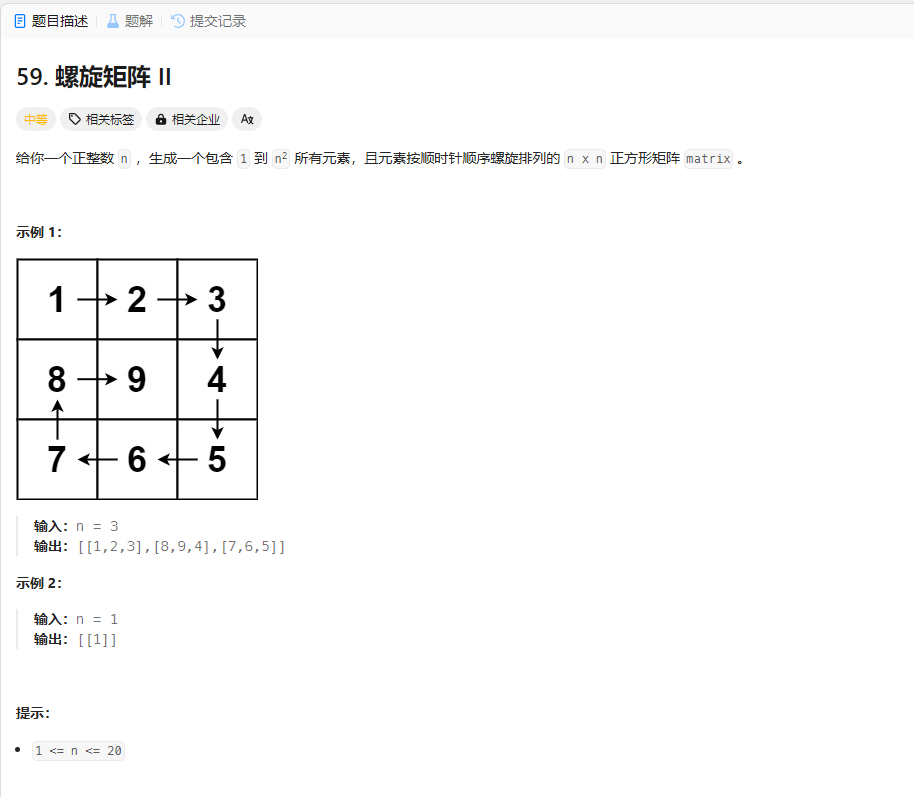
# 题目：



# 示例：n = 4

**第 1 次循环**

条件：left(0) <= right(3) && top(0) <= bottom(3)，成立。

1. **填充上边界**
   * 代码：for (int i = left; i <= right; i++) { matrix[top][i] = num++; }
   * 具体表达式：for (int i = 0; i <= 3; i++) { matrix[0][i] = num++; }
   * 执行：
     + i = 0: matrix[0][0] = 1, num = 2
     + i = 1: matrix[0][1] = 2, num = 3
     + i = 2: matrix[0][2] = 3, num = 4
     + i = 3: matrix[0][3] = 4, num = 5
   * top++: top = 1
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[0, 0, 0, 0],

[0, 0, 0, 0],

[0, 0, 0, 0]

]

1. **填充右边界**
   * 代码：for (int i = top; i <= bottom; i++) { matrix[i][right] = num++; }
   * 具体表达式：for (int i = 1; i <= 3; i++) { matrix[i][3] = num++; }
   * 执行：
     + i = 1: matrix[1][3] = 5, num = 6
     + i = 2: matrix[2][3] = 6, num = 7
     + i = 3: matrix[3][3] = 7, num = 8
   * right--: right = 2
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[0, 0, 0, 5],

[0, 0, 0, 6],

[0, 0, 0, 7]

]

1. **填充下边界**
   * 条件：top(1) <= bottom(3)，成立
   * 代码：for (int i = right; i >= left; i--) { matrix[bottom][i] = num++; }
   * 具体表达式：for (int i = 2; i >= 0; i--) { matrix[3][i] = num++; }
   * 执行：
     + i = 2: matrix[3][2] = 8, num = 9
     + i = 1: matrix[3][1] = 9, num = 10
     + i = 0: matrix[3][0] = 10, num = 11
   * bottom--: bottom = 2
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[0, 0, 0, 5],

[0, 0, 0, 6],

[10, 9, 8, 7]

]

1. **填充左边界**
   * 条件：left(0) <= right(2)，成立
   * 代码：for (int i = bottom; i >= top; i--) { matrix[i][left] = num++; }
   * 具体表达式：for (int i = 2; i >= 1; i--) { matrix[i][0] = num++; }
   * 执行：
     + i = 2: matrix[2][0] = 11, num = 12
     + i = 1: matrix[1][0] = 12, num = 13
   * left++: left = 1
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[12, 0, 0, 5],

[11, 0, 0, 6],

[10, 9, 8, 7]

]

**第 2 次循环**

条件：left(1) <= right(2) && top(1) <= bottom(2)，成立。

1. **填充上边界**
   * 代码：for (int i = left; i <= right; i++) { matrix[top][i] = num++; }
   * 具体表达式：for (int i = 1; i <= 2; i++) { matrix[1][i] = num++; }
   * 执行：
     + i = 1: matrix[1][1] = 13, num = 14
     + i = 2: matrix[1][2] = 14, num = 15
   * top++: top = 2
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[12, 13, 14, 5],

[11, 0, 0, 6],

[10, 9, 8, 7]

]

1. **填充右边界**
   * 代码：for (int i = top; i <= bottom; i++) { matrix[i][right] = num++; }
   * 具体表达式：for (int i = 2; i <= 2; i++) { matrix[i][2] = num++; }
   * 执行：
     + i = 2: matrix[2][2] = 15, num = 16
   * right--: right = 1
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[12, 13, 14, 5],

[11, 0, 15, 6],

[10, 9, 8, 7]

]

1. **填充下边界**
   * 条件：top(2) <= bottom(2)，成立
   * 代码：for (int i = right; i >= left; i--) { matrix[bottom][i] = num++; }
   * 具体表达式：for (int i = 1; i >= 1; i--) { matrix[2][i] = num++; }
   * 执行：
     + i = 1: matrix[2][1] = 16, num = 17
   * bottom--: bottom = 1
   * 结果：

text

收起自动换行复制

[

[1, 2, 3, 4],

[12, 13, 14, 5],

[11, 16, 15, 6],

[10, 9, 8, 7]

]

1. **填充左边界**
   * 条件：left(1) <= right(1)，成立
   * 代码：for (int i = bottom; i >= top; i--) { matrix[i][left] = num++; }
   * 具体表达式：for (int i = 1; i >= 2; i--) { matrix[i][1] = num++; }
   * 执行：范围为空（1 >= 2 不成立），无操作。
   * left++: left = 2
   * 结果不变：

text

收起自动换行复制

[

[1, 2, 3, 4],

[12, 13, 14, 5],

[11, 16, 15, 6],

[10, 9, 8, 7]

]

**第 3 次循环**

条件：left(2) <= right(1) && top(2) <= bottom(1)，不成立（2 <= 1 为假），循环结束。

**最终结果**

text

收起自动换行复制

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

# 代码：

#include <stdio.h>

#include <stdlib.h>

int\*\* generateMatrix(int n) {

// 动态分配一个 n x n 的二维数组

int\*\* matrix = (int\*\*)malloc(n \* sizeof(int\*));

for (int i = 0; i < n; i++) {

matrix[i] = (int\*)malloc(n \* sizeof(int));

for (int j = 0; j < n; j++) {

matrix[i][j] = 0; // 初始化为 0

}

}

int num = 1; // 从 1 开始填充

int left = 0; // 左边界

int right = n - 1; // 右边界

int top = 0; // 上边界

int bottom = n - 1; // 下边界

while (left <= right && top <= bottom) {

// 填充上边界

for (int i = left; i <= right; i++) {

matrix[top][i] = num;

num++;

}

top++; // 上边界向下移动

// 填充右边界

for (int i = top; i <= bottom; i++) {

matrix[i][right] = num;

num++;

}

right--; // 右边界向左移动

if (top <= bottom) {

// 填充下边界

for (int i = right; i >= left; i--) {

matrix[bottom][i] = num;

num++;

}

bottom--; // 下边界向上移动

}

if (left <= right) {

// 填充左边界

for (int i = bottom; i >= top; i--) {

matrix[i][left] = num;

num++;

}

left++; // 左边界向右移动

}

}

return matrix;

}

// 用于测试的打印函数

void printMatrix(int\*\* matrix, int n) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

printf("%2d ", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int n = 3;

int\*\* result = generateMatrix(n);

printMatrix(result, n);

// 释放内存

for (int i = 0; i < n; i++) {

free(result[i]);

}

free(result);

return 0;

}