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

Факультет компьютерных наук Департамент программной инженерии

Потоки: Нахождение обратной квадратной матрицы. ОрепМР. **Пояснительная записка** 

# Оглавление:

Гекст задания	3
Трименяемые расчетные методы	
Список используемых источников	5
Описание работы программы	6
Гестирование программы	7
Триложение №1	11

# 1. Текст задания.

Найти обратную матрицу для матрицы А. 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. Используется OpenMP.

### 2. Применяемые расчетные методы.

Понятие обратной матрицы вводится лишь для квадратных матриц, определитель которых отличен от нуля, то есть для невырожденных квадратных матриц.

Матрица  $A^{-1}$  называется обратной для матрицы A, отпределитель которой отличен от 0, если справедливы равенства:  $A*A^{-1}=E$ , где E – единичная матрица.

Существуют альтернативные методы нахождения обратной матрицы, например, метод Гаусса - Жордана.

Суть метода Гаусса-Жордана заключается в том, что если с единичной матрицей E провести элементарные преобразованиия, которыми невырожденная квадратная матрица A приводится к E, то получится обратная матрица  $A^{-1}$  .

Опишем алгоритм приведения матрицы A порядка n на n, определитель которой не равен нулю, к единичной матрице методом Гаусса - Жордана. После описания алгоритма разберем пример, чтобы все стало понятно.

Сначала преобразуем матрицу так, чтобы элемент  $a_{11}$  стал равен единице, а все остальные элементы первого столбца стали нулевыми.

Если  $a_{11}=0$ , то на место первой строки ставится k-ая строка (k>I), в которой  $a_{k1}\neq 0$ , а на место k-ой строки ставится первая. (Строка с  $a_{k1}\neq 0$  обязательно существует, в противном случае матрица A – вырожденная). После перестановки строк получили «новую» матрицу A, у которой  $a_{11}\neq 0$ .

Теперь умножаем каждый элемент первой строки на  $\overline{a_{11}}$ . Так приходим к «новой» матрице A, у которой  $a_{11}=1$ . Далее к элементам второй строки прибавляем соответствующие элементы первой строки, умноженные на  $-a_{21}$ . К элементам третьей строки – соответствующие элементы первой строки, умноженные на  $-a_{31}$ . И продолжаем такой процесс до n-ou строки включительно. Так все элементы первого столбца матрицы A, начиная со второго, станут нулевыми.

С первым столбцом разобрались, переходим ко второму.

Преобразуем матрицу A так, чтобы элемент  $a_{22}$  стал равен единице, а все остальные элементы второго столбца, начиная с  $a_{32}$ , стали нулевыми.

Если  $a_{2\,2}=0$ , то на место второй строки ставится k-ая строка (k>2), в которой  $a_{k\,2}\neq 0$ , а на место k-ой строки ставится вторая. Так получаем преобразованную матрицу A, у которой  $a_{2\,2}\neq 0$ . Умножаем

все элементы второй строки на  $a_{22}$ . После этого к элементам третьей строки прибавляем соответствующие элементы второй строки, умноженные на  $a_{32}$ . К элементам четвертой строки – соответствующие элементы второй строки, умноженные на  $a_{42}$ . И продолжаем такой процесс до  $a_{42}$ . И продолжаем такой процесс до  $a_{42}$  строки включительно. Так все элементы второго столбца матрицы  $a_{42}$ , начиная с третьего, станут нулевыми, а  $a_{22}$  будет равен единице.

Со вторым столбцом закончили, переходим к третьему и проводим аналогичные преобразования.

Так продолжаем процесс, пока все элементы главной диагонали матрицы A не станут равными единице, а все элементы ниже главной диагонали не станут равными нулю.

С этого момента начинаем обратный ход метода Гаусса-Жордана. Теперь преобразуем матрицу A так, чтобы все элементы n-ozo столбца, кроме  $a_{nn}$ , стали нулевыми. Для этого к элементам (n-1)-ox строки прибавляем соответствующие элементы n-ox строки, умноженные на  $a_{n-1n}$ . К элементам  $a_{n-2n}$  строки — соответствующие элементы  $a_{n-2n}$  строки, умноженные на  $a_{n-2n}$ . И продолжаем такой процесс до первой строки включительно. Так все элементы  $a_{n-2n}$  столбца матрицы  $a_{nn}$ , станут нулевыми.

С последним столбцом разобрались, переходим к (n-1)-ому.

Преобразуем матрицу A так, чтобы все элементы (n-1)-ого столбца до  $a_{n-1}$ , стали нулевыми. Для этого к элементам (n-2)-ой строки прибавляем соответствующие элементы (n-1)-ой строки, умноженные на  $a_{n-2}$   $a_{n-1}$ . К элементам  $a_{n-2}$   $a_{n-1}$ . К элементам  $a_{n-2}$   $a_{n-1}$ . И продолжаем такой процесс до первой строки включительно. Так все элементы  $a_{n-1}$   $a_{n-2}$   $a_{n-1}$   $a_{n-2}$   $a_{n-1}$   $a_{n-1}$   $a_{n-2}$   $a_{n-1}$   $a_{n-2}$   $a_{n-1}$   $a_{n-2}$   $a_{n-2}$   $a_{n-1}$   $a_{n-2}$   $a_{n$ 

Действуя дальше схожим образом, мы получим единичную матрицу.

В программе также используются потоки из стандартной библиотеки c++ std::thread.

- 3. Список используемых источников.
- [1] Инструкция по составлению пояснительной записки [Электронный ресурс]. //URL:\_ http://softcraft.ru/edu/comparch/tasks/mp01/ (Дата обращения: 28.10.2020, режим доступа: свободный)
- [2] Finding inverse of a matrix using Gauss Jordan Method [Электронный ресурс]. //URL:\_https://www.geeksforgeeks.org/finding-inverse-of-a-matrix-using-gauss-jordan-method/ (Дата обращения: 28.10.2020, режим доступа: свободный)
- [3] Нахождение обратной матрицы методом Гаусса [Электронный ресурс]. //URL:\_ http://www.cleverstudents.ru/matrix/finding\_the\_inverse\_matrix.html (Дата обращения: 28.10.2020, режим доступа: свободный)
- [4] std::thread [Электронный ресурс]. //URL:\_https://en.cppreference.com/w/cpp/thread/thread (Дата обращения: 28.10.2020, режим доступа: свободный)
- [5] Multithreading in C++ [Электронный ресурс]. //URL: https://www.geeksforgeeks.org/multithreading-in-cpp/ (Дата обращения: 15.11.2020, режим доступа: свободный)
- [6] Параллельное программирование на OpenMP [Электронный ресурс]. //URL: http://ccfit.nsu.ru/arom/data/openmp.pdf (Дата обращения: 28.11.2020, режим доступа: свободный)

## 4. Описание работы программы.

Инклуды стандартной библиотеки.

```
#include <iostream>
#include <fstream>
#include <fhread>
#include <omp.h>
#include <sstream>
#include <vector>
```

Программа работает с командной строкой. В качестве единственного параметра передается путь к файлу в котором в стандартном виде хранится число, отвечающее за количество строк и столбцов, потоков, плюс матрица записанная по столбцам:

```
Файл Правка Формат Вид Справка

3 2
1 4 5
2 3 4
3 4 5
```

Главный метод программы. Матрицы - вектор векторов. Ввод осуществляется через файл.

```
int main(int argsNumber, char** args) {
    if (argsNumber != 2) {
        std::cout << "Wrong Console Data" << std::endl;
        std::exit(EXIT_FAILURE);
    int N. t:
    std::ifstream in{ args[1] };
    std::vector<std::vector<float>> matrix(N);
    std::vector<float> temp(N);
    for (auto i = 0; i < N; ++i) {
        for (auto j = 0; j < N; ++j) {
            in >> temp[j];
        matrix[i] = temp;
    InverseOfMatrix(matrix, N);
    std::vector<std::thread> threads;
        threads.emplace_back(threadFunc, &matrix);
    for (int i = 0; i < t; ++i) threads[i].join();</pre>
    printf("\n=== Inversed Matrix ===\n");
    PrintInverse(matrix, N, N * 2);
```

Метод, который отвечает за нахождение обратной матрицы. Смотреть подробнее в пункте 2.

В методе Гаусса параллельность применить тяжело. Используется для того, чтобы превратить диагональную матрицу в единичную. Разные потоки отвечают за разные строки диагональной матрицы. В этот раз есть вывод для наглядности работы потоков.

```
InverseOfMatrix(matrix, N);

#pragma omp parallel for
for (int i = 0; i < matrix.size(); ++i) {
    threadFunc((size_t)i, &matrix);
}</pre>
```

```
// Метод делает из диагональной маттрицы единичную. Невероятная польза, это происходит в t раз быстрее.

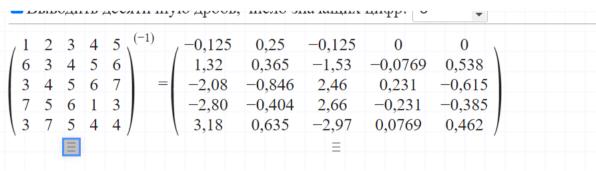
=void threadFunc(size_t i, vector<vector<float>>* A) {
    vector<vector<float>>& matrix = *A;
    float temp_ = matrix[i][i];
    for (size_t j = 0; j < matrix.size() * 2; ++j) {
        matrix[i][j] = matrix[i][j] / temp_;
    }
    printf("\n thread %d calculated row number: %d", omp_get_thread_num(), i);
}
```

Программа выводит на консоль исходную, дополненную и обратную матрицы с точностью до 3 знаков после запятой.

### 5. Тестирование программы.

Пример 1: На матрице 5 на 5. Результат работы программы совпадает с работой веб-приложения сайта matrix.calc.org.

```
)
Консоль отладки Microsoft Visual Studio
=== Matrix ===
   2
     3 4
           5
  3
        5
           6
6
     4
  4
     5
        6
           7
     6
        1
           3
=== Augmented Matrix ===
     6
                    0
        1
           3
              0
                  0
                       1
                           0
           5
     3
        4
               1
                    0
                       0
                          0
        5
     4
           6
                    0
                       0 0
              0
     5
        6
           7
                        0 0
               0
                  0
                     1
                  0
                     0
                       0
thread 0 calculated row number: 0
thread 2 calculated row number: 3
thread 0 calculated row number: 1
thread 1 calculated row number: 2
thread 3 calculated row number: 4
=== Inversed Matrix ===
-0.125 0.250 -0.125 -0.000 -0.000
1.317 0.365 -1.529 -0.077
                              0.538
       -0.846 2.462 0.231
-2.798 -0.404 2.663
                     -0.231 -0.385
3.183 0.635 -2.971 0.077 0.462
C:\Users\User\source\repos\openmp\Debug\openmp.exe (процесс 15708) завершил работу с кодом
Нажмите любую клавишу, чтобы закрыть это окно…
```



таtrix — Блокнот

Файл Правка Формат Вид Ст

5 4

1 2 3 4 5

6 3 4 5 6

3 4 5 6 7

7 5 6 1 3

3 7 5 4 4

Пример 2: Крайний случай: матрица из одно элемента. По факту, нахождение обратного элемента.

```
Консоль отладки Microsoft Visual Studio

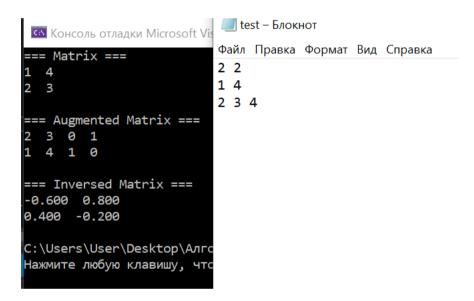
=== Matrix ===
4

=== Augmented Matrix ===
4 1

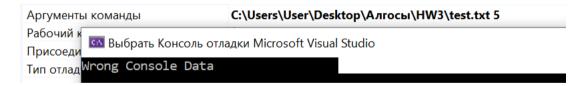
thread 0 calculated row number: 0

=== Inversed Matrix ===
0.250
```

Пример 3: Ввод побитой матрицы. По строкам обрезаются лишние символы.



Пример 4: При некорректных аргументах командной строки выводится сообщение об этом.



```
Код программы:
/**
* Вариант задания 4.
 * Нахождение обратной матрицы для квадратной матрицы размера nxn.
 * Выполнила студентка группы БПИ 199 Вахитова Диана.
 */
#include <iostream>
#include <fstream>
#include <thread>
#include <omp.h>
#include <sstream>
#include <vector>
using namespace std;
// Вывод матрицы.
void PrintMatrix(vector<vector<float>>& ar, int n, int m)
{
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) cout << ar[i][j] << " ";
        printf("\n");
    }
    return;
}
// Вывод обратной матрицы.
void PrintInverse(vector<vector<float>>& ar, int n, int m)
{
    for (int i = 0; i < n; ++i) {
        for (int j = n; j < m; ++j) printf("%.3f ", ar[i][j]);</pre>
        printf("\n");
    }
    return;
}
// Нахождение обратной матрицы.
void InverseOfMatrix(vector<vector<float>>& matrix, int order) {
    float temp_;
    // Печать первоначальной матрицы.
    printf("=== Matrix ===\n");
    PrintMatrix(matrix, order, order);
    for (int i = 0; i < order; ++i) matrix[i].resize(2 * order);</pre>
    for (int i = 0; i < order; ++i) {
        for (int j = 0; j < 2 * order; ++j) {
            if (j == (i + order))
                matrix[i][j] = 1;
        }
    for (int i = order - 1; i > 0; --i) {
        if (matrix[i - 1][0] < matrix[i][0]) {</pre>
            vector<float> temp = matrix[i];
            matrix[i] = matrix[i - 1];
            matrix[i - 1] = temp;
        }
    }
    printf("\n=== Augmented Matrix ===\n");
    PrintMatrix(matrix, order, order * 2);
    // Тут в происходит обратный ход Гаусса, я бы сказала это неудобно параллелить.
    for (int i = 0; i < order; i++) {</pre>
        for (int j = 0; j < order; j++) {</pre>
            if (j != i) {
                temp_ = matrix[j][i] / matrix[i][i];
                for (int k = 0; k < 2 * order; k++) {
                    matrix[j][k] -= matrix[i][k] * temp_;
                }
```

```
}
        }
    return;
}
// Метод делает из диагональной маттрицы единичную. Невероятная польза, это происходит в t раз
быстрее.
void threadFunc(size_t i, vector<vector<float>>* A) {
    vector<vector<float>>& matrix = *A;
    float temp_ = matrix[i][i];
    for (size_t j = 0; j < matrix.size() * 2; ++j) {</pre>
        matrix[i][j] = matrix[i][j] / temp_;
    printf("\n thread %d calculated row number: %d", omp_get_thread_num(), i);
}
int main(int argsNumber, char** args) {
    // Обработка аргументов командной строки.
    if (argsNumber != 3) {
        std::cout << "Wrong Console Data" << std::endl;</pre>
        std::exit(EXIT_FAILURE);
    }
    // Для оптимизации используется работа с файлами.
    int N, t;
    std::ifstream in{ args[1] };
    in \gg N \gg t;
    std::vector<std::vector<float>> matrix(N);
    std::vector<float> temp(N);
    // Заполняем матрицу.
    for (auto i = 0; i < N; ++i) {
        for (auto j = 0; j < N; ++j) {
            in >> temp[j];
        matrix[i] = temp;
    }
    InverseOfMatrix(matrix, N);
    #pragma omp parallel for
    for (int i = 0; i < matrix.size(); ++i) {</pre>
        threadFunc((size_t)i, &matrix);
    // Печатаем результат.
    printf("\n=== Inversed Matrix ===\n");
    PrintInverse(matrix, N, N * 2);
}
```