ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

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

Архитектура вычислительных систем
Домашнее задание №3, вариант 4

ПРОГРАММА ДЛЯ ВЫЧИСЛЕНИЯ ОБРАТНОЙ МАТРИЦЫ С ИСПОЛЬЗОВАНИЕМ ПОТОКОВ

Пояснительная записка

Исполнитель:
студент группы БПИ191
В.И. Беловицкий/

2020 г.

СОДЕРЖАНИЕ

1. ПОСТАВЛЕННАЯ ЗАДАЧА

Найти обратную матрицу для матрицы A. Входные данные: целое положительное число n, произвольная матрица A размерности n x n. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков.

2. ИСПОЛЬЗОВАННЫЙ АЛГОРИТМ

Для поиска обратной матрицы используется **метод Жордана-Гаусса** [1]. К исходной матрице справа присоединяется единичная матрица. В результате работы алгоритма на месте единичной матрицы будет располагаться обратная матрица (а на месте исходной – единичная).

В основном цикле проходим по столбцам исходной матрицы, спускаясь по диагонали. Если на месте, где должен располагаться ведущий элемент находится нулевой элемент, ищем в столбце ниже строку с ненулевым элементом, меняем две строки местами. Если ненулевой элемент не найден, значит определитель матрицы равен 0.

Затем, во вложенном цикле обновляем строки матрицы: если это строка с текущим ведущим элементом, то делим все ее элементы на ведущий (так, чтобы ведущий стал равен 1), вычитаем из строки линейную комбинацию строки с ведущим элементом (так, чтобы элементы под ведущим элементом стали равны 0). Данный цикл разбиваем на потоки.

В качестве многопоточной архитектуры был выбран **итеративный параллелизм** [2] – разбиваем массив строк на несколько частей, каждую часть обрабатывает отдельный, независимый поток.

3. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

В качестве аргумента при запуске программы передается путь до файла с тестовыми данными и путь до файла для записи ответа.

Пример запуска программы:

```
ThreadApp.exe C:\Users\User\FCS\ABC\ThreadHomework1\Debug\test1.txt C:\Users\User\FCS\ABC\ThreadHomework1\Debug\answer1.txt
```

На первой строчке приведено количество потоков, используемых при обработке строк матрицы. На второй строке приведена размерность матрицы n. Далее расположены n строк по n чисел (тип чисел – float).

Пример входных данных в тестовом файле:

```
2
3
1 2 3
2 5 4
0 1 0.5
```

Обратная матрица записывается в файл с ответом. Помимо этого, если размерность матрицы меньше или равна 15, то исходная и обратная матрицы, а также информация о запуске потоков отображаются в консоли.

4. ТЕСТИРОВАНИЕ

Проверим программу на файле test1.txt:

```
Initial matrix:
1  2  3
2  5  4
0  1  0.5

Column #1
Thread # 1 from 1 to 1
Thread # 2 from 2 to 3
Column #2
Thread # 1 from 1 to 1
Thread # 2 from 2 to 3
Column #3
Thread # 1 from 1 to 1
Thread # 2 from 2 to 3

Inverse matrix:
-0.6  0.8  -2.8
-0.4  0.2  0.8
0.8  -0.4  0.4
```

На входе матрица 3x3, доступно 2 потока. Для каждого ведущего элемента обновляем строки матрицы. Первый поток будет обрабатывать 1 строку, второй 2 строки (1 + 1 остаточная).

Проверим на файле test2.txt. Дана матрица 4x4, 2 потока.

```
Initial matrix:
2  1  0  0
3  2  0  0
1  1  3  4
2  -1  2  3

Column #1
Thread # 1 from 1 to 2
Thread # 2 from 3 to 4
Column #2
Thread # 1 from 1 to 2
Thread # 2 from 3 to 4
Column #3
Thread # 1 from 1 to 2
Thread # 2 from 3 to 4
Column #4
Thread # 1 from 1 to 2
Thread # 2 from 3 to 4

Inverse matrix:
2  -1  0  0
-3  2  0  0
31  -19  3  -4
-23  14  -2  3
```

Проверим на файле test3.txt. Матрица 4x4, 3 потока. Но присутствуют два одинаковых столбца, т.е. определитель равен 0. Это выясняется на последней операции, когда не нашлось последнего ведущего элемента (не равного 0).

```
Initial matrix:
2  1  0  0
3  2  0  0
3  2  1  3
2  -1  1  3

Column #1
Thread # 1 from 1 to 1
Thread # 2 from 2 to 2
Thread # 3 from 3 to 4
Column #2
Thread # 1 from 1 to 1
Thread # 2 from 2 to 2
Thread # 3 from 3 to 4
Column #3
Thread # 1 from 1 to 1
Thread # 2 from 2 to 2
Thread # 3 from 3 to 4
Column #4

Error: matrix determinant equals 0
```

Протестируем программу на файле test4.txt. Это случайно сгенерированная матрица 100x100 из чисел от 0 до 999. Результат можно увидеть в приложенном файле answer4.txt.

5. ЛИТЕРАТУРА

1. Метод Жордана-Гаусса [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Метод_Гаусса_—_Жордана (дата обращения: 15.11.2020).

2. Практические приемы построения многопоточных приложений [Электронный ресурс]. URL: http://softcraft.ru/edu/comparch/tasks/t03/ (дата обращения: 18.11.2020).

6. ТЕКСТ ПРОГРАММЫ

```
#include <iostream>
#include <iomanip>
#include <thread>
#include <fstream>
#include <sstream>
#include <vector>
#include <mutex>
#include <string>
using namespace std;

mutex block;

vector<vector<float>> readInput(const std::string& input,  unsigned int* n, unsigned int*
threadNum) {
    ifstream in(input);
    unsigned int thN, N;
    in >> thN >> N;

    *n = N;
    *threadNum = thN;

    vector<vector<float>> matrix(N, vector<float>(N * 2));

    // считываем исходную матрицу
    float num;
    for (unsigned int i = 0; i < N; i++) {
        for (unsigned int j = 0; j < N; j++) {
            in >> num;
            matrix[i][j] = num;
        }
    }

    // заполняем единичную матрицу
    for (unsigned int i = 0; i < N; i++) {
        for (unsigned int j = 0; j < N; j++) {
            if (i == j) {
                matrix[i][j + N] = 1;
            }
            else {
                matrix[i][j + N] = 0;
            }
        }
    }
    return matrix;
}

void writeOutput(const std::string& output, vector<vector<float>> matrix, unsigned int n) {
    std::ofstream out(output);

    for (unsigned int i = 0; i < n; i++) {
        for (unsigned int j = 0; j < n; j++) {
            out << matrix[i][j] << " ";
        }
        out << endl;
    }
}


static void computeRow(vector<vector<float>>& matrix, unsigned int n, unsigned int j, unsigned
int from, unsigned int to) {
```

```
    for (unsigned int i = from; i < to; i++) {
        if (i == j) { // если строка с текущим ведущим элементом, то делим строку на ведущий
элемент
            float lead = matrix[i][j]; // текущий ведущий элементом
            for (unsigned int k = j; k < 2 * n; k++) { // начинаем с j, т.к. до j элементы равны
0
                block.lock();
                matrix[i][k] /= lead;
                block.unlock();
            }
        }
        else { // иначе вычитаем из строки линейную комбинацию строки с ведущим элементом
            float lead = matrix[i][j]; // элемент строки с ведущим элементом
            for (unsigned int k = j; k < 2 * n; k++) {
                block.lock();
                matrix[i][k] -= matrix[j][k] * lead / matrix[j][j];
                block.unlock();
            }
        }
    }
}


void GaussJordan(vector<vector<float>> &matrix, unsigned int n, unsigned int threadNum) {
    // Определяем количество строк, обрабатываемых в одном потоке
    unsigned int rowsPerThread;
    // В последнем потоке может быть задействовано больше строк (т.к. n может быть не кратно
threadNum)
    unsigned int rowsInLastThread;

    if (threadNum > n) {
        // если количество доступных потоков больше размерности, то используем только n потоков
        threadNum = n;
        rowsPerThread = 1;
        cout << "Number of available threads is greater than matrix dimension. " + to_string(n) +
" threads will be used";
    }
    else {
        rowsPerThread = n / threadNum;
    }
    rowsInLastThread = rowsPerThread + n % threadNum;

    vector<thread> threads(threadNum);

    unsigned int temp;
    for (unsigned int j = 0; j < n; j++) {
        cout << "Column #" + to_string(j + 1) << endl;
        temp = j; // индекс строки, в которой должен быть ведущий элемент (i==j)
        if (matrix[temp][j] == 0) { // если на месте ведущего элемента 0, то ищем ненулевой
элемент, находящийся ниже в столбце
            for (unsigned int i = j + 1; i < n; i++) {
                if (matrix[i][j] != 0) {
                    temp = i;
                    break;
                }
            }
            if (matrix[temp][j] == 0) {
                cout << endl << "Error: matrix determinant equals 0" << endl;
                exit(1);
            }
            else {
                // меняем текущую строку со строкой с найденным ненулевым элементом
                for (unsigned int k = j; k < 2 * n; k++) {
                    float tempEl = matrix[temp][k];
```

```
                    matrix[temp][k] = matrix[j][k];
                    matrix[j][k] = tempEl;
                }
            }
        }

        // запускаем потоки, преобразующие строки
        unsigned int from = 0;
        for (unsigned int thI = 0; thI < threadNum; thI++) {
            unsigned int to = from + (thI == threadNum - 1 ? rowsInLastThread : rowsPerThread);
            cout << "Thread # " + to_string(thI + 1) + " from " + to_string(from + 1) + " to " +
to_string(to) << endl;
            threads[thI] = thread(computeRow, ref(matrix), n, j, from, to);
            from += rowsPerThread;
        }

        // Дожидаемся выполнения всех потоков
        for (unsigned int thI = 0; thI < threadNum; thI++) {
            threads[thI].join();
        }

    }
}



int main(int argc, char* argv[]) {
    string input = argv[1];
    string output = argv[2];

    unsigned int threadNum;
    unsigned int n;
    vector<vector<float>> matrix = readInput(input, &n, &threadNum);

    if (n <= 15) {
        cout << "Initial matrix:" << endl;
        for (unsigned int i = 0; i < n; i++) {
            for (unsigned int j = 0; j < n; j++) {
                cout << matrix[i][j] << "  ";
            }
            cout << endl;
        }
    }

    cout << endl;
    GaussJordan(matrix, n, threadNum);
    cout << endl;

    if (n <= 15) {
        cout << "Inverse matrix: " << endl;
        for (unsigned int i = 0; i < n; i++) {
            for (unsigned int j = 0; j < n; j++) {
                cout << matrix[i][j + n] << "  ";
            }
            cout << endl;
        }
    }

    writeOutput(output, matrix, n);

}
```