**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №6**

**по дисциплине «Объектно-Ориентированное Программирование»**

|  |  |  |
| --- | --- | --- |
| Студенты группы № 2372 |  | Полуянов В.Н., Юрин А.А. |
| Преподаватель |  | Егоров С.С. |

Санкт-Петербург

2023

**Задание**

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

Клиентские и серверные части представляют собой приложения, реализованные в работе №5.

Клиентская часть модифицируется таким образом, что реализованные функции матриц могут исполняться по желанию пользователя на областях определения: вещественная, комплексная и рациональная.

Отличие серверной части заключается в том, что классы «Матрица» и «Квадратная матрица» параметризуются. Параметром класса делается абстрактный тип **number**, при этом файл number.h исключается из серверного приложения

Реализовать и отладить программу, удовлетворяющую сформулированным требованиям и заявленным целям. Разработать контрольные примеры и оттестировать на них программу. Оформить отчет, сделать выводы по работе.

**Спецификации классов**

**Class TApplication(client)**

Атрибуты:

-comm - объект TCommunicator

-interface - объект TInterface

Методы:

+TApplication(int, char \*\*) - конструктор

+fromCommunicator(QByteArray) - принимает сообщение от коммуникатора

+toCommunicator(QString) – отправляет сообщение коммуникатору

**Class TApplication(server)**

Атрибуты:

-comm - объект TCommunicator

Методы:

+TApplication(int, char \*\*) - конструктор

+receive(QByteArray) – получает и обрабатывает запрос

**Class TCommunicator**

Атрибуты:

-ready - отражает состояние коммуникатора

-params – параметры

Методы:

+TCommunicator(TCommParams &, QObject \*parent = nullptr) - конструктор

+isReady() – возвращает состояние коммуникатора

+send(QByteArray) – отправляет запрос

+receive() – принимает запрос

**Class TMatrix**

Атрибуты:

-unsigned short sizeX – определяет количество столбцов матрицы

-unsigned short sizeY – определяет количество строк матрицы

-std::vector<std::vector<number>> values - определяет значения матрицы

+unsigned int getRank() – вычисляет и возвращает ранг

Методы:

+TMatrix() – конструктор класса без предустановленного размера

+TMatrix(unsigned short, unsigned short) – конструктор с изменяемым размером

+setValues(std::vector<std::vector<number>>&) – изменяет значения матрицы

+setSize(unsigned short) – изменяет размер матрицы (расширяет или сжимает её с сохранением значений, новые - нули)

+getSizeX() – возвращает текущий размер строк

+getSizeY() – возвращает текущий размер столбцов

+operator<<(std::ostream&, TMatrix&) – выводит матрицу в поток вывода

+getDeterminant() – возвращает number определитель матрицы

+getRank() – возвращает unsigned int ранг матрицы

+transpose() – транспонирует матрицу

#toUpperTriangularForm() – приводит матрицу к верхнему ступенчатому виду с помощью метода Гаусса. Используется для вычисления определителя и ранга матрицы.

#sizeX – количество строк

#sizeY – количество столбцов

#values – значения матрицы. Можно распечатать и изменить с помощью методов.

**Class TRational**

Атрибуты:

-SEPARATOR – разделитель в текстовой репрезентации объекта

-numerator – числитель

-divisor – знаменатель

Методы:

+TRational() – стандартный конструктор. Задает число 0

+TRational(double&) – конструктор целого числа

+TRational(double&, double&) – конструктор дробного числа

+operator<< (QString&, TRational) – выводит число в QString

+operator+ (Trational) – перегрузка оператора сложения

+operator- (Trational) – перегрузка оператора вычитания

+operator\* (Trational) – перегрузка оператора умножения

+operator/ (Trational) – перегрузка оператора деления

+operator== (TRational num) – перегрузка оператора равно

+operator!= (TRational num) – перегрузка оператора не равно

**Class TSquareMatrix**

Атрибуты:

Методы:

+TSquareMatrix(unsigned short) - создает матрицу произвольного размера, заполненную нулями

+setSize(unsigned short) - изменяет размер матрицы (сохраняет старые значения)

+getSize() – возвращает текущий размер

+getDeterminant() – вычисляет и возвращает определитель

**Class TInterface**

Атрибуты:

-double real – вещественная часть комплексного числа

-double image –мнимая часть комплексного числа

Методы:

+Complex() – конструктор класса (по умолчанию)

+Complex(const double&) – конструктор класса (передаётся вещественная)

+Complex(const double&, const double&) – конструктор класса (передаётся вещественная и мнимая части)

+explicit Complex(const QByteArray&) – конструктор, передаётся массив байтов

+friend std::istream& operator>> (std::istream&, Complex&) – консольный ввод

+friend std::ostream& operator<< (std::ostream&, Complex) – консольный вывод

+friend QString& operator<< (QString&, Complex) – выводит число в QString

+friend QByteArray& operator>> (QByteArray&, Complex&) – записывает число в QByteArray

+Complex operator+ (Complex) const – перегрузка оператора сложения

+Complex operator- (Complex) const – перегрузка оператора вычетания

+Complex operator\* (Complex) const – перегрузка оператора умножения

+Complex operator/ (Complex) const – перегрузка оператора деления

+bool operator== (Complex num) const – перегрузка оператора равно

+bool operator!= (Complex num) const – перегрузка оператора не равно

**Class TInterface**

Атрибуты:

-int matrixSize – размер создаваемой матрицы

-std::vector<std::vector<std::vector<QLineEdit\*>>> numbers – вектор полей ввода значений матрицы

-QPushButton \*print\_btn – кнопка «вывести матрицу»

-QPushButton \*determinant\_btn – кнопка «определитель»

-QPushButton \*rank\_btn – кнопка «ранг»

-QPushButton \*transpose\_btn – кнопка «транспонировать»

-QRadioButton \*doubleMode – режим вещественных чисел

-QRadioButton \*complexMode – режим комплексных чисел

-QRadioButton \*rationalMode – режим рациональных чисел

+request(QString) – запрос к серверу

Методы:

+TInterface() – конструктор класса

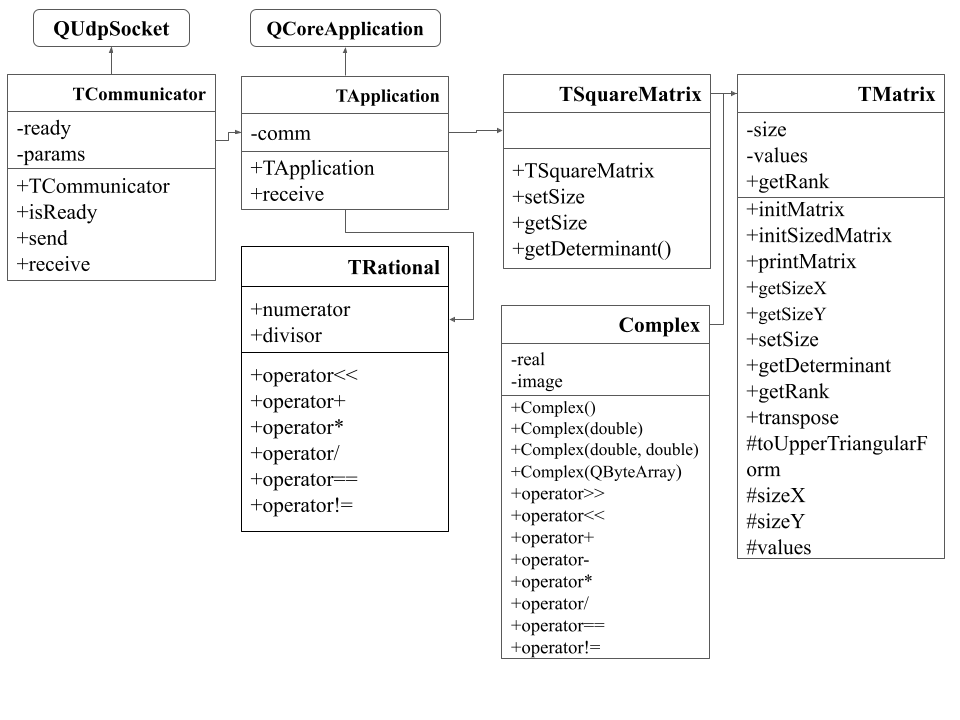
+~TInterface() – деструктор класса

+answer(QString) - получает ответ сервера и выводит результат

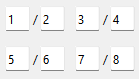
+formRequest() ­ - формирует запрос к серверу

**Диаграмма классов, дополненная атрибутами и методами**

|  |
| --- |
| Клиентская часть |
|  |
| Серверная часть |
|  |



**Описание контрольного примера с исходными и ожидаемыми (расчетными) данными**

****

Расчетные результаты для вещественных чисел:

* Определитель равен -8
* Ранг равен 2

Расчетные результаты для матрицы комплексных чисел:

* Определитель равен -16i
* Ранг равен 2

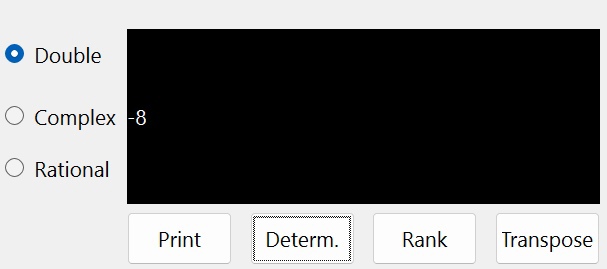
При тестировании работы программы были получены идентичные (правильные) результаты.

**Скриншоты работы программы на контрольных примерах. Два примера**

1. *Первый пример*





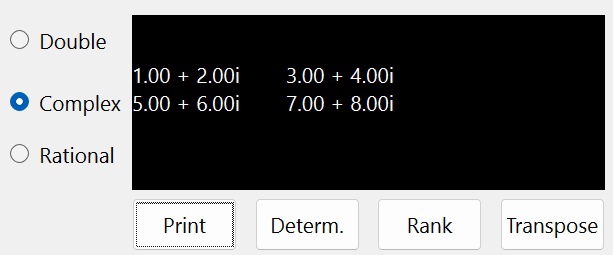
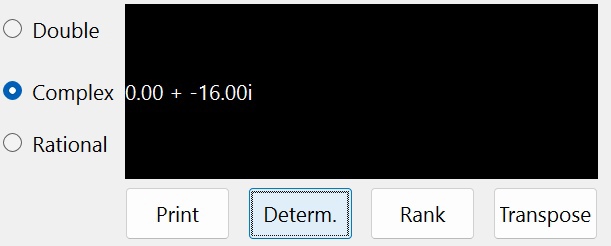
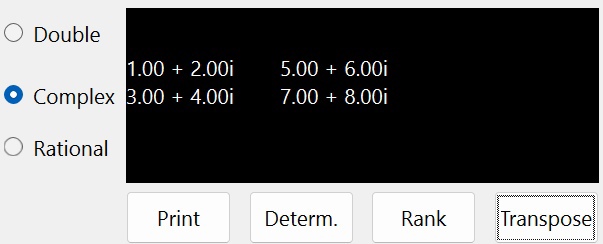






1. *Второй пример*



**Вывод**

В данной практической работе было создано распределенное приложение, состоящее из серверной и клиентской частей. Взаимодействие между ними реализовано через сетевой обмен текстовыми сообщениями. Клиент был реализован на основе GUI приложения из практической работы №3, а сервер – на основе консольного приложения из практической работы №1. Параметризирован базовый класс матрицы, создан производный класс «квадратная матрица» с параметром – абстрактным классом number.