# 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 源程序改错题**

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。

源程序

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行引用自定义函数时，引用数组只需引用数组名，正确形式为：

sort(a,10);

1. 第18行的语句逻辑错误，为实现升序排序应将符号“<”改为符号“>”，正确形式为：

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

错误修改后源程序:

#include<stdio.h>

int main(void)

{

int a[10] = {27, 13, 5, 32, 23, 3, 17, 43, 55, 39};

void sort(int [],int);

int i;

sort(a,10);

for(i = 0; i < 10; i++)

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

printf("\n");

return 0;

}

void sort(int b[], int n)

{

int i, j, t;

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

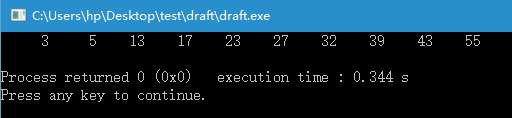
for ( j = 0; j < n - i - 1; j++)

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

t = b[j], b[j] = b[j+1], b[j+1] = t;

}

1. 错误修改后运行结果:



**5.2.2 源程序完善、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。

源程序：

#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。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：

（1）修改第一个下划线处的语句为：

a[j-1]

修改第二个下划线处的语句为：

a[i-1]

修改第三个下划线处的语句为：

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

替换后的程序如下所示：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M];

int i, j, k;

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(++j > i - 1) j = 0;

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

if(j)

for(k = --j; k < i; k++)

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

}

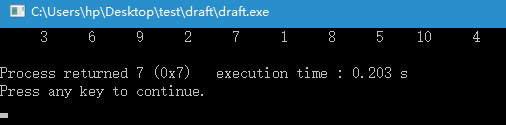
for(i = 0; i < M - 1; i++)

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

printf("%6d\n", a[0]);

}

运行截图：



（2）修改程序为：用0标记出圈人，当报数者不等于0时报数加一。

修改后源程序为：

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M];

int i, j, k;

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

a[i] = i + 1;

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

{

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

{

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

if (j)

{

if (a[j - 1] != 0) k++;

}

else if(j==0)

{

if (a[M - 1] != 0) k++;

}

}

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

if (j) a[j-1] = 0;

else if (!j) a[M - 1] = 0;

}

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

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

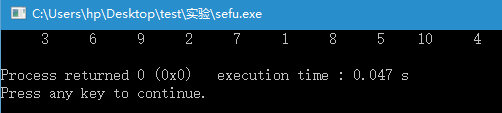
for(i=0;i<M-1;i++)

if(a[i]!=0)

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

}

运行截图：



**5.2.3 跟踪调试源程序**

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。

源程序：

#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++]) ;

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

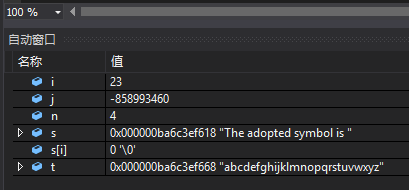
s[i++] = t[j++];

s[i] = '\0';

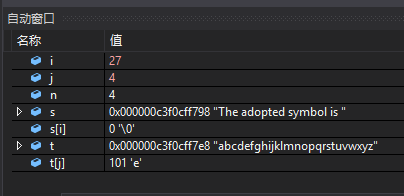
}

解答：

1. 单步执行源程序。进入函数strncat后，当光条落在for语句所在行时，i的值如图：



当光条落在strncat函数块结束标记（右花括号 }）所在行时, s、t的值如图：



1. 函数出错的原因：将n个字符加在了原字符串’\0’的后面，使其无法被输出，可在添加字符前加上语句i--，即修改后源程序：

#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--;

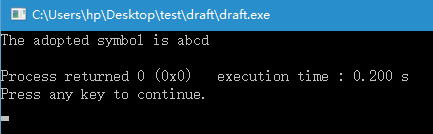
for(j = 0; j < n && t[j];)

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

s[i] = '\0';

}

修改后程序运行结果为The adopted symbol is abcd，如图：



**5.2.4 程序设计**

编写并上机调试运行能实现以下功能的程序。

（1）编写一个程序,从键盘读取数据，对一个3×4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：
2. 定义一个3\*4的二元数组为矩阵，并向二元数组中输入数字以对矩阵赋值；
3. 利用for循环输出未转置前的矩阵；
4. 定义函数transform将矩阵转置储存到另一个数组里，即：

int transform(int a[N][N],int b[N][N])

{

int m,n;

for(m=0;m<3;m++)

for(n=0;n<4;n++)

b[n][m]=a[m][n];

return 0;

}；

1. 使用for循环输出转置之后的矩阵；
2. 结束。
3. 程序清单：

#include<stdio.h>

#define N 10

int transform(int a[N][N],int b[N][N]);

int main (void)

{

int a[N][N],b[N][N];

int m,n;

//input

for(m=0;m<3;m++)

for(n=0;n<4;n++)

scanf("%d",&a[m][n]);

//before

for(m=0;m<3;m++)

{

for(n=0;n<4;n++)

printf("%5d",a[m][n]);

printf("\n");

}

printf("\n");

//transform

transform (a,b);

//after

for(m=0;m<4;m++)

{

for(n=0;n<3;n++)

printf("%5d",b[m][n]);

printf("\n");

}

}

int transform(int a[N][N],int b[N][N])

{

int m,n;

for(m=0;m<3;m++)

for(n=0;n<4;n++)

b[n][m]=a[m][n];

return 0;

}

1. 测试
2. 测试数据：

为方便分析测试结果，选择矩阵的值依次由1到12。如表5-1所示。

表5-1 编程题5.2.4.(1)的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例  （程序输入)从左到右从上到下  依次输入矩阵内的数值 | | | | 理 论 结 果 |
| 1 | 2 | 3 | 4 | 行列转置后的矩阵结果 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

1. 对应测试测试用例的运行结果如图5-1所示。

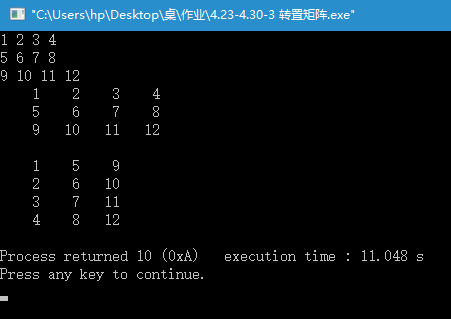


图5-1 编程题5.2.4.(1)的测试用例的运行结果

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。

（2）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：
2. 利用8\*sizeof(int)判断整数在内存中用二进制表示的长度；
3. 自定义函数itob，通过对数字内存表示的第i位与1<<i进行按位且操作，从而判断二进制中该为是否为1，是则向字符串中输入’1’，否则输入’0’；
4. 输入数字组数N，表示对N个数字进行处理，若未输入完N组数字，转3.1，否则，转4

3.1 对每个输入的数字使用自定义函数itob进行处理，处理后得到字符串a[length]，利用for循环将其输出后转3；

1. 结束。
2. 程序清单：

#include<stdio.h>

const int length=8\*sizeof(int);

void itob(int n,char a[])

{

int i;

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

a[length-i-1]=(n&(1<<i))?'1':'0';

}

int main(void)

{

int N,n,i;

char a[length];

scanf("%d",&N);

while(N--)

{

scanf("%d",&n);

itob(n,a);

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

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

printf("\n");

}

}

1. 测试
2. 测试数据

为便于分析结果，选择N分别为1、7、-1。如表5-2所示。

表5-2 编程题5.2.4.(2)的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| （N=3，3组数据）数字 |
| 用例1 | 1 | 输出结果 00000000000000000000000000000001 |
| 用例2 | 7 | 输出结果 00000000000000000000000000000111 |
| 用例3 | -1 | 输出结果 11111111111111111111111111111111 |

1. 对应测试测试用例1、2、3的运行结果如图5-2所示。

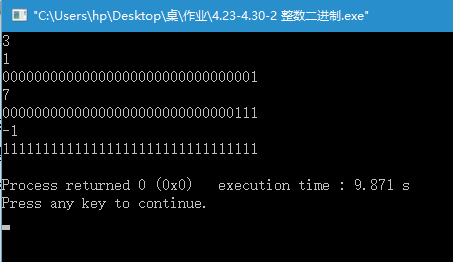


图5-2 编程题5.2.4.(2)的测试用例1、2、3的运行结果

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。

（3）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：
2. 自定义函数bubble\_sort利用冒泡排序对成绩进行排序，自定义函数binarysearch利用二分查找实现搜索成绩；
3. 输入n，代表将要输入n组数据，依次输入n组学生姓名与其对应的成绩，转2.1；
   1. 利用冒泡排序bubble\_sort对成绩进行排序并输出；
   2. 输入m，代表将要查找m个分数对应的数据；
   3. 输入m个分数，利用二分查找binarysearch将查找结果存储在数组中；
   4. 将查找到的数据依次输出，若未查找到则输出”Not found!”；
   5. 转3；
4. 结束。
5. 程序清单：

#include<stdio.h>

#include<string.h>

#define N 100

void bubble\_sort(char a[N][50],int b[],int n)

{

int p,i,temp;

char temp2[50];

for(p=0;p<n-1;p++)

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

if(b[i]<b[i+1])

{

temp=b[i];

b[i]=b[i+1];

b[i+1]=temp;

strcpy(temp2,a[i]);

strcpy(a[i],a[i+1]);

strcpy(a[i+1],temp2);

}

}

int binarysearch(int a[], int n, int key)

{

int low = 0, mid, high = n - 1;

while (low <= high)

{

mid = (low + high) / 2;

if (a[mid]>key) low = mid + 1;

else if (a[mid]<key) high = mid - 1;

else return mid;

}

return -1;

}

int main(void)

{

int i, n;

char a[N][50];

int b[N];

scanf("%d", &n);

getchar();

//input

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

scanf("%s %d", a[i], &b[i]);

//sort

bubble\_sort(a,b,n);

//output after sort

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

printf("%-20s %d\n", a[i], b[i]);

printf("\n");

//search

int m, key, save[N];

scanf("%d", &m);

getchar();

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

{

scanf("%d", &key);

save[i] = binarysearch(b, n, key);

}

//output after search

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

{

if (save[i] != -1)

printf("%-20s %d\n", a[save[i]], b[save[i]]);

else

printf("Not found!\n");

}

}

1. 测试
2. 测试数据

为便于分析结果，输入五组数据并进行排序、查找。如表5-3所示。查找数据输入如表5-4所示。

表5-3 编程题5.2.4.(3)的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | 理 论 结 果  （排序后） |
| 学生姓名 | 学生成绩 |
| 输入用例1 | ZhangChuanChao | 88 | LiSi 100 |
| 输入用例2 | XiaoHong | 95 | XiaoHong 95 |
| 输入用例3 | XiaoMing | 90 | XiaoMing 90 |
| 输入用例4 | LiSi | 100 | ZhangChuanChao 88 |
| 输入用例5 | XiDaDa | 83 | XiDaDa 83 |

表5-4 编程题5.2.4.(3)的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 查找成绩 | 理论结果 |
| 查找用例1 | 90 | XiaoMing 90 |
| 查找用例2 | 88 | ZhangChuanChao 88 |
| 查找用例3 | 83 | XiDaDa 83 |
| 查找用例4 | 59 | Not found！ |

1. 对应测试用例的运行结果如图5-3所示。

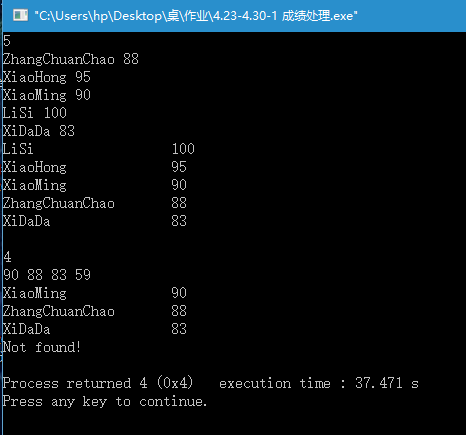


图5-3 编程题5.2.4.(3)的测试用例一的运行结果

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。

**5.2.5 选做题**

（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。

解答：

1. 解题思路：
2. 提示并输入字符串s、t，插入位置n；
3. 利用自定义函数strnins(s,t,n)实现字符串t插入s，具体流程转2.1；
   1. 将s第n个字符之后的所有字符存入字符串temp；
   2. 将字符串t插入字符串s的第n个字符之后；
   3. 将字符串temp中存储的剩余字符填充回字符串s中，最后在字符串中输入’\0’以结尾，转3；
4. 输出处理后的字符串转4；
5. 结束。
6. 程序清单：

#include<stdio.h>

#include<string.h>

#define N 100

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

int main (void)

{

char a[2][N];

int n;

printf("Please input string s:\n");

scanf("%[^\n]",a[0]);

getchar();

printf("Please input string t:\n");

scanf("%[^\n]",a[1]);

getchar();

printf("Please input the number to insert t to s:\n");

scanf("%d",&n);

strnins(a[0],a[1],n);

printf("After insert:\n%s\n",a[0]);

}

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

{

char temp[N];

int i,tempn=n;

//save

for(i=0;tempn<=strlen(s);)

temp[i++]=s[tempn++];

//insert

for(i=0;t[i]!='\0';)

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

//copy

for(i=0;temp[i]!='\0';)

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

s[n]='\0';

}

1. 测试
2. 测试数据

为便于分析结果，选择字符串s、t与插入位置n分别为88、6666666、1，888888888、66、5，The words are :、abcdefg、9。如表5-2所示。

表5-4 编程题5.2.4.(2)的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 |
| 字符串s | 字符串t | 插入位置n |
| 用例1 | 88 | 6666666 | 1 | 输出结果 866666668 |
| 用例2 | 888888888 | 66 | 5 | 输出结果 88888668888 |
| 用例3 | The words are : | abcdefg | 9 | 输出结果 The wordsabcdefg are : |

1. 对应测试测试用例1的运行结果如图5-4所示。

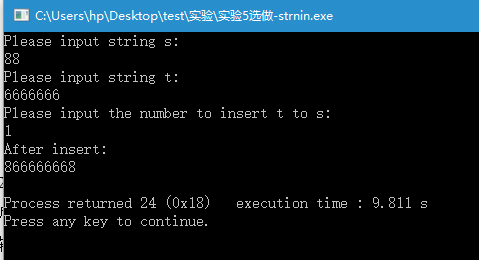


图5-4 选做题5.2.5.(1)的测试用例1的运行结果

1. 对应测试测试用例2的运行结果如图5-5所示。

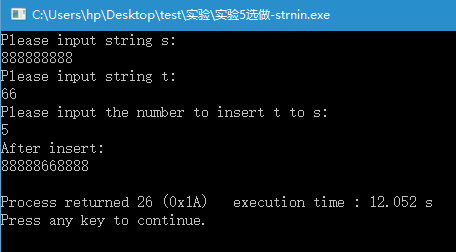


图5-5 选做题5.2.5.(1)的测试用例2的运行结果

1. 对应测试测试用例3的运行结果如图5-6所示。

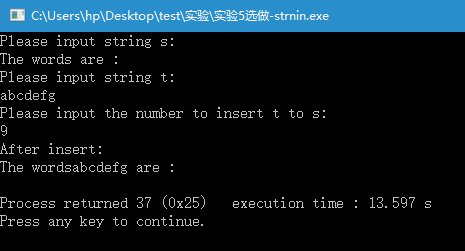


图5-6 编程题5.2.5.(1)的测试用例3的运行结果

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。

（2）编写一个实现八皇后问题的程序，即：在8×8放个国际象棋盘上放置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。

解答：

1. 解题思路：
2. 为方便对程序进行逻辑分析，定义大小为9的数组，取后八个数作为棋盘上的1~8列，数组中元素的值代表行数；
3. 初始化8\*8的棋盘，未放置皇后；
4. 在第一列第一行放置第一个皇后；
5. 在第下一列第一行放置一个皇后，转4.1；
   1. 通过函数check判断，若放置的皇后与之前任意皇后位于同一行或同一斜线，返回4并将放置皇后的位置下移，否则转4.2；
   2. 若未到达最后一列且行数未超出棋盘，则转4，若行数超出棋盘，回溯到上一列重新放置皇后并转4.1，否则转4.3；
   3. 若到达最后一列且行数未超出棋盘，解的个数计数加1，后输出可行的放置方案即八个数字，每个数字对应该列的皇后位置，若超出棋盘，回溯到上一行放置皇后，回溯完成所有情况后，转5；
6. 输出总的解的个数并转6；
7. 结束。
8. 程序清单：

#include<stdio.h>

#define n 8

int x[n+1];

int check(int k);

int main(void)

{

int count=0;

int i,k=1;

//flash

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

x[i]=0;

//main

while(k)

{

x[k]=x[k]+1; //place a queen

while(x[k]<=n && !check(k))

x[k]=x[k]+1;//find a right queen

if(x[k]<=n && k<n)//next queen

k++;

else if(x[k]<=n && k==n)//output

{

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

printf("%d ",x[i]);

printf("\n\n");

count++;

}

else//go back to last line

{

x[k]=0;

k--;

}

}

printf("sum up : %d\n",count);

}

int check(int k)

{

int i;

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

if(x[k]==x[i] || (k-i)==(x[k]-x[i]) || (k-i)==(x[i]-x[k]))//flase?

return 0;

return 1;

}

1. 测试
2. 测试数据

无。

1. 对应测试的运行结果如图5-7所示。

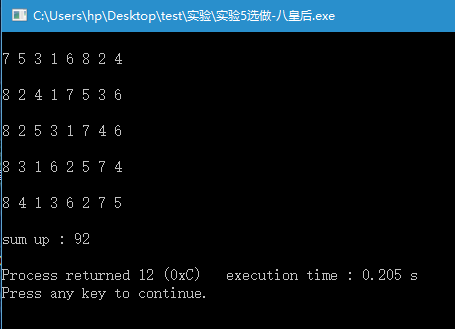


图5-7 选做题5.2.5. (2)的测试运行结果

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。

## 5.3 自设题

**（1）**自设实验题目：定义一个函数strrindex(s,t)，返回t在s中最右边出现的位置（s中的第1个字符的位置为0），找不到返回-1。

**（2）**实验目的：通过设计实验程序，理解字符数组的存储方式，并熟练对字符数组进行的操作。

**（3）**实验程序：

#include <stdio.h>

#include<string.h>

int strrindex(char s[],char t[])

{

int a,b,r=-1;

a=strlen(s);

b=strlen(t);

for(;a>=0;a--)

{

while(s[a]==t[b])

{

for(;(s[a]==t[b])&&(b>=0)&&(a>=0);a--,b--)

if(b==0)

{

r=a;

return r;

}

}

}

return -1;

}

#define N 80

int main()

{

char s[N+1],t[N+1];

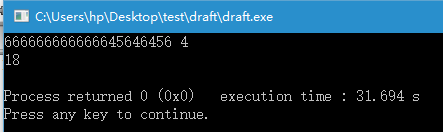
scanf("%s%s",s,t);

printf("%d\n",strrindex(s,t));

return 0;

}

**（4）**实验用例：如图



**（5）**实验结论：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。

## 5.4 实验小结

在实验中主要遇到的问题有：

1. 在5.2.2源程序完善、修改、替换的瑟夫问题中耗费了较多的时间填补未完成的程序，期间利用跟踪调试逐步分析程序直至最后调试成功；
2. 在5.2.2源程序完善、修改、替换的瑟夫问题中，替换原程序的时候因为没有把握if、else if语句的连锁关系，使更改后的程序的逻辑出现了错误，后通过跟踪调试发现问题，并在询问老师后理清了语句的使用规则，排除了问题；
3. 解决选做题八皇后问题时有较大难度，通过一些搜索学习，最后采用了自己理解较为简单的一种方法来处理问题。

体会：

1. 在编写程序时需严加注意语句的使用规则，越是细节的方面越不可轻视，如else if在无花括号限制的情况下会与最近的if语句连锁，若不注意可能造成结果与预期不符合；
2. 善于使用跟踪调试，会对个人理解他人程序有很大帮助，熟练使用跟踪调试也能提升程序员自己的编程效率；
3. 在5.2.4程序设计（2）中应注意编译环境的不同，需采用sizeof在不同情况下给出不同的解答以使程序具有可移植性，在其他程序的编写过程中也需要注意这类问题；
4. 在尝试解决八皇后问题，接触到了诸如回溯、八重嵌套、十行内处理n皇后问题等难度较高的编程问题，意识到自己的水平还十分有限，还需要通过各种各样的途径进行进一步的学习、长久的坚持。