```
/*
将一个给定字符串根据给定的行数，以从上往下、从左到右进行 Z 字形排列。

比如输入字符串为 "LEETCODEISHIRING" 行数为 3 时，排列如下：

L   C   I   R
E T O E S I I G
E   D   H   N

之后，你的输出需要从左往右逐行读取，产生出一个新的字符串，比如："LCIRETOESIIGEDHN"。

请你实现这个将字符串进行指定行数变换的函数：

string convert(string s, int numRows);

示例 1:

输入: s = "LEETCODEISHIRING", numRows = 3
输出: "LCIRETOESIIGEDHN"

示例 2:

输入: s = "LEETCODEISHIRING", numRows = 4
输出: "LDREOEIIECIHNTSG"
解释:

L     D     R
E   O E   I I
E C   I H   N
T     S     G

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/zigzag-conversion
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:

- 自己画几个找规律，我找到的规律是每行下标与起始点的关系.

方法一:C++_找规律

```
class Solution
{
    public:
        string convert(string s, int numRows)
        {
            if(numRows<=1)
            {
                return s;
            }

```

```
            string  ret_val(s)      ;
            int     ret_num     = 0 ;
            int     i           = 0 ;
            int     count       = 0 ;/*0-偶数次 1-奇数次*/
            int     temp        = 0 ;
            // 第0行
            temp = 0;
            while(temp < s.size())
            {
                ret_val[ret_num++] = s[temp];
                temp += 2*(numRows-1);
            }
            // 第1--> numRows-2 行
            for(i = 1 ; i < numRows-1;i++)
            {
                temp    =   i   ;
                count   =   0   ;
                while(temp < s.size())
                {
                    ret_val[ret_num++] = s[temp];
                    if(count==0)
                    {
                        temp += 2*(numRows-i-1);
                        count = 1;
                    }
                    else
                    {
                        temp += 2*i;
                        count = 0;
                    }
                }
            }
            // 第numRows-1行
            temp = numRows-1;
            while(temp < s.size())
            {
                ret_val[ret_num++] = s[temp];
                temp += 2*(numRows-1);
            }

            return ret_val;

        }
};

/*
执行结果:
通过
显示详情
执行用时 :20 ms, 在所有 cpp 提交中击败了51.28% 的用户
内存消耗 :10 MB, 在所有 cpp 提交中击败了97.22%的用户
*/
```

方法二:C++_找规律(代码优化)

```
class Solution
```

```
{
    public:
        string convert(string s, int numRows)
        {
            if(numRows<=1)
            {
                return s;
            }

            string  ret_val(s)                  ;
            int     ret_num                 = 0 ;
            int     i                       = 0 ;
            int     count                   = 0 ;/*0-偶数次 1-奇数次,2-顶层或者底层*/
            int     temp                    = 0 ;


            for(i = 0 ; i < numRows;i++)
            {
                temp  = i;
                count = 0;
                if(i == 0 || i == numRows-1)
                {
                    count = 2;
                }
                while(temp < s.size())
                {
                    ret_val[ret_num++] = s[temp];
                    if(count==0)
                    {
                        temp += 2*(numRows-i-1);
                        count = 1;
                    }
                    else if(count==1)
                    {
                        temp += 2*i;
                        count = 0;
                    }
                    else
                    {
                        temp += 2*(numRows-1);
                    }
                }
            }
            return ret_val;

        }
};

/*
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 cpp 提交中击败了70.03% 的用户
内存消耗 :9.8 MB, 在所有 cpp 提交中击败了98.99%的用户
*/

```

AlimyBreak
2019.12.10