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

**Факультет компьютерных наук  
Департамент программной инженерии**

**НАХОЖДЕНИЕ РАНГА МАТРИЦЫ(вариант 5)**

**Отчёт**

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:  
студент группы БПИ198  
Баранов Г. А.

**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.

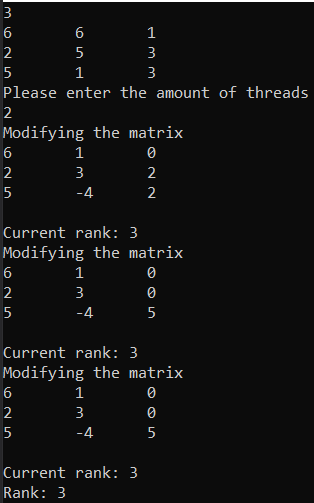
1. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Так как ранг матрицы в приведенной программе вычисляется методом Гаусса [3], то программа работает с помощью итеративного параллелизма [2]. Данный метод был выбран для ускорения вычислений новых строк матрицы путем реализации алгоритма Гаусса.

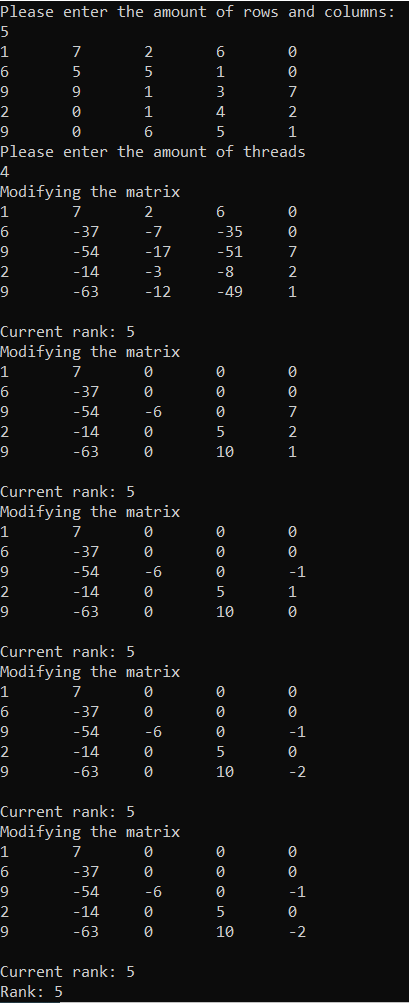
На вход программе подается размерность матрицы и количество потоков, которые будут работать над задачей.

Программа работает следующим образом: она получает данные о размерности матрицы, генерирует ее из случайных элементов от 0 до 9. Далее вызывается метод, который проходит по строкам матрицы и ищет такие, что ведущий элемент в них не равен 0, и они до этого не были использованы. После чего следующие за ведущим элементом элементы делятся на ведущий, а строка прибавляется к остальным с коэффициентом равным ведущему элементу текущей строки. Таким образом выполняется метод Гаусса [3].

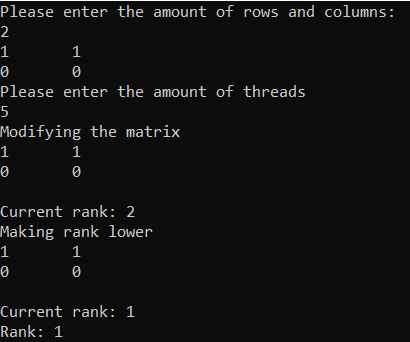
1. ТЕСТИРОВАНИЕ ПРОГРАММЫ

**

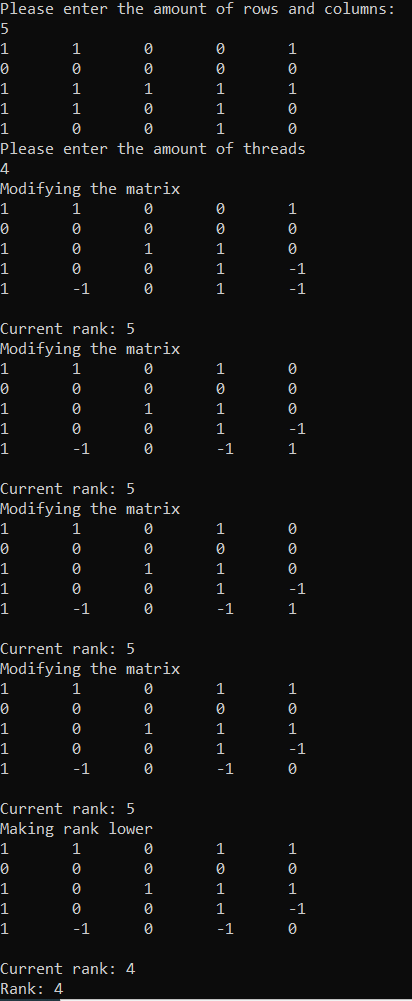
*Рисунок 1 – Нахождение ранга невырожденной матрицы 3 на 3*

**

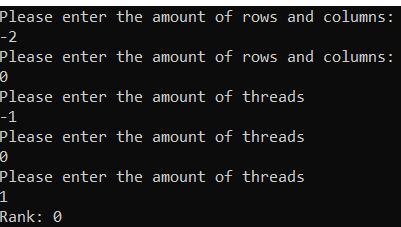
*Рисунок 2 - Нахождение ранга невырожденной матрицы 5 на 5*

**

*Рисунок 3 – Нахождение ранга вырожденной матрицы 2 на 2*

**

*Рисунок 4 – Нахождение ранга матрицы 5 на 5 с нулевой строкой*

**

*Рисунок 5 – ввод некорректных данных*

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] http://softcraft.ru/ (дата обращения: 10.11.2020).
2. Парадигмы параллельного программирования. [Электронный ресурс] http://www.williamspublishing.com/PDF/5-8459-0388-2/part.pdf (дата обращения: 12.11.2020).

3. Метод Гаусса. [Электронный ресурс] <https://e-maxx.ru/algo/linear_systems_gauss> (дата обращения 14.11.2020)

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

#include <iostream>

#include<vector>

#include<thread>

#include<mutex>

#include<string>

#include<ctime>

#include<omp.h>

using namespace std;

//Создание матрицы рандомом с остатком деления на 10

double\*\* createMatrix(int N) {

double\*\* Matrix = new double\* [N];

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

{

Matrix[i] = new double[N];

}

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

{

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

{

Matrix[i][j] = rand() % 10;

cout << Matrix[i][j] << "\t";

}

cout << endl;

}

return Matrix;

}

void print\_matrix(double\* A[], int N) {

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

for (size\_t g = 0; g < N; g++)

{

cout << (int)A[f][g] << "\t";

}

cout << endl;

}

cout << endl;

}

void substract\_el(int iThread, int thrNum, int N, double\* A[], int j, int i, double EPS) {

for (int k = iThread; k < N; k += thrNum) { //Для каждого потока будет свой цикл

if (k != j && A[k][i] != EPS) //Если это не j-ая строка и ведущий элемент ненулевой

for (int p = i + 1; p < N; ++p) // отнимаем j-ую строку от остальных

A[k][p] -= A[j][p] \* A[k][i];

}

}

void rang\_mat(int thrNum, vector<bool> line\_used, double\* A[], int N, int &rank)

{

const double EPS = 0;

int rang = N;

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

int j;

for (j = 0; j < N; ++j) {

if (!line\_used[j] && A[j][i] != EPS) { //Проверка что текущая строка матрицы не использована и что элемент больше 0

break;

}

}

if (j == N) {

cout << "Making rank lower" << endl;

--rang;

}

else{

int k;

line\_used[j] = true; //запонимнаем выбранную строку

for (int p = i + 1; p < N; ++p)

A[j][p] /= A[j][i]; //Делим все элементы строки на первый ненулевой элемент

cout << "Modifying the matrix" << endl;

#pragma omp parallel for private(k) num\_threads(thrNum)

for (k = 0; k < thrNum; ++k)

{

//cout << "Launching thread " << k << endl;

substract\_el(k, thrNum, N, ref(A), j , i, EPS);

}

}

print\_matrix(A, N);

cout << "Current rank: " << rang << endl;

}

rank = rang;

}

int main()

{

//Делаем, чтобы при разных тестах были разные значения рандома

srand(time(NULL));

int N;

do {

cout << "Please enter the amount of rows and columns: " << endl;

cin >> N;

} while (N < 0);

double\*\* Matrix = createMatrix(N); //Создаем матрицу

int threadNumber;

do {

cout << "Please enter the amount of threads" << endl;

cin >> threadNumber; //Узнаем число потоков

} while (threadNumber <= 0);

vector<int> rang = vector<int>(threadNumber);

vector<bool> line\_used(N);

int rank = N;

rang\_mat(threadNumber, ref(line\_used), Matrix, N, ref(rank)); //Считаем ранг

cout << "Rank: " << rank;

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

{

delete[] Matrix[i];

}

delete[] Matrix; //Удаление матрицы из памяти

}