

```

1  /*
2  给定一个正整数 n，生成一个包含 1 到 n2 所有元素，且元素按顺时针顺序螺旋排列的正方形矩阵。
3  示例：
4  输入：3
5  输出：
6  [
7    [ 1, 2, 3 ],
8    [ 8, 9, 4 ],
9    [ 7, 6, 5 ]
10 ]
11 来源：力扣（LeetCode）
12 链接：https://leetcode-cn.com/problems/spiral-matrix-ii
13 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。
14 */

```

分析:

- 根据题意，我们必须一圈一圈写，直到最后中心位置剩下1个空格未填或4个空格未填。
 - 方法一:尾递归法
 - 方法二:迭代法

方法一:C++_尾递归法

```

1  class Solution
2  {
3      void __fillN4Edge(vector<vector<int>>& vvi,int n,int rows_start,int
idx_start)
4      {
5          int cols_start = rows_start;
6          if(n==0)
7          {
8              return;
9          }
10         if(n==1)
11         {
12             vvi[rows_start][cols_start] = idx_start;
13             return ;
14         }
15
16         if(n==2)
17         {
18             vvi[rows_start][cols_start]           = idx_start++;
19             vvi[rows_start][cols_start+1]         = idx_start++;
20             vvi[rows_start+1][cols_start+1]       = idx_start++;
21             vvi[rows_start+1][cols_start]         = idx_start;
22
23             return ;
24         }
25
26         /* up edge */
27         for(int i = 0 ; i < n ; i++)
28         {
29             vvi[rows_start][cols_start+i] = idx_start++;

```

```

30     }
31
32     /* right edge */
33     for(int i = 1 ; i < n ; i++)
34     {
35         vvi[rows_start+i][cols_start+n-1] = idx_start++;
36     }
37
38     /* bottom edge*/
39     for(int i = cols_start+n-2 ; i >=cols_start ; i--)
40     {
41         vvi[rows_start+n-1][i] = idx_start++;
42     }
43
44     /* left edge*/
45     for(int i = rows_start+n-2; i > rows_start;i--)
46     {
47         vvi[i][cols_start] = idx_start++;
48     }
49
50     /* tail recursion*/
51     __fillN4Edge(vvi,n-2,rows_start+1,idx_start);
52 }
53
54 public:
55     vector<vector<int>> generateMatrix(int n)
56     {
57         vector<vector<int>> ret_val;
58
59         if(n>0)
60         {
61             vector<int> temp;
62             for(int i = 0; i < n ; i++)
63             {
64                 temp.push_back(0);
65             }
66             for(int i = 0; i < n ; i++)
67             {
68                 ret_val.push_back(temp);
69             }
70
71             __fillN4Edge(ret_val,n,0,1);
72         }
73         return ret_val;
74     }
75 };
76
77
78 /*
79 执行结果:
80 通过
81 显示详情
82 执行用时 :8 ms, 在所有 C++ 提交中击败了68.56% 的用户
83 内存消耗 :9.1 MB, 在所有 C++ 提交中击败了11.81%的用户
84 */

```

方法一:C++_迭代法

```
1  class Solution
2  {
3      public:
4          vector<vector<int>> generateMatrix(int n)
5          {
6              vector<vector<int>> ret_val;
7              int i = 0;
8
9              if(n>0)
10             {
11                 vector<int> temp;
12                 for(i = 0; i < n ; i++)
13                 {
14                     temp.push_back(0);
15                 }
16                 for(i = 0; i < n ; i++)
17                 {
18                     ret_val.push_back(temp);
19                 }
20
21                 //__fillN4Edge(ret_val,n,0,1);
22                 int rows_start = 0;
23                 int idx_start = 1;
24                 int cols_start = 0;
25                 while(n>0)
26                 {
27                     if(n==1)
28                     {
29                         ret_val[rows_start][cols_start] = idx_start;
30                         break;
31                     }
32                     if(n==2)
33                     {
34                         ret_val[rows_start][cols_start] =
35                         idx_start++;
36                         ret_val[rows_start][cols_start+1] =
37                         idx_start++;
38                         ret_val[rows_start+1][cols_start+1] =
39                         idx_start++;
40                         ret_val[rows_start+1][cols_start] =
41                         idx_start;
42                         break;
43                     }
44
45                     /* up edge */
46                     for(i = 0 ; i < n ; i++)
47                     {
48                         ret_val[rows_start][cols_start+i] = idx_start++;
49                     }
50
51                     /* right edge */
52                     for(i = 1 ; i < n ; i++)
53                     {
54                         ret_val[rows_start+i][cols_start+n-1] =
55                         idx_start++;
56                     }
57
58                     /* left edge */
59                     for(i = n-1 ; i > 0 ; i--)
60                     {
61                         ret_val[rows_start+i][cols_start] = idx_start++;
62                     }
63
64                     /* down edge */
65                     for(i = n-1 ; i > 0 ; i--)
66                     {
67                         ret_val[rows_start][cols_start+i] = idx_start++;
68                     }
69
70                     n -= 4;
71                 }
72             }
73
74             return ret_val;
75         }
76     }
```

```

51         }
52
53         /* bottom edge*/
54         for(i = cols_start+n-2 ; i >=cols_start ; i--)
55         {
56             ret_val[rows_start+n-1][i] = idx_start++;
57         }
58
59         /* left edge*/
60         for(i = rows_start+n-2; i > rows_start;i--)
61         {
62             ret_val[i][cols_start] = idx_start++;
63         }
64
65         n = n - 2;
66         rows_start++;
67         cols_start++;
68     }
69
70
71     }
72
73     return ret_val;
74 }
75 };
76
77
78 /*
79 执行结果:
80 通过
81 显示详情
82 执行用时 :4 ms, 在所有 C++ 提交中击败了94.85% 的用户
83 内存消耗 :8.9 MB, 在所有 C++ 提交中击败了51.11%的用户
84 */
85

```