

**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

**Задание №4. Вариант 5.**

**Пояснительная записка (отчет)**

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

**Москва 2020 г.**

**Задание №4**

**Листов 16**

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

**Москва 2020 г.**

## Содержание

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

**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).
FromFactorial	Находит число, факториал которого равен передаваемому параметру.
FindRank	Находит ранг матрицы.

### 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 ненулевых миноров.

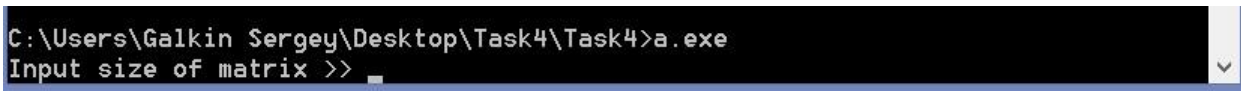
Программа проверяет: если все элементы матрицы равны 0, то ранг этой матрицы 0. Если же в матрице есть ненулевые элементы, то ранг минимум 1. Далее программа берет на проверку ранг 2. Если хоть 1 минор размерностью 2 в этой матрице ненулевой (его определитель ненулевой), то ранг 2 подтверждается и на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

При этом программа использует разделение по потокам с помощью OpenMP. Каждый поток занимается проверкой своего ранга, при этом все потоки суммируются в одну переменную. Далее, что очевидно, просто необходимо найти число, факториал которого равен результату. Это и есть конечный (максимально получившийся) ранг матрицы.

#### 4. ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для запуска программы с использованием OpenMP необходимо открыть cmd (Windows) в папке исходного файла и прописать “с++ Task4.cpp -fopenmp”, что скомпилирует исходный файл с использованием OpenMP. Далее, нужно прописать a.exe, что запустит исполняемый файл.

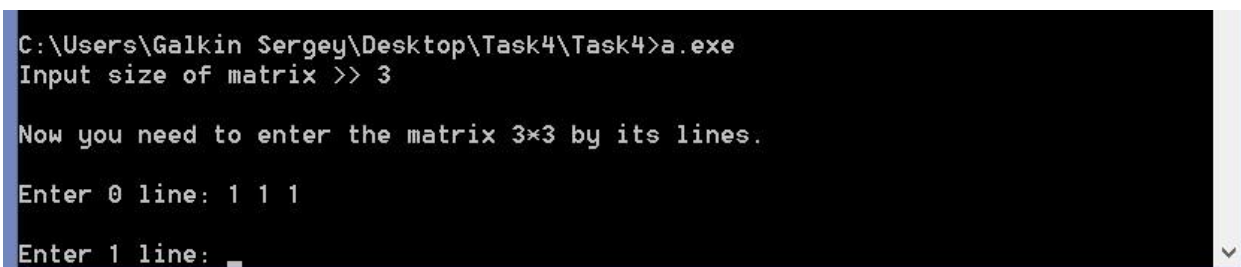
При запуске программы появляется запрос на ввод размерности матрицы (n). (рис. 1).



```
C:\Users\Galkin Sergey\Desktop\Task4\Task4>a.exe
Input size of matrix >> 
```

Рис. 1. Запуск программы

После ввода размерности необходимо ввести саму матрицу: будет предложено ввести n строк, на каждой строке по n чисел через пробел (рис. 2).



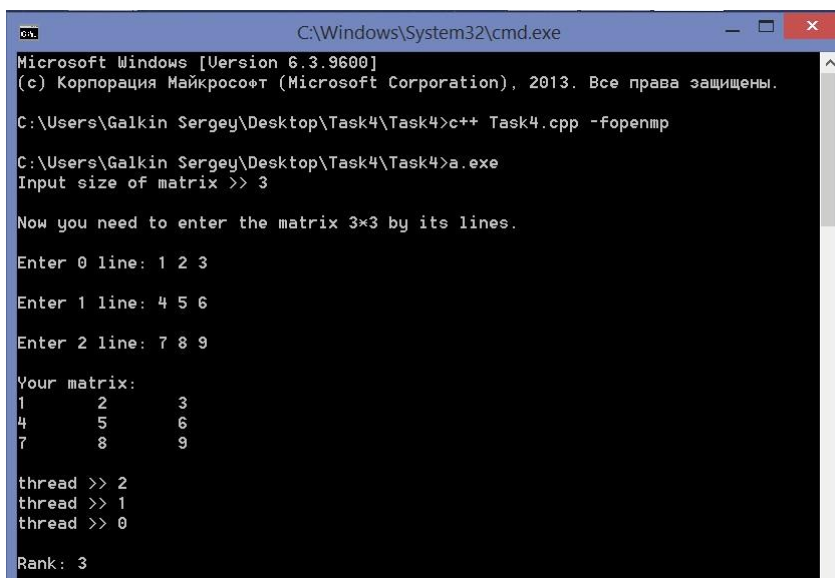
```
C:\Users\Galkin Sergey\Desktop\Task4\Task4>a.exe
Input size of matrix >> 3

Now you need to enter the matrix 3x3 by its lines.

Enter 0 line: 1 1 1
Enter 1 line: 
```

Рис. 2. Ввод данных

После ввода данных будут выведены: отформатированная матрица, информация о задействованных потоках и ранг матрицы (рис. 3).



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.

C:\Users\Galkin Sergey\Desktop\Task4\Task4>c++ Task4.cpp -fopenmp
C:\Users\Galkin Sergey\Desktop\Task4\Task4>a.exe
Input size of matrix >> 3

Now you need to enter the matrix 3x3 by its lines.

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

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

thread >> 2
thread >> 1
thread >> 0

Rank: 3
```

Рис. 3. Вывод ответа

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

```
C:\Users\Galkin Sergey\Desktop\Task4\Task4>a.exe
Input size of matrix >> -1

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

Wrong input. Try again.
Input size of matrix >> 2

Now you need to enter the matrix 2x2 by its lines.

Enter 0 line: e 1

Wrong input! Try again.

Enter 0 line: 1 1

Enter 1 line: 1 p2

Wrong input! Try again.

Enter 1 line: 1 1

Your matrix:
1      1
1      1

thread >> 0
thread >> 1

Rank: 1
```

Рис. 4. Некорректный ввод



## ТЕКСТ ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <omp.h>

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(), [](char a) {return a >= '0' && a <= '9'; }) ==
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 собравшееся 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 число, факториал которого равен передаваемому параметру.
/// </summary>
/// <param name="fact"> Факториал числа. </param>
/// <returns> Число. </returns>
static int FromFactorial(int fact) {
    int sumRes = 0;

    for (int i = 1; i < fact; ++i) {
        sumRes += i;

        if (sumRes == fact) return i;
    }

    return 1;
}

/// <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 n) {
    int sumRank = 0;

    #pragma omp parallel reduction (+: sumRank)
    {
        #pragma omp for
        // Проходимся для каждого ранга.
        for (int currentRank = 1; currentRank <= n; ++currentRank) {
            cout << "thread >> " << omp_get_thread_num() << endl;
            switch (currentRank) {
                case 1:
                    // Проверка на ранг = 1.
                    if (CheckOnZeroElems(matrix, n))
                        sumRank = 1;
                    else {
                        sumRank = 0;
                        break;
                    }
                    break;
                default:
                    // Проверка на все другие ранги.
                    if (CheckMinors(matrix, currentRank, n))
                        sumRank = currentRank;
            }
        }
    }
}

```

```

        else break;
    }
}

if (sumRank != 0)
    return FromFactorial(sumRank);
else return 0;
}

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

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

    // Находим ранг этой матрицы.
    int rank = FindRank(matrix, n);

    // Выводим результат.
    cout << "Rank: " << rank << endl;
}

```

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

- 1) Лекция
- 2) Информация с [сайта](#)
- 3) Видеоуроки на платформе YouTube.
- 4) Сайт [mathprofi.ru](http://mathprofi.ru)