**第一周实验报告**

陈锦涛 22920202204542

##### 第一题

#include<iostream>

using namespace std;

void mp\_sort(int a[])//冒泡排序函数

{

for(int i=0;i<4;i++)//进行4次冒泡

for(int j=0;j<4-i;j++) //冒泡过程

{

if(a[j]<a[j+1])swap(a[j],a[j+1]);

}

for(int i=0;i<5;i++)cout << a[i] << " ";

cout << endl;

}

int main()

{

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

int b[8]={3,5,4,1,2};

mp\_sort(a);

mp\_sort(b);

return 0;

}

样例输出



思路:通过冒泡方法完成排序

##### 第二题

#include<iostream>

using namespace std;

int main()

{

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

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

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

cout << a[j][i] << " ";//倒置输出

}

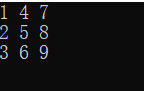
puts("");

}

return 0;

}

样例输出



##### 第三题

#include<iostream>

using namespace std;

int main()

{

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

bool found=true;

for(int i=0;i<4;i++)

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

{

if(j!=i && a[i][j]!=a[j][i]) {

found=false;

break;

}

}

if(!found)cout << "no" << endl;

else cout << "yes" << endl;

return 0;

}

样例输出 yes

##### 第四题

#include<iostream>

using namespace std;

int main()

{

int \*p;

int m=0;

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

for(int i=0;i<5;i++) if(num[i]>m){

m=num[i]; //擂台法找最大值

p=&num[i]; //记录最大值得地址

}

swap(\*p,num[4]); //将最大值地址内的数和最后一个数进行交换

for(int i=0;i<5;i++)cout << num[i] << ' ';

return 0;

}

样例输出



##### 第五题

#include<iostream>

using namespace std;

const int N=7;

int main()

{

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

int b[N];

int \* p=a; //定义首地址

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

{

b[i]=\*(p+i) ; //记录逆序数组

cout << b[i] << ' ';//采用首地址偏移i个位置使其正好以逆序输出

}

cout << endl;

return 0;

}

样例输出



##### 第六题

#include<iostream>

using namespace std;

const int N=7;

int main()

{

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

int b[N];

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

int \*p=&a[i]; // 定义第i个元素的地址

b[i]=\*((p+N-1-i\*2));

//将第i个元素的地址偏移至第N-1-2i的地址,使其正好逆序，并存储

cout << b[i] << ' ';

}

return 0;

}

样例输出



##### 第七题

#include<iostream>

using namespace std;

int main()

{

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

int \*p=a[0];

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

{

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

{

cout << \*p << ' ';

p++; //使该地址偏移一个位置至下一个元素的地址

}

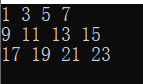
cout << endl;

}

return 0;

}

样例输出



##### 第八题

#include<iostream>

using namespace std;

int main()

{

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

int (\*p)[4] = & a[0]; //行指针定义法

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

{

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

cout << \*(\*(p+i)+j) << ' '; //p+i表示a[i]的地址,\*(p+i)+j表示a[i][j]的地址

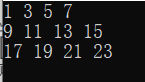
cout << endl;

}

return 0;

}

样例输出



##### 第九题

(1)

#include<iostream>

using namespace std;

float fac(int u)

{

if(u==1||u==0)return 1;//回溯至u=1或0时，返回1;

return u\*fac(u-1); //递归调用

}

int main()

{

int m;

float k;

printf("input m:");

scanf("%d",&m);

k=fac(m);

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

return 0;

}

样例输入 3

样例输出

input m:3

result=6.000000

(2)

#include<iostream>

using namespace std;

const int N=1e5+10;

int a[N];

void fac(int u,int \*p)

{

float k=1;

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

k\*=\*(p+i); //指针访问数组地址内容,循环完成阶乘

printf("result=%f",k); //输出

}

int main()

{

int m;

printf("input m:");

scanf("%d",&m);

a[0]=1; //0!=1

for(int i=1;i<=m;i++)a[i]=i; //将每个数存进数组里

int \*p;

p=a; //定义首地址

fac(m,p);

return 0;

}

样例输入 3

样例输出

input m:3

result=6.000000

##### 第十题

#include<iostream>

#include<cstring>

using namespace std;

char \*strcat(char str1[],char str2[])

{

char \*h;

h=str1+strlen(str1); //定义h为str1的最后一个元素的下一个地址

int i=0,j=strlen(str2);

while(j--)

\*(h++) = str2[i++];

//h+strlen(str1)-1是str1最后一个字符的地址

//将该地址的后面的地址的元素定义为str2的各元素就可连接起来

return str1;

}

int main()

{

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

char \*s;

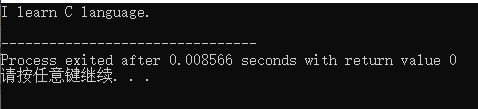
s=strcat(str1,str2);

printf("%s\n",s);

return 0;

}

样例输出



思路：将str1最后一个字符的地址的下一个地址定义为str2的首地址

##### 第十一题

#include<iostream>

using namespace std;

int m,n;

int dfs(int a,int b) //a表示向右,b表示向下

{

if(a==m || b==n)return 1;

//当向右或者向下走到边界时，就只有向下或向右一直走 这一种走法

else if(a<m && b<m)return dfs(a+1,b)+dfs(a,b+1);

//当向右或者向下都没有到边界时，有向右和向下两种走法

}

int main()

{

cin >> m >> n;

int k=dfs(1,1); //初始位置是(1,1)

printf("共有%d条路径",k);

return 0;

}

**测试数据**

样例1输入 3 2

样例输出

样例2输入 5 6

样例输出

思路:本题采用**深搜法**,本质上也就是求在**m+n**个总体中n个相同单位可以摆放的位置数量

##### 第十二题

#include<iostream>

using namespace std;

int a[100][100];

int m,n;

//x表示向右，y表示向下

//使用和第十题类似的深搜法，但是要增加了判断条件

int dfs(int x,int y)

{

//判断是否到达终点，若已到达，返回1

if(x==m && y==n)return 1;

//当此时的位置不处于边界线上，则有两种走法.

//判断这两种走法分别是否遇到障碍物 ,则可分4种情况

if(x<m && y<n ){

if(a[x+1][y]==1 && a[x][y+1]==1)return 0;

if(a[x+1][y]==1 && a[x][y+1]==0)return dfs(x,y+1);

if(a[x+1][y]==0 && a[x][y+1]==1)return dfs(x+1,y);

if(a[x+1][y]==0 && a[x][y+1]==0)return dfs(x+1,y)+dfs(x,y+1);

}

//当此时的位置到达了右边界或下边界时

//只有一直向下或者一直向右一种走法

//再判断能否继续向下或者继续向右即可

if(x<m && y==n){

if(a[x+1][y]==1)return 0;

if(a[x+1][y]==0)return dfs(x+1,y);

}

if(x==m && y<n){

if(a[x][y+1]==1)return 0;

if(a[x][y+1]==0)return dfs(x,y+1);

}

}

int main()

{

cin >> m >> n;

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

for(int j=1;j<=n;j++)cin >> a[i][j];

cout << dfs(1,1) << endl ;

}

测试数据：

样例1输入

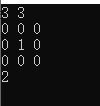
3 3

0 0 0

0 1 0

0 0 0

样例输出



样例2输入

3 3

0 0 0

0 1 1

0 0 0

样例输出



##### 第十三题

#include<iostream>

using namespace std;

int f[20][20]; //此数组是用来数组a中的元素是否进行了旋转

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

{

int x=matrixRowSize,y=x-1;

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

for(int j=0;j<=y;j++)

{

//进行3此反转即可完成四个对应元素之间的旋转

//matrix+x\*i+j 对应元素a[i][j]的地址

if(!f[i][j]){

swap(\*(matrix+i\*x+j),\*(matrix+x\*j+y-i));

swap(\*(matrix+x\*(y-i)+y-j),\*(matrix+x\*(y-j)+i));

swap(\*(matrix+x\*i+j),\*(matrix+x\*(y-i)+y-j));

f[i][j]=f[y-j][i]=f[y-i][y-j]=f[j][y-i]=1; //标记这个四个元素，表示已经完成旋转

}

}

}

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");

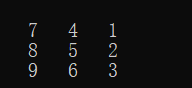
printf("%4d",\*p);

}

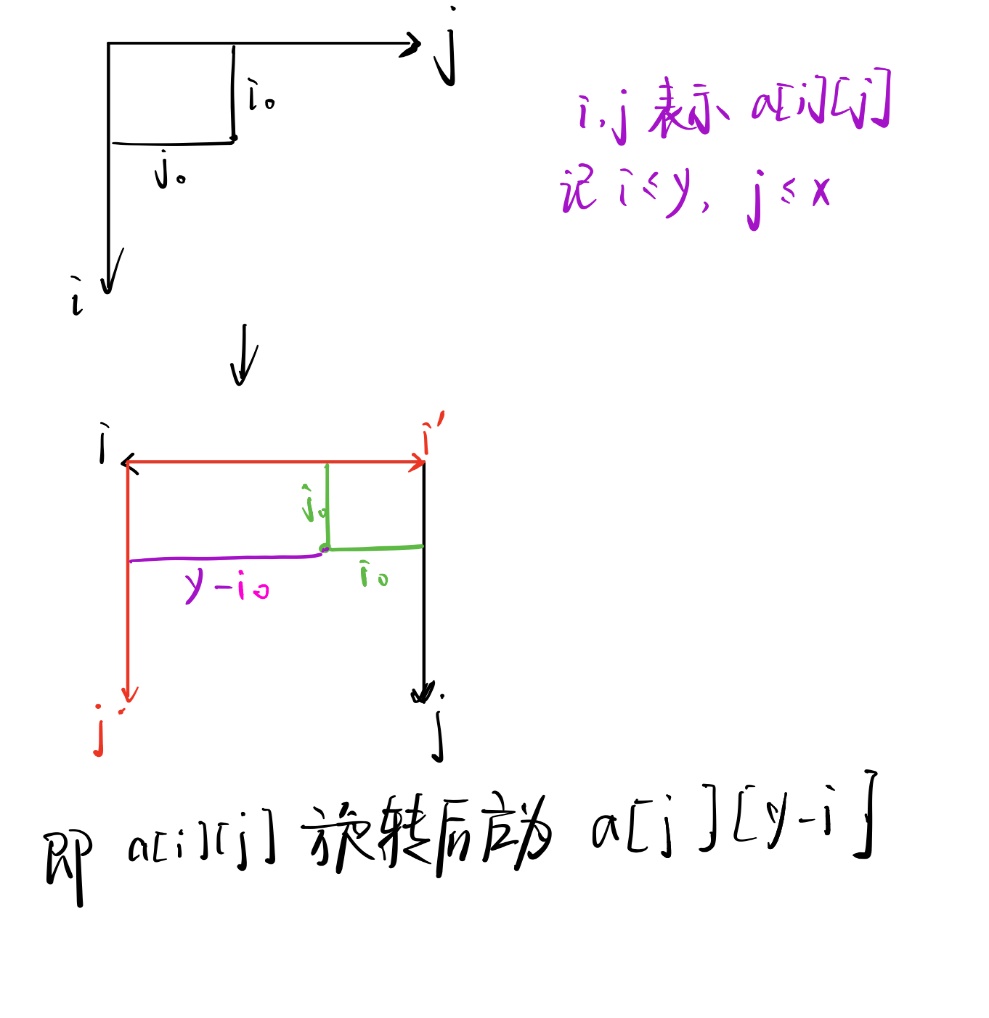
return 0;

}

样例输出



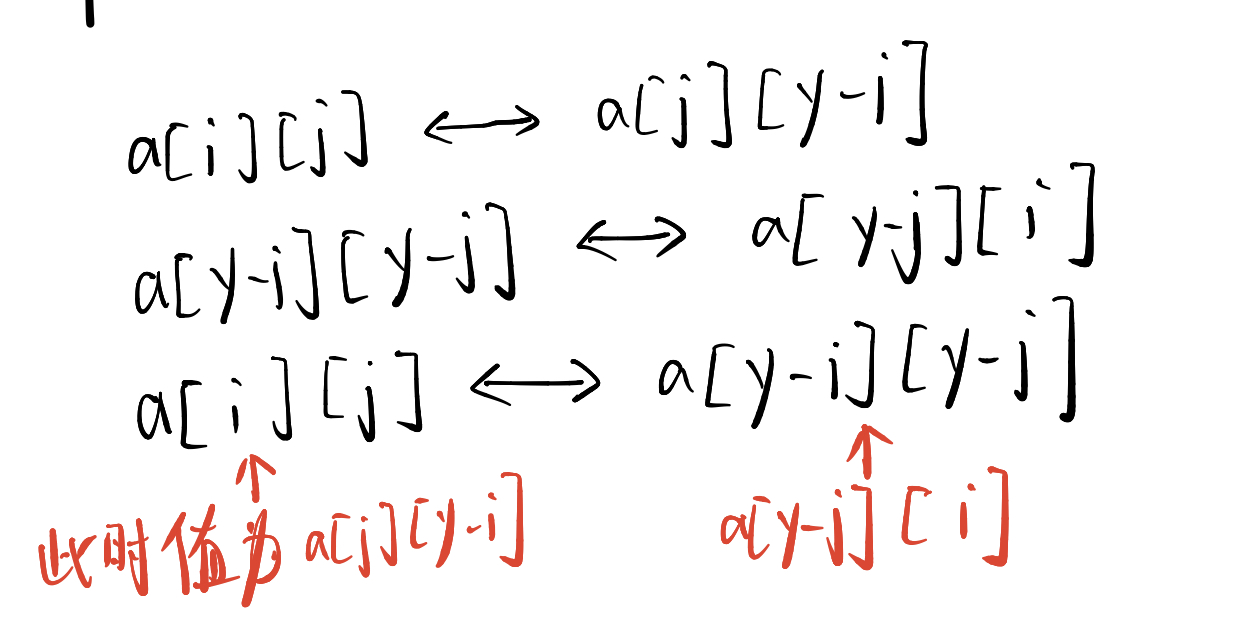
**思路：**a[i][j]经旋转后的坐标为a[y-j][i]

****

a[i][j] 🡪 a[j][y-i] 🡪 a[y-i][y-j] 🡪 a[y-j][i] 🡪 a[i][j]

要完成这个变换进行三次变换即可

如下



##### 第十四题

#include<iostream>

using namespace std;

int a[100][100];

void rotate(int m,int n)

{

    //up,down,left,right分别是上下左右四个边界的位置

    int up=0,down=0,right=0,left=0;

    //i=j=0表示初始时的a[0][0]，r表示已经输出的元素个数

    int i=0,j=0,r=0;

    while(1)

    {

        //一个循环内应依次向右，向下，向左，向上遍历所有元素

        for(j=left;j<n-right;j++)    //向右遍历

        {printf("%d ",a[i][j]);r++;}//每次输出的同时记录下输出的个数，为r

        j--;                         //上面的j溢出，故这里要-1，下面同理

        for(i=up+1;i<m-down;i++)     //向下遍历

        {printf("%d ",a[i][j]);r++;}

        i--;

        for(j--;j>=left;j--)           //j--是防止其输出了两个相同的元素，向z左遍历

        {printf("%d ",a[i][j]);r++;}

        j++;

        for(i--;i>up;i--)          //向上遍历

        {printf("%d ",a[i][j]);r++;};

        i++;

        up++;down++;right++;left++;//当顺时针走完一圈后，所有边界的位置都 +1

        if(r==m\*n)break;           //当输出完所有元素后，跳出循环，函数结束

    }

}

int main()

{

    int m,n;

    scanf("%d%d",&m,&n);

    for(int i=0;i<m;i++)

    for(int j=0;j<n;j++)scanf("%d",&a[i][j]);

    rotate(m,n);

    return 0;

}

**测试数据**

**样例1输入**

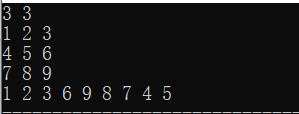
**3 3**

**1 2 3**

**4 5 6**

**7 8 9**

**样例1输出**



**样例2输入**

**4 5**

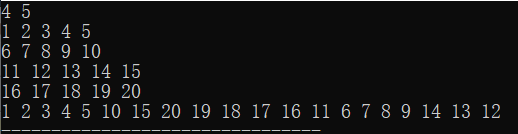
**1 2 3 4 5**

**6 7 8 9 10**

**11 12 13 14 15**

**16 17 18 19 20**

**样例2输出**



**思路：**

**要将矩阵顺时针螺旋输出，应从外圈开始一圈一圈的遍历，每一圈可视为一个循环，每个循环应依次从右，下，左，上四个方向遍历所有元素，如此便能输出所有元素**