

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程2003班**

**学 号： U202010783**

**姓 名： 刘铭宸**

**指导教师： 唐赫**

**报告日期： 2020.12.5**

**软件工程**

**目 录**

[**5 数组实验 1**](#_Toc404837938)

[5.1 实验目的 1](#_Toc404837921)

[5.2 实验内容 1](#_Toc404837922)

[5.3 实验小结 1](#_Toc404837923)7

# 5 数组试验

## 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 源程序改错与跟踪调试**

**1．程序改错**

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

ProgramingLnguage

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。

/\*实验5-1程序改错与跟踪调试题程序\*/

#include<stdio.h>

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

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[]="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++]) ;

while((t[i++] = 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];

}

**解答：**

（1）错误修改：

1) a、b数组定义过小，会导致数组越界，正确形式为：

char a[1000] = "Language", b[1000] = "Programming";

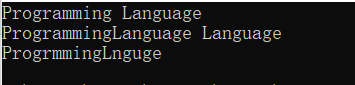
2) 第15行的while (t[i++]);会导致i多加了一遍，正确形式为：

while (t[++i]);

3)strdelc函数中s数组缺少结束符\0，应在最后加上：

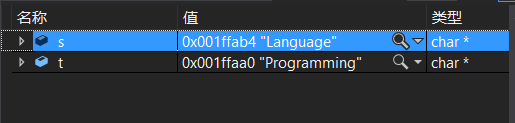
s[k] = '\0';

（2）错误修改后运行结果：

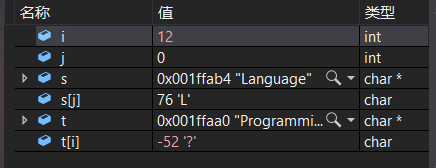


**2．程序跟踪调试**

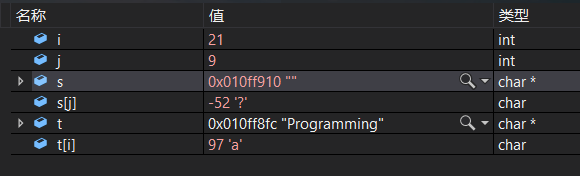
(1) 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。



如图，进入strcate时，数组s中的内容为Language，数组t中的内容为Programming，结果正确。

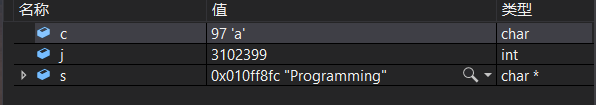


如图，当单步执行光条刚落在第二个while语句所在行时，i的值为12，t[i]为’?’，结果错误，i的值应为11，t[i]应为’\0’。

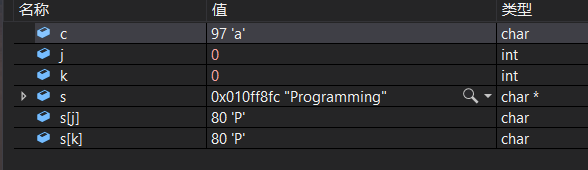
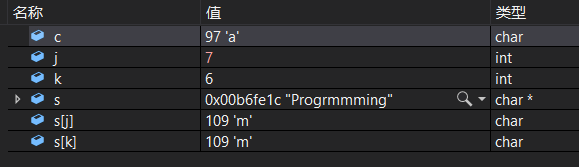


如图，当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t为Programming，s为空，没有实现字符串拼接。

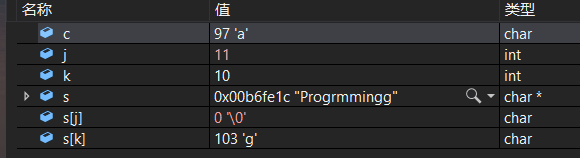
（2）跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。



如图，进入函数strdelc时，字符数组s的内容为Programming，字符c的值为a，结果正确。

如图，在单步执行for语句过程中，当s[j]==c时，k不变，j加一，s中删除了字符c；否则k和j都加一，s不变。结果没有问题。



如图，当单步执行光条落在strdelc函数块结束标记“}”所在行时，s的内容为Progrmmingg，实现了删除字符的功能，但由于没有添加结束符，导致s中最后一个字符重复两次，结果错误。

**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。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a放圈中人的编号，数组b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 ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 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 \*/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

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;

}

**解答：**

当j不为0时，表明编号为a[j-1]的人报数为N，所以将其存入数组b；当j为0时，表明最后一个人刚好报到N，所以此时将最后一个人的编号a[i-1]存入数组b；一轮报数结束后，需要压缩数组a，将报数为N的人的编号从a中删除。

#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 \*/

//printf("%d", j);

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;

}

（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：**

定义数组a存放圈中人的编号，当有人报数为N时，将其编号置为0，在输出时增加编号不为零的判断条件即可。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M]; /\* 数组a存放圈中人的编号 \*/

int i,x=M,k=0;

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

a[i] = i+1;

while(x>1){

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

if (a[i]) k++;

if (k == N) {

printf("%6d", a[i]); /\* 按次序输出出圈人的编号 \*/

a[i] = 0;

k = 0;

x--;

}

}

}

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

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

return 0;

}



**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）通过分析本题的算法流程，可建立如下算法流程图。

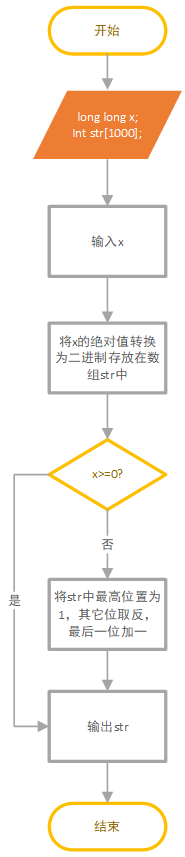


图5-1 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

#include <math.h>

int main(void) {

long long x = 0, k = 0;

int str[1000] = { 0 };

scanf("%lld", &x);

int m = abs(x);

while (m) {

int b = m / 2;

str[k] = m - b \* 2;

k++;

m /= 2;

}

if (x >= 0) {

for (int i = 31; i >= 0; i--)printf("%d", str[i]);

}

else {

int n = 0;

str[31] = 1;

printf("%d", str[31]);

for (int j = 30; j >= 0; j--) {

if (str[j] == 1)str[j] = 0;

else str[j] = 1;

}

if (str[0] == 0)str[0] = 1;

else {

while (str[n]) {

if (str[n] == 1)str[n] = 0;

else str[n] = 1;

n++;

}

str[n] = 1;

}

for(int q=30;q>=0;q--)

printf("%d", str[q]);

}

printf("\n");

return 0;

}

3）测试

（a） 测试数据：

7

-7

0

200

-200

（b） 对应测试数据的运行结果截图











（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。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. 解题思路：

通过分析本题的算法流程，可对需要设计的函数建立如下算法流程图。

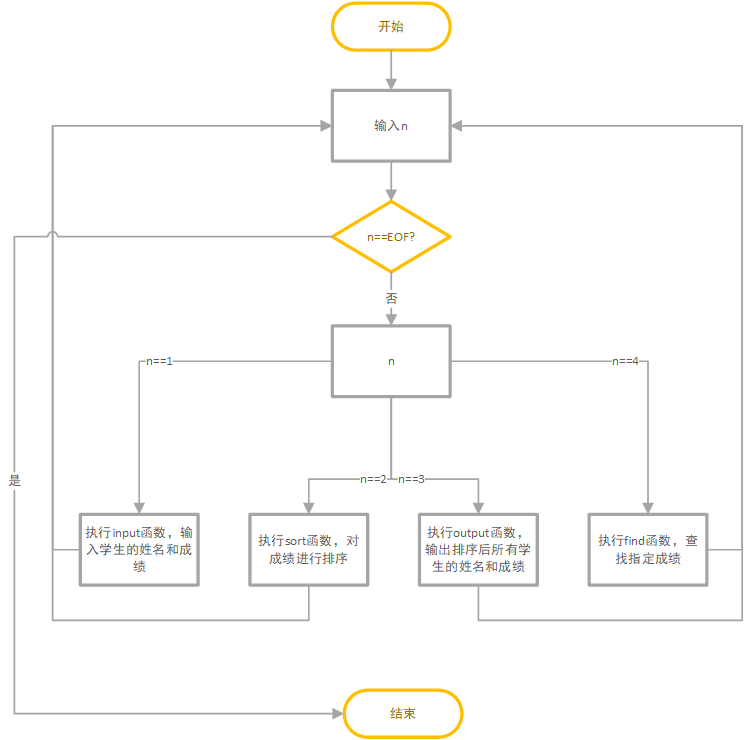


图5-2 编程题2的程序流程图

2）程序清单

#include <stdio.h>

#include <string.h>

void input(char name[][100], int score[100], int\*);

void sort(char name[][100], int score[100], int\*);

void output(char name[][100], int score[100], int\*);

void find(char name[][100], int score[100], int\*);

int main(void) {

int n = 1, score[100] = { 0 }, k = 0;

char name[100][100];

while ((scanf("%d", &n)) != EOF) {

switch (n)

{

case 0:return 0;

case 1:input(name, score, &k); break;

case 2:sort(name, score, &k); break;

case 3:output(name, score, &k); break;

case 4:find(name, score, &k); break;

}

}

//printf("（程序退出）\n");

return 0;

}

void input(char name[][100], int score[100], int\* k) {

scanf("%d", k);

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

scanf("%s", name[i]);

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

}

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

}

void sort(char name[][100], int score[100], int\* k) {

for (int i = 0; i < \*k - 1; i++) {

for (int j = 0; j < \*k - i - 1; j++) {

if (score[j] < score[j + 1]) {

int temp = score[j];

score[j] = score[j + 1];

score[j + 1] = temp;

char temp1[100] = { 0 };

strcpy(temp1, name[j]);

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

strcpy(name[j + 1], temp1);

}

}

}

printf("Reorder finished!\n");

}

void output(char name[][100], int score[100], int\* k) {

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

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

}

void find(char name[][100], int score[100], int\* k) {

//sort(name, score, k);

int low = 0, high = \*k - 1, mid=0,s;

scanf("%d", &s);

while (low <= high) {

mid = (low + high) / 2;

if (score[mid] > s)low = mid + 1;

else if (score[mid] == s) { printf("%s %d\n", name[mid], score[mid]); break; }

else high = mid - 1;

}

if(score[mid]!=s) printf("not found!\n");

}

3）测试

（a） 测试数据：

1

6

Jack 95

Mike 90

Joe 75

Andy 95

Rose 89

Sophia 77

2

3

4

89

0

（b） 对应测试数据的运行结果截图

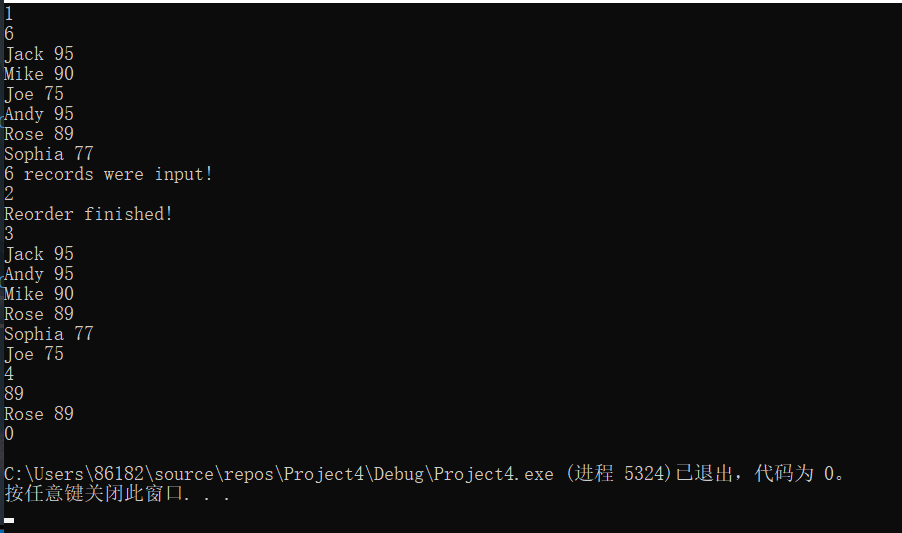


图5-3 编程题2的测试运行结果

（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。”

**解答：**

1. 解题思路：

用回溯法在递归时判断是否满足条件，如果不满足，则不用递归下去，返回上一层进行处理。递归时判断当前的皇后是否和前面的皇后在一条对角线上，如果在一条直线上，就不需要递归下去了，返回上一层；如果不在，就继续递归，下一个继续进行判断，直到满足条件为止。

2）程序清单

#include <stdio.h>

int N, tot, row[100], lex[100], rex[100];

int maplex[100][100], maprex[100][100];

void pre () {

int count = 0;

for(int i = 1; i <= N; i++) {

count ++;

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

maplex[j][N-i+j] = count;

}

for(int i = 2; i <= N; i++) {

count ++;

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

maplex[i+j-1][j] = count;

}

for(int i = 1; i <= N; i++)

for(int j = 1; j <= N; j++)

maprex[i][j] = maplex[i][N-j+1];

}

void dfs (int n) {

if(n == N + 1) {

tot ++;

return ;

}

for(int j = 1; j <= N; j ++) {

if((!row[j]) && (!lex[maplex[n][j]]) && (!rex[maprex[n][j]])) {

row[j] = lex[maplex[n][j]] = rex[maprex[n][j]] = 1;

dfs(n+1);

row[j] = lex[maplex[n][j]] = rex[maprex[n][j]] = 0;

}

}

}

int main () {

scanf("%d", &N);

pre();

dfs(1);

if(tot == 0)

printf("无解");

else

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

return 0;

}

3）测试

（a） 测试数据：

1

2

6

10

（b） 对应测试测试用例1的运行结果如图5-4所示。



图5-4 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图5-5所示。



图5-5 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图5-6所示。



图5-6 编程题3的测试用例三的运行结果

对应测试测试用例4的运行结果如图5-7所示。



图5-7 编程题3的测试用例四的运行结果

## 5.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在源程序完善和修改替换中，我知道了要如何解决约瑟夫问题，并通过改进算法减小了算法的复杂度。在编程题1中我加深巩固了正数和负数的二进制存储方法以及二进制的运算；在编程题3中了解了如何求解N皇后问题，并对回溯算法有了一定的理解，增强了我解决问题的能力，也学会了合理设计函数，用以简化程序的编写。