

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2022年 月 日**

**网络空间安全学院**

**目 录**

[5 数组程序设计实验 1](#_Toc120285342)

[5.1 实验目的 1](#_Toc120285343)

[5.2 实验内容 1](#_Toc120285344)

[5.3 实验小结 27](#_Toc120285345)

[参考文献 27](#_Toc120285346)

# 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 实验内容

**5.2.1 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，观察字符数组s, j和k值的变化，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）跟踪分析源程序strcate函数：

1) 进入strcate时，字符数组t的内容为：”Programming\0”，字符数组s的内容为：”Lnguage\0”；当光标落在第二个while语句所在行时，i为：12；t[i]为：空字符’\0’;

程序的问题：进入第二个while语句时，i的位置指向了字符串t末尾空字符的后面一个空字符，应当使i指向末尾的第一个空字符。

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为：”Programming\0Language\0”，字符数组s的内容为：”Language\0”，由于字符数组t中有多余的空字符’\0’存在，未实现字符串连接。

2）程序修改：

在源程序第15行后添加语句：i++;

使第一个while结束i的位置指向字符串t末尾的第一个空字符。

（2）跟踪分析源程序strdec函数：

1) 进入strdelc时，字符数组s的内容为：” ProgrammingLanguage\0”，字符c的值为：’a’； 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：” ProgrmmingLngugeage\0”，未实现所要求的删除操作。

2）程序修改：

在源程序第22行后添加语句：s[k] = ‘\0’;

程序在for语句执行结束后，k的位置应当为字符串s的末尾，应修改为空字符’\0’。

（3）错误修改后运行结果如图2-1所示。

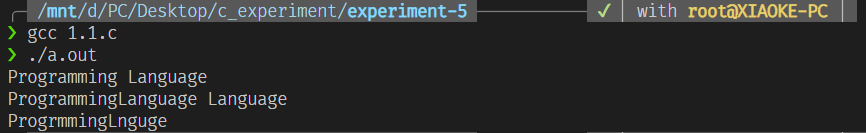


图2-1 实验1-1修改后运行结果截图

**5.2.2 源程序完善和修改替换**

(1) 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。

1 #include<stdio.h>

2 #define M 10

3 #define N 3

4 int main(void)

5 {

6 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 \*/

7 int i, j, k;

8 for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

9 a[i] = i + 1;

10 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 \*/

11 for(k = 1; k <= N; k++) /\* 1至N报数 \*/

12 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 \*/

13 b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/

14 if(j)

15 for(k = --j; k < i-1; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

16 \_\_\_\_\_\_\_\_\_\_\_\_\_\_;

17 }

18 for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

19 printf(“%6d”, b[i]);

20 printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/

21 return 0;

22 }

②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。

**解答：**

1. 填入代码完善源程序：
2. 第13行处第一个空位：a[j - 1]
3. 第13行处第二个空位：a[I - 1]
4. 第16行处：a[k] = a[k + 1]
5. 完善代码后程序运行结果如图2-2所示。

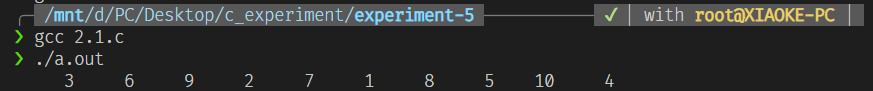


图2-2 实验2-1完善代码后结果截图

1. 采用标记的方法修改源程序：
2. 修改思路：使用一个元素个数与人数相等的out数组记录每个人是否出圈，如第i个人已经出圈，则out[1] = 1，否则out[i] = 0；当有人出圈时，修改对应的out数组标记即可，当轮到一个人报数时，首先先检查其在out数组中的标记，如果标记为1，即已出圈，则下一个人报数。
3. 修改后的源程序如下：

/\*使用标记法修改后的源程序\*/

1 #include <stdio.h>

2 #define M 10

3 #define N 3

4 int main(void)

5 {

6 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 \*/

7 int out[M] = {0}; // 标记出圈的人

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 = -1; i > 1; i--)

12 {

/\* 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 \*/

13 for (k = 1; k <= N; k++) /\* 1至N报数 \*/

14 {

15 j++;

16 while (out[j] == 1 || j >= M)

17 {

18 j++;

19 f (j >= M) /\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

20 j = 0;

21 }

22 }

23 out[j] = 1;

24 b[M - i] = a[j]; /\* 将报数为N的人的编号存入数组b \*/

25 }

26 for (i = 0; i < M - 1; i++) /\* 按次序输出出圈人的编号 \*/

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

28 for (i = 0; i < M - 1; i++)

29 {

30 if (out[i] == 0)

31 printf("%6d\n", a[i]); /\* 输出圈中最后一个人的编号 \*/

32 }

33 return 0;

34 }

1. 使用标记法修改代码后程序运行结果如图2-3所示。

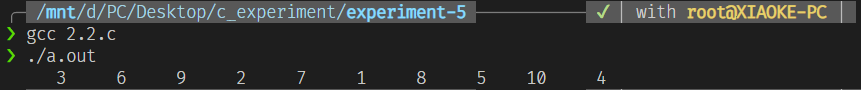


图2-3 实验2-2修改代码后结果截图

**5.2.3 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数组中，然后输出该整数的二进制表示。

**解答：**

1. 算法流程如图2-4所示。

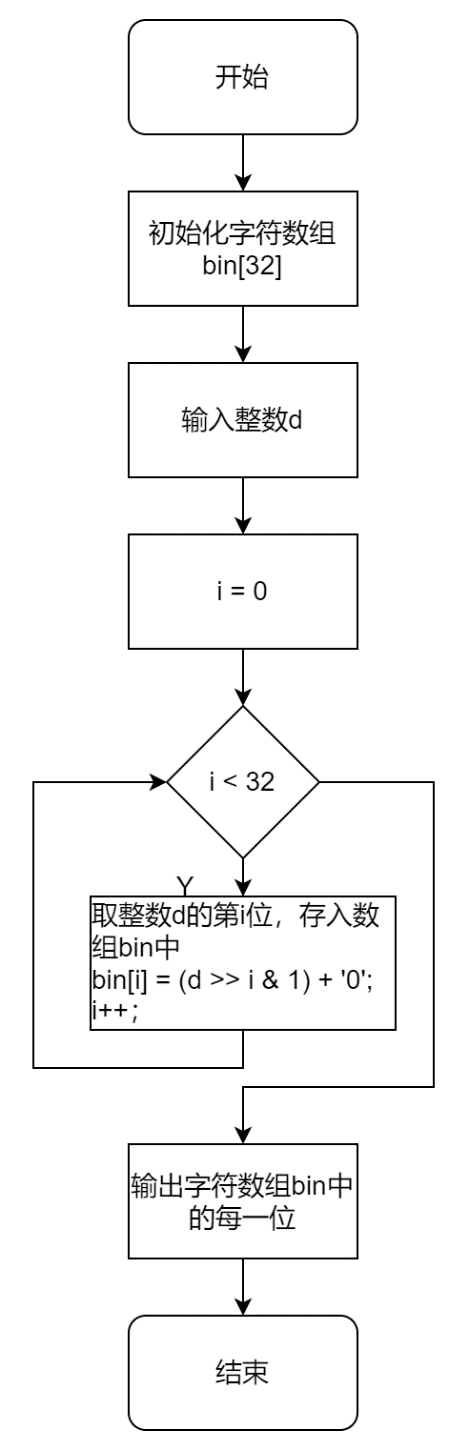


图2-4 程序设计题1的程序流程图

2）源程序清单

1 #include <stdio.h>

2 int main()

3 {

4 int d;

5 char bin[32];

6 scanf("%d", &d);

7 for (int i = 0; i < 32; i++) // 从低位到高位保存

8 {

9 bin[i] = (d >> i & 1) + '0';

10 }

11 for (int i = 0; i < 32; i++) printf("%c", bin[31 - i]);

12 return 0;

13 }

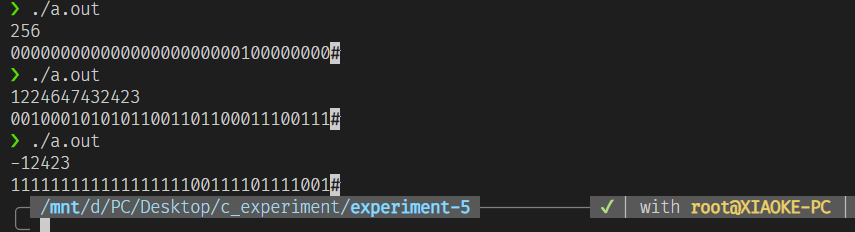
3）测试

（a） 测试数据：

表2-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| 用例1 | 256 | 00000000000000000000000100000000 |
| 用例2 | 1224647432423 | 00100010101011001101100011100111 |
| 用例3 | -12423 | 11111111111111111100111101111001 |

（b） 对应测试数据的运行结果截图

图2-5 程序设计题1的测试用例的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（2）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. 算法流程图

主程序部分流程图如图2-6所示。

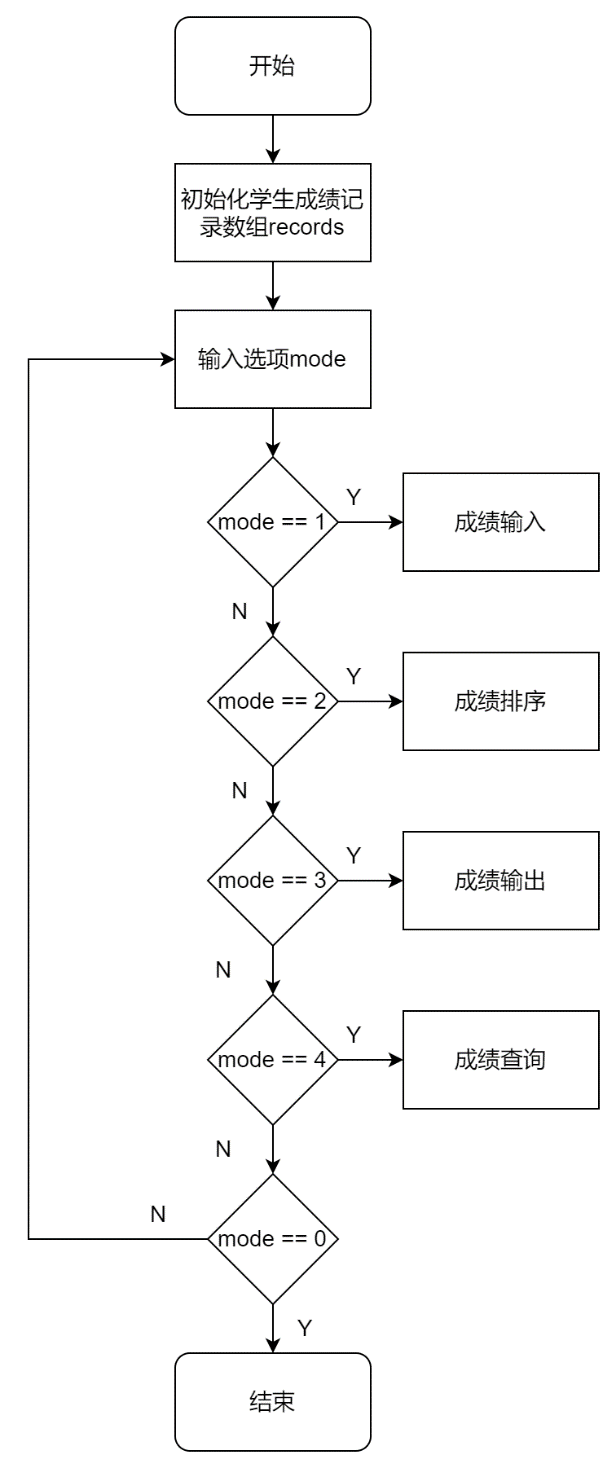


图2-6 程序设计题2主程序流程图

成绩输入部分流程图如图2-7（左）所示。

成绩排序部分流程图如图2-7（右）所示。

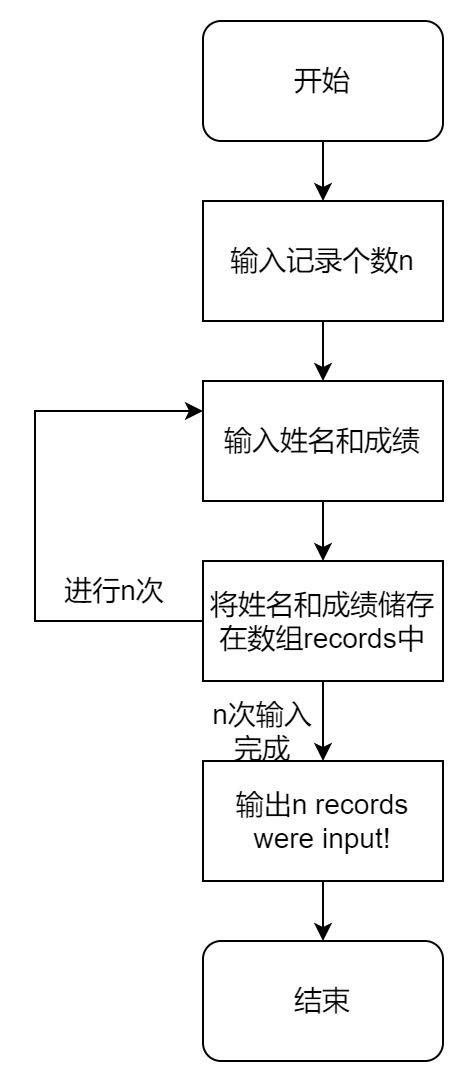
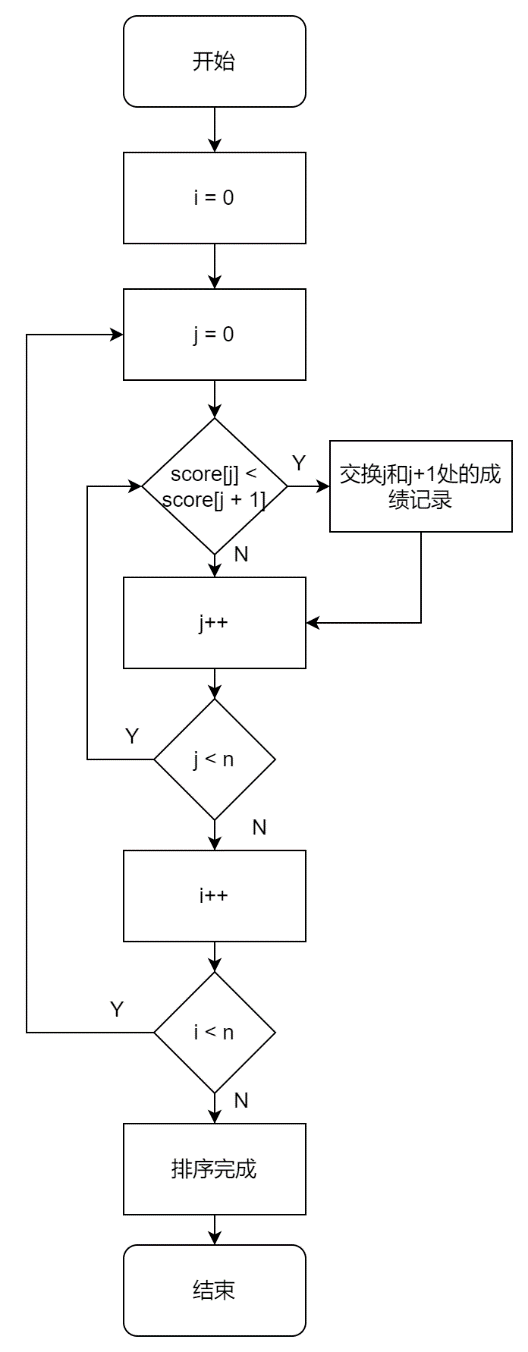
 

图2-7 程序设计题2成绩输入（左）和排序（右）流程图

成绩输出部分流程图如图2-8所示。

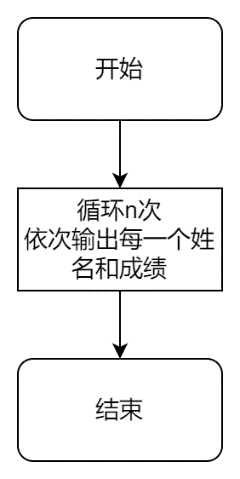


图2-8 程序设计题2输出成绩流程图

成绩二分查找查找部分流程图如图2-9所示。

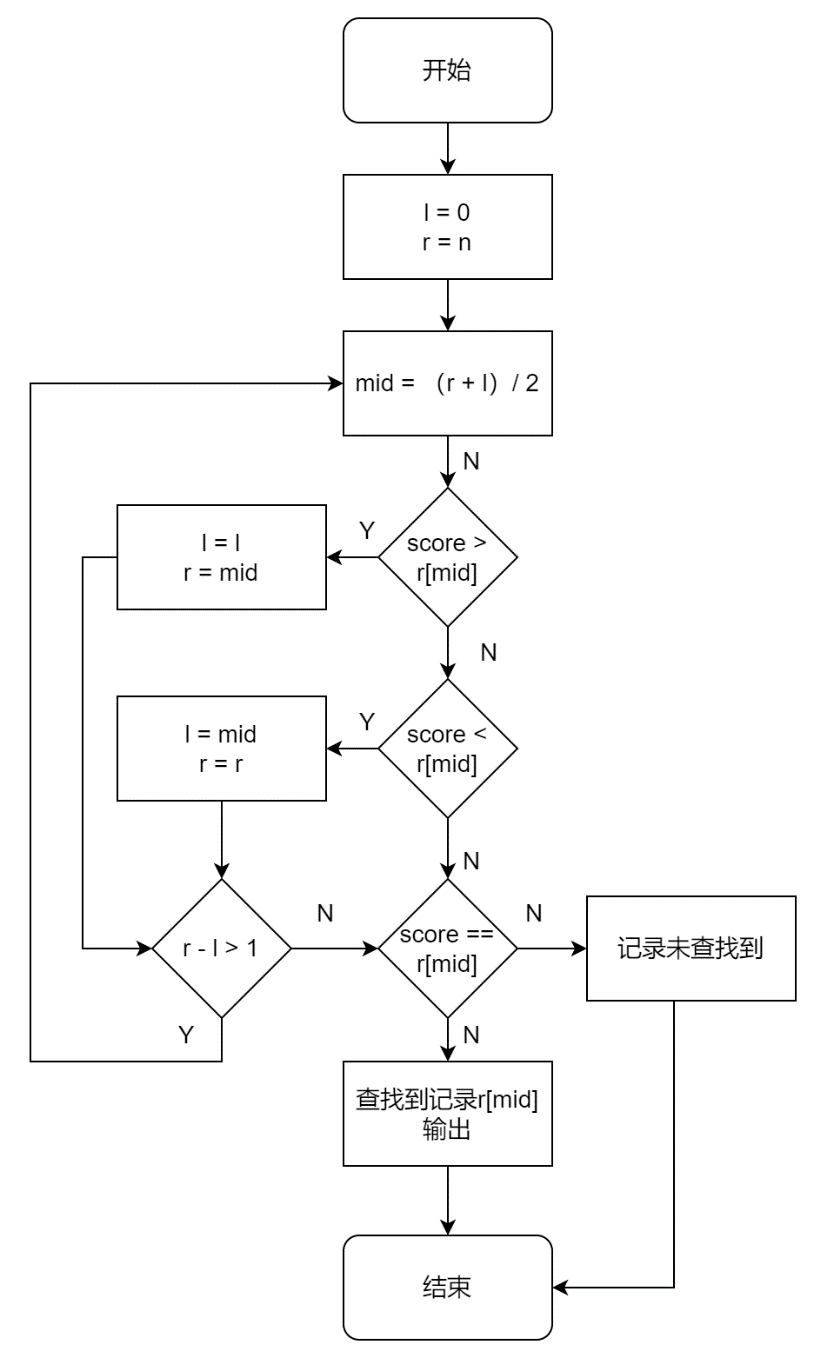


图2-9 程序设计题2成绩二分查找流程图

2）程序清单

#include <stdio.h>

#include <string.h>

#include <malloc.h>

typedef struct stu // 使用结构体存储每一个学生的姓名和成绩

{

char name[20];

int score;

} stu;

int inputRecord(stu \*r, int n); // 成绩输入

void reorder(stu \*r, int n); // 成绩排序

void outputRecord(stu \*r, int n); // 输出成绩

void findRecord(stu \*r, int n); // 查找指定成绩

int main()

{

int mode, n = 0;

stu records[20]; // 结构体数组存储所有小和尚信息

while (scanf("%d", &mode)) // 输入选项

{

switch (mode) // 判断为某一个选项

{

case 1:

n += inputRecord(records, n); // 选项1，输入记录

break;

case 2:

reorder(records, n); // 选项2，对记录排序

break;

case 3:

outputRecord(records, n); // 选项3，输出记录

break;

case 4:

findRecord(records, n); // 选项4，查找记录

break;

}

if (mode == 0) // 结束循环

break;

}

return 0;

}

int inputRecord(stu \*r, int a)

{

stu \*k = r + a;

int n, i;

scanf("%d", &n); // 输入人数n

for (i = 0; i < n; i++) // 以此输入每一个人的名字和成绩

{

scanf("%s %d", k->name, &(k->score));

k++;

}

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

return n;

}

void reorder(stu \*r, int n)

{

int i, j;

for (i = 0; i < n; i++) // 使用冒泡排序对成绩进行排列

{

for (j = 0; j < n - 1; j++)

{

if ((r + j)->score < (r + j + 1)->score) // 按从小到大顺序排序

{

int tmpd = (r + j)->score;

char tmpc[20];

strcpy(tmpc, (r + j)->name); // 交换位置

(r + j)->score = (r + j + 1)->score;

(r + j + 1)->score = tmpd;

strcpy((r + j)->name, (r + j + 1)->name);

strcpy((r + j + 1)->name, tmpc);

}

}

}

printf("Reorder finished!\n");

}

void outputRecord(stu \*r, int n)

{

for (int i = 0; i < n; i++) // 循环输出数组中所有记录

{

printf("%s %d\n", (r + i)->name, (r + i)->score);

}

}

void findRecord(stu \*re, int n)

{

int score;

scanf("%d", &score); // 输入要查找的成绩

int l = 0, r = n, mid; // 二分查找

while (r - l > 1)

{

mid = (r + l) / 2;

if (score > (re + mid)->score) // 大于中间值，向左查找

{

l = l;

r = mid;

}

else if (score < (re + mid)->score) // 小于中间值，向右查找

{

l = mid;

r = r;

}

else // 等于中间值或未找到

{

break;

}

}

if (score == (re + mid)->score) // 等于中间值，输出记录

{

printf("%s %d\n", (re + mid)->name, (re + mid)->score);

}

else // 未找到

{

printf("not found!\n");

}

}

3）测试

（a） 测试数据：

测试输入：

1

6

Jack 95

Mike 90

Joe 75

Andy 95

Rose 89

Sophia 87

2

3

0

预期输出：

6 records were input!

Reorder finished!

Jack 95

Andy 95

Mike 90

Rose 89

Sophia 87

Joe 75

（b） 对应测试测试用例的运行结果如图2-10所示。

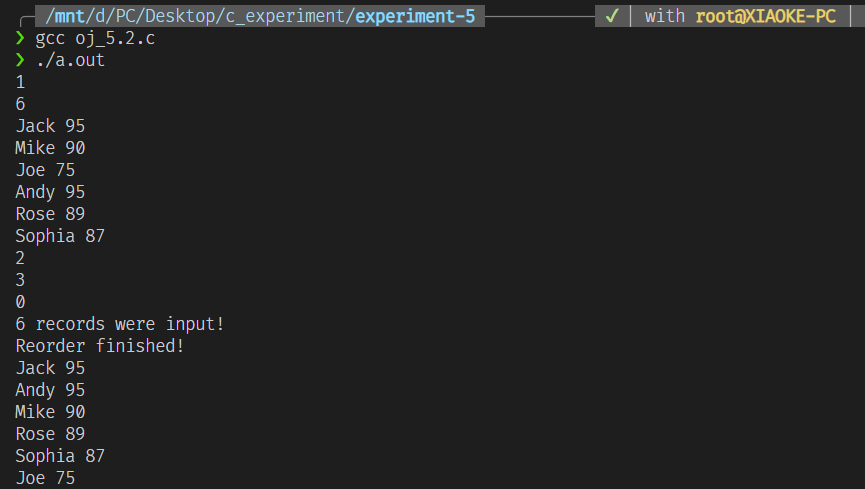


图2-10 程序设计题2测试用例运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）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”。

1) 解题思路

1.向第i行放置皇后时，需要考虑第i行的每一列是否能够放置皇后

2.如果第j列的前i-1行都没有皇后，且经过i，j的左对角线、右对角线上都没有皇后，则位置i，j可以放置皇后

3.重复2判断下一列i，j+1位置能否放置皇后

4.有能够放置皇后的位置后，重复1、2、3判断第i+1行能否放置皇后，直到最后一行的最后一列

5.每再最后一行成功放置一个皇后时，结果ans加一

2）程序流程图

主程序部分如图2-11所示。

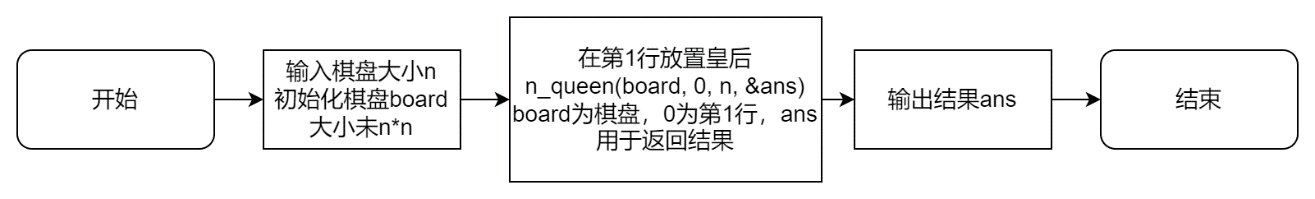


图2-11 程序设计题3主程序流程图

在第i行放置皇后部分如图2-12所示。

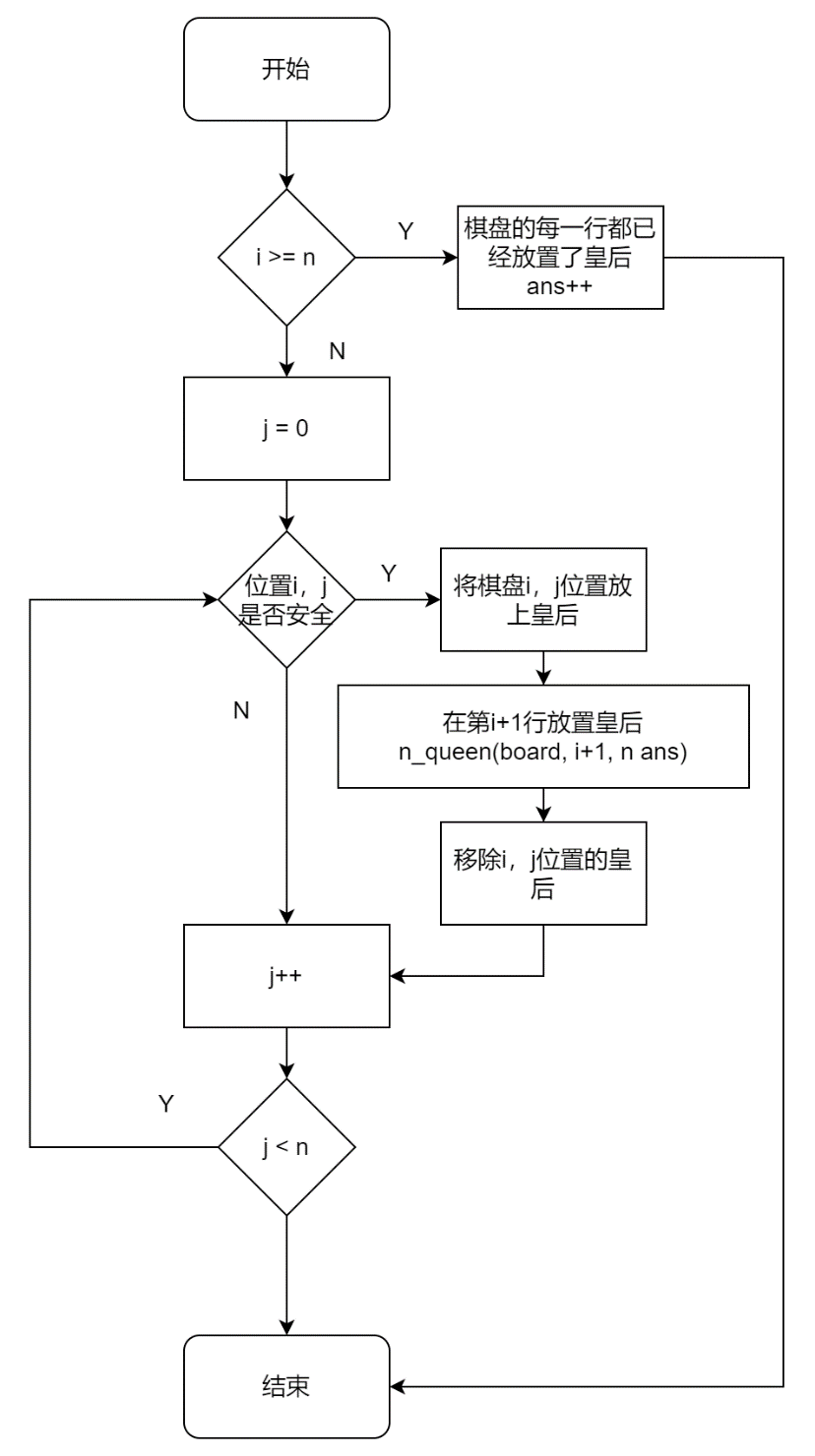


图2-12在第i行放置皇后的流程图

3）程序清单

#include <stdio.h>

#include <malloc.h>

typedef char bool;

#define true 1

#define false 0

bool is\_safe(bool \*board, int i, int j, int n); // 检查第i行第j列是否安全

void n\_queen(bool \*board, int i, int n, int \*ans); // 第i行皇后

int main()

{

int n;

scanf("%d", &n);

int ans = 0; // 记录结果

bool \*board = malloc(sizeof(bool) \* n \* n); // 棋盘

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

{

\*(board + i) = false;

}

n\_queen(board, 0, n, &ans);

if (ans > 0)

{

printf("%d", ans);

}

else

{

printf("无解");

}

}

void n\_queen(bool \*board, int i, int n, int \*ans)

{

if (i >= n) // 递归结束

{

\*ans = \*ans + 1;

}

else

{

int j;

for (j = 0; j < n; j++)

{

if (is\_safe(board, i, j, n)) // 如果当前位置可以放置皇后，则继续检查下一行放置皇后的位置

{

\*(board + i \* n + j) = true; // 在此位置放置皇后

n\_queen(board, i + 1, n, ans); // 进行下一行

\*(board + i \* n + j) = false; // 将皇后取出，检查下一列

}

}

}

}

bool is\_safe(bool \*board, int i, int j, int n)

{

int m, k;

// 检查列上是否有皇后

for (m = i - 1, k = j; m >= 0; m--)

{

if (\*(board + m \* n + k)) // 上方有皇后

{

return false;

}

}

// 检查左对角线上是否有皇后

for (m = i - 1, k = j - 1; (m >= 0) && (k >= 0); m--, k--)

{

if (\*(board + m \* n + k)) // 左对角线有皇后

{

return false;

}

}

// 检查右对角线上是否有皇后

for (m = i - 1, k = j + 1; (m >= 0) && (k < n); m--, k++)

{

if (\*(board + m \* n + k)) // 右对角线有皇后

{

return false;

}

}

return true;

}

4）测试

（a） 测试数据：

表2-3 编程题3测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| 用例1 | 1 | 1 |
| 用例2 | 2 | 无解 |
| 用例3 | 4 | 2 |
| 用例4 | 8 | 92 |
| 用例5 | 10 | 724 |

（b） 对应测试测试用例的运行结果如图2-13所示。

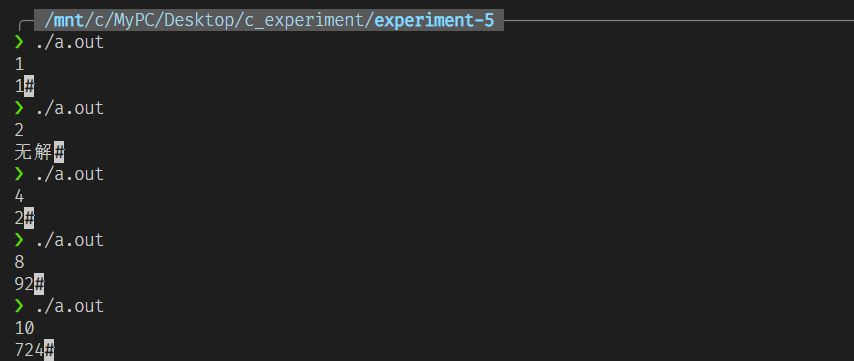


图2-13 程序设计题3测试用例运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（4）实现一个铁路购票系统的简单作为分配算法，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，每次最多够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）。

1) 算法流程图

购票程序主程序部分如图2-14所示。

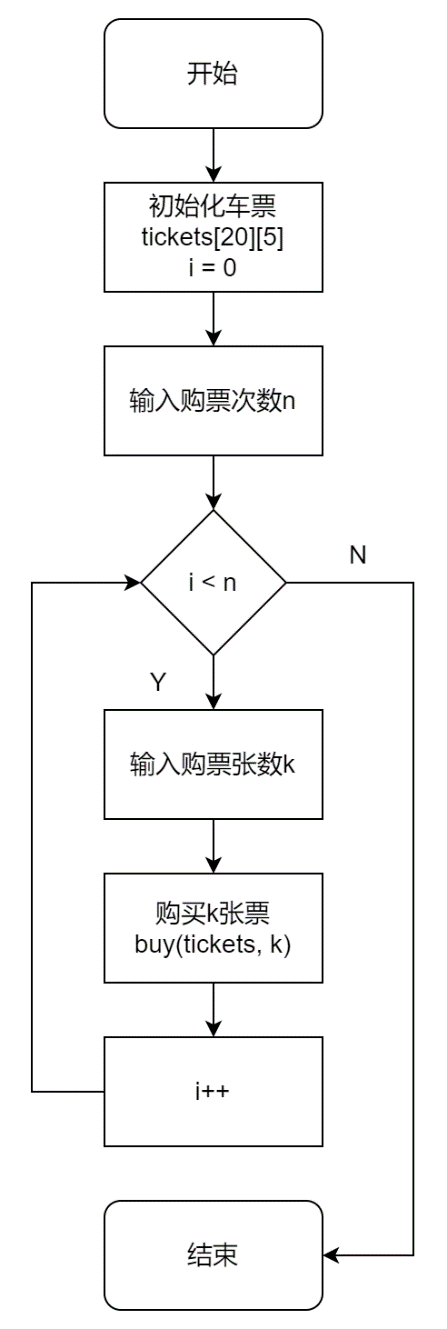


图2-14 程序设计题4主程序流程图

购票部分如图2-15所示。

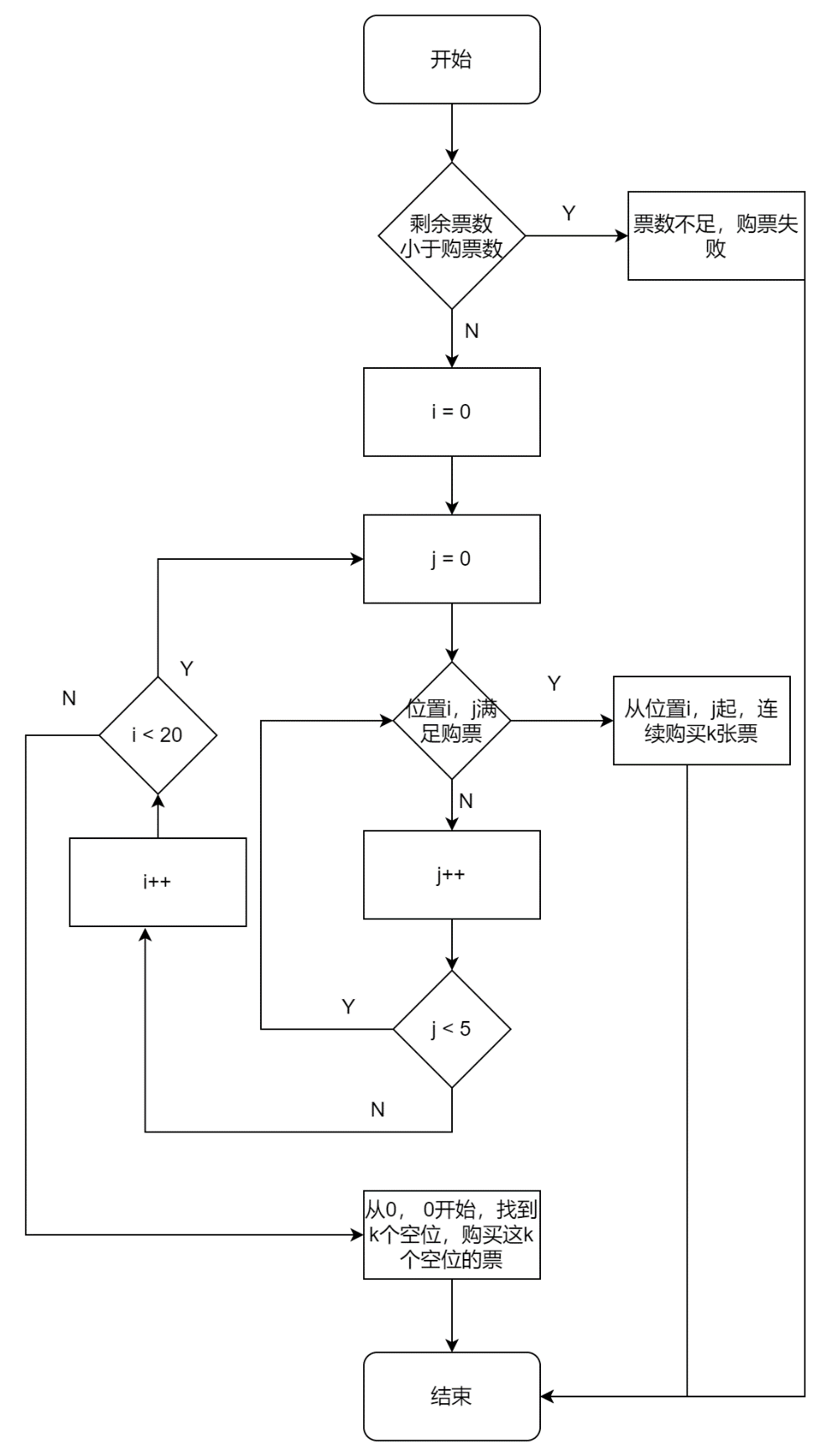


图2-15 程序设计题4购票部分流程图

2）程序清单

#include <stdio.h>

int buy(int tickets[20][5], int k); // 买k张票

int main()

{

int tickets[20][5] = {0}; // 车票是否售出

int n, k;

scanf("%d", &n);

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

{

scanf("%d", &k);

buy(tickets, k);

}

return 0;

}

int buy(int tickets[20][5], int k)

{

int i, j, l;

int res = 0; // 剩余票数

for (i = 0; i < 20; i++)

{

for (j = 0; j < 5; j++)

{

if (tickets[i][j] == 0)

{

res++;

}

}

}

if (res < k) // 剩余票数不足，购票失败

{

return 0;

}

int flag = 0; // 当前位置能否满足k张票

for (i = 0; i < 20; i++)

{

for (j = 0; j < 5; j++)

{

if (tickets[i][j] == 0)

{

flag = 1; // 假设当前位置满足要求

for (l = j; (l - j) < k; l++)

{

if ((l >= 5) || tickets[i][l] == 1)

{

flag = 0; // 已售出，不满足

break;

}

}

if (flag == 1) // 满足要求售出

{

for (l = j; (l - j) < k; l++)

{

tickets[i][l] = 1;

printf("%d%c", i + 1, l == 4 ? 'F' : 'A' + l);

if ((l - j) < k - 1)

{

printf(" ");

}

}

printf("\n");

return 1;

}

}

}

}

if (flag == 0) // 无连续座位

{

l = 0;

for (i = 0; i < 20; i++)

{

for (j = 0; j < 5; j++)

{

if (tickets[i][j] == 0)

{

tickets[i][j] = 1;

l++;

printf("%d%c", i + 1, j == 4 ? 'F' : 'A' + j);

if (l == k)

{

printf("\n");

return 1;

}

else

{

printf(" ");

}

}

}

}

}

return 0;

}

3）测试

（a） 测试数据：

测试输入1：

4

2 5 4 2

预期输出1：

1A 1B

2A 2B 2C 2D 2F

3A 3B 3C 3D

1C 1D

测试输入2：

4

4 5 4 2

预期输出2：

1A 1B 1C 1D

2A 2B 2C 2D 2F

3A 3B 3C 3D

4A 4B

（b） 测试输入1对应测试测试用例的运行结果如图2-16所示。

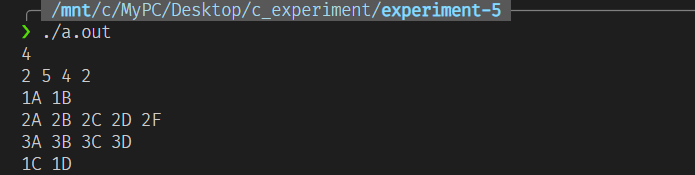


图2-16 程序设计题4测试输入1运行结果

测试输入2对应测试测试用例的运行结果如图2-17所示。

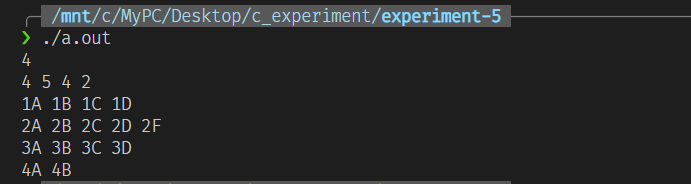


图2-17 程序设计题4测试输入2运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（5）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。

1）解题思路

1.交换数组的两段，可以将整个数组分为5个部分，第一部分在交换区间前，第五部分在交换区间之后，在交换前后，第一部分和第五部分的位置和元素都不会发生改变。

2.在交换时会改变的部分为第二部分、第三部分和第四部分，交换后，第四部分的元素交换到第二部分，第二部分元素交换到第四部分，第三部分可能发生位置改变。

3.分析交换过程可以发现，交换时只需要通过先对交换部位整体逆序，再分别对发生交换的三个部分进行逆序，即可实现交换，如图2-18所示。

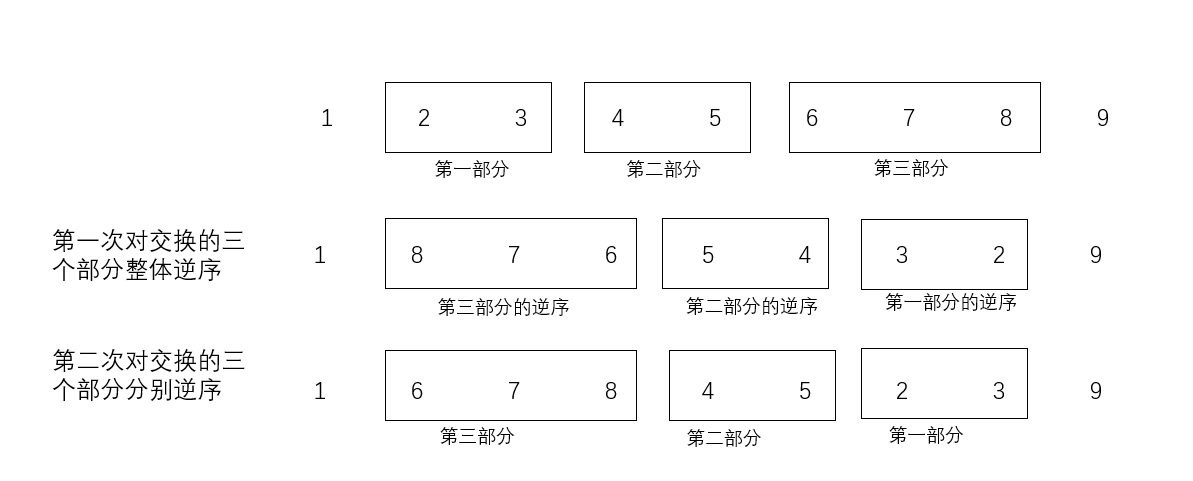


图2-18 程序设计题5交换原理图

2) 算法流程图

交换数组程序如图2-19所示。

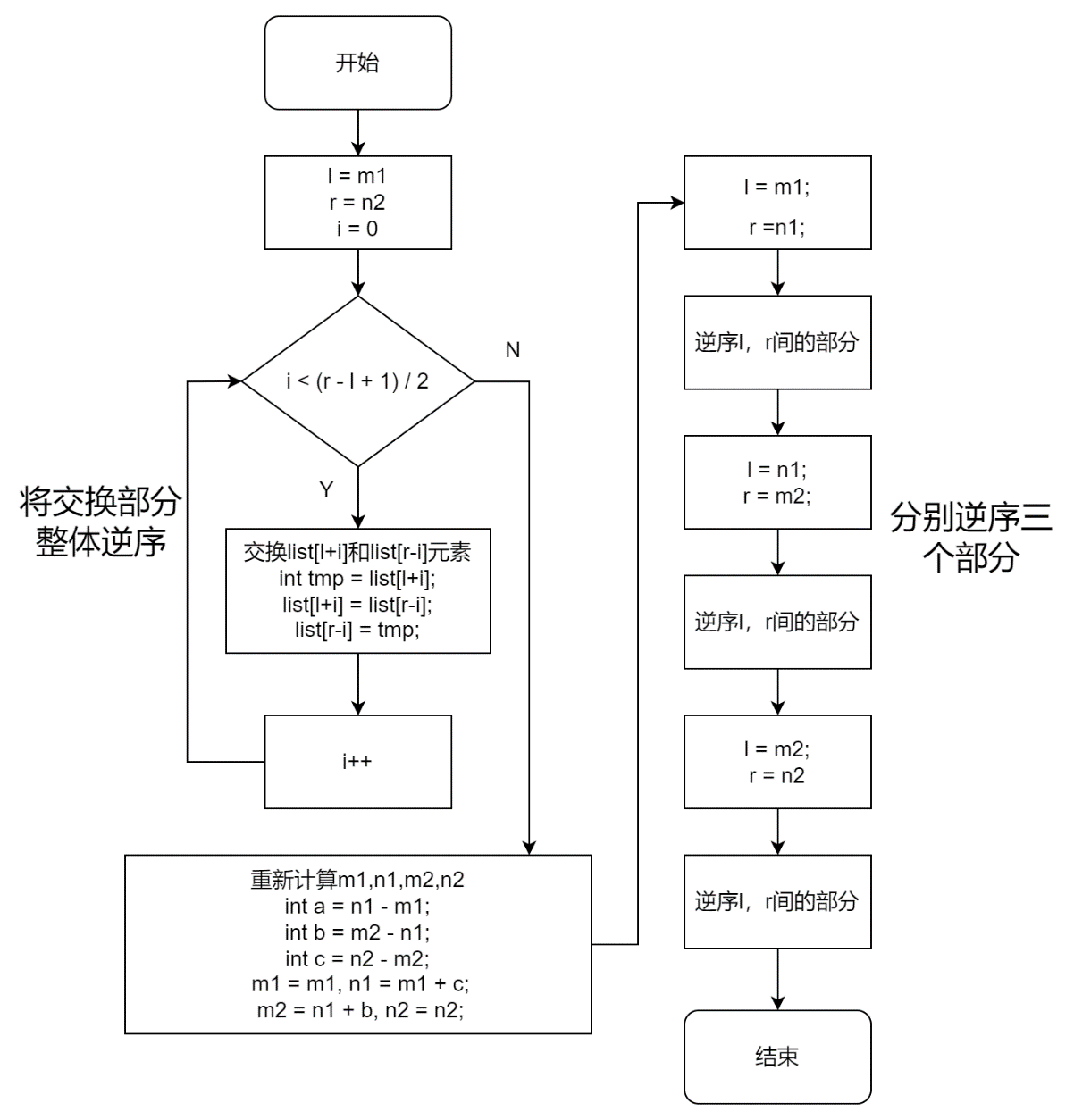


图2-19 程序设计题5交换数组流程图

3）程序清单

#include <stdio.h>

#include <malloc.h>

void swap(int \*list, int m1, int n1, int m2, int n2); // 交换函数

int main()

{

int n;

scanf("%d", &n);

int \*list = malloc(sizeof(int) \* n);

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

{

scanf("%d", list + i);

}

int m1, n1, m2, n2;

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

swap(list, m1, n1, m2, n2);

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

{

printf("%d ", \*(list + i));

}

return 0;

}

void swap(int \*list, int m1, int n1, int m2, int n2) // 将交换部分分为3段，先整体逆序，再分段逆序

{

int i, l, r;

l = m1, r = n2;

for (i = 0; i < (r - l + 1) / 2; i++) // 第一次逆序整个交换的三部分

{

\*(list + l + i) ^= \*(list + r - i); // 使用异或交换两个数 a = a ^ b

\*(list + r - i) ^= \*(list + l + i); // b = a ^ a

\*(list + l + i) ^= \*(list + r - i); // a = a ^ b

}

// 重新计算m1,n1,m2,n2

int a = n1 - m1, b = m2 - n1, c = n2 - m2;

m1 = m1, n1 = m1 + c, m2 = n1 + b, n2 = n2;

l = m1, r = n1;

for (i = 0; i < (r - l + 1) / 2; i++) // 逆序第三部分

{

\*(list + l + i) ^= \*(list + r - i); // 使用异或交换两个数 a = a ^ b

\*(list + r - i) ^= \*(list + l + i); // b = a ^ a

\*(list + l + i) ^= \*(list + r - i); // a = a ^ b

}

l = n1 + 1, r = m2 - 1;

for (i = 0; i < (r - l + 1) / 2; i++) // 逆序第二部分

{

\*(list + l + i) ^= \*(list + r - i); // 使用异或交换两个数 a = a ^ b

\*(list + r - i) ^= \*(list + l + i); // b = a ^ a

\*(list + l + i) ^= \*(list + r - i); // a = a ^ b

}

l = m2, r = n2;

for (i = 0; i < (r - l + 1) / 2; i++) // 逆序第一部分

{

\*(list + l + i) ^= \*(list + r - i); // 使用异或交换两个数 a = a ^ b

\*(list + r - i) ^= \*(list + l + i); // b = a ^ a

\*(list + l + i) ^= \*(list + r - i); // a = a ^ b

}

}

4）测试

（a） 测试数据：

测试输入1：

7

1 2 3 4 5 6 7

1 2 4 6

预期结果1：

1 5 6 7 4 2 3

测试输入2：

9

1 2 3 4 5 6 7 8 9

0 2 5 8

预期结果2：

6 7 8 9 4 5 1 2 3

（b） 测试输入1对应测试测试用例的运行结果如图2-20所示。

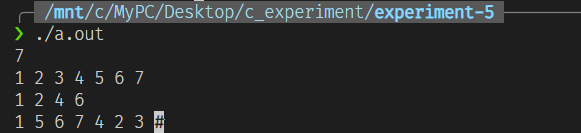


图2-20 程序设计题5测试输入1运行结果

测试输入2对应测试测试用例的运行结果如图2-21所示。

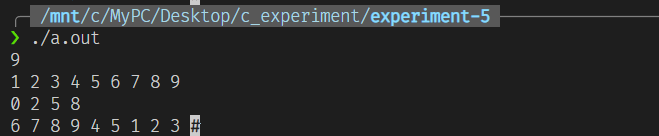


图2-21 程序设计题5测试输入2运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 5.3 实验小结

在本次实验中，对数组、二维数组进行了考察，加深了我对与数组部分的了解，也有了更多的函数调用、传参与函数返回的理解。在N皇后问题中，使用递归函数可以有效地减小编程难度，递归函数需要清楚递归返回的条件以及何时进行递归，否则可能会发生无限递归的情况并导致程序错误。

参考文献

[1] 卢萍,李开,王多强等. C语言程序设计, 北京：清华大学出版社, 2021

[2] 卢萍,李开,王多强等. C语言程序设计典型题解与实验指导, 北京：清华大学出版社, 2019