**华 中 科 技 大 学**

**网络空间安全学院**

本科：《C语言程序设计实验》

实验报告

姓 名

班 级 计算机类（II）240x班

学 号

联系方式 （邮箱或电话）

分 数

评 分 人

**报告要求**

**1. 报告不可以抄袭，发现雷同者均记为0分。**

**2. 报告中不可以只粘贴大段代码，应是文字与图、表结合的，需要说明流程的时候，也应该用流程图或者伪代码来说明；如果发现有只有大段代码粘贴者，报告需重写。**

**3. 报告格式严格按照要求规范，并作为评分标准。**

**4. 报告要求双面打印。**

**课程目标评价标准**

（1）课程目标2的实验报告评价标准

|  |  |  |  |
| --- | --- | --- | --- |
| 评 价 标 准 | | | |
| 优秀 | 良好 | 及格 | 不及格 |
| 结合网络查询、检索的文献资料信息，能够很好地在实验报告中完整地描述程序设计的全流程。能够深刻的理解格式规范在实验报告中的意义和作用，并能够很好的应用到报告编写过程中。 | 结合网络查询、检索的文献资料信息，能够较好地在实验报告中较完整地描述程序设计的全流程。能够较好的理解格式规范在实验报告中的意义和作用，并能够较好的应用到报告编写过程中，但不够全面。 | 结合网络查询、检索的文献资料信息，能够在实验报告中描述程序设计的全流程，但不够完整。能够理解格式规范在实验报告中的意义和作用，并能够大部分应用到报告编写过程中。 | 不能够结合网络查询、检索的文献资料信息，应用到实验报告中描述程序设计的全流程。不能够理解格式规范在实验报告中的意义和作用，在报告编写过程中不能够遵循规范要求。 |

**实验报告评分细则**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **评 分 项 目** | | | **满**  **分** | **得分** | **评分标准** |
| 实验  内容  （目标2） | | 内容完整 | 20 |  | 20-14：完全包含程序设计的全流程，并且所有实验均完成，参考文献来源丰富。  13-8：包含程序设计的全流程，并且所有实验基本完成，参考文献基本满足要求。  7-0：程序设计流程缺失，实验内容有缺失，参考文献贫乏。 |
| 代码实现 | 20 |  | 20-14：功能正确，完全按照要求实现。  13-8：功能基本正确，基本按照要求实现。  7-0：功能存在缺陷，未按要求实现。 |
| 测试及结果分析 | 20 |  | 20-14：测试数据覆盖所有可能情况，测试方法科学、完整，结果分析准确完备。  13-8：测试数据基本覆盖可能情况，测试方法描述基本正确、完整，结果分析准确完备。  7-0：测试仅针对公开数据集的通过性进行描述。 |
| 文档格式（段落、行间距、缩进、图表、编号等）  （目标2） | | | 20 |  | 基本要求：目录、标题、行间距、缩进、正文字体字号按照模板要求执行，图、表清晰且有标号。  20-14：格式规范美观，满足要求。  13-8：基本满足要求。  7-0：格式较为混乱。 |
| 感想（含思政）  （目标2） | | | 10 |  | 10-8：感想真实具体。  7-5：感想比较空洞。  4-0：没有写什么感想。 |
| 遇到的问题和解决方案，及意见和建议  （目标2） | | | 10 |  | 10-8：问题和解决方案具体，意见和建议有的放矢。  7-5：问题和解决方案不太具体，意见和建议不够明确。  4-0：没有写什么内容。 |
| 实验报告总分 | | | 100 |  |  |
| 教师签名 |  | | | 日期 |  |

**目录**

[5 数组程序设计实验 1](#_Toc404837938)

[5.1 实验目的 1](#_Toc404837921)

[5.2 实验环境 1](#_Toc404837921)

[5.3 实验内容 1](#_Toc404837922)

[5.4 实验小结 1](#_Toc404837923)

[6 指针程序设计实验 7](#_Toc404837948)

[6.1 实验目的 1](#_Toc404837921)

[6.2 实验环境 1](#_Toc404837921)

[6.3 实验内容 1](#_Toc404837922)

[6.4 实验小结 1](#_Toc404837923)

[实验总结 7](#_Toc404837948)

[参考文献 9](#_Toc404837957)

# 5 数组程序设计实验

## 5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

## 5.2实验环境

操作系统：Windows

编程语言：C

集成开发环境IDE：Visual Studio 2022

## 5.3 实验内容

**5.3.1 程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgrmingLnguge

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。
2. 跟踪进入函数 strdelc 时，观察字符数组 s 中的内容和字符 c 的值，分析结果是否正确。单步执行 for 语句过程中，分析该结果是否存在问题。当单步执行光条落在 strdelc 函数块结束标记“}”所在行时，字符串 s为何值？分析是否实现了所要求的删除操作。

/\*实验 5-1 程序改错与跟踪调试题程序\*/

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a); printf("%s %s\n",b,a);
9. strdelc(b, 'a'); printf("%s\n",b);
10. return 0;
11. }
12. void strcate(char t[],char s[])
13. {
14. int i = 0, j = 0;
15. while(t[i++]) ;
16. while((t[i++] = s[j++] )!= '\0');
17. }
18. void strdelc(char s[], char c)
19. {
20. int j,k;
21. for(j=k=0; s[j] != '\0'; j++)
22. if(s[j] != c) s[k++] = s[j];
23. }

**解答：**

（1）跟踪观察函数strcate：

1. 字符数组t和s中的内容：

|  |  |  |
| --- | --- | --- |
| **名称** | **值** | **类型** |
| t | 0x00000002df4ffc68"Programming" | char \* |
| s | 0x00000002df4ffc38 "Language" | char \* |

1. i与t[i]的值：

i=12;t[i]='?'

该结果存在问题：字符串Programming\0实际总共有12个字符，数组最大下标为11，当进行i++操作时，t[12]没有对应字符。

1. 字符数组t和s分别的值：

|  |  |  |
| --- | --- | --- |
| **名称** | **值** | **类型** |
| t | 0x000000a1619ff698"Programming" | char\* |
| s | 0x000000a1619ff668"Language" | char\* |

可知，没有实现字符串的连接

（2）跟踪观察函数 strdelc：

1. 进入strcate时，字符数组 s 中的内容和字符 c 的值：

|  |  |  |
| --- | --- | --- |
| **名称** | **值** | **类型** |
| s | 0x0000005ab977f8e8"Programming" | char\* |
| c | 'a' | char |

1. 单步执行for语句，分析结果：

结果存在问题，最终输出的字符串为Progrmmingg，删除了字符'a'，但是字母g重复

1. 当单步执行光条落在 strdelc 函数块结束标记“}”所在行时，字符串 s的值：

最终输出的字符串为Progrmmingg，删除了字符'a'，但是字母g重复。

1. 基于以上的单步执行观察结果，说明还存在如下逻辑错误：
2. 6 char a[]="Language", b[]="Programming"; 应修改成：

6 char a[50] = "Language", b[50] = "Programming";

给出数组元素数量，防止越界

1. strcate 函数应修改成：

void strcate(char t[], char s[])

{

int i = 0, j = 0;

while (t[i++]);

int m = i - 1;

while ((t[m++] = s[j++]) != '\0');

}

向前一位，防止出现t[12]无对应字符却相连的情况

1. strdelc 函数应修改成：

void strdelc(char s[], char c)

{

int j, k;

for (j = k = 0; s[j] != '\0'; j++)

{

if (s[j] != c)

s[k++] = s[j];

}

s[k] = '\0';

}

强制将字符串末尾转化为\0，防止剩余字符串出现。

1. 修改后的源程序：

#include<stdio.h>

void strcate(char[], char[]);

void strdelc(char[], char);

int main(void)

{

char a[50] = "Language", b[50] = "Programming";

printf("%s %s\n", b, a);

strcate(b, a); printf("%s %s\n", b, a);

strdelc(b, 'a'); printf("%s\n", b);

return 0;

}

void strcate(char t[], char s[])

{

int i = 0, j = 0;

while (t[i++]);

int m = i - 1;

while ((t[m++] = s[j++]) != '\0');

}

void strdelc(char s[], char c)

{

int j, k;

for (j = k = 0; s[j] != '\0'; j++)

{

if (s[j] != c)

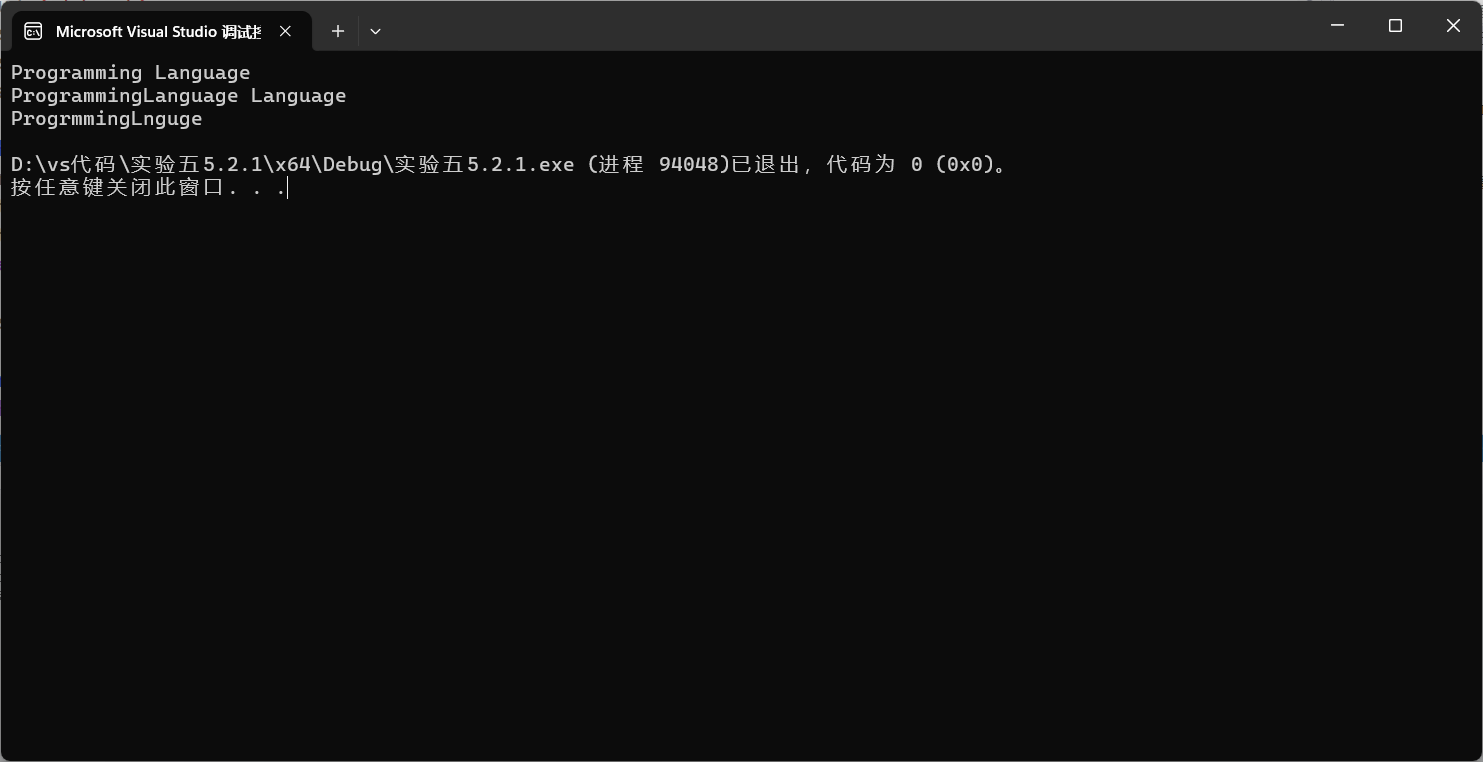
s[k++] = s[j];

}

s[k] = '\0';

}

（4）错误修改后运行结果如图5-1所示：

图5-1 程序改错与跟踪调试题程序修改后运行结果截图

**5.3.2 源程序完善和修改替换**

下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。

1. 请在源程序中的下划线处填写合适的代码来完善该程序。
2. #include<stdio.h>
3. #define M 10
4. #define N 3
5. int main(void)
6. {
7. int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/
8. int i, j, k;
9. for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/
10. a[i] = i + 1;
11. for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

1. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
2. if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/
3. b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/
4. if(j)
5. for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
6. \_\_\_\_\_\_\_\_\_\_\_\_\_\_;
7. }
8. for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/
9. printf(“%6d”, b[i]);
10. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
11. return 0;
12. }

13行分别填入：a[j-1]、a[M - 1]

16行填入：a[k] = a[k + 1]

补充完善后的源程序运行结果如图所示：

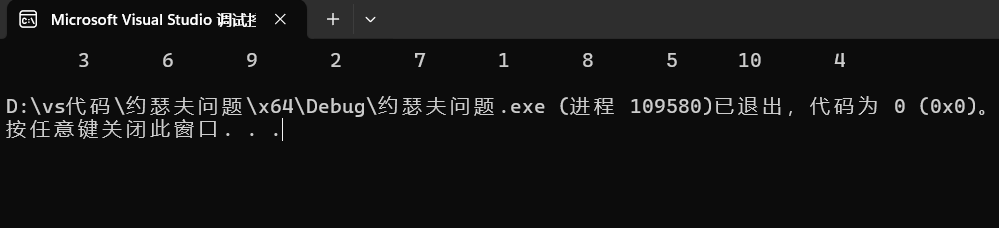


图5-2 源程序完善和修改替换题程序完善后运行结果截图

1. 上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

优化后的源程序：

#include<stdio.h>

#define M 10

#define N 3

int main()

{

int a[M], b[M] = { 0 };

int i, j = 0, k = 0;

for (i = 0; i < M; i++)

a[i] = 1;

i = 0;

while (b[M - 1] == 0)

{

if (i == M) i = 0;

if (a[i] == 1) j++;

if (j == N) {

b[k] = i + 1, a[i] = 0, j = 0, k++;

}

i++;

}

for (i = 0; i < M; i++)

{

printf("%6d", b[i]);

}

printf("\n");

}

优化完善后的源程序运行结果如图所示：

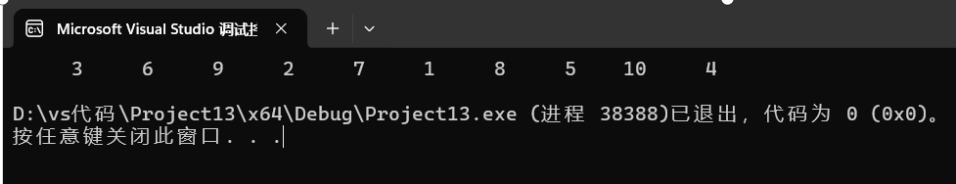


图5-3 源程序完善和修改替换题程序修改后运行结果截图

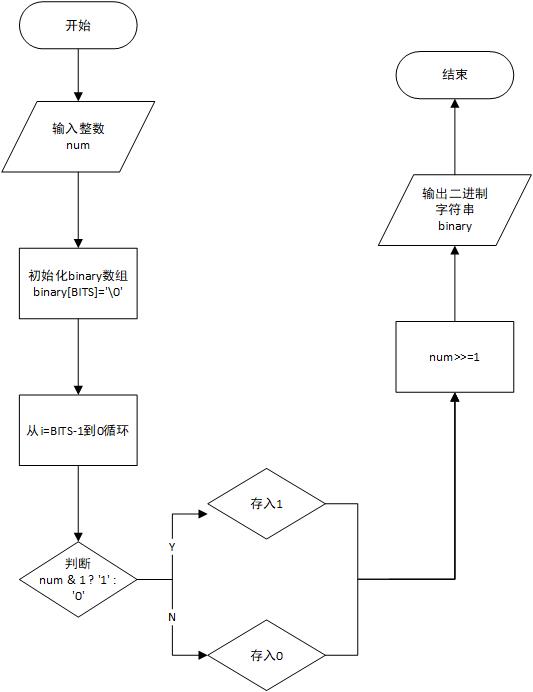
**5.3.3 程序设计**

1. 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符号数组中，然后输出该整数的二进制表示。

例如：7（输入）00000000000000000000000000000111（输出）

**解答：**

1. **设计**。算法流程如图5-4所示。



5-4 程序设计题1的程序流程图

1. **实现。**源程序清单。

#include <stdio.h>

#define BITS 32

int main() {

int num;

char binary[BITS + 1];

scanf("%d", &num);

binary[BITS] = '\0';

for (int i = BITS - 1; i >= 0; i--) {

binary[i] = (num & 1) ? '1' : '0';

num >>= 1;

}

printf("%s\n", binary);

return 0;

}

1. **测试**
2. 测试数据设计：如表5-1所示。

表5-1 程序设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序 输入 | 理 论 结 果 | 运 行 结 果 |
| x |  |  |
| 例1 | 25 | 00000000000000000000000000011001 | 00000000000000000000000000011001 |
| 例2 | 7 | 00000000000000000000000000000111 | 00000000000000000000000000000111 |
| 例3 | 89 | 00000000000000000000000001011001 | 00000000000000000000000001011001 |

## 对应测试数据的运行结果如图5-5所示：

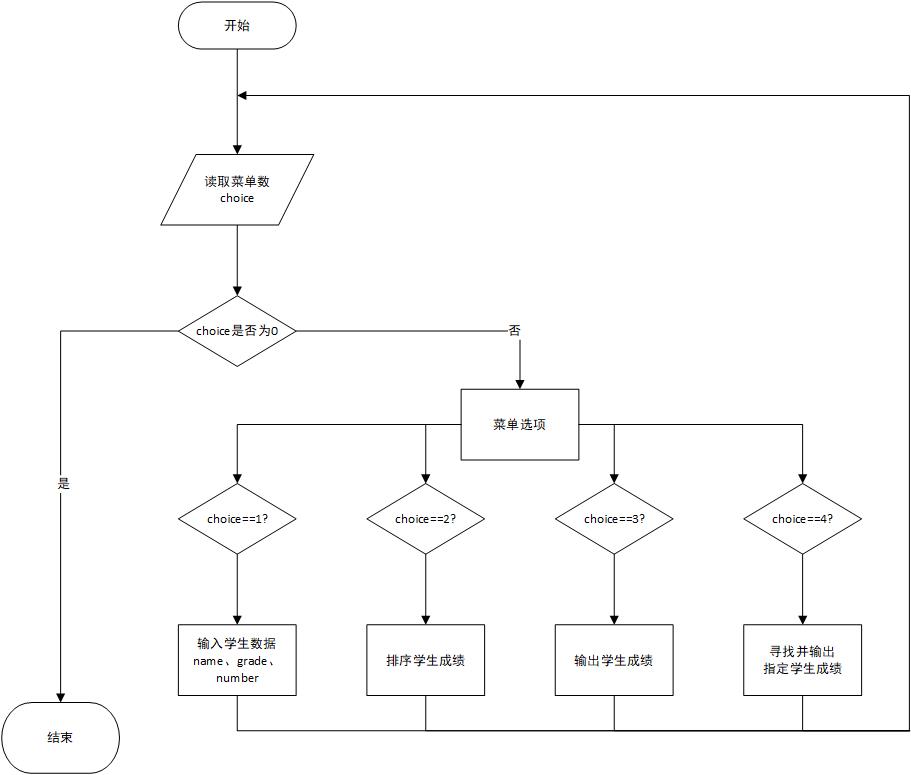
## 

图5-5 设计程序题1的运行截图

1. 编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项(0-4,其中0为退出)。对应菜单选项1-4，该程序具有以下功能：  
   ①“成绩输入”，输入n个学生的姓名和C语言课程的成绩。  
   ②“成绩排序”，将成绩按从高到低的次序排序，姓名同时进行相应调整。成绩相同的，按照输入先后次序排列。  
   ③“成绩输出”，输出排序后所有学生的姓名和C语言课程的成绩。  
   ④“成绩查找”，输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

**解答：**

1. **设计**。算法流程如图5-6所示。



5-6 程序设计题2的程序流程图

1. **实现。**源程序清单。

#include<stdio.h>

#include<string.h>

#define MAX\_STUDENTS 50

void inputscores(char name[][50], int grade[], int\* number);

void reorder(char name[][50], int grade[], int number);

void outputscores(char name[][50], int grade[], int number);

void findscores(char name[][50], int grade[], int number, int score);

int main() {

int choice, number;

char name[MAX\_STUDENTS][50]; // Multi-dimensional array for storing student names

int grade[MAX\_STUDENTS];

do {

scanf("%d", &choice);

switch (choice) {

case 0:

break;

case 1:

scanf("%d", &number);

printf("%d records were input!\n", number);

inputscores(name, grade, &number);

break;

case 2:

printf("Reorder finished!\n");

reorder(name, grade, number);

break;

case 3:

outputscores(name, grade, number);

break;

case 4:

{

int score;

scanf("%d", &score);

findscores(name, grade, number, score);

break;

}

default:

break;

}

} while (choice != 0);

return 0;

}

void inputscores(char name[][50], int grade[], int\* number) {

for (int i = 0; i < \*number; i++) {

scanf("%s", name[i]); // Read the name properly

scanf("%d", &grade[i]);

}

}

void reorder(char name[][50], int grade[], int number) {

for (int i = 0; i < number - 1; i++) {

for (int j = i + 1; j < number; j++) {

if (grade[i] < grade[j]) { // Swap grades and names if necessary

// Swap grades

int temp\_grade = grade[i];

grade[i] = grade[j];

grade[j] = temp\_grade;

// Swap names

char temp\_name[50];

strcpy(temp\_name, name[i]);

strcpy(name[i], name[j]);

strcpy(name[j], temp\_name);

}

}

}

}

void outputscores(char name[][50], int grade[], int number) {

for (int i = 0; i < number; i++) {

printf("%s %d\n", name[i], grade[i]);

}

}

void findscores(char name[][50], int grade[], int number, int score) {

// Perform binary search after sorting the grades

int left = 0, right = number - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (grade[mid] == score) {

printf("%s %d\n", name[mid], grade[mid]);

return;

}

else if (grade[mid] < score) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

printf("not found!\n");

}

1. **测试**
2. 测试数据设计：

例1

输入：

1/\*菜单命令1，表示成绩输入\*/

6/\*表示输入6位同学的成绩\*/

Jack 95/\*输入格式：姓名 成绩。以下连续6行为6位同学成绩\*/

Mike 90

Joe 75

Andy 95

Rose 89

Sophia 77

2/\*菜单命令2，表示成绩排序\*/

3/\*菜单命令3，表示成绩输出\*/

4/\*菜单命令4，表示成绩查找\*/

89/\*表示查找成绩为89的记录\*/

0/\*表示退出\*/

输出：

6 records were input!/\*菜单命令1的输出\*/

Reorder finished!/\*菜单命令2的输出\*/

Jack 95/\*菜单命令3的输出\*/

Andy 95

Mike 90

Rose 89

Sophia 77

Joe 75

Rose 89/\*菜单命令4的输出\*/

例2

输入：

1/\*菜单命令1，表示成绩输入\*/

3/\*表示输入3位同学的成绩\*/

Huang 100/\*输入格式：姓名 成绩。以下连续3行为3位同学成绩\*/

Wu 93

Liao 90

2/\*菜单命令2，表示成绩排序\*/

3/\*菜单命令3，表示成绩输出\*/

4/\*菜单命令4，表示成绩查找\*/

93/\*表示查找成绩为93的记录\*/

0/\*表示退出\*/

输出：

3 records were input!/\*菜单命令1的输出\*/

Reorder finished!/\*菜单命令2的输出\*/

Huang 100/\*菜单命令3的输出\*/

Wu 93

Liao 90

Wu 93/\*菜单命令4的输出\*/

例3

输入：

1/\*菜单命令1，表示成绩输入\*/

7/\*表示输入7位同学的成绩\*/

Huang 100

Liao 91

Wang 99

Chen 93

Wu 80

Xiao 99

An 85

2/\*菜单命令2，表示成绩排序\*/

3/\*菜单命令3，表示成绩输出\*/

4/\*菜单命令4，表示成绩查找\*/

88/\*表示查找成绩为88的记录\*/

0/\*表示退出\*/

输出：

7 records were input!/\*菜单命令1的输出\*/

Reorder finished!/\*菜单命令2的输出\*/

Huang 100/\*菜单命令3的输出\*/

Wang 99

Xiao 99

Chen 93

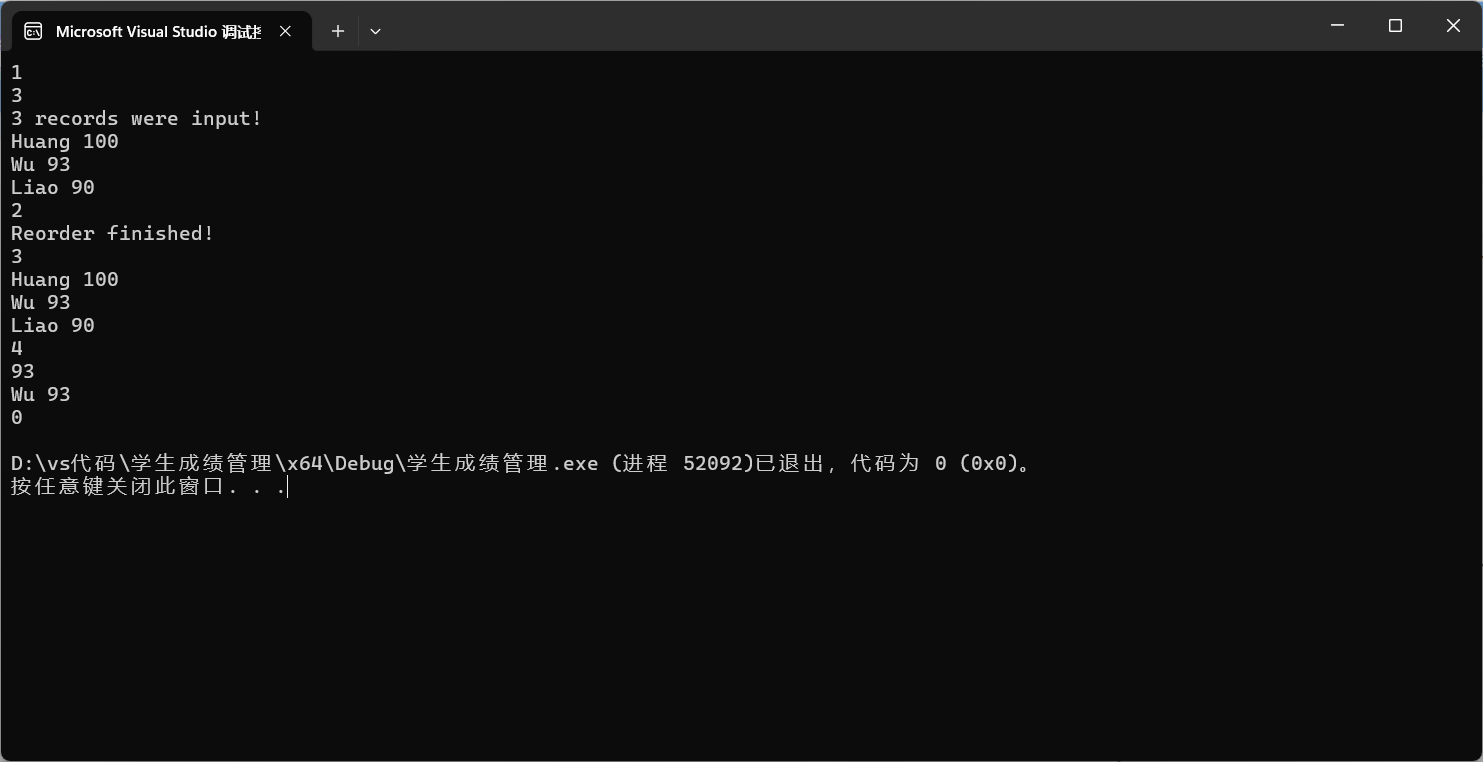
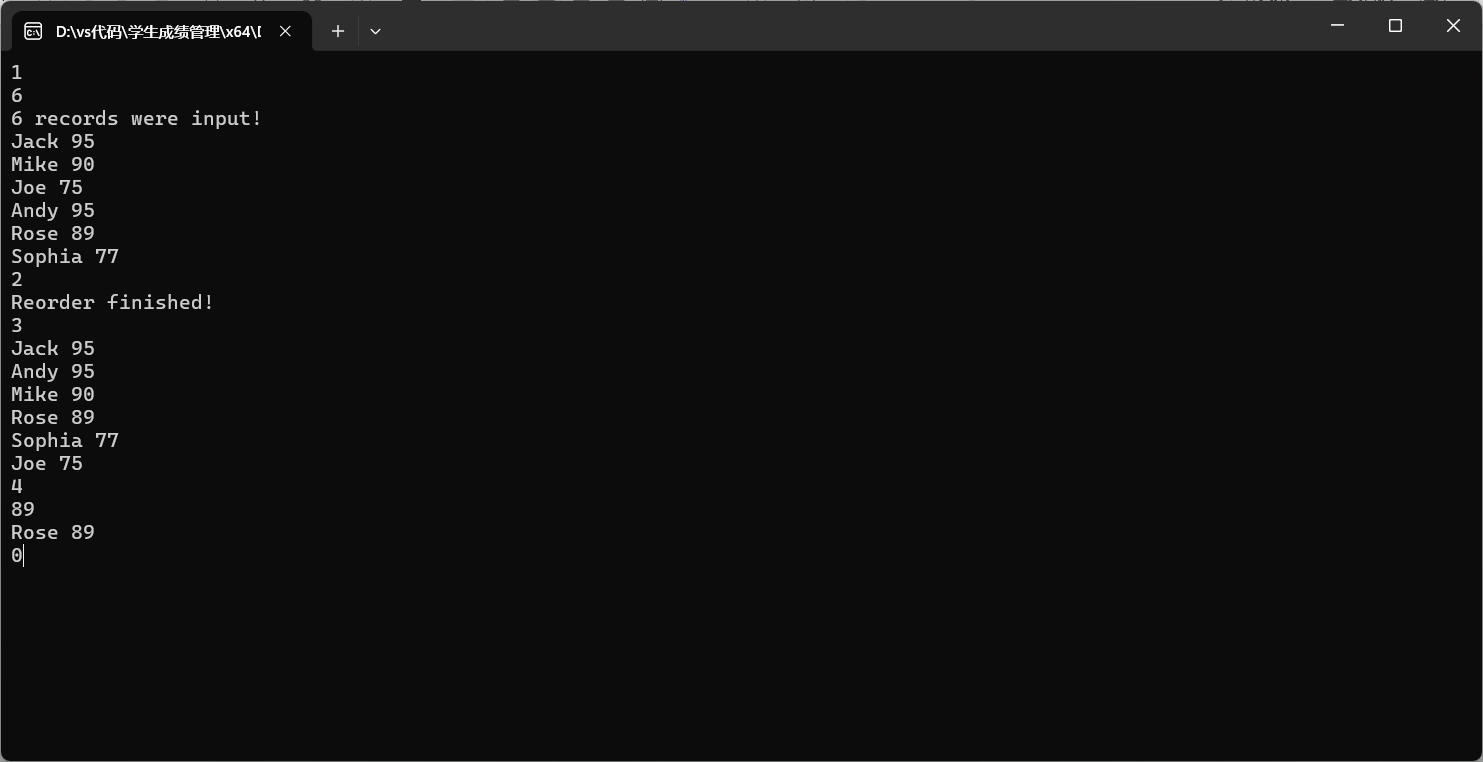
Liao 91

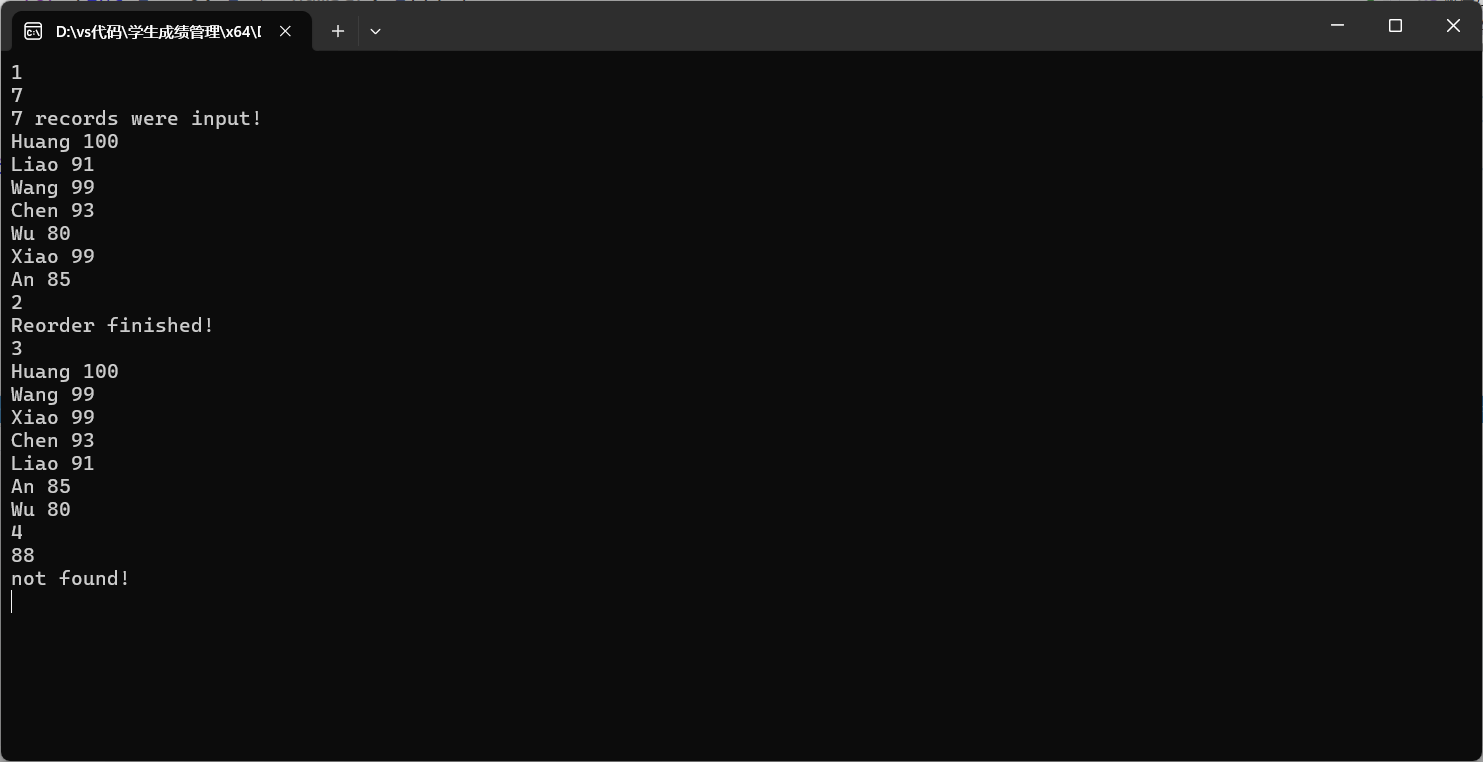
An 85

Wu 80

not found!/\*菜单命令4的输出\*/

## 对应测试数据的运行结果如图5-7所示：

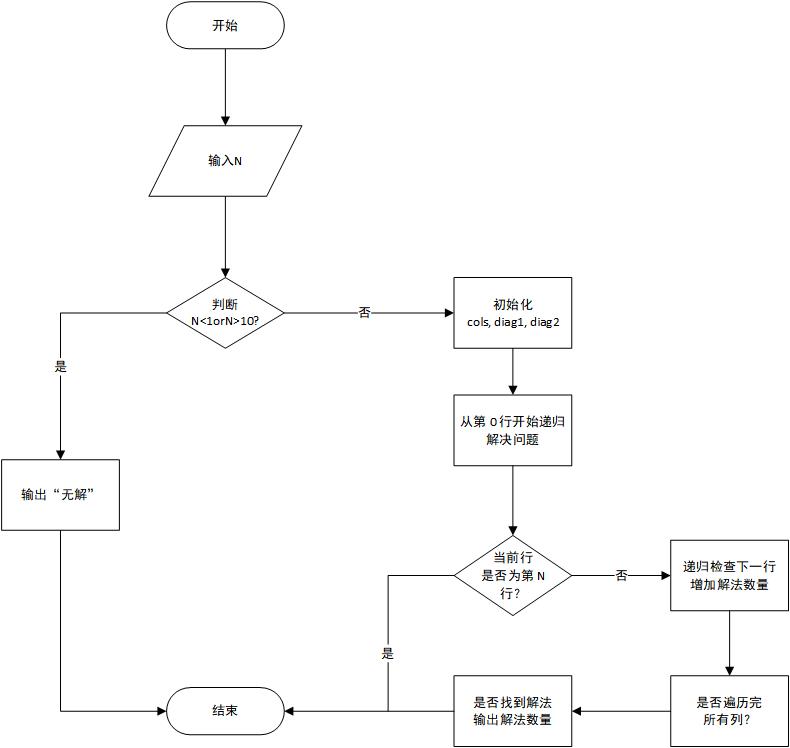


图5-7 设计程序题2的运行截图

1. 求解N皇后问题，即在N×N的棋盘上摆放N个皇后，要求任意两个皇后不能在同一行、同一列、同一对角线上。输入棋盘的大小N（N取值1-10），如果能满足摆放要求，则输出所有可能的摆放法的数量，否则输出“无解”。

**解答：**

1. **设计**。算法流程如图5-8所示。



5-8 程序设计题3的程序流程图

1. **实现。**源程序清单。

#include <stdio.h>

#include <stdlib.h>

// 定义最大棋盘大小

#define MAX\_N 10

// 记录解决方案的数量

int count = 0;

// 检查当前行、列和对角线是否安全

int is\_safe(int row, int col, int cols[], int diag1[], int diag2[]) {

return !cols[col] && !diag1[row - col + MAX\_N - 1] && !diag2[row + col];

}

// 回溯函数，尝试在每一行放置皇后

void solve(int row, int N, int cols[], int diag1[], int diag2[]) {

if (row == N) {

// 找到一种合法的解法

count++;

return;

}

for (int col = 0; col < N; col++) {

if (is\_safe(row, col, cols, diag1, diag2)) {

// 标记当前列和对角线占用

cols[col] = 1;

diag1[row - col + MAX\_N - 1] = 1;

diag2[row + col] = 1;

// 递归尝试下一行

solve(row + 1, N, cols, diag1, diag2);

// 撤销标记，回溯

cols[col] = 0;

diag1[row - col + MAX\_N - 1] = 0;

diag2[row + col] = 0;

}

}

}

int main() {

int N;

scanf("%d", &N);

if (N < 1 || N > 10) {

printf("无解\n");

return 0;

}

// 用于标记列和对角线的占用情况

int cols[MAX\_N] = { 0 };

int diag1[2 \* MAX\_N - 1] = { 0 }; // 主对角线

int diag2[2 \* MAX\_N - 1] = { 0 }; // 副对角线

count = 0;

solve(0, N, cols, diag1, diag2);

if (count == 0) {

printf("无解\n");

}

else {

printf("%d\n", count);

}

return 0;

}

1. **测试**
2. 测试数据设计：如表5-2所示。

表5-2 程序设计题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序 输入 | 理 论 结 果 | 运 行 结 果 |
| x |  |  |
| 例1 | 6 | 4 | 4 |
| 例2 | 10 | 724 | 724 |
| 例3 | 3 | 无解 | 无解 |

1. 对应测试数据的运行结果如图5-9所示：

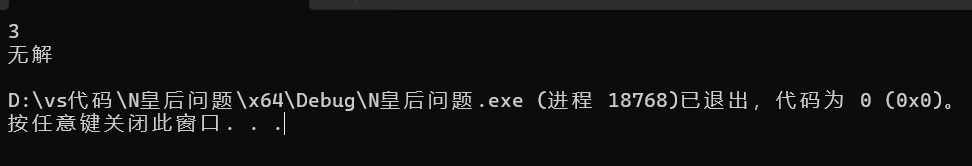
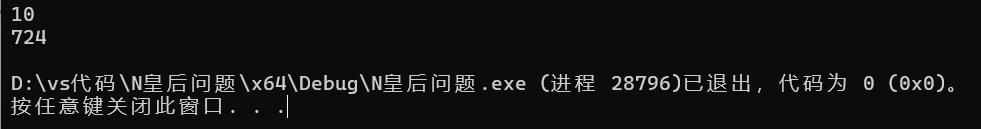
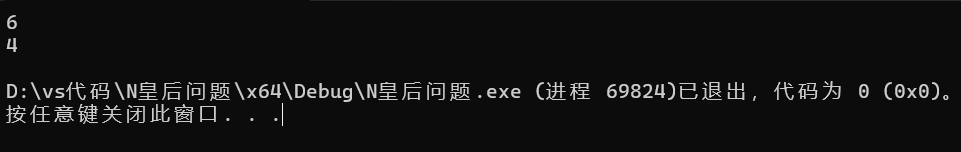
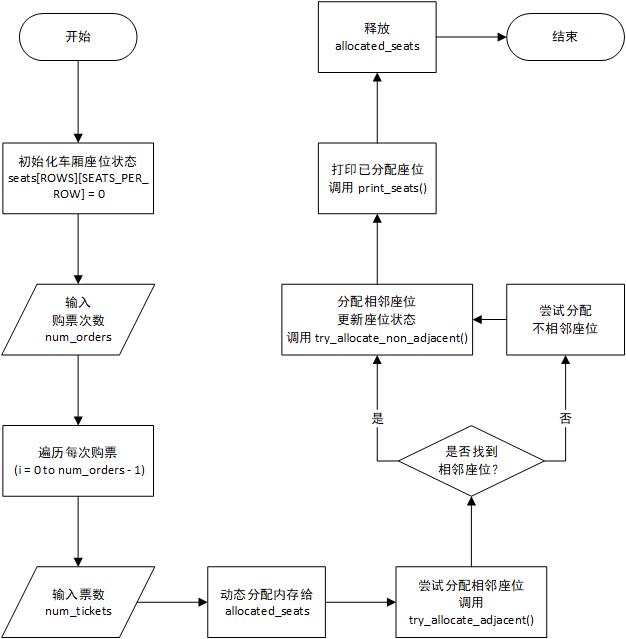


图5-9 设计程序题3的运行截图

1. 实现一个铁路购票系统的简单分配算法，用来处理一节车厢的座位分配。假设一节车厢有20排，每一排有5个座位，用A、B、C、D、F表示，第一排是1A、1B、1C、1D、1F，第二排是2A、2B、2C、2D、2F，以此类推，第20排是20A、20B、20C、20D、20F。购票时，每次最多购买5张，座位的分配策略是：如果这几张票能安排在同一排相邻座位，则应该安排在编号最小的相邻座位；否则，应该安排在编号最小的几个空座位中（不考虑是否相邻）。

**解答：**

1. **设计**。算法流程如图5-10所示。



5-10 程序设计题4的程序流程图

1. **实现。**源程序清单。

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#define ROWS 20 // 车厢的排数

#define SEATS\_PER\_ROW 5 // 每排座位数（修改为5个座位）

// 模拟车厢座位 20排5个座位，0代表空座，1代表已占座

int seats[ROWS][SEATS\_PER\_ROW] = { 0 }; // 初始化时所有座位都为空

void print\_single\_seats(char row\_label[3], int row, int seat) {

char m;

switch (seat) {

case 0: case 1: case 2: case 3:

m = 'A' + seat;

break;

case 4:

m = 'A' + seat + 1; // 对应 A、B、C、D、F

break;

}

row\_label[0] = (row + 1);

row\_label[1] = m;

row\_label[2] = '\0';

printf("%d%c", row\_label[0], row\_label[1]);

}

// 输出座位分配情况

void print\_seats(int\* seats, int num\_seats) {

char row\_label[3]; // 用于存储行号和座位标记

for (int i = 0; i < num\_seats; i++) {

int row = seats[i] / SEATS\_PER\_ROW; // 行号

int seat = seats[i] % SEATS\_PER\_ROW; // 座位号

// 生成行号和座位号标签，座位号是 'A' 到 'F'

// 'A' + seat 确保是 'A' 到 'F'

if (i != 0) {

printf(" "); // 除了第一个座位外，前面要加空格

} // 输出每个座位的标签

print\_single\_seats(row\_label, row, seat);

}

}

// 尝试在同一排找相邻的空座位

bool try\_allocate\_adjacent(int num\_tickets, int\* allocated\_seats) {

for (int row = 0; row < ROWS; row++) {

int count = 0;

for (int col = 0; col < SEATS\_PER\_ROW; col++) {

if (seats[row][col] == 0) { // 发现空座

allocated\_seats[count++] = row \* SEATS\_PER\_ROW + col;

if (count == num\_tickets) {

// 找到了足够的相邻空座

for (int i = 0; i < num\_tickets; i++) {

seats[allocated\_seats[i] / SEATS\_PER\_ROW][allocated\_seats[i] % SEATS\_PER\_ROW] = 1;

}

return true;

}

}

else {

count = 0; // 如果发现非空座位，重置相邻座位计数

}

}

}

return false;

}

// 尝试在整个车厢内找空座位

void try\_allocate\_non\_adjacent(int num\_tickets, int\* allocated\_seats) {

int count = 0;

for (int row = 0; row < ROWS; row++) {

for (int col = 0; col < SEATS\_PER\_ROW; col++) {

if (seats[row][col] == 0) {

allocated\_seats[count++] = row \* SEATS\_PER\_ROW + col;

if (count == num\_tickets) {

for (int i = 0; i < num\_tickets; i++) {

seats[allocated\_seats[i] / SEATS\_PER\_ROW][allocated\_seats[i] % SEATS\_PER\_ROW] = 1;

}

return;

}

}

}

}

}

int main() {

int num\_orders; // 购票次数

scanf("%d", &num\_orders);

for (int i = 0; i < num\_orders; i++) {

int num\_tickets;

scanf("%d", &num\_tickets);

// 动态分配内存给 allocated\_seats 数组

int\* allocated\_seats = (int\*)malloc(num\_tickets \* sizeof(int));

if (allocated\_seats == NULL) {

printf("内存分配失败\n");

return 1; // 内存分配失败，退出程序

}

if (!try\_allocate\_adjacent(num\_tickets, allocated\_seats)) {

try\_allocate\_non\_adjacent(num\_tickets, allocated\_seats);

}

print\_seats(allocated\_seats, num\_tickets);

// 如果不是最后一次购票，输出换行符

if (i != num\_orders - 1) {

printf("\n");

}

// 释放动态分配的内存

free(allocated\_seats);

}

return 0;

}

1. **测试**
2. 测试数据设计：如表5-3所示。

表5-3 程序设计题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| X  Y |  |  |
| 例1 | 1  5 | 1A 1B 1C 1D 1F | 1A 1B 1C 1D 1F |
| 例2 | 4  2 5 4 2 | 1A 1B  2A 2B 2C 2D 2F  3A 3B 3C 3D  1C 1D | 1A 1B  2A 2B 2C 2D 2F  3A 3B 3C 3D  1C 1D |
| 例3 | 6  5 3 1 4 2 4 | 1A 1B 1C 1D 1F  2A 2B 2C  2D  3A 3B 3C 3D  4A 4B  5A 5B 5C 5D 5F | 1A 1B 1C 1D 1F  2A 2B 2C  2D  3A 3B 3C 3D  4A 4B  5A 5B 5C 5D 5F |

1. 对应测试数据的运行结果如图5-11所示：

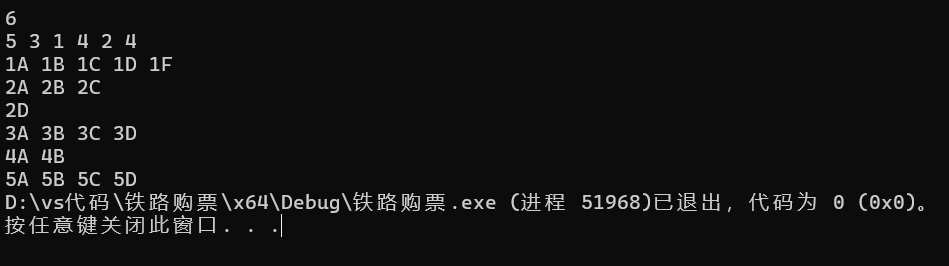
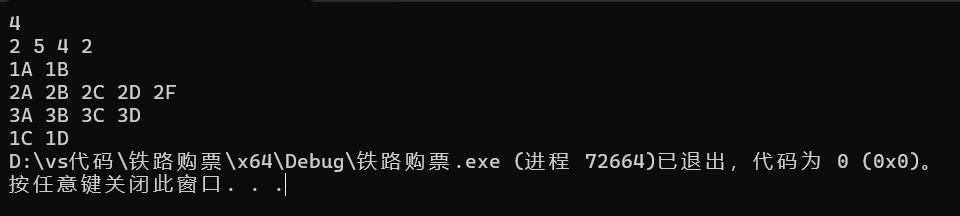
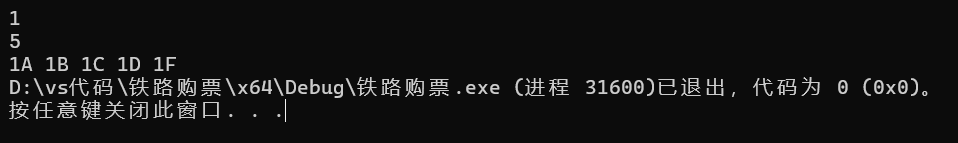


图5-11 设计程序题4的运行截图

1. 将数组中指定的两段数据交换。输入n个整数到数组a中，再输入正整数m1、n1、m2、n2（0≤m1≤n1<m2≤n2<n），将数组中由m1、n1指定的一段数据和由m2、n2指定的一段数据交换位置，其它数据位置不变，输出重新排列后的数组元素。  
   要求：1. 将交换数组中两段数据的功能定义为函数。2. 所有的操作都在数组a上完成，不允许使用其它数组。3.输出在main函数中完成，不能在自定义函数中完成。

**解答：**

1. **设计。**算法流程如图5-12所示。

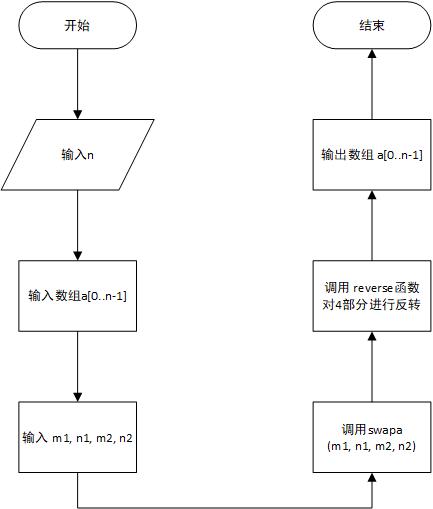
****

图5-12 程序设计题5的程序流程图

1. **实现。**源程序清单。

#include <stdio.h>

int a[1000];

void reverse(int start, int end)

{

while (start < end)

{

int temp = a[start];

a[start] = a[end];

a[end] = temp;

start++;

end--;

}

}

void swapa(int m1, int n1, int m2, int n2)

{

reverse(m1, n1);

reverse(m2, n2);

reverse(m1, n2);

reverse(m1 + n2 - m2 + 1, n2 - (n1 - m1) - 1);

}

int main()

{

int n;

scanf("%d", &n);

for (int i = 0; i < n; i++)

scanf("%d", &a[i]);

int m1, n1, m2, n2;

scanf("%d%d%d%d", &m1, &n1, &m2, &n2);

swapa(m1, n1, m2, n2);

for (int i = 0; i < n; i++)

{

printf("%d ", a[i]);

}

return 0;

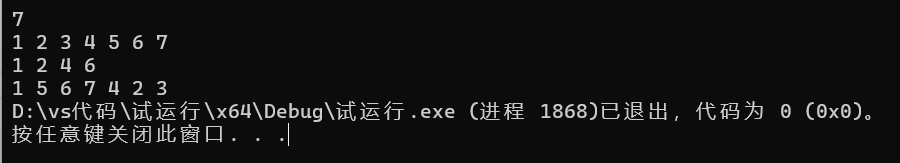
}

1. **测试**
2. 测试数据设计：如表5-4所示。

表5-4 程序设计题5的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| X  Y  Z |  |  |
| 例1 | 7  1 2 3 4 5 6 7  1 2 4 6 | 1 5 6 7 4 2 3 | 1 5 6 7 4 2 3 |
| 例2 | 5  2 1 6 3 7  2 2 3 3 | 2 1 3 6 7 | 2 1 3 6 7 |
| 例3 | 4  -3 2 0 -1  0 0 1 3 | 2 0 -1 -3 | 2 0 -1 -3 |

1. 对应测试数据的运行结果如图5-13所示：



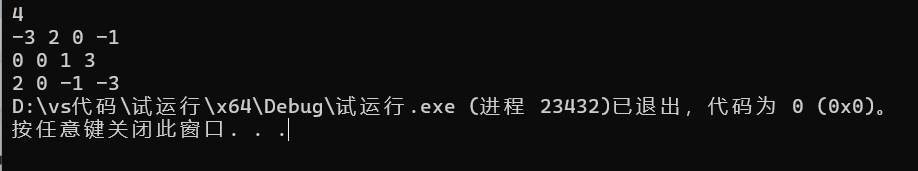


图5-13 程序设计题5的运行截图

## 5.4 实验小结

在本次实验中，我们主要进行了数组程序设计实验，包括程序改错与跟踪调试、源程序完善和修改替换、程序设计等多个环节。以下是实验过程中遇到的问题及解决方案的总结。

1. 程序改错与跟踪调试：

在strcate函数中，我们发现数组下标越界的问题，通过调整循环条件和数组下标解决了这一问题。

在strdelc函数中，我们发现字符删除后未正确处理字符串末尾，通过在函数末尾添加`s[k] = '\0';`解决了这一问题。

2. 源程序完善和修改替换：

在约瑟夫问题中，我们通过数组压缩的方式实现了圈中人的编号和出圈人的编号的跟踪，但这种方法效率较低。后来我们采用了标记法，提高了程序的效率。

3. 程序设计：

在设计整数二进制表示的程序时，我们通过位运算将整数的每一位转化为对应的数字字符，并存放到字符数组中。

在设计成绩管理系统时，我们采用了模块化程序设计思想，将功能分解为输入、排序、输出和查找等模块，并通过函数实现这些功能。

在解决N皇后问题时，我们通过回溯法找到了所有可能的摆放法，并计算出解决方案的数量。

在铁路购票系统的设计中，我们实现了一个简单的座位分配算法，优先分配同一排的相邻座位，如果无法满足，则分配编号最小的几个空座位。

在实验过程中，我们深刻体会到了算法设计的重要性以及调试工具的便利性。通过不断的测试和调试，我们提高了解决问题的能力，并加深了对数组操作和程序设计的理解。

# 实验总结

感想

通过本次实验，我深刻体会到了程序设计的复杂性和挑战性，同时也感受到了解决问题后的成就感。在解决瑟夫问题和N皇后问题时，我意识到了算法的重要性，以及在面对复杂问题时需要的耐心和细致。此外，通过铁路购票系统的设计，我认识到了程序设计在实际生活中的应用价值，增强了我的社会责任感和实践能力。

意见及建议

1. 实验指导书的改进：建议实验指导书能够提供更多的示例代码和调试技巧，帮助学生更快地理解和掌握实验内容。

2. 实验环境的优化：建议实验室提供更多的调试工具和资源，以便学生能够更高效地进行程序调试。

3. 实验内容的丰富：建议增加更多的实际应用案例，让学生能够将理论知识与实际问题相结合，提高解决实际问题的能力。

4. 思政教育的融入：建议在实验教学中融入更多的思政元素，如通过讨论程序设计在社会发展中的作用，增强学生的国家意识和社会责任感。

通过本次实验，我不仅提高了编程技能，也增强了解决实际问题的能力。希望在未来的学习中，能够将这些知识和技能应用到更多的领域中，为社会发展做出贡献。