

Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

Департамент

Программная инженерия

Самостоятельная работа по дисциплине

«Архитектура вычислительных систем»

Тема работы: Разработать программу с применением библиотеки OpenMP. 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.

Выполнил: студент группы БПИ191(1) Бен Мустафа Анас Риадович.

Преподаватель: Легалов Александр Иванович

Вариант 5

Разработать программу с применением библиотеки OpenMP. 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.

1. Описание принципа построения работы программы.

Для решения данной задачи использовалась модель построения многопоточных приложений, называемая **итеративный параллелизм**, используемый для реализации нескольких потоков (часто идентичных), каждый из которых содержит циклы. Потоки программы, описываются итеративными функциями и работают совместно над решением одной задачи.

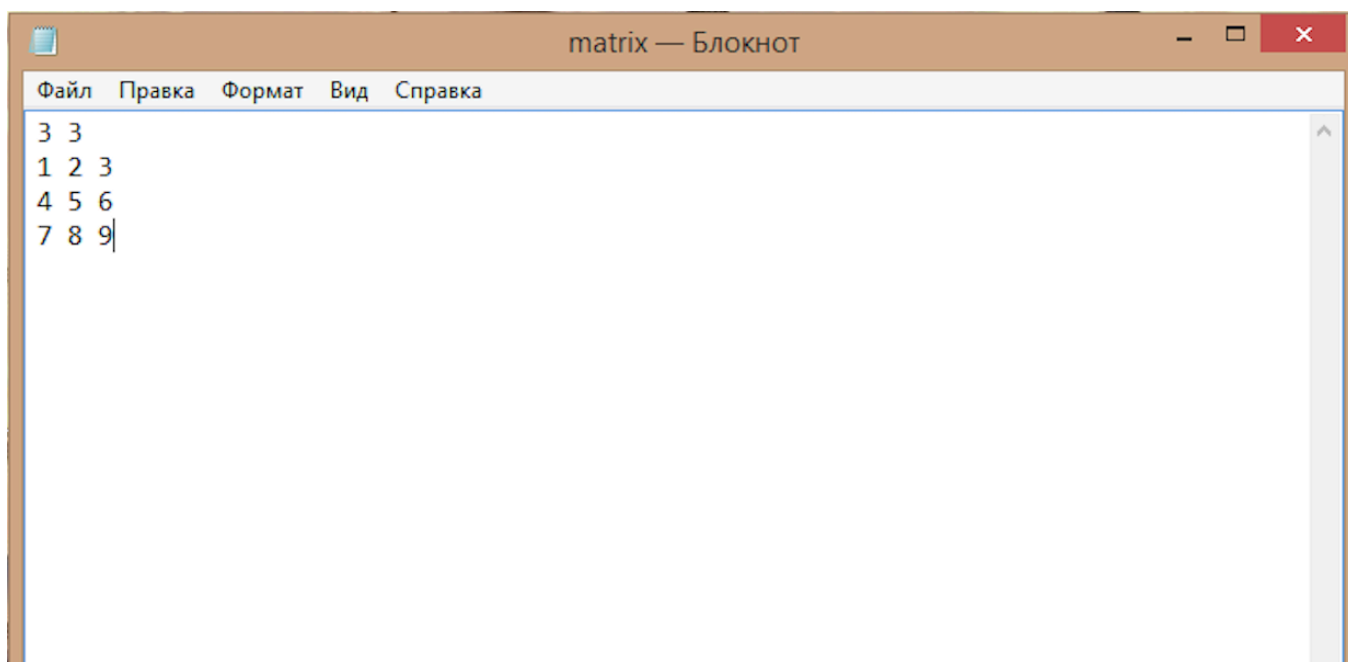
Вычисления ранга матрицы были проведены с помощью модифицированной версии метода Гаусса (наиболее простой способ с точки зрения программной реализации).

Информация по алгоритму взята с ресурса: https://studopedia.ru/21_129494_modifitsirovannyi-metod-gaussa.html

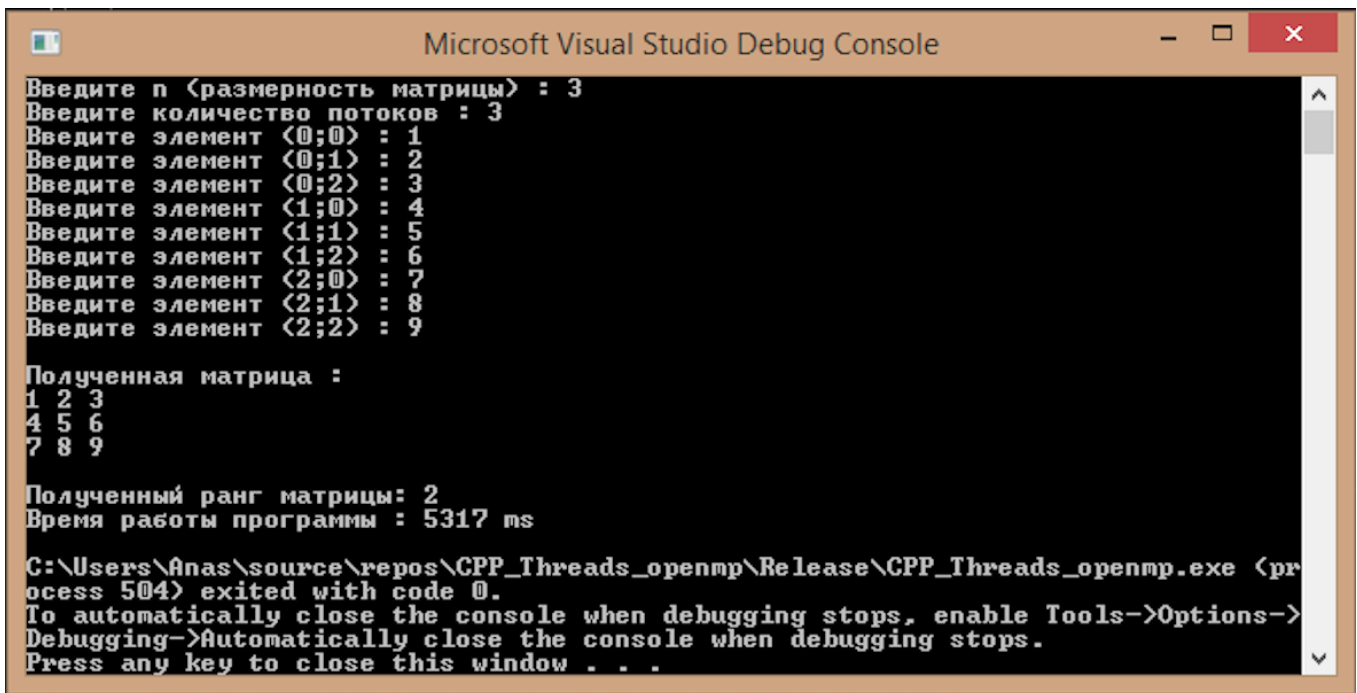
2. Описание входных данных.

Для удобства использования программы входные данные могут быть представлены в двух видах:

1. Данные могут быть переданы в программу через текстовый файл в следующем формате, где левый параметр первой строки - размерность матрицы, правый параметр первой строки - количество используемых потоков. Затем матрица размера $n \times n$.



2. Данные могут быть переданы в программу пользователем вручную.



```
Microsoft Visual Studio Debug Console

Введите n (размерность матрицы) : 3
Введите количество потоков : 3
Введите элемент <0;0> : 1
Введите элемент <0;1> : 2
Введите элемент <0;2> : 3
Введите элемент <1;0> : 4
Введите элемент <1;1> : 5
Введите элемент <1;2> : 6
Введите элемент <2;0> : 7
Введите элемент <2;1> : 8
Введите элемент <2;2> : 9

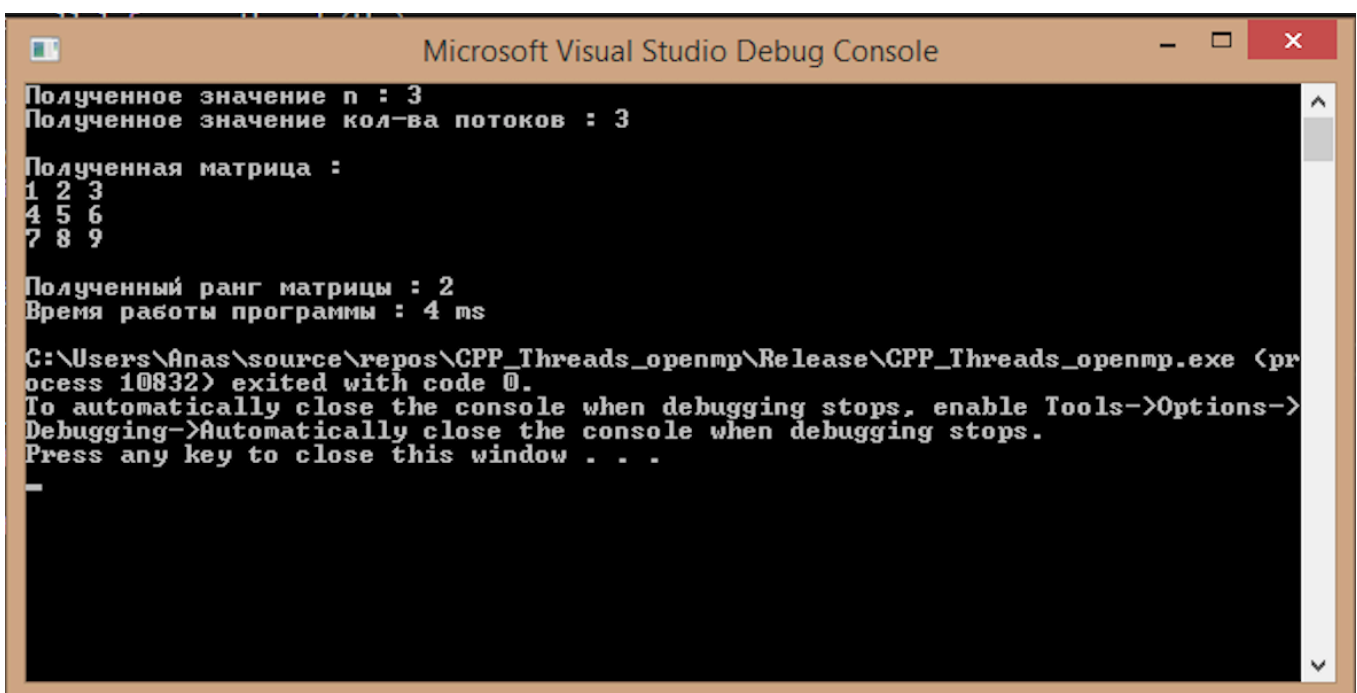
Полученная матрица :
1 2 3
4 5 6
7 8 9

Полученный ранг матрицы: 2
Время работы программы : 5317 ms

C:\Users\Anas\source\repos\CPP_Threads_openmp\Release\CPP_Threads_openmp.exe (process 504) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

3. Описание выходных данных.

Вне зависимости от типа входных данных (вручную или текстовым документом), в качестве выходных данных программы пользователь получает исходную матрицу, посчитанный ранг исходной матрицы, а также итоговое время работы программы. Алгоритм обрабатывает практически мгновенно. (0ms на фото ниже - не ошибка).



```
Microsoft Visual Studio Debug Console

Полученное значение n : 3
Полученное значение кол-ва потоков : 3

Полученная матрица :
1 2 3
4 5 6
7 8 9

Полученный ранг матрицы : 2
Время работы программы : 4 ms

C:\Users\Anas\source\repos\CPP_Threads_openmp\Release\CPP_Threads_openmp.exe (process 10832) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Список используемых источников

1. <https://pro-prof.com/archives/4335>
2. <https://habr.com/ru/company/intel/blog/85273/>
3. http://edu.mmcs.sfedu.ru/pluginfile.php/48306/mod_resource/content/1/OpenMP.pdf

Полный текст программы

```
main.cpp x
1  #include <iostream>
2  #include <omp.h>
3  #include <vector>
4  #include <fstream>
5  #include <thread>
6  #include <chrono>
7  #include <locale> // Обязательно для функции setlocale()
8
9  using namespace std;
10
11 int n, quantityOfThreads;
12 thread* threads;
13 vector<bool> usedMatrixLines;
14 int rankOfMatrix;
15 int** matrix;
16 const double EPS = 1E-9;
17
18 void inputMatrix() {
19     int elem;
20
21     matrix = new int* [n]; // Инициализация матрицы
22     for (int i = 0; i < n; i++) {
23         matrix[i] = new int[n];
24     }
25
26     for (int i = 0; i < n; ++i) {
27         for (int j = 0; j < n; ++j) {
28             cout << "Введите элемент (" << i << " " << j << " ) : ";
29             cin >> elem;
30             matrix[i][j] = elem;
31         }
32     }
33 }
34
35 void printMatrix() {
36     std::cout << endl << "Полученная матрица : " << endl;
37
38     for (int i = 0; i < n; i++) {
39         for (int j = 0; j < n; j++) {
40             std::cout << matrix[i][j] << " ";
41         }
42         std::cout << endl;
43     }
44 }
45
46 void GetMatrixFromFile(string pathToFile) {
47     ifstream in(pathToFile);
48     in >> n >> quantityOfThreads;
49
50     cout << "Полученное значение n : " << n << endl << "Полученное значение количества потоков : " << quantityOfThreads << endl;
51     rankOfMatrix = n;
52     usedMatrixLines = vector<bool>(n);
53
54     matrix = new int* [n];
55
56     for (int i = 0; i < n; i++) {
57         matrix[i] = new int[n];
58     }
59
60     for (int i = 0; i < n; ++i) {
```

```

61         for (int j = 0; j < n; ++j) {
62             in >> matrix[i][j];
63         }
64     }
65 }
66
67 void compute_rank() {
68     vector<bool> row_selected(n, false);
69
70     // Одна прагма, добавленная перед циклом, сделает всю работу за нас.
71     // Цикл будет выполняться в quantityOfThreads разных потоках.
72     // При этом параметры цикла будут автоматически распределены между всеми потоками.
73     #pragma omp parallel for num_threads(ths)
74     for (int i = 0; i < n; ++i) {
75         int j;
76
77         for (j = 0; j < n; ++j)
78             if (!row_selected[j] && abs(matrix[j][i]) > EPS)
79                 break;
80
81         if (j == n)
82             --rankOfMatrix;
83
84         else {
85             row_selected[j] = true;
86             for (int p = i + 1; p < n; ++p)
87                 matrix[j][p] /= matrix[j][i];
88             for (int k = 0; k < n; ++k)
89                 if (k != j && abs(matrix[k][i]) > EPS)
90                     for (int p = i + 1; p < n; ++p)
91                         matrix[k][p] -= matrix[j][p] * matrix[k][i];
92         }
93     }
94 }
95
96 int main(int argc, char* argv[]) {
97     setlocale(LC_ALL, "Russian"); // Для локализации
98
99     // Начинаем отсчет времени со старта работы программы
100     auto begin = std::chrono::steady_clock::now();
101
102     // В случае, если пользователь не передает никаких параметров, делаем ручной ввод
103     if (argc == 1) {
104         cout << "Введите n (размерность матрицы) : ";
105         cin >> n;
106         cout << "Введите количество потоков : ";
107         cin >> quantityOfThreads;
108
109         if (quantityOfThreads > n || n < 1 || quantityOfThreads < 1) {
110             cout << endl << "Некорректный ввод!" << endl <<
111                 "1. Количество потоков должно быть <= размерности матрицы." << endl <<
112                 "2. Размерность матрицы должна быть >= 1." << endl <<
113                 "3. Количество потоков должно быть >= 1." << endl << endl << endl << endl << endl;
114             return 0;
115         }
116     }
117
118     usedMatrixLines = vector<bool>(n);
119     rankOfMatrix = n;
120 }

```

```

97 int main(int argc, char* argv[]) {
98
99     setlocale(LC_ALL, "Russian"); // Для локализации
100
101     // Начинаем отсчет времени со старта работы программы
102     auto begin = chrono::steady_clock::now();
103
104     // В случае, если пользователь не передает никаких параметров, делаем ручной ввод
105     if (argc == 1) {
106         cout << "Введите n (размерность матрицы) : ";
107         cin >> n;
108         cout << "Введите количество потоков : ";
109         cin >> quantityOfThreads;
110
111         if (quantityOfThreads > n || n < 1 || quantityOfThreads < 1) {
112             cout << endl << "Некорректный ввод!" << endl <<
113                 "1. Количество потоков должно быть <= размерности матрицы." << endl <<
114                 "2. Размерность матрицы должна быть >= 1." << endl <<
115                 "3. Количество потоков должно быть >= 1." << endl << endl << endl << endl << endl;
116             return 0;
117         }
118
119         usedMatrixLines = vector<bool>(n);
120         rankOfMatrix = n;
121
122         inputMatrix(); // ввод пользователем матрицы
123         printMatrix(); // вывод матрицы на экран
124         compute_rank(); // функция подсчета ранга матрицы
125
126         cout << endl << "Полученный ранг матрицы: " << rankOfMatrix << endl; // Вывод результатов работы программы
127
128         auto end = chrono::steady_clock::now();
129         auto elapsed_ms = chrono::duration_cast<std::chrono::milliseconds>(end - begin);
130         cout << "Время работы программы : " << elapsed_ms.count() << " ms\n";
131     }
132     else if (argc == 2) { // Если передан файл с матрицей – считываем его
133
134         GetMatrixFromFile(argv[1]);
135
136         if (quantityOfThreads > n || n < 1 || quantityOfThreads < 1) {
137             cout << endl << "Некорректный ввод!" << endl <<
138                 "1. Количество потоков должно быть <= размерности матрицы." << endl <<
139                 "2. Размерность матрицы должна быть >= 1." << endl <<
140                 "3. Количество потоков должно быть >= 1." << endl << endl << endl << endl << endl;
141             return 0;
142         }
143
144         rankOfMatrix = n;
145         printMatrix(); // вывод полученной матрицы на экран
146         compute_rank(); // функция подсчета ранга матрицы
147
148         cout << endl << "Полученный ранг матрицы : " << rankOfMatrix << endl; // Вывод результатов работы программы
149
150         auto end = chrono::steady_clock::now();
151         auto elapsed_ms = chrono::duration_cast<std::chrono::milliseconds>(end - begin);
152         cout << "Время работы программы : " << elapsed_ms.count() << " ms\n";
153     }
154
155     return 0;
156 }

```