**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»

Виконав Перевірила:

Студент групи ІП-03 Сергієнко А. А.  
Пашковський Євгеній Сергійович  
номер у списку групи: 18

Київ 2020

**Завдання**

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.

1. Розміри матриці **m** та **n** взяти самостійно у межах від 7 до 10.
2. При тестуванні програми необхідно підбирати такі вхідні набори початкових значень 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 18:

Задано двовимірний масив (матрицю) цілих чисел ***A[m,n]***. Відсортувати окремо кожен стовпчик масиву методом швидкого сортування (методом Хоара) за незбільшенням.

**Текст програми**

#include <stdio.h>

#include <windows.h>

#define decrease 1

#define increase -1

int main () {// 0 1 2 3 4 5 6 7 8 9

const int MATRIX\_WITH\_DATA[7][10] = { {-1 , 19 , 2 , 2 , 0 , 3 , 2 , 1 , -1 , -3 /\*0\*/},

{-12 , -40 , 4 , 2 , 0 , 230 ,-4 , 1 , -1 , -8 /\*1\*/},

{ 0 , 4 , 2 , -2 , 0 , 1 , 1 , 1 , -1 , -1 /\*2\*/},

{ 2 , 0 , 4 , 9 , 2 , 4 , 1 , 0 , -2 , -5 /\*3\*/},

{-1 , -2 , 5 , 3 , -1 , -2 , 8 , 1 , -1 , -2 /\*4\*/},

{ 5 , 0 , 5 , 0 , 3 , 4 , 2 , 1 , -1 , -7 /\*5\*/},

{ 20 , 2 , 5 , -3 , -23 , 4 , 1 , 1 , -1 , -1 /\*6\*/} };

const int RAW\_LENGTH = 10; //Довжина рядка

const int COL\_LENGTH = 7; //Довжина стовпчика

COORD GetConsoleCursorPosition(HANDLE hConsoleOutput) {

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

if (GetConsoleScreenBufferInfo(hConsoleOutput, &csbi)) {

return csbi.dwCursorPosition;

} else {

COORD invalid = { 0, 0 };

return invalid;

}

}

HANDLE hout = GetStdHandle(STD\_OUTPUT\_HANDLE);

void gotoX(int x){

x += GetConsoleCursorPosition(hout).X;

COORD pos = {x, GetConsoleCursorPosition(hout).Y};

SetConsoleCursorPosition(hout, pos);

}

void gotoY(int y){

y += GetConsoleCursorPosition(hout).Y;

COORD pos = {GetConsoleCursorPosition(hout).X, y};

SetConsoleCursorPosition(hout, pos);

}

void drawMatrix(int matrix[COL\_LENGTH][RAW\_LENGTH]) {

printf("\n\n");

for (int i = 0; i < COL\_LENGTH; i++) {

for (int j = 0; j < RAW\_LENGTH; j++){

gotoX(2);

printf("%3d", matrix[i][j]);

}

printf("\n\n");

}

printf("\n");

}

int sortMatrixByHoar(int matrix[COL\_LENGTH][RAW\_LENGTH], int direction) {

int useHoar(int m[COL\_LENGTH][RAW\_LENGTH], int L, int R, int i, int dir) {

int K = L;

int M = R;

int T = m[L][i]; //опорний елемент

while (L < R) {

while (m[R][i] \* direction < T \* direction && L < R) { //рухаємо правий вказівник

R--;

}

//знайшли число, що неправильно розташоване відносно опорного елементу правим вказівником

if (L != R) {

m[L][i] = m[R][i];

L++;

}

while (m[L][i] \* direction > T \* direction && L < R) { //рухаємо лівий вказівник

L++;

}

//знайшли число, що неправильно розташоване відносно опорного елементу лівим вказівником

if (L != R) {

m[R][i] = m[L][i];

R--;

}

} //(L <= R) -> розміщаємо опорний елемент

m[L][i] = T;

int P = L; // координата опорного елементу

// рекурсивно визиваємо для лівої частини

L = K;

R = P - 1;

if (P != L && P != R) {

m = useHoar(m, L, R, i, dir);

}

//рекурсивно визиваємо для правої частини

L = P + 1;

R = M;

if (P != L && P != R) {

m = useHoar(m, L, R, i, dir);

}

// немає елементів по один бік від опорного елемента -> повертаємо відсортований масив

return m;

}

//сортуємо кожен стовпчик масиву

for (int i = 0; i < RAW\_LENGTH; i++) {

matrix = useHoar(matrix, 0, COL\_LENGTH - 1, i, direction);

}

return matrix;

}

int x = 0;

while (x != 1 && x != 2 && x != 3) {

printf("Enter the matrix for testing (1 = original, 2 = sorted, 3 = reversed-sorted): ");

scanf("%d", &x);

}

printf("\n");

switch (x) {

case 1: //сортуємо масив з випадковими числами

printf("Original matrix: \n");

drawMatrix(MATRIX\_WITH\_DATA);

printf("Sorted matrix: \n");

drawMatrix(sortMatrixByHoar(MATRIX\_WITH\_DATA, decrease));

break;

case 2: //сортуємо завчасно відсортований масив

printf("Original matrix (originally sorted): \n");

drawMatrix(sortMatrixByHoar(MATRIX\_WITH\_DATA, decrease));

printf("Re-sorted sorted matrix: \n");

drawMatrix(sortMatrixByHoar(sortMatrixByHoar(MATRIX\_WITH\_DATA, decrease), decrease));

break;

case 3: //сортуємо завчасно обернено відсортований масив

printf("Original matrix (originally reversed-sorted): \n");

drawMatrix(sortMatrixByHoar(MATRIX\_WITH\_DATA, increase));

printf("Re-sorted reversed-sorted matrix: \n");

drawMatrix(sortMatrixByHoar(sortMatrixByHoar(MATRIX\_WITH\_DATA, increase), decrease));

break;

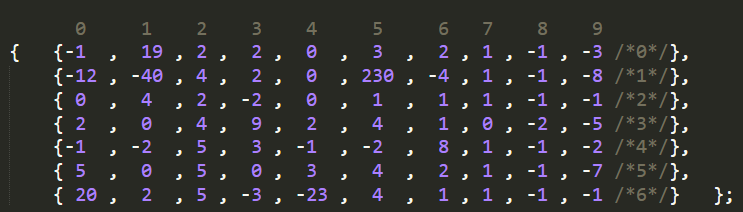
}

system("pause");

return 0;

}

**Вхідні дані**



**Тестування програми**

