```
/*
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]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/spiral-matrix
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 由于本题输入不确定方阵,按圈遍历的话，最后一圈的结果可能是: 一行n列，m行一列,m行n列.
  - 方法一:按圈遍历,分类尾递归处理
  - 方法二:按圈遍历,分类迭代法

方法一:C++_按圈遍历,分类尾递归处理

```
class Solution
{
    private:
        vector<int> ret_val;
        int m;  /* 行数*/
        int n;  /* 列数*/
        int size;
        void __travelMatrix( vector<vector<int>>& vvi , int rows_start)
        {
            int cols_start  =   rows_start;
            int i           =   0;
            if((m<=0)||(n<=0))
            {
                return ;
            }
```

```
16
17              if(m==1)
18              {
19                  for(i = 0; i < n ; i++)
20                  {
21                      ret_val.push_back(vvi[rows_start][cols_start+i]);
22                  }
23
24                  return;
25              }
26
27              if(n==1)
28              {
29                  for(i = 0 ; i < m;i++)
30                  {
31                      ret_val.push_back(vvi[rows_start+i][cols_start]);
32                  }
33                  return;
34              }
35
36
37
38              /* up edge */
39              for(i = 0 ; i < n ; i++)
40              {
41                  ret_val.push_back(vvi[rows_start][cols_start+i]);
42              }
43
44              /* right edge */
45              for(i = 1 ; i < m ; i++)
46              {
47                  ret_val.push_back(vvi[rows_start+i][cols_start+n-1]);
48              }
49
50              /* bottom edge*/
51              for(i = cols_start+n-2 ; i >=cols_start ; i--)
52              {
53                  ret_val.push_back(vvi[rows_start+m-1][i]);
54              }
55
56              /* left edge*/
57              for(i = rows_start+m-2; i > rows_start;i--)
58              {
59                  ret_val.push_back(vvi[i][cols_start]);
60              }
61
62              m = m - 2;
63              n = n - 2;
64              __travelMatrix(vvi,rows_start+1);
65          }
66
67      public:
68          vector<int> spiralOrder(vector<vector<int>>& matrix)
69          {
70              ret_val.clear();
71              if(matrix.size() > 0)
72              {
73                  m = matrix.size();
```

```
                n = matrix[0].size();
                size = m*n;
                __travelMatrix(matrix,0);
            }
            return ret_val;
        }
};


/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了77.88% 的用户
内存消耗 :8.8 MB, 在所有 C++ 提交中击败了13.18%的用户
*/
```

方法二:C++_按圈遍历,分类迭代法

```
class Solution
{
    public:
        vector<int> spiralOrder(vector<vector<int>>& matrix)
        {
            vector<int>     ret_val             ;
            int             m           = 0     ;
            int             n           = 0     ;
            int             rows_start  = 0     ;
            int             cols_start  = 0     ;
            int             i           = 0     ;

            if(matrix.size() > 0)
            {
                m = matrix.size();
                n = matrix[0].size();

                while(    (m > 0)
                        &&(n > 0)
                )
                {
                    if(m==1)
                    {
                        for(i = 0; i < n ; i++)
                        {
                            ret_val.push_back(matrix[rows_start]
[cols_start+i]);
                        }
                        break;
                    }
                    if(n==1)
                    {
                        for(i = 0 ; i < m;i++)
                        {
                            ret_val.push_back(matrix[rows_start+i]
[cols_start]);
                        }
```

```
                        break;
                    }

                    /* up edge */
                    for(i = 0 ; i < n ; i++)
                    {
                        ret_val.push_back(matrix[rows_start]
[cols_start+i]);
                    }

                    /* right edge */
                    for(i = 1 ; i < m ; i++)
                    {
                        ret_val.push_back(matrix[rows_start+i]
[cols_start+n-1]);
                    }

                    /* bottom edge*/
                    for(i = cols_start+n-2 ; i >=cols_start ; i--)
                    {
                        ret_val.push_back(matrix[rows_start+m-1][i]);
                    }

                    /* left edge*/
                    for(i = rows_start+m-2; i > rows_start;i--)
                    {
                        ret_val.push_back(matrix[i][cols_start]);
                    }

                    m = m - 2;
                    n = n - 2;
                    rows_start++;
                    cols_start++;
                }
            }
            return ret_val;
        }
};


/*
执行结果:
通过
显示详情
执行用时 :0 ms, 在所有 C++ 提交中击败了100.00% 的用户
内存消耗 :8.6 MB, 在所有 C++ 提交中击败了52.60%的用户
*/
```

AlimyBreak
2019.10.02