Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2.2

з дисципліни
«Алгоритми і структури даних»

Виконала:
Студентка групи ІМ-12
Миць Вікторія Ігорівна
Номер у списку групи: 19

Перевірила:
Молчанова А. А

Київ 2021

Завдання:

Постановка задачі

1. Задано двовимірний масив (матрицю) цілих чисел ***A*[*m*,*n*]** або ***A*[*n*,*n*]**, де ***m*** та ***n*** – натуральні числа (константи), що визначають розміри двовимірного масиву. Виконати сортування цього масиву або заданої за варіантом його частини у заданому порядку заданим алгоритмом (методом).

Сортування повинно бути виконано безпосередньо у двовимірному масиві «на тому ж місці», тобто без перезаписування масиву та/або його будь-якої частини до інших одно- або двовимірних масивів, а також без використання спискових структур даних.

2. Розміри матриці ***m*** та ***n*** взяти самостійно у межах від 7 до 10.

3. При тестуванні програми необхідно підбирати такі вхідні набори початкових значеннь матриці, щоб можна було легко відстежити коректність виконання сортування і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

Варіант 19:

Задано квадратну двовимірний масив (матрицю) цілих чисел ***A*[*n*,*n*]**. Відсортувати головну діагональ масиву алгоритмом №2 методу вставки (з лінійним пошуком справа) за незменшенням.

Код алгоритму на С:

```c
#include <stdio.h>

void print_matrix(int m, int n, int matrix[m][n]) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%3d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void scan_matrix(int m, int n, int matrix[m][n]) {
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void sort(int m, int matrix[m][m]) {
    for (int i = 1; i < m; i++) {
        int element = matrix[i][i];
        int j = i;
        while (matrix[j - 1][j - 1] > element && j > 0) {
            matrix[j][j] = matrix[j - 1][j - 1];
            j--;
        }
        matrix[j][j] = element;
    }
}

int main() {
    int n;
    scanf("%d", &n);

    int matrix[n][n];
    scan_matrix(n, n, matrix);
    print_matrix(n, n, matrix);

    sort(n, matrix);

    printf("Sorted matrix: \n");
    print_matrix(n, n, matrix);

    return 0;
}
```

Результати тестувань

```
vika@MacBook-Pro-vika 2.2 Сортування % ./main < test_case_1.txt
 -5   0   0   0   0   0   0   0   0
  0   6   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0
  0   0   0 -43   0   0   0   0   0
  0   0   0   0   9   0   0   0   0
  0   0   0   0   0  22   0   0   0
  0   0   0   0   0   0   2   0   0
  0   0   0   0   0   0   0   9   0
  0   0   0   0   0   0   0   0   3
Sorted matrix:
-43   0   0   0   0   0   0   0   0
  0  -5   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0
  0   0   0   2   0   0   0   0   0
  0   0   0   0   3   0   0   0   0
  0   0   0   0   0   6   0   0   0
  0   0   0   0   0   0   9   0   0
  0   0   0   0   0   0   0   9   0
  0   0   0   0   0   0   0   0  22
vika@MacBook-Pro-vika 2.2 Сортування % ./main < test_case_2.txt
  1   0   0   0   0   0   0   0
  0   2   0   0   0   0   0   0
  0   0   4   0   0   0   0   0
  0   0   0   5   0   0   0   0
  0   0   0   0   7   0   0   0
  0   0   0   0   0   9   0   0
  0   0   0   0   0   0   9   0
  0   0   0   0   0   0   0  10
Sorted matrix:
  1   0   0   0   0   0   0   0
  0   2   0   0   0   0   0   0
  0   0   4   0   0   0   0   0
  0   0   0   5   0   0   0   0
  0   0   0   0   7   0   0   0
  0   0   0   0   0   9   0   0
  0   0   0   0   0   0   9   0
  0   0   0   0   0   0   0  10
vika@MacBook-Pro-vika 2.2 Сортування % ./main < test_case_3.txt
 98   0   0   0   0   0   0   0   0   0
  0  66   0   0   0   0   0   0   0   0
  0   0  66   0   0   0   0   0   0   0
  0   0   0  63   0   0   0   0   0   0
  0   0   0   0  43   0   0   0   0   0
  0   0   0   0   0   2   0   0   0   0
  0   0   0   0   0   0   2   0   0   0
  0   0   0   0   0   0   0   1   0   0
  0   0   0   0   0   0   0   0  -4   0
  0   0   0   0   0   0   0   0   0 -99
Sorted matrix:
-99   0   0   0   0   0   0   0   0   0
  0  -4   0   0   0   0   0   0   0   0
  0   0   1   0   0   0   0   0   0   0
  0   0   0   2   0   0   0   0   0   0
  0   0   0   0   2   0   0   0   0   0
  0   0   0   0   0  43   0   0   0   0
  0   0   0   0   0   0  63   0   0   0
  0   0   0   0   0   0   0  66   0   0
  0   0   0   0   0   0   0   0  66   0
  0   0   0   0   0   0   0   0   0  98
```