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

ПРАКТИЧЕСКАЯ РАБОТА №5

по дисциплине «Объектно-ориентированное программирование»

ТЕМА: РАСПРЕДЕЛЁННОЕ ПРИЛОЖЕНИЕ

Беляев К.В.

Студенты гр. 3372 _____ Лазарев Ф.Н.

Преподаватель _____ Егоров С.С.

Санкт-Петербург
2024

Задание на Практическую работу

Студенты Беляев К.В., Лазарев Ф.Н.

Группа 3372

Исходные данные: создать распределенное приложение, включающее клиентскую и серверную части, взаимодействующие посредством сетевого обмена сообщениями.
Клиентская часть представляет собой GUI приложение, реализующее интерфейс аналогичный работе №4.
Серверная часть представляет собой консольное приложение, предназначенное для выполнения перечисленных в меню работы №3 функций над полиномом с комплексными элементами.

Спецификации классов

Таблица 1. Первичный протокол класса ClientApplication

Атрибуты		
идентификатор	область видимости	семантическое описание
*comm	private	Объект класса Communicator
*interface	private	Вывод меню в консоль
Методы		
идентификатор	область видимости	семантическое описание
ClientApplication	public	Конструктор класса
fromCommunicator	public slot	Принятие ответа из Серверной части
toCommunicator	public slot	Отправление запроса в Серверную часть

Таблица 2. Первичный протокол класса ServerApplication

Атрибуты		
идентификатор	область видимости	семантическое описание
*comm	private	Объект класса Communicator
*polynom	private	Объект класса Polynom
*roots	private	Массив корней полинома
An	private	Старший коэффициент
rootsAmount	private	Кол-во корней
Методы		
идентификатор	область видимости	семантическое описание
pushBack	public	Добавление нового корня
recieve	public slot	Работа с запросами из Клиентской части

Таблица 3. Первичный протокол класса TCommunicator

Атрибуты		
идентификатор	область видимости	семантическое описание
ready	private	Готовность принять запрос
params	private	Параметры коммуникатора
Методы		
идентификатор	область видимости	семантическое описание
TCommunicator	public	Объект класса
recieved	public signal	Работа с запросами
send	public slot	Получение запроса из клиентской части
recieve	private slot	Получение ответа из серверной части

Таблица 4. Протокол класса Array

Атрибуты(старые)			
идентификатор	тип	область видимости	семантическое описание
length	int	private	Целочисленная длина массива
arr	number*	private	Указатель на первый элемент массива
Методы(старые)			
идентификатор	область видимости		семантическое описание
Array	public		Конструктор класса. Создает массив заданной длины, по умолчанию – 0
~Array	public		Деструктор класса
getLength	public		Получение длины массива

fill	public	Заполнение массива числами с консоли
resize	public	Изменение размера массива
changeElement	public	Изменение выбранного элемента числом с консоли
printArray	public	Вывод массива в консоль
averageValue	public	Подсчет среднего значения элементов массива
SKO	public	Подсчет СКО элементов массива
shakerSort	public	Сортировка массива по убыванию – если передается параметр 1, по возрастанию – если передается 0 или не передается ничего

Таблица 5. Протокол класса TComplex

Атрибуты(старые)			
идентификатор	тип	область видимости	семантическое описание
re	double	private	Вещественная часть комплексного числа
im	double	private	Мнимая часть комплексного числа
separator	Qchar	private	Разделение аргументов в запросе
Методы(старые)			
идентификатор	область видимости		семантическое описание
TComplex()	public		Конструктор класса по умолчанию

TComplex(double re, double im)	public	Конструктора класса, принимающий вещественное и мнимое части комплексного числа
TComplex(double re)	public	Конструктор класса, принимающий вещественную часть комплексного числа
getRe	public	Получение вещественной части комплексного числа
getIm	public	Получение мнимой части комплексного числа
module	public	Вычисление модуля комплексного числа
operator+	public	Оператор сложения
operator-	public	Оператор вычитания
operator/	public	Оператор деления
operator*	public	Оператор умножения
operator+=		Оператор сложения с присваиванием
operator-=	public	Оператор вычитания с присваиванием
operator/=	public	Оператор деления с присваиванием
operator*=	public	Оператор умножения с присваиванием
operator=	public	Оператор присваивания
operator==	public	Оператор «равно»
operator!=	public	Оператор «неравно»
operator<	public	Оператор «меньше»
operator>	public	Оператор «больше»

pow	public	Вычисление корня из комплексного числа
Методы(новые)		
идентификатор	область видимости	семантическое описание
setSeparator	public	Установление разделителя аргументов
Operator QString()	public	Преобразование в QString
operator >>	public	Оператор “>>”для работы с потоком ввода
operator <<	public	Оператор “<<” для работы с потоком вывода
operator >>	public	Оператор “>>” для работы с QByteArray

Таблица 6. Протокол класса Polynom

Атрибуты(старые)			
идентификатор	тип	область видимости	семантическое описание
roots	Number*	private	Массив корней полинома
coefficients	Number*	private	Массив коэффициентов полинами
degree	int	private	Степень полинома
Методы(старые)			
идентификатор	область видимости		семантическое описание
Polynom()	public		Конструктор класса по умолчанию

polynomWithDegrees	public	Вывод полинома в каноническом виде
polynomWithRoots	public	Вывод полинома с корнями
valueAtPoint	public	

Таблица 7. Протокол класса Tinterface

Атрибуты(старые)			
идентификатор	тип	область видимости	семантическое описание
coeffsLabel	QLabel *	private	Лэйбл коэффициента
imIndicator	QLabel *	private	Лэйбл индикатора i
reCoeffsLE	QLineEdit *	private	Поле ввода действительной части
imCoeffsLE	QLineEdit *	private	Поле ввода мнимой части
addRootBTN	QPushButton *	private	Кнопка добавления корня
changeRootBTN	QPushButton *	private	Кнопка изменения корня
changeRootLineEdit	QLineEdit *	private	Поле ввода индекса корня
leadingCoeff	QLabel *	private	Лэйбл старшего коэффициента
reLeadCoeff	QLineEdit *	private	Поле ввода действительной части старшего коэффициента

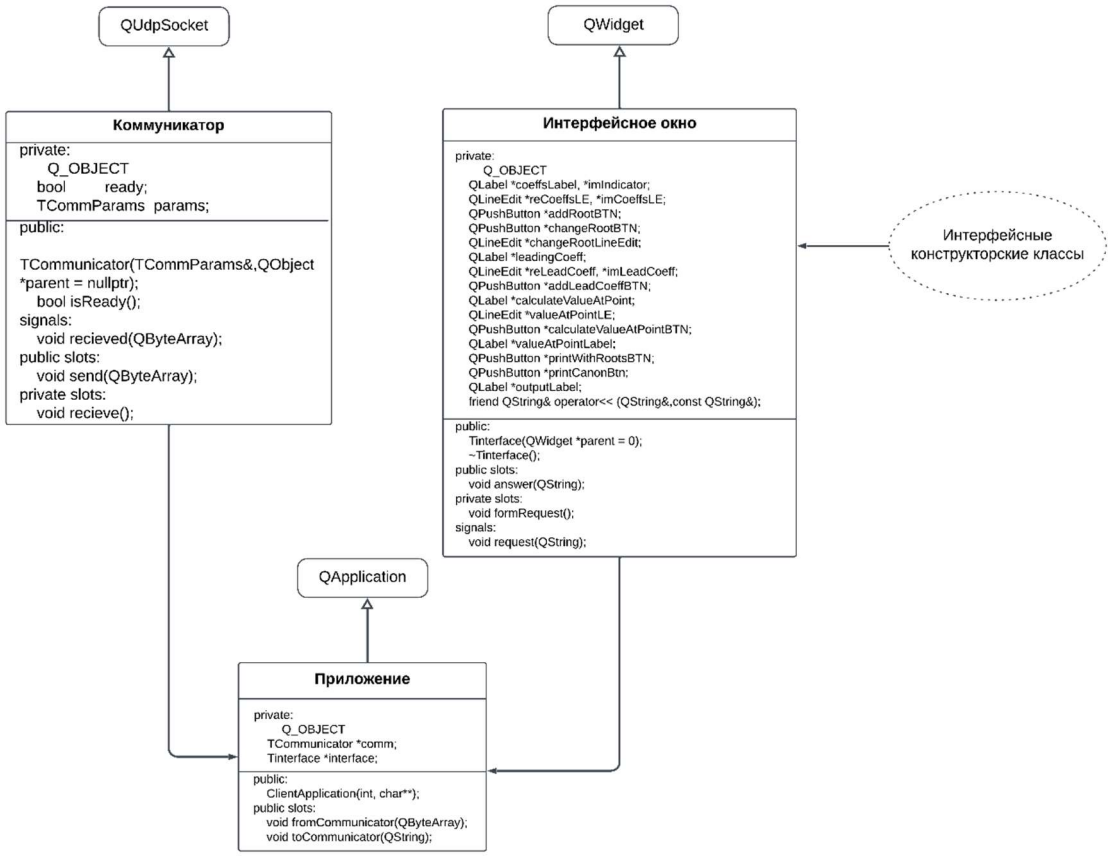
imLeadCoeff	QLineEdit *	private	Поле ввода мнимой части старшего коэффициента
addLeadCoeffBTN	QPushButton *	private	Кнопка добавления старшего коэффициента
calculateValueAtPoint	QLabel *	private	Лэйбл значения полинома в точке
valueAtPointLE	QLineEdit *	private	Поле ввода точки
calculateValueAtPointBTN	QPushButton *	private	Кнопка вычисления значения полинома в точке
valueAtPointLabel	QLabel *	private	Лэйбл значения полинома в точке
printWithRootsBTN	QPushButton *	private	Кнопка вывода полинома с корнями
printCanonBtn	QPushButton *	private	Кнопка вывода полинома в каноническом виде
outputLabel	QLabel *	private	Лэйбл с полиномом
polynom	Polynom *	private	Указатель на объект полинома
roots	number *	private	Массив с корнями полинома

rootsAmount	int	private	Количество корней полинома
An	number	private	Старший коэффициент полинома
Методы(старые)			
идентификатор	область видимости		семантическое описание
Tinterface	public		Конструктор класса
~Tinterface	public		Деструктор класса
Методы(новые)			
идентификатор	область видимости		семантическое описание
answer	public slot		Вывод ответа из серверной части
formRequest	private slots		Формирование запроса
request	public signal		Отправление запроса в серверную часть

Диаграмма классов

На рисунке 1 представлена диаграмма классов, дополненная атрибутами и методами.

Клиентская часть



Серверная часть

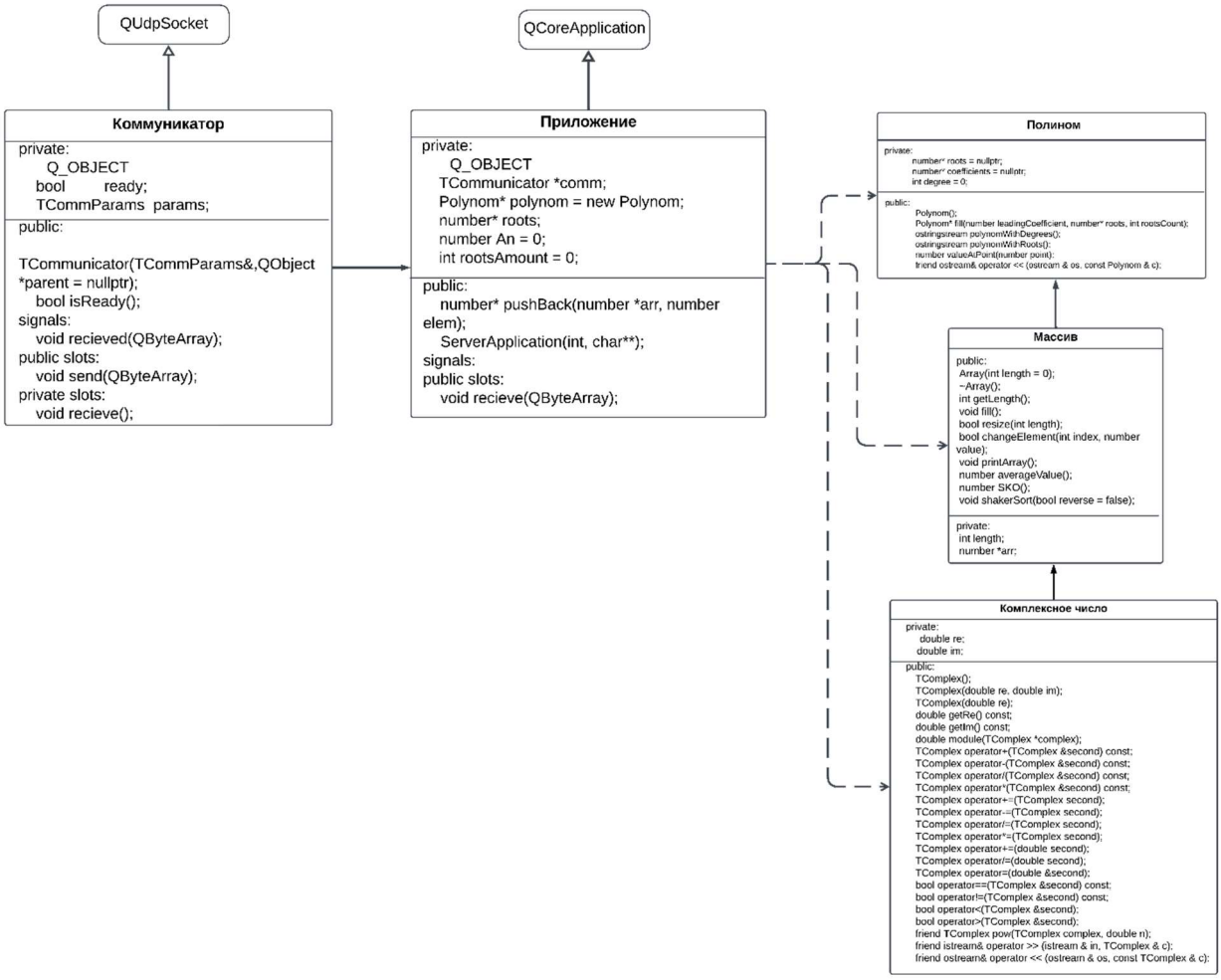


Рисунок 1 – Диаграмма классов.

Описание контрольного примера с исходными и ожидаемыми расчетными данными

1. При создании полинома вводится коэффициент an : 1 1 и корни полинома: 2 3 4 5 6 7

2. Создается полином p(x) = (1+1i)x3 + (3-27i)x2 + (-139+85i)x + (279+113i)

3. Изменяется коэффициент an на (3 + 2i), при вводе 3 2. Полином принимает вид: p(x) = (3+2i)x3 +(-6-69i)x2 + (-305+282i)x + (754+143i)

4. Предусмотрено изменение корня вводом индекса корня и новым значением: 1 и 3 4 соответственно. Полином меняет вид на: p(x) = (3+2i)x3 +(-5-64i)x2 + (-263+232i)x + (585+130i)

5. Для вычисления значения функции в точке, необходимо ввести x: 5. Значение: p(5) =-480-60i

СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ НА КОНТРОЛЬНЫХ ПРИМЕРАХ

После запуска программы на экране появляется окно, с кнопками, полями ввода и полями с текстом на рисунке 2.

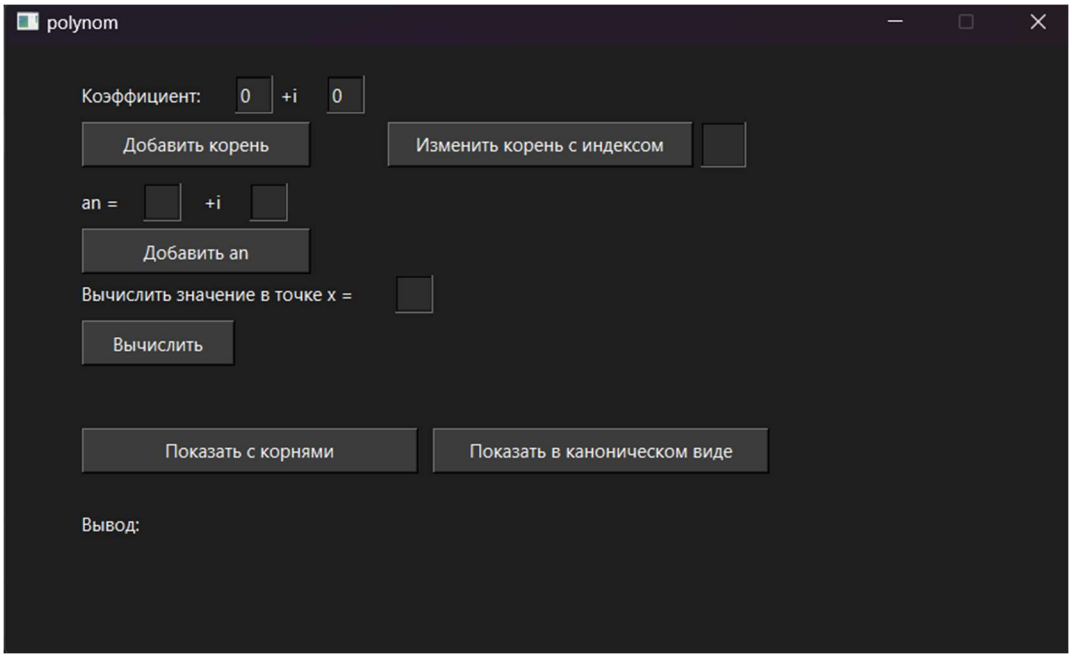


Рисунок 2 – Запуск программы

Для создания полинома нужно коэффициент an и корни полинома. Вводятся коэффициент an и корни полинома в соответствующие поля. Дальше нужно нажать кнопки «Добавить корень» и «Добавить an». Кнопку «Добавить корень» нужно нажимать после каждого введенного нового корня. На рисунке 3 показан ввод коэффициента an и корней полинома.

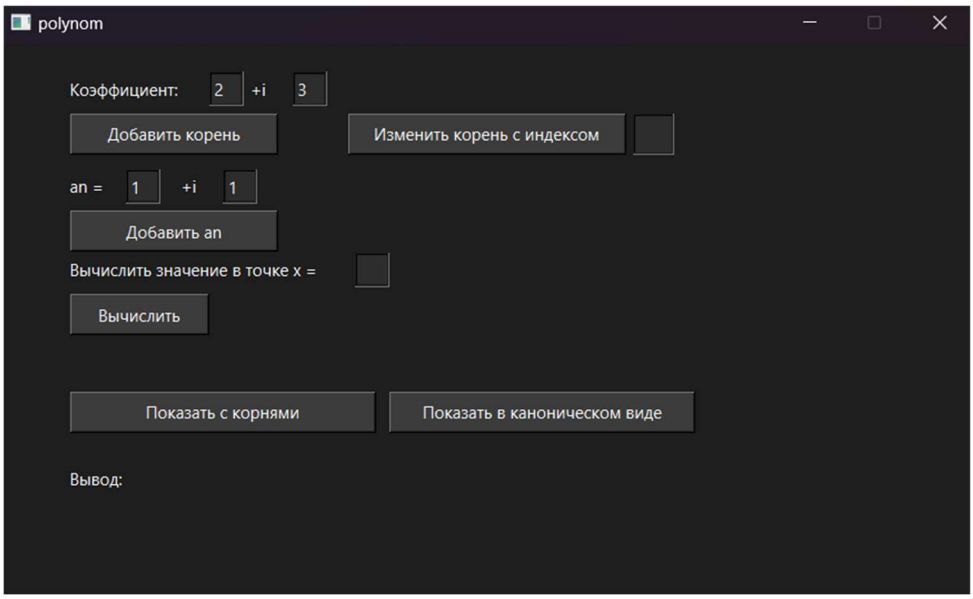


Рисунок 3 – Ввод корня и коэффициента из контрольного примера

Чтобы вывести полином, нажмем кнопку «Показать с корнями» или «Показать в каноническом виде». В окне появится полином с введёнными на предыдущем шаге корнями и коэффициентом (рисунок 4).

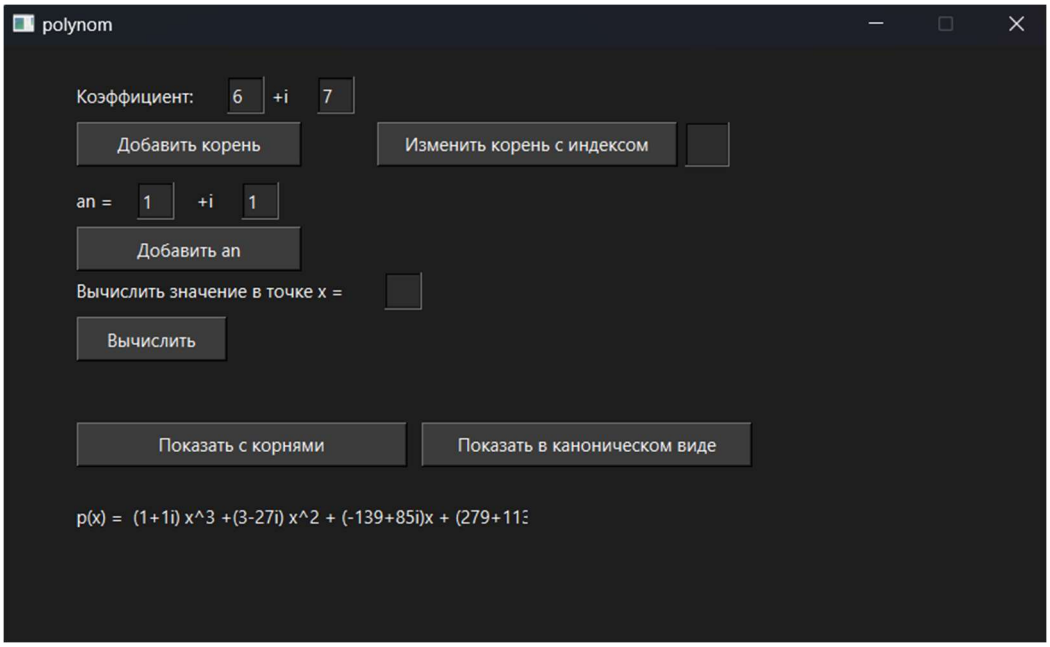


Рисунок 4 – Вывод полинома на экран

Изменим старший коэффициент полинома. Введем 3 2 в поле ввода коэффициента и нажмем кнопку добавить an. На рисунке 5 показаны результаты работы программы.

Рисунок 5 – Изменение старшего коэффициента

Теперь изменим один из корней. Вводим в соответствующие поля индекс корня и новое значение и нажимаем кнопку «Изменить корень с индексом». Результат замены корня видно на рисунке 6.

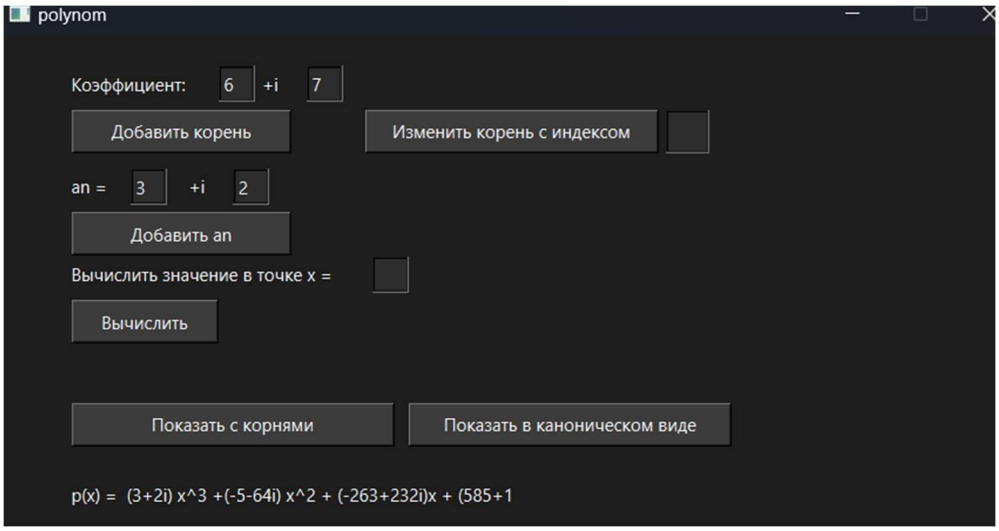


Рисунок 6 – Изменение корня полинома

Чтобы вычислить значение полинома в данной точке, необходимо ввести «4» и нажать клавишу Enter. На экране появится запрос точки x. На рисунке 7 показан результат работы программы.

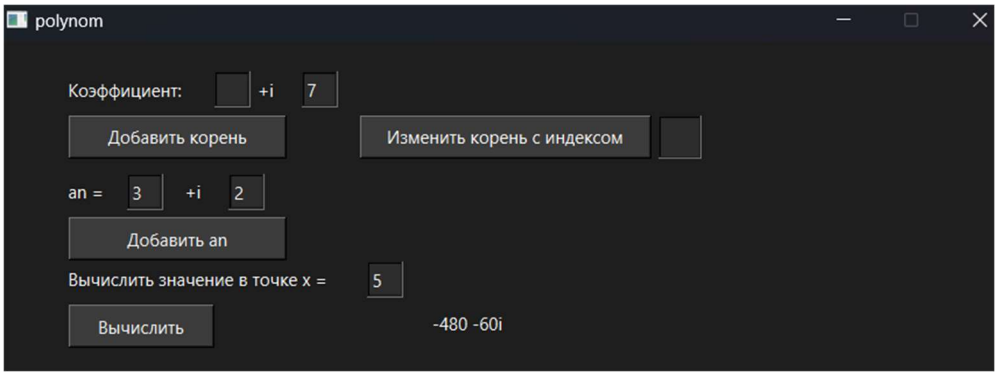


Рисунок 7 – Вычисленное значение

Наконец, чтобы выйти из программы, нужно нажать на крестик в правом верхнем углу, и программа автоматически закроется.

ВЫВОДЫ ПО ВЫПОЛНЕНИЮ РАБОТЫ

В рамках данной практической работы была реализована и отлажена программа, предназначенная для создания GUI приложения с клиентской и серверной частью, реализующего функции перечисленные в описании работы №4. Также был разработан контрольный пример для проведения проверки, с чем программа справилась успешно.