

```

1  /*
2  给定一个单词列表，只返回可以使用在键盘同一行的字母打印出来的单词。键盘如下图所示。
3  American keyboard
4
5  示例：
6  输入：["Hello", "Alaska", "Dad", "Peace"]
7  输出：["Alaska", "Dad"]
8
9  注意：
10
11     你可以重复使用键盘上同一字符。
12     你可以假设输入的字符串将只包含字母。
13
14  来源：力扣（LeetCode）
15  链接：https://leetcode-cn.com/problems/keyboard-row
16  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。
17  */

```

分析:

- 常规方法
  - 建立三个map存放三行字母的映射值;
  - 遍历每组字符串统计行属性情况;
  - 根据行属性统计情况决定是否要把原每组字符压入返回值.

方法一:C++\_Map

```

1  class Solution
2  {
3      private:
4          map<char,int> mci1;
5          map<char,int> mci2;
6          map<char,int> mci3;
7
8          bool issameLine(string& s)
9          {
10             int    temp    = 0        ;
11             int    i        = 0        ;
12             bool    ret_val = true    ;
13
14             for(i = 0; i < s.size(); i ++ )
15             {
16                 if(mci1.count(s.at(i)))
17                 {
18                     temp |= 0x01;
19                 }
20                 else if(mci2.count(s.at(i)))
21                 {
22                     temp |= 0x02;
23                 }
24                 else

```

```

25         {
26             temp |= 0x04;
27         }
28     }
29
30     switch(temp)
31     {
32         case 0x01:
33         case 0x02:
34         case 0x04:
35             return true;
36         break;
37         default:
38             return false;
39         break;
40
41     }
42 }
43
44
45 public:
46     vector<string> findWords(vector<string>& words)
47     {
48         vector<string> ret_val ;
49         int i = 0 ;
50
51         for(i=0;i<words.size();i++)
52         {
53             if(isSameLine(words[i]))
54             {
55                 ret_val.push_back(words[i]);
56             }
57         }
58         return ret_val;
59     }
60
61     solution()
62     {
63         /* qwertyuiop */
64         mci1['q'] = 0;
65         mci1['w'] = 1;
66         mci1['e'] = 2;
67         mci1['r'] = 3;
68         mci1['t'] = 4;
69         mci1['y'] = 5;
70         mci1['u'] = 6;
71         mci1['i'] = 7;
72         mci1['o'] = 8;
73         mci1['p'] = 9;
74
75         mci1['Q'] = 10;
76         mci1['W'] = 11;
77         mci1['E'] = 12;
78         mci1['R'] = 13;
79         mci1['T'] = 14;
80         mci1['Y'] = 15;
81         mci1['U'] = 16;
82         mci1['I'] = 17;

```

```

83         mci1['o'] = 18;
84         mci1['p'] = 19;
85
86         /* asdfghjkl */
87         mci2['a'] = 20;
88         mci2['s'] = 21;
89         mci2['d'] = 22;
90         mci2['f'] = 23;
91         mci2['g'] = 24;
92         mci2['h'] = 25;
93         mci2['j'] = 26;
94         mci2['k'] = 27;
95         mci2['l'] = 28;
96         mci2['A'] = 30;
97         mci2['S'] = 31;
98         mci2['D'] = 32;
99         mci2['F'] = 33;
100        mci2['G'] = 34;
101        mci2['H'] = 35;
102        mci2['J'] = 36;
103        mci2['K'] = 37;
104        mci2['L'] = 38;
105
106        /* zxcvbnm */
107        mci3['z'] = 40;
108        mci3['x'] = 41;
109        mci3['c'] = 42;
110        mci3['v'] = 43;
111        mci3['b'] = 44;
112        mci3['n'] = 45;
113        mci3['m'] = 46;
114
115        mci3['Z'] = 50;
116        mci3['X'] = 51;
117        mci3['C'] = 52;
118        mci3['V'] = 53;
119        mci3['B'] = 54;
120        mci3['N'] = 55;
121        mci3['M'] = 56;
122    }
123 };
124
125 /*
126 执行结果:
127 通过
128 显示详情
129 执行用时 :4 ms, 在所有 C++ 提交中击败了74.02% 的用户
130 内存消耗 :8.8 MB, 在所有 C++ 提交中击败了5.80%的用户
131 */

```