**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

**5.2.1**源程序改错

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：**

#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[0],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;

}

【**分析及改正**】本程序共存在2处错误，分析如下：

（1）sort(a[0],10);

错误原因：函数形参是指针（或数组首地址），而传进去的是数组元素

改正：sort(a,10);

（2）sort(a,10);

错误原因：使用函数前未声明

改正：在主函数前加上声明

**5.2.** 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;

}

**【分析】前两个空及判断j是否为0，来决定b存什么，因此为a[j-1]:a[i-1];第二个空则是简单的把后面的人往前移动，所以是a[k]=a[k+1];**

**【程序】#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++)

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++)

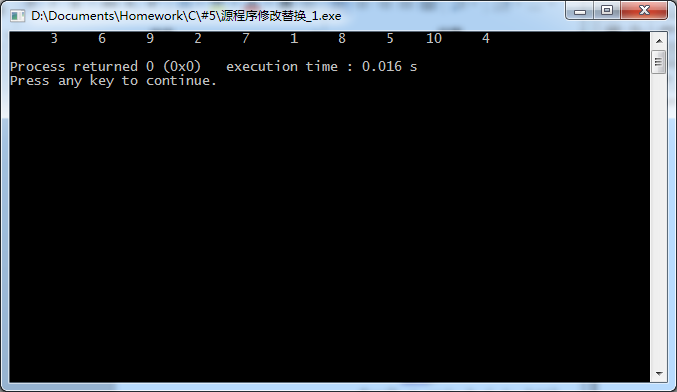
printf("%6d",b[i]);

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

return 0;

}

**【测试】**输出与手动模拟的结果相同



(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。

**【分析】**因为源程序已经有b[n]作为做标记的数组，所以只将压缩数组删除，并用在计数中做出相应的修改（以把剔除的人的编号改为0作为标记）即可。

**【程序】**

#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 > 0; i--){

for(k = 0;;){

k=a[j]?k+1:k;

if(k==N)

break;

if(++j > M-1)

j = 0;

}

b[M-i] = a[j];

a[j]=0;

}

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

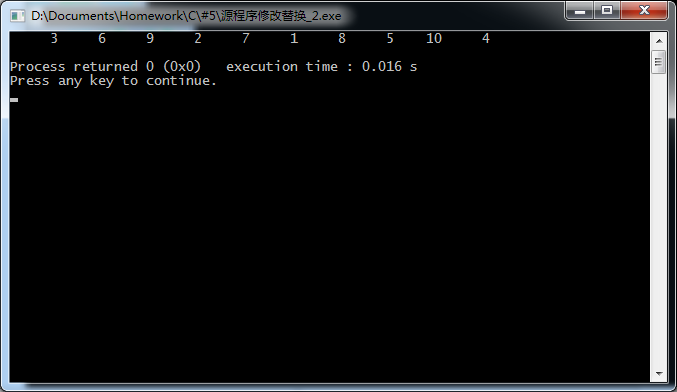
printf("%6d",b[i]);

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

return 0;

}

**【测试】**与上一题的功能相同



**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++];

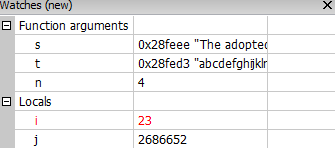
s[i] = '\0';

}

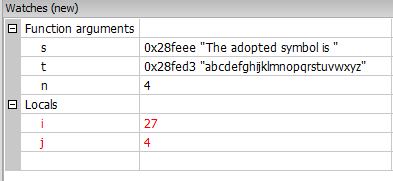
（1）

**【跟踪】**

落在for时，i为23



落在函数块结束标记所在行时，s和t如下图所示



（2）

**【分析】**传入strncat的并不是两个数组本身，而是两个数组的副本，只有传入指针才能进行深度复制。并且源程序忘了修改源字符串后面的’\0’

**【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--;

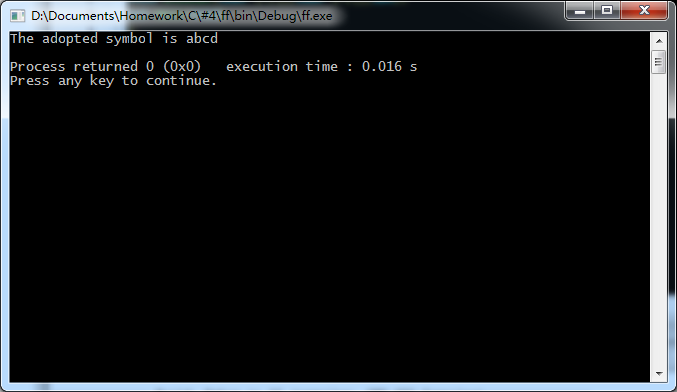
for(j = 0; j < n && (\*t)[j];)

(\*s)[i++] = (\*t)[j++];

(\*s)[i] = '\0';

}

**【测试】**输出与预想相符



**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。

(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。

(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!”。

（1）**【分析】**关键步骤是如何求转置矩阵，可初始化一个4\*3的矩阵在从原矩阵进行转录即可，或在读入是同时进行转置，更节省时间。

**【程序】**

#include<stdio.h>

int main(void){

int src[3][4],dst[4][3];

int i,j;

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

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

scanf("%d",&src[i][j]),dst[j][i]=src[i][j];

for(i=0;(putchar('\n')),i<3;i++)

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

printf("%-4d",src[i][j]);

for(i=0;(putchar('\n')),i<4;i++)

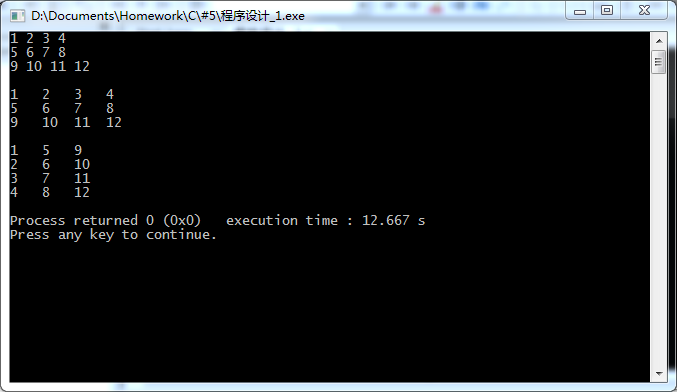
for(j=0;j<3;j++)

printf("%-4d",dst[i][j]);

return 0;

}

**【测试】**输入一个矩阵，可以正确输出其本身和其转置



（2）**【分析】**使用“与”位运算和一个足够大的字符串数组即可，关键是如何确定字符串的最大长度，它与整形变量的长度有关，输出时从第一个非零位开始输出即可。

**【程序】**

#include<stdio.h>

#define MAX\_STR\_LEN (sizeof(int)<<3)

int main(void){

int in,i;

char out[MAX\_STR\_LEN+1];

scanf("%d",&in);

for(i=0;i<MAX\_STR\_LEN;i++)

out[i]=((in)&(1<<(MAX\_STR\_LEN-i-1)))?'1':'0';

out[i]='\0';

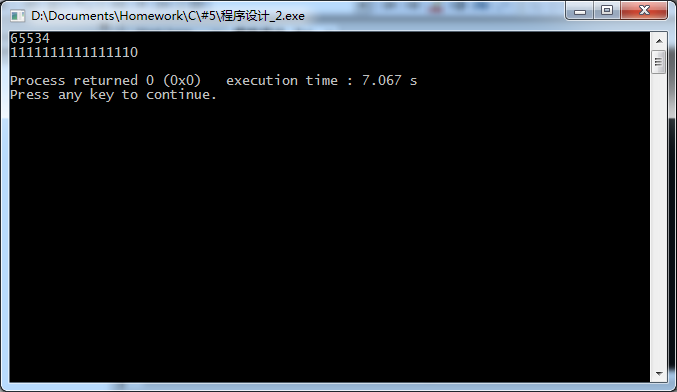
for(i=0;out[i]=='0';i++);

printf("%s\n",i==MAX\_STR\_LEN?"0":&out[i]);

return 0;

}

**【测试】**程序能正确输出输入整数的二进制形式



（3）**【分析】**使用两个数组分别表示名字和成绩，调整的时候同步调整，搜索的时候使用成绩数组进行二分查找即可。

**【程序】**

#include<stdio.h>

#include<stdlib.h>

#define MAX\_STU\_NUM 200

#define MAX\_NAM\_LEN 100

typedef struct{

int score;

char name[MAX\_NAM\_LEN];

}Stu;

void m\_sort(Stu \*, int);

Stu b\_find(Stu \*, int, int);

int main(void){

Stu stu[MAX\_STU\_NUM];

int N;

scanf("%d",&N); //N students

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

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

}

m\_sort(stu,N);

putchar('\n');

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

printf("%s %d\n",stu[N-i-1].name,stu[N-i-1].score);

printf("\nfind:");

int target;

scanf("%d",&target);

Stu temp = b\_find(stu,N,target);

if(temp.score==-1)

printf("%s\n","not found!\n");

else

printf("%s %d\n",temp.name,temp.score);

}

void m\_sort(Stu \*begin, int len){

if(len==1)

return;

m\_sort(begin,len/2);

m\_sort(&begin[len/2],len-len/2);

Stu buff[len];

int i=0,j=len/2,counter=0;

for(;i<len/2 && j<len;counter++){

if(begin[i].score<begin[j].score)

buff[counter]=begin[i],i++;

else

buff[counter]=begin[j],j++;

}

for(;i<len/2;counter++,i++)

buff[counter]=begin[i];

for(;j<len;counter++,j++)

buff[counter]=begin[j];

for(counter=0;counter<len;counter++)

begin[counter]=buff[counter];

return;

}

Stu b\_find(Stu \*stu,int len,int target){ //Assume the Stu sequence is already sorted

if(len==1 && stu[0].score!=target){

Stu temp = {-1,"not found"};

return temp;

}

if(stu[len/2].score==target)

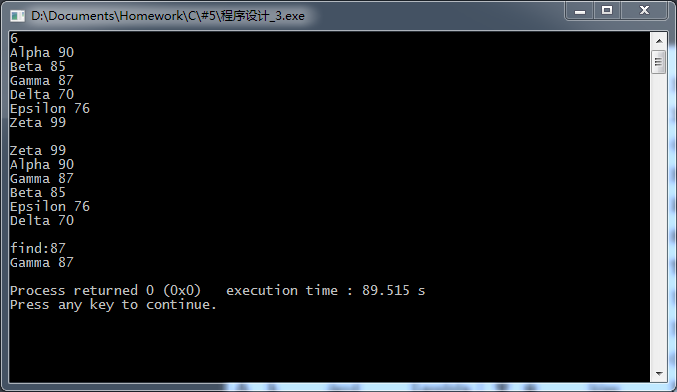
return stu[len/2];

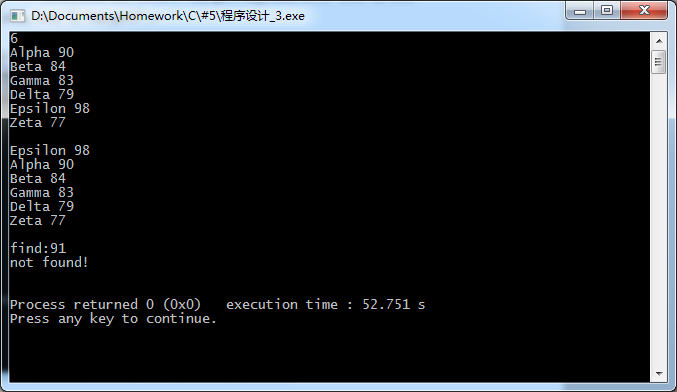
else

return stu[len/2].score<target?b\_find(stu+len/2,len-len/2,target):b\_find(stu,len/2,target);

}

**【测试】**录入6个学生的，分别用找得到和找不到的成绩进行查找测试。





输出结果与预想一致。

**5．**选做题

编写并上机调试运行能实现以下功能的函数和程序。

(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。

(2) 编写一个实现八皇后问题的程序，即：在8\*8方格国际象棋盘上放置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。

（1）

**【分析】**将s从t处分为两段，将t拼接到前面一段后再将后面一段添加到前述拼接好的字符串之后。

**【程序】**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char \*strnins(char \*, const char \*, int n);

int main(void){

char s[100],t[100];

int n;

scanf("%s %s %d",s,t,&n);

char \*out = strnins(s,t,n);

printf("%s",out);

}

char \*strnins(char \*s, const char \*t, int n){

char \*temp = (char \*)malloc(strlen(s)+strlen(t)+1);

strcpy(temp,s);

temp[n]='\0';

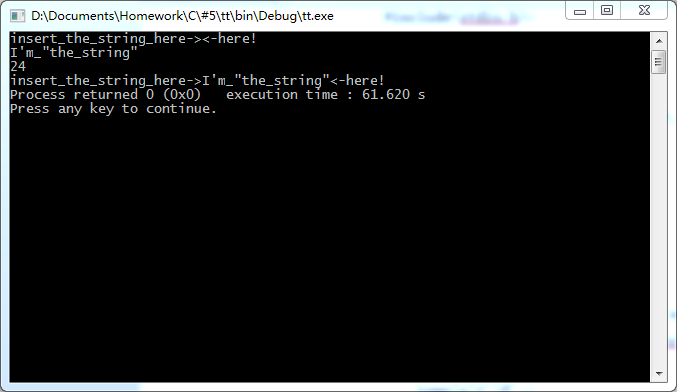
strcat(temp,t);

strcat(temp,s+n);

return temp;

}

**【测试】**输入s，t，n观察是否按题目要求插入



（2）

**【分析】**用一个长为8的数组表示棋盘，第i个元素存的值j表示皇后在第i行第j列，判断各种情况是否符合条件。不考虑旋转以及翻转。如果用八重循环将会造成8^8=16777216种情况，会造成时间复杂度很高，而且会考虑大量不必要的情况。将每一行的皇后的所在列标号排成一串，再用字典序全排列，只需判断8!=40320种情况，此算法更优。但用回溯法判断的可能性将更少，因为在深度搜索遇到第一个不符合条件的行就会进行回溯。因此本题采用回溯法。

**【程序】**

#include<stdio.h>

#include<stdlib.h>

int counter=0;

void search(int (\*)[8],int );

int can\_place(int (\*)[8], int, int);

int main(void){

int board[8];

search(&board,0);

}

void search(int (\*board)[8],int row){

if(row==8){ //reach the last line

printf("%d:\n",++counter);

int i,j;

for(i=0;i<8;i++){

for(j=0;j<(\*board)[i];j++)

putchar('.');

putchar('Q');

for(;j<8;j++)

putchar('.');

putchar('\n');

}

return;

}

int col=0;

for(;col<8;col++){

if(can\_place(board,row,col)){

(\*board)[row]=col;

search(board,row+1);

}

}

if(col==8)

return;

}

int can\_place(int (\*board)[8], int row, int col){

int x,y;

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

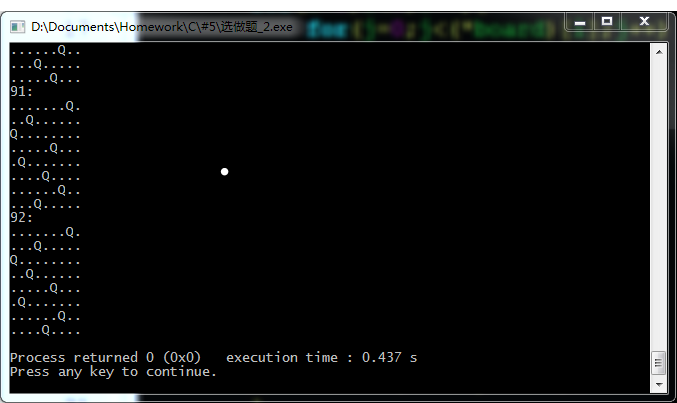
if((\*board)[i]==col || abs(i-row)==abs((\*board)[i]-col))

return 0;

return 1;

}

**【测试】**此为部分输出结果，全部输出结果已重定向到选做题\_2\_out.txt(双击可打开)



**5．3实验体会**

通过此次试验，我进一步了解了数组的说明、初始化和使用。掌握了一维数组作为函数参数时实参和形参的用法。并掌握了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，最重要的是学会了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。

此外，我对程序的调试有了更深入的了解，学习了算法的优化，了解了好的算法对于时间及空间复杂度的重要性。