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

Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи № 9
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОБХОДУ МАСИВІВ»

Варіант 2

Виконав студент: ІП-13 Бабашев Олексій Дмитрович

Перевірив: Вечерковська Анастасія Сергіївна

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 2

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 1).

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Обчислення змінної, що описана в п.1, згідно з варіантом (табл. 1).

2	Задано матрицю дійсних чисел A[n,n]. Знайти суму елементів, розташованих вище головній діагоналі матриці.

1) **Постановка задачі:** Згенерувати квадратну матрицю дійсних елементів. Використовуючи алгоритм обходу матриці знайти суму елементів розташованих над головною діагоналлю.

2) **Побудова математичної моделі:**
Таблиця змінних:

Змінна	**Тип**	**Назва**	**Призначення**
Матриця	Дійсний	Matrix	Вихідне дане
Матриця повернена функцією	Дійсний	matrix	Проміжне дане
Одномірний масив(результат)	Дійсний	SUM	Вихідне дане
Змінна у функції для обчислення суми елементів над головною діагоналлю	Дійсний	res	Проміжне дане
Розмір матриці n на n	Цілий	n	Проміжне дане
Параметр функції розмір матриці n на n	Цілий	size	Проміжне дане
Лічильник	Цілий	i	Проміжне дане
Лічильник	Цілий	j	Проміжне дане

Таблиця функцій:

Назва	Синтаксис	Опис
initialize_matrix	initialize_matrix(size)	Функція для ініціалізації динамічної матриці(двомірного масиву) та заповнення її рандомними дійсними елементами.
print	print(matrix, size)	Функція для виводу матриці.
Sum	Sum(matrix, size)	Функція для алгоритма обходу матриці, та обчислення суми елементів розташованих над головною діагоналлю.
clear	clear(matrix, size)	Функція для очищення пам'яті

Математичне формулювання задачі зводиться до обчислення суми елементів розташованих над головною діагоналлю, знайдених за допомогою алгоритму обходу матриці по рядкам, реалізованого у функції **Sum**. Матриця ініціалізована та заповнена за допомогою функції **initialize_matrix.** Очищаємо пам'ять функцією **clear**.

Розв'язання:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо ініціалізацію матриці.

Крок 3. Деталізуємо обхід матриці та обчислення результату.

Крок 4. Деталізуємо очищення пам'яті.

Крок 5. Деталізуємо підпрограми.

3) Псевдокод:

Початок

Введення n

Matrix= **initialize_matrix**(n)

print(Matrix, n)

SUM = **Sum**(Matrix, n)

Вивести SUM

clear(Matrix, n)

Кінець

Підпрограми:

initialize_matrix(size)

Початок

Повторити (i = 0 , i<size , i++)

Повторити (j = 0 , j<size , j++)

matrix [i][j] = rand()

Все повторити

Все повторити

Повернути matrix

Кінець

print(matrix, size)

Початок

Повторити (i = 0 , i<size , i++)

Повторити (j = 0 , j<size , j++)

Вивести matrix[i][j]

Все повторити

Все повторити

Кінець

Sum(matrix, size)

Початок

res = o

Повторити (i = 0 , i<size , i++)

Повторити (j = 0 , j<size , j++)

Якщо (i<j)

res+=matrix[i][j]

Все якщо

Все повторити

Все повторити

Повернути res

Кінець

clear(matrix, size)

Початок

Повторити (i = 0 , i<size , i++)

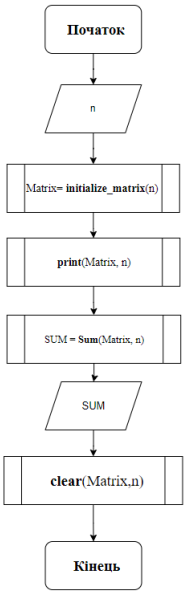
Видалити matrix[i]

Все повторити

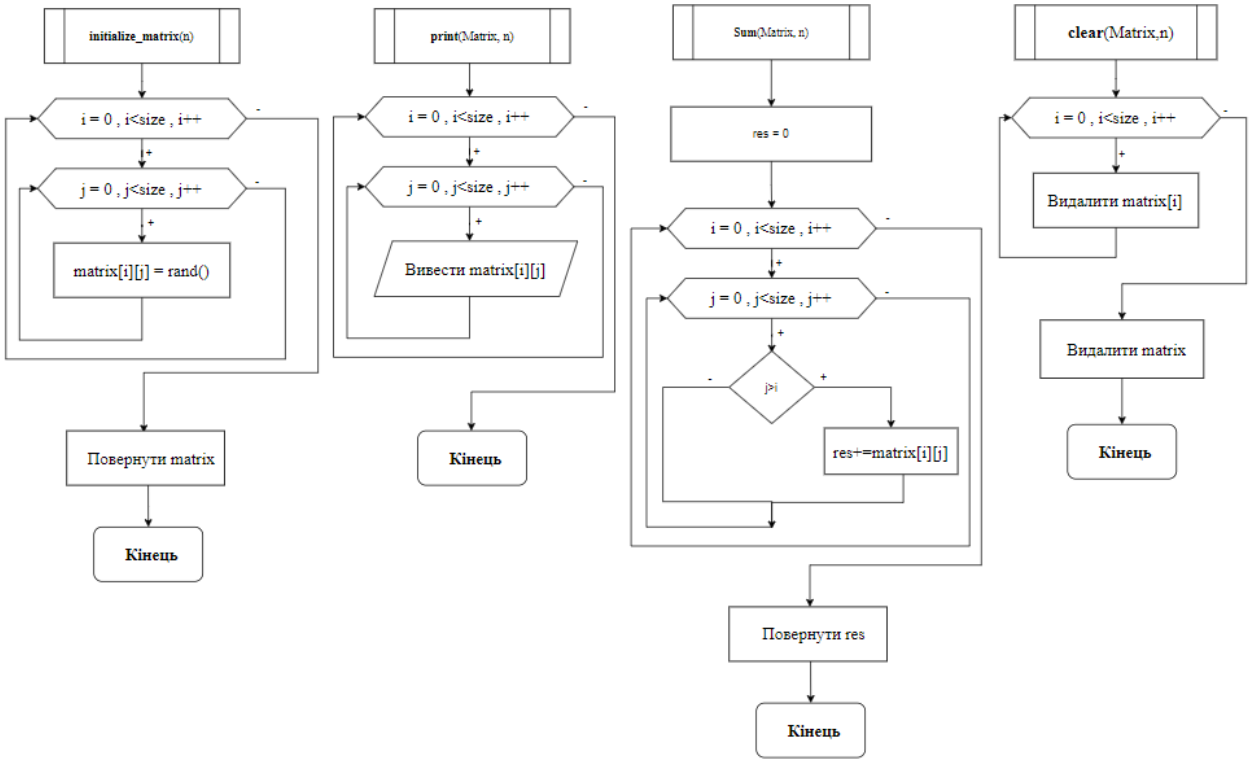
Видалити matrix

Кінець

4) Блок-схема:



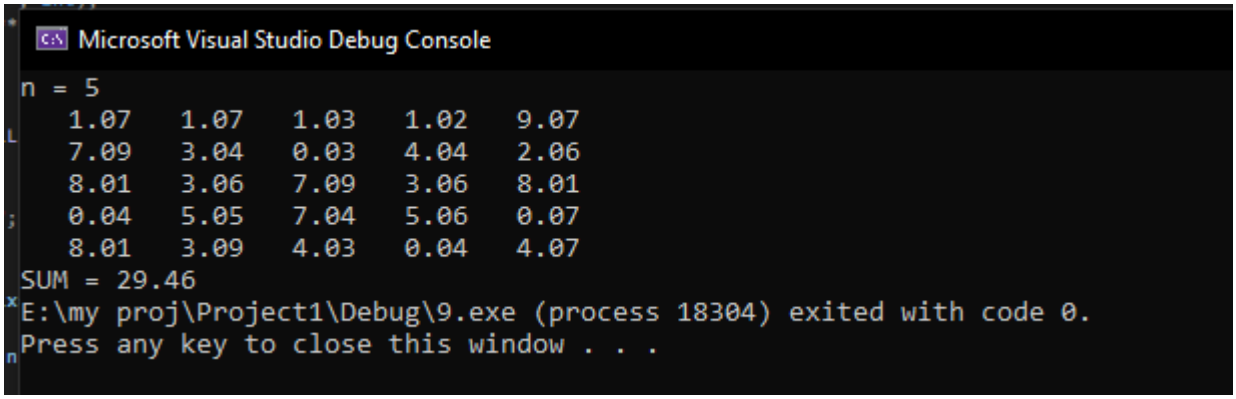
Підпрограми:

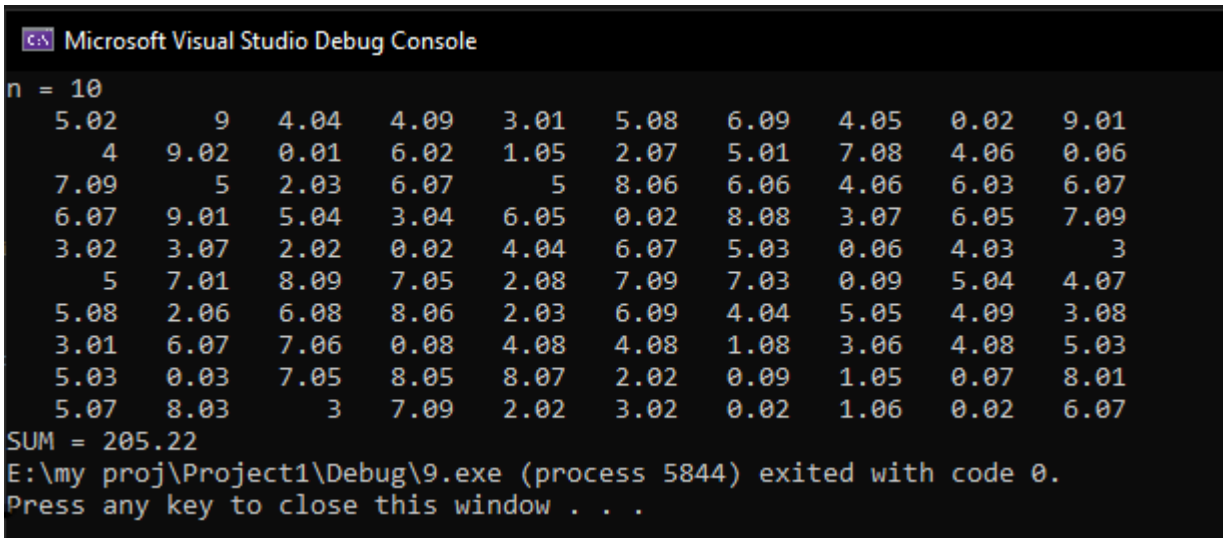


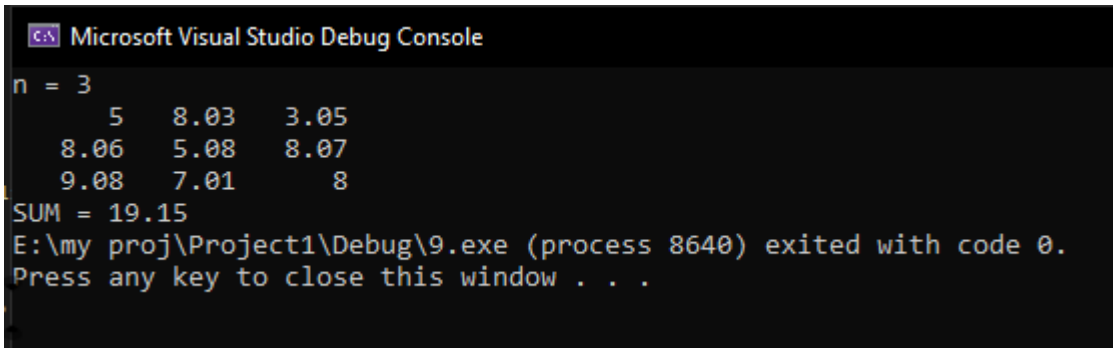
5) Код c++

```cpp
#include ...

using namespace std;

double** initialize_matrix(int);
void print(double**, int);
double Sum(double**, int);
void clear(double**, int);

int main()
{
    srand(time(NULL));

    int n;
    cout << "n = ";
    cin >> n;

    double** Matrix= initialize_matrix(n);

    print(Matrix, n);

    double SUM = Sum(Matrix, n);

    cout << "SUM = " << SUM;

    clear(Matrix, n);
}

double** initialize_matrix(int size) {
    double** matrix = new double* [size];
    for (int i = 0; i < size; i++) {
        matrix[i] = new double[size];
    }
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            matrix[i][j] = double(rand() % 10)/100.0 + (rand() % 10);
        }
    }
    return matrix;
}

void print(double** matrix, int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            cout << setw(7) << matrix[i][j] ;
        }
        cout << endl;
    }
}

double Sum(double** matrix, int size) {
    double res = 0;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            if (j > i) {
                res += matrix[i][j];
            }
        }
    }
    return res;
}


void clear(double** matrix, int size) {
    for (int i = 0; i < size; i++) {
        delete[] matrix[i];
    }
    delete matrix;
}
```

6) **Випробування алгоритму:**







7) **Висновок:** дослідив алгоритми обходу масивів, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій.