重庆邮电大学

学生实验报告册

课程名称：A2080580-软件技术基础

学　　院：自动化学院

专业班级：自动化与电气工程类08122104

姓　　名：王忠全

学　　号：2021212981

指导教师：陈卓老师

成　　绩：

学年学期：2022-2023 学年 □春■秋学期

重庆邮电大学教务处制

实验项目名称	学生名册管理程序和核酸队列管理程序的设计		
实验地点	线上	实验时间	8-15 周，周 五 5.6 节
实验指导教师	陈卓老师	成绩	

一、实验目的

1.掌握顺序表、链表、栈、队列、二叉树等重要数据结构的工作原理，能够编写其操作接口代码，并完成功能调试;

2.掌握顺序、掌握折半(二分)查找算法，冒泡排序和插入排序算法，并能完成代码编写与调试。

二、实验内容

1.基于链式存储结构的学生名册管理程序设计

利用包含头节点的单链表设计一个学生名册管理程序，记录每个学生的学号、姓名和电话号码，并通过数字化菜单为用户提供插入、删除、查找和打印等功能。通过菜单的选择，用户可以对链表内的数据进行管理。需要实现的功能包括:

1) 添加学生信息。可一次添加 n 人，n 为键盘录入的整数。
2) 在指定位置插入一个学生的数据。
3) 修改学生信息。根据学号定位到链表中相应的节点，用户可以对姓名和电话号码进行修改。
4) 删除特定学号的学生。
5) 在屏幕上输出所有学生的数据。需要显示学生的总人数；一个人的学号、姓名和电话应显示在同一行。

以上功能对应的菜单在执行时应当没有顺序和次数的限制。设计程序时需要对各种特殊情况（如空链表）加以充分的考虑。

2.基于顺序存储结构的核酸检测队列管理程序设计

利用顺序存储结构设计一个核酸检测队列管理程序，记录每个参检人员的姓名和身份证号，并通过数字化菜单为用户提供入队、出队、查找和屏幕输出等功能。通过菜单的选择，用户可以对排队的人员进行管理。

需要实现的功能包括:

1) 入队功能。增加的人数，以及人员的信息来自于键盘输入。
2) 出队检测。可以一次安排 10 人混检，也可以安排单检。出队检测后，人员的信息从排队人员中删除。
3) 查找功能。输入身份证号，判断相应人员是否完成了核酸检测，或者是还在队伍中排队，或者没有来排队。该功能需要对已经完成检测的人员信息进行存储。
4) 输出功能。按照身份证号排序，输出当前正在排队的人员的信息。
5) 队列状态的显示和预警。队列状态的基本信息主要指当前排队人数，其他信息可以自行扩展（如排在第一位的是谁，预估当前队列完成检测的时间等）。队列状态的信息可以伴随数字化菜单显示；也可以通过菜单进行查看。当排队人数超过设定的上限时，入队功能自动失效。

以上功能对应的菜单在执行时应当没有顺序和次数的限制。在程序中需要对各种特殊情况加以充分的考虑（如队列为满或为空）。

完成以上程序代码的编写、调试，提供运行结果截图与相应的文字说明。

三、设计方案

1.基于链式存储结构的学生名册管理程序设计

(1)分析程序的需求：需要用到链式数据结构，结构体定义，需要完成包含头节点的单链表设计一个学生名册管理程序，完成五大功能添加学生信息、指定位置插入一个学生的数据、修改学生信息、删除特定学号的学生、在屏幕上输出所有学生的数据。

并且考虑到了程序没有顺序、次数限制，各个功能可以重复使用。并且对两种特殊情况进行考虑分析：空链表、输入错误信息。

考虑到主函数与各个模块间的传参，没有使用全局变量，运用指向指针的指针。

(2)菜单设计：以简洁美化的展示界面位置，引导用户自行选择。

(3)程序设计：

①头文件申明、结构体定义

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100

typedef struct student
{
    char name[N];     // 学生名字
```

```
    int x_num;          // 学号
    int tel_num;        // 电话号码
    struct student *next; //指针
}student;
```

②系统展示与功能选择模块

实现方法：输入参数，再返回函数值到主程序，执行功能选择。

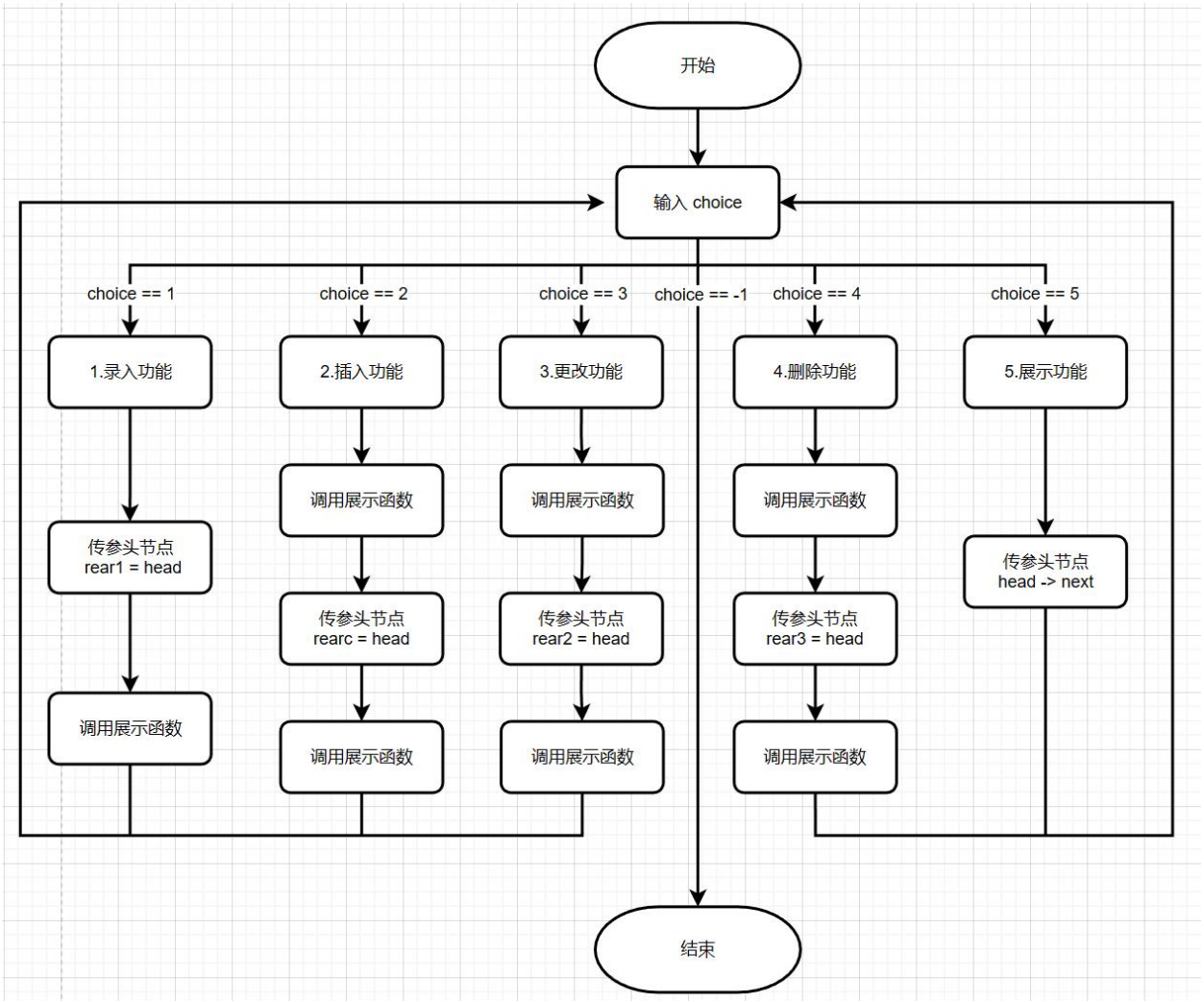

图 3.1.1 系统展示与功能选择模块流程图

③录入功能模块

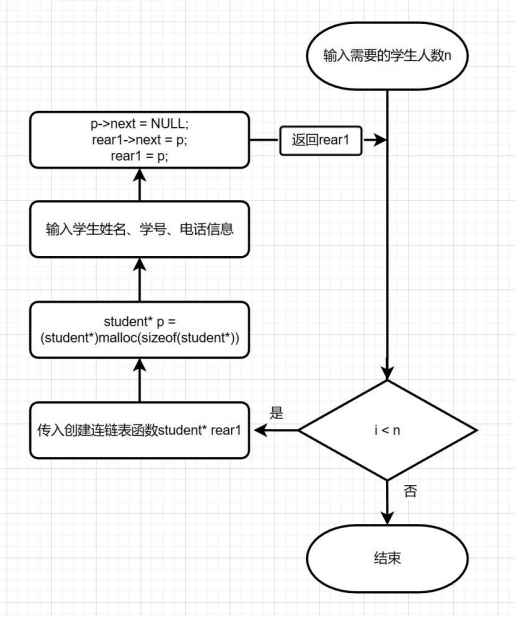

图 3.1.2 录入功能模块流程图

④插入链表模块

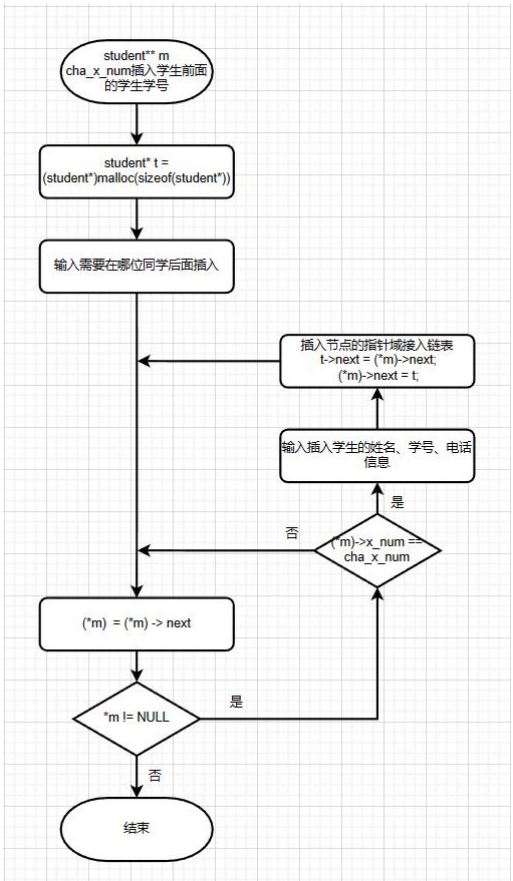

图 3.1.3 插入链表模块流程图

⑤更改链表模块

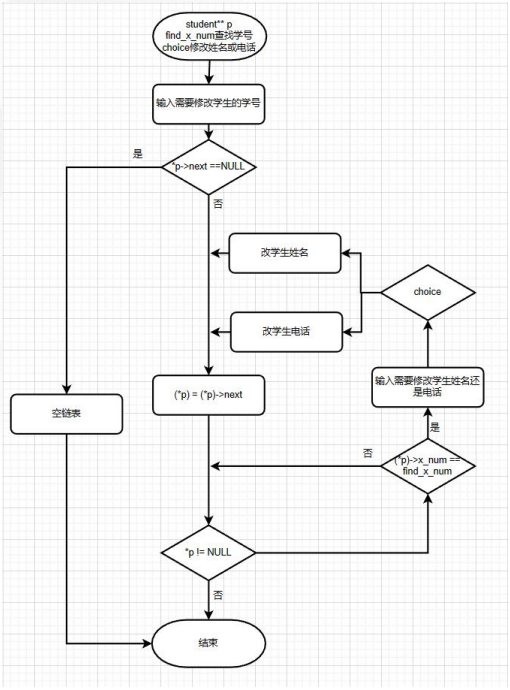

图 3.1.4 更改链表模块流程图

⑥删除链表模块

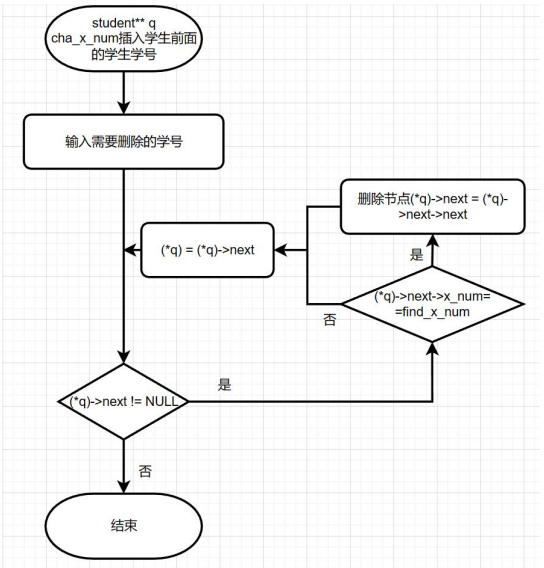

图 3.1.5 删除链表模块流程图

⑦打印学生信息模块

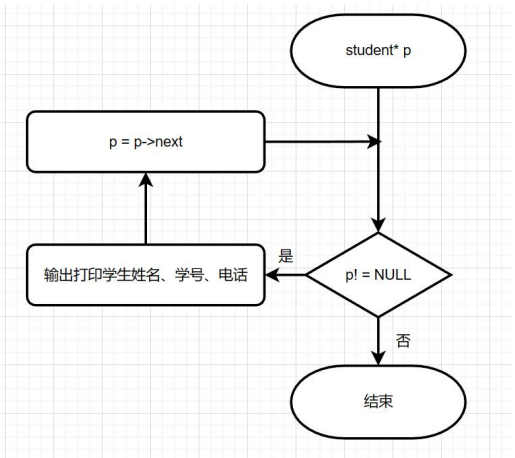

图 3.1.6 打印学生信息模块流程图

2.基于顺序存储结构的核酸检测队列管理程序设计

(1)分析程序的需求：需要用到队列数据结构，结构体定义，利用顺序存储结构设计一个核酸检测队列管理程序，记录每个参检人员的姓名和身份证号，并通过数字化菜单完成四大功能：入队、出队、查找和屏幕输出等功能。通过菜单的选择，用户可以对排队的人员进行管理。并且将人员划分为三部分：排队、已检、未检类别分别存储判别。

并且考虑到了队列状态的显示和预警，在主程序上显示排队信息，队列状态的信息可以伴随数字化菜单显示，且也可以通过菜单功能进行查看。通过显示可以清楚看到队伍排队顺序，排在第几位，检测需要花费的时间等。

运用了两部分结构体，没有使用全局变量，通过取地址各个函数传参进行变换队列。并且运用循环队列知识，使得队列可以适用排队人数动态变化。

(2)菜单设计：以简洁美化的展示界面位置，引导用户自行选择。

(3)程序设计：

①头文件申明、结构体定义

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 10       // 名字最大 10 个字
#define List_MAX 50  // 队伍最大 50 人

typedef struct people
{
    char name[MAX]; //
    int id;
}people;

typedef struct list
{
    people list[List_MAX];  // 队伍数组
    int front;  // 头
    int rear;   // 尾
}list;
```

②系统展示与功能选择模块

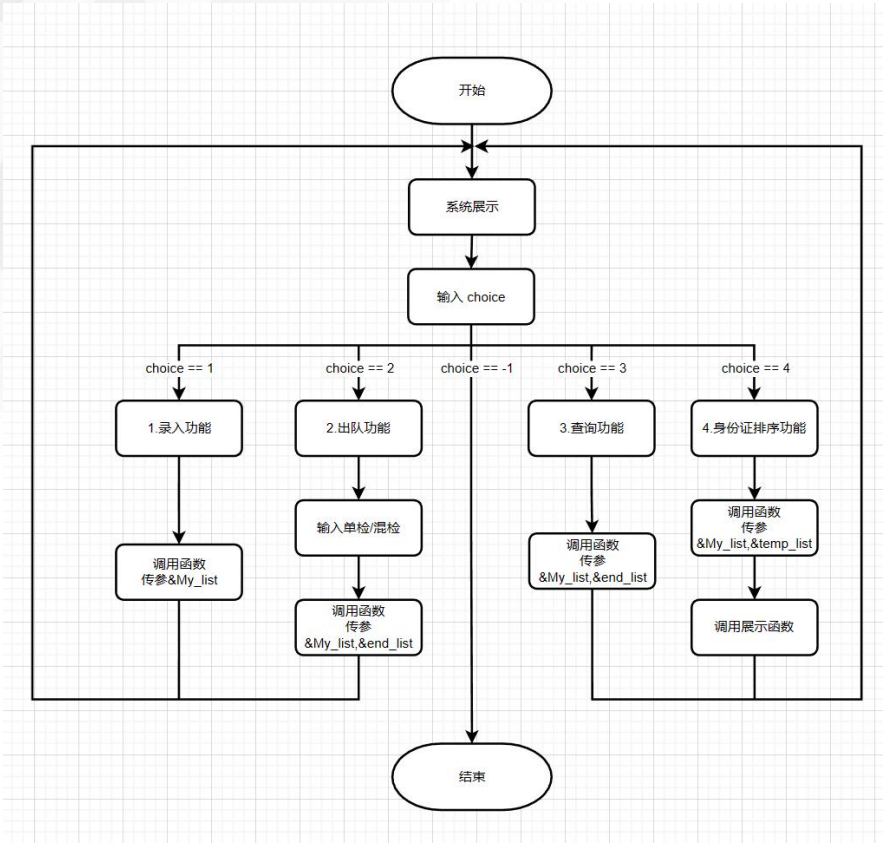

图 3.2.1 系统展示与功能选择模块流程图

③入队模块

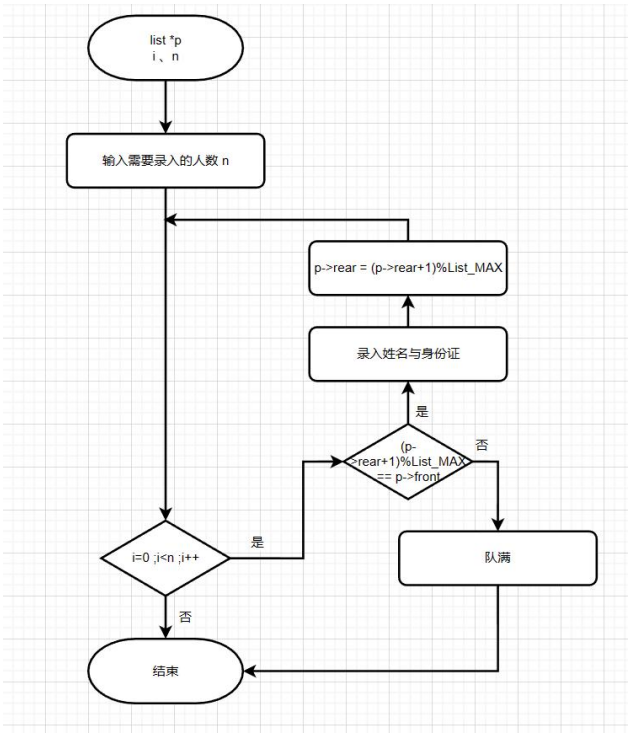

图 3.2.2 入队模块流程图

④出队模块

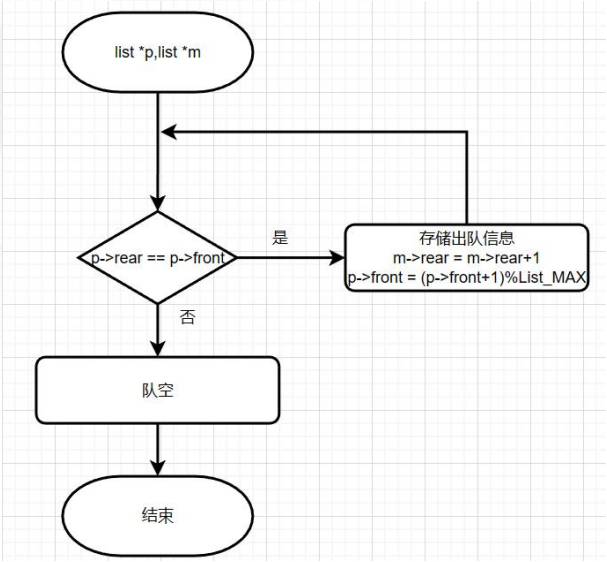

图 3.2.3 出队模块流程图

⑤查询模块

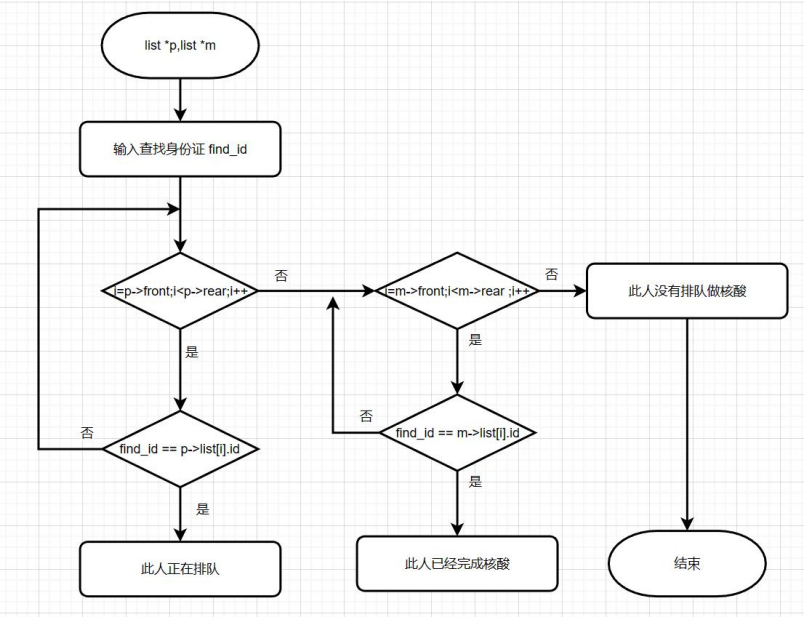

图 3.2.4 查询模块流程图

⑥身份证排序展示模块

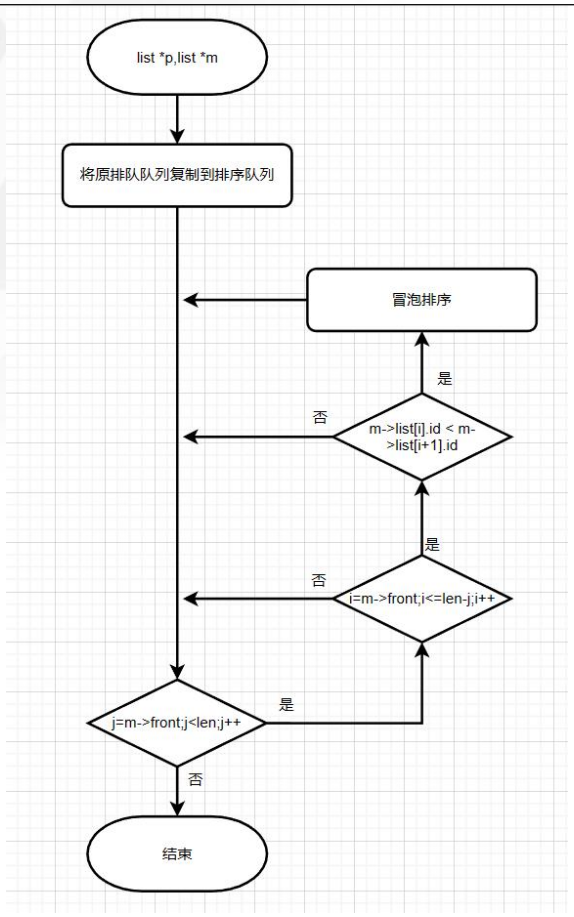

图 3.2.5 身份证排序展示模块流程图

⑦排队信息展示模块

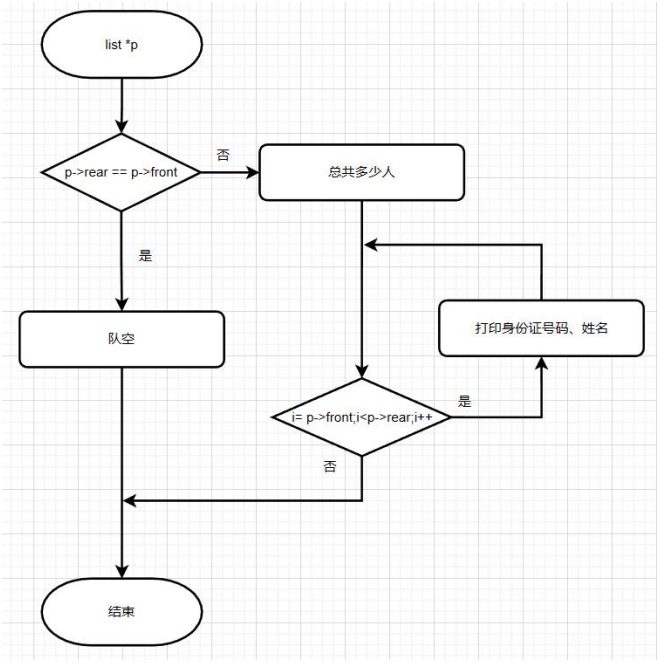

图 3.2.6 排队信息展示模块流程图

四、实验仪器设备、材料

运行软件的平台 Windows 11
软件编写工具 CodeBlocks 、Dev-C++

五、实验步骤

1.基于链式存储结构的学生名册管理程序设计

①分析实验要求

明确了是需要利用链式存储结构，定义了结构体，申明了头指针，并且根据老师课堂传授知识，运用指向指针的指针**来改变链表。

②根据设计需求，分别设计了每一个函数板块的程序。

int System_Show(int x)	系统界面展示函数
student* Creat_list(student* rear1)	创建链表函数
void List_ALL_show(student* p)	信息展示
void List_single_show(student* p)	单次展示
void change(student** p)	更改链表函数
void cha(student** m)	插入链表
void delted(student** q)	删除链表函数
int main()	主函数

③调试优化程序

输入设计验证值，录入系统与预期功能实现比较，进行 debug 与调试程序，修改完善程序，并做程序可视化，优化程序结构。

2.基于顺序存储结构的核酸检测队列管理程序设计

①分析实验要求

明确了是需要利用队列存储结构，定义了结构体，队列顺序表，并且根据陈老师课堂传授知识，运用冒泡排序、入队出队等数据结构知识。

②根据设计需求，分别设计了每一个函数板块的程序。

void in_list(list *p)	入队模块
void pop_list(list *p,list *m)	出队模块
void find_list(list *p,list *m)	查询函数
void id_show(list *p,list *m)	身份证排序展示功能
void List_show(list *p)	排队信息展示

int main()　　主函数

③调试优化程序

输入设计验证值，录入系统与预期功能实现比较，进行 debug 与调试程序，修改完善程序，并做程序可视化，优化程序结构。

六、实验结果及分析（或设计总结）

测试结果及分析，包括:

1) 运行结果截图。

1.基于链式存储结构的学生名册管理程序设计

①初始化界面展示

图 6.1.1 初始化界面展示

②添加学生信息功能

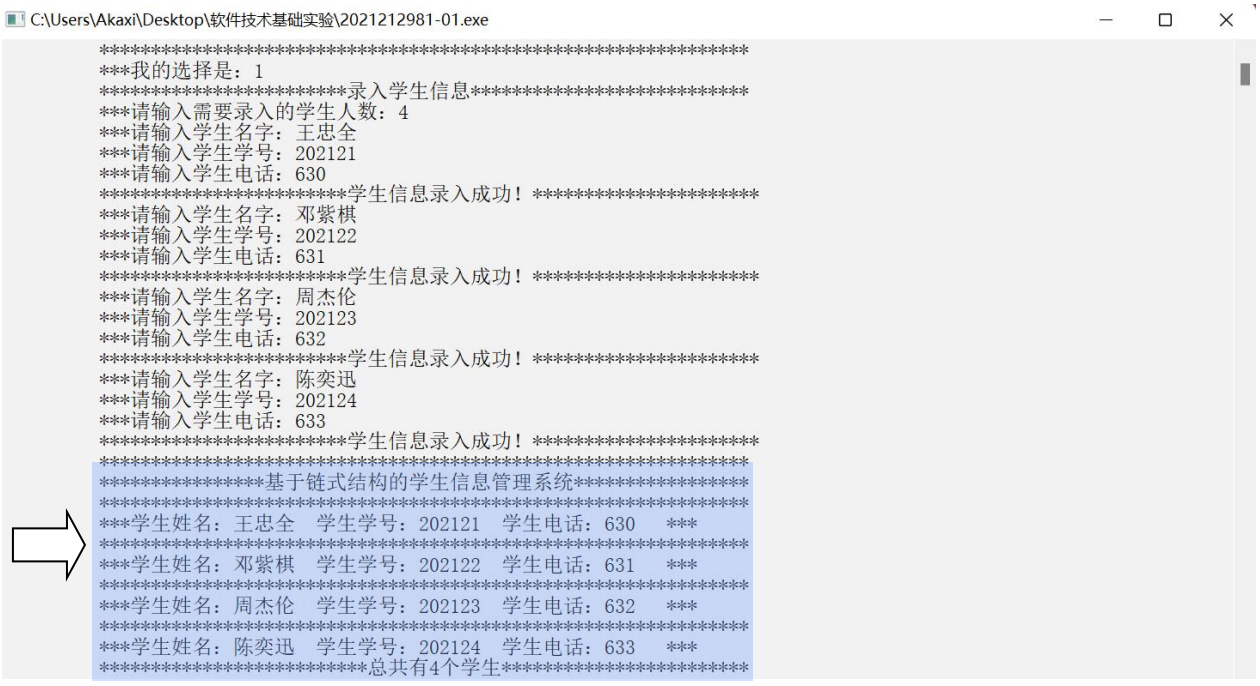

图 6.1.2 录入 4 位学生基本信息展示

如图 6.1.2，输入 1 选择添加学生信息，输入需要录入 n=4 位同学信息如图，并且系统自动输出录入学生的总信息。

③插入学生信息功能

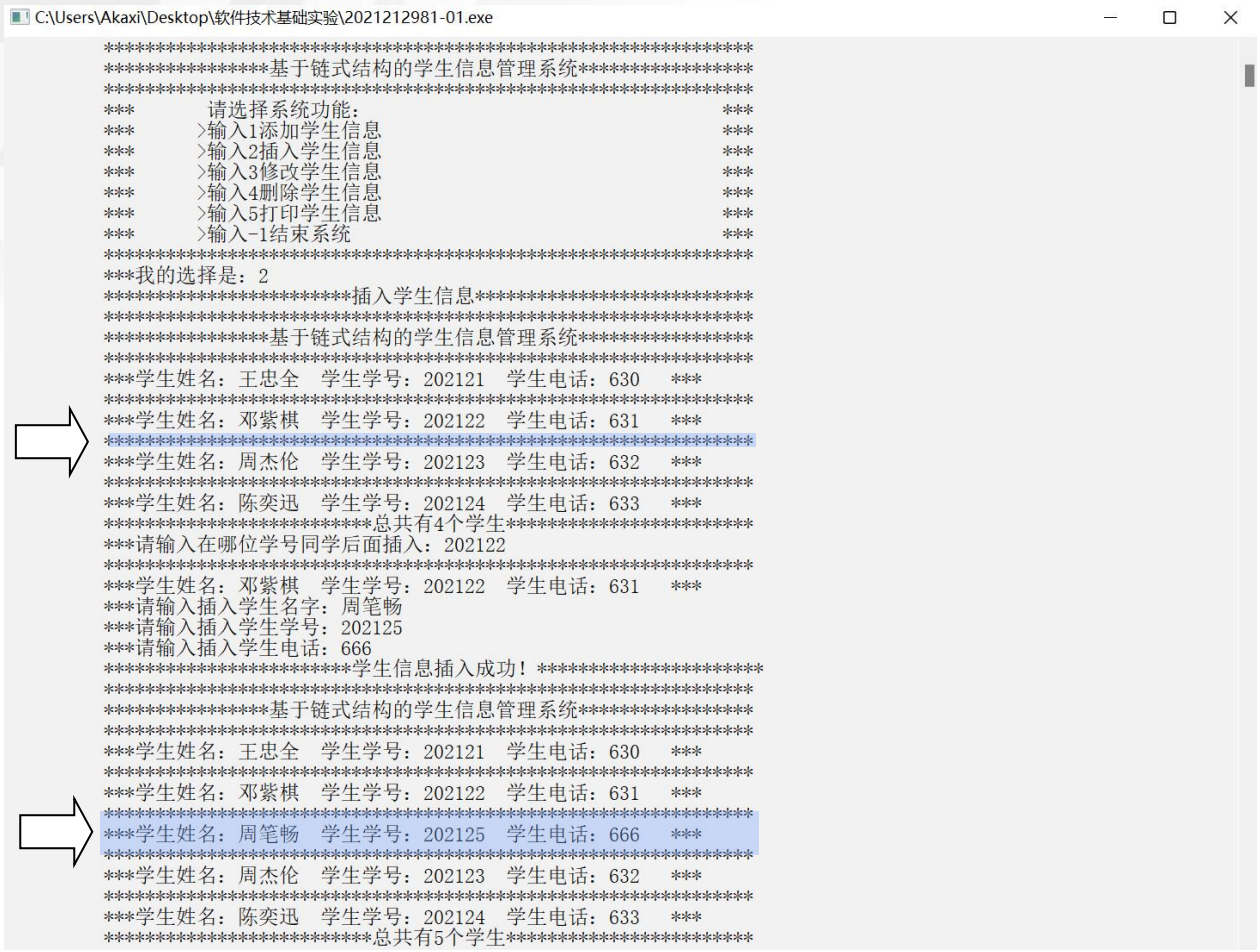

图 6.1.3 插入周笔畅同学信息展示

如图 6.1.3，输入 2 选择插入学生信息，输入邓紫棋的学号 202122，在其后面插入周笔畅的学生信息，并且输出整个链表，可以看见插入后输出信息正确。

④修改学生信息功能

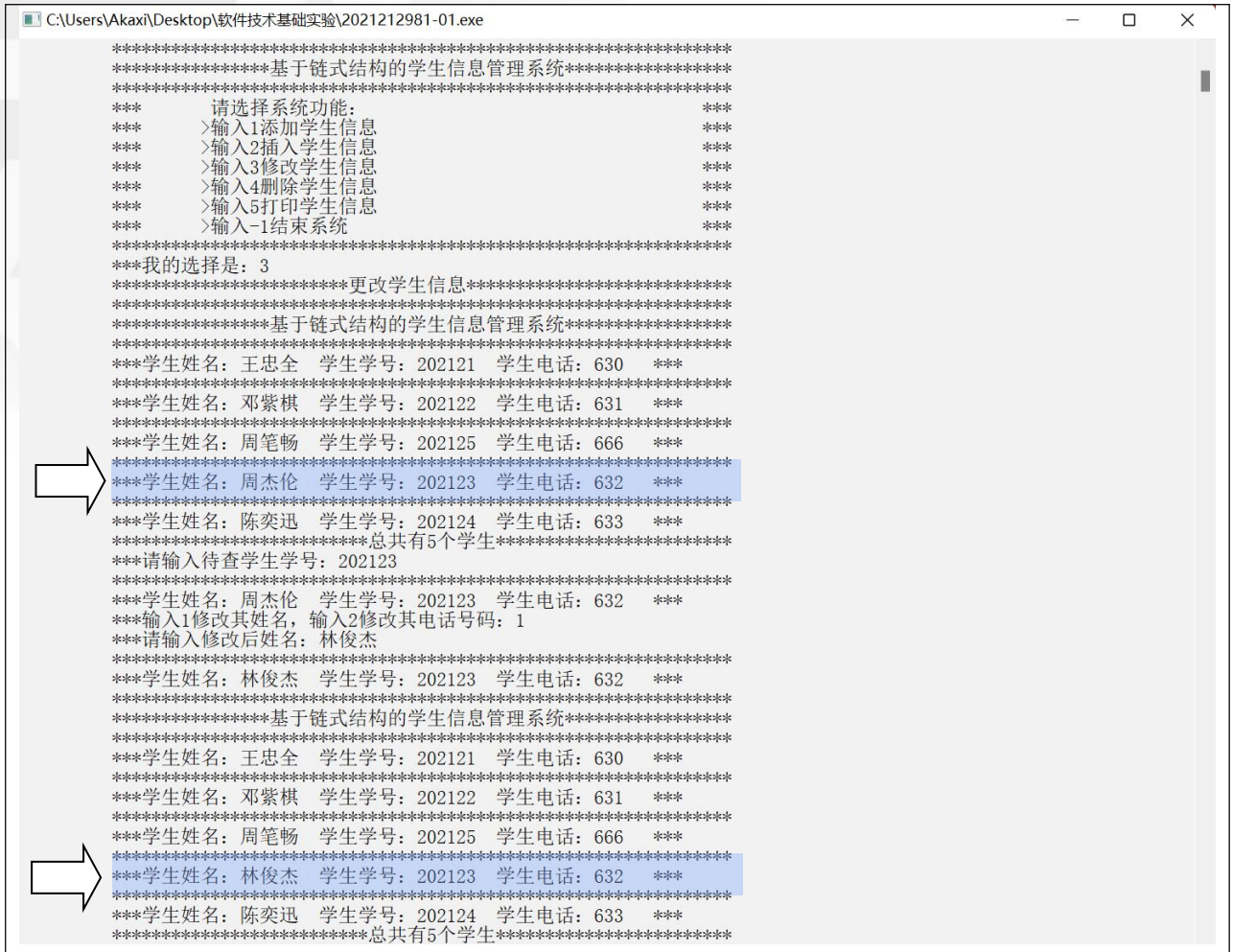

图 6.1.4 修改周杰伦同学信息展示

如图 6.1.4，将周杰伦同学姓名修改为林俊杰，输入周杰伦学号 202123，并且输入 1 更改其姓名。并且输出打印，结果成功。

⑤删除学生信息功能

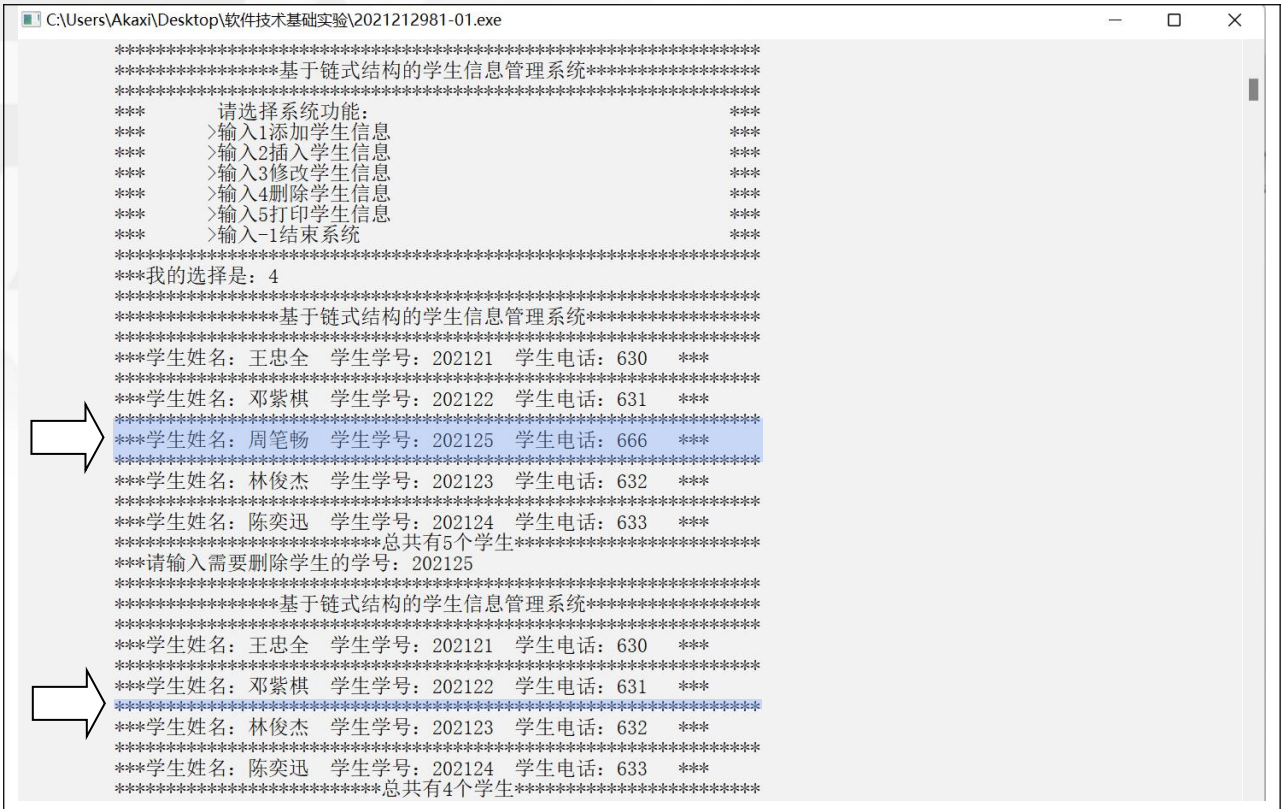

图 6.1.5 删除周笔畅同学信息展示

如图 6.1.5，输入 4 删除学号为 202125 的周笔畅同学，并且打印输出，可见已经删除了周笔畅同学，结果删除成功。

⑥打印学生信息功能

图 6.1.6 打印全部同学信息展示

如图 6.1.6，输入 5 打印前面的所有学生信息链表，并且输出全部学生人数，结果打印成功。

⑦结束系统功能

图 6.1.7 程序结束界面

输入-1 结束整个系统，并且最后输出程序运行结果：即学生链表信息。

2.基于顺序存储结构的核酸检测队列管理程序设计

①初始化界面展示

图 6.2.1 程序结束界面

②录入排队人员信息功能

图 6.2.2 录入 5 位排队人员信息

如图 6.2.1，输入 1 录入 5 位核酸检测排队人员姓名与身份证号码信息，并且打印输出，在系统功能菜单清晰可见排队情况，总共多少人、从前往后、检测时间。

③安排人员核酸检测功能

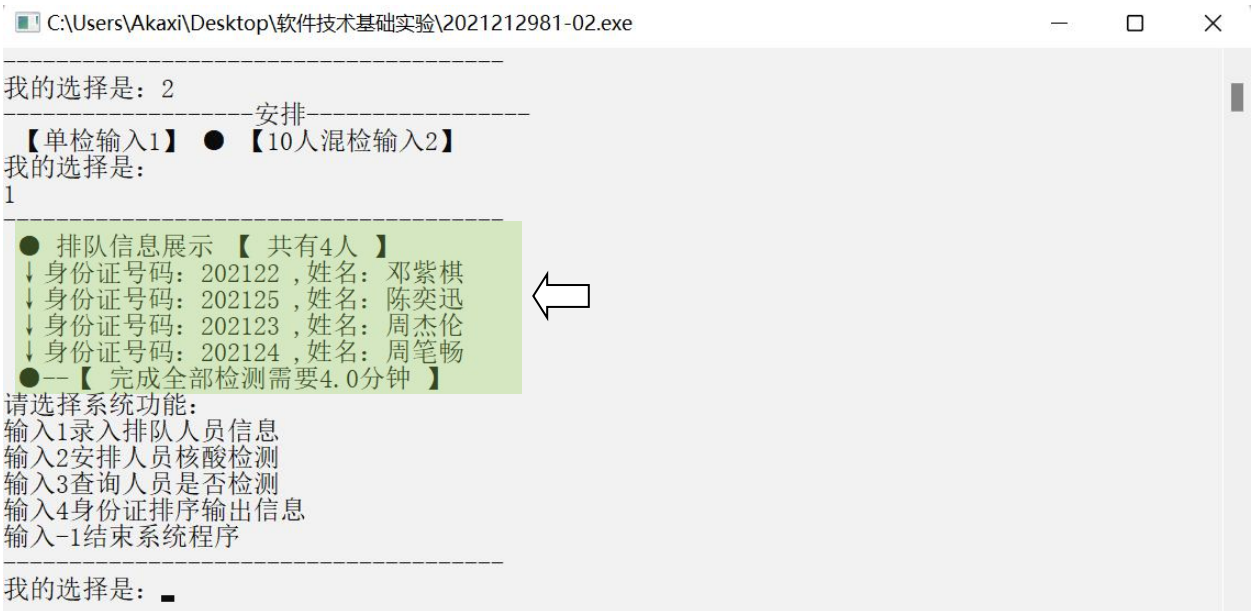

图 6.2.3 安排人员核酸检测功能

如图 6.2.3，输入 2 选择安排核酸检测功能，输入 1 选择单检，并且将已检人员存储起来，输出打印还在排队人员，结果成功。

④查询人员是否检测功能

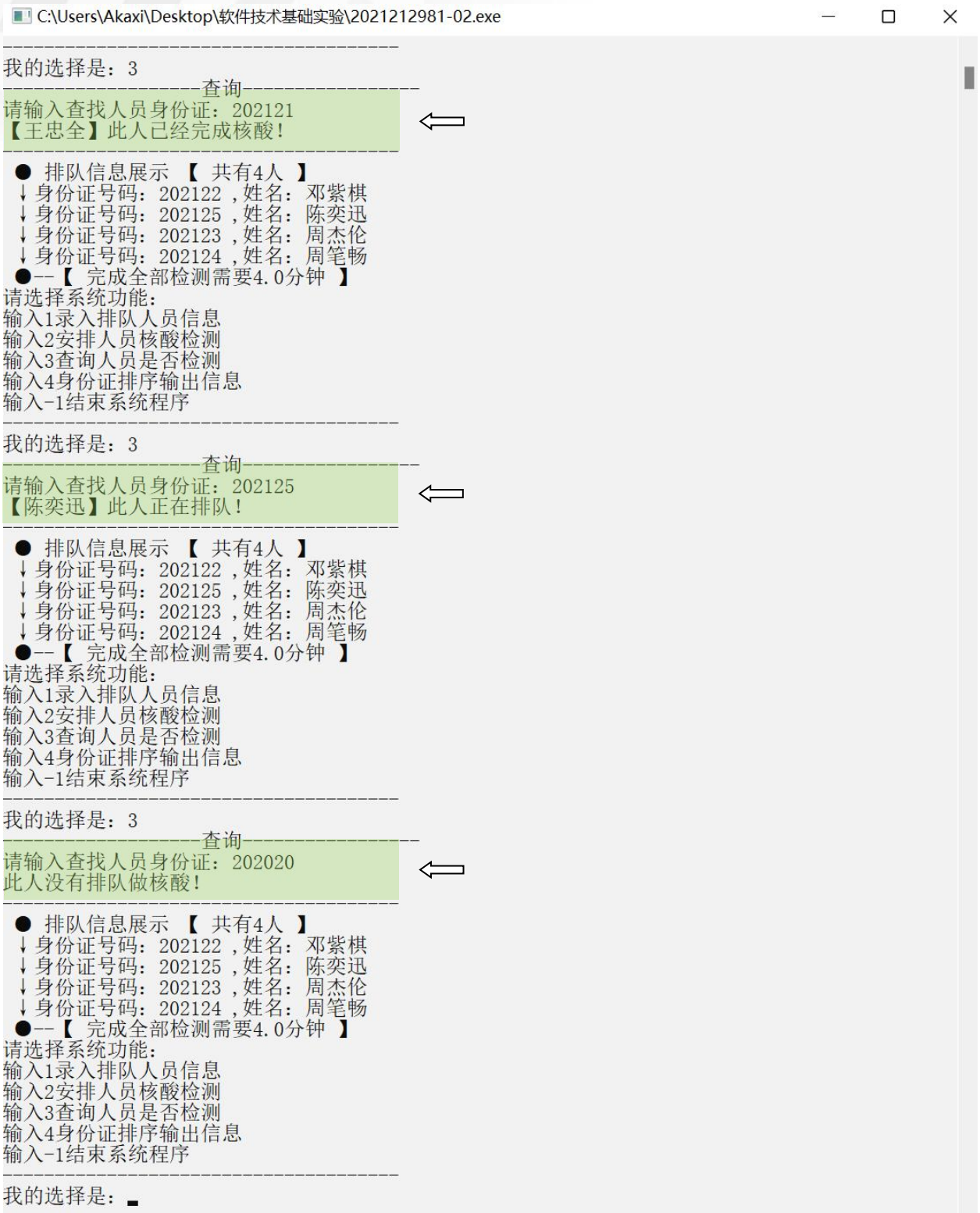

图 6.2.4 查询核酸检测信息功能

如图 6.2.4，输入 3 进行查询功能，分别查询已做核酸的【202121 王忠全】、正在排队的

【202125 陈奕迅】、没有做核酸的【202020】三类情况，结果查询成功。

⑤身份证排序展示功能

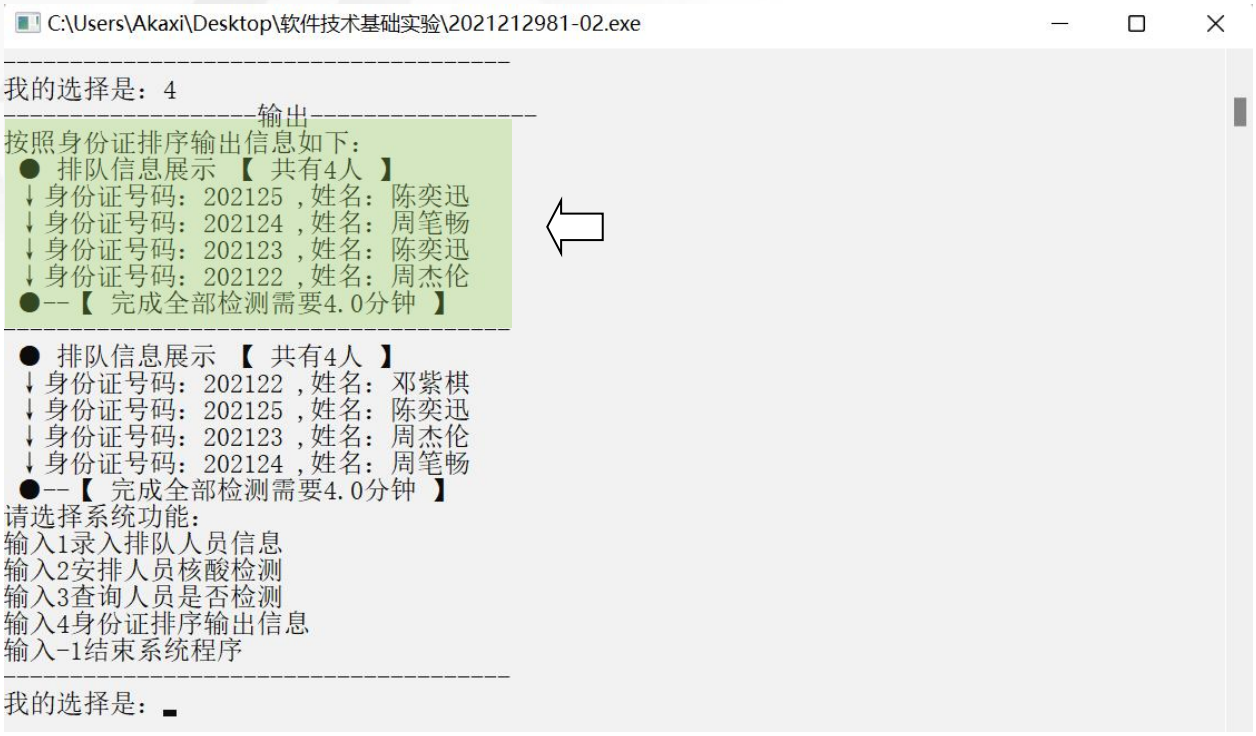

图 6.2.5 身份证排序展示功能

如图 6.2.5，输入 4 进行排序，按照身份证号码降序，对排队人员信息进行排序输出，从 202125->202122 降序成功并且输出，且不改变原排队顺序。

⑥结束系统功能

图 6.2.6 程序结束界面

2) 分析程序中的不足，以及可以改善的地方。对于不能实现的功能，需要分析原因。

①不足：界面不够简洁、部分功能介绍不够完整，输出展示界面不够精美，可以后续进行修改调整可视化界面。

②不足：程序编写还是不够精致、通俗，对于【基于链式存储结构的学生名册管理程序设计】运用了刘老师上课讲的指向指针的指针，各个存储空间比较复杂，主函数与形参

之间比较模糊。对于【基于顺序存储结构的核酸检测队列管理程序设计】在冒泡排序时，程序设计比较繁琐，可以进行进一步改善。

③功能：实现了两个实验设计的所有规定功能，但是在核酸队列的拓展功能时，设计了总人数和检测时间与排队系统界面可视化功能，可以进一步探索核酸队列的其他功能。

3) 实验课程总结与学习心得。

①在一定程度上改变了我的写代码习惯。之前我是先写主函数，而且喜欢把所有功能堆砌在主函数里面，缺乏分块编写的思维。通过学习陈卓老师的课程与编写实验，我逐渐养成了“分而治之”的思想，与系统化模块化思想，每个独立函数功能，主函数与函数之间的关系，形参与变量等等，这对于我的写代码习惯有莫大帮助。

②在一定程度上纠正了我的错误。以前总喜欢写全局变量，但是通过本次实验，我的两个程序设计都没有用到全局变量，都是主函数之间的参数传递，地址引用等等，这对于我理解计算机存储结构，以及计算机的数据结构有莫大帮助。

③在一定程度上培养了我的性格与品质。编写这两个程序时，难免会出现报错，于是需要反复不停的 debug，根据运行结果以及逻辑判断，一次又一次的调试，记得调试【基于顺序存储结构的核酸检测队列管理程序设计】时，就不知不觉从晚上 20:00 调试到了凌晨 1:00 并且孜孜不倦。我知道坚持与从错误中学习的重要性，这才能使我学懂，学好。对于代码的浓厚兴趣也使我在其他语言有所帮助，比如 python、C++等等，写代码不能草草了事模模糊糊，更多的是要注入心血与精力才能浇灌出一个良好系统程序。

④致谢。感谢陈卓老师的孜孜不倦教导，通过一个学期的数据结构实验学习，真真正正把书本上的理论运用到计算机实践，并且从手中实现一个又一个程序，是一趟奇妙的代码世界的旅行，感谢陈卓老师的一路陪伴与指导，我会带着这份美好在接下的学习中不断向前。

【End】

附录：源程序

1. 基于链式存储结构的学生名册管理程序设计

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100

typedef struct student
{
    char name[N];       // 学生名字
    int x_num;          // 学号
    int tel_num;        // 电话号码
    struct student *next; //指针
}student;


// 系统界面展示函数
int System_Show(int x)
{
    printf("
**************************************
**********************\n");
    printf("              **********基于
链式结构的学生信息管理系统
**********\n");
    printf("
**************************************
**********************\n");
    printf("              ***       请选择系统
功                能                :
***\n");
    printf("              ***       >输入 1 添加
学          生          信          息
***\n");
    printf("              ***       >输入 2 插入
学          生          信          息
***\n");
    printf("              ***       >输入 3 修改
学          生          信          息
***\n");
    printf("              ***       >输入 4 删除
学          生          信          息
***\n");
    printf("              ***       >输入 5 打印
学          生          信          息
***\n");
    printf("              ***       >输入-1 结束
系                                    统
***\n");
    printf("
**************************************
**********************\n");
    printf("              ***我的选择是：");
    scanf("%d",&x);  // 用户选择
    return x;
}

// 创建链表函数
student* Creat_list(student* rear1)
{
    student*          p          =
(student*)malloc(sizeof(student*));

    getchar();
    printf("              ***请输入学生名字：");
    scanf("%s",p->name);
    printf("              ***请输入学生学号：");
    getchar();
    scanf("%d",&p->x_num);
    printf("              ***请输入学生电话：");
    getchar();
    scanf("%d",&p->tel_num);
    printf("
**********学生信息录入成功!
**********\n");

    p->next = NULL;
    rear1->next = p;
    rear1 = p;
    return rear1;
}

// 信息展示函数
void List_ALL_show(student* p)
{
    int count =0;
```

```
    printf("
**************************************
**********************\n");
    printf("                ****************基于
链式结构的学生信息管理系统
*****************\n");
    while(p!= NULL)
    {
        count++;
        printf("
**************************************
**********************\n");
        printf("                ***学生姓名：%s
学生学号：%d    学生电话：%d
***\n",p->name,p->x_num,p->tel_num);

        p = p->next;
    }
    printf("
*************************总共有%d 个学生
***********************\n",count);
}

// 单次展示
void List_single_show(student* p)
{
    printf("
**************************************
**********************\n");
    printf("            ***学生姓名：%s  学生
学号：%d    学生电话：%d
***\n",p->name,p->x_num,p->tel_num);
}


// 更改链表函数
void change(student** p)
{
    int find_x_num =0; //查找学号
    int choice = 0;
    char change_name[N] ={'\0'};
    int change_num = 0;
    printf("                ***请输入待查学生学
号：");
    scanf("%d",&find_x_num);
    int T=0;
    if((*p)->next == NULL)
    {
        printf("                ***链表为空!");
    }
    else
    {
        *p = (*p)->next ;
        while((*p) != NULL )
        {
            if((*p)->x_num == find_x_num)
            {
                List_single_show(*p);    // 展
示
                printf("                ***输入 1
修改其姓名，输入 2 修改其电话号码：");
                scanf("%d",&choice);
                if(choice == 1)
                {
                    printf("                ***请
输入修改后姓名：");

scanf("%s",change_name);
                    strcpy((*p)->name,
change_name);
                    List_single_show(*p);   //
展示
                }
                else if(choice == 2)
                {
                    printf("                ***请
输入修改后电话号码：");

scanf("%d",&change_num);
                    (*p)->tel_num           =
change_num;
                    List_single_show(*p);   //
展示
                }
                else
                {
                    printf("输入错误！");
                }
                T = 1; // 找到
            }
            (*p) = (*p)->next ;
```

```
            }
            if(T==0)
            {
                printf("              ***没有此学号同学！\n");
            }
        }

}


// 插入链表函数
void cha(student** m)
{
    int cha_x_num =0;
    student*                t                =(student*)malloc(sizeof(student*));

    printf("              ***请输入在哪位学号同学后面插入：");
    scanf("%d",&cha_x_num);
    *m = (*m)->next ;
    while((*m) != NULL )
    {
        if((*m)->x_num == cha_x_num)
        {
            List_single_show(*m);    // 展示
            getchar();
            printf("              ***请输入插入学生名字：");
            scanf("%s",t->name);
            printf("              ***请输入插入学生学号：");
            getchar();
            scanf("%d",&t->x_num);
            printf("              ***请输入插入学生电话：");
            getchar();
            scanf("%d",&t->tel_num);
            printf("**********学生信息插入成功!**********\n");
            t->next = (*m)->next;
            (*m)->next = t;
        }
        (*m) = (*m)->next;
    }
}


// 删除链表函数
void delted(student** q)
{
    int find_x_num =0;
    printf("              ***请输入需要删除学生的学号：");
    scanf("%d",&find_x_num);
    while((*q)->next != NULL)
    {
        if((*q)->next->x_num == find_x_num)
        {
            (*q)->next  =  (*q)->next->next;  // 删除
        }
        (*q) = (*q)->next ;
    }

}

// 主函数
int main()
{
    student*                head                =(student*)malloc(sizeof(student*));
    head->name[N] ='\0';
    head->x_num=0;
    head->tel_num=0;
    head->next=NULL;

    student* rear1 = head;
    student* rear2 = head;
    student* rear3 = head;
    student* rearc = head;
    int choice = 0 ,i=0 , n=0;      // 用户选择

    while(choice != -1)
    {
        if(choice == 1)
        {
            printf("**********录入学生信息**********\n");
```

```
            printf("              ***请输入需要录入的学生人数: ");
            scanf("%d",&n);
            for(i=0;i<n;i++)
            {
                rear1 = Creat_list(rear1); // 创建链表
            }
            List_ALL_show(head->next); // 展示链表
        }

        if(choice == 2)
        {
            printf("************************ 插入学生信息 ****************************\n");
            List_ALL_show(head->next); // 展示链表
            rearc = head;
            cha(&rearc);
            List_ALL_show(head->next); // 展示链表
        }

        if(choice == 3)
        {
            printf("************************ 更改学生信息 ****************************\n");
            List_ALL_show(head->next); // 展示链表
            rear2 = head;
            change(&rear2);   // 更改物品信息
            List_ALL_show(head->next);
        }

        if(choice == 4)
        {
            List_ALL_show(head->next);
        }

        choice = System_Show(choice);   // 展示界面，下一次循环
    }
    List_ALL_show(head->next); // 展示系统
    printf("************************系统程序正常结束!**********************\n");
}
```

2. 基于顺序存储结构的核酸检测队列管理程序设计

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 10          // 名字最大 10 个字
#define List_MAX 50   // 队伍最大 50 人

typedef struct people
{
    char name[MAX];
    int id;
}people;

typedef struct list
{
    people list[List_MAX];   // 队伍数组
    int front;   // 头
    int rear;    // 尾
}list;

    list My_list;      // 排队
    list end_list;    // 已检
    list temp_list;   // 排序

// 入队模块
void in_list(list *p)
{
    int i,n; //录入人数
```

```
    char inName[MAX] ={'\0'};
    int inID = 0;
    printf("请输入需要录入的人数： ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        if(      (p->rear+1)%List_MAX      ==
p->front )
        {
            printf("排队人数已达上限 50 人");
            break; //结束
        }
        else
        {
            printf("请输入排队人员姓名:");
            scanf("%s",inName);
            printf("请输入排队人员身份证号
码:");
            scanf("%d",&inID);

            strcpy(p->list[p->rear].name,
inName); // 录入
            p->list[p->rear].id = inID;       //录
入

            p->rear = (p->rear+1)%List_MAX ;
// 录入成功指针向前
        }
    }
}

// 出队模块
void pop_list(list *p,list *m)
{
    if(p->rear == p->front)
    {
        printf("队空！ ");
    }
    else
    {
        m->list[m->rear].id                    =
p->list[p->front].id ;              // 存储已检信
息
        strcpy(          m->list[m->rear].name,
p->list[p->front].name);
        m->rear = m->rear+1 ;
        p->front = (p->front+1)%List_MAX ;
    }
}

// 查询函数
void find_list(list *p,list *m)
{
    int find_id = 0; // 查找的身份证
    int i =0 ,flag =0;
    printf("请输入查找人员身份证： ");
    scanf("%d",&find_id);
    for(i=p->front;i<p->rear;i++)
    {
        if(find_id == p->list[i].id)
        {
            printf("【%s】此人正在排队！
\n",p->list[i].name);
            flag = 1;
            break;
        }
    }
    for(i=m->front;i<m->rear ;i++)
    {
        if(find_id == m->list[i].id)
        {
            printf("【%s】此人已经完成核酸！
\n",m->list[i].name);
            flag =1;
            break;
        }
    }
    if(flag == 0)
    {
        printf("此人没有排队做核酸！ \n");
    }
}

// 身份证排序展示功能
void id_show(list *p,list *m)
{
    int t_id;
    char t_name[10];
    int i ,j ,len;
    for( i=p->front ;i<=p->rear;i++ )   // 复制
    {
```

```
            m->list[i].id = p->list[i].id ;
            strcpy( m->list[i].name, p->list[i].name);
        }
        m->front = p->front;   // 长度匹配
        m->rear = p->rear;


        len
=((m->rear)-(m->front)+List_MAX)%List_MAX;
// 长度
        for(j=m->front;j<len;j++) // 冒泡排序
        {
            for(i=m->front;i<=len-j;i++)
            {
                if( m->list[i].id < m->list[i+1].id)
                {
                    t_id = m->list[i].id;
                    strcpy(                t_name,
p->list[i].name);

                    m->list[i].id = m->list[i+1].id;
                    strcpy(          m->list[i].name,
m->list[i+1].name);

                    m->list[i+1].id = t_id;
                    strcpy(       m->list[i+1].name,
t_name);
                }
            }
        }

}

// 排队信息展示
void List_show(list *p)
{
    int i;
    if(p->rear == p->front)   // 队空
    {
        printf(" ● 排队信息展示 【 共有 0
人 】 \n");
        printf(" ●--【 完成全部检测需要 0 分
钟 】\n");
    }
    else
    {
        printf(" ● 排队信息展示 【 共有%d
人                                    】
\n",((p->rear)-(p->front)+List_MAX)%List_MAX)
;
        for(i= p->front;i<p->rear;i++)
        {
            printf(" ↓身份证号码: %d ,姓
名: %s \n",p->list[i].id,p->list[i].name);
        }
        printf(" ●--【 完成全部检测需要%d.0
分              钟                    】
\n",((p->rear)-(p->front)+List_MAX)%List_MAX)
;
    }

}

// 主函数
int main()
{
    int choice = 0 , h_choice =0; // 选择
    int i;



    My_list.rear=0;   // 队尾
    My_list.front=0; // 队头
    end_list.rear=0;
    end_list.front=0;
    temp_list.rear=0;
    temp_list.front=0;

    printf("--------------------------------------\n");
    printf("基于顺序存储结构的核酸检测队列
管理系统\n");
    do
    {

printf("--------------------------------------\n");
        List_show(&My_list);   // 排队信息展
示


        printf("请选择系统功能: \n");
        printf("输入 1 录入排队人员信息\n");
        printf("输入 2 安排人员核酸检测\n");
        printf("输入 3 查询人员是否检测\n");
```

```
        printf("输入4身份证排序输出信息\n");
        printf("输入-1 结束系统程序\n");

printf("-------------------------------------\n");
        printf("我的选择是： ");
        scanf("%d",&choice);
        if(choice == 1)
        {
            printf("------------------- 录 入
----------------\n");
            in_list(&My_list);
            printf("---------------- 录 入 成
功!--------------\n\n");

        }

        else if(choice ==2)
        {
            printf("------------------- 安 排
----------------\n");
            printf(" 【单检输入 1】 ● 【10
人混检输入 2】 \n");
            printf("我的选择是： ");
            scanf("%d",&h_choice);
            if(h_choice ==1)
            {
                pop_list(&My_list,&end_list);
            }
            if(h_choice ==2)
            {
                for(i=1;i<=10;i++)
                {

pop_list(&My_list,&end_list);
                }
            }
        }

        else if(choice ==3)
        {
            printf("------------------- 查 询
----------------\n");
            find_list(&My_list,&end_list);
        }

        else if(choice ==4)
        {
            printf("------------------- 输 出
----------------\n");
            id_show(&My_list,&temp_list);
            printf("按照身份证排序输出信息
如下： \n");
            List_show(&temp_list);
        }

    }while(choice != -1);
    printf("---------------- 程 序 正 常 结
束!--------------\n");
    return 0;
}
```