**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное** **учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»**

Факультет компьютерных наук

Департамент программной инженерии

# **Отчет к домашнему заданию по дисциплине**

**«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ-195 Гуницкий Р.Я.

**Москва 2020**

СОДЕРЖАНИЕ

[1. ТЕКСТ ЗАДАНИЯ 3](#_Toc55996340)

[2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ 3](#_Toc55996341)

[3. ТЕСТИРОВАНИЕ ПРОГРАММЫ 4](#_Toc55996342)

[ИСТОЧНИКИ 5](#_Toc55996343)

[ПРИЛОЖЕНИЕ 1 6](#_Toc55996344)

[КОД ПРОГРАММЫ 6](#_Toc55996345)

1. ТЕКСТ ЗАДАНИЯ

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

1. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Для вычисления ранга матрицы, матрица приводится к ступенчатому виду методом Гауса, после чего подсчитывается количество ее ненулевых строк. В программе реализована парадигма итеративного параллелизма. Данный метод был выбран для ускорения вычислений новых строк матрицы путем реализации алгоритма Гаусса.

На вход программе подается размерность матрицы и количество потоков, которые будут работать над задачей.

Программа работает следующим образом: считывается размер матрицы и количество потоков, которые будут приводить ее к ступенчатому виду, после чего запускается цикл в котором находится первая строка с минимальным индексом ведущего элемента. Далее выбираются строки индекс ведущего элемента которых равен индексу ведущего элемента выбранной строки и из элементов этих строк вычитаются элементы выбранной строки, умноженные на ведущий элемент строки из которой производится вычитание и поделенный на ведущий элемент выбранной строки. Таким образом в итоге получается ступенчатая матрица и количество ее ненулевых строк равно ее рангу.

1. ТЕСТИРОВАНИЕ ПРОГРАММЫ

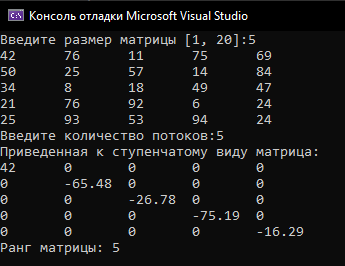


Рисунок 1 – Нахождение ранга матрицы 5 на 5 с 5 потоками

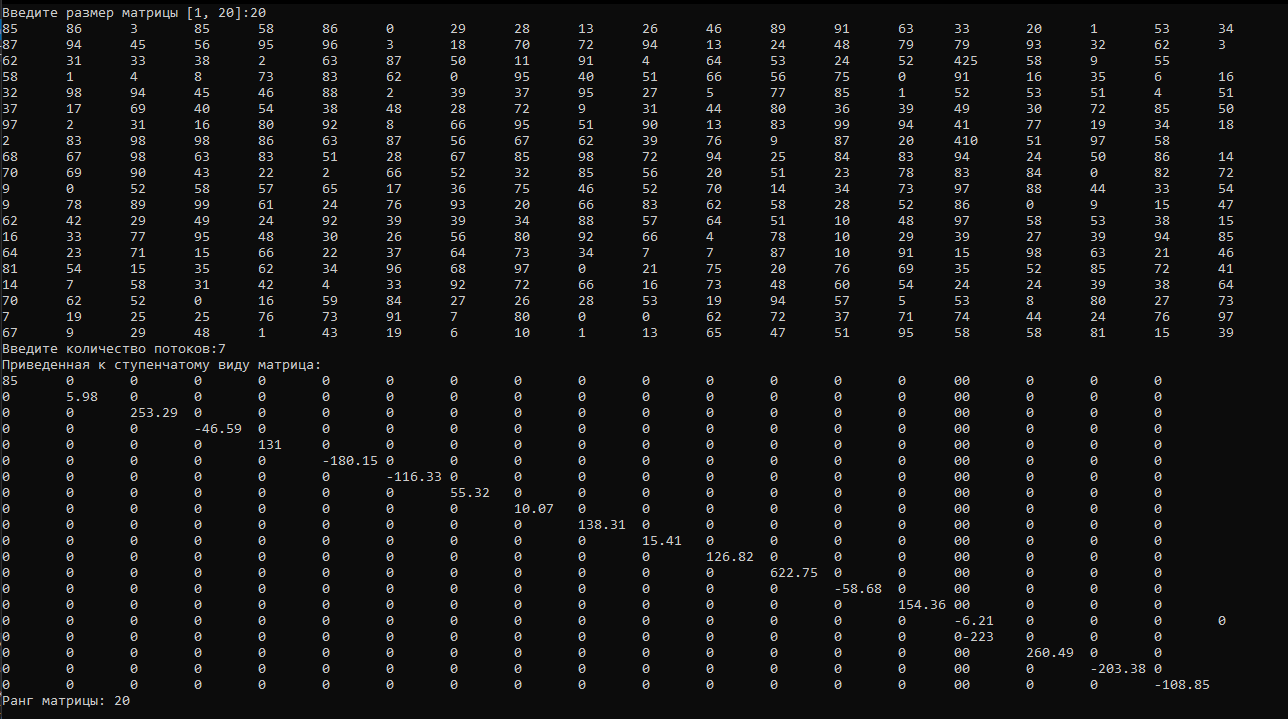


Рисунок 2 - Нахождение ранга матрицы 20 на 20 с 7 потоками

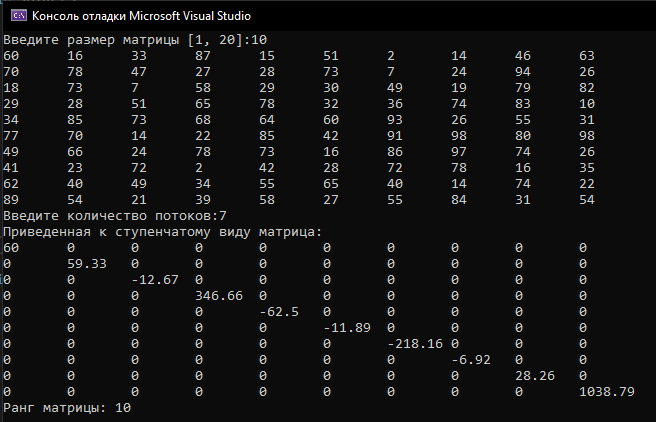


Рисунок 3 – Нахождение ранга матрицы 10 на 10 с 7 потоками

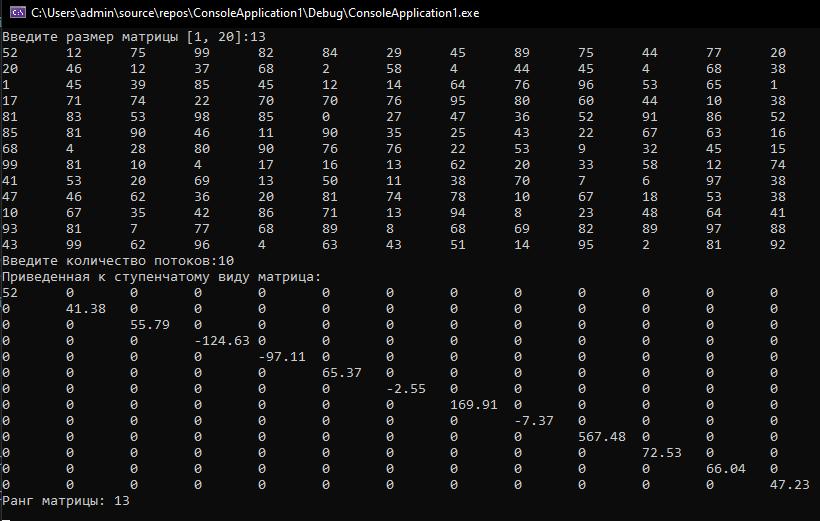


Рисунок 4 – Нахождение ранга матрицы 13 на 13 с 10 потоками

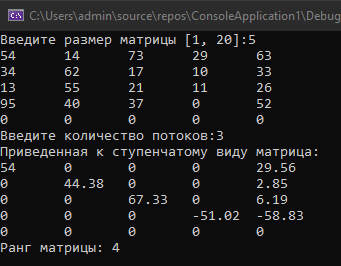


Рисунок 5 – Нахождение ранга матрицы 5 на 5 с одной нулевой строкой с 3 потоками

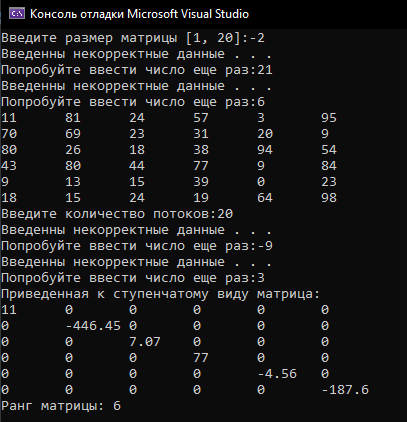


Рисунок 6 – ввод некорректных данных

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] [http://softcraft.ru/](http://softcraft.ru/%20)
2. Парадигмы параллельного программирования. [Электронный ресурс] [http://www.williamspublishing.com/PDF/5-8459-0388-2/part.pdf](http://www.williamspublishing.com/PDF/5-8459-0388-2/part.pdf%20)

3. Метод Гаусса. [Электронный ресурс] [https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%93%D0%B0%D1%83%D1%81%D1%81%D0%B0%20)

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

#include <iostream>

#include<vector>

#include<thread>

#include<mutex>

#include<string>

#include<ctime>

#include<cmath>

std::mutex mtx1;

/// <summary>

/// Выводит матрицу в консоль

/// </summary>

/// <param name="matrix"></param>

/// <param name="size"></param>

void printMatrix(double\*\* matrix, int size) {

for (size\_t i = 0; i < size; i++)

{

for (size\_t j = 0; j < size; j++)

{

std::cout << matrix[i][j] << "\t";

}

std::cout << std::endl;

}

}

/// <summary>

/// Создает случайную матрицу

/// </summary>

/// <param name="size"></param>

/// <returns></returns>

double\*\* createMatrix(int size) {

double\*\* matrix = new double\* [size];

for (size\_t i = 0; i < size; i++)

matrix[i] = new double[size];

for (size\_t i = 0; i < size; i++)

for (size\_t j = 0; j < size; j++) {

matrix[i][j] = rand() % 100;

}

return matrix;

}

/// <summary>

/// Удаляет матрицу

/// </summary>

/// <param name="matrix"></param>

/// <param name="sizeOfMatrix"></param>

void deleteMatrix(double\*\* matrix, int sizeOfMatrix) {

for (size\_t i = 0; i < sizeOfMatrix; i++)

{

delete[] matrix[i];

}

delete[] matrix;

}

/// <summary>

/// Считывает число

/// </summary>

/// <param name="num"></param>

/// <param name="minValue"></param>

/// <param name="maxValue"></param>

void ReadNumber(int& num, int minValue, int maxValue = INT\_MAX) {

std::cin >> num;

while (num < minValue || num > maxValue) {

std::cout << "Введенны некорректные данные . . ." << std::endl;

std::cout << "Попробуйте ввести число еще раз:";

std::cin >> num;

}

}

/// <summary>

/// Вычетает из эелементов строк, индекс ведущего эелемента которых равен

/// индексу ведущего элемента выбранной строки, эелементы выбранной строки

/// умноженные на matrix[i][firstElemIndex] и деленные на matrix[lineInd][firstElemIndex]

/// </summary>

/// <param name="matrix">матрица</param>

/// <param name="size">размер матрицы</param>

/// <param name="lineInd">индекс выбранной строки</param>

/// <param name="elemInd">индекс элемента в этой строке</param>

/// <param name="firstElemIndex">индекс ведущего элемента выбранной строки</param>

void SubLines(double\*\* matrix, int size, int lineInd, int elemInd, int firstElemIndex) {

for (size\_t i = 0; i < size; i++)

if (i != lineInd && matrix[i][firstElemIndex] != 0)

{

mtx1.lock();

double elem1 = matrix[lineInd][elemInd];

double elem2 = matrix[i][firstElemIndex];

double elem3 = matrix[lineInd][firstElemIndex];

matrix[i][elemInd] -= floor((elem1 \* elem2 / elem3) \* 100.0 + 0.5) / 100.0;

if (abs(matrix[i][elemInd]) < 0.00001)

matrix[i][elemInd] = 0;

mtx1.unlock();

}

}

void ThreadsSubs(double\*\* matrix, int size, int lineInd, int firstElemIndex, int startInd, int endInd) {

for (int i = endInd - 1; i >= startInd; i--) {

SubLines(matrix, size, lineInd, i, firstElemIndex);

}

}

/// <summary>

/// Находит индекс ведущего элемента строки

/// </summary>

/// <param name="line">строка</param>

/// <param name="size">размер матрицы</param>

/// <returns></returns>

int FindStartIndex(double\* line, int size) {

for (size\_t i = 0; i < size; i++)

if (line[i] != 0)

return i;

return -1;

}

/// <summary>

/// Находит ранг ступенчатой матрицы

/// </summary>

/// <param name="matrix"></param>

/// <param name="size"></param>

/// <returns></returns>

int Rg(double\*\* matrix, int size) {

int rg = size;

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

int countZeroes = 0;

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

if (matrix[i][j] == 0)

countZeroes++;

}

if (countZeroes == size)

rg--;

}

return rg;

}

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

std::cout << "Введите размер матрицы [1, 20]:";

int sizeOfMatrix;

ReadNumber(sizeOfMatrix, 1, 20);

double\*\* matrix = createMatrix(sizeOfMatrix);

printMatrix(matrix, sizeOfMatrix);

std::cout << "Введите количество потоков:";

int countThread;

ReadNumber(countThread, 1, sizeOfMatrix);

int maxIndex = -1;

for (size\_t i = 0; i < sizeOfMatrix; i++)

{

//Находим строку с минимальным индексом ведущего эелемента

//и схохраняем ее индекс и индекс ее ведущего элемента

int elemInd = sizeOfMatrix + 1; //индекс ведущего эелемента

int lineInd = sizeOfMatrix + 1; //индекс строки

for (size\_t j = 0; j < sizeOfMatrix; j++)

{

int startInd = FindStartIndex(matrix[j], sizeOfMatrix);

if (startInd > maxIndex && startInd < elemInd)

{

lineInd = j;

elemInd = startInd;

maxIndex = startInd;

}

}

//Если индекс ведущего элемента не изменился, то в матрице больше нет

//ненудевых строк, поэтому выходим из цикла

if (elemInd == sizeOfMatrix + 1)

break;

if (countThread > sizeOfMatrix - elemInd)

countThread = sizeOfMatrix - elemInd;

//Разбиваем задачу на потоки

int countColumnsForThread = sizeOfMatrix / countThread;

std::thread\* threads = new std::thread[countThread];

//Каждый поток занимается своей частью матрицы

for (size\_t i = 0; i < countThread; i++)

{

int startInd = i \* countColumnsForThread;

int endInd = i < countThread - 1 ? (i + 1) \* countThread : sizeOfMatrix;

threads[i] = std::thread(ThreadsSubs, matrix, sizeOfMatrix, lineInd, elemInd, startInd, endInd);

}

//Объединяем потоки и удаляем их

for (size\_t i = 0; i < countThread; i++)

threads[i].join();

delete[] threads;

}

std::cout << "Приведенная к ступенчатому виду матрица:" << std::endl;

printMatrix(matrix, sizeOfMatrix);

std::cout << "Ранг матрицы: " << Rg(matrix, sizeOfMatrix) << std::endl;

deleteMatrix(matrix, sizeOfMatrix);

}