

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

Исполнитель: студент группы БПИ197(1) Галкин Никита Сергеевич

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Задание №3. Вариант 5.

Пояснительная записка (отчет)

Москва 2020 г.

Задание №3

Листов 16

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Москва 2020 г.

Содержание

<u>1. ФОРМУЛИРОВКА ЗАДАНИЯ</u>	<u>4</u>
<u>2. ПРОЦЕДУРЫ ПРОГРАММЫ.....</u>	<u>5</u>
<u>3. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ</u>	<u>6</u>
<u>4. ТЕСТИРОВАНИЕ ПРОГРАММЫ.....</u>	<u>7</u>
<u>ПРИЛОЖЕНИЕ №1.....</u>	<u>9</u>
<u>ПРИЛОЖЕНИЕ №2.....</u>	<u>16</u>

1. ФОРМУЛИРОВКА ЗАДАНИЯ

Определить ранг матрицы. Входные данные: целое положительное число n , произвольная матрица A размерности $n \times n$. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков.

2. МЕТОДЫ ПРОГРАММЫ

Таблица 2.1

МЕТОДЫ	Назначение
main	Входная точка.
CheckOnDigit	Проверяет строку, можно ли ее превратить в число.
GetN	Принимает от пользователя положительное целое число.
GetSplittedElements	Проверяет, можно ли разделить строку на числа.
SetMatrix	Принимает от пользователя матрицу размером n на n.
DisplayMatrix	Выводит матрицу в консоль.
CheckOnZeroElems	Проверяет матрицу на наличие ненулевых элементов.
detOfTwo	Находит определитель матрицы размером 2 на 2.
CreateMatrix	Создает матрицу размером n на n.
Determinant	Находит определитель матрицы.
CheckMinors	Проверяет матрицу на наличие ненулевых миноров (определенного размера).
FindRank	Находит ранг матрицы.
min	Находит минимальный элемент среди двух предложенных.

3. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Для нахождения ранга матрицы используется метод полного перебора миноров. Начиная с минимального ранга матрицы (0) и заканчивая максимально возможным (размерность матрицы), проверяем матрицу на наличие неотрицательных миноров.

Программа проверяет: если все элементы матрицы равны 0, то ранг этой матрицы 0. Если же в матрице есть ненулевые элементы, то ранг минимум 1. Далее программа берет на проверку ранг 2. Если хоть 1 минор размерностью 2 в этой матрице ненулевой (его определитель ненулевой), то ранг 2 подтверждается и на проверку берется следующий ранг (и так до конца).

4. ТЕСТИРОВАНИЕ ПРОГРАММЫ

При запуске программы открывается консоль, появляется запрос на ввод размерности матрицы (n). (рис. 1).

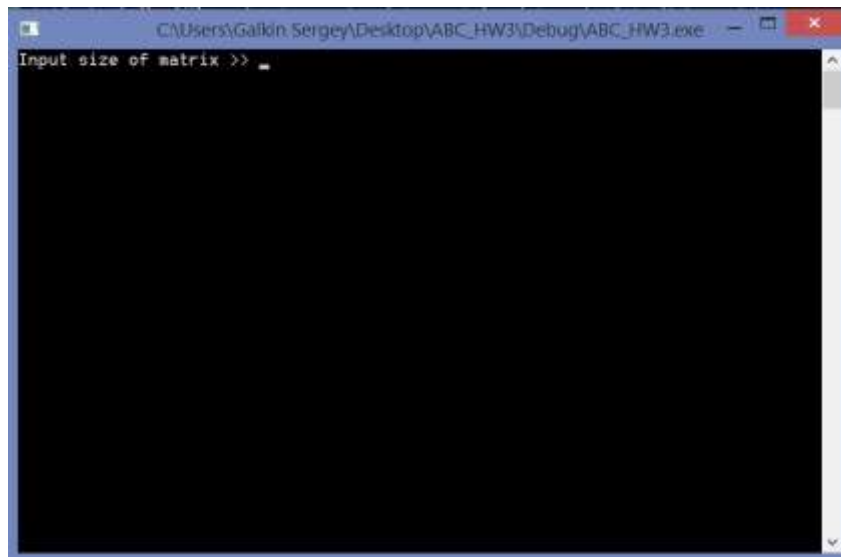


Рис. 1. Запуск программы

Далее необходимо вписать значение размерности матрицы и нажать Enter. После этого необходимо вводить данные матрицы: будет предложено ввести n строк, на каждой строке по n чисел через пробел (рис. 2).

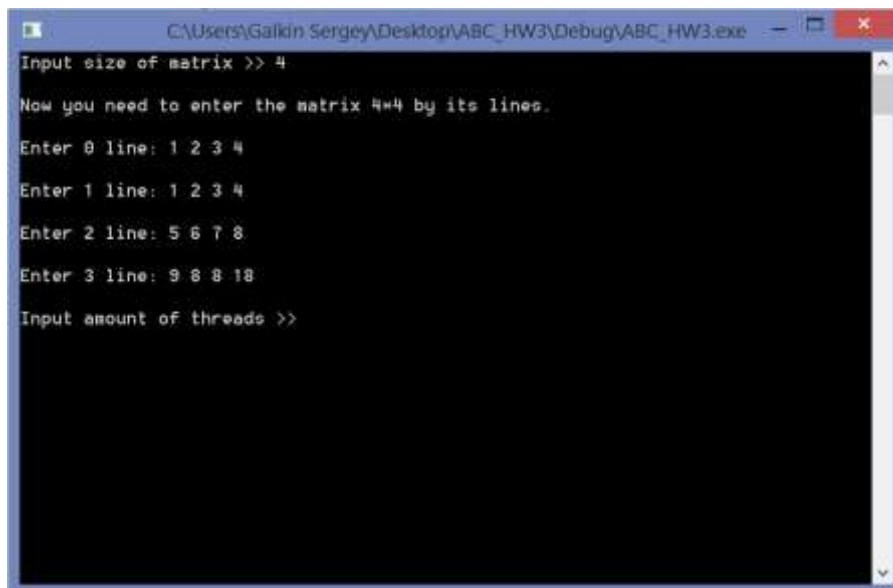
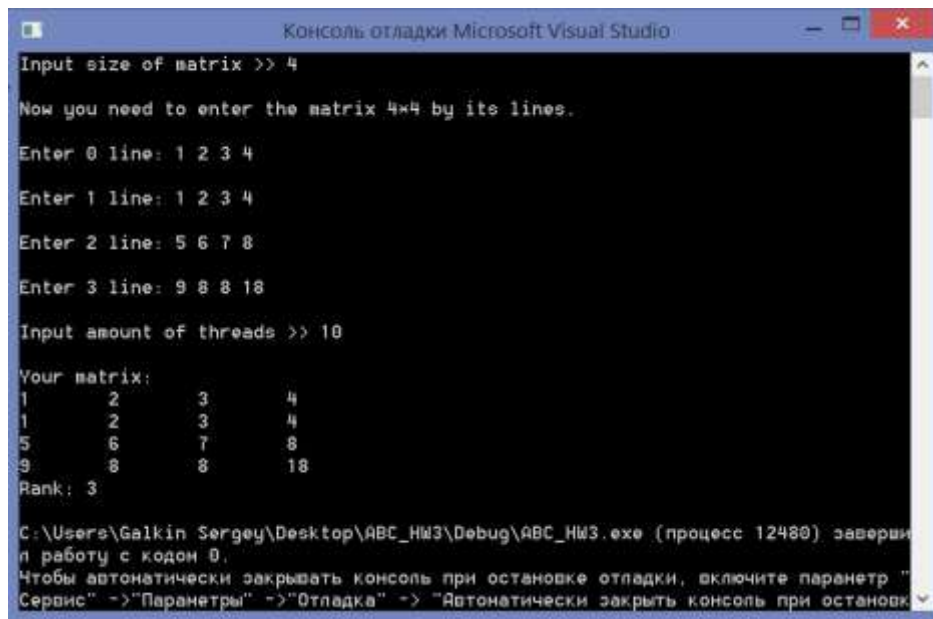


Рис. 2. Ввод данных

Далее необходимо ввести значение количества потоков и нажать Enter. Программа подсчитает результат, выведет матрицу и ответ (рис. 3).



```

Консоль отладки Microsoft Visual Studio

Input size of matrix >> 4

Now you need to enter the matrix 4x4 by its lines.

Enter 0 line: 1 2 3 4
Enter 1 line: 1 2 3 4
Enter 2 line: 5 6 7 8
Enter 3 line: 9 8 8 18

Input amount of threads >> 10

Your matrix:
1      2      3      4
1      2      3      4
5      6      7      8
9      8      8      18

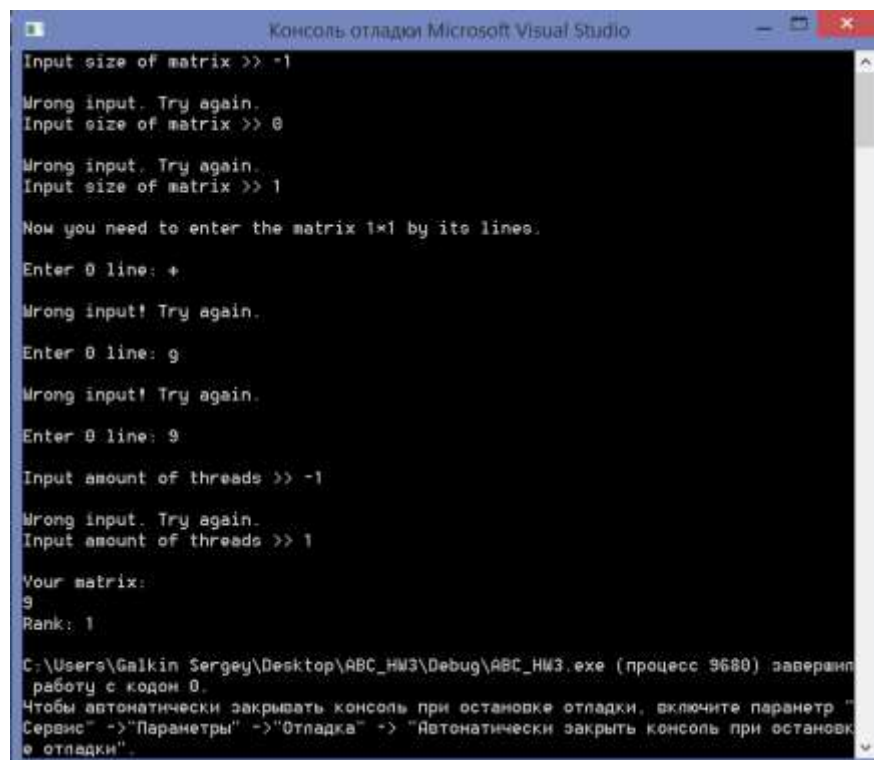
Rank: 3

C:\Users\Galkin Sergey\Desktop\ABC_HW3\Debug\ABC_HW3.exe (процесс 12480) завершил
работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "
Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остано

```

Рис. 3. Вывод ответа

В случае некорректных входных данных программа выдаст информацию об ошибке и попросит ввести данные заново (рис. 4).



```

Консоль отладки Microsoft Visual Studio

Input size of matrix >> -1

Wrong input. Try again.
Input size of matrix >> 0

Wrong input. Try again.
Input size of matrix >> 1

Now you need to enter the matrix 1x1 by its lines.

Enter 0 line: +

Wrong input! Try again.

Enter 0 line: 9

Wrong input! Try again.

Enter 0 line: 9

Input amount of threads >> -1

Wrong input. Try again.
Input amount of threads >> 1

Your matrix:
9

Rank: 1

C:\Users\Galkin Sergey\Desktop\ABC_HW3\Debug\ABC_HW3.exe (процесс 9680) завершил
работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "
Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остано

```

Рис. 4. Некорректный ввод

ТЕКСТ ПРОГРАММЫ

```

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <future>

using namespace std;

/// <summary>
/// Проверяет строку, можно ли ее превратить в число.
/// </summary>
/// <param name="line"> Строка. </param>
/// <returns> да / нет. </returns>
static bool CheckOnDigit(string line) {
    return count_if(line.begin(), line.end(), isdigit) == line.size();
}

/// <summary>
/// Принимает от пользователя положительное целое число.
/// </summary>
/// <returns> Число. </returns>
static int GetN(string message) {
    string line;
    int number;
    bool state = false;

    do {
        cout << message;

        getline(cin, line);
        cout << endl;

        // Проверка на то, что число.
        if (!CheckOnDigit(line))
            cout << "Wrong input. Try again." << endl;
        else {
            number = stoi(line);

            // Проверка на то, что число положительное.
            if (number > 0)
                state = true;
            else cout << "Wrong input. Try again." << endl;
        }
    } while (!state);

    return number;
}

/// <summary>
/// Проверяет, можно ли разделить строку на числа (через пробел).
/// </summary>
/// <param name="line"> Строка. </param>
/// <param name="elems"> Числа. </param>
/// <returns> Есть ошибка в строке (есть не число) или нет. </returns>
static bool GetSplittedElements(string line, vector<int>& elems) {
    string number = "";
    // Чистим вектор перед работой.

```

```

elems.clear();

for (char element : line) {
    // Разделителем является пробел.
    if (element != ' ') {
        // Проверка на то, что элемент строки - цифра.
        if (isdigit(element))
            number += element;
        else {
            cout << "Wrong input! Try again.\n" << endl;
            return false;
        }
    }
    else {
        // Если попадаете пробел, переносим собранное из цифр
число в вектор.
        elems.push_back(stoi(number));
        number = "";
    }
}

// Записываем последнее число (если не было в конце разделителя).
if (number != "")
    elems.push_back(stoi(number));

return true;
}

/// <summary>
/// Принимает от пользователя матрицу размером n x n.
/// </summary>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Матрица. </returns>
static vector<vector<int>>> SetMatrix(int n) {
    vector<vector<int>>> matrix;
    string line;
    bool state;
    vector<int> elems;

    cout << "Now you need to enter the matrix " << n << "*" << n << " by
its lines.\n" << endl;

    for (int i = 0; i < n; ++i) {
        state = false;

        do {
            cout << "Enter " << i << " line: ";
            getline(cin, line);
            cout << endl;

            // Проверка на то, что цифры.
            if (GetSplittedElements(line, elems)) {
                // Проверка на длину.
                if (elems.size() == n)
                    state = true;
                else cout << "Wrong length! Try again.\n" << endl;
            }
        } while (!state);

        matrix.push_back(elems);
    }
}

```

```

        elems.clear();
    }

    return matrix;
}

/// <summary>
/// Выводит матрицу в консоль.
/// </summary>
/// <param name="matrix"> Матрица. </param>
/// <param name="n"> Размерность матрицы. </param>
static void DisplayMatrix(vector<vector<int>> matrix, int n) {
    cout << "Your matrix:" << endl;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << matrix[i][j] << "\t";
        }
        cout << endl;
    }
}

/// <summary>
/// Проверяет матрицу на наличие ненулевых элементов.
/// </summary>
/// <param name="matrix"> Матрица. </param>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Есть ненулевые элементы или нет. </returns>
static bool CheckOnZeroElems(const vector<vector<int>>& matrix, int n) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (matrix[i][j] != 0)
                return true;
        }
    }

    return false;
}

/// <summary>
/// Находит определитель матрицы размером 2 на 2.
/// </summary>
/// <param name="matrix"> Матрица. </param>
/// <returns> Определитель. </returns>
static int detOfTwo(vector<vector<int>> matrix)
{
    return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
}

/// <summary>
/// Создает матрицу размером n на n.
/// </summary>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Матрица. </returns>
static vector<vector<int>> CreateMatrix(int n) {
    vector<vector<int>> newMatrix;
    vector<int> row;

    // Заполняем новую матрицу.
    for (int i = 0; i < n; ++i) {
        row.clear();
    }
}

```

```

        for (int j = 0; j < n; ++j)
            row.push_back(0);

        newMatrix.push_back(row);
    }

    return newMatrix;
}

/// <summary>
/// Находит определитель матрицы.
/// </summary>
/// <param name="matrix"> Матрица. </param>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Определитель. </returns>
static int Determinant(vector<vector<int>> matrix, int n)
{
    // Определитель.
    int det = 0;

    // Если длина стороны матрицы 2, то можно найти определитель.
    if (n == 2)
    {
        det += detOfTwo(matrix);
    }
    // Иначе рекурсия продолжается.
    else
    {
        // Новая меньшая матрица, определитель которой нужно найти.
        vector<vector<int>> newMatrix = CreateMatrix(n - 1);

        // Перебор всех миноров (раскладывая по правому столбцу).
        for (int i = 0; i < n; ++i) {
            // Образование минора.
            for (int k = 0; k < n; ++k) {
                for (int j = 0; j < n - 1; ++j) {
                    if (k < i)
                        newMatrix[k, j] = matrix[k, j];
                    else if (k > i)
                        newMatrix[k - 1, j] = matrix[k, j];
                }
            }

            // Увеличение / уменьшение определителя в зависимости от
            знака (чередуется).
            if ((i + n - 1) % 2 == 0)
                det += matrix[i][n - 1] * Determinant(newMatrix, n
- 1);
            else
                det -= matrix[i][n - 1] * Determinant(newMatrix, n
- 1);
        }

        return det;
    }
}

/// <summary>
/// Проверяет матрицу на наличие ненулевых миноров (определенного
размера).
/// </summary>

```

```

/// <param name="matrix"> Матрица. </param>
/// <param name="rank"> Проверяемый ранг (размер минора). </param>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Есть ненулевые миноры или нет. </returns>
static bool CheckMinors(const vector<vector<int>>& matrix, int rank, int
n) {
    vector<vector<int>> minor = CreateMatrix(rank);

    // Перебираем все миноры и находим их определитель.
    for (int i = 0; i <= (n - rank); ++i) {
        for (int j = 0; j <= (n - rank); ++j) {
            // Собираем минор в векторе.
            for (int p = i; p < (i + rank); ++p) {
                for (int k = j; k < (j + rank); ++k) {
                    minor[p - i][k - j] = matrix[p][k];
                }
            }
            // Если минор ненулевой.
            if (Determinant(minor, rank) != 0)
                return true;
        }
    }

    return false;
}

/// <summary>
/// Находит ранг матрицы.
/// </summary>
/// <param name="matrix"> Матрица. </param>
/// <param name="n"> Размерность матрицы. </param>
/// <returns> Ранг. </returns>
static int FindRank(const vector<vector<int>>& matrix, int start, int n) {
    int rank = 0;

    // Проходимся для каждого ранга.
    for (int currentRank = start; currentRank <= n; ++currentRank) {
        switch (currentRank) {
            case 1:
                // Проверка на ранг = 1.
                if (CheckOnZeroElems(matrix, n))
                    rank = 1;
                else {
                    rank = 0;
                    break;
                }
                break;
            default:
                // Проверка на все другие ранги.
                if (CheckMinors(matrix, currentRank, n))
                    rank = currentRank;
                else break;
        }
    }

    return rank;
}

/// <summary>
/// Нахождение минимального числа среди двух.
/// </summary>

```

```

/// <param name="a"> Первое число. </param>
/// <param name="b"> Второе число. </param>
/// <returns> Минимальное число. </returns>
static int min(int a, int b) {
    if (a < b) return a;
    else return b;
}

int main()
{
    srand(time(0));

    // Принимаем размерность матрицы.
    int n = GetN("Input size of matrix >> ");
    // Создаем матрицу и принимаем ее значения.
    vector<vector<int>> matrix = SetMatrix(n);

    // Принимаем от пользователя число потоков.
    int threads = GetN("Input amount of threads >> ");

    // Создаем вектор результатов с потоков.
    vector<future<int>> threadsVec;
    // Шаг.
    int shift = n / threads;

    for (size_t i = 0; i < threads - 1; i++)
    {
        threadsVec.push_back(async(FindRank, matrix, i * shift, min((i
+ 1) * shift, n)));
    }
    threadsVec.push_back(async(FindRank, matrix, (threads)*shift, n));

    // Выводим получившуюся матрицу.
    DisplayMatrix(matrix, n);

    // Находим ранг этой матрицы.
    int rank = 0;
    int counter = 0;
    int max = 0;

    if (shift == 0)
    {
        // Находим максимально получившийся ранг (это и есть
результат).
        for (auto& i : threadsVec) {
            rank = i.get();

            if (max <= rank)
                max = rank;
        }
        rank = max;
    }
    else {
        // Собираем результат.
        for (auto& i : threadsVec) {
            int curRank = i.get();
            if (counter == 0)
                if (curRank == 0)
                    break;

            if (rank > curRank)

```

```
        break;

        rank = curRank;
        counter++;
    }
}
// Выводим результат.
cout << "Rank: " << rank << endl;
}
```

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

- 1) Информация с сайта <http://www.softcraft.ru/>
- 2) Видеоуроки на платформе YouTube.
- 3) Сайт mathprofi.ru