一、数组与指针

1. 程序要求：实现对5个数（a[0]～a[4]，数组元素从键盘读入）进行从大到小起泡法排序。输入第一组数据：1 2 3 4 5；输入第二组数据：3 5 4 1 2；验证程序的正确性。
2. 程序要求：实现三阶方阵转置，并打印转置后的方阵。

测试主函数为：

int main()

{

int a[3][3]={1,2,3,4,5,6,7,8,9};

**...//实现三阶方阵a的转置，并打印出来**

return 0;}

1. 程序要求：检查一个方阵是否关于主对角线对称。若不是，输出”no”；若是，输出”yes”。

测试主函数为：

int main()

{

int a[ ][4]={0,1,2,3,1,4,5,6,2,5,7,8,3,6,8,9};

int found=1;

**...//判断方阵是否为对称阵，若不是，found置为0**

if (found==0)

printf("no\n");

else

printf("yes\n");

return 0;}

1. 程序要求：找到数组中最大的元素，将它和最后一个元素对调。并将元素对调后的数组打印出来。（以上功能均要求用**指针实现**）

测试主函数为：

int main()

{ int num[5]={1,3,5,4,2};

**...**

return 0;

}

1. 用指针（**首地址偏移法**）实现将存放在数组中的一组数据按逆序重排，并打印重排后的数组。

测试主函数为：

#define N 7

int main( )

{int a[N]={1, 2, 3, 4, 11, 12, 13};

**...**

return 0;

}

1. 用**指针移动法**实现将存放在数组中的一组数据按逆序重排，并打印重排后的数组。

测试主函数为：

#define N 7

int main( )

{int a[N]={1, 2, 3, 4, 11, 12, 13};

**...**

return 0;

}

1. 用**元素指针**实现输出二维数组中的全部元素。

测试主函数为：

int main( )

{ int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};

**...**

return 0;}

1. 用**行指针**实现输出二维数组中的全部元素。

测试主函数为：

int main( )

{ int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};

**...**

return 0;}

1. 程序功能：fac函数通过指针返回n！，主函数定义指针，调用fac函数，打印结果。分别用两种方法实现fac，程序要求如下：
2. 定义fac为有返回值函数，结果通过return语句返回主函数；
3. 定义fac为无返回值函数，结果通过指针间接访问法访问主函数中的变量。

测试主函数为：

void main( )

{ int m;

float k;

printf("input m:");

scanf("%d",&m);

**...fac(...);//要求fac函数实现k=m!**

printf("result=%f",k);

}

10.编写函数strcat，实现两个字符串的连接，返回值为连接后字符串的首地址。

测试主函数为：

void main()

{ char str1[30] = "I learn ", \*str2 = "C language.";

char \*s;

**s = strcat(str1,str2);//strcat函数的返回值是指针** printf("%s\n", s); }

11.一个机器人位于一个 m x n 网格的左上角（起始点在下图中标记为“Start”）。



机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。问总共有多少条不同的路径？

示例 :

输入: m = 3, n = 2

输出: 3

解释:

从左上角开始，总共有 3 条路径可以到达右下角。

1. 向右 -> 向右 -> 向下

2. 向右 -> 向下 -> 向右

3. 向下 -> 向右 -> 向右

测试主函数：

int uniquePaths(int m, int n) {

**...//补充函数，返回路径数**

}

int main(){

int m=3,n=2;

int k=uniquePaths(m,n);

printf("共有%d条路径",k);

return 0;

}

12.一个机器人位于一个 m x n 网格的左上角，机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角。现在考虑网格中有障碍物。那么从左上角到右下角将会有多少条不同的路径？

网格中的障碍物和空位置分别用 1 和 0 来表示。

说明：m 和 n 的值均不超过 100。

示例 1:

输入:

[

  [0,0,0],

  [0,1,0],

  [0,0,0]

]

输出: 2

解释:

3x3 网格的正中间有一个障碍物。

从左上角到右下角一共有 2 条不同的路径：

1. 向右 -> 向右 -> 向下 -> 向下

2. 向下 -> 向下 -> 向右 -> 向右

参考答案：

int uniquePathsWithObstacles(int\* obstacleGrid, int obstacleGridRowSize, int obstacleGridColSize){

**...//补充函数，返回路径数**

}

int main(){

int a[3][3]={0,0,0,0,1,0,0,0,0};

int k=uniquePathsWithObstacles(a[0],3,3);

printf("共有%d条路径",k);

return 0;

}

13.给定一个 n × n 的二维矩阵表示一个图像，将图像顺时针旋转 90 度。

说明：

你必须在原地旋转图像，这意味着你需要直接修改输入的二维矩阵。请不要使用另一个矩阵来旋转图像。

示例 1:

给定 matrix =

[

[1,2,3],

[4,5,6],

[7,8,9]

],

原地旋转输入矩阵，使其变为:

[

[7,4,1],

[8,5,2],

[9,6,3]

]

参考答案：

void rotate(int\* matrix, int matrixRowSize, int matrixColSize){

**...//补充函数，实现原地旋转功能**

}

int main(){

int a[3][3]={1,2,3,4,5,6,7,8,9};

rotate(a[0],3,3);

int \*p;

for(p=a[0];p<a[0]+9;p++)

{

if((p-a[0])%3==0) printf("\n");

printf("%4d",\*p);

}

return 0;

}

14.给定一个包含 m x n 个元素的矩阵（m 行, n 列），请按照顺时针螺旋顺序，返回矩阵中的所有元素。

示例:

输入:

[

[ 1, 2, 3 ],

[ 4, 5, 6 ],

[ 7, 8, 9 ]

]

输出: [1,2,3,6,9,8,7,4,5]

参考答案：

int\* spiralOrder(int\* matrix, int matrixRowSize, int matrixColSize){

**...//补充函数，返回结果（一维数组）的首元素指针**

}

int main(){

int a[3][3]={1,2,3,4,5,6,7,8,9};

int matrixRowSize=3,matrixColSize=3;

int \*returnnum=spiralOrder(a[0], matrixRowSize, matrixColSize);

int \*p;

for(p=returnnum;p<returnnum+9;p++)

{

printf("%4d",\*p);

}

return 0;

}

**二、字符串与指针**

**1、通过字符指针变量输出一个字符串，比如输出“I love China!”**

**2、将字符串a “I am student”复制为字符串b，然后输出字符串b。**

**3、实现字符串"I love "和“China”的拼接。**

**4、实现字符串的逆序输出。**

**5、统计带数字的字符串中的数字个数，并且输出数字。**

**6、统计字符串中的单词个数，单词之间用空格隔开。**

**7、字符替换。要求用函数replace将用户输入的字符串中的字符t(T)都替换为e(E)，并返回替换字符的个数。**

**8、编写一个程序，输入星期，输出该星期的英文名。用指针数组处理。**

**9、有5个字符串，首先将它们按照字符串中的字符个数由小到大排列， 再分别取出每个字符串的第三个字母合并成一个新的字符串输出（若少于三个字符的输出空格）。要求：利用字符串指针和指针数组实现。**

**10、定义一个动态数组，长度为变量n，用随机数给数组各元素赋值，然后对数组各单元排序，定义swap函数交换数据单元，要求参数使用指针传递。**

三、位操作

1. 不同系统对带符号负数的右移方式可能不同，有的系统是逻辑右移，即高位补0；有的系统是算术右移，即高位补1。试编写程序验证本系统中右移操作是逻辑右移还是算术右移。
2. 两个不同长度的带符号整型数进行逻辑运算时，若长度短的数是负数时，不同系统处理方式可能不同，有的系统是高位补0，有的系统是符号扩展，即高位补1，试编写程序验证本系统在这种情况下是左端高位是补0还是补1。
3. 输入一个十进制整数（正、负），转换为十六进制显示，即实现printf(“%x”,x)的功能，但不能用这个函数，也不能用算术运算符，只能用位操作符。
4. 输入一个十进制整数（正、负），转换为二进制显示，不能用算术运算符，只能用位操作符。
5. 输入一个由0、1系列组成的16位二进制带符号数，转换为十六进制数和十进制数显示。例：输入0000 0000 0011 1001，输出0039，57

输入1111111111000111，输出FFC7，-57

1. 定义一个有3、5、6、9位4个位段的结构变量，键盘分别输入十进制数值赋给这4个位段，再分别以十进制、十六进制和二进制显示。

7、按4字节浮点数格式用0和1组成的字符串从键盘输入，转换为十进制数显示。例：输入0 1000 0100 1100 1100 1000 1111 0101 110，显示57.57。