```
/*
根据[百度百科](https://baike.baidu.com/item/%E7%94%9F%E5%91%BD%E6%B8%B8%E6%88%8F/2926434?fr=aladdin)，生命游戏，简称为生命，是英国数学家约翰·何顿·康威在1970年发明的细胞自动机。

给定一个包含 m × n 个格子的面板，每一个格子都可以看成是一个细胞。每个细胞具有一个初始状态 live（1）即为活细胞， 或 dead（0）即为死细胞。每个细胞与其八个相邻位置（水平，垂直，对角线）的细胞都遵循以下四条生存定律：

    如果活细胞周围八个位置的活细胞数少于两个，则该位置活细胞死亡；
    如果活细胞周围八个位置有两个或三个活细胞，则该位置活细胞仍然存活；
    如果活细胞周围八个位置有超过三个活细胞，则该位置活细胞死亡；
    如果死细胞周围正好有三个活细胞，则该位置死细胞复活；

根据当前状态，写一个函数来计算面板上细胞的下一个（一次更新后的）状态。下一个状态是通过将上述规则同时应用于当前状态下的每个细胞所形成的，其中细胞的出生和死亡是同时发生的。

示例：

输入：
[
  [0,1,0],
  [0,0,1],
  [1,1,1],
  [0,0,0]
]
输出：
[
  [0,0,0],
  [1,0,1],
  [0,1,1],
  [0,1,0]
]

进阶：

    你可以使用原地算法解决本题吗？请注意，面板上所有格子需要同时被更新：你不能先更新某些格子，然后使用它们的更新后的值再更新其他格子。
    本题中，我们使用二维数组来表示面板。原则上，面板是无限的，但当活细胞侵占了面板边界时会造成问题。你将如何解决这些问题？

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/game-of-life
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:
方法一:非原地解法,统计细胞周围8个点的活细胞数目，然后再根据本细胞存活情况进行分情况处理

```
class Solution
{
    private:
```

```
        int __calLiveCells(vector<vector<int>>& vvi , int row , int col)
        {
            int ret_val = 0;

            /*左上*/
            if(row > 0 && col > 0)
            {
                ret_val += vvi[row-1][col-1];
            }

            /*左中*/
            if( col > 0)
            {
                ret_val += vvi[row][col-1];
            }
            /*左下*/
            if(row+1 < vvi.size() && col > 0)
            {
                ret_val += vvi[row+1][col-1];
            }

            /*中上*/
            if(row > 0)
            {
                ret_val += vvi[row-1][col];
            }

            /*中下*/
            if(row+1 < vvi.size())
            {
                ret_val += vvi[row+1][col];
            }


            /*右上*/
            if(row>0 && col+1 < vvi[0].size())
            {
                ret_val += vvi[row-1][col+1];
            }

            /*右中*/
            if(col+1 < vvi[0].size())
            {
                ret_val += vvi[row][col+1];
            }

            /*右下*/
            if(row+1 < vvi.size() && col+1 < vvi[0].size())
            {
                ret_val += vvi[row+1][col+1];
            }
            return ret_val;
        }
    public:
        void gameOfLife(vector<vector<int>>& board)
        {
            vector<vector<int>> bak(board);

```

```
62              int rows = bak.size();
63              int cols = bak[0].size();
64
65              for(int i = 0 ; i < rows ; i++)
66              {
67                  for(int j = 0 ; j < cols ; j++)
68                  {
69                      int liveCount = __calLiveCells(bak,i,j);
70                      switch((bak[i][j] << 4) + liveCount)
71                      {
72                          case 0x00:
73                          case 0x01:
74                          case 0x02:
75                          case 0x04:
76                          case 0x05:
77                          case 0x06:
78                          case 0x07:
79                          case 0x08:
80                              board[i][j] = 0;
81                          break;
82                          case 0x03:
83                              board[i][j] = 1;
84                          break;
85
86                          /* 活细胞 */
87                          case 0x10:
88                          case 0x11:
89                              /*如果活细胞周围八个位置的活细胞数少于两个，则该位置活
    细胞死亡；*/
90                              board[i][j] = 0;
91                          break;
92                          case 0x12:
93                          case 0x13:
94                              /*如果活细胞周围八个位置有两个或三个活细胞，则该位置活
    细胞仍然存活；*/
95                              board[i][j] = 1;
96                          break;
97                          case 0x14:
98                          case 0x15:
99                          case 0x16:
100                         case 0x17:
101                         case 0x18:
102                             /*如果活细胞周围八个位置有超过三个活细胞，则该位置活细
    胞死亡；*/
103                             board[i][j] = 0;
104                         break;
105                         default: break;
106                     }
107                 }
108             }
109         }
110 };
111
112
113 /*
114 执行结果：
115 通过
116 显示详情
```

```
执行用时 :4 ms, 在所有 cpp 提交中击败了82.40% 的用户
内存消耗 :8.8 MB, 在所有 cpp 提交中击败了19.85%的用户
*/
```

AlimyBreak
2019.10.31