

Міністерство освіти і наука України  
Тернопільський національний технічний університет  
Імені Івана Пулюя

Кафедра комп'ютерних систем та  
мереж

Лабораторна робота № 2  
З дисципліни “Програмування мовою JAVA”  
на тему “Операції з даними та масивами”

Виконав(ла):  
студент групи СІс-21  
Підлатюк Денис Іванович

Перевірив(ла):  
Луцків Андрій Мирославович

Тернопіль 2025

## 2. Теоретичні відомості

### 18. Реалізувати множення вектора на матрицю.

Значення вектора і матриці заповнені псевдовипадковими числами з рівномірним розподілом у діапазоні від [0;1)

Розмірність вектора і матриці задається як аргумент командного рядка.

## 3. Виконання завдання

1. Необхідно зрозуміти, як саме необхідно множити для успішного виконання завдання. Матриці множиться так, як вказано на рисунку

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} + a_{14}b_{41}$$
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

$2 \times 4 \qquad 4 \times 3 \qquad 2 \times 3$

$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} + a_{24}b_{42}$$
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

Рисунок 1 - формула множення матриць

Для даної задачі нам потрібно помножити вектор на матрицю, вектор має мати значення  $1 \times n$ , а матриця -  $n \times m$ , якщо їх помножити, то результат -  $1 \times m$

2. Щоб реалізувати дану задачу на Java, нам необхідно скористатись масивами. Для зручності програму розділено на 2 частини - функцію множення та основну частину.

Код matrixmulti.java -

```
import java.util.Random;

/**
 * Клас для операцій над матрицями та
 векторами.
 */
public class matrixmulti {

    /**
     * Генерує вектор заданої довжини зі
     значеннями у діапазоні [0, 1).
     * @param length довжина вектора
     * @return масив з випадковими значеннями
     */
    public static double[]
generateRandomVector(int length) {
        double[] vector = new double[length];
        Random rand = new Random();
        for (int i = 0; i < length; i++) {
            vector[i] = rand.nextDouble();
        }
        return vector;
    }

    /**
     * Генерує матрицю розмірності n × m з
     випадковими значеннями [0, 1).
     * @param n кількість рядків
     * @param m кількість стовпців
     * @return двовимірний масив
     */
    public static double[][]
generateRandomMatrix(int n, int m) {
        double[][] matrix = new double[n][m];
        Random rand = new Random();
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
```

```

        matrix[i][j] =
rand.nextDouble();
    }
}
return matrix;
}

/**
 * Множить вектор (1×n) на матрицю (n×m),
результат – вектор (1×m).
 * @param vector вхідний вектор
 * @param matrix вхідна матриця
 * @return результат множення у вигляді
вектора
 */
public static double[]
multiplyVectorByMatrix(double[] vector,
double[][] matrix) {
    int m = matrix[0].length;
    double[] result = new double[m];

    for (int j = 0; j < m; j++) {
        for (int i = 0; i < vector.length;
i++) {
            result[j] += vector[i] *
matrix[i][j];
        }
    }
    return result;
}

/**
 * Виводить вектор у консоль.
 * @param vector вектор
 */
public static void printVector(double[]
vector) {
    for (double v : vector) {

```

```

        System.out.printf("%.4f ", v);
    }
    System.out.println();
}

/**
 * Виводить матрицю у консоль.
 * @param matrix матриця
 */
public static void printMatrix(double[][]
matrix) {
    for (double[] row : matrix) {
        for (double val : row) {
            System.out.printf("%.4f ", val);
        }
        System.out.println();
    }
}
}

```

#### Код mainmatrix.java

```

/**
 * Основний клас для запуску програми множення
 вектора на матрицю.
 * Аргументи командного рядка:
 * args[0] – розмірність вектора (n)
 * args[1] – кількість стовпців матриці (m)
 */
public class mainmatrix {
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Використання: java
mainmatrix <розмірність вектора n> <кількість
стовпців матриці m>");
            return;
        }
    }
}

```

```

        int n = Integer.parseInt(args[0]);
        int m = Integer.parseInt(args[1]);

        // Генерація вектора та матриці
        double[] vector =
matrixmulti.generateRandomVector(n);
        double[][] matrix =
matrixmulti.generateRandomMatrix(n, m);

        // Вивід вхідних даних
        System.out.println("Вектор:");
        matrixmulti.printVector(vector);

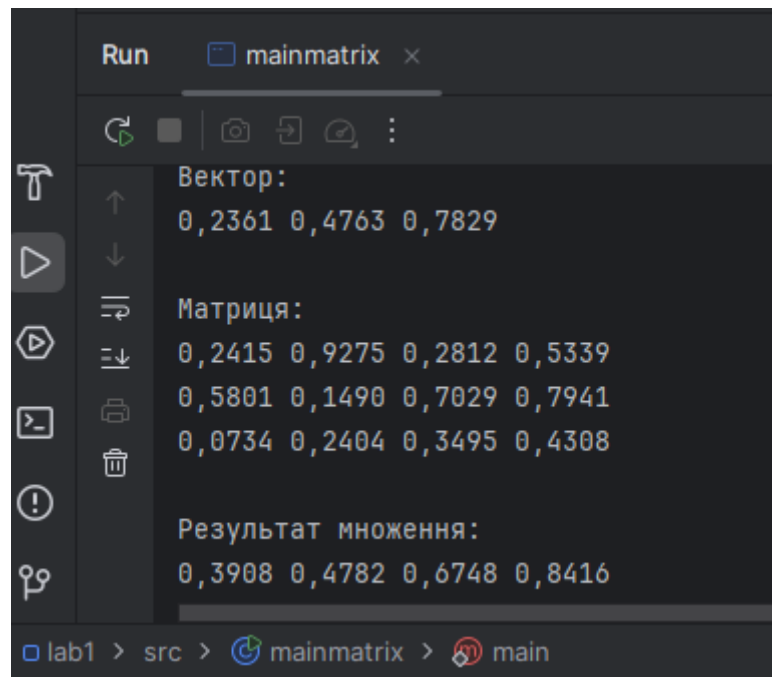
        System.out.println("\nМатриця:");
        matrixmulti.printMatrix(matrix);

        // Множення
        double[] result =
matrixmulti.multiplyVectorByMatrix(vector, matrix);

        // Результат
        System.out.println("\nРезультат множення:");
        matrixmulti.printVector(result);
    }
}

```

3. Для запуску програми необхідно вказати параметри перед запуском, в середовищі IDEA це вказується у вкладці run with parameters, у якому ми вводимо довжину вектора та розмір матриці.



```
Run  mainmatrix x
Вектор:
0,2361 0,4763 0,7829

Матриця:
0,2415 0,9275 0,2812 0,5339
0,5801 0,1490 0,7029 0,7941
0,0734 0,2404 0,3495 0,4308

Результат множення:
0,3908 0,4782 0,6748 0,8416

lab1 > src > mainmatrix > main
```

Рисунок 2 - Результат виконання

Кожен запуск програми буде мати різні значення, адже ми використали модуль random, який генерує псевдовипадкові числа у певному діапазоні.

#### 4. Контрольні запитання

1. Яка алгоритмічна складність реалізованого алгоритму?
2. Який складність по пам'яті реалізованого алгоритму?
3. MergeSort: у чому суть парадигми "розділяй-та-володарюй"?
4. Для комп'ютерних систем з малим обсягом оперативної пам'яті (вбудовані комп'ютери, мобільні комп'ютерні системи): який метод сортування є кращим і чому: selection sort, mergesort чи quicksort?

1. Алгоритмічна складність полягає у тому, що ми використовуємо  $n$  множень для кожного  $m$  стовбців.  
 $O(n*m)$
2.  $O(n+n*m+m)$   
Сумарно - лінійна відносно розміру вхідних даних

3. Парадигма "розділяй-та-володарюй" для mergesort полягає у:  
Розділенні масиву на дві частини, застосування рекурсивного сортування на обидві частини та об'єднання сортованих масивів в один.
4. Для таких систем найкращим рішенням буде selection sort, бо він має найменші вимоги до пам'яті, порівняно з іншими методами.

### Висновки

Отже, під час виконання даної лабораторної роботи було ознайомлено з базовими арифметичними операціями та структурою масивів в java.