

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 密码202201**

**学 号： U202116003**

**姓 名： 侯竣**

**指导教师： 吴俊军**

**报告日期： 2022年11月21日**

**网络空间安全学院**

目录

**[5 数组程序设计实验](#_Toc28452_WPSOffice_Level1)** **[1](#_Toc28452_WPSOffice_Level1)**

[5.1 实验目的](#_Toc21468_WPSOffice_Level2) [1](#_Toc21468_WPSOffice_Level2)

[5.2 实验内容及要求](#_Toc28458_WPSOffice_Level2) [1](#_Toc28458_WPSOffice_Level2)

[5.3 实验小结](#_Toc24351_WPSOffice_Level2) [23](#_Toc24351_WPSOffice_Level2)

**[参考文献](#_Toc21468_WPSOffice_Level1)** **[23](#_Toc21468_WPSOffice_Level1)**

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 实验内容及要求

1. 源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. 问题回答：
2. 跟踪进入strcate时，字符数组t和s中的内容分别为"Programming"和"Language"。

当单步执行光条刚落在第二个while语句所在行时，i=12，t[i]='L'。是有问题的, 应该为’\0’, 即字符串结尾

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"Programming"和"Language"，显然未实现字符串连接。

1. 跟踪进入函数strdelc时，字符串s="Programming"，字符c='a'。

单步执行for语句过程中，字符串s, j和k值的变化如图5-1所示。

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"Progrmmingg"，如图5-2, 显然未实现删除功能。

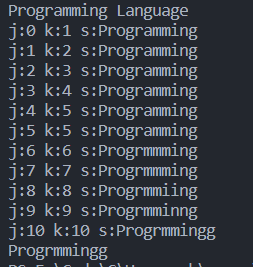


图5-1 字符数组s, j和k值截图



图5-2 实验5-1函数strdelc结束时字符串s截图

1. 错误修改：
2. 第6行，b数组后面空间不够，改为30。
3. 第15行后，添加i--。
4. 第22行后添加s[k]='\0'。

修改后源程序为:

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[30]="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. i--;
17. while((t[i++] = s[j++] )!= '\0');
18. }
19. void strdelc(char s[], char c)
20. {
21. int j,k;
22. for(j=k=0; s[j] != '\0'; j++)
23. if(s[j] != c) s[k++] = s[j];
24. s[k]='\0';
25. }
26. 错误修改后运行结果为:

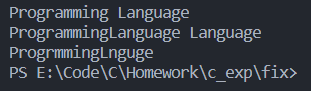


图5-3 实验5-1修改后运行结果截图

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--){
11. /\* 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 \*/
12. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
13. 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 \*/
14. b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/
15. if(j)
16. for(k = --j; k < i-1; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
17. \_\_\_\_\_\_\_\_\_\_\_\_\_\_;
18. }
19. for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/
20. printf(“%6d”, b[i]);
21. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
22. return 0;
23. }

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. 三处填空分别为：
2. a[j-1]
3. a[i-1]
4. a[k]=a[k+1]
5. （1）算法思路如下：
6. 先定义a，b两个一维数组，先循环给a编号
7. 再用for循环进行报号。
8. 利用for循环，每N个报一次，并通过是否等于0去除已经出圈
9. 将每次报号的人存入b数组，并将a数组中，报号者记为0。
10. 依次输出b数组的值, 即报号顺序, 最后输出a数组不为0的值, 即为最后一个出圈的人。
11. 源程序：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

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 \*/

int i, j, k;

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

a[i] = i + 1;

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

{

for (k = 1; k <= N; k++){

if(a[j]==0) k--; //如果a[j]已经出圈，k--, 再往前走一步

if (++j > M - 1) j = 0;

}

b[M - i] = j ? a[j - 1] : a[M - 1]; //b[M-i]存放出圈人的编号

if(j)

a[j-1]=0; //a[j-1]出圈

else

a[M-1]=0; //a[M-1]出圈

}

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

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

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

if(a[i]!=0) printf("%6d",a[i]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

1. 结果：



图5-4 实验5-2用例1的运行结果

1. 程序设计
2. 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. 实验思路如下：
2. 定义一个大小为33的字符数组a初始化为0, 用于存储结果
3. 定义常量mask 0x00000001。
4. 获取输入, 与mask循环相与得到二进制, 加上’0’存入a数组中保存结果, 并将num右移1位
5. 流程图

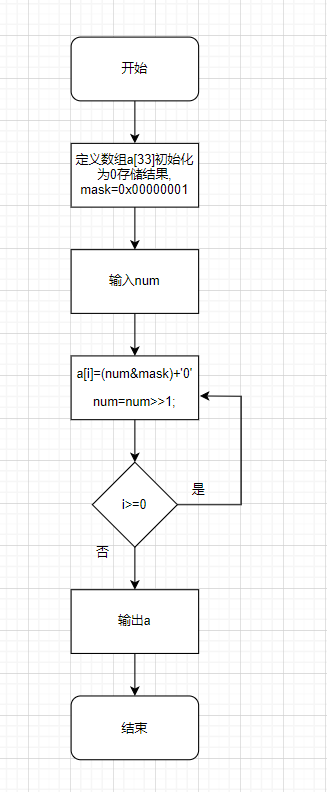


图5-5 算法流程图

1. 程序清单如下：

#include <stdio.h>

#define mask 0x00000001

int main() {

int num;

scanf("%d",&num);

char a[33]={0};

for(int i=31;i>=0;i--){

a[i]=(num&mask)+'0';

num=num>>1;

}

printf("%s",a);

return 0;

}

1. 测试。
2. 测试数据如表5-2所示。

表5-2 程序设计题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程 序 输 入 | 理论结果 | 实际结果 |
| x |
| 用例1 | 14 | 00000000000000000000000000001110 | 00000000000000000000000000001110 |
| 用例2 | -45 | 11111111111111111111111111010011 | 11111111111111111111111111010011 |
| 用例3 | 0 | 00000000000000000000000000000000 | 00000000000000000000000000000000 |

1. 运行结果:



图5-6 程序设计题1用例1的运行结果截图



图5-7 程序设计题1用例2的运行结果截图



图5-8 程序设计题1用例3的运行结果截图

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。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. 实验思路如下：
2. 定义变量i，用于接收输入的菜单操作，定义变量n，用于表示需存储的数量，定义字符数组和整型数组用来存储姓名和成绩。
3. 定义InputScore函数，参数为n(输入人数)，成绩数组和名字数组, 循环读入。
4. 定义SortScore函数，参数为n, 成绩数组和名字数组，使用冒泡排序，将整型数组又小到大排序，同步更改名字数组。
5. 定义PrintScore函数，参数为n, 成绩数组和名字数组, 循环输出姓名与成绩。
6. 定义FindScore函数，二分查找输入的成绩，找到则输出成绩与对应的姓名，否则输出“not found!”。
7. 主函数读入各输入, 用switch分发到各函数进行操作
8. 程序清单如下：

#include <stdio.h>

#include "string.h"

// 成绩输入

void InputScore(int n,char b[][100],int a[])

{

int i;

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

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

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

}

// 成绩排序

void SortScore(int n,char b[][100],int a[]){

int i,t;

char c[100];

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

{

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

{

if (a[j]>=a[j + 1])

{

t = a[j], a[j] = a[j + 1], a[j + 1] = t;

strcpy(c, b[j]);

strcpy(b[j], b[j + 1]);

strcpy(b[j + 1], c);

}

}

}

printf("Reorder finished!\n");

}

// 成绩输出

void PrintScore(int n,char b[][100],int a[]){

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

printf("%s %d\n",b[i],a[i]);

}

}

// 寻找成绩

void FindScore(int x,int n,char b[][100],int a[])

{

int l=0, r=n-1, mid;

while(l<=r){

mid=(l+r)/2;

if(x<a[mid]) r=mid-1;

else if(x>a[mid]) {

l=mid+1;

}

else{

printf("%s %d\n", b[mid],a[mid]);

break;

}

}

if(x != a[mid]) printf("not found!\n");

}

int main(){

int i,n,k;

while ((scanf("%d",&i))!=0)

{

static int t;

if(i==1){

scanf("%d",&n);

}

t=n;

int a[20];

char b[20][100];

switch (i)

{

case 0:

return 0;

case 1:

InputScore(t,b,a);

break;

case 2:

SortScore(t,b,a);

break;

case 3:

PrintScore(t,b,a);

break;

case 4:

scanf("%d",&k);

FindScore(k,t,b,a);

break;

}

}

return 0;

}

1. 测试。
   1. 测试数据如表5-3所示。

表5-3 程序设计题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 输入0. | 结束程序。 |
| 用例2 | 1  6  Jack 95  Mike 90  Joe 75  Andy 95  Rose 89  Sophia 77  2  3  0 | 6 records were input!  Reorder finished!  Jack 95  Andy 95  Mike 90  Rose 89  Sophia 77  Joe 75 |

* 1. 运行结果:



图5-9 程序设计题2用例1的运行结果截图

对应测试用例2的运行结果如下图所示。

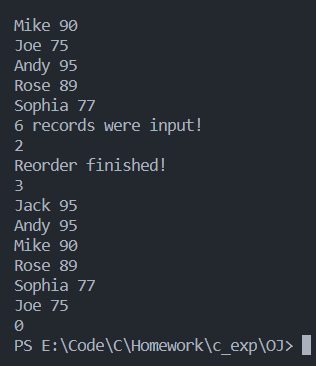


图5-10 程序设计题2用例2的运行结果截图

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. 实验思路如下：使用回溯法

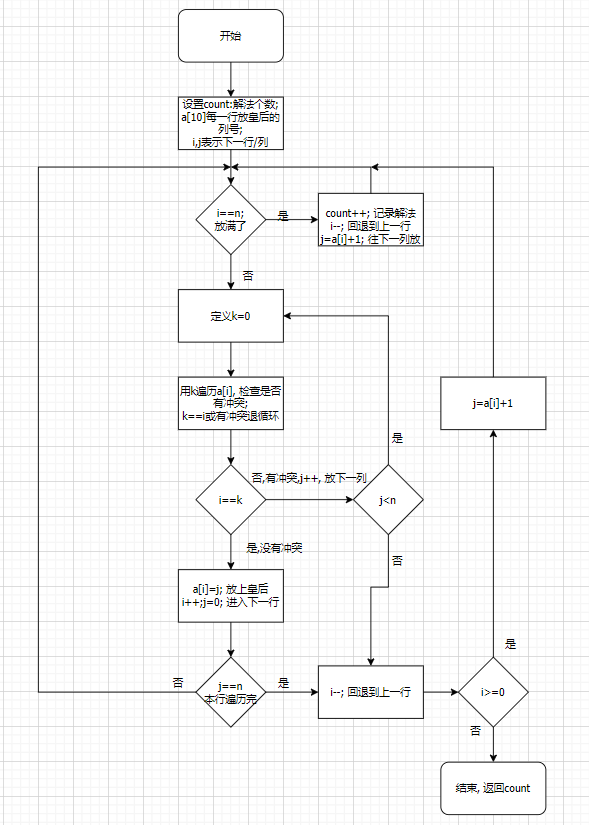


图5-11 算法设计思路

1. 源程序如下：

#include <stdio.h>

#include <math.h>

int NQueen(int n);

int main(void)

{

int n;

scanf("%d", &n);

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

printf("无解");

return 0;

}

int count = NQueen(n);

if (count == 0)

printf("无解");

else

printf("%d", count);

return 0;

}

int NQueen(int n){

int count = 0;

int a[10] = {0};

int i = 0, j = 0;

while (i >= 0){

if (i == n){

count++;

i--;

j = a[i] + 1;

continue;

}

for (; j < n; j++){

int k = 0;

for (; k < i; k++){

if (a[k] == j || abs(a[k] - j) == abs(k - i))

break;

}

if (k == i){

a[i] = j;

i++;

j = 0;

break;

}

}

if (j == n){

i--;

if (i >= 0)

j = a[i] + 1;

}

}

return count;

}

1. 测试。
   1. 测试数据如表5-4所示。

表5-4 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 |
| size |
| 用例1 | 1 | 1 | 1 |
| 用例2 | 2 | 无解！ | 无解！ |
| 用例3 | 4 | 2 | 2 |
| 用例4 | 5 | 10 | 10 |
| 用例5 | 7 | 40 | 40 |
| 用例6 | 10 | 724 | 724 |

* 1. 对应测试结果:

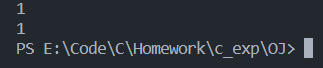


图5-12 程序设计题3用例1的运行结果截图

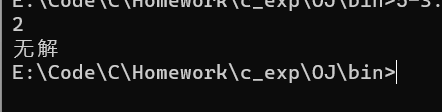


图5-13 程序设计题3用例2的运行结果截图

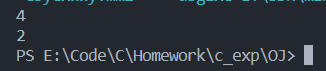


图5-14 程序设计题3用例3的运行结果截图

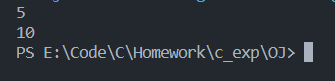


图5-15 程序设计题3用例4的运行结果截图

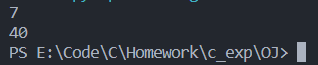


图5-16 程序设计题3用例5的运行结果截图

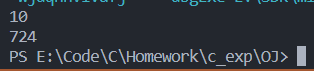


图5-17 程序设计题3用例6的运行结果截图

1. 本关任务：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）

**解答：**

1. 解答思路如下:

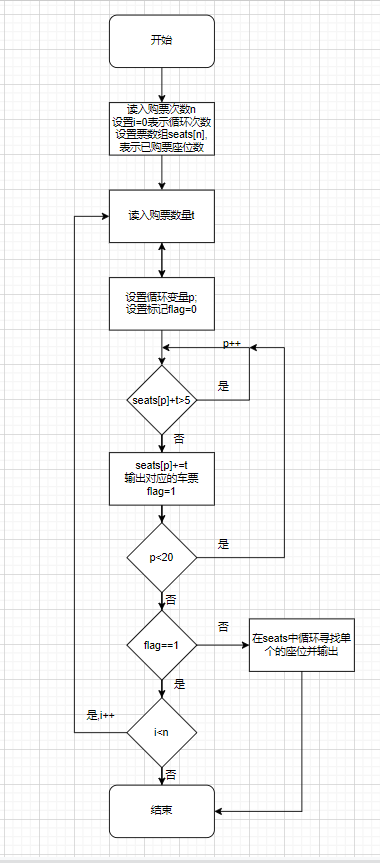


图5-18 算法设计思路

1. 源程序如下：

#include<stdio.h>

int main(){

int seats[20] = {0};

char SeatName[] = {'A','B','C','D','F'};

int n;

scanf("%d", &n);

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

{

int t = 0;

scanf("%d", &t);

int flag = 0; //是否找到了相邻座位

for(int p=0;p<20;p++){

if(seats[p]+t<=5){

seats[p]+=t;

flag = 1;

for(int j=0;j<t;j++){

printf("%d%c", p+1, SeatName[j+seats[p]-t]);

if(j!=t-1)

printf(" ");

}

printf("\n");

break;

}

}

if(flag==0){

for(int p=0;p<20;p++){

if(seats[p]<5){

while(seats[p]<5){

seats[p]++;

printf("%d%c ", p+1, SeatName[seats[p]-1]);

t--;

if(t==0)

return 0;

}

}

}

}

}

return 0;

}

1. 测试

表5-4 程序设计题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 4  5 4 4 2 | 1A 1B 1C 1D 1F  2A 2B 2C 2D  3A 3B 3C 3D  4A 4B |
| 用例2 | 21  4 5 3 4 5 5 5 5 5 5 5 4 4 4 4 4 4 4 3 3 3 | 1A 1B 1C 1D  2A 2B 2C 2D 2F  3A 3B 3C  4A 4B 4C 4D  5A 5B 5C 5D 5F  6A 6B 6C 6D 6F  7A 7B 7C 7D 7F  8A 8B 8C 8D 8F  9A 9B 9C 9D 9F  10A 10B 10C 10D 10F  11A 11B 11C 11D 11F  12A 12B 12C 12D  13A 13B 13C 13D  14A 14B 14C 14D  15A 15B 15C 15D  16A 16B 16C 16D  17A 17B 17C 17D  18A 18B 18C 18D  19A 19B 19C  20A 20B 20C  1F 3D 3F |

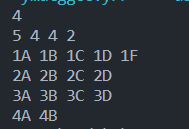


图5-19 测试用例1

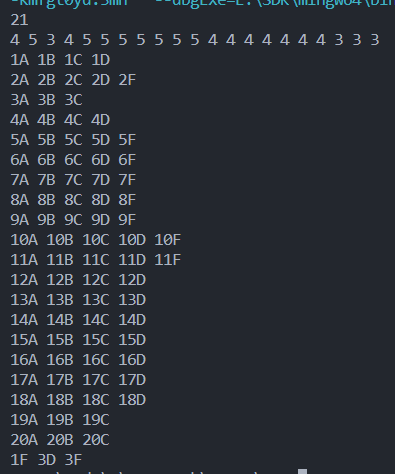


图5-20 测试用例2

1. 本关任务：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。

**解答：**

1. 实验思路如下：
2. 判断n1-m1与n2-m2的大小, 选择比较短的一段移动到比较长的一段
3. 再移动较长段剩下未移动的部分
4. 源程序如下:

#include <stdio.h>

int main(){

int a[100],n,m1,n1,m2,n2;

scanf("%d",&n);

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

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

}

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

// 交换m1~n1和m2~n2的数据

int temp;

// 交换较短的一段

if(n1-m1<n2-m2){

for(int i=0;i<n1-m1+1;i++){

temp=a[m1+i];

a[m1+i]=a[m2+i];

a[m2+i]=temp;

}

}else{

for(int i=0;i<n2-m2+1;i++){

temp=a[m1+i];

a[m1+i]=a[m2+i];

a[m2+i]=temp;

}

}

// 移动剩下的数据

if(n1-m1<n2-m2){

for(int i=n1-m1+1;i<n2-m2+1;i++){

temp=a[m2+i];

for(int j=m2+i;j>m1+i;j--){

a[j]=a[j-1];

}

a[m1+i]=temp;

}

}else{

for(int i=n2-m2+1;i<n1-m1+1;i++){

temp=a[m1+i];

for(int j=m1+i;j<m2+i;j++){

a[j]=a[j+1];

}

a[m2+i-1]=temp;

}

}

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

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

}

return 0;

}

1. 测试

表5-4 程序设计题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例1 | 7  1 2 3 4 5 6 7  1 2 4 6 | 1 5 6 7 4 2 3 |
| 用例2 | 1 2 3 4 5 6 7  1 1 4 5 | 1 5 6 3 4 2 7 |

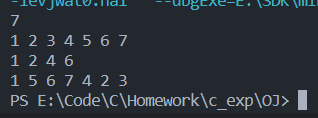


图5-21 测试用例1

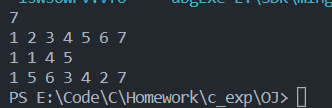


图5-22 测试用例2

## 5.3 实验小结

1. 首先对字符串处理函数的原理以及数组有较深体会, 要善于利用debug检查数组越界问题。

2. 第三题N皇后问题, 一般都会使用回溯法+递归解决N皇后问题, 这次尝试了使用非递归模式, 而是用循环解决问题, 富有挑战性

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