**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Информационных систем**

**ОТЧЕТ**

**по практической работе №5**

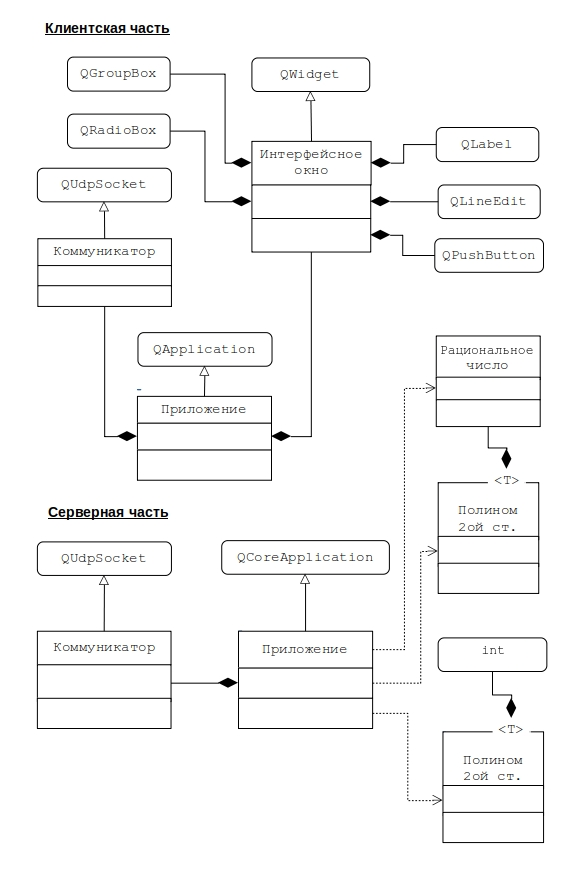
**по дисциплине «Объектно-ориентированное программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8374 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пихтовников К.С. |
| Студент гр. 8374 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Подсекин Г.С. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Егоров С.С. |

Санкт-Петербург

2021

**Задание на практическую работу**

****

*Рис.1. Диаграмма классов работы №5*

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 вычисления корней, значения и представления полинома 2-ой степени в классической и канонической формах на множествах вещественных и рациональных чисел. Для этого следует модифицировать описание класса **TPolinom,** представив его в виде параметризованного класса с параметром **number,** который может принимать значения как **int**, так и **TRational**.

Указания:

1. Заголовочный файл number.h из приложения исключить.

2. Содержимое файла реализации polinom.cpp перенести в заголовочный файл polinom.h (шаблон класса TPolinom) и исключить из проекта.

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

**Спецификация классов**

**Клиентская часть распределенного приложения:**

Класс TInterface

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибуты:  QLabel \* name\_a, \*delimeter\_a;  QLabel \* name\_b, \*delimeter\_b;  QLabel \* name\_c, \*delimeter\_c; | область видимости - private.  Являются подсказками, которые показывают пользователю, что нужно вводить |
| Атрибуты:  QLineEdit\*a\_numerator,\*a\_denominator;  QLabel \* name\_b, \*delimeter\_b;  QLabel \* name\_c, \*delimeter\_c | область видимости - private.  Переменные, которые хранят введенные числитель и знаменатель коэффициента а, b и c |
| Атрибут QPushButton \*value\_btn | область видимости - private.  Кнопка для отображения значения уравнения |
| Атрибут QPushButton \*root\_btn | область видимости - private.  Кнопка для отображения корней уравнения |
| Атрибут QLabel \*name\_x, \*delimeter\_x | область видимости - private.  Являются подсказками, которые показывают пользователю, что нужно вводить значение x |
| Атрибут QLineEdit \*x\_numerator, \*x\_denominator | область видимости - private.  Переменные, которые хранят введенные числитель и знаменатель значения x |
| Атрибут QPushButton \*print\_classic\_btn | область видимости - private.  Кнопка для отображения уравнения в классическом виде |
| Атрибут QPushButton \*print\_canonic\_btn | область видимости - private.  Кнопка для отображения уравнения в каноническом виде |
| Атрибут QLabel \*output | область видимости - private.  Поле, предназначенное для вывода результатов |
| Метод TInterface(QWidget \*parent = nullptr) | Область видимости public.  Конструктор класса |
| Метод ~TInterface() | Область видимости public.  Деструктор класса |
| Метод void answer(QString) | Тип формального параметра - Qstring, область видимости public.  Метод обрабатывает сигнал, поступивший из серверной части распределенного приложения |
| Метод void formRequest() | Формальных параметров нет, область видимости private.  Метод формирует запрос с клиентской части приложения. |
| Метод void request(QString) | Тип формального параметра - Qstring, область видимости private.  Метод преобразует сформированный запрос в сигнал. |

*Таблица 1. Класс Tinterface*

Класс TApplication

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут TCommunicator \*comm | объект класса Tcommunicator, область видимости - private. |
| Атрибут TInterface \*interface | объект класса TInterface, область видимости private |
| Метод void fromCommunicator(QByteArray) | Тип формального параметра - QByteArray, область видимости public.  Метод отображает в интерфейсе результат запроса. |
| Метод void toCommunicator(QString) | Тип формального параметра - Qstring, область видимости public.  Метод передает сообщение в серверную часть приложения. |

*Таблица 2. Класс Tapplication*

Класс TCommParams

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут QHostAddress rHost | область видимости — private.  Адрес приема |
| Атрибут quint16 rPort | область видимости — private.  Порт приема |
| Атрибут QHostAddress sHost | область видимости — private.  Адрес отправки |
| Атрибут quint16 sPort | область видимости — private.  Порт отправки |

*Таблица 3. Класс TcommParams*

Класс TCommunicator

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут bool ready | область видимости — private.  Принимает 1, если объект класса TCommunicator был успешно привязан к указанному адресу, 0 в обратном случае |
| Атрибут TCommParams params | область видимости — private.  Параметры для подключения. |
| Метод bool isReady() | Формальных параметров нет, область видимости public.  Метод возвращает переменную true типа bool, если ready равно 1, и false, если ready равно 0 |
| Метод void recieved(QByteArray) | Тип формального параметра - QByteArray, область видимости private.  Метод преобразует сформированный запрос в сигнал. |
| Метод void send(QByteArray) | Тип формального параметра - QByteArray, область видимости public.  Метод отправляет сформированную датаграмму при наличии соединения между серверной и клиентской частью. |
| Метод void recieve() | Формальных параметров нет, область видимости private.  Метод, при наличии датаграммы, принимает её. |

*Таблица 4. Класс TcommParams*

**Серверная часть распределенного приложения:**

Класс TPolinom

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут number a, b, c | область видимости – private, хранит значение коэффициентов. По умолчанию коэффициенты равны: a=1, b=2, c=1. |
| Атрибут EPrintMode printMode | область видимости – private, содержит вид уравнения для печати, который выбрал пользователь |
| Метод TPolinom(number,number,number) | Конструктор класса |
| Метод Void setPrintMethod(EPrintMode ePrintMethod) | Тип формального параметра - EPrintMode, область видимости public.  Метод устанавливает вид полинома, в котором его необходимо вывести (классический или канонический) |
| Методы getA(), getB(), getC() | Формальных параметров нет, тип number, область видимости public.  Предназначены для получения коэффициентов a, b, c. |
| Метод Int QuantityOfRoots() | Формальных параметров нет, тип возвращаемого значения - int, область видимости public.  Возвращает количество корней полинома |
| Метод number getValue(number x) | Тип формального параметра - number, тип возвращаемого значения - number, область видимости public.  Метод вычисляет и возвращает значение полинома. |
| Метод number Discriminant() | Формальных параметров нет, тип возвращаемого значения - number, область видимости private.  Возвращает значение дискриминанта |
| Метод Bool RootsCheck (number\*, int, number, number, number) | Тип возвращаемого значения - bool, область видимости private. Типы формальных параметров: указатель на массив с корнями (number\*), int количество корней, number коэффициенты a, b, c.  Метод позволяет определить, являются ли корни уравнения целыми числами. |
| Метод number \*Roots() | Формальных параметров нет, тип возвращаемого значения - number, область видимости public.  Возвращает указатель на массив с корнями. |
| Метод ostream& operator << (ostream& os, TPolinom& p) | Тип возвращаемого значения – ostream object, область видимости – public.  Данный метод выводит на экран полином в классической или канонической форме. |
| Метод friend QString& operator << (QString&, TPolinom&) | Тип возвращаемого значения – QString object, область видимости — public.  Является перегруженным оператором, который позволяет перевести тип полинома в строку. |

*Таблица 5. Класс Tpolinom*

Класс TRational

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут int numerator | область видимости - private.  Является числителем дроби (целое число) |
| Атрибут unsigned int denominator | область видимости - private.  Является знаменателем дроби (целое число, большее нуля) |
| Static QChar SEPARATOR | область видимости — private.  Является символом разделителя дроби |
| Метод TRational() | Конструктор класса |
| Метод TRational(const int&) | Конструктор класса |
| Метод unsigned int NOK(const TRational&) | Тип формального параметра - const TRationals&, тип возвращаемого значения - unsigned int, область видимости private.  Предназначен для вычисления наименьшего общего кратного. |
| Метод int NOD(const int&,const int&) | Тип формального параметра - const int&, тип возвращаемого значения - int, область видимости private.  Предназначен для вычисления наибольшего общего кратного. |
| Метод void decrease() | Формальных параметров нет, область видимости private.  Метод позволяет привести дробь к несократимому виду |
| Метод TRational operator+ (const TRational&) | Тип формального параметра - const TRationals&, тип возвращаемого значения - объект класса TRational,  область видимости public.  Перегрузка оператора сложения |
| Метод TRational operator\* (const TRational&) | Тип формального параметра - const TRational&,  тип возвращаемого значения - объект класса TRational,  область видимости public.  Перегрузка оператора умножения |
| Метод TRational operator\* (const int&) | Тип формального параметра - const int&, тип возвращаемого значения - объект класса TRational, область видимости public.  Перегрузка оператора умножения на число |
| Метод TRational operator/ (const TRational&) | Тип формального параметра - const TRational&, тип возвращаемого значения - объект класса TRational, область видимости public.  Перегрузка оператора деления |
| Методы:  bool operator > (int)  bool operator < (int)  bool operator == (int) | Тип формального параметра - int, тип возвращаемого значения - bool, область видимости public.  Перегрузка операторов сравнения (для целых чисел) |
| Метод bool operator == (TRational) | Тип формального параметра - TRational, тип возвращаемого значения - bool, область видимости public.  Перегрузка оператора сравнения (проверка сравнения дробей) |
| Метод TRational operator- (const TRational&) | Тип формального параметра - const TRational&,  тип возвращаемого значения - объект класса TRational,  область видимости public.  Перегрузка бинарного минуса |
| Метод TRational operator- () | Формальных параметров нет,  тип возвращаемого значения - объект класса TRational,  область видимости public.  Перегрузка унарного минуса |
| Метод friend TRational sqrt(TRational) | Тип формального параметра - TRational, тип возвращаемого значения - объект класса TRational, область видимости public.  Перегрузка оператора, который вычисляет квадратный корень |
| Метод friend ostream& operator<< (ostream& os, const TRational&) | Тип формальных параметров- ostream& os, const TRational&, тип возвращаемого значения - ostream object, область видимости – public.  Перегрузка оператора вывода |
| Метод friend istream& operator>> (istream& is, TRational&) | Тип формальных параметров - istream& is, TRational&, тип возвращаемого значения – istream object, область видимости – public.  Перегрузка оператора ввода |
| Метод TRational(const int&,const int&) | Конструктор класса |
| Метод friend QString& operator<< (QString& os, const TRational&) | Тип возвращаемого значения – QString object, область видимости — public.  Является перегруженным оператором, который позволяет перевести тип полинома в строку. |

*Таблица 6. Класс Trational*

Класс TApplication

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут TCommunicator \*comm | объект класса Tcommunicator, область видимости - private. |
| Метод void recieve(QByteArray) | Тип формального параметра - QByteArray, область видимости public.  Метод обрабатывает сообщение с запросом от клиентской части распределенного приложения. |

*Таблица 7. Класс Tapplication*

Класс TCommParams

|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут QHostAddress rHost | область видимости — private.  Адрес приема |
| Атрибут quint16 rPort | область видимости — private.  Порт приема |
| Атрибут QHostAddress sHost | область видимости — private.  Адрес отправки |
| Атрибут quint16 sPort | область видимости — private.  Порт отправки |

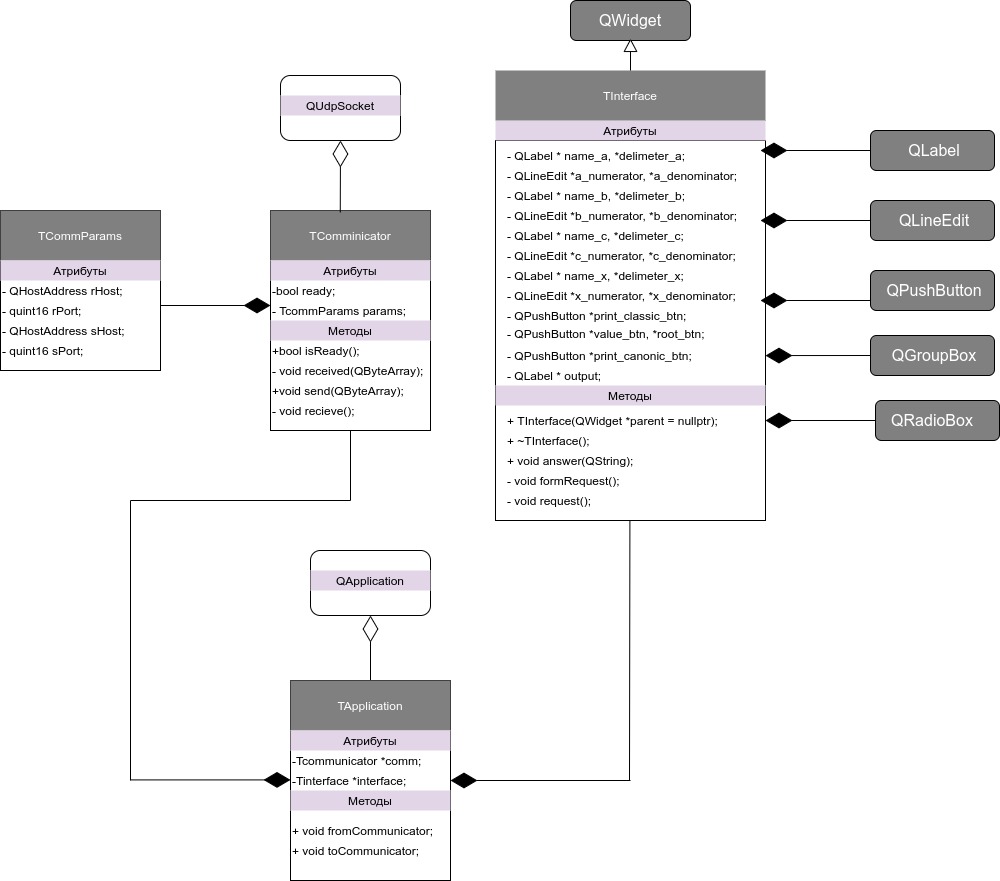
*Таблица 8. Класс TcommParams*

Класс TCommunicator

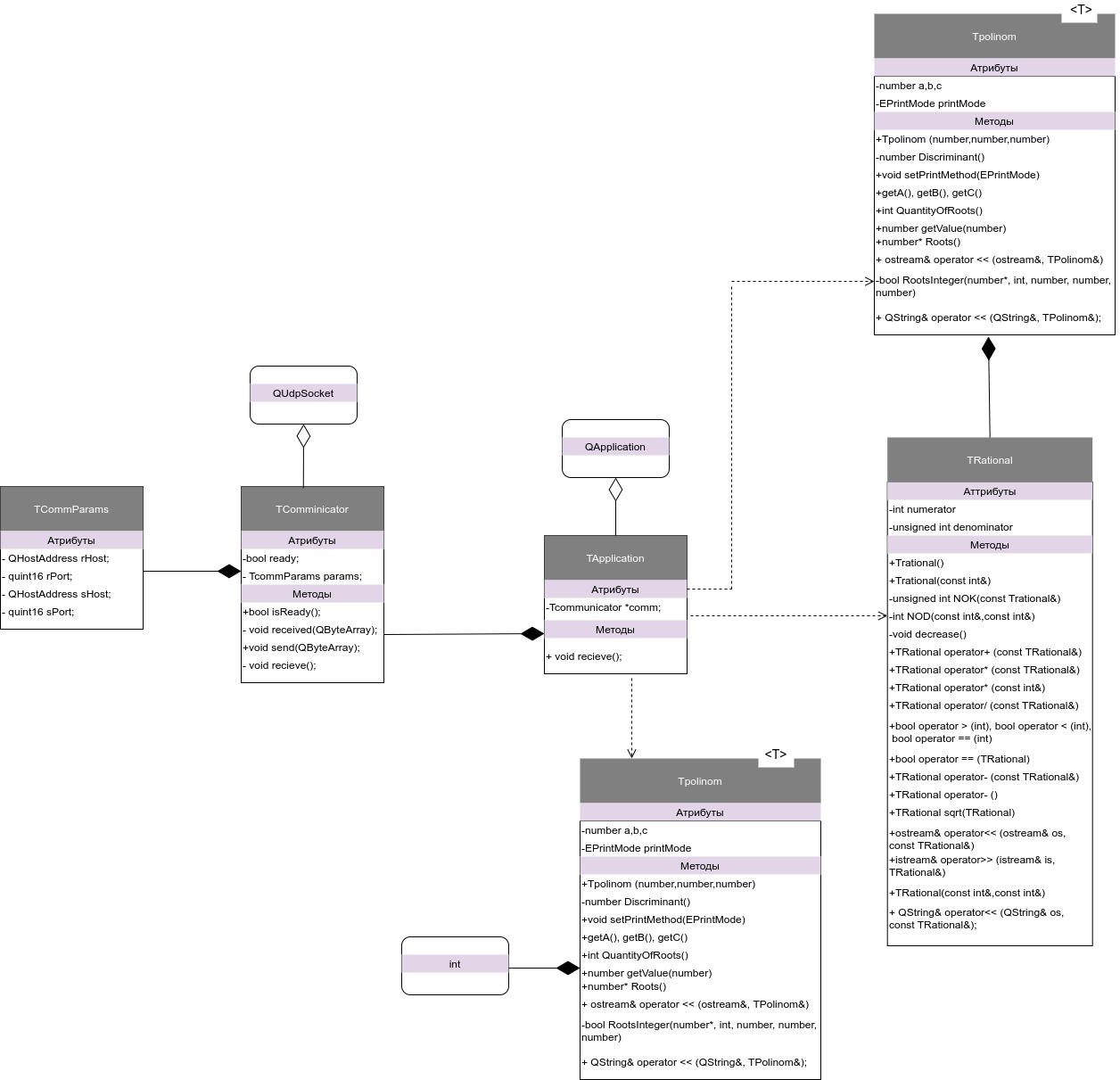
|  |  |
| --- | --- |
| Метод/атрибут | Описание |
| Атрибут bool ready | область видимости — private.  Принимает 1, если объект класса TCommunicator был успешно привязан к указанному адресу, 0 в обратном случае |
| Атрибут TCommParams params | область видимости — private.  Параметры для подключения. |
| Метод bool isReady() | Формальных параметров нет, область видимости public.  Метод возвращает переменную true типа bool, если ready равно 1, и false, если ready равно 0 |
| Метод void recieved(QByteArray) | Тип формального параметра - QByteArray, область видимости private.  Метод преобразует сформированный запрос в сигнал. |
| Метод void send(QByteArray) | Тип формального параметра - QByteArray, область видимости public.  Метод отправляет сформированную датаграмму при наличии соединения между серверной и клиентской частью. |
| Метод void recieve() | Формальных параметров нет, область видимости private.  Метод, при наличии датаграммы, принимает её. |

*Таблица 9. Класс TcommParams*

**Диаграмма классов**

**

*Рис.2. Реализация диаграммы классов клиентской части*

**

*Рис.3. Реализация диаграммы классов серверной части*

|  |  |
| --- | --- |
| Символ | Значение |
| + | public - открытый доступ |
| - | private - только из операций того же класса |
| # | protected - только из операций этого же класса и классов, создаваемых на его основе |

*Таблица 10. Обозначение аттрибутов и методов класса*

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

**Пример 1(для дробей):**

Исходные данные:

Коэффициенты:

a = ; b = ; c = -;

Ожидаемые данные:

x1=-2, x2=0.5

p(1/6)==-

Классический вид:

Канонический вид:

**Пример 2(для дробей):**

Исходные данные:

Коэффициенты:

a=, b=, c=

Ожидаемые данные:

x1=x2=-0.5

p(1/3)==

Классический вид:

Канонический вид:

**Пример 3(для целых чисел):**

Исходные данные:

Коэффициенты:

a=-1, b=4, c=0

Ожидаемые данные:

x1=4, x2=0

p(1)=-1\*1\*1+4\*1=3

Классический вид:

Канонический вид:

**Пример 4(для целых чисел):**

Исходные данные:

Коэффициенты:

a=2, b=4, c=7

Ожидаемые данные:

Корней нет

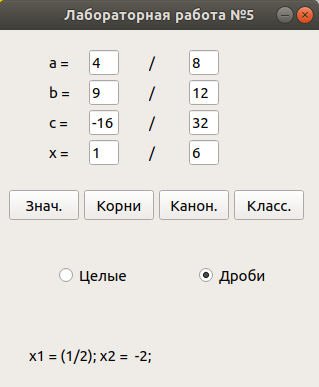
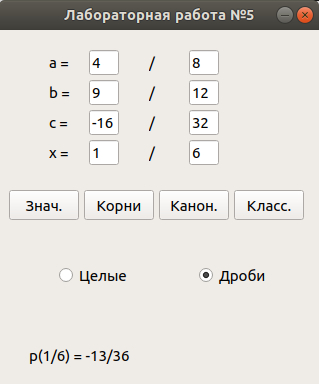
p(5)=2\*5\*5+4\*5+7=77

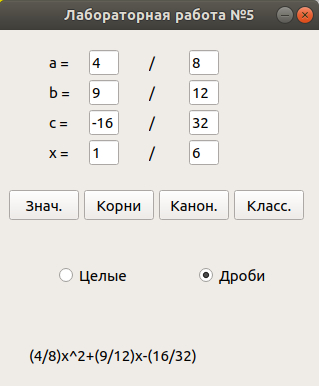
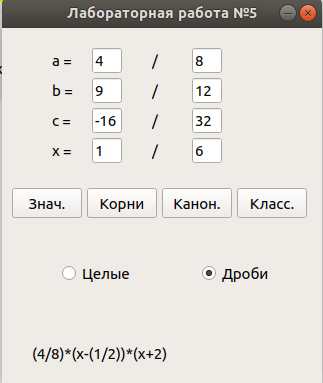
Классический вид:

Канонический вид: полином не имеет корней, поэтому его невозможно вывести в канонической форме.

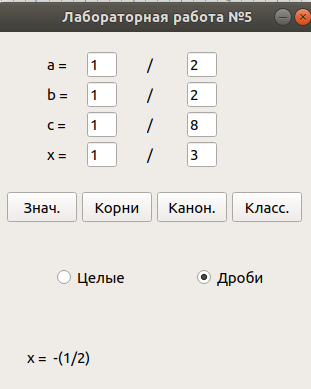
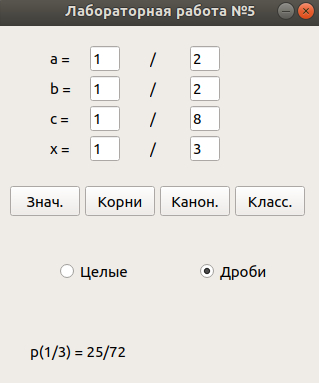
**Скриншоты программы на контрольных примерах**

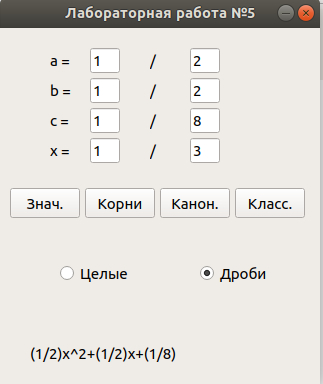
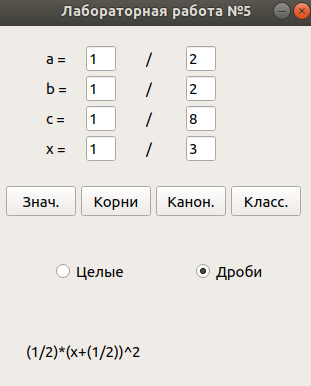
**Пример 1(для дробей):**



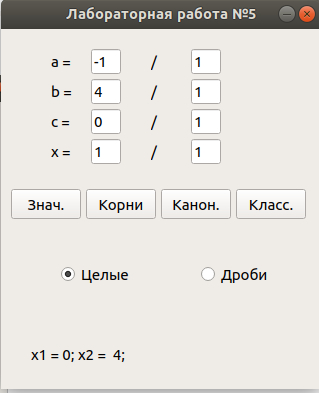
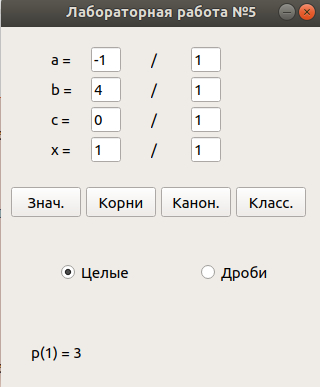


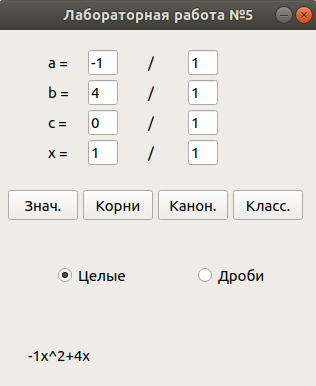
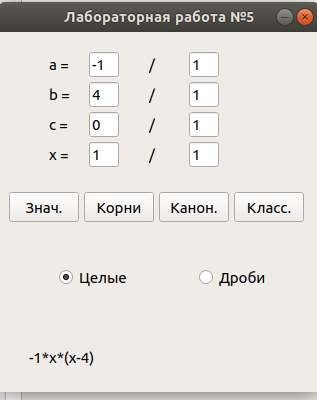
**Пример 2(для дробей):**



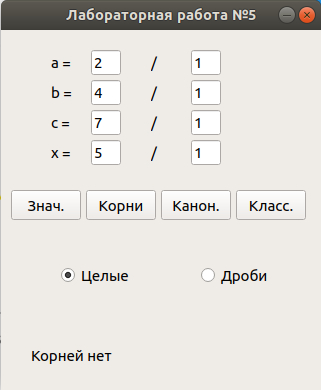
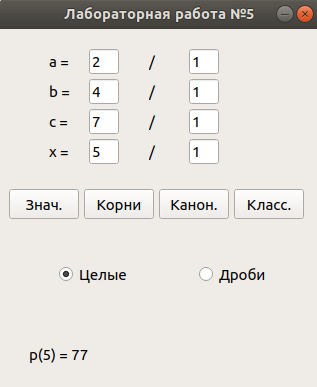


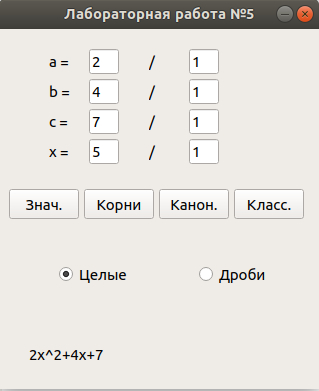
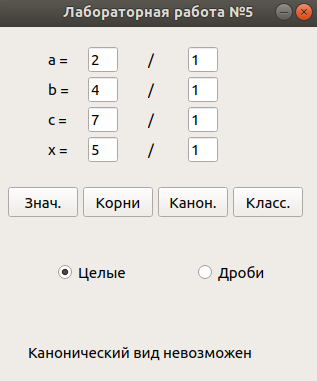
**Пример 3(для целых чисел):**

****

****

**Пример 4(для целых чисел):**

****

****

**Вывод**

В ходе данной лабораторной работы было создано 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 вычисления корней, значения и представления полинома 2-ой степени в классической и канонической формах на множестве рациональных чисел. Для этого было модифицировано описание класса **TPolinom,** который был представлен в виде параметризованного класса с параметром **number,** который может принимать значения как **int**, так и **TRational**.

Помимо этого, была создана диаграмма классов клиентской части(рис.2) и серверной части(рис.3 ), а также произведена отладка работы программы. Разработаны контрольные примеры с исходными и ожидаемыми данными, которые затем были протестированы в созданном распределенном приложении. Все результаты совпали.