基础内容

IO加速：

    ios::sync\_with\_stdio(false);

    cin.tie(0), cout.tie(0);

字符数字转换：

    //1.1.int->string by sstream

    int a[10];

    for (int i = 0; i < 10; i++)

        a[i] = i;

    stringstream s;

    for (int i = 0; i < 10; i++)

        s << a[i];

    string buf;  s >> buf;  cout << buf << endl;

    //1.2 int->string by to\_string

    double b = 1.222233;

    string double\_s = to\_string(b);//重载适应 int、float 和 double

    cout << double\_s << endl;

    //2.1.string->int by sstream

    string int\_s = "112233";

    stringstream s3(int\_s);

    int temp; s3 >> temp;

    cout << temp / 10 << " " << temp % 10 << endl;

    //2.2 string->int by atoi

    string str = "+668";

    cout << atoi(str.c\_str()) << endl;

    //int atoi(const char\* str) 将数字格式的字符串转换为整数类型,字符串最前可带正负号

    //c\_str()是string类的方法，string->char 来适应 strlen，atoi等函数

String内置函数：

s[i].find(" ") != string::npos

//找到返回从0开始的下标，没找到返回 string::npos

string attrs = s.substr(0, index);

//s.substr(start\_index,length) 从原字符串 中得到 子字符串

优先级队列初始化方式：

      priority\_queue< int, vector<int>, greater<int>> pq;

//越小的整数优先级越大，在队头，先出队列

二维差分：

#include<iostream>

using namespace std;

const int N=1010;

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

//差分定义：b[]是a[]的差分，则a[]是b[]的前缀和，公式为

//b[i]=a[i]-a[i-1] a[i]=b[0]+b[1]+...+b[i]

//类比到二维差分，b[][]是a[][]的差分，则a[][]是b[][]的前缀和

//a[i, j] = b[i, j] + a[i - 1, j] + a[i, j - 1] - a[i - 1, j - 1]

//b[i][j]=a[i][j] - a[i-1][j] - a[i][j-1] + a[i-1][j-1]

//若要把a[x1][y1]到a[x2][y2]中的数全加上c：

void insert(int x1,int y1,int x2,int y2,int num){

b[x1][y1]+=num;//从 x1,y1 辐射到 ∞,∞ 都会增加num

b[x2+1][y1]-=num;

b[x1][y2+1]-=num;

b[x2+1][y2+1]+=num;//注意这里的下标，按照大小矩形计算

}

int main(){

int n,m,q;cin>>n>>m>>q;

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

for(int j=1;j<=m;j++){

cin>>a[i][j];//初始化原数组

insert(i,j,i,j,a[i][j]);//初始化差分数组

}

}

while(q--){

int x1,y1,x2,y2,c;

cin>>x1>>y1>>x2>>y2>>c;

insert(x1,y1,x2,y2,c);//更新差分数组

}

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

for(int j=1;j<=m;j++){

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

//和一维前缀和中同理，如果要求a[i]=b[0]+b[1]+...+b[n];

//则b[1]+=b[0];b[2]+=b[1] 最后获得的b[n]就是a[n]

cout<<b[i][j]<<" ";

}

cout<<endl;

}

}