Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

 «Магнитогорский государственный технический университет им. Г. И. Носова**»**

(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Кафедра вычислительной техники и программирования

Лабораторная работа №1

по дисциплине «Метрология и стандартизация программного обеспечения»

название лабораторной работы: «Оценка характеристик программы на основе лексического анализа»

Исполнитель: Сарынин А. В. студент 3 курса, группа АВб-22-1

Руководитель: Сибилева Н.С., ст. преподаватель каф. ВТиП

Магнитогорск, 2025

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc64291256)

[1 Постановка задания 4](#_Toc64291257)

[2 Выполнения задания 4](#_Toc64291258)

[Заключение 6](#_Toc64291259)

# Введение

Целью данной работы является анализ качества исходного кода программы для нахождения минимального положительного элемента матрицы и его индексов. В рамках исследования применяются метрики Холстеда, позволяющие оценить сложность кода через подсчет операторов, операндов и их взаимодействий. Такой анализ помогает выявить избыточность, потенциальные ошибки и возможности оптимизации, что особенно важно для повышения читаемости, поддерживаемости и эффективности программного обеспечения.

# Постановка задания

Необходимо реализовать задание на любом известном Вам языке программирования согласно варианту. Предусмотреть ввод необходимых значений с клавиатуры, генерацию случайных значений величин и вывод полученных решений на экран.

# Выполнения задания

Программа была написана языке программирования Java и предоставлена ниже.

package org.example;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ввод размеров матрицы

System.out.print("Введите количество строк m: ");

int m = scanner.nextInt();

System.out.print("Введите количество столбцов n: ");

int n = scanner.nextInt();

// Проверка корректности размеров

if (m <= 0 || n <= 0) {

System.out.println("Размеры матрицы должны быть положительными числами!");

return;

}

// Создание и заполнение матрицы

int[][] matrix = new int[m][n];

System.out.println("Введите элементы матрицы:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Поиск минимального положительного элемента

int minPositive = Integer.MAX\_VALUE;

int minRow = -1;

int minCol = -1;

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

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

int current = matrix[i][j];

if (current > 0 && current < minPositive) {

minPositive = current;

minRow = i;

minCol = j;

}

}

}

// Вывод результата

if (minRow != -1) {

System.out.println("Минимальный положительный элемент: " + minPositive);

System.out.println("Индексы: [" + minRow + "][" + minCol + "]");

} else {

System.out.println("В матрице нет положительных элементов.");

}

}

}

Результат выполнения данной программы представлен на рисунке 1.

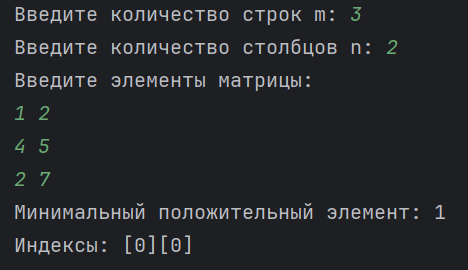


Рисунок 1 - Результат выполнения программы

В результате программа выводит минимальный положительный элемент и его индекс

Для расчета и анализа качества кода были собраны исходные данные, представленные в таблице 1 и таблице 2.

Таблица 1 – Операторы, операции и операнды программы.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Операнд | Номера строк | Количество повторений |
| 1 | scanner | 5, 9, 11, 24 | 4 |
| 2 | m | 9, 14, 20, 22, 33 | 5 |
| 3 | n | 11, 14, 20, 23, 33, 34 | 6 |
| 4 | matrix | 20, 24, 35 | 3 |
| 5 | minPositive | 29, 36, 37 | 3 |
| 6 | minRow | 30, 38, 45, 47 | 4 |
| 7 | minCol | 31, 39, 47 | 3 |
| 8 | current | 35, 36 | 2 |
| 9 | Integer.MAX\_VALUE | 29 | 1 |
| 10 | System.out | 8, 10, 15, 21, 46, 47, 49 | 7 |
| 11 | i, j | 22, 23, 33, 34, 35, 38, 39, 47 | 8 (по 4 на каждый) |
| Всего | 11 уникальных |  | 40 |

Таблица 2 – Операторы и операции

|  |  |  |  |
| --- | --- | --- | --- |
| № | Оператор/Операция | Номера строк | Количество повторений |
| 1 | import | 1 | 1 |
| 2 | public class | 3 | 1 |
| 3 | public static void | 4 | 1 |
| 4 | new | 5, 20 | 2 |
| 5 | if | 14, 45 | 2 |
| 6 | for | 22, 23, 33, 34 | 4 |
| 7 | int | 9, 11, 20, 29, 30, 31, 35 | 7 |
| 8 | return | 16 | 1 |
| 9 | = | 9, 11, 20, 24, 29, 30, 31, 35, 37, 38, 39 | 11 |
| 10 | && | 36 | 1 |
| 11 | <, > | 14, 36 | 2 (по 1 на каждый) |
| 12 | [] | 20, 24, 35 | 3 |
| 13 | . (вызов метода) | 5, 8, 10, 15, 21, 24, 46, 47, 49 | 9 |
| Всего | 13 уникальных |  | 43 |

Теперь необходимо рассчитать все необходимые характеристики, подставив имеющиеся данные. Результат расчета характеристик представлен в таблице 3.

Таблица 3 – Итоговая таблица показателей.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Характеристика | Обозначение и формула | Значение |
| 1 | Число уникальных операторов (n1) | - | 13 |
| 2 | Число уникальных операндов (n2) | - | 11 |
| 3 | Общее число операторов (N1) | - | 43 |
| 4 | Общее число операндов (N2) | - | 40 |
| 5 | Число входных/выходных переменных | n\*2 = 11 (вход: m, n, matrix; выход: minPositive, индексы) | 5 |
| 6 | Словарь программы (n) | n = n1 + n2 = 13 + 11 | 24 |
| 7 | Длина реализации (N) | N = N1 + N2 = 43 + 40 | 83 |
| 8 | Объем программы (V) | V = N \* log2(n) = 83 \* log2(24) | ≈ 83 \* 4.58 ≈ 380.14 |
| 9 | Потенциальный объем (V\*) | V\* = (n\*2 + 2) \* log2(n\*2 + 2) = (5 + 2) \* log2(7) | ≈ 7 \* 2.81 ≈ 19.67 |
| 10 | Уровень реализации (L) | L = V\* / V = 19.67 / 380.14 | ≈ 0.0517 |
| 11 | Уровень языка (λ) | λ = L \* V\* = 0.0517 \* 19.67 | ≈ 1.017 |
| 12 | Основные выводы | Низкий уровень реализации (L < 0.1) указывает на избыточность кода. Требуется рефакторинг для упрощения логики. | - |

# Заключение

Проведенный анализ на основе метрик Холстеда позволил оценить качество кода программы для поиска минимального положительного элемента матрицы. Основные выводы:

Низкий уровень реализации (L ≈ 0.0517) указывает на избыточность кода. Это связано с большим количеством операторов и операндов, что усложняет понимание логики.

Высокий объем программы (V ≈ 380.14) подтверждает необходимость рефакторинга для сокращения повторяющихся конструкций и упрощения структуры.

Рекомендации по оптимизации:

Уменьшение вложенных циклов за счет вынесения повторяющихся операций в отдельные методы.

Использование более лаконичных конструкций (например, потокового ввода/вывода).

Удаление избыточных переменных (например, объединение проверок условий).

Улучшение этих аспектов повысит уровень реализации программы, снизит вероятность ошибок и упростит её дальнейшее сопровождение. Применение метрик Холстеда доказало свою эффективность как инструмент для объективной оценки качества кода и формирования направлений его оптимизации.