Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний

інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 9 з дисципліни

«Алгоритми та структури даних-1.

Основи алгоритмізації»

«ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОБХОДУ МАСИВІВ»

Варіант 25

Виконав студент ІП-12 Піонтківський Віталій Петрович

(шифр, прізвище, ім'я, по батькові)

Перевірив \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( прізвище, ім'я, по батькові)

Київ 2021

**Лабораторна робота 9**

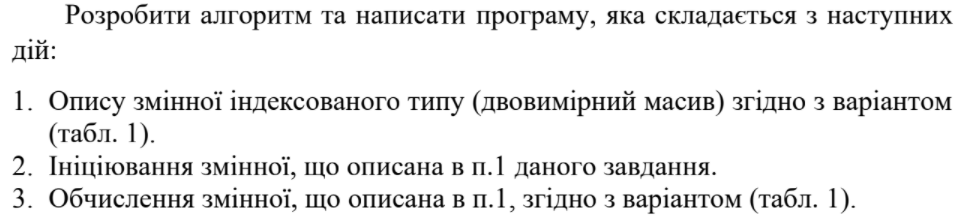
**ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОБХОДУ МАСИВІВ**

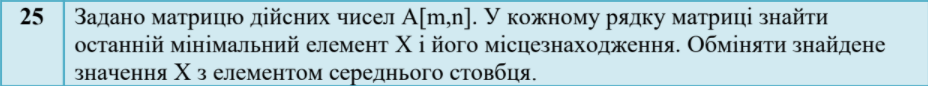
Мета – дослідити алгоритми обходу масивів, набути практичних

навичок використання цих алгоритмів під час складання програмних

специфікацій.

**Задача 25.**



****

**Побудова математичної моделі**

Для вирішення даної задачі константами визначимо розмір матриці, згенеруємо її з псевдовипадкових чисел (від 0 до 100), по рядкам знайдемо останнє найменше число в рядку та поміняємо місцями з середнім (Якщо кількість стовпців парна – вважав середнім менше. Наприклад 6 стовпців – 3ій середній).

Складемо таблицю імен змінних.

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Кількість рядків матриці (константа) | Додатній | height | Початкове дане |
| Кількість стовпців матриці (константа) | Додатній | width | Початкове дане |
| Матриця | матриця з додатних чисел | matrix | Результат |
| Лічильник циклу | Додатній | i | Проміжне дане |
| Лічильник циклу | Додатній | j | Проміжне дане |
| Номер стовпця найменшого елементу в рядку | Додатній | minw | Проміжне дане |
| Найменший елемент рядка | Додатній | min | Проміжне дане |
| Номер середнього стовпця | Додатній | middle | Проміжне дане |
| Генерація чисел у матриці | Функція | generate | Проміжне дане |
| Виведення матриці | Функція | output | Проміжне дане |
| Знаходження мінімального елементу кожного рядка, його місцезнаходження та зміна його з середнім елементом | Функція | solve | Проміжне дане |

Розв’язання. Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

*Крок 1*. Визначимо основні дії.

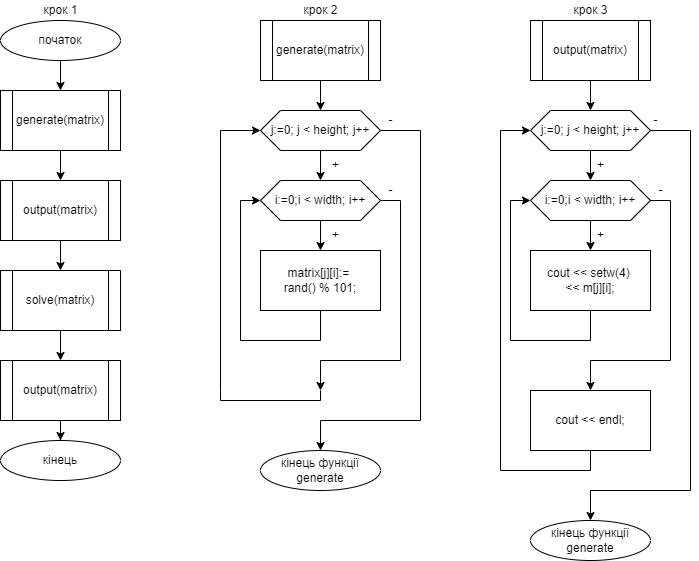
*Крок 2*. Деталізуємо дію генерування чисел у матриці.

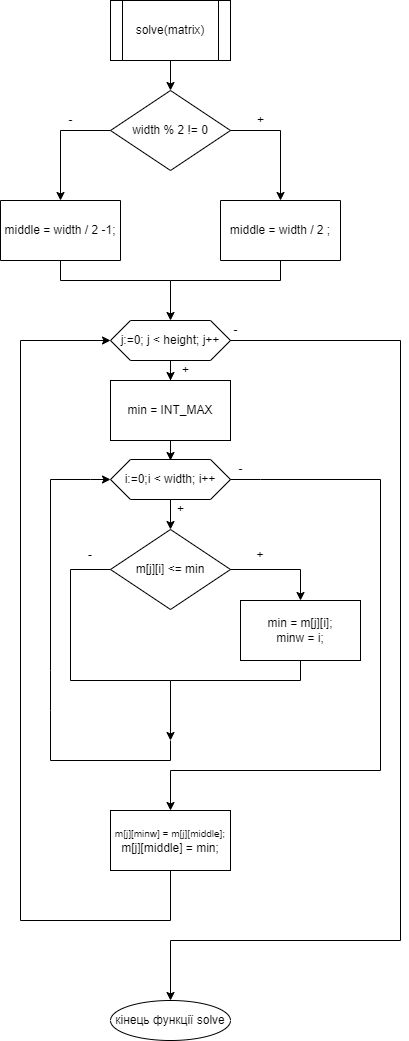
*Крок 3*. Деталізуємо дію виведення матриці.

*Крок 4*. Деталізуємо дію знаходження мінімального елементу кожного рядка, його місцезнаходження та зміна його з середнім елементом.

**Псевдокод**

|  |
| --- |
| Крок 1  **початок**  generate(matrix);  output(matrix);  solve(matrix);  output(matrix);  **кінець**  **початок функції generate**(matrix)  Генерація чисел у матриці  **кінець функції generate**  **початок функції output**(matrix)  Виведення матриці  **кінець функції output**  **початок функції solve**(matrix)  Знаходження мінімального елементу кожного рядка, його місцезнаходження та зміна його з середнім елементом  **кінець функції solve** |
| Крок 2  **початок**  generate(matrix);  output(matrix);  solve(matrix);  output(matrix);  **кінець**  **початок функції generate**(matrix)  **повторити**  **поки** j = 0; j < height; j++;  **повторити**  **поки** i = 0; i < width; i++;  matrix[j][i]:= rand() % 101;  **все повторити**  **все повторити**  **кінець функції generate**  **початок функції output**(matrix)  Виведення матриці  **кінець функції output**  **початок функції solve**(matrix)  Знаходження мінімального елементу кожного рядка, його місцезнаходження та зміна його з середнім елементом  **кінець функції solve** |
| Крок 3  **початок**  generate(matrix);  output(matrix);  solve(matrix);  output(matrix);  **кінець**  **початок функції generate**(matrix)  **повторити**  **поки** j = 0; j < height; j++;  **повторити**  **поки** i = 0; i < width; i++;  matrix[j][i]:= rand() % 101;  **все повторити**  **все повторити**  **кінець функції generate**  **початок функції output**(matrix)  **повторити**  **поки** j = 0; j < height; j++;  **повторити**  **поки** i = 0; i < width; i++;  cout << setw(4) << m[j][i];  **все повторити**  cout <<endl;  **все повторити**  **кінець функції output**  **початок функції solve**(matrix)  Знаходження мінімального елементу кожного рядка, його місцезнаходження та зміна його з середнім елементом  **кінець функції solve** |
| Крок 4  **початок**  generate(matrix);  output(matrix);  solve(matrix);  output(matrix);  **кінець**  **початок функції generate**(matrix)  **повторити**  **поки** j = 0; j < height; j++;  **повторити**  **поки** i = 0; i < width; i++;  matrix[j][i]:= rand() % 101;  **все повторити**  **все повторити**  **кінець функції generate**  **початок функції output**(matrix)  **повторити**  **поки** j = 0; j < height; j++;  **повторити**  **поки** i = 0; i < width; i++;  cout << setw(4) << m[j][i];  **все повторити**  cout <<endl;  **все повторити**  **кінець функції output**  **початок функції solve**(matrix)  **якщо** middle % 2 != 0;  **то** middle:= width/2;  **інакше** middle:= width/2 – 1;  **все якщо**  **повторити**  **поки** j = 0; j < height; j++;  min:= INT\_MAX;  **повторити**  **поки** i = 0; i < width; i++;  **якщо** matrix[j][i] <= min  **то** min := matrix[j][i];  minw := i;  **все повторити**  matrix[j][minw] = matrix[j][middle];  matrix[j][middle] = min;  **все повторити**  **кінець функції solve** |

**Блок-схема**

****

**Код**

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

const int height = 6;

const int width = 6;

void generate(int m[height][width]); // generate numbers in matrix

void output( int m[height][width]); // output matrix

void solve( int m[height][width]); // operations from task

int main()

{

srand(time(NULL));

int matrix[height][width];

generate(matrix);

output( matrix);

cout << "===========================================" << endl;

solve(matrix);

output(matrix);

}

void generate(int m[height][width]) {

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

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

m[j][i] = rand() % 101;

}

}

}

void output(int m[height][width]) {

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

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

cout << setw(4) << m[j][i];

}

cout << endl;

}

}

void solve(int m[height][width]) {

int middle;

if (width % 2 != 0) {

middle = width / 2 ;

}

else middle = width / 2 -1;

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

int min = INT\_MAX, minw;

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

if (m[j][i] <= min) {

min = m[j][i];

minw = i;

}

}

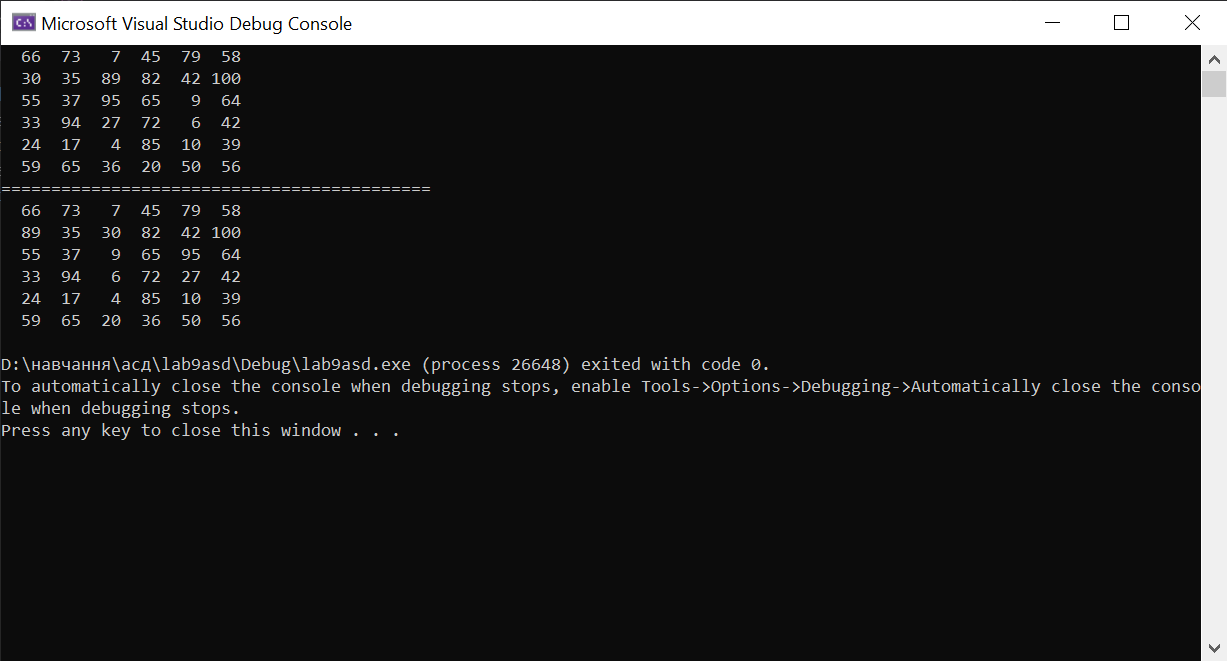
m[j][minw] = m[j][middle];

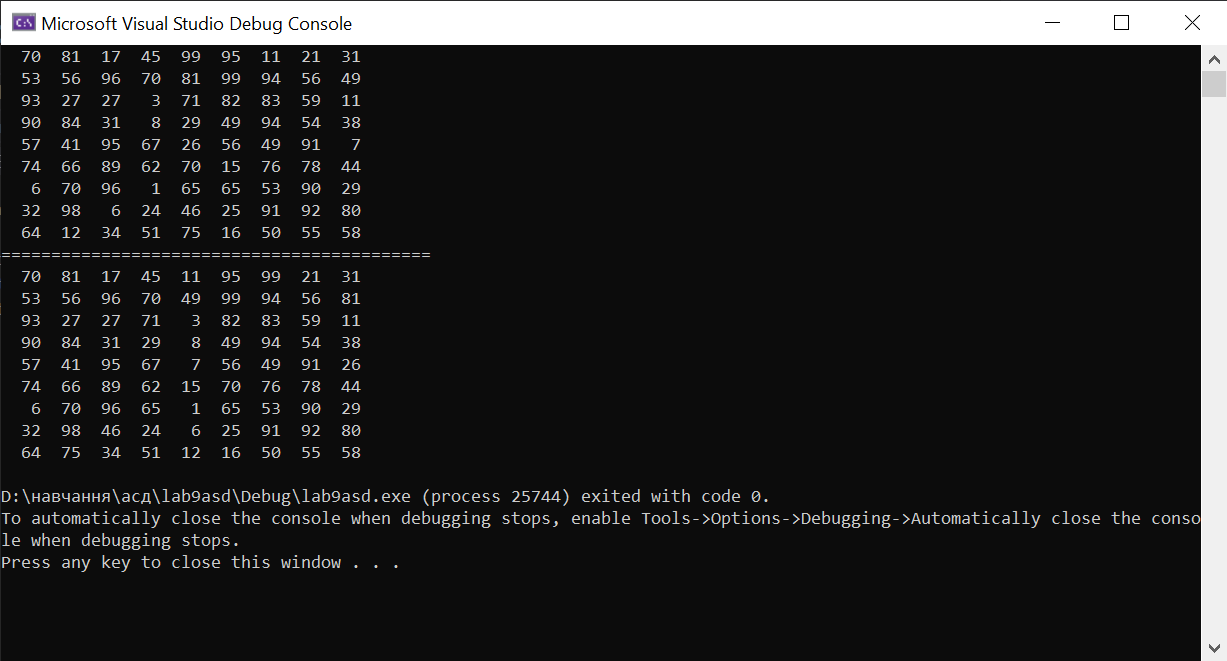
m[j][middle] = min;

}

}

**Перевірка**

****



**Висновок**

Було досліджено алгоритми обходу масивів, були набуті практичні навички використання цих алгоритмів під час складання програмних специфікацій на прикладі даного завдання: генерації матриці, знаходження найменшого елемента кожного рядка і зміна його з елементом середнього стовпця. Особливостями роботи є варіативність щодо парної і непарної кількості стовпців, «запам’ятовування» найменшого елемента і номеру його стовпця.