

【华为OD机考 统一考试机试C卷】螺旋数字矩阵（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，**C卷真题已基本整理完毕**

抽到原题的概率为2/3到3/3，**也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。**

另外订阅专栏还可以联系笔者开通在线 **OJ** 进行刷题，提高刷题效率。

真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明

专栏：2023华为OD机试(B卷+C卷+D卷) (C++JavaJSPy)

华为OD面试真题精选：华为OD面试真题精选

在线OJ：点击立即刷题，模拟真实机考环境

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：

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

输入描述

输入一行，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

用例1

输入：

1 | 9 4

输出：

1 | 1 2 3
2 | * * 4
3 | 9 * 5
4 | 8 7 6

说明：

9个数字写成4行，最少需要3列

用例2

输入：

3 5

输出：

1 | 1
2 | 2
3 | 3
4 | *
5 | *

说明：

3个数字写5行，只有一列，数字不够用*号填充

用例3

输入:

```
1 | 120 7
```

输出:

```
1 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
2 | 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 19
3 | 45 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 63 20
4 | 44 83 114 115 116 117 118 119 120 * * * * * 99 64 21
5 | 43 82 113 112 111 110 109 108 107 106 105 104 103 102 101 100 65 22
6 | 42 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 23
7 | 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24
```

解题思路

本题不难，主要就是模拟。按照题目的要求填充矩阵就可以了。

C++

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <cmath>
4 |
5 | int main() {
6 |     int n, m;
7 |     std::cin >> n >> m; // 读取要填充的数字个数n和矩阵的行数m
8 |
9 |     int cols = std::ceil(static_cast<double>(n) / m); // 计算矩阵的列数
10 |    std::vector<std::vector<int>> matrix(m, std::vector<int>(cols, 0)); // 创建一个整型矩阵，默认初始化为0
11 |
12 |    int num = 1; // 用于填充的数字从1开始
13 |    int top = 0, bottom = m - 1, left = 0, right = cols - 1;
14 |    while (num <= n) {
15 |        for (int i = left; i <= right && num <= n; i++) { // 从左到右填充上边界
16 |            matrix[top][i] = num++;
17 |        }
18 |    }
```

```

19     top++; // 上边界下移
20     for (int i = top; i <= bottom && num <= n; i++) { // 从上到下填充右边界
21         matrix[i][right] = num++;
22     }
23     right--; // 右边界左移
24     for (int i = right; i >= left && num <= n; i--) { // 从右到左填充下边界
25         matrix[bottom][i] = num++;
26     }
27     bottom--; // 下边界上移
28     for (int i = bottom; i >= top && num <= n; i--) { // 从下到上填充左边界
29         matrix[i][left] = num++;
30     }
31     left++; // 左边界右移
32 }
33
34 for (int i = 0; i < m; i++) { // 遍历矩阵的每一行
35     for (int j = 0; j < cols; j++) { // 遍历矩阵的每一列
36         if (matrix[i][j] == 0) { // 如果当前位置是0, 则输出 '*'
37             std::cout << '*';
38         } else { // 否则输出当前位置的数字
39             std::cout << matrix[i][j];
40         }
41         if (j < cols - 1) { // 在同一行的数字之间打印空格
42             std::cout << " ";
43         }
44     }
45     std::cout << std::endl; // 每打印完一行后换行
46 }
47
48 return 0;
}

```

Java

```

1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int n = scanner.nextInt(); // 读取要填充的数字个数n
7     }
8 }

```

```

/
8   int m = scanner.nextInt(); // 读取矩阵的行数m
9   scanner.close(); // 输入完毕后关闭scanner

10  int cols = (int) Math.ceil(n / (double) m); // 计算矩阵的列数
11  int[][] matrix = new int[m][cols]; // 创建一个整型矩阵, 默认初始化为0
12
13  int num = 1; // 用于填充的数字从1开始
14  int top = 0, bottom = m - 1, left = 0, right = cols - 1;
15  while (num <= n) {
16      for (int i = left; i <= right && num <= n; i++) { // 从左到右填充上边界
17          matrix[top][i] = num++;
18      }
19      top++; // 上边界下移
20      for (int i = top; i <= bottom && num <= n; i++) { // 从上到下填充右边界
21          matrix[i][right] = num++;
22      }
23      right--; // 右边界左移
24      for (int i = right; i >= left && num <= n; i--) { // 从右到左填充下边界
25          matrix[bottom][i] = num++;
26      }
27      bottom--; // 下边界上移
28      for (int i = bottom; i >= top && num <= n; i--) { // 从下到上填充左边界
29          matrix[i][left] = num++;
30      }
31      left++; // 左边界右移
32  }
33
34  for (int i = 0; i < m; i++) { // 遍历矩阵的每一行
35      for (int j = 0; j < cols; j++) { // 遍历矩阵的每一列
36          if (matrix[i][j] == 0) { // 如果当前位置是0, 则输出 '*'
37              System.out.print('*');
38          } else { // 否则输出当前位置的数字
39              System.out.print(matrix[i][j]);
40          }
41          if (j < cols - 1) { // 在同一行的数字之间打印空格
42              System.out.print(" ");
43          }
44      }
45      System.out.println(); // 每打印完一行后换行
46  }
47

```

```
48 |     }  
    }
```

JavaScript

```
1  const readline = require('readline').createInterface({  
2    input: process.stdin,  
3    output: process.stdout  
4  });  
5  
6  readline.on('line', (input) => {  
7    const [n, m] = input.split(' ').map(Number); // 读取要填充的数字个数n和矩阵的行数m  
8    const cols = Math.ceil(n / m); // 计算矩阵的列数  
9    const matrix = Array.from({ length: m }, () => Array(cols).fill(0)); // 创建一个整型矩阵, 默认初始化为0  
10  
11    let num = 1; // 用于填充的数字从1开始  
12    let top = 0, bottom = m - 1, left = 0, right = cols - 1;  
13    while (num <= n) {  
14      for (let i = left; i <= right && num <= n; i++) { // 从左到右填充上边界  
15        matrix[top][i] = num++;  
16      }  
17      top++; // 上边界下移  
18      for (let i = top; i <= bottom && num <= n; i++) { // 从上到下填充右边界  
19        matrix[i][right] = num++;  
20      }  
21      right--; // 右边界左移  
22      for (let i = right; i >= left && num <= n; i--) { // 从右到左填充下边界  
23        matrix[bottom][i] = num++;  
24      }  
25      bottom--; // 下边界上移  
26      for (let i = bottom; i >= top && num <= n; i--) { // 从下到上填充左边界  
27        matrix[i][left] = num++;  
28      }  
29      left++; // 左边界右移  
30    }  
31  
32    for (let i = 0; i < m; i++) { // 遍历矩阵的每一行  
33      let row = '';  
34      for (let j = 0; j < cols; j++) { // 遍历矩阵的每一列  
35        row += matrix[i][j] === 0 ? '*' : matrix[i][j]; // 如果当前位置是0, 则输出 '*', 否则输出当前位置的数字  
36      }  
37      console.log(row);  
38    }  
39  });
```

```

36         if (j < cols - 1) row += ' '; // 在同一行的数字之间添加空格
37     }
38     console.log(row); // 打印每一行
39 }
40
41 readline.close();
42 });

```

Python

```

1  import math
2
3  n, m = map(int, input().split()) # 读取要填充的数字个数n和矩阵的行数m
4  cols = math.ceil(n / m) # 计算矩阵的列数
5  matrix = [[0 for _ in range(cols)] for _ in range(m)] # 创建一个整型矩阵, 默认初始化为0
6
7  num = 1 # 用于填充的数字从1开始
8  top, bottom, left, right = 0, m - 1, 0, cols - 1
9  while num <= n:
10     for i in range(left, right + 1): # 从左到右填充上边界
11         if num <= n:
12             matrix[top][i] = num
13             num += 1
14     top += 1 # 上边界下移
15     for i in range(top, bottom + 1): # 从上到下填充右边界
16         if num <= n:
17             matrix[i][right] = num
18             num += 1
19     right -= 1 # 右边界左移
20     for i in range(right, left - 1, -1): # 从右到左填充下边界
21         if num <= n:
22             matrix[bottom][i] = num
23             num += 1
24     bottom -= 1 # 下边界上移
25     for i in range(bottom, top - 1, -1): # 从下到上填充左边界
26         if num <= n:
27             matrix[i][left] = num
28             num += 1
29     left += 1 # 左边界右移
30
31

```

```

31 for row in matrix: # 遍历矩阵的每一行
32     print(' '.join('*' if val == 0 else str(val) for val in row)) # 如果当前位置是0, 则输出 '*', 否则输出当前位置的数字

```

C语言

```

1  #include <stdio.h>
2  #include <math.h>
3
4  int main() {
5      int n, m;
6      scanf("%d %d", &n, &m); // 读取要填充的数字个数n和矩阵的行数m
7
8      int cols = (int)ceil((double)n / m); // 计算矩阵的列数
9      int matrix[m][cols]; // 创建一个整型矩阵, 默认初始化为0
10     for (int i = 0; i < m; i++) {
11         for (int j = 0; j < cols; j++) {
12             matrix[i][j] = 0;
13         }
14     }
15
16     int num = 1; // 用于填充的数字从1开始
17     int top = 0, bottom = m - 1, left = 0, right = cols - 1;
18     while (num <= n) {
19         for (int i = left; i <= right && num <= n; i++) { // 从左到右填充上边界
20             matrix[top][i] = num++;
21         }
22         top++; // 上边界下移
23         for (int i = top; i <= bottom && num <= n; i++) { // 从上到下填充右边界
24             matrix[i][right] = num++;
25         }
26         right--; // 右边界左移
27         for (int i = right; i >= left && num <= n; i--) { // 从右到左填充下边界
28             matrix[bottom][i] = num++;
29         }
30         bottom--; // 下边界上移
31         for (int i = bottom; i >= top && num <= n; i--) { // 从下到上填充左边界
32             matrix[i][left] = num++;
33         }
34         left++; // 左边界右移
35     }
36 }

```



```
36 |
37 |     for (int i = 0; i < m; i++) { // 遍历矩阵的每一行
38 |         for (int j = 0; j < cols; j++) { // 遍历矩阵的每一列
39 |             if (matrix[i][j] == 0) { // 如果当前位置是0, 则输出'*'
40 |                 printf("* ");
41 |             } else { // 否则输出当前位置的数字
42 |                 printf("%d ", matrix[i][j]);
43 |             }
44 |         }
45 |         printf("\n"); // 每打印完一行后换行
46 |     }
47 |
48 |     return 0;
49 | }
```

文章目录

[华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷](#)

[题目描述](#)

[输入描述](#)

[输出描述](#)

[用例1](#)

[用例2](#)

[用例3](#)

[解题思路](#)

[C++](#)

[Java](#)

[javaScript](#)

[Python](#)

[C语言](#)

机考真题 华为OD



CSDN @算法大师