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 1:

输入：
```
[
 [ 1, 2, 3 ],
 [ 4, 5, 6 ],
 [ 7, 8, 9 ]
]
```
输出： `[1,2,3,6,9,8,7,4,5]`

示例 2:

输入：
```
[
  [1, 2, 3, 4],
  [5, 6, 7, 8],
  [9,10,11,12]
]
```
输出： `[1,2,3,4,8,12,11,10,9,5,6,7]`

思路，该题就是对数组的遍历，分为4个遍历状态，往左，往下，往右，往上，其中，状态是可以转移的，当其中一个状态到了尽头，就会触发他的下一个状态，下一个状态到尽头时触发下一个，轮回触发。然后，当进行一个轮回后，可以访问的边界减少1，已这个规律，将数添加进数组

代码：
```
class Solution {
public:
    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        int i=0;
        vector<int> result;
        if(matrix.size()==0){
            return result;
        }
        int x=0,y=0;
        int bj=0;
        while(result.size()!=matrix.size()*matrix[0].size()){
            if(i==0){
                result.push_back(matrix[x][y]);
                if(y+1==matrix[0].size()-bj){
                    x++;
                    i=1;
                }else{
                    y++;
```

```
                }
            }else if(i==1){
                result.push_back(matrix[x][y]);
                if(x+1==matrix.size()-bj){
                    y--;
                    i=2;
                }else{
                    x++;
                }
            }else if(i==2){
                result.push_back(matrix[x][y]);
                if(y==0+bj){
                    x--;
                    i=3;
                    bj++;
                }else{
                    y--;
                }
            }else{
                result.push_back(matrix[x][y]);
                if(x==0+bj){
                    y++;
                    i=0;
                }else{
                    x--;
                }
            }
        }
        return result;
    }
};
```