```
/*
给定一个二维网格和一个单词，找出该单词是否存在于网格中。

单词必须按照字母顺序，通过相邻的单元格内的字母构成，其中“相邻”单元格是那些水平相邻或垂直相邻的单元格。同一个单元格内的字母不允许被重复使用。

示例:

board =
[
  ['A','B','C','E'],
  ['S','F','C','S'],
  ['A','D','E','E']
]

给定 word = "ABCCED", 返回 true.
给定 word = "SEE", 返回 true.
给定 word = "ABCB", 返回 false.

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/word-search
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:

- 回溯法,在主函数中,遍历每一个元素开始进行一个一个字符的对比,在$help$函数中进行四个方向的探索,若有一条路满足就返回$true$.

---

方法一:C++_回溯法

```
class Solution
{

    private:

        int arr[8] = {-1,0,1,0,0,-1,0,+1};
        bool helper(    vector<vector<char>>&   board       ,
                        string&                 word        ,
                        int                     cur_idx     ,
                        int                     cur_row     ,
                        int                     cur_col     ,
                        vector<vector<int>>&    visited
                    )
        {

            if(cur_idx == word.size())
            {
                return true;
            }
```

```
20
21              if(cur_row<0 || cur_row >= board.size() || cur_col < 0 ||
    cur_col >= board[0].size())
22              {
23                  return false;
24              }
25
26              if(visited[cur_row][cur_col]==1)
27              {
28                  return false;
29              }
30
31              if(board[cur_row][cur_col]!=word[cur_idx])
32              {
33                  return false;
34              }
35              else
36              {
37                  visited[cur_row][cur_col] = 1;
38                  for(int i = 0 ; i < 4 ; i++)
39                  {
40
     if(helper(board,word,cur_idx+1,cur_row+arr[2*i],cur_col+arr[2*i+1],visited
    ))
41                          return true;
42
43                  }
44                  visited[cur_row][cur_col] = 0;
45              }
46              return false;
47          }
48
49
50      public:
51          bool exist( vector<vector<char>>&   board   ,
52                      string                  word
53                    )
54          {
55              int r = board.size();
56              int c = board[0].size();
57              int i = 0;
58              int j = 0;
59              vector<vector<int>>    visited = vector<vector<int>>
    (r,vector<int>(c,0));
60
61              for( i = 0 ; i < r;i++)
62              {
63                  for( j = 0 ; j < c;j++)
64                  {
65
66                      if(helper(board,word,0,i,j,visited))
67                      {
68                          return true;
69                      }
70                  }
71              }
72
73              return false;
```

```
        }
};


/*
执行结果:
通过
显示详情
执行用时 :24 ms, 在所有 cpp 提交中击败了94.24% 的用户
内存消耗 :10.5 MB, 在所有 cpp 提交中击败了90.51%的用户
*/
```

---

AlimyBreak
2019.12.04