s%-12s%-2s\n",

ps->data[i].name,

ps->data[i].age,

ps->data[i].sex,

ps->data[i].tele,

ps->data[i].addr);

/\*printf("名字:%s\n", ps->data[i].name);

printf("年龄:%d\n", ps->data[i].age);

printf("性别:%s\n", ps->data[i].sex);

printf("电话:%s\n", ps->data[i].tele);

printf("住址:%s\n", ps->data[i].addr);\*/

}

}

}

void delcontact( struct contact\* ps) {

char name[MAX\_NAME];

printf("请输入要删除的人的名字:");

scanf("%s", name);

//1.查找要删除的人]

int i = 0;

for (i = 0;i < ps->size;i++) {

if (0 == strcmp(ps->data[i].name, name)) { //strcmp里面内容相等就输出0

break;

}

}

//2.删除

if (i == ps->size) {

printf("要删除的数据不存在\n");

}

else {

int j = 0;

for (j = i;j < ps->size - 1;j++) {

ps->data[j] = ps->data[j + 1];

}

ps->size--;

printf("删除成功!\n");

}

}

void modifycontact(struct contact \*ps) {

char name[MAX\_NAME];

printf("请输入要修改的联系人姓名:");

scanf("%s",name);

//查找联系人

int i;

for (i = 0;i < ps->size;i++) {

if (strcmp(ps->data[i].name, name) == 0)

{

break;

}

}

if (i == ps->size) {

printf("要修改的联系人不存在!\n");

}

else {

//输入新的联系人信息

printf("请输入新的名字:");

scanf("%s", ps->data[i].name);

printf("请输入年龄:");

scanf("%d", &(ps->data[i].age));

printf("请输入性别:");

scanf("%s", ps->data[i].sex);

printf("请输入电话号码:");

scanf("%s", ps->data[i].tele);

printf("请输入地址:");

scanf("%s", ps->data[ps->size].addr);

printf("修改成功!\n");

}

}

void menu() {

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

printf("\*\*\*\*\* 1.add 2.del \*\*\*\*\*\n");

printf("\*\*\*\*\* 3.search 4.modify \*\*\*\*\*\n");

printf("\*\*\*\*\* 5.show 0.tui chu \*\*\*\*\*\n");

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

}

int main() {

int input = 0;

//创建通讯录con

struct contact con; //con就是通讯录,里面包含：data指针和size,capacity.

initcontact(&con);//结构体传地址才能改变结构体内部的量

do {

menu();

printf("请输入:>");

scanf("%d", &input);

switch (input) {

case 1:

addcontact(&con);

break;

case 2:

delcontact(&con);

break;

case 3:

break;

case 4:

break;

case 5:

showcontact(&con);

break;

case 6:

break;

default:

printf("输入错误，请输入正确的数字!!!\n");

break;

case 0:

printf("感谢您的使用!\n");

break;

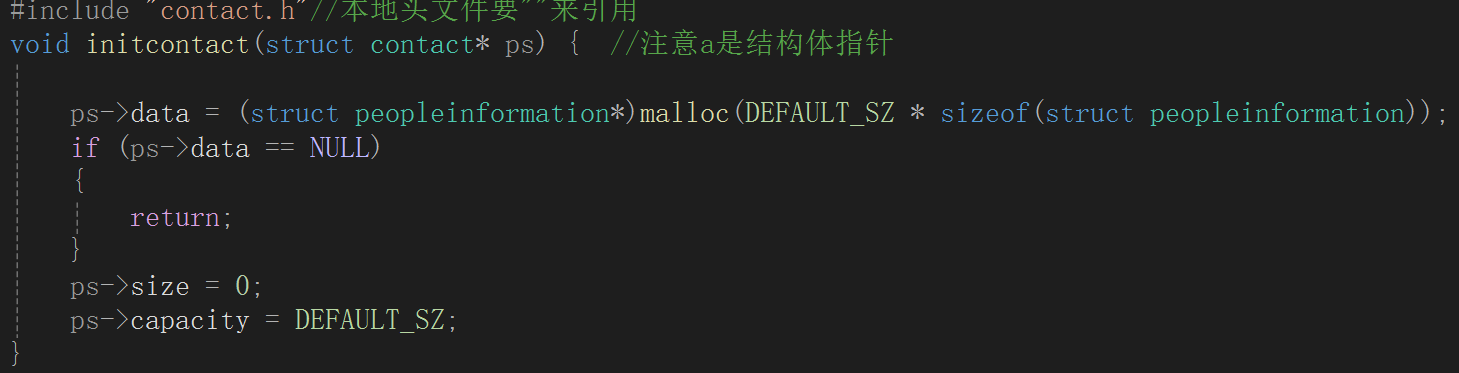
}

} while (input);

return 0;

}

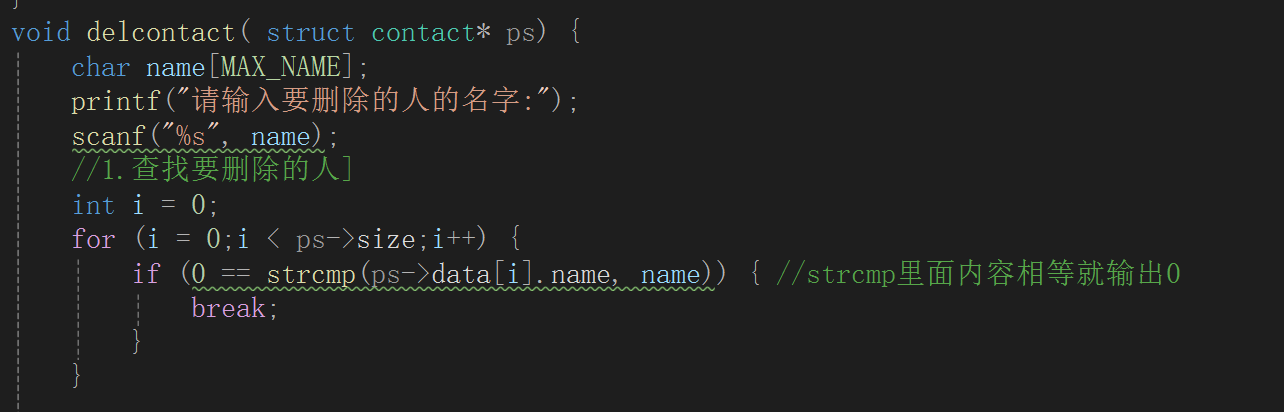
### **6.1 初始化模块**

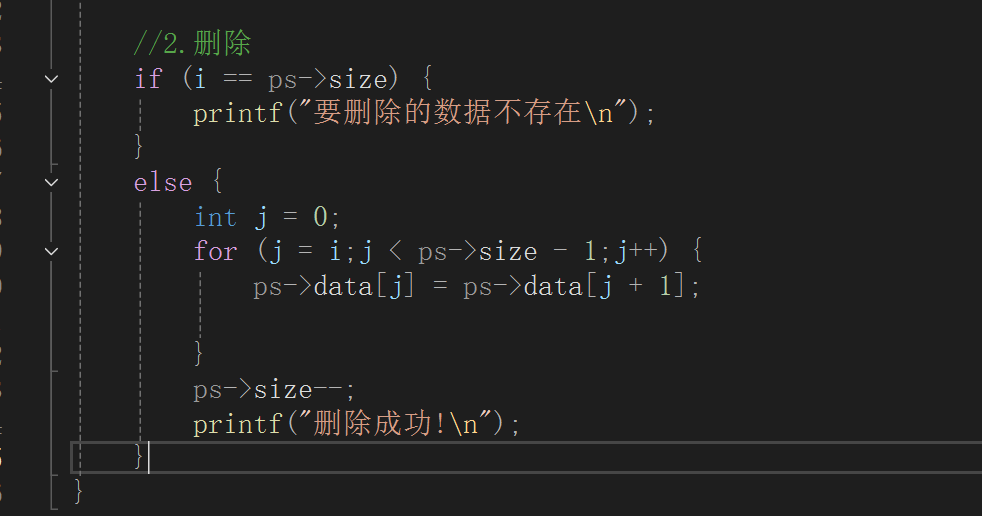
****说明****：初始化通讯录，分配动态内存并设置初始状态

### **6.2 添加模块**



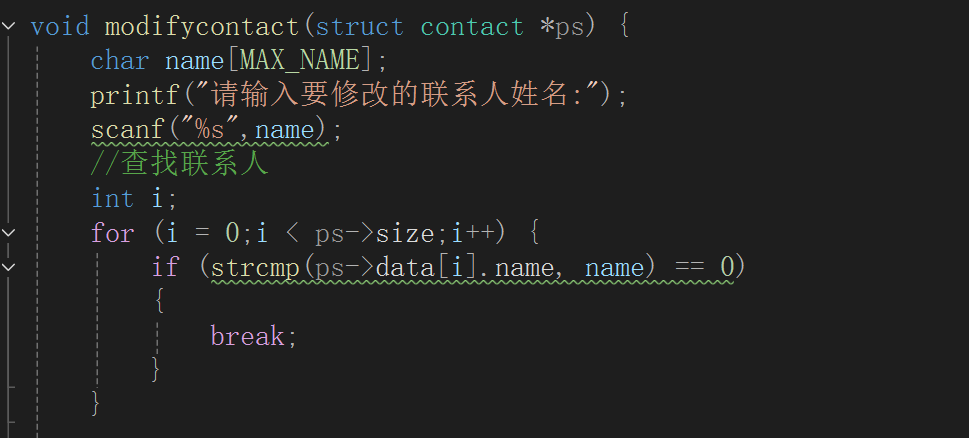
### **6.3 删除模块**

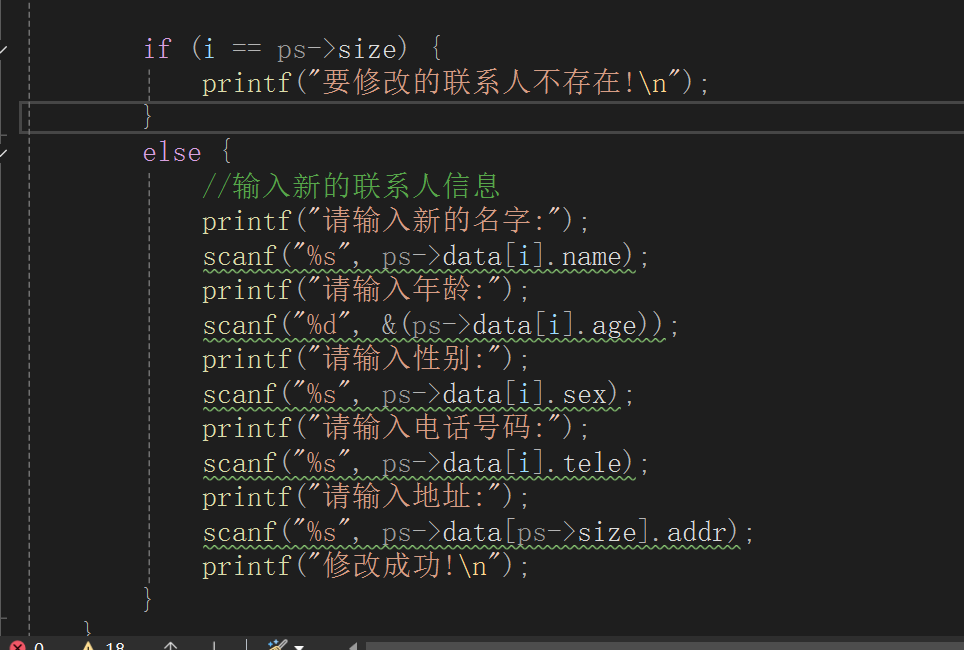




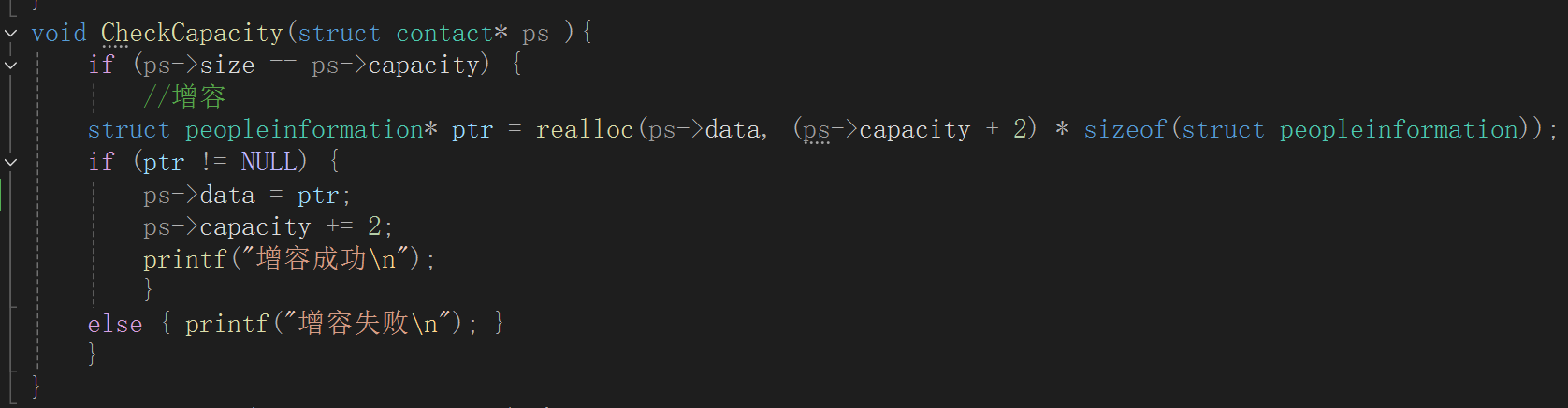
****说明****：通过strcmp比较姓名，找到后通过数组元素前移实现删除

6.4修改模块





6.5检查动态内存是否需要扩容模块



**7 测试结果及分析**

### **添加联系人**

* ****操作****：连续添加 3 个联系人（未触发扩容）
* ****结果****：显示正常，容量仍为 3

### **触发扩容**

* ****操作****：添加第 4 个联系人
* ****结果****：控制台输出 “增容成功，当前容量：5”，添加成功

### **删除联系人**

* ****操作****：删除已存在的联系人姓名
* ****结果****：显示列表中该联系人消失，数据量正确减少

### 检查动态内存是否需要扩容模块

操作：通过size（现通讯录储存数据）与capacity（通讯录最大容量）进行 比较，判断通讯录是否需要扩容

结果：当需要扩容时，程序会自动扩容并显示“扩容成功！”

**8 小结**

### **系统优点**

* 实现动态扩容功能，避免固定数组的空间浪费
* 模块化设计，各功能模块职责清晰
* 提供友好的菜单交互界面

### **待改进点**

* 未实现数据持久化（文件存储）
* 查询、排序功能尚未完成
* 输入校验不完善（如年龄负数、电话格式等）

### **后续计划**

* 添加文件读写功能（fread/fwrite）
* 实现按姓名查询、按年龄排序等功能
* 完善输入数据合法性校验

**9 小组成员分工情况（即小组成员工作分派和奉献度）**

| **成员** | **任务分工** | **奉献度** |
| --- | --- | --- |
| 陆佳涛（组长） | 总体架构设计、主函数实现，ppt | 50% |
| 王子乔 | 添加 / 显示模块开发，代码调试 | 30% |
| 王子乔 | 删除 / 扩容模块开发、文档整理 | 20% |

《**程序设计基础课程设计》项目评分**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **考核项目** | **考核细项** | **具体要求** | **分数** | **得分** |
| **程序实现**  **（45分）** | 实现效果 | 程序功能完整，各功能模块之间逻辑清晰。程序能正常运行，且测试结果准确。 | 25 | 25 |
| 编码规范 | 采用结构化程序设计方法，接口（函数参数、返回值等）标准，编码规范。 | 10 | 10 |
| 课堂外技巧 | 对知识有扩展，能运用课堂外内容解决问题。 | 10 | 10 |
| **报告撰写**  **（40分）** | 任务分析与设计 | 能够清晰的阐述系统分析和设计过程。 | 15 | 15 |
| 核心代码解析 | 能够对各功能模块的核心代码进行清晰的展示和解析。 | 15 | 15 |
| 总结和分析 | 有程序测试运行的结果，并能对项目完成情况进行总结和分析。 | 10 | 10 |
| **团队合作与答辩**  **（15）** | 团队分工 | 团队分工合理、明确，每个成员都有明确的团队角色和工作任务 | 5 | 5 |
| 自述情况 | 现场表达清楚、条理清晰。主讲完整介绍项目总体情况及特色，小组成员重点讲解自己完成的情况。 | 5 | 5 |
| 回答现场提问 | 团队清楚正确地回答现场教师的提问。 | 5 | 5 |
| **项目总分** | | | | 100 |
| **评语： 团队分工十分明确，两人配合默契，在编译代码的同时进行密切交流，快速完成了整个通讯录系统的编译。在调试环节，尽管在面对更困难的动态内存时遭受了些许困扰，但两人互相查漏补缺，成功且快速地发现了错误并修正。在本次程序设计基础课程设计中表现出色，成功设计出了，操作简洁，错误较少，易于理解，使用内存更少的通讯录系统，值得嘉奖！**    　　　评阅教师签名：  　　　　　　　　　　　　　　　　　　　　　　　　2025　　年　　6月　11　日 | | | | |