

# Документация по проекту с матрицами

Timofei Avetisov

## 1 Вступление

Данный проект был написан за несколько дней, так что не советую читать ~~ровно~~ код. Но с его помощью можно делать разные операции с матрицами, которые обычно лень выполнять вручную.

## 2 Как этим пользоваться

Для начала необходимо скачать всё с GitHub. Откройте проект и приготовьтесь. Работать нужно только с файлом `main.cpp`. В начале обязательно напишите:

```
#include "Header/Matrix/Matrix.h"
```

Это автоматически подключит `<bits/stdc++.h>`. Если нужно что-то ещё — смело подключайте. Далее пишите код.

Немного расскажу про `atmla::Fraction`. У класса есть два поля - `numerator`, `denominator`. Конструкторы есть следующие:

1. `atmla::Fraction f(int n, int m)` - создаст дробь  $\frac{n}{m}$
2. `atmla::Fraction f(int n)` - создаст дробь  $\frac{n}{1}$
3. `atmla::Fraction f(double n)` - создаст дробь, которая с точностью до  $1e-5$  будет равна `n`
4. Так же поддерживается потоковый ввод, например `std::cin >> f` - принимает строку, строка может быть двух видов:

**Число** Целое число `n` - получится дробь  $\frac{n}{1}$

**Дробь** Строка виде `n/m` - получится дробь  $\frac{n}{m}$  (`n`, `m` - целые числа)

## 3 Как взаимодействовать с объектами

На данный момент существует всего 2 класса, которые лежат в `namespace atmla` (увековечил своё имя). Для создания матрицы используйте:

```
atmla::Matrix<datatype> name(parameters);
```

`datatype` может быть `int`, `double` или `atmla::Fraction` (я рекомендую использовать `atmla::Fraction`). Конструкторы бывают следующие:

1. `matrix(int n, int m)` — создаёт пустую матрицу размером  $n \times m$ .
2. `matrix(std::vector<std::vector<datatype>> values)` — создаёт матрицу из вектора `values` (не знаю, зачем это нужно). Тип данных `values` может отличаться от типа данных матрицы (всё нормально приведётся).
3. `matrix(int n, std::string special, datatype lambda)` — создаёт специальную матрицу размером  $n \times n$ . `special` может принимать следующие значения:
  - "identity" — создаёт единичную матрицу.
  - "zero" — создаёт нулевую матрицу.

- "jordan cell" — Жорданова клетка с элементами `lambda` на диагонали (по умолчанию 0).
- "diagonal" — диагональная матрица с элементами `lambda` на диагонали (по умолчанию 0).

После создания матрицы её размеры изменить нельзя, но можно менять значения. Для этого предусмотрены следующие методы класса:

1. `matrix.Set_values_from_vector(std::vector<std::vector<datatype>> values)` — тип данных в векторе должен совпадать с типом данных матрицы. Эта функция просто запишет данные из вектора в матрицу.
2. `matrix.Set_values_manually()` — вводите все данные вручную.
3. Так же поддерживается потоковый ввод матрицы `std::cin >> matrix`

**ВАЖНО** на данной момент есть небольшая проблема, `atmla::Fraction` можно вводить только по одному элементу в строке.

## 4 Функции вывода и операторы

Матрицу можно вывести несколькими способами:

1. Метод `matrix.print()` — просто выведет матрицу (почти бесполезно).
2. Вывод в стандартный поток вывода:

```
std::cout << matrix;
```

Существуют специальные способы вывода:

1. Метод `matrix.print_system(std::vector<datatype> values)` — выведет матрицу и столбец `values` как СЛУ (система линейных уравнений). Тип данных в `values` может отличаться от типа данных матрицы.
2. Метод `matrix.print_matrix_system(atmla::Matrix<datatype> other)` — выведет систему матричных уравнений.

Также перегружены стандартные операторы для матриц:

- Сложение, вычитание, умножение матриц.
- Умножение, деление, сложение, вычитание на скаляр (каждый элемент матрицы будет поделен на скаляр).

## 5 Другие функции

1. `matrix.Transpose()` — транспонирует матрицу.
2. `matrix.Determinant()` — возвращает определитель матрицы.
3. `matrix.Trace()` — возвращает след матрицы.
4. `matrix.gauss(std::vector<datatype> values)` — приводит матрицу к улучшенному ступенчатому виду, изменяя также `values`. После этого вызовите `matrix.print_system(values)`, чтобы получить решение. Тип данных `values` может отличаться от типа данных матрицы.
5. `matrix.Inverse()` — находит обратную матрицу.

Наводя курсор на функцию, вы увидите информацию о её параметрах и возможных ошибках.

## 6 Заключение

После написания кода, в терминале зайдите в папку `builds`, выполните команду `cmake ..`, затем `cmake -build ..`. После этого в папке `build` появится исполняемый файл `matrix.exe`, связанный с `main.cpp`.

Удачи!