格子填数【NOIP 普及组 1995 年第 2 题】

【问题描述】

在$n \times n$（$n \leq 8$）方阵里填入 1，2，3，…，n×n，要求填成蛇形。

例如 n=4 时方阵为：

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

【输入样例】

```
4
```

【输出样例】

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

【分析】

可以用二维数组来储存题目中的方阵。

从 1 开始依次填写。设“笔”的坐标为(row,col)，则一开始 row=0，col=n-1，即第 0 行，第 n-1 列。

“笔”的移动轨迹是：下，下，下，左，左，左，上，上，上，右，右，下，下，左，上。总之，先是下，到不能填了为止，然后是左，接着是上，最后是右。

“不能填”是指再走就出界（例如 4→5），或者再走就要走到以前填过的格子（例如 12→13）。如果把所有格子初始为 0，就能很方便地判断是否出界。

【参考程序】

```
1. #include<iostream>
```

```
using namespace std;
const int N = 8;
int main()
{
    int a[N][N] = {};
    int n; //阵列大小
    cin >> n;
    int row = 0; //当前行
    int col = n - 1; //当前列
    int current = 1; //当前待填充数字
    a[row][col] = current;
    while (current < n*n)
    {
        //向下走
        while (row+1<n && !a[row+1][col])
        {
            current++;
            row++;
            a[row][col] = current;
        }
        //向左走
        while (col-1>=0 && !a[row][col-1])
        {
            current++;
            col--;
            a[row][col] = current;
        }
        //向上走
        while (row-1>=0 && !a[row-1][col])
        {
            current++;
            row--;
            a[row][col] = current;
        }
        //向右走
        while (col+1<n && !a[row][col+1])
        {
            current++;
            col++;
            a[row][col] = current;
        }
    }
    for(row=0; row<n; row++)
    {
```

```
46.         for (col=0; col<n; col++)
47.         {
48.             cout << a[row][col] << '\t';
49.         }
50.         cout << endl;
51.     }
52.     return 0;
53. }
```

【说明】

外层循环内嵌的 4 条 while 内层循环十分相似，因此这里只介绍第一条：不断向下走，并且填数。

填数的原则是：先判断下一步是否越界，如果不越界就移动并进行填数。

越界只需判断 row+1<n，因为 col 值并没有修改。

下一个格子是 (row+1, col)，因此检查下一个格子是否填写，只需判断是否 a[row+1][col]==0，简写成!a[row+1][col]。

这里不用担心当 row+1==n 时，a[row+1][col]会越界。因为如果 row+1<n 为假，将不会计算!a[row+1][col]，也就不会越界了。