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 31

Виконав студент ІП-11 Трикош Іван Володимирович

Перевірила Мартинова О. П.

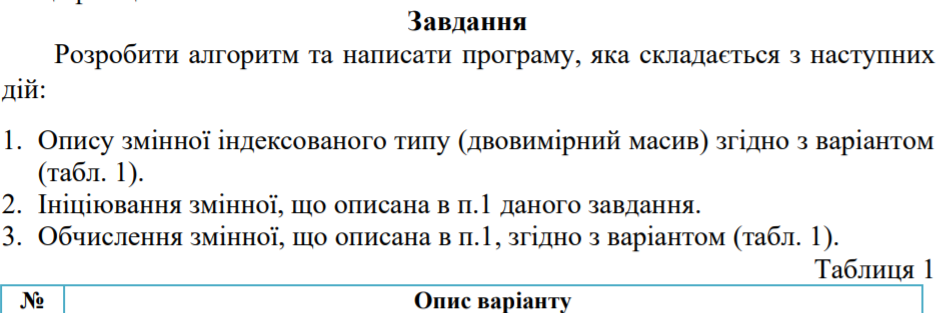
Київ 2021

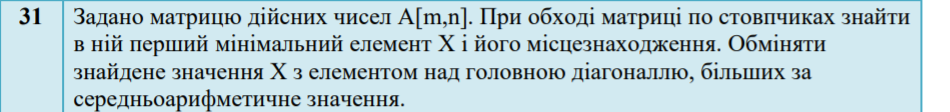
**Лабораторна робота 9**

**Дослідження алгоритмів обходу масивів**

**Мета** – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Варіант 31.





**Постановка задачі –** спочатку потрібно ініціалізувати двовимірний масив випадковими дійсними числами та вивести його. Далі обійти цей масив за стовпцями та знайти середнє арифметичне елементів над верхньою діагоналлю і знайти мінімальний елемент і його індекси. Потім знову обійти масив за стовпцями і якщо елемент над верхньою діагоналлю більший за середнє арифметичне елементів над верхньою діагоналлю, то замінити його на мінімальний елемент. Вивести оброблений масив.

**Побудова математичної моделі**

Складемо таблицю імен змінних:

|  |  |  |  |
| --- | --- | --- | --- |
| Змінна | Тип | Ім’я | Призначення |
| Кількість рядків | Цілий (const) | m | Початкове дане |
| Кількість стовпців | Цілий (const) | n | Початкове дане |
| Двовимірний масив | Дійсний | A | Початкове дане і результат |
| Мінімальний елемент | Дійсний | x | Проміжне дане |
| Індекс рядка елемента Х | Цілий | row | Проміжне дане |
| Індекс стовпця елемента Х | Цілий | column | Проміжне дане |
| Середнє арифм. елементів | Дійсний | sum | Проміжне дане |
| Змінна для обходу за стовпцями | Булевий | flag | Проміжне дане |
| Кількість елем. над головною діагоналлю | Цілий | a | Проміжне дане |
| Покажчик на перший елемент масиву A | Вказує на дійсний тип | p | Покажчик |
| Параметр циклу | Цілий | i | Лічильник |
| Параметр циклу | Цілий | j | Лічильник |
| Посилання на мінімальний елемент | Вказує на дійсний тип | x | Посилання |
| Посилання на індекс рядка елемента Х | Вказує на цілий тип | row | Посилання |
| Посилання на індекс стовпця елемента Х | Вказує на цілий тип | column | Посилання |
| Посилання на середнє арифм. елементів | Вказує на дійсний тип | sum | Посилання |

Таким чином, математичне формулювання задачі зводиться до ініціалізації масиву випадковими дійсними числами, знаходження мінімального елемента і його індекси, знаходження середнього арифметичного елементів масиву, що знаходяться над головною діагналлюі, і обходу матриць за стовпцями: якщо елемент над головною діагоналлю більший за середнє арифметичне, то замінити його на мінімальний елемент.

**Псевдокод алгоритму**

*Крок 1. Визначимо основні дії*

*Крок 2. Оголошуємо масив, його розмір та допоміжні змінні*

*Крок 3. Викликаємо та деталізуємо функцію (ініціалізація та виведення початкового масиву) і її тіло*

*Крок 4. Викликаємо та деталізуємо функцію (обробка масиву) і її тіло*

*Крок 5. Викликаємо та деталізуємо функцію (вивід обробленого масиву) і її тіло*

**Крок 1**

**Початок**

Оголошуємо масив, його розмір та допоміжні змінні

Викликаємо функцію для ініціалізації та виводу початкового масиву

Викликаємо функцію для обробки масиву

Викликаємо функцію для виводу обробленого масиву

**Кінець**

**Крок 2**

**Початок**

const int m = 5

const int n = 5

double A[m][n]

double x = 10.1

int row

int column

double sum = 0

Викликаємо функцію для ініціалізації та виводу початкового масиву

Викликаємо функцію для обробки масиву

Викликаємо функцію для виводу обробленого масиву

**Кінець**

**Крок 3**

**Початок**

const int m = 5

const int n = 5

double A[m][n]

double x = 10.1

int row

int column

double sum = 0

Inp\_Out\_A(&A[0][0], m, n)

Викликаємо функцію для обробки масиву

Викликаємо функцію для виводу обробленого масиву

**Кінець**

**Підпрограма 1**

Inp\_Out\_A(double\* p, int m, int n)

int i = 0

**Повторити** m **разів**

int j = 0

**Повторити** n **разів**

\*(p + m \* i + j) = (double)(rand()) / RAND\_MAX \* 10

Вивід \*(p + m \* i + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Крок 4**

**Початок**

const int m = 5

const int n = 5

double A[m][n]

double x = 10.1

int row

int column

double sum = 0

Inp\_Out\_A(&A[0][0], m, n)

Solution(&A[0][0], m, n, x, row, column, sum)

Викликаємо функцію для виводу обробленого масиву

**Кінець**

**Підпрограма 1**

Inp\_Out\_A(double\* p, int m, int n)

int i = 0

**Повторити** m **разів**

int j = 0

**Повторити** n **разів**

\*(p + m \* i + j) = (double)(rand()) / RAND\_MAX \* 10

Вивід \*(p + m \* i + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Підпрограма 2**

Solution(double\* p, int m, int n, double& x, int& row, int& column,

double& sum)

bool flag = false

int a = 0

int i = 0

**Повторити** m **разів**

**Якщо (**flag == false) **то**

int j = 0

**Повторити** n **разів**

**Якщо** (i > j) **то**

sum += \*(p + n \* j + i)

a++

**Все якщо**

**Якщо (**\*(p + n \* j + i) < x) **то**

x = \*(p + n \* j + i)

row = j

column = i

**Все якщо**

j++

**Все повторити**

**Інакше**

int j = n - 1

**Повторити** n **разів**

**Якщо** (i > j) **то**

sum += \*(p + n \* j + i)

a++

**Все якщо**

**Якщо (**\*(p + n \* j + i) < x) **то**

x = \*(p + n \* j + i)

row = j

column = i

**Все якщо**

j--

**Все повторити**

**Все якщо**

flag = !flag

i++

**Все повторити**

sum = sum / a

flag = false

int i = 0

**Повторити** m **разів**

**Якщо (**flag == false) **то**

int j = 0

**Повторити** n **разів**

**Якщо** (j < i) **то**

**Якщо (**\*(p + n \* j + i) > sum) **то**

\*(p + n \* j + i) = x

**Все якщо**

**Все якщо**

j++

**Все повторити**

**Інакше**

int j = n - 1

**Повторити** n **разів**

**Якщо** (j < i) **то**

**Якщо (**\*(p + n \* j + i) > sum) **то**

\*(p + n \* j + i) = x

**Все якщо**

**Все якщо**

j--

**Все повторити**

**Все якщо**

flag = !flag

i++

**Все повторити**

**Кінець**

**Крок 5**

**Початок**

const int m = 5

const int n = 5

double A[m][n]

double x = 10.1

int row

int column

double sum = 0

Inp\_Out\_A(&A[0][0], m, n)

Solution(&A[0][0], m, n, x, row, column, sum)

Out\_answer(&A[0][0], m, n, x, row, column, sum)

**Кінець**

**Підпрограма 1**

Inp\_Out\_A(double\* p, int m, int n)

int i = 0

**Повторити** m **разів**

int j = 0

**Повторити** n **разів**

\*(p + m \* i + j) = (double)(rand()) / RAND\_MAX \* 10

Вивід \*(p + m \* i + j)

j++

**Все повторити**

i++

**Все повторити**

**Кінець**

**Підпрограма 2**

Solution(double\* p, int m, int n, double& x, int& row, int& column,

double& sum)

bool flag = false

int a = 0

int i = 0

**Повторити** m **разів**

**Якщо (**flag == false) **то**

int j = 0

**Повторити** n **разів**

**Якщо** (i > j) **то**

sum += \*(p + n \* j + i)

a++

**Все якщо**

**Якщо (**\*(p + n \* j + i) < x) **то**

x = \*(p + n \* j + i)

row = j

column = i

**Все якщо**

j++

**Все повторити**

**Інакше**

int j = n - 1

**Повторити** n **разів**

**Якщо** (i > j) **то**

sum += \*(p + n \* j + i)

a++

**Все якщо**

**Якщо (**\*(p + n \* j + i) < x) **то**

x = \*(p + n \* j + i)

row = j

column = i

**Все якщо**

j--

**Все повторити**

**Все якщо**

flag = !flag

i++

**Все повторити**

sum = sum / a

flag = false

int i = 0

**Повторити** m **разів**

**Якщо (**flag == false) **то**

int j = 0

**Повторити** n **разів**

**Якщо** (j < i) **то**

**Якщо (**\*(p + n \* j + i) > sum) **то**

\*(p + n \* j + i) = x

**Все якщо**

**Все якщо**

j++

**Все повторити**

**Інакше**

int j = n - 1

**Повторити** n **разів**

**Якщо** (j < i) **то**

**Якщо (**\*(p + n \* j + i) > sum) **то**

\*(p + n \* j + i) = x

**Все якщо**

**Все якщо**

j--

**Все повторити**

**Все якщо**

flag = !flag

i++

**Все повторити**

**Кінець**

**Підпрограма 3**

Out\_answer(double\* p, int m, int n, double& x, int& row, int& column, double& sum)

Вивід x і його індексів

Вивід sum

int i = 0

**Повторити** m **разів**

int j = 0

**Повторити** n **разів**

Вивід \*(p + m \* i + j)

j++

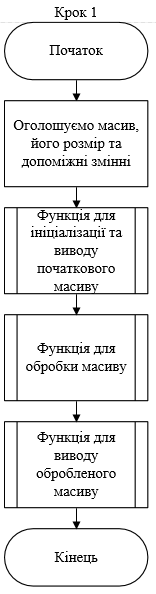
**Все повторити**

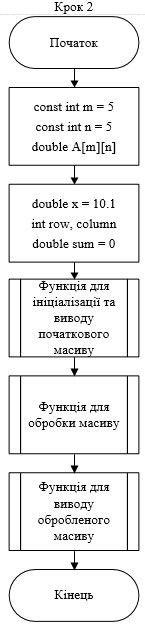
i++

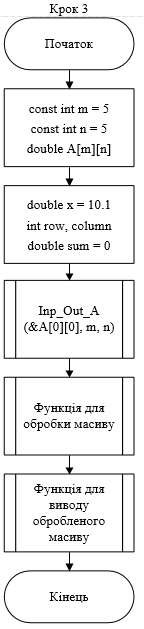
**Все повторити**

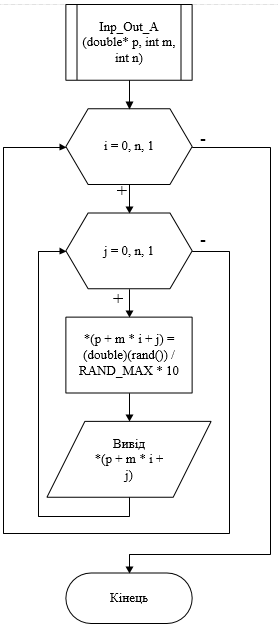
**Кінець**

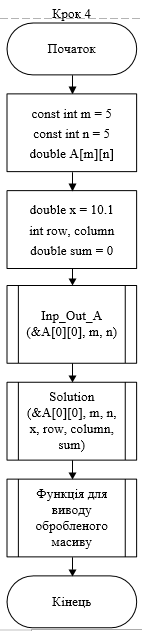
**Блок-схема алгоритму**

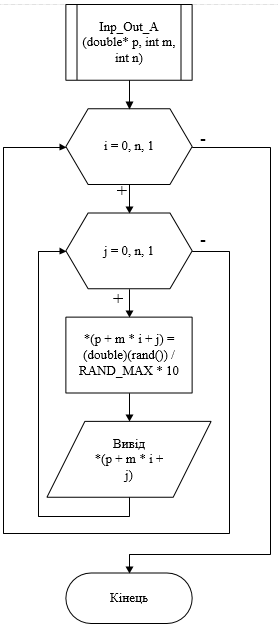
****

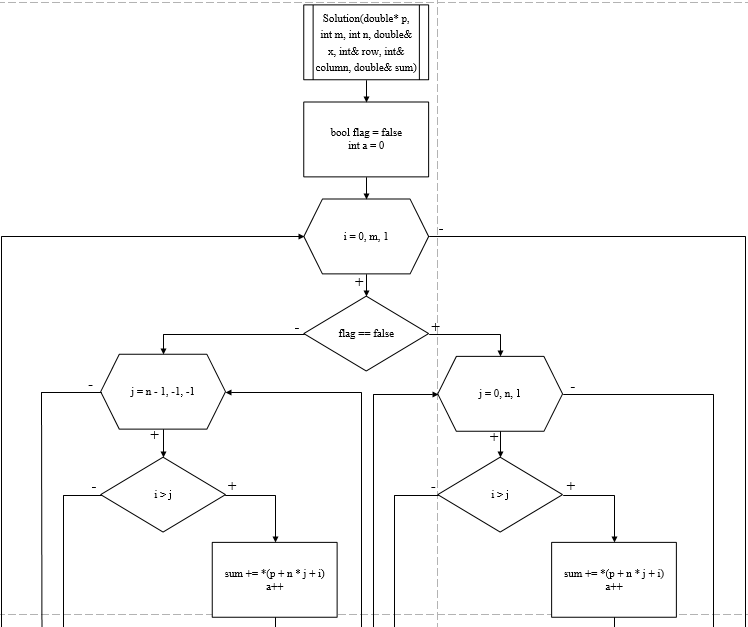
****

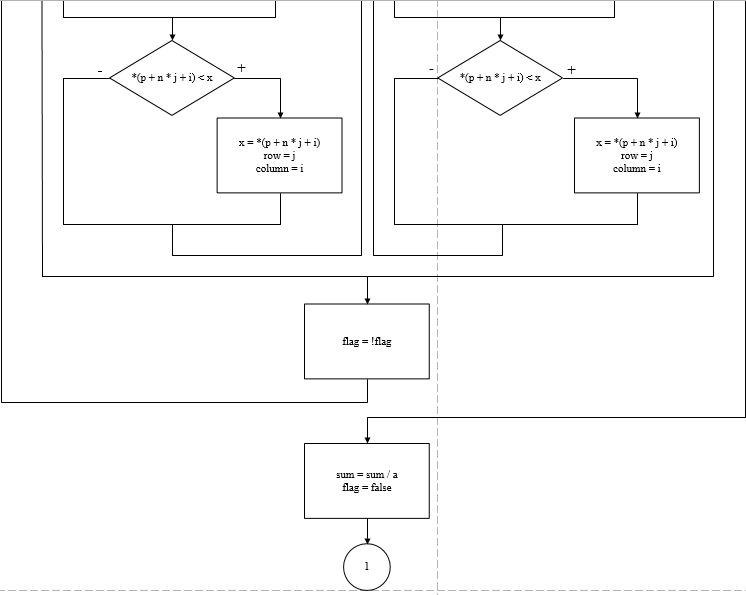
****

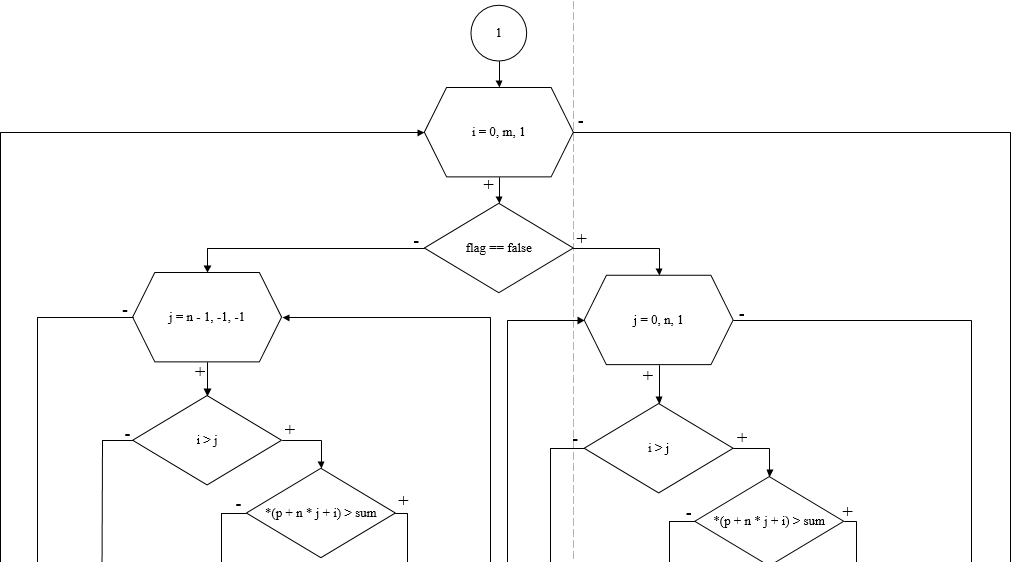
****

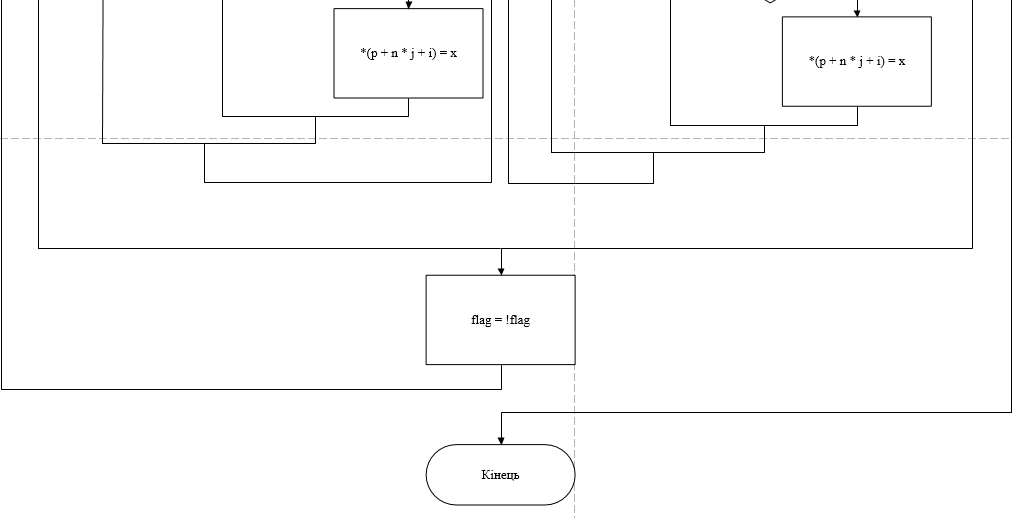
****

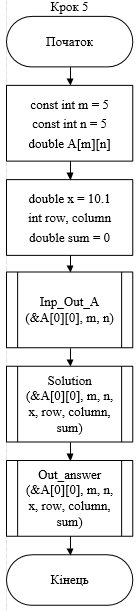
****

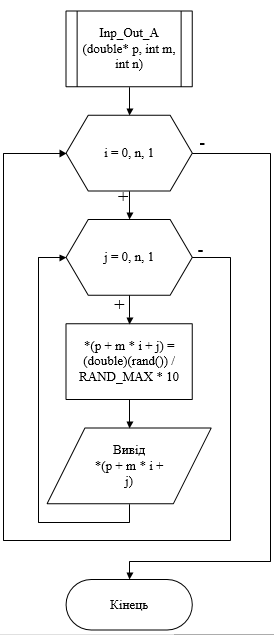
****

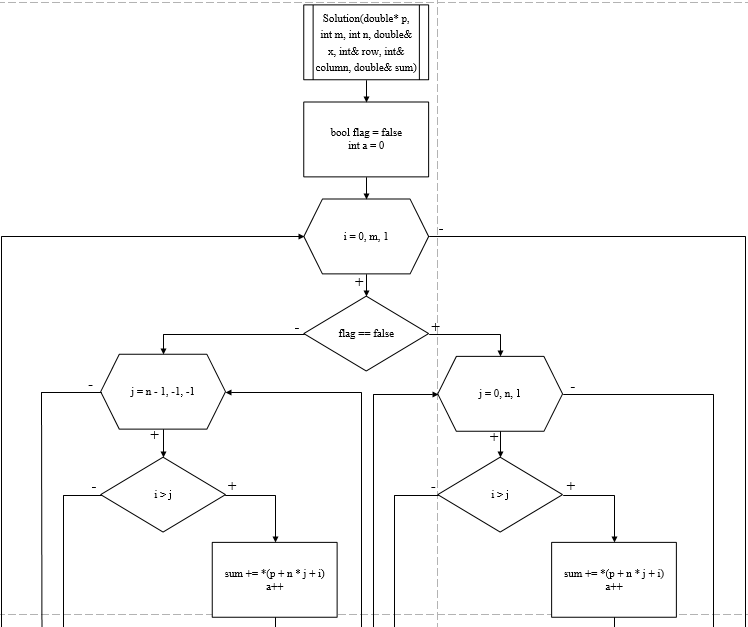
****

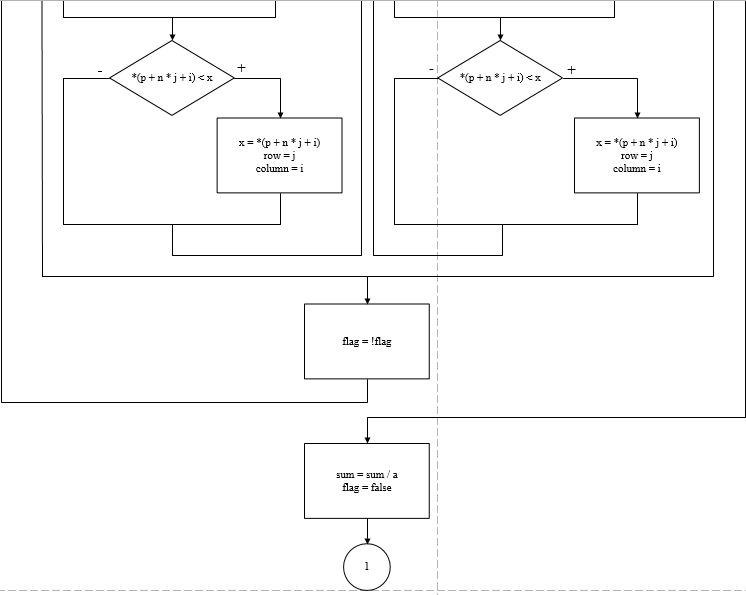
****

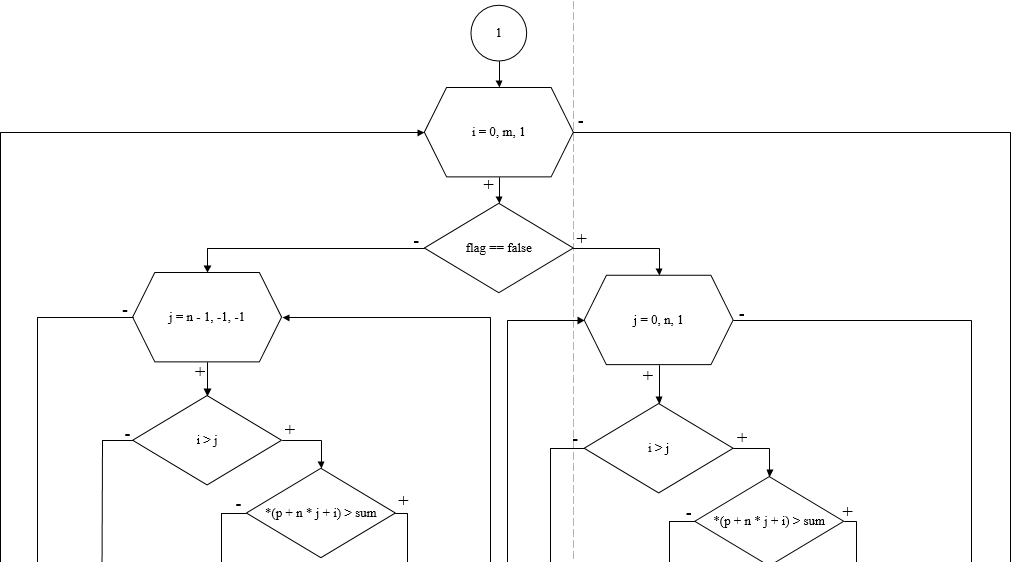
****

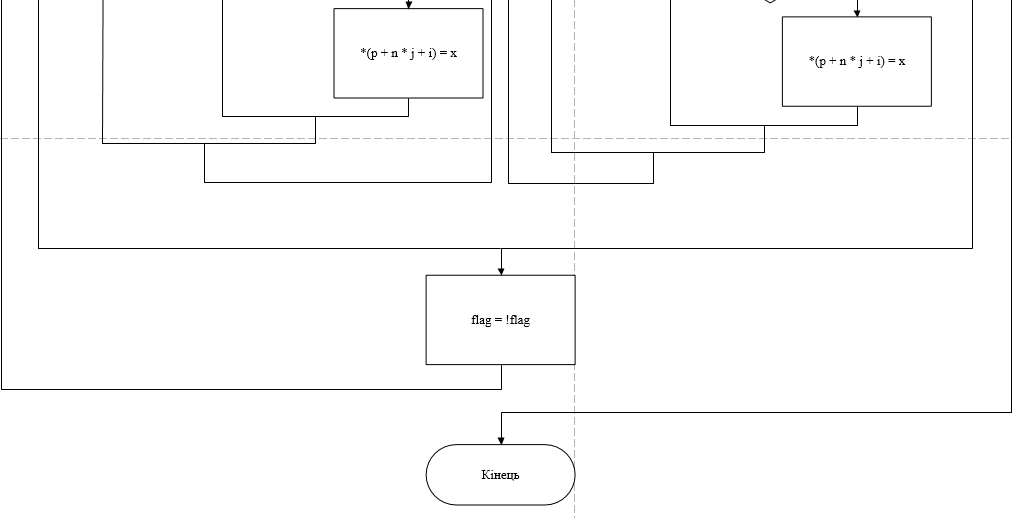
****

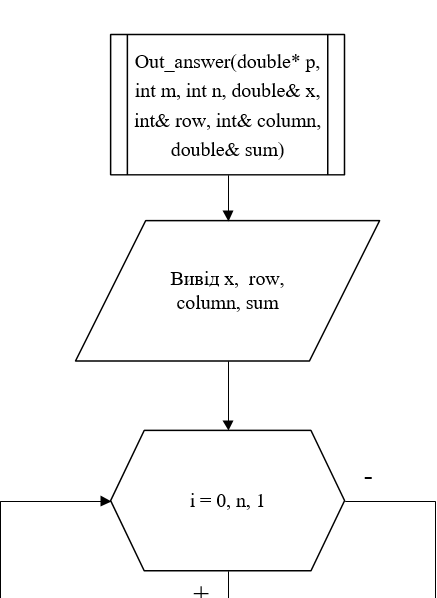
****

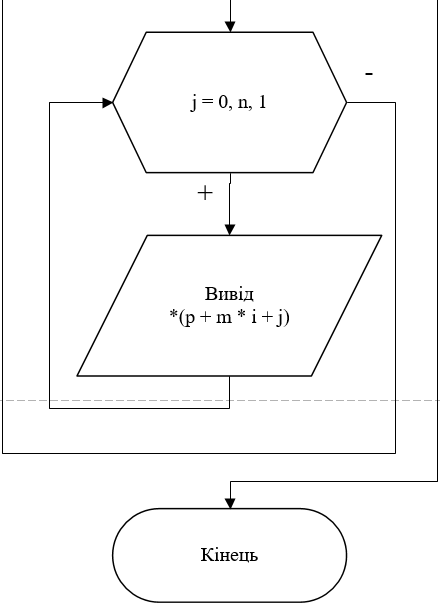
****

****

****

****

****

****

**Код програми на С++**

#include <iostream>

#include <stdlib.h>

using namespace std;

void Inp\_Out\_A(double\*, int, int); // Ініціалізуємо та виводимо перший масив

void Solution(double\*, int, int, double&, int&, int&, double&); // Оброблюємо // масив

void Out\_answer(double\*, int, int, double&, int&, int&, double&); // Виводимо // кінцевий масив

int main()

{

const int m = 5; // Кількість рядків

const int n = 5; // Кількість стовпців

double A[m][n]; // Оголошуємо масив

double x = 10.1; // Мінімальний елемент

// Індекси мінімального елемента

int row;

int column;

// Середньоарифметичне значення елементів

double sum = 0;

Inp\_Out\_A(&A[0][0], m, n); // Ініціалізуємо та виводимо перший масив

Solution(&A[0][0], m, n, x, row, column, sum); // Оброблюємо масив

Out\_answer(&A[0][0], m, n, x, row, column, sum); // Виводимо кінцевий // масив

}

void Inp\_Out\_A(double\* p, int m, int n)

{

srand(time(NULL));

cout << "Array A:";

for (int i = 0; i < m; i++)

{

cout << "\n";

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

{

\*(p + m \* i + j) = (double)(rand()) / RAND\_MAX \* 10; // Випадкове // дійсне число від 0 до 10

printf(" %.2f", \*(p + m \* i + j)); // Виводимо елементи масиву

}

}

}

void Solution(double\* p, int m, int n, double& x, int& row, int& column, double& sum)

{

bool flag = false; // Для обходу масиву за стовпцями

int a = 0; // Кількість елементів над головною діагоналлю

// Знаходимо середньоарифметичне значення елементів і мінімальний //елемент

for (int i = 0; i < m; i++)

{

if (flag == false)

{

for (int j = 0; j < n; j++) // Проходимо згори вниз

{

if (i > j)

{

sum += \*(p + n \* j + i); // Накопичуємо суму

a++; // Збільшуємо кількість елементів

}

if (\*(p + n \* j + i) < x) // Знаходимо мінімальний елемент

{

x = \*(p + n \* j + i);

row = j;

column = i;

}

}

}

else

{

for (int j = n - 1; j >= 0; j--) // Проходимо знизу вгору

{

if (i > j)

{

sum += \*(p + n \* j + i); // Накопичуємо суму

a++; // Збільшуємо кількість елементів

}

if (\*(p + n \* j + i) < x) // Знаходимо мінімальний елемент

{

x = \*(p + n \* j + i);

row = j;

column = i;

}

}

}

flag = !flag; // Змінюємо на протилежне значення

}

sum = sum / a; // Середнє арифметичне значення

flag = false; // Змінюємо на протилежне значення

/\* Обходимо масив і змінюємо елемент над головною діагоналлю,

що більший за середньоарифметичне значення, на мінімальний елемент\*/

for (int i = 0; i < m; i++)

{

if (flag == false)

{

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

{

if (j < i)

{

if (\*(p + n \* j + i) > sum) // Перевіряємо елемент

{

\*(p + n \* j + i) = x; // Змінюємо його

}

}

}

}

else

{

for (int j = n - 1; j >= 0; j--)

{

if (j < i)

{

if (\*(p + n \* j + i) > sum) // Перевіряємо елемент

{

\*(p + n \* j + i) = x; // Змінюємо його

}

}

}

}

flag = !flag; // Змінюємо на протилежне значення

}

}

void Out\_answer(double\* p, int m, int n, double& x, int& row, int& column, double& sum)

{

printf("\n\nX = %.2f", x); // Виводимо мінімальний елемент

cout << " row = " << row + 1 << " column = " << column + 1; // Виводимо //його індекси

printf("\nSum = %.2f\n", sum); // Виводимо середньоарифметичне значення

cout << "\nAnswer: ";

// Виводимо масив

for (int i = 0; i < m; i++)

{

cout << "\n";

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

{

printf(" %.2f", \*(p + m \* i + j));

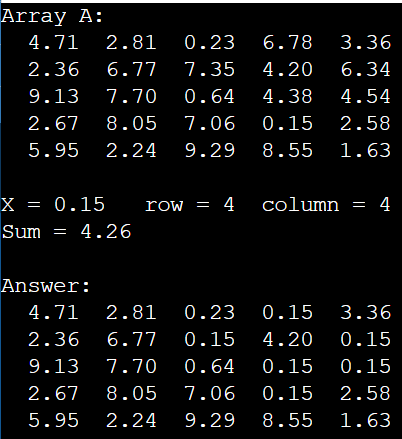
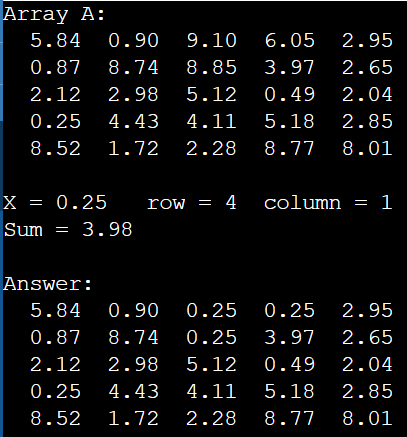
}

}

}

**Випробування алгоритму**

|  |  |
| --- | --- |
| Блок | Дія |
|  | Початок |
| 1 | Виклик Inp\_Out\_A(&A[0][0], m, n) |
|  |  |
|  | Inp\_Out\_A(double\* p, int m, int n) |
|  | Перша ітерація: |
| 2 | i = 0 |
| 3 | j = 0 |
| 4 | \*(p + m \* i + j) = 4.71 |
| 5 | Вивід 4.71 |
|  | Остання ітерація: |
| 6 | i = 4 |
| 7 | j = 4 |
| 8 | \*(p + m \* i + j) = 1.63 |
| 9 | Вивід 1.63 |
|  |  |
| 10 | Виклик Solution(&A[0][0], m, n, x, row, column, sum) |
|  | Solution(double\* p, int m, int n, double& x, int& row,  int& column, double& sum) |
|  | Перша ітерація: |
| 11 | i = 0 |
| 12 | j = 0 |
| 13 | sum = 4.71 |
| 14 | a = 1 |
| 15 | x = 4.71 |
| 16 | row = 0 column = 0 |
|  | Остання ітерація: |
| 17 | i = 4 |
| 18 | j = 4 |
| 19 | sum = 42.6 |
| 20 | a = 10 |
| 21 | x = 0.15 |
| 22 | row = 3 column = 3 |
|  |  |
| 23 | sum = 4.26 |
|  | Перша ітерація: |
| 24 | i = 0 |
| 25 | j = 0 |
| 26 | Поточний елемент не змінюється |
|  | Остання ітерація: |
| 27 | i = 4 |
| 28 | j = 4 |
| 29 | Поточний елемент не змінюється |
|  |  |
| 30 | Виклик Out\_answer(&A[0][0], m, n, x, row, column, sum) |
|  |  |
|  | Out\_answer(double\* p, int m, int n, double& x, int& row, int& column, double& sum) |
| 31 | Вивід 0.15 4 4 4.26 |
|  | Перша ітерація: |
| 32 | i = 0 |
| 33 | j = 0 |
| 34 | Вивід 4.71 |
|  | Остання ітерація: |
| 35 | i = 4 |
| 36 | j = 4 |
| 37 | Вивід 1.63 |
|  | Результат : 4.71 2.81 0.23 0.15 3.36  2.36 6.77 0.15 4.20 0.15  9.13 7.70 0.64 0.15 0.15  2.67 8.05 7.06 0.15 2.58  5.95 2.24 9.29 8.55 1.63 |
|  | Кінець |

**** ****

**Висновок –** я дослідив алгоритми обходу масивів (по стовпцях), набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Покращив свої навички у створенні блок-схем та псевдокоду та в написанні програм на мові С++.

У результаті виконання програми я одержав масив, у якому всі елементи вище головної діагоналі, які більші за середньоарифметичне значення змінюються на мінімальний елемент.