

```

1  /*
2  包含整数的二维矩阵 M 表示一个图片的灰度。你需要设计一个平滑器来让每一个单元的灰度成为平均灰度（向下舍入），平均灰度的计算是周围的8个单元和它本身的值求平均，如果周围的单元格不足八个，则尽可能多的利用它们。
3  示例 1:
4  输入:
5  [[1,1,1],
6   [1,0,1],
7   [1,1,1]]
8  输出:
9  [[0, 0, 0],
10 [0, 0, 0],
11 [0, 0, 0]]
12 解释:
13 对于点 (0,0), (0,2), (2,0), (2,2): 平均(3/4) = 平均(0.75) = 0
14 对于点 (0,1), (1,0), (1,2), (2,1): 平均(5/6) = 平均(0.83333333) = 0
15 对于点 (1,1): 平均(8/9) = 平均(0.88888889) = 0
16 注意:
17     给定矩阵中的整数范围为 [0, 255]。
18     矩阵的长和宽的范围均为 [1, 150]。
19 来源: 力扣 (LeetCode)
20 链接: https://leetcode-cn.com/problems/image-smoother
21 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。
22 */

```

分析:

- 通过检测下标索引是否越界来计算总和和均值。

方法一:C++_下标检查

```

1  class Solution
2  {
3
4      private:
5          int __neighborAver(vector<vector<int>>& M , int rows , int cols ,
6  int row , int col)
7          {
8              int temp    =  M[row][col];
9              int count    =  1;
10
11              /*上一行*/
12              if(row-1>=0)
13              {
14                  temp += M[row-1][col];
15                  count++;
16                  if(col-1 >=0)
17                  {
18                      temp += M[row-1][col-1];
19                      count++;
20                  }
21
22                  if(col+1 < cols)

```

```

22         {
23             temp += M[row-1][col+1];
24             count++;
25         }
26     }
27
28     /*下一行*/
29     if(row+1< rows)
30     {
31         temp += M[row+1][col];
32         count++;
33         if(col-1 >=0)
34         {
35             temp += M[row+1][col-1];
36             count++;
37         }
38
39         if(col+1 < cols)
40         {
41             temp += M[row+1][col+1];
42             count++;
43         }
44     }
45
46     /*中间一行*/
47     if(col-1 >=0)
48     {
49         temp += M[row][col-1];
50         count++;
51     }
52
53     if(col+1 < cols)
54     {
55         temp += M[row][col+1];
56         count++;
57     }
58
59     return (int)(temp/count);
60 }
61
62 public:
63     vector<vector<int>> imageSmoother(vector<vector<int>>& M)
64     {
65         int r    = M.size();
66         int c    = M[0].size();
67
68         vector<vector<int>> vvi(r,vector<int>(c,0));
69
70         for(int i = 0; i < r; i++)
71         {
72             for(int j = 0 ; j < c; j++)
73             {
74                 vvi[i][j] = __neighborAver(M,r,c,i,j);
75             }
76         }
77         return vvi;
78     }
79 };

```

```
80
81  /*
82  执行结果:
83  通过
84  显示详情
85  执行用时 :168 ms, 在所有 cpp 提交中击败了97.11% 的用户
86  内存消耗 :17.4 MB, 在所有 cpp 提交中击败了88.18%的用户
87  */
```

AlimyBreak
2019.11.22