```
                while (curC != dC) {
                        System.out.print(m[tR][curC] + " ");
                        curC++;
                }
                while (curR != dR) {
                        System.out.print(m[curR][dC] + " ");
                        curR++;
                }
                while (curC != tC) {
                        System.out.print(m[dR][curC] + " ");
                        curC--;
                }
                while (curR != tR) {
                        System.out.print(m[curR][tC] + " ");
                        curR--;
                }
        }
}
```

将正方形矩阵顺时针转动 90°

【题目】

给定一个 $N \times N$ 的矩阵 matrix，把这个矩阵调整成顺时针转动 90° 后的形式。

例如：

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

顺时针转动 90° 后为：

13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

【要求】

额外空间复杂度为 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

这里仍使用分圈处理的方式，在矩阵中用左上角的坐标(tR,tC)和右下角的坐标(dR,dC)就可以表示一个子矩阵。比如，题目中的矩阵，当(tR,tC)=(0,0)、(dR,dC)=(3,3)时，表示的子矩阵就是整个矩阵，那么这个子矩阵最外层的部分如下。

```
1    2    3    4
5              8
9              12
13   14   15   16
```

在这个外圈中，1，4，16，13 为一组，然后让 1 占据 4 的位置，4 占据 16 的位置，16 占据 13 的位置，13 占据 1 的位置，一组就调整完了。然后 2，8，15，9 为一组，继续占据调整的过程，最后 3，12，14，5 为一组，继续占据调整的过程。然后(tR,tC)=(0,0)、(dR,dC)=(3,3)的子矩阵外层就调整完毕。接下来令 tR 和 tC 加 1，即(tR,tC)=(1,1)，令 dR 和 dC 减 1，即(dR,dC)=(2,2)，此时表示的子矩阵如下。

```
6    7
10   11
```

这个外层只有一组，就是 6，7，11，10，占据调整之后即可。所以，如果子矩阵的大小是 $M \times M$，一共就有 M-1 组，分别进行占据调整即可。

具体过程请参看如下代码中的 rotate 方法。

```
public void rotate(int[][] matrix) {
        int tR = 0;
        int tC = 0;
        int dR = matrix.length - 1;
        int dC = matrix[0].length - 1;
        while (tR < dR) {
                rotateEdge(matrix, tR++, tC++, dR--, dC--);
        }
}

public void rotateEdge(int[][] m, int tR, int tC, int dR, int dC) {
        int times = dC - tC; // times 就是总的组数
        int tmp = 0;
        for (int i = 0; i != times; i++) { // 一次循环就是一组占据调整
                tmp = m[tR][tC + i];
```

```
                m[tR][tC + i] = m[dR - i][tC];
                m[dR - i][tC] = m[dR][dC - i];
                m[dR][dC - i] = m[tR + i][dC];
                m[tR + i][dC] = tmp;
            }
        }
```

“之”字形打印矩阵

【题目】

给定一个矩阵 matrix，按照“之”字形的方式打印这个矩阵，例如：

1　　2　　3　　4

5　　6　　7　　8

9　　10　　11　　12

“之”字形打印的结果为：1，2，5，9，6，3，4，7，10，11，8，12

【要求】

额外空间复杂度为 $O(1)$。

【难度】

士 ★☆☆☆

【解答】

本书提供的实现方法是这样处理的：

1．上坐标(tR,tC)初始为(0,0)，先沿着矩阵第一行移动(tC++)，当到达第一行最右边的元素后，再沿着矩阵最后一列移动(tR++)。

2．下坐标(dR,dC)初始为(0,0)，先沿着矩阵第一列移动(dR++)，当到达第一列最下边的元素时，再沿着矩阵最后一行移动(dC++)。

3．上坐标与下坐标同步移动，每次移动后的上坐标与下坐标的连线就是矩阵中的一条斜线，打印斜线上的元素即可。

4．如果上次斜线是从左下向右上打印的，这次一定是从右上向左下打印，反之亦然。总之，可以把打印的方向用 boolean 值表示，每次取反即可。

具体请参看如下代码中的 printMatrixZigZag 方法。