# 实验5 数组实验

## 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 源程序改错

下面是用来将数组a中元素按升序排序后输出的源程序。分析源程序中存在的问题，并对源程序进行修改，使之能够正确完成任务。

1. #include<stdio.h>
2. int main(void)
3. {
4. int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};
5. void sort(int [],int);
6. int i;
7. sort(a[0],10);
8. for(i = 0; i < 10; i++)
9. printf("%6d",a[i]);
10. printf("\n");
11. return 0;
12. }
13. void sort(int b[], int n)
14. {
15. int i, j, t;
16. for (i = 0; i < n - 1; i++)
17. for ( j = 0; j < n - i - 1; j++)
18. if(b[j] < b[j+1])
19. t = b[j], b[j] = b[j+1], b[j+1] = t;
20. }

**解答：**

1. 错误修改：
2. 第7行函数调用时数组错误，正确形式为：

sort(a,10);

1. 第18行程序逻辑错误，正确形式为：

if(b[j] > b[j+1])

1. sort函数未声明，正确形式为：

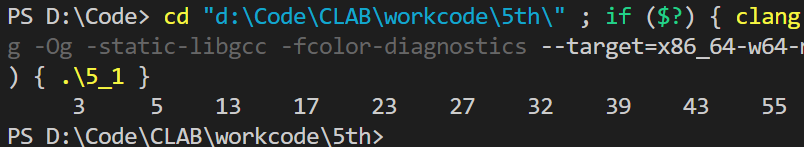
void sort(int b[], int n);

1. 错误修改后运行结果：

预测：将数组a中整数按升序排序后输出，为

3 5 13 17 23 27 32 39 43 55

结果如图所示：



**图5-1 源程序改错题1的测试运行结果**

### 5.2.2 源程序修改替换

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。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--) {
12. /\* i表示圈中人个数，初始为M个，剩1个人时结束循环；
13. j表示当前报数人的位置 \*/
14. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
15. if(++j > i - 1) j = 0; /\* 最后一个人报数后第一个接着报，形成一个圈 \*/
16. b[M-i] = j? : ; /\* 将报数为N的人的编号存入数组b \*/
17. if(j)
18. for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
19. ;
20. }
21. for(i = 0;i < M – 1; i++) /\* 按次序输出出圈人的编号 \*/
22. printf(“%6d”, b[i]);
23. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
24. return 0;
25. }

**解答：**

完善程序

#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--){

/\* 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 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

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 \*/

b[M-i] = j? a[j-1] : a[i-1] ; /\* 将报数为N的人的编号存入数组b \*/

if(j)

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

a[k]=a[k+1] ;

}

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

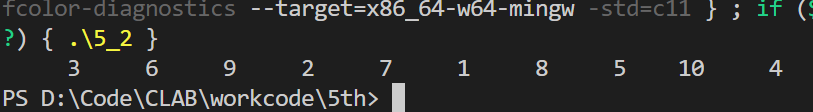
printf(“%6d”, b[i]);

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

return 0;

}

测试运行结果：应该将圈中人编号按出圈顺序输出，推测为：3 6 9 2 7 1 8 5 10 4



**图5-2 源程序修改替换题完善后的测试运行结果**

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。因此，请采用做标记的办法修改（1）中的程序，并使修改后的程序与（1）中的程序具有相同的功能；

**解答：**

修改替换后的程序如下所示：

#include <stdio.h>

#define M 10

#define N 3

#define OUT -1

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 count = M;

int luckydog;

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

a[i] = i + 1;

int m = 0, j = 0;//m与下文k皆用于标识a数组下标

while(count){

int k = 1;

while(k != N || a[m%M]==OUT)//模运算可达到使下标达到最大后自动从头开始

if(a[m++%M]!=OUT)

k++;

b[j++] = a[m % M];

a[m % M] = OUT;

if(count == 1)

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

if(b[i] != OUT)

luckydog = b[i];

count--;

}

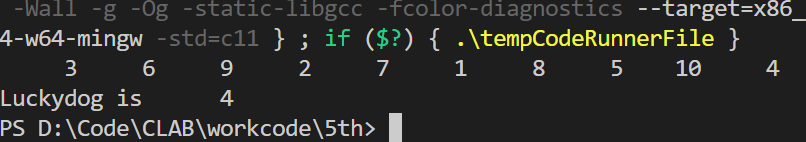
for (int i = 0; i < M; i++) /\* 按次序输出出圈人的编号 \*/

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

printf("\nLuckydog is %6d\n", luckydog); /\* 输出圈中最后一个人的编号 \*/

return 0;

}



**图5-3 源程序修改替换题修改替换后的测试运行结果**

### 5.2.3 跟踪调试题

在下面所给的源程序中，函数strncat(s,t,n)本来应该将字符数组t的前n个字符连接到字符数组s中字符串的尾部。但函数strncat在定义时代码有误，不能实现上述功能。请按下面的要求进行操作，并回答问题和排除错误。

1. 单步执行源程序。进入函数strncat后观察表达式s、t和i。当光条落在for语句所在行时，i为何值？当光条落在strncat函数块结束标记（右花括号 }）所在行时, s、t分别为何值？
2. 分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。
3. #include<stdio.h>
4. void strncat(char [],char [],int);
5. int main(void)
6. {
7. Char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";
8. strncat(a, b, 4);
9. printf("%s\n",a);
10. return 0;
11. }
12. void strncat(char s[],char t[], int n)
13. {
14. int i = 0, j;
15. while(s[i++]) ;
16. for(j = 0; j < n && t[j];)
17. s[i++] = t[j++];
18. s[i] = '\0';
19. }

**解答：**

1. 单步执行源程序。进入函数strncat后观察表达式s、t和i。当光条落在for语句所在行时，i为何值？

答：i的值为 23

当光条落在strncat函数块结束标记（右花括号 }）所在行时, s、t分别为何值？

答：s地址和t地址

1. 分析函数出错的原因，排除错误，使函数正确实现功能，最后写出程序的输出结果。

答：当while(s[i++]) ;执行完后，i=23不为22，s的末尾为字符串结束标记，若i为23，在字符串结束标记后没有意义。

修改后的程序为：

#include<stdio.h>

void strncat(char [],char [],int);

int main(void)

{

char a[50]="The adopted symbol is ",b[27]="abcdefghijklmnopqrstuvwxyz";

strncat(a, b, 4);

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

return 0;

}

void strncat(char s[],char t[], int n)

{

int i = 0, j;

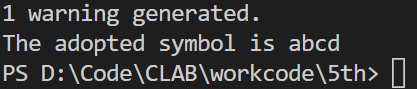
while(s[i]) i++;

for(j = 0; j < n && t[j];)

s[i++] = t[j++];

s[i] = '\0';

}



**图5-4 调试题的测试的运行结果**

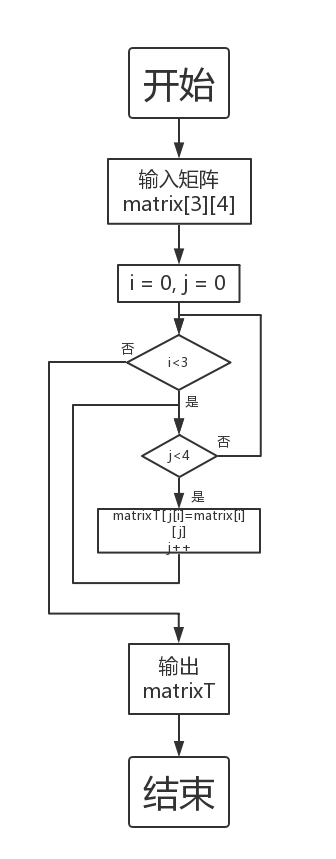
### 5.2.4 程序设计题

1. 编写一个程序,从键盘读取数据，对一个3x4矩阵进行赋值，求其转置矩阵，然后输出原矩阵和转置矩阵。

**解答：**

1. 解题思路：

算法流程如图所示。



**图5-5 编程题1的程序流程图**

1. 源程序清单

#include <stdio.h>

int main(void){

int matrix[3][4] = {};

int matrixT[4][3] = {};

for (int i = 0; i < 3; i++){

for (int j = 0; j < 4; j++){

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

}

}

for (int i = 0; i < 3; i++)

for (int j = 0; j < 4; j++)

matrixT[j][i] = matrix[i][j];

for (int i = 0; i < 3; i++){

for (int j = 0; j < 4; j++)

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

printf("\n");

}

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

for (int i = 0; i < 4; i++){

for (int j = 0; j < 3; j++)

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

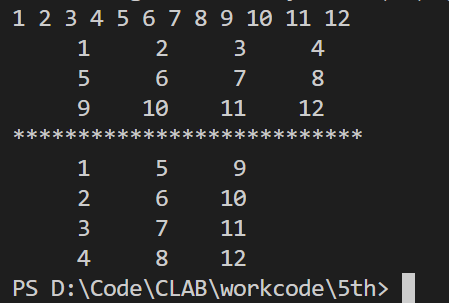
printf("\n");

}

return 0;

}

1. 测试结果



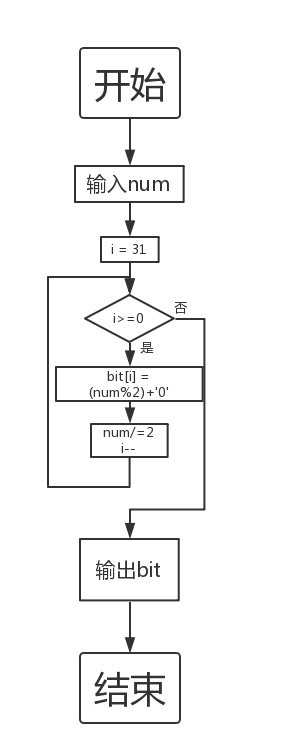
**图5-6 编程题1的运行结果**

1. 编写一个程序, 其功能要求是：输入一个整数，将它在内存中二进制表示的每一位转换成为对应的数字字符，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. 解题思路：

程序流程图：



**图5-7 编程题2的程序流程图**

1. 程序清单

#include <stdio.h>

int main(void){

int num;

char bit[33] = {};

bit[32] = '\0';

scanf("%d", &num);

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

{

bit[i] = (num % 2) + '0';

num /= 2;

}

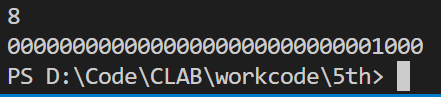
printf("%s\n", bit);

return 0;

}

1. 测试

预测结果：输入8 得到00000000 00000000 00000000 00001000



**图5-8 编程题2的测试运行结果**

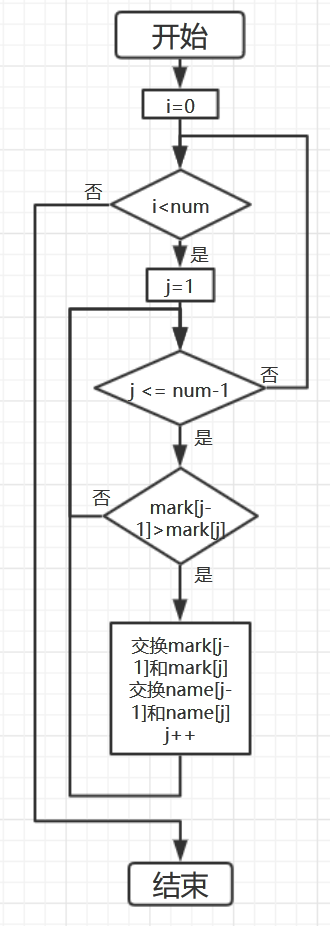
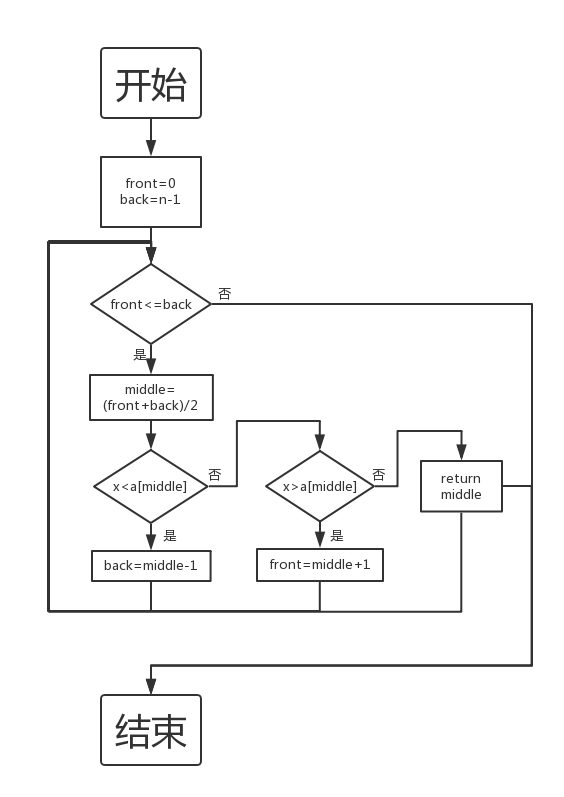
说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 编写一个程序, 其功能要求是：输入n个学生的姓名和C语言课程的成绩，将成绩按从高到低的次序排序，姓名同时作相应调整，输出排序后学生的姓名和C语言课程的成绩。然后，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，输出该成绩同学的姓名和C语言课程的成绩；否则输出提示“not found!”。

**解答：**

1. 解题思路：

分别利用一个字符数组和一个整形数组存储学生姓名和成绩，利用冒泡排序排列成绩的同时对姓名进行排列，最后同时输出即可得到排序后成绩表。利用有序表，进行二分查找。

 ****

**A.冒泡排序流程图 B.二分法查找流程图**

**图5-9 编程题3的程序流程图**

1. 程序清单

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAXSTU 100

void sort(char name[][20], int mark[], int num);

void swap(int \*a, int \*b);

void swaps(char \*a, char \*b);

int BinarySearch(int a[], int x, int n);

int main(void){

char name[MAXSTU][20];

int mark[MAXSTU];

int num;

char option = 0;

int n;

int searchmark;

int index;

printf("Please input name and grade like this\nxiaoming 59\n");

for (num = 0; scanf("%s %d", name[num], &mark[num]) != EOF; num++)

;

sort(name, mark, num-1);

do{

option = 0;

fflush(stdin);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\nwhat do you want?\n");

printf("1.print list\n2.search\nq to quit\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

option = getchar();

if(option == '1')

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

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

else if(option == '2'){

printf("input a mark\n");

scanf("%d", &searchmark);

index = BinarySearch(mark, searchmark, n);

if(index != -1)

printf("find %d is %s grade\n", mark[index], name[index]);

else

printf("not found\n");

}

} while (option != 'q');

return 0;

}

void sort(char name[][20], int mark[], int num){

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

for (int j = 1; j <= num - i; j++)

if(mark[j - 1] > mark[j]){

swap(&mark[j - 1], &mark[j]);

swaps(name[j - 1], name[j]);

}

}

void swap(int \*a, int \*b){

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void swaps(char \*a, char \*b){

char \*temp = malloc(sizeof(char\*));

strcpy(temp, a);

strcpy(a, b);

strcpy(b, temp);

}

int BinarySearch(int a[], int x, int n){

int front = 0, back = n - 1, middle;

while(front<=back){

middle = (front + back) / 2;

if(x < a[middle])

back = middle - 1;

else if(x>a[middle])

front = middle + 1;

else

return middle;

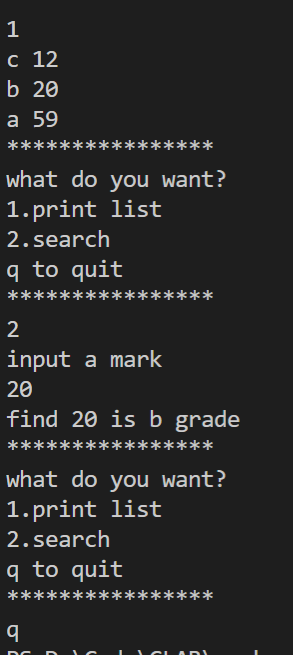
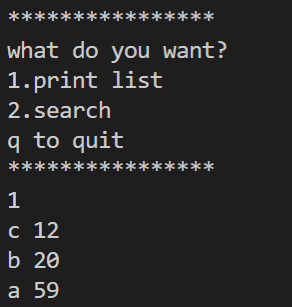
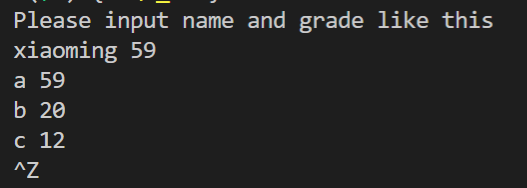
}

return -1;

}

1. 测试结果

程序运行结果如图所示。



**图5-10 编程题3的测试运行结果**

### 5.2.5 选做题程序设计

编写并上机调试运行能实现以下功能的函数和程序。

1. 编写函数strnins(s,t,n),其功能是：可将字符数组 t中的字符串插入到字符数组 s中字符串的第n个字符的后面。
2. 解题思路：首先定位至字符数组s中第n个字符，将其后的所有字符均向后移动字符数组t长度位，再将t中字符插入空位中，此算法简单易行但时间复杂度高。
3. 程序代码：

#include<stdio.h>

#include<string.h>

void strnins(char s[], char t[], int n);

int main(void){

char s[1000] = {}, t[1000] = {};

int n;

scanf("%s %s %d", s, t, &n);

strnins(s, t, n);

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

return 0;

}

void strnins(char s[], char t[], int n){

char temp;

for (int i = strlen(t); i; i--)

for (int i = strlen(s); i > n-1 ; i--)

s[i + 1] = s[i];

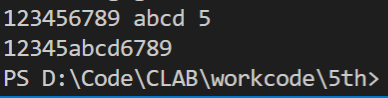
for (int i = 0; t[i]; i++){

s[n + i] = t[i];

}

}

1. 运行结果截图：



**图5-11 选做题1的测试运行结果**

1. 编写一个实现八皇后问题的程序，即：在8\*8方格国际象棋盘上放置8个皇后，任意两个皇后不能位于同一行、同一列或同一斜线（正斜线或反斜线）上，并输出所有可能的放法。
2. 解题思路：此题使用递归实现回溯法，由于未系统学习算法知识，并未独立完成。利用一个一维数组表示棋盘，每一个数组元素中数字表示该行皇后的列编号，问题转化为全排列生成问题。利用数学知识可使用行列编号来完成皇后间是否可以互相攻击的判断标准：x0 == x1 || x0-y0 == x1-y1即为攻击到。
3. 程序清单：

#include<stdio.h>

#define TRUE 1

#define FALSE 0

#define MAX 8

void print(char queen[]);

int judge(int index);

static char queen[MAX] = {};

static int counter = 0;

int main(void){

judge(0);

printf("there are totally %d solutions\n", counter);

return 0;

}

void print(char queen[]){

printf("The %dth solution:\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n", counter);

for (int i = 0; i < 8; i++){

for (int j = 0; j < queen[i]; j++)

printf("# ");

printf("Q ");

for (int j = queen[i] + 1; j < 8; j++)

printf("# ");

printf("\n");

}

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

}

int judge(int index){

if(index == MAX){

counter++;

print(queen);

return TRUE;

}

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

int flag = TRUE;

queen[index] = i;//将index行的后移动到第i列位置，(index, i)

for (int j = 0; j < index; j++){//j是行数

if(queen[j] == queen[index] ||

(index - j) == (queen[index] - queen[j]) ||

(j - index) == (queen[index] - queen[j])){

flag = FALSE;//判断是否有必要递归

break;

}

}

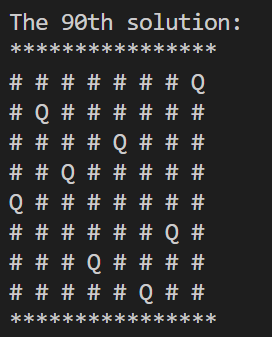
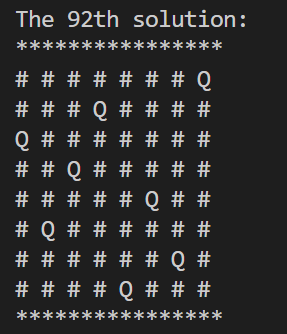
if(flag)

judge(index + 1);

}

}

1. 运行截图：（节选）

**图5-12 选做题2的测试运行结果**

## 5.3 实验小结

第五次C语言上机实验，主要学习并熟练了数组的使用，同时学习了二分查找算法和选择法排序算法的思想。

1. **改错实验**

此题首先加深了我对排序算法的印象，其次，使我学会了如何向函数传递一个数组，在函数形参列表中声明的数组，在函数体内部，实际上通过指针运算完成，因此传值时应该传入指针常量，即数组首地址。同时我意识到，在函数内部用sizeof运算符无法得到数组的长度，会引起难以发现的错误。

1. **程序修改替换实验**

约瑟夫环问题首先让我感受到，如果没有使用清晰的变量名称，没有写上必要的注释，阅读程序代码将会十分困难。在完成语句填空以及改写后，老师告诉我约瑟夫环还可用循环链表进行处理，十分便捷。这让我意识到了需要学习的内容还很多，还需继续努力。

1. **跟踪调试题**

在继续学习使用调试器的同时，也让我理解了为何对于边界要格外注意的问题

1. **程序设计实验**

第一题向我介绍了一个新的数学概念，同时加强了我对二维数组中行和列的理解，也让我认识到，数组这一概念实际上可以用于模拟数学中N维空间坐标系的概念。

第二题重温进制转换的同时，进一步熟悉了对字符数组的使用。

第三题首先使我熟悉了二分查找的原理以及使用细节，也另外编写程序比对了遍历查找与二分查找的速度差异，认识到，大多数情况下，最直接的算法都不会是最优的，需要经过思考才可以提高效率。然后，在对排序时对字符数组的交换上出现了一些困难，在使用函数完成交换字符数组这一步时，由于对函数形参中数组与指针的关系以及一些概念的模糊，导致耗费了大量时间，在重新复习了相关知识，加上老师讲解之后，完成了这一工作。另外，在这题中，设计了一个简单的命令行界面下的UI，虽然简陋，但是也感受到了一个好的UI的作用。这其中为了解决循环输出菜单的问题，学习使用了标准库函数fflush()，了解了一些输入输出流的深入内容。

1. **选做题**

第一个选做题，选择了最易想到的方式。将字符数组s中第n位往后的数组元素全部后移直到空位足够容纳字符数组t，再将字符数组t填入空位中。经过助教提醒，我意识到这种方式效率极低，想到在之前二分查找处得到的启示，对此题要求的函数功能实现的算法进行了改进，用足够长的数组来存储数组s的后半段，再将三个数组按顺序连接，避免了耗时的数组元素移动操作。

第二个选做题，遇到了著名的数学问题——八皇后问题。起初利用二维数组模拟棋盘遍历查找的方式实现，但是首先循环嵌套相当多层，代码看起来混乱、难以理解。且这种方法耗时极高，在写报告时计算得到的结果，这种全部遍历的算法，一共有近45亿种可能的结果，需要从这其中找到所有解相当耗费时间，而且花费了大量时间生成与判断毫无价值的数据。因此，在咨询老师及查询资料后，了解到了什么是递归回溯算法，可以在已知无法继续摆放任何棋子之后，回到上一步操作继续寻找下一个安全位置的组合，可以节省大量的时间。同时使用一维数组来模拟棋盘，本身也排除了大量无用组合，将数据范围缩减至大概八万种，比起之前的范围，可以说是非常优秀的算法了。经过这一题，我继续意识到自己还需要学习很多知识，单方面地认为计算机运算速度很快，可以不加考虑地使用暴力求解的方式是很懒惰，很不负责任的。同时激起了我对学习算法的热情，现在很期待将来的数据结构课程以及算法课程。

**总结：**此次上机实验题很有难度，也十分有趣，想说的话已经在各题目小结中说完，希望我自己可以在以后的学习中保持热情。