验5 数组实验

* 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。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);
9. printf("%s %s\n",b,a);
10. strdelc(b, 'a');
11. printf("%s\n",b);
12. return 0;
13. }
14. void strcate(char t[],char s[])
15. {
16. int i = 0, j = 0;
17. while(t[i++]) ;
18. while((t[i++] = s[j++] )!= '\0');
19. }
20. void strdelc(char s[], char c)
21. {
22. int j,k;
23. for(j=k=0; s[j] != '\0'; j++)
24. if(s[j] != c) s[k++] = s[j];
25. }

**解答：**

1. 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为12（如图5-1所示），t[i]为'L'。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"（如图5-2所示），未实现字符串连接。

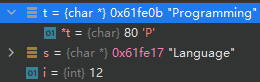


图5-1 实验5-1光条刚落在第二个while语句所在行时i的值截图



图5-2 实验5-1光条落在strcate函数块结束标记所在行时字符数组t和s内容截图

1. 跟踪进入函数strdelc时，字符数组s中的内容为"Programming"，字符c的值为'a'。单步执行for语句过程中，字符数组s, j和k值的变化如图5-3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5-4所示）未实现删除功能。

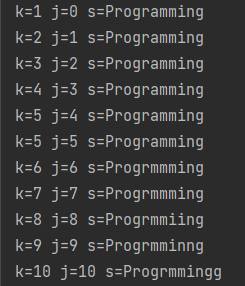


图5-3 实验5-1函数strdelc中单步执行for语句过程中，字符数组s, j和k值截图



图5-4 实验5-1函数strdelc结束时字符串s截图

1. 错误修改：
   1. 第6行，b数组更改范围，改为20。
   2. 第18行，添加i--。
   3. 第24行后添加s[k]='\0';。

修改后，源程序的清单如下：

#include<stdio.h>

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

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[20]="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';

}

1. 错误修改后运行结果如图5-4所示。

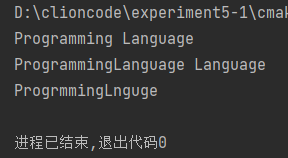


图5-5 实验5-1修改后运行结果截图

* + 1. 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. 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）实验思路如下：
6. 先定义a，b两个一维数组，利用for循环给a中每个元素标号。
7. 再用for循环进行报号。
8. 利用for循环，每3个报一次，并进行判断，若报号者为第一个人，则将位置j记为0，若报号者已经报过号了，则此次不计入报号，并从下一个开始继续报号。
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. 利用for循环将b中的报号者序号依次输出，最后输出a中唯一不为0的序号，即为获胜者。
11. 程序清单如下：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M],c[M-1];

int i, j, k,t;

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

a[i] = i + 1;

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

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

{

if(a[j]==0) k--;

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

}

b[M - i] = j ? a[j - 1] : a[M - 1];

if(j)

a[j-1]=0;

else

a[M-1]=0;

}

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

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

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

if(a[i]!=0) printf("%6d",a[i]);

return 0;

}

1. 测试。
2. 测试数据如表5-1所示。

表5-1 实验5-2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 理论结果 | 运行结果 |
|
| 用例1 | 3 6 9 2 7 1 8 5 10 4 | 3 6 9 2 7 1 8 5 10 4 |

1. 对应测试用例1的运行结果如图5-6所示。

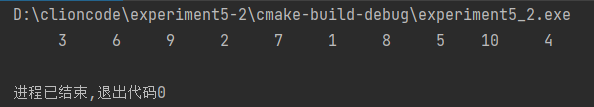


图5-6 实验5-2用例1的运行结果

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. **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. 实验思路如下：
2. 定义一个大小为32的字符数组，并初始化为‘0’。
3. 定义常量mask 0x00000001。
4. 使用for循环，每次循环取对应数字的一位并存储在字符数组中，并将该数字右移一位。再将该数组顺序打印，即为所得。
5. 程序清单如下：

#include <stdio.h>

int main() {

int m;

char a[33]={'0'};

scanf("%d",&m);

const int mask=0x00000001;

int i,j;

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

{

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

m>>=1;

}

for(j=0;j<32;j++)

printf("%c",a[j]);

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 | 10 | 00000000000000000000000000001010 | 00000000000000000000000000001010 |
| 用例2 | -33 | 11111111111111111111111111011111 | 11111111111111111111111111011111 |
| 用例3 | 0 | 00000000000000000000000000000000 | 00000000000000000000000000000000 |

1. 对应测试用例1的运行结果如图5-7所示。

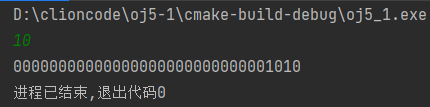


图5-7 程序设计题1用例1的运行结果截图

对应测试用例2的运行结果如图5-8所示。

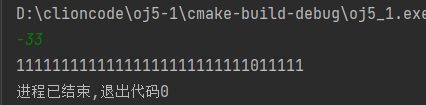


图5-8 程序设计题1用例2的运行结果截图

对应测试用例3的运行结果如图5-9所示。

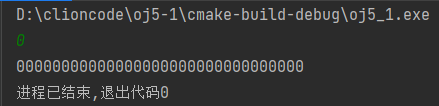


图5-9 程序设计题1用例3的运行结果截图

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值，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INPUT函数，并将n，整形数组和字符数组作为参数，利用for循环将姓名与成绩存储进数组中，存储完毕，输出“n records were input!”。
4. 定义SORT函数，并将n，整形数组和字符数组作为参数，利用冒泡排序，将整型数组又小到大排序，并同时用strcpy函数更改姓名数组，完成后输出“Reorder finished!”
5. 定义OUTPUT函数，并将n，整形数组和字符数组作为参数，利用for循环，输出排序后的姓名与成绩。
6. 定义FIND函数，利用二分查找，查找输入的成绩，若找到则输出成绩与对应的姓名，若没找到，则输出“not found!”。
7. 主函数利用switch将4个函数分割，并用while函数实现不断的输入。
8. 程序清单如下：

#include <stdio.h>

#include "string.h"

void INPUT(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);

}

void SORT(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");

}

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

{

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

{

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

}

}

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

{

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{

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

break;

}

}

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

}

int main(){

int i,n,k;

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

{

if(i==1)

{

scanf("%d",&n);

}

static int t;

t=n;

int a[20];

char b[20][100];

switch (i)

{

case 0:

return 0;

case 1:

INPUT(t,b,a);break;

case 2:

SORT(t,b,a);break;

case 3:

OUTPUT(t,b,a);break;

case 4:

scanf("%d",&k);

FIND(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 | 输入1，输入5；  再输入  a 80  b 77  c 88  d 98  e 78；  输入2；  输入3；  输入4，输入78；  输入4，输入100；  输入0。 | 依次输出  5 records were input!；  Reorder finished!；  d 98  c 88  a 80  e 78  b 77；  e 78；  not found!；  结束程序。 |

* 1. 对应测试用例1的运行结果如图5-10所示。

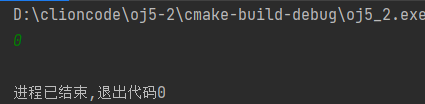


图5-10 程序设计题2用例1的运行结果截图

对应测试用例2的运行结果如下图所示。

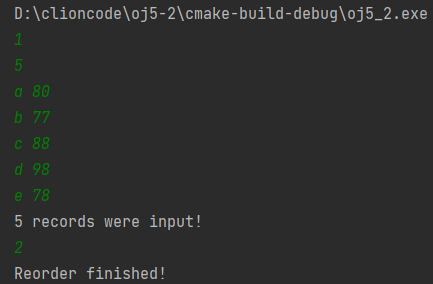


图5-11 程序设计题2用例2的运行结果截图

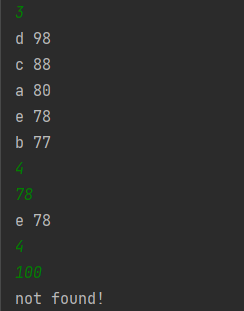


图5-12 程序设计题2用例2的运行结果截图

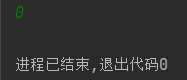


图5-13 程序设计题2用例2的运行结果截图

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. 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：
2. 定义全局变量count=0，size，和二维数组a[10][10]作为棋盘。
3. 定义check函数用来检测该位置的左斜上和右斜上和上方是否有皇后存在
4. 定义ok=1。
5. 若该位置不能放置皇后，则ok=0。
6. 返回ok的值。
7. 定义solve函数，将行数row作为参数，从第一行第一列进行check，若check为1，则执行solve（row+1），并将该位置标记为1，solve（row+1）结束后，将该位置还原为0，实现回溯，以size>=row为递归结束条件，每次递归结束都令count++。
8. 主函数中，接收输入的size，令solve从第0行开始执行，最后输出count的值。
9. 程序清单如下：

#include <stdio.h>

int size,count=0;

int a[10][10]={0};

int check(int row,int col);

void solve(int row){

int i;

if(row>=size) count++;

else

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

{

if(check(row,i))

{

a[row][i]=1;

solve(row+1);

a[row][i]=0;

}

}

}

int check(int row,int col)

{

int ok=1;

for(int i=0;i<row&&ok;i++)

{

if(a[i][col]==1)

ok=0;

}

for(int j=1;j<=col&&ok&&j<=row;j++)

{

if(a[row-j][col-j]==1)

ok=0;

}

for(int m=1;row-m>=0&&ok&&col+m<size;m++)

{

if(a[row-m][col+m]==1)

ok=0;

}

return ok;

}

int main(){

scanf("%d",&size);

solve(0);

if(count!=0) printf("%d",count);

else printf("无解");

return 0;

}

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. 对应测试用例1的运行结果如图5-14所示。

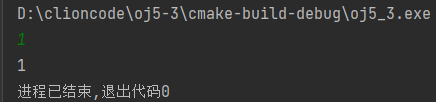


图5-14 程序设计题3用例1的运行结果截图

对应测试用例2的运行结果如图5-15所示。

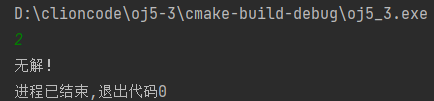


图5-15 程序设计题3用例2的运行结果截图

对应测试用例3的运行结果如图5-16所示。

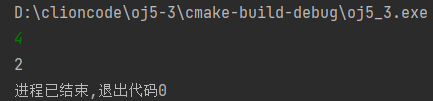


图5-16 程序设计题3用例3的运行结果截图

对应测试用例4的运行结果如图5-17所示。

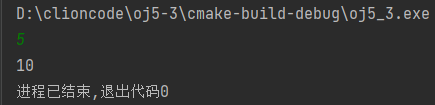


图5-17 程序设计题3用例4的运行结果截图

对应测试用例5的运行结果如图5-18所示。

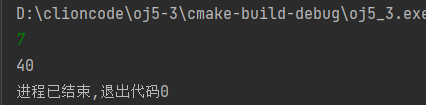


图5-18 程序设计题3用例5的运行结果截图

对应测试用例6的运行结果如图5-19所示。

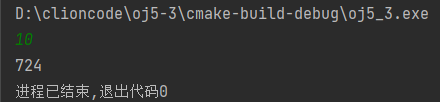


图5-19 程序设计题3用例6的运行结果截图

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。

5.3 实验小结

本次实验，首先对字符串处理函数的原理和应用加深了印象，包括数组的范围等问题，都有较深体会。其次在头歌的第二题中，应该算是我第一次接触类似完整程序的代码，其中对各个函数间的关系让我有很深刻的认识，也让我知道一些微小的地方，也可能对整个程序有着很大的影响，比如>和>=的区别。最后就是N皇后的问题，回溯算法是我仍然需要好好研究的一个问题。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计, 北京：科学出版社, 2013

[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