#include <iostream> // Подключение библиотеки для ввода/вывода данных

#include <fstream> // Подключение библиотеки для работы с файлами

#include <vector> // Подключение библиотеки для работы с динамическими массивами (векторами)

#include <algorithm> // Подключение библиотеки для использования алгоритмов, например, sort

#include <locale> // Подключение библиотеки для изменения локали программы

using namespace std; // Использование стандартного пространства имен

// Функция для вывода матрицы

void printMatrix(const vector<vector<int>>& matrix) {

for (const auto& row : matrix) { // Проход по каждой строке матрицы

for (int val : row) { // Проход по каждому элементу в строке

cout << val << " "; // Вывод значения с пробелом

}

cout << endl; // Переход на новую строку после каждой строки матрицы

}

}

int main() {

setlocale(LC\_ALL, "rus"); // Установка локали на русский язык

ifstream inputFile("matrix.txt"); // Открытие файла для чтения матрицы

if (!inputFile) { // Проверка на успешное открытие файла

cerr << "Ошибка открытия файла." << endl; // Вывод ошибки в поток ошибок

return 1; // Выход из программы с ошибкой

}

int n; // Объявление переменной для размера матрицы

inputFile >> n; // Чтение размера матрицы из файла

vector<vector<int>> matrix(n, vector<int>(n)); // Создание матрицы размером n x n

for (int i = 0; i < n; ++i) { // Внешний цикл по строкам

for (int j = 0; j < n; ++j) { // Внутренний цикл по столбцам

inputFile >> matrix[i][j]; // Чтение значения элемента матрицы из файла

}

}

inputFile.close(); // Закрытие файла после чтения

// Обход матрицы по спирали и сохранение элементов в вектор

vector<int> spiralElements; // Вектор для хранения элементов в спиральном порядке

int top = 0, bottom = n - 1, left = 0, right = n - 1; // Инициализация границ обхода

while (top <= bottom && left <= right) { // Пока не выйдем за пределы матрицы

// Слева направо

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

spiralElements.push\_back(matrix[top][i]); // Добавляем элементы верхней строки

}

top++; // Сдвиг верхней границы вниз

// Сверху вниз

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

spiralElements.push\_back(matrix[i][right]); // Добавляем элементы правого столбца

}

right--; // Сдвиг правой границы влево

// Справа налево

if (top <= bottom) { // Проверка, что еще есть строки для обхода

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

spiralElements.push\_back(matrix[bottom][i]); // Добавляем элементы нижней строки

}

bottom--; // Сдвиг нижней границы вверх

}

// Снизу вверх

if (left <= right) { // Проверка, что еще есть столбцы для обхода

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

spiralElements.push\_back(matrix[i][left]); // Добавляем элементы левого столбца

}

left++; // Сдвиг левой границы вправо

}

}

// Сортировка элементов в векторе по убыванию

sort(spiralElements.begin(), spiralElements.end(), greater<int>()); // Сортируем вектор от большего к меньшему

// Заполнение матрицы отсортированными элементами по спирали

int index = 0; // Инициализация индекса для доступа к элементам вектора

top = 0, bottom = n - 1, left = 0, right = n - 1; // Сброс границ для заполнения матрицы

while (top <= bottom && left <= right) { // Пока не выйдем за пределы матрицы

// Слева направо

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

matrix[top][i] = spiralElements[index++]; // Заполнение верхней строки отсортированными элементами

}

top++; // Сдвиг верхней границы вниз

// Сверху вниз

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

matrix[i][right] = spiralElements[index++]; // Заполнение правого столбца

}

right--; // Сдвиг правой границы влево

// Справа налево

if (top <= bottom) { // Проверка, что еще есть строки для заполнения

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

matrix[bottom][i] = spiralElements[index++]; // Заполнение нижней строки

}

bottom--; // Сдвиг нижней границы вверх

}

// Снизу вверх

if (left <= right) { // Проверка, что еще есть столбцы для заполнения

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

matrix[i][left] = spiralElements[index++]; // Заполнение левого столбца

}

left++; // Сдвиг левой границы вправо

}

}

cout << "Отсортированная матрица по спирали:" << endl; // Сообщение перед выводом матрицы

printMatrix(matrix); // Вызов функции для вывода отсортированной матрицы

return 0; // Успешный выход из программы

}