目录

[5 数组实验 3](#_Toc118746864)

[5.1 实验目的 3](#_Toc118746865)

[5.2 实验内容及要求 3](#_Toc118746866)

[5.3 实验小结 18](#_Toc118746867)

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

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。

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。

01 /\*实验5程序改错与跟踪调试题程序\*/

02 #include<stdio.h>

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

04 void strdelc(char [],char );

05 int main(void)

06 {

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

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

09 strcate(b,a); printf("%s %s\n",b,a);

10 strdelc(b, 'a'); printf("%s\n",b);

11 return 0;

12 }

13 void strcate(char t[],char s[])

14 {

15 int i = 0, j = 0;

16 while(t[i++]) ;

17 while((t[i++] = s[j++] )!= '\0');

18 }

19 void strdelc(char s[], char c)

20 {

21 int j,k;

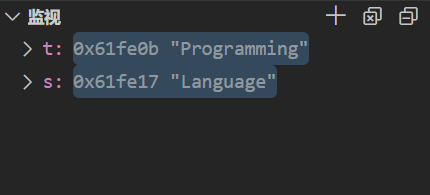
22 for(j=k=0; s[j] != '\0'; j++)

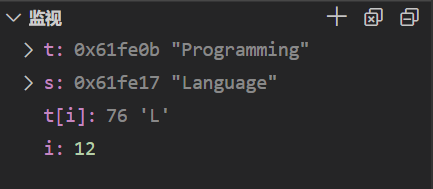
23 if(s[j] != c) s[k++] = s[j];

24 }

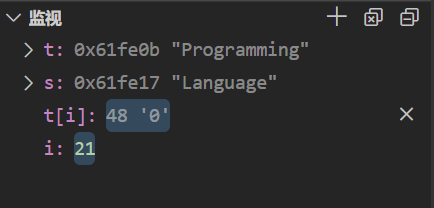
解答：

1. strcate。
   1. 参数传递正确。

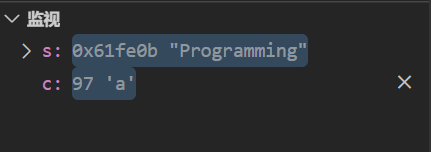
图5-1 源程序改错（1.1）观察变量图

* 1. i=12,t[i]= 'L'。存在问题。'g'在t[10]，而'L'在t[12]，中间有多余的元素'\0'。

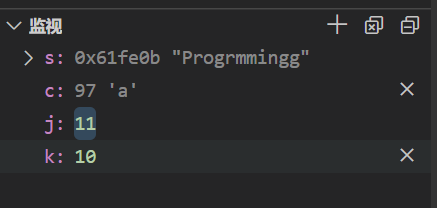
5-2 源程序改错（1.2）观察变量图

* 1. t="Programming",s="Language"。存在问题，没有成功连接。

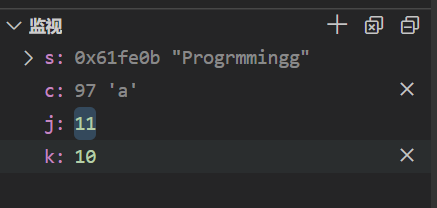
5-3 源程序改错（1.3）观察变量图

1. strdelc。
   1. s="Programming"，c='a'，正确。

5-4 源程序改错（2.1）观察变量图

* 1. 结果正确。

5-5 源程序改错（2.2）观察变量图

* 1. s="Progrmmingg"，实现了。

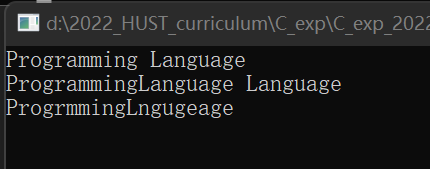
5-6 源程序改错（2.3）观察变量图

1. 改错后的运行结果
   1. 改错
      1. 第7行，b定义时需要给定长度。改为：

char a[] = "Language", b[100] = "Programming";

* + 1. 第16行结束后，t应该自减。添加：

t--；

* 1. 修改后的运行截图。

5-7 源程序修改后运行截图

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。

#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;

}

1. 填空后的程序

#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[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];

}

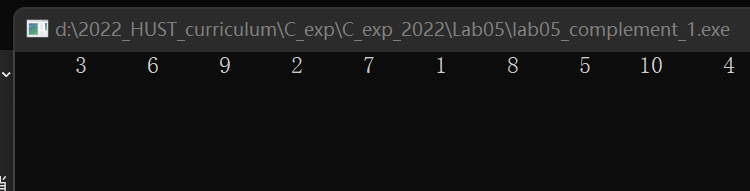
for (i = 0; i < M - 1; i++) /\* 按次序输出出圈人的编号 \*/

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

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

return 0;

}

1. 填空后的运行截图

5-8 填空后运行截图

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。

1. 修改后的源程序

#include <stdio.h>

#define M 10

#define N 3

int main(){

int a[M+1]={},now=0;

// a: 0 unkilled;1 killed

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

// M-1 to kill; 1 to live

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

if(++now > M)

now=1;

if(a[now])

j--;

}

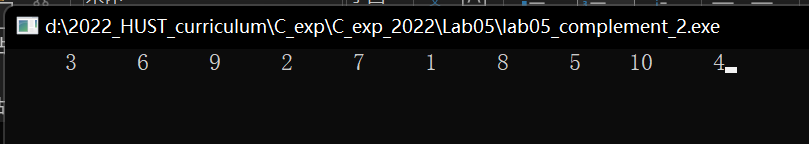
printf("%6d",now);

a[now] = 1;//killed

}

return 0;

}

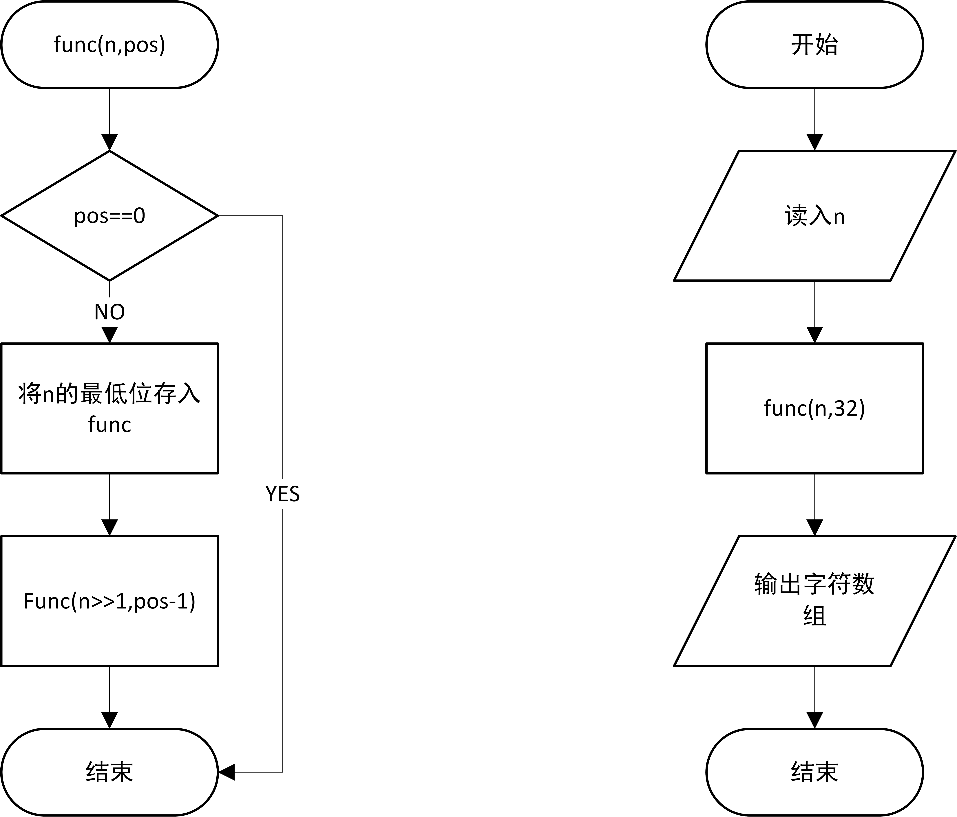
1. 运行截图

5-9 源程序修改后运行截图

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。
   1. 程序流程图

5-8 程序设计（1）算法流程图

* 1. 源代码

#include<stdio.h>

char str[40];

void func(int n,int pos){

if(pos==0)

return ;

str[pos-1] = '0'+(n&1);

func(n>>1,pos-1);

}

int main(){

int n;

scanf("%d",&n);

func(n,32);

for(int i=0;i<32;i++){

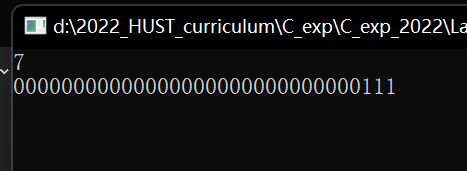
printf("%c",str[i]);

}

return 0;

}

* 1. 测试截图

图5-9 程序设计（1）运行截图

（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。

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!”。

1. 程序设计思路
   * + 1. 输入需求数字
       2. 按照需求数字进行对应操作：
          1. 输入
          2. 排序：采用冒泡排序确保稳定性
          3. 查找：采用特殊处理过的二分查找确保出现相同成绩时输出靠前的元素。
          4. 输出
       3. 返回1
2. 程序源码

#include <stdio.h>

#include <string.h>

struct Student

{

// public:

char name[1000];

int score;

int order;

} stu[1000];

typedef struct Student Stt;

int n;

void score\_input()

{

scanf("%d", &n);

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

{

scanf("%s%d", stu[i].name, &stu[i].score);

stu[i].order = i;

}

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

}

void stu\_swap(Stt \*s1, Stt \*s2)

{

// printf("before:%s %d\n%s %d\n", s1->name, s1->score, s2->name, s2->score);

Stt t = \*s1;

strcpy(s1->name, s2->name);

strcpy(s2->name, t.name);

s1->order = s2->order;

s2->order = t.order;

s1->score = s2->score;

s2->score = t.score;

// printf("after:%s %d\n%s %d\n", s1->name, s1->score, s2->name, s2->score);

};

void score\_sort()

{

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

{

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

{

if (stu[j].score < stu[j + 1].score)

stu\_swap(&stu[j + 1], &stu[j]);

}

}

// printf("Reorder finished!\n")

}

void score\_print()

{

score\_sort();

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

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

}

int score\_find()

{

score\_sort();

int x;

scanf("%d", &x);

int l = 0, r = n - 1;

while (l < r)

{

int mid = (l + r) >>1;

if (stu[mid].score > x)

l = mid+1;

else

r = mid;

}

if (l == r && stu[l].score == x)

return l;

else

return -1;

}

void func()

{

while (1)

{

int temp;

int exit\_flag = 0;

scanf("%d", &temp);

switch (temp)

{

case 0:

exit\_flag = 1;

break;

case 1:

score\_input();

break;

case 2:

score\_sort();

printf("Reorder finished!\n");

break;

case 3:

score\_print();

break;

case 4:

;

int ret = score\_find();

if (ret == -1)

printf("not found!\n");

else

printf("%s %d\n", stu[ret].name, stu[ret].score);

break;

}

if (exit\_flag)

break;

}

}

int main()

{

func();

return 0;

}

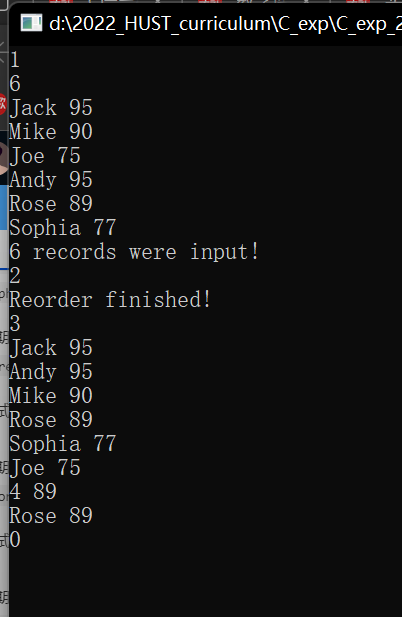
1. 程序运行截图

图5-10 程序设计（2）运行截图

（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。”

1. 求解思路
   * + 1. 输入n
       2. 若n为1，则答案加1，结束当前递归
       3. 从第n层开始，每层枚举摆放的列号
       4. 检查当前列号是否合法
          1. 若合法，返回2，进入下一层
          2. 若不合法，返回2，枚举下一个列号
       5. 输出答案
2. 源程序清单

#include <stdio.h>

int arr[15];

int ans;

void queen(int pos, int n)

{

/// @brief 求解第 pos 层的可行方案-从第n层到第1层的顺序

/// @param pos 当前行数

/// @param n 总共的行数

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

{

// iterater:enumerate i

// i : 第 n 行 皇后的列标

int flag = 0;

for (int j = n; j > pos; j--)

{

// checker: i is legal?

// j：n ~ pos+1 行

if (arr[j] == i // 同一列

|| arr[j] - j == i - pos // 同一个主对角线

|| arr[j] + j == pos + i) // 同一个次对角线

{

flag = 1;

break;

}

}

if (flag)

continue;

// 失败则放弃当前分支

arr[pos] = i;

if (pos == 1)

{

//到达最小规模问题

ans += 1;

}

else

{

queen(pos - 1, n);

}

}

}

int main()

{

int n;

scanf("%d", &n);

queen(n, n);

if (ans == 0)

{

printf("无解");

}

else

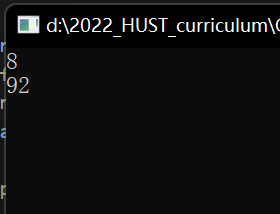
{

printf("%d", ans);

}

return 0;

}

1. 测试

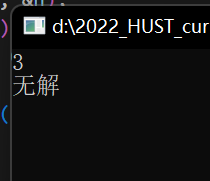
图5-11 程序设计（3）运行截图（1）

图5-12 程序设计（3）运行截图（2）

5.3 实验小结

（1）掌握了组的说明、初始化和使用。

（2）掌握了维数组作为函数参数时实参和形参的用法。

（3）掌握了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）掌握了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。