МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по практической работе №4
по дисциплине «Объектно-ориентированное программирование»
Тема: Распределенное приложение, включающее клиентскую и серверную части, для вычислений над квадратной матрицей на множестве рациональных чисел

Студенты гр. 1361 _____ Горбунова Д.А.
_____ Кравцов И.Ю.
Преподаватель _____ Егоров С.С.

Санкт-Петербург
2024

ЗАДАНИЕ НА ПРАКТИЧЕСКУЮ РАБОТУ

Тема работы: создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.

Исходные данные:

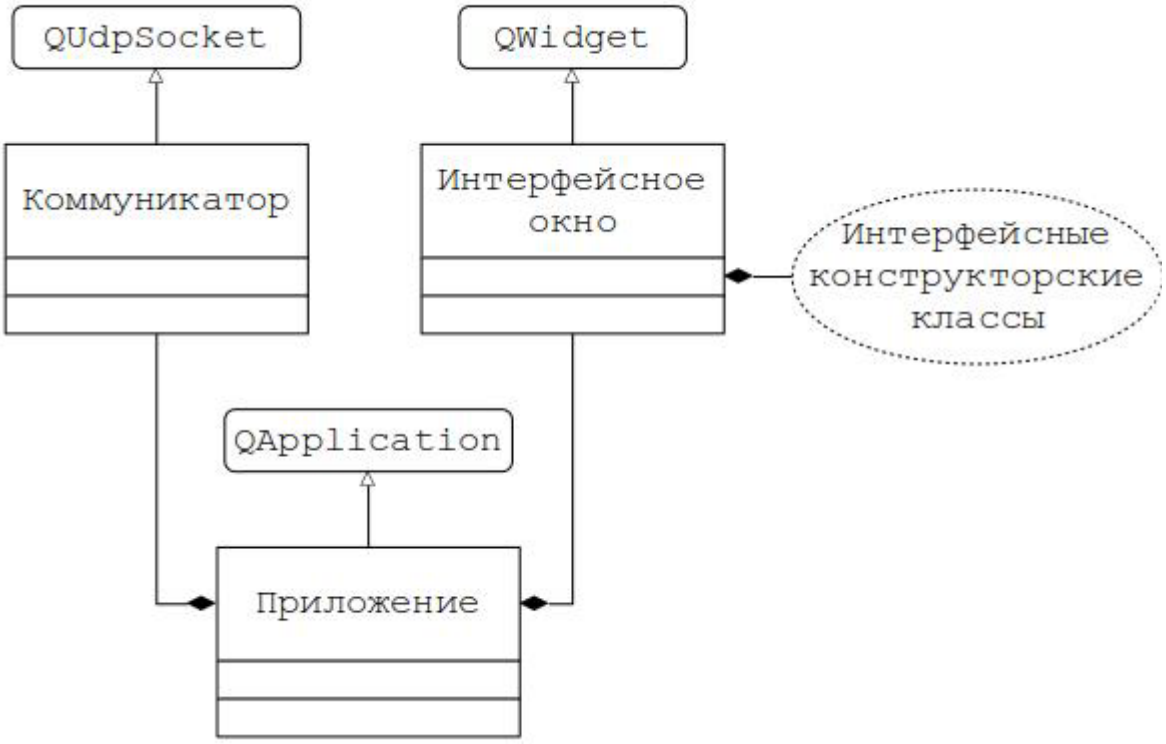

Рис.1 – Диаграмма классов клиентской части

Рис.2 – Диаграмма классов серверной части

Описание работы:

Создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.

Клиентская часть представляет собой GUI приложение, реализующее интерфейс аналогичный работе №3.

Серверная часть представляет собой консольное приложение, предназначенное для выполнения перечисленных в меню работы №1 функций над квадратной матрицей с рациональными элементами.

Диаграммы классов для клиентского и серверного приложений представлены на рис.1 и рис.2 соответственно.

СОДЕРЖАНИЕ

1. СПЕЦИФИКАЦИЯ РАЗРАБОТАННЫХ КЛАССОВ

Классы проекта серверного приложения:

1.1. Класс `TRational` («Рациональные числа»):

Атрибуты класса:

a) `int m_numerator, m_denominator`

Область видимости: private.

Описание: хранение числителя и знаменателя дроби соответственно.

Стандартное значение: 0.

Чтобы предотвратить несанкционированный доступ, атрибуты класса ограничены доступом и могут быть использованы только внутри модуля, где определен класс квадратной матрицы.

Методы класса:

a) `TRational()`

Область видимости: private.

Назначение: Конструктор, инициализирует объект класса TRational с заданными числителем и знаменателем, используя значения по умолчанию 0 и 1 соответственно.

b) `int numerator() const`

Область видимости: private.

Назначение: возвращает числитель дроби.

c) `int denominator() const`

Область видимости: private.

Назначение: возвращает знаменатель дроби.

d) `Void normalize()`

Область видимости: private.

Назначение: нормализует дробь, упрощая её до наименьших возможных значений числителя и знаменателя.

e) `operator+, operator-, operator*, operator/, operator==, operator!=, operator>`

Область видимости: private.

Назначение: операторы для выполнения арифметических операций и сравнения дробей.

f)`abs()`

Область видимости: private.

Назначение: статический метод, возвращающий абсолютное значение дроби.

g) `int gcd (int a, int b)`

Область видимости: private.

Назначение: статический метод, вычисляющий наибольший общий делитель двух чисел.

h) `toString()`

Область видимости: private.

Назначение: метод, возвращающий строковое представление дроби.

1.2. Класс `ColsoleApp`

Атрибуты класса:

a) `comms`

Область видимости: private

Назначение: указатель на объект класса `TCommunicator`

b) `matrix`

Область видимости: private

Назначение: указатель на объект класса `Matrix*`

Методы класса:

a) `void requestForm(int n)`

Область видимости: private

Назначение: метод формирования запроса.

b) `void msgReciever (QString)`

Область видимости: public

Назначение: метод обработки принятых сообщений

c) `ConsoleApp(int, char**);`

Область видимости: public

Назначение: Конструктор консольного приложения

1.3. Класс `Matrix`

Атрибуты класса

a) `Int row, int column`

Область видимости: private

Назначение: хранят в себе строки и столбцы матрицы.

b) `vector<vector<number>> m_values;`

Область видимости: private

Назначение: хранит в себе массив матрицы.

Методы класса:

a) `Matrix();`

Область видимости: public

Назначение: Конструктор по умолчанию

b) `Matrix (int rows, int cols);`

Область видимости: public

Назначение: Конструктор с параметрами

c) `void generate_1();`

Область видимости: public

Назначение: метод заполнения матрицы в виде единичной

d) `void print_screen();`

Область видимости: public

Назначение: метод вывода матрицы в консоль

e) `number determinant() const;`

Область видимости: public

Назначение: метод вычисления определителя

f) `void transpose();`

Область видимости: public

Назначение: метод транспонирования метрицы (был переписан)

g) `void CinMatrix(int n);`

Область видимости: public

Назначение: метод записывания матрицы в память

h) `int rows() const;`

Область видимости: public

Назначение: метод получения количества строк

i) `int cols() const;`

Область видимости: public

Назначение: метод получения количества столбцов

j) `int rank();`

Область видимости: public

Назначение: метод вычисления ранга

k) `void print_screen(std::stringstream& ss) const;`

Область видимости: public

Назначение: метод вывода матрицы в строковую переменную

l) `void setMatrixValue(int row, int col, int num, int det);`

Область видимости: public

Назначение: метод записи по элементно матрицу

m) `void setDimension(int dimension);`

Область видимости: public

Назначение: метод изменение размера матрицы

n) `number getElement(int row, int col);`

Область видимости: public

Назначение: метод получения единичного элемента в матрицы

Классы проекта клиентского приложения:

1.1. Класс `MatrixDisplayWindow` (дочерний класс от QDialog).

Атрибуты класса:

a) `QTextEdit *matrixTextEdit`

Область видимости: private

Назначение: указатель на объект.

Методы:

a) `explicit MatrixDisplayWindow(QWidget *parent = nullptr)`

Область видимости: public

Назначение: конструктор.

b) `void displayMatrix(const std::stringstream &matrixText)`

Область видимости: public

Назначение: отображение матрицы в текстовом поле.

1.2. Класс `Interface` (дочерний класс от `QWidget`)

Атрибуты класса:

a) `QPushButton *_1x1Btn, *_2x2Btn, *_3x3Btn, *_editBtn;`

Область видимости: private

Назначение: указатель на объект.

b) `QPushButton *_outputMatrixBtn, *_transporiseBtn,`

```
*_rangBtn, *_determinantBtn, *_inputMatrixBtn;
```

Область видимости: private

Назначение: указатель на объект.

c) `QLineEdit *_sizeEdit;`

Область видимости: private

Назначение: указатель на объект.

d) `QGridLayout *gridLayout;`

Область видимости: private

Назначение: указатель на объект.

e) `QFrame *_displayWall;`

Область видимости: private

Назначение: указатель на объект.

f) `QMap <quint16, QLineEdit*> _numeratorEditMap,`
`_denominatorEditMap;`

Область видимости: private

Назначение: экземпляры карт хранения.

g) `int _row;`

Область видимости: private

Назначение: объект хранения размера строк.

h) `int _column;`

Область видимости: private

Назначение: объект хранения размера столбцов.

Методы:

a) `Interface(QWidget *parent = nullptr);`

Область видимости: public

Назначение: Конструктор по умолчанию.

b) `~Interface();`

Область видимости: public

Назначение: Декструктор.

c) `void recieveMSG (QString);`

Область видимости: public

Назначение: метод обработки входящих запросов

d) `void onSetMatrixLayout(int row, int column, bool bIsResize=false);`

Область видимости: private

Назначение: метод позволяющий изменять размер и ограничения сеток в пользовательском интерфейсе

e) `void onMatrixLayoutDivision();`

Область видимости: private

Назначение: метод динамического изменения размера матрицы в пользовательском интерфейсе

f) `void onInputMatrixBtnClicked();`

Область видимости: private

Назначение: метод для инициации обработки ввода матрицы

g) `void onOutputMatrixBtnClicked();`

Область видимости: private

Назначение: метод для инициации обработки вывода матрицы

h) `void onTransposeMatrixBtnClicked();`

Область видимости: private

Назначение: метод для инициации обработки транспонирования матрицы

i) `void onDeterminantMatrixBtnClicked();`

Область видимости: private

Назначение: метод для инициации обработки определителя матрицы

j) `void onRangMatrixBtnClicked();`

Область видимости: private

Назначение: метод для инициации обработки ранга матрицы

k) `void removeNumeratorDenominatorFields();`

Область видимости: private

Назначение: метод для очистки пользовательского интерфейса от полей

ввода чисел и знаменателей дробей

l) `void SENDER(QString);`

Область видимости: private

Назначение: сигнал

m) `void removeLabel();`

Область видимости: private

Назначение: метод для очистки графического интерфейса

n) `virtual void resizeEvent(QResizeEvent *event);`

Область видимости: private

Назначение: метод служит для обновления сетки макета при увеличении размера окна

o) `QString requestForm(int n);`

Область видимости: private

Назначение: метод для формирования ответа на запрос

1.3. Класс `app` (дочерний класс от `QApplication`)

Атрибуты класса:

a) `interface;`

Область видимости: private

Назначение: указатель на экземпляр класса `Interface`

b) `comms;`

Область видимости: private

Назначение: указатель на экземпляр класса `TCommunicator`

Методы класса:

a) `app(int , char **);`

Область видимости: public

Назначение: конструктор класса по умолчанию

b) `void dove(QString);`

Область видимости: public

Назначение: метод для отправки сообщений через канал связи

c) `void postOffice(QString);`

Область видимости: public

Назначение: метод для принятия сообщений через канал связи

Так же были реализованы классы для канала связи:

1.1. Класс `communicator`

Атрибуты класса:

a) `QUdpSocket* socket;`

Область видимости: public

Назначение: метод для принятия сообщений через канал связи

b) `TCommonParam param;`

Область видимости: public

Назначение: метод для принятия сообщений через канал связи

c) `bool ready;`

Область видимости: public

Назначение: метод для принятия сообщений через канал связи

Методы класса:

a) `explicit TCommunicator(TCommonParam&, QObject *parent = nullptr);`

Область видимости: public

Назначение: Конструктор по умолчанию.

b) `~TCommunicator();`

Область видимости: public

Назначение: Декструктор

c) `bool isReady();`

Область видимости: public

Назначение: метод для проверки готовности канала связи

d) `void send(const QString& message);`

Область видимости: public

Назначение: метод для отправки сообщений по канал связи

e) `void recieved(const QString& message);`

Область видимости: private

Назначение: сигнал

f) `void processPendingDatagrams();`

Область видимости: private

Назначение: метод для принятия сообщений по канал связи

2. ДИАГРАММА КЛАССОВ

Диаграмма классов представлена на рисунке 2 и 3. На них обозначены атрибуты классов, их методы и операторы. Знаками «+» обозначены методы, атрибуты, имеющие общедоступный тип доступа и операторы; соответственно «-» – приватный. Также через двоеточие указаны типы возвращаемых значений, где они есть. В круглых скобках представлены типы параметров соответствующих методов, операторов, конструкторов. На рисунке 2 показана диаграмма классов клиентской части, а на рисунке 3 диаграмма классов серверной части.

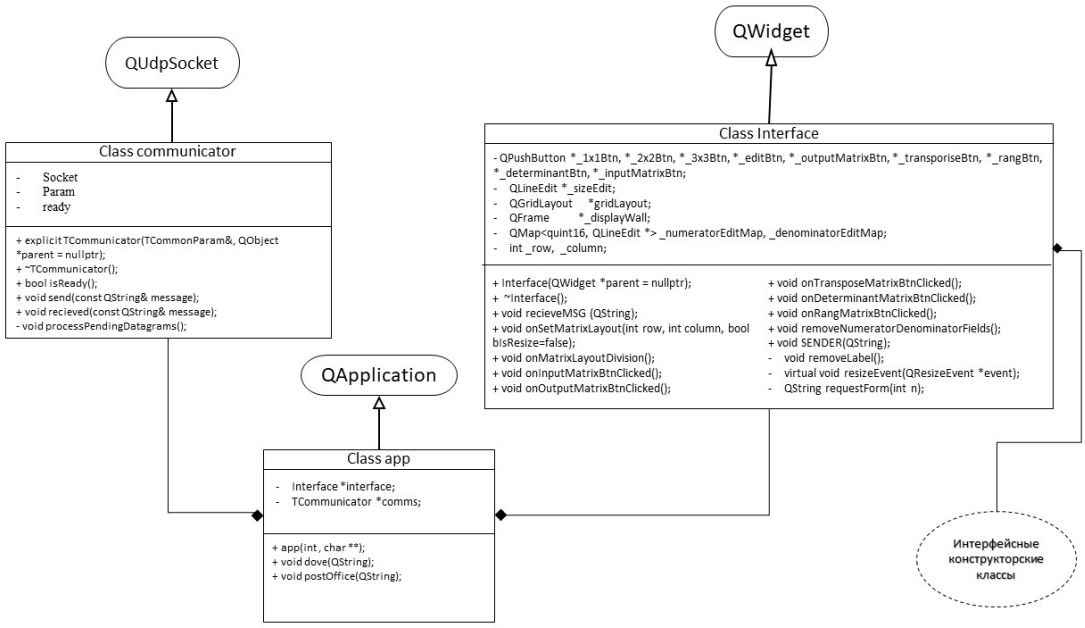

Рисунок 3 – Диаграмма классов со стороны клиента

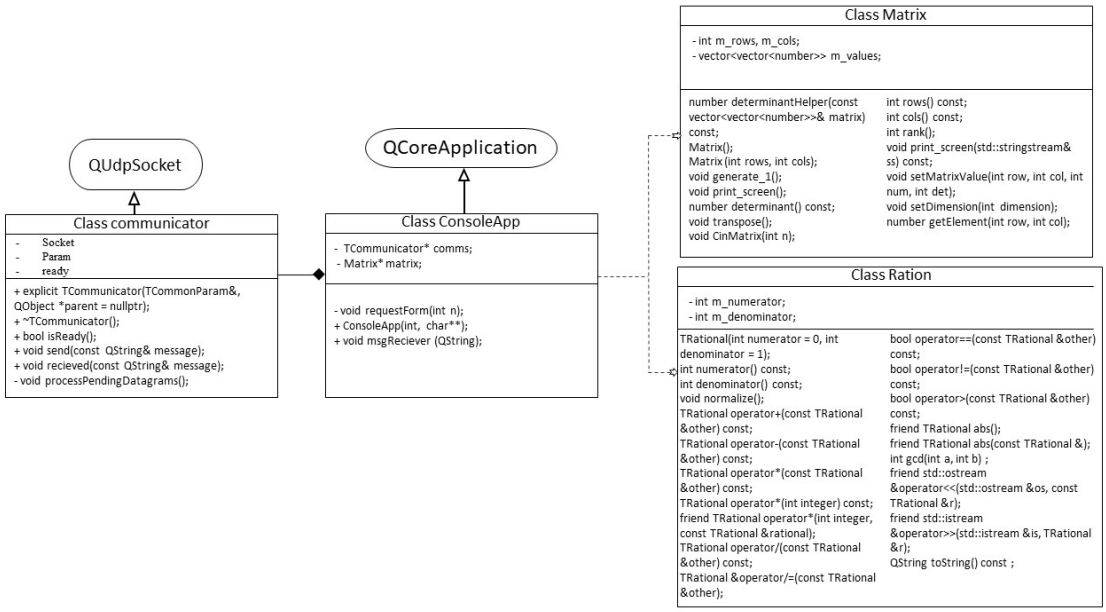

Рисунок 4 – Диаграмма классов со стороны сервера

3. ОПИСАНИЕ КОНТРОЛЬНОГО ПРИМЕРА

Исходные данные для контрольного примера:

$$A = \begin{pmatrix} \frac{1}{2} & \frac{3}{4} \\ \frac{7}{5} & \frac{8}{3} \end{pmatrix}$$

Чтобы найти транспонированную матрицу поменяем строки столбцы матрицы местами:

$$\mathbf{A^T} = \begin{pmatrix} \frac{1}{2} & \frac{7}{5} \\ \frac{3}{4} & \frac{8}{3} \end{pmatrix}$$

Далее ищем определитель матрицы:

$$\det A = \begin{vmatrix} \frac{1}{2} & \frac{3}{4} \\ \frac{7}{5} & \frac{8}{3} \end{vmatrix} = \frac{1}{2} * \frac{8}{3} - \frac{7}{5} * \frac{3}{4} = \frac{4}{3} - \frac{21}{20} = \frac{17}{60}$$

Т.к. $\det A \neq 0$, то линейно зависимых строк или столбцов нет $\Rightarrow$ $\text{rank } A = 2$.

4. РАБОТА ПРОГРАММЫ НА КОНТРОЛЬНЫХ ПРИМЕРАХ

При запуске приложения, программа выводит на экран доступные варианты работы с матрицей (рисунок 3).

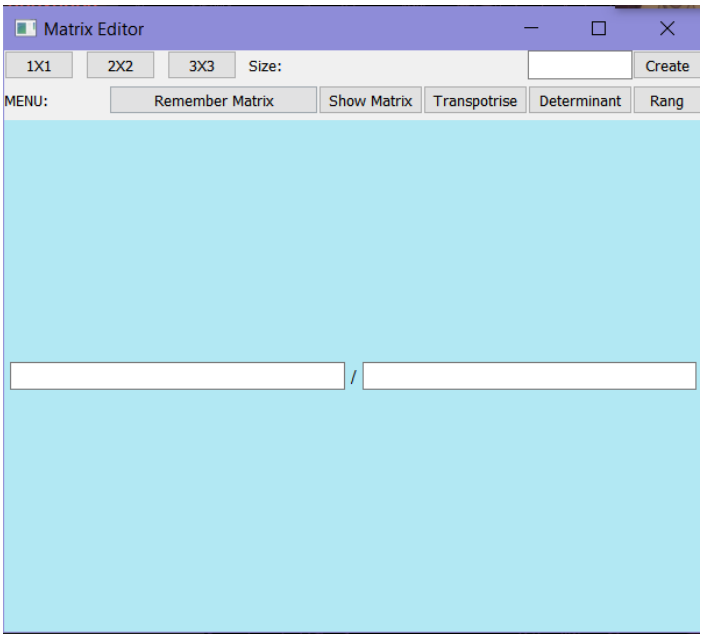

Рисунок 3 – Меню

Далее производим ввод пользовательской матрицы в приложении.

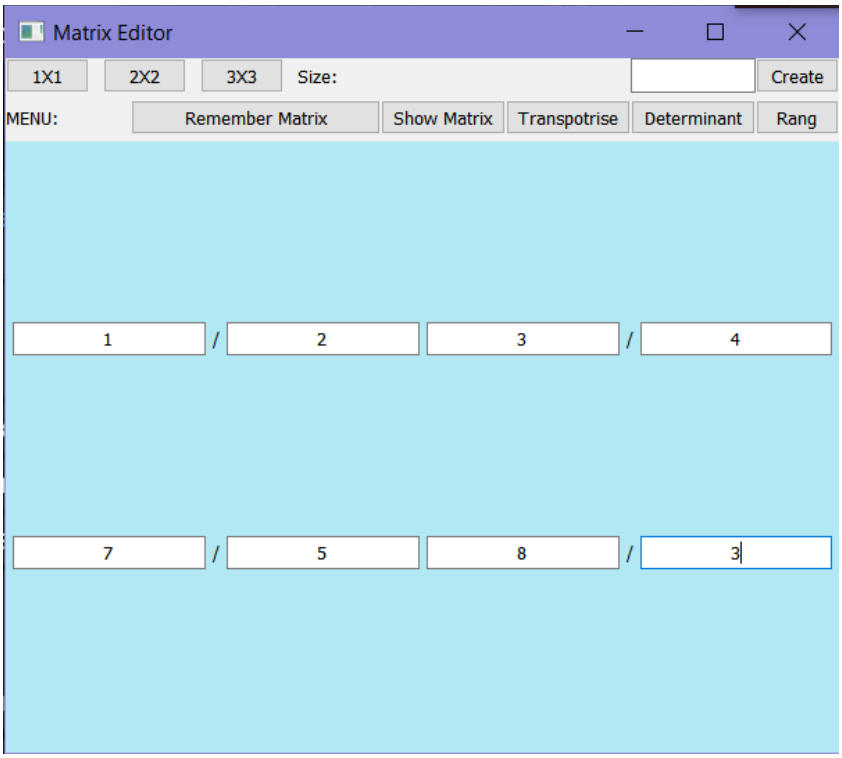

Рисунок 4 – Ввод пользовательской матрицы

После того как мы ввели коэффициенты, можем, нажав кнопку “Remember

Matrix”, записать данную матрицу в память. Приложение выведет следующее сообщение (рисунок 5).

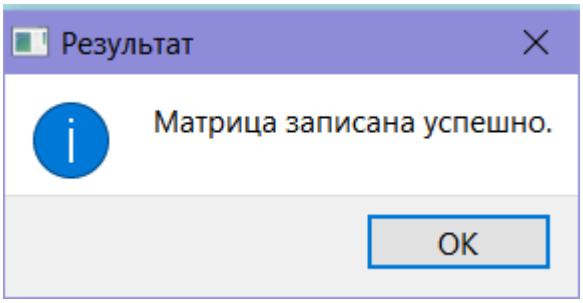

Рисунок 5 – Результат кнопки «Remember Matrix»

После того мы определили матрицу, ее можно вывести, нажав кнопку «Show Matrix» (рисунок 6).

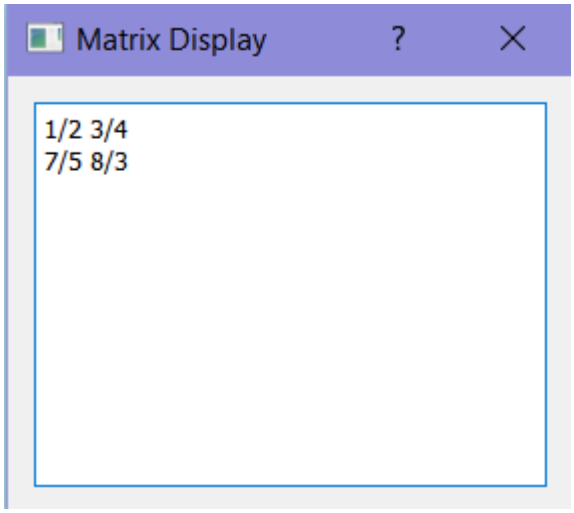

Рисунок 6 – Вывод матрицы.

После того, как у нас матрица задана, мы производим над ней вычисления. Сначала вычислим транспонированную матрицу, нажав кнопку“Transpotrise” (рисунок 7).

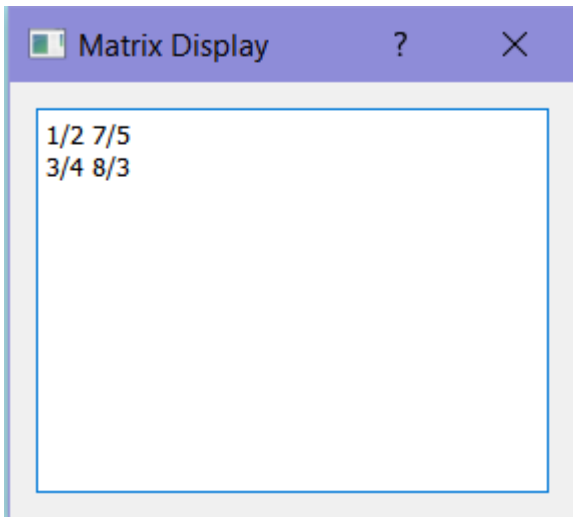

Рисунок 7 – Транспонирование матрицы.

Далее вычисляем ранг матрицы, нажав кнопку “Rang” (рисунок 8).

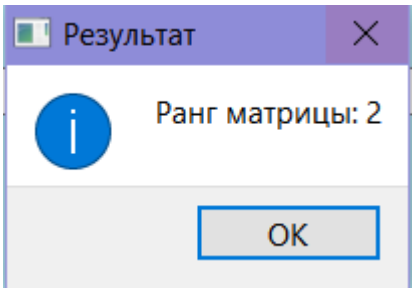

Рисунок 8 – Ранг матрицы.

Если же выбрать кнопку “Determinant, то программа вычислит определитель матрицы (рисунок 9).

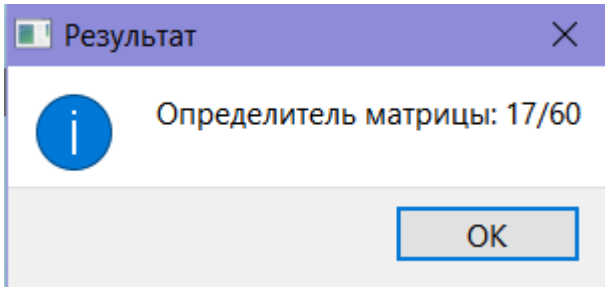

Рисунок 9 – Определитель матрицы.

Для завершения программы достаточно нажать на крест в верхнем правом углу.

При каждом действии сервер в ответ на запрос выводит сообщение об отправке ответа.

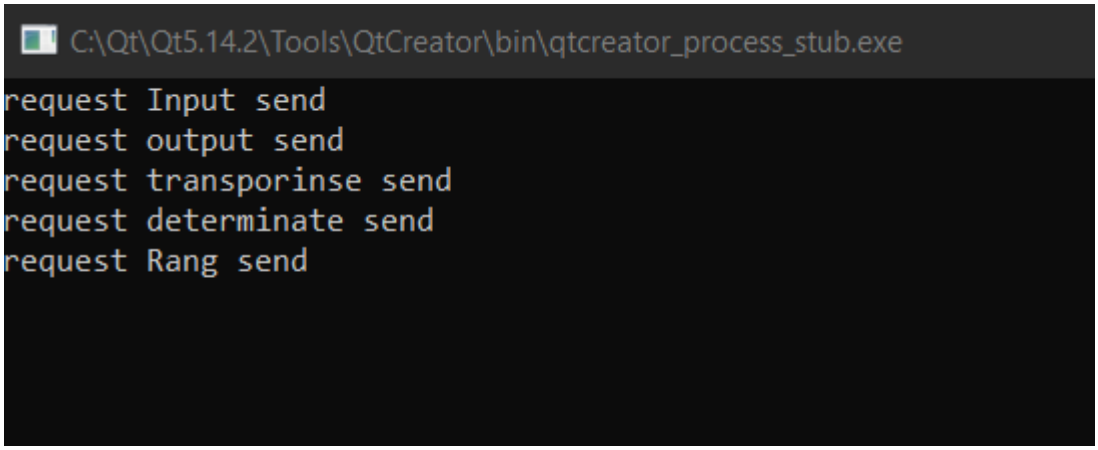

Рисунок 10 – результат работы сервера в консольном приложении

ВЫВОДЫ ПО ВЫПОЛНЕННОЙ РАБОТЕ

В результате выполнения данной практической работы на языке программирования C++ было написано GUI приложение, а также серверная часть приложения, предназначенное для заданных вычислений над квадратной матрицей, заданной на множестве рациональных чисел.

В ходе выполнения работы были реализованы четыре класса:

• TCommunicator

• Application (Клиентская часть)

• Application (Серверная часть)

• Widget

Области видимости классов были успешно применены с подробным объяснением их использования. Были разработаны диаграммы классов, которые точно отражают структуру как приложения, так и серверной части.

Для проверки корректности работы приложения был представлен контрольный пример с расчетными значениями, результаты которого подтвердили правильность всех расчетов.