顺时针打印矩阵

输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字。

示例 1：

输入：matrix = [[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：matrix = [[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]

解题思路：递归输出 先考虑并控制最外圈输出顺序，其余可通过递归实现行++，列--控制

```
class Solution {
public:

    void print(int tx,int ly,int dx,int ry,vector<vector<int>>&matrix,vecto
r<int>&ret)
    {
        for(int j=ly;j<=ry;++j)
        {
            ret.push_back(matrix[tx][j]);//上面第一行
        }
        for(int i=tx+1;i<=dx;++i)
        {
            ret.push_back(matrix[i][ry]);//最靠右一列
        }
        int h=dx-tx+1;
        if(h>1)//如果为一行  则不需执行  不需要第三步
        {
            for(int rj=ry-1;rj>=ly;--rj)
            {
                ret.push_back(matrix[dx][rj]);//最下面一行  从右到左 反向输出
            }
        }
        int w=ry-ly+1;
        if(w>1)//如果为一列   则不需继续执行  不需要第四步
        {
            for(int ri=dx-1;ri>=tx+1;--ri)
            {
                ret.push_back(matrix[ri][ly]);//最左一列 从下向上输出
            }
        }
```

```
    }

    vector<int> spiralOrder(vector<vector<int>>& matrix) {
        vector<int>ret;
        if(matrix.empty()) return ret;
        int tx=0;//控制行  第一行
        int ly=0;//控制列  最左一列
        int dx=matrix.size()-1; //最后一行
        int ry=matrix[0].size()-1; //最右一列
        while(tx<=dx&&ly<=ry)
        {
            print(tx++,ly++,dx--,ry--,matrix,ret);
        }
        return ret;
    }
};
```