

一、编程题

ACM：螺旋数字矩阵

题目描述：疫情期间，小明隔离在家，百无聊赖，在纸上写数字玩。他发明了一种写法：

给出数字个数 n 和行数 m ($0 < n \leq 999$, $0 < m \leq 999$)，从左上角的1开始，按照顺时针螺旋向内写方式，依次写出 $2, 3, \dots, n$ ，最终形成一个 m 行矩阵。

小明对这个矩阵有些要求：

1. 每行数字的个数一样多
2. 列的数量尽可能少
3. 填充数字时优先填充外部
4. 数字不够时，使用单个*号占位

输入描述：两个整数，空格隔开，依次表示 n 、 m

输出描述：符合要求的唯一矩阵

补充说明：

示例1

输入：9 4

输出：1 2 3

* * 4

9 * 5

8 7 6

说明：9个数字写成4行，最少需要3列

示例2

输入：3 5

输出：1

2

3

*

*

说明：3个数字写5行，只有一列，数字不够用*号填充

示例3

输入：120 7

输出：1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 19

45 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 63 20

44 83 114 115 116 117 118 119 120 * * * * * 99 64 21

43 82 113 112 111 110 109 108 107 106 105 104 103 102 101 100 65 22

42 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 23

41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24

代码：

```
import java.util.Scanner;
```

// 注意类名必须为 Main, 不要有任何 package xxx 信息

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int m = sc.nextInt();  
        int n = sc.nextInt();  
        int j = m / n;  
        if (m % n != 0) {
```

```

        j++;
    }
    int[][] arr = new int[n][j];
    for (int i = 0; i < arr.length; i++) {
        for (int k = 0; k < arr[i].length; k++) {
            arr[i][k] = 0;
        }
    }
    int i = 1;
    Temp temp = new Temp(0, -1, 1);
    while (i <= m) {
        temp = setY(arr, true, i, temp.i, temp.j + 1, m);
        i = temp.n;
        if (i <= m) {
            temp = setX(arr, true, i, temp.i + 1, temp.j, m);
            i = temp.n;
        }
        if (i <= m) {
            temp = setY(arr, false, i, temp.i, temp.j - 1, m);
            i = temp.n;
        }
        if (i <= m) {
            temp = setX(arr, false, i, temp.i - 1, temp.j, m);
            i = temp.n;
        }
    }
    for (int u = 0; u < arr.length; u++) {
        for (int y = 0; y < arr[u].length; y++) {
            boolean flag = false;
            if (y == arr[u].length - 1) {
                flag = true;
            }
            if (!flag) {
                if (arr[u][y] == 0) {
                    System.out.print("*" + " ");
                } else {
                    System.out.print(arr[u][y] + " ");
                }
            } else {
                if (arr[u][y] == 0) {
                    System.out.println("*");
                } else {
                    System.out.println(arr[u][y] + " ");
                }
            }
        }
    }

```

```

        }
    }
}

```

```

public static Temp setY(int[][] arr, boolean flag, int n, int i, int j, int max) {
    if (flag) {
        while (j < arr[i].length && arr[i][j] == 0 && n <= max) {
            arr[i][j] = n;
            n++;
            j++;
        }
        j--;
    } else {
        while (j >= 0 && arr[i][j] == 0 && n <= max) {
            arr[i][j] = n;
            n++;
            j--;
        }
        j++;
    }
    Temp temp = new Temp(i, j, n);
    return temp;
}

```

```

public static Temp setX(int[][] arr, boolean flag, int n, int i, int j, int max) {
    if (flag) {
        while (i < arr.length && arr[i][j] == 0 && n <= max) {
            arr[i][j] = n;
            n++;
            i++;
        }
        i--;
    } else {
        while (i >= 0 && arr[i][j] == 0 && n <= max) {
            arr[i][j] = n;
            n++;
            i--;
        }
        i++;
    }
    Temp temp = new Temp(i, j, n);
    return temp;
}

```

```
}  
class Temp{  
    int i = 0;  
    int j = 0;  
    int n = 0;  
  
    public Temp(int i, int j, int n) {  
        this.i = i;  
        this.j = j;  
        this.n = n;  
    }  
}
```