**Министерство образования и науки Российской Федерации**

**Пермское федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский Национальный Исследовательский Политехнический Университет»**

**Электротехнический факультет**

**Кафедра «Информационные технологии и автоматизированные системы»**

**ОТЧЁТ**

По творческой работе на тему

«Разработка калькулятора»

Выполнил студент группы РИС-20-1б

Кукшинова А.А.

Проверил доцент кафедры ИТАС

Полякова О.А.

Пермь 2021

**Цель работы** –продемонстрировать достижения, полученных студентом за время обучения на первом курсе.

**Постановка задачи**

Задача – реализовать на языке C++ с использованием технологии Qt проект, представляющих собой приложение с графическим интерфейсом:

1. Узкоспециализированный калькулятор;

Исходные данные:

1. Специализация калькулятора: расчёт многочленов:
   1. Сложение многочленов;
   2. Вычитание многочленов;
   3. Умножение многочленов;
   4. Деление многочленов;
   5. Возведение в степень
   6. Получение числа при подстановке значения *x*;

**Анализ задачи**

**Часть №1: калькулятор многочленов:**

**Одночлен** – это одно число, которое может быть представлено как в виде обыкновенного числа, так и в виде выражения, представленным в виде произведение числа на некоторую неизвестную переменную (например, *x*). Несколько примеров одночленов приведены ниже:

**Многочлен** – это алгебраическая сумма нескольких одночленов, одночлен – частный случай многочлена, содержащий один элемент. Примеры представлены ниже:

Как правило, элементы многочленов записываются слева направо по мере убывания степени основания переменной *x*. Такой способ представления многочленов является наиболее удобным.

На основе данного факта, общий вид многочленов выглядит следующим образом:

Где, *A, B, C* …. (другие больши́е буквы) – это коэффициенты многочлена, *n* – старшая степень многочлена.

Таким образом, многочлен можно представлять в виде коэффициентов – массива чисел, с которыми можно производить различные преобразования без необходимости иметь значение х. Например:

Подобную абстракцию многочлена имеет смысл вывести в реализацию отдельного класса, экземпляр которого будет представлять собой абстрактный многочлен, с которым в дальнейшем будет происходить работа.

#include <QString>

class Polynom

{

private:

int degree;

double\* coef;

QString varName;

}

**Реализация математических операторов для работы с многочленами:**

Возможности языка программирования C++ позволяют перегружать операторы сложения, вычитания, умножения и деления, ими имеет смысл воспользоваться.

**Оператор сложения «+» и вычитания «-»:**

Оператор сложения является самым простым с точки зрения технической реализации, так как достаточно почле́нно произвести сложение коэффициентов двух многочленов, например:

В свою очередь, операция вычитания является противоположной операции сложения, действия те же, но только со знаком минус.

Соответственно, реализация операций сложения и вычитания будут являться одинаковой реализацией в коде.

В коде, реализация операторов сложения и вычитания выглядит следующим образом:

Polynom operator + (const Polynom& p1, const Polynom& p2)

{

return plus\_minus(p1, p2, plusu);

}

Polynom operator - (const Polynom& p1, const Polynom& p2)

{

return plus\_minus(p1, p2, minusu);

}

double plusu(double a, double b)

{

return a + b;

}

double minusu(double a, double b)

{

return a - b;

}

Polynom plus\_minus(const Polynom& p1, const Polynom& p2, double op(double, double))

{

Polynom

\*pmax,

\*pmin;

if (p1.degree > p2.degree)

{

pmax = (Polynom\*)&p1;

pmin = (Polynom\*)&p2;

}

else

{

pmax = (Polynom\*)&p2;

pmin = (Polynom\*)&p1;

}

int min\_degree = pmin->degree;

int max\_degree = pmax->degree;

Polynom res(max\_degree);

for (int i = 0; i < max\_degree; i++)

{

if (i < min\_degree)

res.coef[i] = op(pmax->coef[i], pmin->coef[i]);

else

res.coef[i] = pmax->coef[i];

}

return res;

}

**Оператор умножения «\*»:**

Умножение многочленов происходит почле́нно следующим образом: каждый член первого многочлена перемножается со всеми членами второго многочлена.

Таким образом, старшая степень полученного многочлена вырастет, соответственно, новый многочлен будет иметь больше коэффициентов на выходе, их количество будет равно сумме старших степеней двух перемножаемых многочленов.

Поскольку происходит работа всех коэффициентов со всеми, их можно перебрать с помощью двух вложенных циклов for.

Реализация операторов выглядит следующим образом:

Polynom operator \* (const Polynom& p1, const Polynom& p2)

{

Polynom res(p1.degree + p2.degree - 1);

for (int i = 0; i < p1.degree; i++)

for (int j = 0; j < p2.degree; j++)

res.coef[i + j] += p1.coef[i] \* p2.coef[j];

return res;

}

**Оператор деления «/»:**

Оператор деления является самым сложным с точки зрения оператора с точки зрения понимания и реализации. Поскольку работа идёт с многочленами, то всплывает следующее ограничение, реализуемо только деление многочленов без остатка, так как остаток никак не получится выразить в виде другого многочлена.

Одним из способов деления многочленов является деление уголком, рассмотрим работу алгоритма на примере:

Частное будет найдено в ходе выполнения следующих шагов:

1. Делится первый элемент делимого многочлена на старший элемент многочлена-делителя и записываем в ответ:

Ответ:

1. Полученное частное умножается на делитель:
2. Вычисленное произведение вычитаем из делимого многочлена:
3. Действия повторяются заново, пока в остатке не останется выражение, которое при делении без остатка не вернёт 0.

Ответ:

Ответ:

В итоге, получается ответ в виде многочлена, однако остаток от деления никуда записать не получится, потому что выражение -123/(x-3) невозможно привести к стандартному виду.

Реализация в коде на языке программирования C++ выглядит следующим образом:

Polynom operator / (const Polynom& p1, const Polynom& p2)

{

Polynom temp = p1;

int rdeg = temp.degree - p2.degree + 1;

Polynom res(rdeg);

for (int i = 0; i < rdeg; i++)

{

res.coef[rdeg - i - 1] =

temp.coef[temp.degree - i - 1] / p2.coef[p2.degree - 1];

for (int j = 0; j < p2.degree; j++)

temp.coef[temp.degree - j - i - 1] -=

p2.coef[p2.degree - j - 1] \* res.coef[rdeg - i - 1];

}

temp.reduce();

return res;

}

**Получение значения при подстановке x:**

Для того, чтобы получить число зная х, достаточно к результату равному нулю прибавлять число равному х, возведённому на индекс коэффициента, который в свою очередь будет умножен на сам коэффициент.

Реализация метода в коде:

double Polynom::GetValueFromX(double x)

{

double value = 0;

for (int i = degree - 1; i >= 0; i--)

value += pow(x, i) \* coef[i];

return value;

}

Функция, возвращающая строку для вывода многочлена:

Для того, чтобы выводить ответ, необходимо преобразовать имеющиеся коэффициенты в строку, который обязан содержать переменную х, степень каждого элемента (исключая нулевые элементы, х в первой в нулевой с степени, правильную расстановку знаков и т.п.). Реализовано два метода:

1. Метод, реализующий вывод самого многочлена;
2. Метод, реализующий вывод многочлена с его именем;

QString Polynom::ToString() const

{

return varName + ": " + ToStringWithoutName();

}

QString Polynom::ToStringWithoutName() const

{

QString str = "";

bool isFirstCoef = true;

for (int i = degree - 1; i >= 0; i--)

{

if (abs(coef[i] - 0) > pow(10, -6))

{

QString strCoef = (abs(coef[i]) == 1)? ""

: QString::number(abs(coef[i]));

QString strOper =

(coef[i] > 0)? (isFirstCoef)? "" : "+" : "-";

if (i == 1)

str += strOper + strCoef + "x ";

else if (i == 0)

str += strOper + QString::number(abs(coef[i]));

else

str += strOper + strCoef + "x^" +

QString::number(i) + " ";

isFirstCoef = false;

}

}

return str;

}

**Реализация взаимодействия пользователя с компонентами приложения посредством использования графического интерфейса:**

Интерфейс калькулятора реализован средствами технологии Qt и IDE Qt Creator, позволяющим верстать элементы графического интерфейса без необходимости писать код вручную.

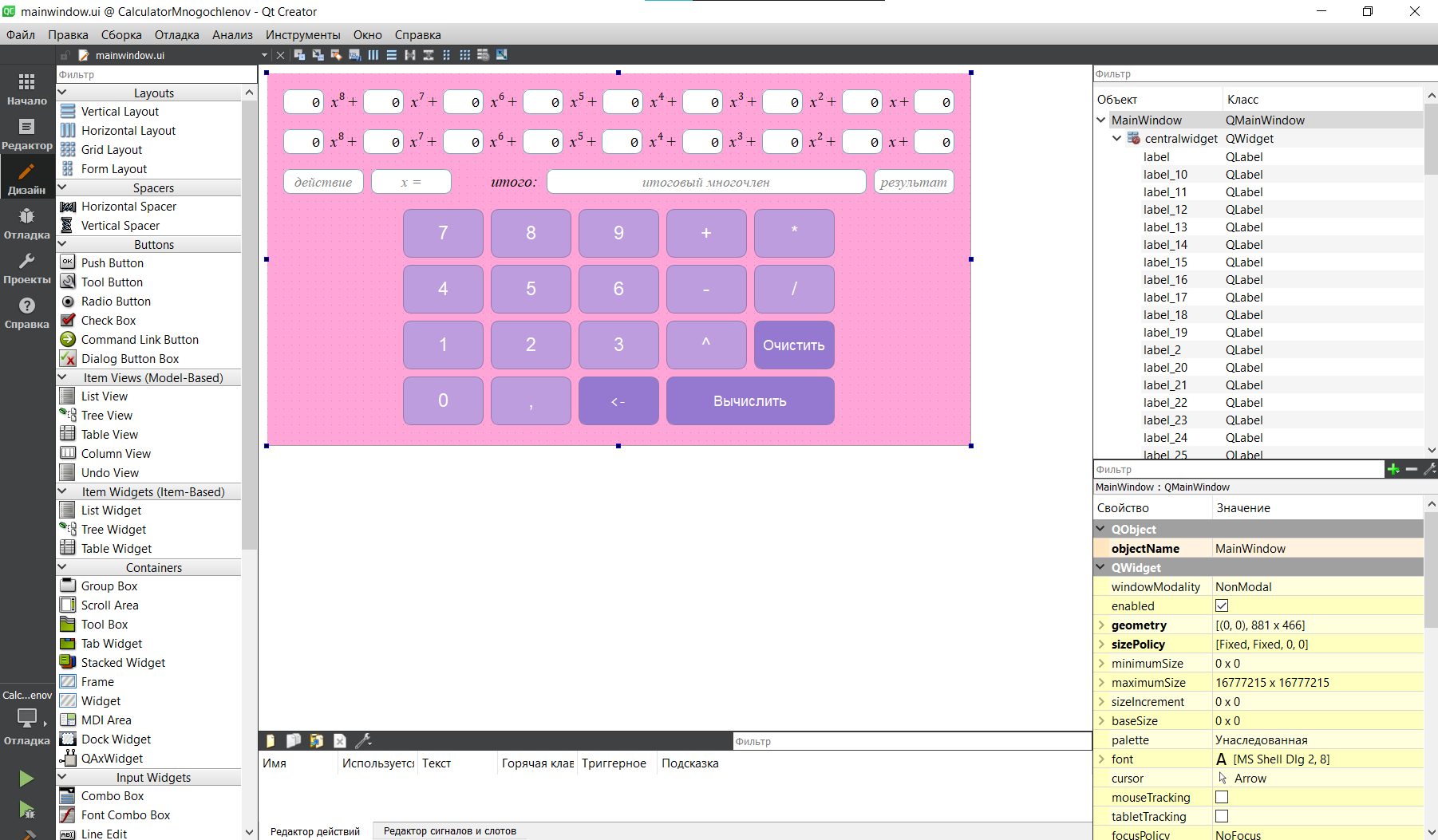


Рисунок - конструктор графического интерфейса в Qt Creator

Функционал интерфейса калькулятора:

* Ввод данных в приложения посредством ввода символов в текстовые поля;
* Отображение вводимых данных в режиме реального времени;
* Вывод полученных данных после команд, реализованных в виде рычагов управления приложением, посредством нажатия кнопок;

Интерфейс калькулятора поделён на 6 логических блоков:

1. Блоки с объявлением коэффициентов для каждой переменной;
2. Блок, отвечающий за ввод математических выражений;
3. Блок, отвечающий за ввод значения х;
4. Блок вывода итогового многочлена;
5. Блок вывода полученного ответа;

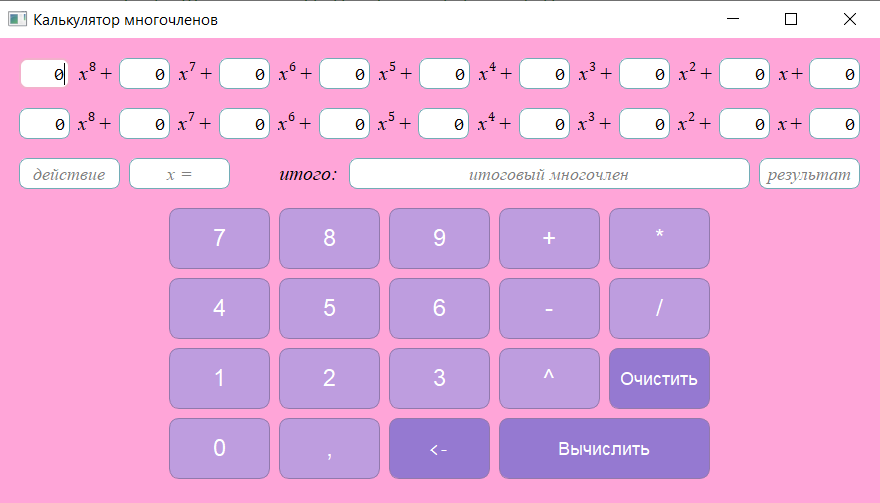


Рисунок - логическое разделение интерфейса калькулятора

Модель интерфейса представленная выше имеет следующие преимущества:

1. Отсутствие необходимости частотного использования мыши при вводе и редактировании многочленов или выражения;
2. Универсальность и множественное переиспользование одного многочлена в выражении: достаточно объявить многочлен один раз для того, чтобы была возможность использовать его неограниченное количество раз;

UML-диаграмма приложения:

