

**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：**

**软件学院**

**目 录**

[5 数组实验 1](#_Toc184322287)

[5.1 实验目的 1](#_Toc184322288)

[5.2 实验内容及要求 1](#_Toc184322289)

[5.3 实验小结 13](#_Toc184322290)

**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. 调试分析
   1. 刚落在第二个while时，i=12，t[i]=’\0’  
      最后t, s的值与之前相同，没有实现字符串连接。
   2. s结尾的处理有问题，其他没有问题。  
      最后s=”Progrmmingg”，实现了删除操作，但结尾有误。
2. 修改后完整代码

#include<stdio.h>

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

void strdelc(char [], char);

int main(void) {

char a[] = "Language", b[30] = "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++]);

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];

s[k] = '\0';

}

**2、 源程序完善和修改替换**

(1) 下面的源程序用于求解约瑟夫问题：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 ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 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;

}

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. 填空
   1. a[j - 1]
   2. a[i - 1]
   3. a[k] = a[k + 1];
2. 修改后程序

#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，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 \*/

do {

j = (j + 1) % M;

} while (!a[j]);

}

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

a[j] = 0;

do {

j = (j + 1) % M;

} while (!a[j]);

}

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

if (a[i])

printf("%6d\n", a[i]);

return 0;

}

1. 修改后运行效果

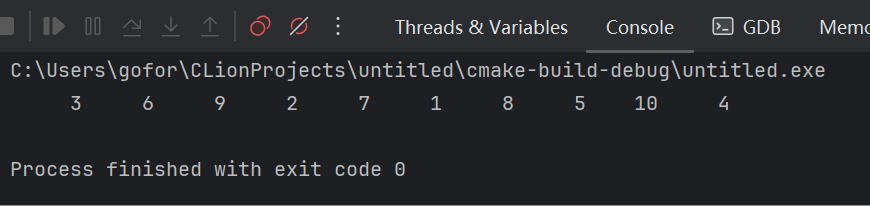
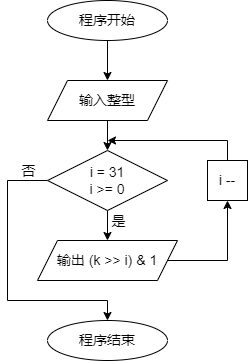


图5-1程序修改题1运行结果图

**3. 程序设计**

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验5，第10关实验5-1、第11关实验5-2，以及第12关实验5-3。

（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

图5-2 编程题1的程序流程图

1. 源程序清单

#include<stdio.h>

int main() {

int k;

scanf("%d", &k);

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

printf("%d", (k >> i) & 1);

}

}

1. 测试数据

表5-1 编程题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 7 | 00000000000000000000000000000111 | 00000000000000000000000000000111 |
| 用例2 | 2148473647 | 01111111111111111111111111111111 | 01111111111111111111111111111111 |

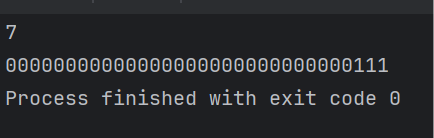
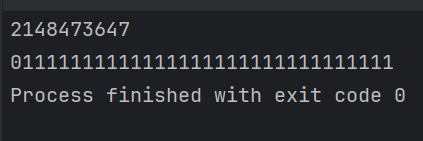
1. 运行效果

图5-3 编程题1的运行结果图

（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。

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：**

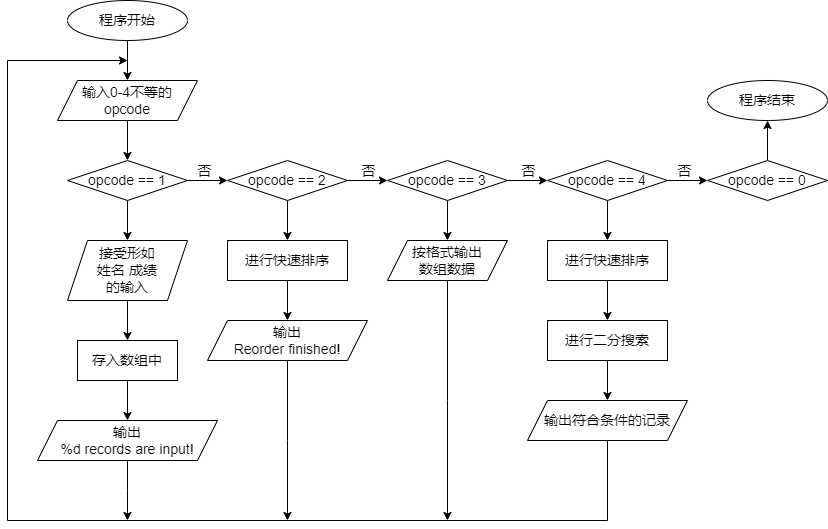
1. 算法流程

图5-4 编程题2的程序流程图

1. 源程序清单

#include<stdio.h>

int count = 0;

int scores[1000];

char names[1000][100];

void addStudent(const char name[], const int score) {

scores[count] = score;

for (int i = 0; i < 100; i++) {

names[count][i] = name[i];

}

count++;

}

void replace(const int src, const int dest) {

scores[dest] = scores[src];

for (int i = 0; i < 100; i++) {

names[dest][i] = names[src][i];

}

}

void sortStudents(const int start, const int end) {

if (start >= end) return;

int i = start, j = end;

const int pivot = scores[i];

char pname[100];

for (int ind = 0; ind < 100; ind++) {

pname[ind] = names[i][ind];

}

while (i < j) {

while (i < j && scores[j] <= pivot) j--;

if (i < j) {

replace(j, i);

i++;

}

while (i < j && scores[i] > pivot) i++;

if (i < j) {

replace(i, j);

j--;

}

}

for (int ind = 0; ind < 100; ind++) {

names[i][ind] = pname[ind];

}

scores[i] = pivot;

sortStudents(start, i - 1);

sortStudents(i + 1, end);

}

void exportStudents() {

// sortStudents(0, count - 1);

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

printf("%s %d\n", names[i], scores[i]);

}

}

void binarySearch(const int score, const int start, const int end) {

// sortStudents(0, count - 1);

const int index = start + (end - start) / 2;

if (start == end) {

if (scores[index] == score)

printf("%s %d\n", names[index], scores[index]);

else printf("not found!\n");

return;

}

if (score < scores[index]) {

binarySearch(score, index + 1, end);

} else if (score > scores[index]) {

binarySearch(score, start, index - 1);

} else {

for (int i = start; i <= end; i++) {

if (scores[i] == score) {

printf("%s %d\n", names[i], scores[i]);

}

}

}

}

int main() {

int opcode;

scanf("%d", &opcode);

while (opcode != 0) {

if (opcode == 1) {

int n;

scanf("%d", &n);

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

char name[100];

int score;

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

addStudent(name, score);

}

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

} else if (opcode == 2) {

sortStudents(0, count - 1);

printf("Reorder finished!\n");

} else if (opcode == 3) {

exportStudents();

} else if (opcode == 4) {

int score;

scanf("%d", &score);

binarySearch(score, 0, count - 1);

}

scanf("%d", &opcode);

}

}

1. 测试数据

表5-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 1  6  Jack 95  Mike 90  Joe 75  Andy 95  Rose 89  Sophia 77  2  3  4  89  4  88  0 | 6 records were input!  Reorder finished!  Jack 95  Andy 95  Mike 90  Rose 89  Sophia 77  Joe 75  Rose 89  not found! | 6 records were input!  Reorder finished!  Jack 95  Andy 95  Mike 90  Rose 89  Sophia 77  Joe 75  Rose 89  not found! |

1. 文本

   描述已自动生成运行效果

图5-5 编程题2的运行结果图

（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：**

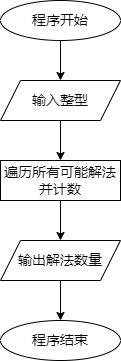
1. 算法流程

图5-6 编程题3的程序流程图

1. 源程序清单

#include<stdio.h>

int board[15][15];

int count = 0;

void nQueens(int row, int n) {

if (row == n) {

count += 1;

return;

}

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

int valid = 1;

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

if (board[j][i] == 1) {

valid = 0;

break;

}

}

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

if (i - j >= 0 && board[row - j][i - j] == 1) {

valid = 0;

break;

}

if (i + j < n && board[row - j][i + j] == 1) {

valid = 0;

break;

}

}

if (valid) {

board[row][i] = 1;

nQueens(row + 1, n);

board[row][i] = 0;

}

}

}

int main() {

int n;

scanf("%d", &n);

nQueens(0, n);

if (count) {

printf("%d", count);

} else {

printf("无解");

}

return 0;

}

1. 测试数据

表5-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 1 | 1 | 1 |
| 用例2 | 8 | 92 | 92 |

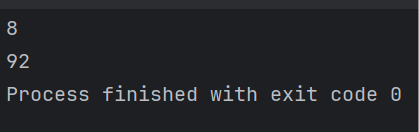
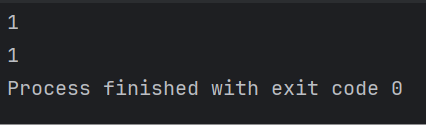
1. 运行效果

图5-7 编程题3的运行结果图

5.3 实验小结

1. 在本次实验中，我学习了数组的基本使用，以及遍历、递归等算法，加深了我对计算机语言的了解。
2. 在本次实验中，我多次使用了集成开发环境自带的断点等功能，这有助于快速的问题定位与解决。