```
/*
在二维平面上，有一个机器人从原点 (0, 0) 开始。给出它的移动顺序，判断这个机器人在完成移动后是否在 (0, 0) 处结束。

移动顺序由字符串表示。字符 move[i] 表示其第 i 次移动。机器人的有效动作有 R（右），L（左），U（上）和 D（下）。如果机器人在完成所有动作后返回原点，则返回 true。否则，返回 false。

注意：机器人“面朝”的方向无关紧要。 “R” 将始终使机器人向右移动一次，“L” 将始终向左移动等。此外，假设每次移动机器人的移动幅度相同。




示例 1:

输入: "UD"
输出: true
解释：机器人向上移动一次，然后向下移动一次。所有动作都具有相同的幅度，因此它最终回到它开始的原点。因此，我们返回 true。

示例 2:

输入: "LL"
输出: false
解释：机器人向左移动两次。它最终位于原点的左侧，距原点有两次 “移动” 的距离。我们返回 false，因为它在移动结束时没有返回原点。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/robot-return-to-origin
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:

- UD 为一组, LR为一组,每组的两种移动次数相等即可返回true,否则返回false.

方法一:C_分别统计步数.

```
bool judgeCircle(char * moves)
{
    int  i          = 0;
    int count_lr    = 0;
    int count_ud    = 0;
    while (moves[i])
    {
        switch (moves[i])
        {
            case 'R':
                count_lr++;
            break;

```

```
            case 'L':
                count_lr--;
            break;

            case 'U':
                count_ud++;
            break;

            case 'D':
                count_ud--;
            break;
            default:
            break;
        }
        i++;
    }
    return ((count_lr==0)&&(count_ud == 0));
}
/*
执行结果:
通过
显示详情
执行用时 :8 ms, 在所有 C 提交中击败了90.37% 的用户
内存消耗 :7.2 MB, 在所有 C 提交中击败了74.34%的用户
*/
```

AlimyBreak
2019.09.04