

Колледж космического машиностроения и технологий

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «**Прикладное программирование**»**

**Тема: «**Разработка приложения

Калькулятор с Обратной Польской Записью**»**

Выполнил студент

Мырза Н. Ю.

Группа П1-18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

Королёв 2021 г.

Оглавление

[Введение 3](#_Toc70506742)

[1 Теоретическая часть 4](#_Toc70506743)

[1.1 Описание предметной области 4](#_Toc70506744)

[1.2 Описание существующих разработок 12](#_Toc70506745)

[2 Проектная часть 15](#_Toc70506746)

[2.1 Диаграмма прецедентов 15](#_Toc70506747)

[2.2 Выбор инструментов 15](#_Toc70506748)

[2.2.1 Язык программирования 16](#_Toc70506749)

[2.2.2 Среда разработки программного обеспечения 19](#_Toc70506750)

[2.3 Проектирование сценария 21](#_Toc70506751)

[2.4 Диаграммы классов 22](#_Toc70506752)

[2.5 Описание главного модуля. 22](#_Toc70506753)

[2.6 Описание спецификаций к модулям 25](#_Toc70506754)

[2.7 Описание модулей 25](#_Toc70506755)

[2.8 Описание тестовых наборов модулей 29](#_Toc70506756)

[3 Эксплуатационная часть 30](#_Toc70506757)

[3.1 Руководство оператора 30](#_Toc70506758)

[3.1.1 Назначение программы 30](#_Toc70506759)

[3.1.2 Условия выполнения программы 30](#_Toc70506760)

[3.1.3 Выполнение программы 30](#_Toc70506761)

[3.2 To-Do лист 32](#_Toc70506762)

[Заключение 33](#_Toc70506763)

[Список литературы и интернет ресурсов 34](#_Toc70506764)

# Введение

Данный курсовой проект заключается в разработке калькулятора, основанного на Обратной Польской Записи . Разработка программы будет происходить на языке программирования C# с использованием WPF (Windows Presentation Foundation) .

В первой части будут рассмотрены предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты разработки программы и модули программы.

В третьей части описаны взаимодействие пользователя с программой и сообщения оператору.

В заключительной части будут приведены общие выводы по проекту.

# Теоретическая часть

## Описание предметной области

**Калькулятор** ([лат.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) calculātor «счётчик») — [электроное](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) [вычислительное устройство](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) для выполнения [операций над числами](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8_%D0%BD%D0%B0%D0%B4_%D1%87%D0%B8%D1%81%D0%BB%D0%B0%D0%BC%D0%B8) или [алгебраическими фор­мулами](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F).

Калькулятор заменил [механические вычислительные устройства](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0), такие, как [абаки](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D0%B0%D0%BA_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [счёты](https://ru.wikipedia.org/wiki/%D0%A1%D1%87%D1%91%D1%82%D1%8B), [логарифмические линейки](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BA%D0%B0), механические или электромеханические [арифмометры](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%BE%D0%BC%D0%B5%D1%82%D1%80), а также математические таблицы (прежде всего — [таблицы логарифмов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%BE%D0%B2)).

**Виды калькуляторов**

В зависимости от возможностей и целевой сферы применения калькуляторы делятся на: простейшие, бухгалтерские, инженерные (научные), финансовые. В отдельные классы обычно выделяют [программируемые калькуляторы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80), дающие возможность выполнения сложных вычислений по предварительно заложенной программе, а также графические — поддерживающие построение и отображение [графиков](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D0%BA_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). Специализированные калькуляторы предназначены для выполнения вычислений в достаточно узкой сфере ([финансовые](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BD%D0%B0%D0%BD%D1%81%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80), строительные и т. п.)

По исполнению калькуляторы могут быть настольными или компактными (карманными). Отдельные модели имеют интерфейсы для подключения [персонального компьютера](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [печатающего устройства](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%82%D0%B5%D1%80), внешнего модуля памяти или иных внешних устройств.

Современные [персональные компьютеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), [сотовые телефоны](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD), [КПК](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BC%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)(Карманный Персональный Компьютер) и даже [наручные часы](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%80%D1%83%D1%87%D0%BD%D1%8B%D0%B5_%D1%87%D0%B0%D1%81%D1%8B) могут иметь [программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), выполняющие функции калькулятора.

Термином «калькулятор» также называются специализированные программы, встраиваемые в веб-сайты (например, «*калькулятор калорий*», «*калькулятор размеров одежды*» и пр.) или в бытовую технику (например, *простой медицинский калькулятор* может встраиваться в [спортивный тренажёр](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%BE%D1%80%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B5%D0%BD%D0%B0%D0%B6%D0%B5%D1%80)).

**Простейшие** калькуляторы предназначены для выполнения только ординарных арифметических расчётов. Имеют небольшие размеры и вес, обычно не более одного дополнительного регистра памяти и минимальное число функций (как правило, только [арифметические операции](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D0%BA%D0%B0) и, возможно, одна-две функции, такие, как извлечение [квадратного корня](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%80%D0%B5%D0%BD%D1%8C), [обратная функция](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F), смена знака или вычисление процентов). Не поддерживают представление чисел с плавающей запятой. Как правило, имеют 8-разрядный [семисегментный индикатор](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%B8%D1%81%D0%B5%D0%B3%D0%BC%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D0%B4%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80" \o "Семисегментный индикатор), диапазон представляемых чисел: от ±10−7 до ±(108−1);

**Инженерные** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *scientific*, изредка употребляется русская [калька](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D0%BB%D1%8C%D0%BA%D0%B0_(%D0%BB%D0%B5%D0%BA%D1%81%D0%B8%D0%BA%D0%B0)) «*научный калькулятор*»): предназначены для научных и инженерных расчётов различной степени сложности. Ориентированы на научных работников, инженеров, студентов технических специальностей и старших школьников.

Работают с представлением чисел в форматах как с естественной, так и с плавающей запятой (во втором случае порядок обычно имеет два, реже — три разряда, мантисса — не менее восьми разрядов, так что максимальный диапазон поддерживаемых ненулевых значений — от 1·10−999 до 9,999999999·10999 по модулю), многие современные конструкции также позволяют непосредственно оперировать обыкновенными [дробями](https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%BE%D0%B1%D1%8C_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), в том числе выполнять с ними операции, преобразовывать обыкновенные дроби из правильных в неправильные и обратно, обыкновенные дроби в десятичные и обратно.

Реализуют алгебраическую логику, с приоритетами операций и скобками; реже применяется [обратная польская запись](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C). Поддерживают вычисление [элементарных функций](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) . Обычный минимум: [квадрат](https://ru.wikipedia.org/wiki/%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82_(%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0)) и квадратный корень, обратная функция, десятичные и натуральные [логарифмы](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC) и антилогарифмы, прямые и обратные [тригонометрические функции](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B8%D0%B3%D0%BE%D0%BD%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8); развитые модели реализуют более широкий набор элементарных функций, могут также поддерживать [статистические](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) расчёты, переводы мер из одной системы в другую, преобразования углов из системы [градус, минута, секунда](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%83%D1%81,_%D0%BC%D0%B8%D0%BD%D1%83%D1%82%D0%B0,_%D1%81%D0%B5%D0%BA%D1%83%D0%BD%D0%B4%D0%B0) в десятичные доли градуса и обратно, [логические функции](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8), работу в различных системах счисления, тригонометрические расчёты с углами в градусах, [радианах](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%B0%D0%BD) и [градах](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4,_%D0%BC%D0%B8%D0%BD%D1%83%D1%82%D0%B0,_%D1%81%D0%B5%D0%BA%D1%83%D0%BD%D0%B4%D0%B0). Общее число поддерживаемых функций может составлять до нескольких сотен.

Число дополнительных [регистров памяти](https://ru.wikipedia.org/w/index.php?title=%D0%A0%D0%B5%D0%B3%D0%B8%D1%81%D1%82%D1%80_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8&action=edit&redlink=1) — не менее одного, но может доходить до десятка и более. Из-за большого количества поддерживаемых функций клавиатура инженерных калькуляторов содержит клавиши двойного/тройного назначения; в некоторых моделях на одну кнопку может быть возложено до четырёх функций. Наиболее развитые модели поддерживают не только числовые, но и [символьные вычисления](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F).

**Бухгалтерские** калькуляторы ориентированы на профессиональные арифметические расчёты с [денежными суммами](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D1%8C%D0%B3%D0%B8), то есть на применение [бухгалтерами](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D1%85%D0%B3%D0%B0%D0%BB%D1%82%D0%B5%D1%80) и [кассирами](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%81%D1%81%D0%B8%D1%80). Обычно выпускаются в настольном исполнении, имеют корпус с крупными клавишами и дисплеем большого размера. Клавиатура может дополнительно содержать клавиши для более удобного ввода денежных сумм (кнопки «00» и «000»), поддерживается большее, чем в инженерных калькуляторах, число знаков (индикатор вмещает до 12—15 цифр), режимы работы с фиксированным количеством разрядов дробной части и автоматическое [округление](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D1%80%D1%83%D0%B3%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5)). Обычно имеют не более одного-двух регистров памяти, но поддерживают арифметические операции с записью в регистр и вычисление процентов. Как правило, не имеют на клавиатуре кнопок двойного/тройного назначения. Некоторые модели реализуют арифметическую логику: операции сложения и вычитания нажимаются после ввода числа, но функции умножения и деления производятся в обычной форме.

**Финансовые** ориентированы на выполнение финансовых расчетов и поддерживают стандартный минимальный набор математических функций, к которому добавляются операции со сложными процентами и специфические функции, применяемые в [банковской сфере](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и иных финансовых приложениях: расчет [аннуитета](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BD%D1%83%D0%B8%D1%82%D0%B5%D1%82), [перпетуитета](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%BF%D0%B5%D1%82%D1%83%D0%B8%D1%82%D0%B5%D1%82" \o "Перпетуитет), [дисконтов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D0%BE%D0%BD%D1%82), размера выплат по кредитам, приведённого денежного потока и тому подобное. Как правило, реализуют алгебраическую логику с приоритетами операций и скобками.

**Программируемые** калькуляторы по функциональным возможностям находятся на уровне сложных инженерных калькуляторов, но дополнительно они дают возможность многократно повторять вычисления, создавая и исполняя программы пользователя. Как правило имеют большое количество регистров памяти (10 и более), могут иметь интерфейсы для подключения внешних устройств, персонального компьютера, дополнительных модулей памяти, аппаратных [датчиков](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D1%87%D0%B8%D0%BA), исполнительных устройств. По функциональности наиболее развитые программируемые калькуляторы приближаются к простейшим портативным [компьютерам](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), формально отличаясь от них исключительно своей узкой специализацией. Существует несколько способов [программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) калькуляторов (см. [статью](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80#%D0%A1%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D1%8B_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), в зависимости от модели калькулятор может поддерживать один или два из них

**Графические** калькуляторы — разновидность программируемых калькуляторов — имеют графический экран и поддерживают команды, которые позволяют отображать [графики функций](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D0%BA_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) или даже выводить на экран произвольные рисунки. Почти все графические калькуляторы являются программируемыми. Графический дисплей может также иметь обычный инженерный калькулятор для поддержки натурального ввода формул и отображения таблиц, но он не называется графическим калькулятором.

**Печатающие** калькуляторы, оснащённые встроенным [печатающим устройством](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BD%D1%82%D0%B5%D1%80), обеспечивающим вывод производимых вычислений, результатов, итогов, графиков на бумажную ленту. Выделяются в отдельный класс в маркетинговых материалах и рыночной аналитике. Современные печатающие калькуляторы по конструкции и вычислительным возможностям обычно относятся к классу бухгалтерских. Ранее со встроенными печатающими устройствами выпускались некоторые инженерные и программируемые калькуляторы, но современные модели этих типов чаще просто имеют интерфейс для подключения внешнего печатающего устройства.

**Специализированные** калькуляторы — калькуляторы или [программно-аппаратные комплексы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE-%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D0%BB%D0%B5%D0%BA%D1%81), предназначенные для выполнения узкоспециализированных вычислений. Например, [штурманский калькулятор](https://ru.wikipedia.org/wiki/%D0%A8%D1%82%D1%83%D1%80%D0%BC%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%B0%D0%BB%D1%8C%D0%BA%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80) для навигационных вычислений, калькулятор для расчета конструкций зданий и сооружений (Construction Master) и т. п.

**Виды по логике операций**

Калькулятор реализует один (редко – два) из трёх вариантов логики операций, то есть порядка ввода команд, который требуется для выполнения арифметических вычислений (команд сложения, вычитания, умножения и деления). Это арифметическая логика, алгебраическая логика и логика вычислений с [обратной польской записью](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C). Первые две базируются на [инфиксной нотации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C) (когда в записи формулы знак [бинарной операции](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F) помещается между операндами), последняя — на постфиксной нотации (когда знак операции помещается после операндов, к которым он относится).

* [**Арифметическая логика**](https://ru.wikipedia.org/w/index.php?title=%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0&action=edit&redlink=1) базируется на инфиксной нотации без приоритетов и скобок. Для выполнения операции «a \* b» (где «\*» — произвольная бинарная операция) пользователь сначала набирает значение a, затем нажимает одну из клавиш бинарной операции («+», «-», «×», «÷», возможно также «yx»), затем набирает значение b и нажимает клавишу «=». Выполняется введённая операция над числами a и b, а её результат отображается на дисплее. Если вместо «=» пользователь снова нажмёт клавишу бинарной операции, то произойдёт то же самое — ранее введённая операция выполнится и её результат отобразится, но этот результат станет первым операндом для той операции, клавиша которой была нажата.

Так, например, для вычисления значения выражения **«30 \* 5 + 45»** пользователь должен последовательно нажать клавиши: **«3»**, **«0»**, **«×»**, **«5»**, **«+»**, **«4»**, **«5»**, **«=»**. При этом после нажатия плюса выполнится умножение 30 на 5, ранее введённое, на дисплее отобразится результат 150, а после знака равенства отобразится окончательный результат 195. Арифметическая логика не предполагает наличия приоритетов операций, все операции выполняются в том порядке, в котором вводятся. Так, попытка вычисления выражения **1 + 2 × 3** путём нажатия кнопок в последовательности **«1»**, **«+»**, **«2»**, **«×»**, **«3»**, **«=»** приведёт к неверному результату, потому что сначала будет выполнено сложение, и только потом умножение, что даст в результате 9, а не 7, как должно получиться согласно правилам математики. Чтобы получить правильный результат, пользователь должен изменить порядок ввода: выполнить сначала операцию умножения, а только затем — сложения.

* **Арифметическая постфиксная** **логика** – разновидность арифметической логики, в которой используется постфиксная нотация для сложения и вычитания. Отличительной особенностью калькуляторов с этой логикой является наличие клавиш с обозначениями «+=» и «-=». Нажатие на эти клавиши приводит к вычислению, соответственно, суммы и разности последних двух введённых чисел. Например, чтобы вычислить 2 — 3, необходимо нажать [2] [+=] [3] [-=]. При этом операции умножения и деления выполняются обычным образом. В настоящее время калькуляторы с такой логикой производятся и используются для бухгалтерских вычислений.
* **Алгебраическая логика** строится на инфиксной записи операций, но, в отличие от арифметической, учитывает в вычислениях принятые в математике приоритеты операций и позволяет пользоваться скобками. Единичная бинарная операция выполняется точно так же, как и в случае арифметической логики, но при выполнении цепочных вычислений при вводе операции, приоритет которой выше, чем приоритет ранее введённой, либо при вводе открывающейся скобки, калькулятор сохраняет во внутренних регистрах ранее введённые операнды и позволяет продолжить ввод. И лишь когда пользователь нажмёт клавишу «=», либо введёт операцию с меньшим приоритетом или закрывающуюся скобку, выполняется вычисление результата введённого выражения либо его части.

Алгебраическая логика позволяет выполнять вычисления по математическим формулам, вводя данные, операции и скобки в том порядке, в котором они записаны в формуле, не задумываясь о правильном порядке выполнения операций. Платой за это удобство является усложнение калькулятора, поскольку для хранения операндов, над которыми ещё не выполнены операции, требуются дополнительные операционные регистры. Каждая вложенная пара скобок и каждая высокоприоритетная операция, расположенная в цепочке вычислений после низкоприоритетной, требует два операционных регистра: для сохранения операнда и отложенной операции. Так, например, при вычислении формулы:



Рис. 1 . Формула .

согласно правилам приоритета ни одна из операций не может быть выполнена до ввода последнего параметра **j**; к моменту, когда пользователь введёт первую закрывающуюся скобку, калькулятор должен сохранить в операционных регистрах 10 операндов и 9 операций.

Так как число регистров ограничено, для калькуляторов с алгебраической логикой существует предел сложности выражения, которое может быть вычислено без преобразования. Простейшие инженерные калькуляторы могут иметь ограничение в 3-5 чисел, ожидающих выполнения (соответственно — столько же пар вложенных скобок и отложенных операций в вычисляемой формуле), более сложные — до десятка и более.

Приоритет и ассоциативность сложения, вычитания, умножения и деления соответствуют принятым в математике, но прочие бинарные операции могут разными калькуляторами выполняться по-разному. Например, цепочечное возведение в степень «**2 ^ 3 ^ 4 =**» в разных моделях может означать 234 или (23)4, а «**− 2 ^ 2 =**» — как (−2)2, так и −(22). Для гарантии правильности расчётов необходимо внимательно изучить документацию конкретной модели калькулятора, а в неоднозначных ситуациях — использовать дополнительные скобки. Некоторые модели калькуляторов автоматически вставляют в поле ввода дополнительные скобки для отображения приоритетов операций.

* **Обратная бесскобочная логика** – тип логики базируется на так называемой обратной [польской нотации](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F) (RPN, Reverse Polish Notation, обратной бесскобочной записи) выражений, в которой сначала записываются подряд значения операндов, а после них — знак выполняемой операции.

Архитектура калькуляторов с обратной бесскобочной логикой характеризуется наличием [стека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA) операционных регистров размером не менее трёх (обычно обозначаемых X, Y, Z) и специфической команды, обозначаемой на клавиатуре как **«↑»** (также **«ENTER»**, **«В↑»**,**«E↑»**). Вводимое с клавиатуры или извлекаемое из регистра памяти значение помещается в регистр Х и отображается на дисплее. По команде **«↑»** происходит сдвиг значений в стеке в направлении X→Y→Z→ (и далее, если в стеке больше регистров), то есть эта операция позволяет разделить ввод последовательных операндов. При нажатии пользователем клавиши любой операции эта операция производится над находящимися в стеке операндами (обычно — над значениями в регистрах Y и X), а результат помещается в регистр X. Остальные значения в стеке при этом сдвигаются обратно в направлении →Z→Y. Ниже в таблице показан порядок вычисления выражения «1 + 2 × 3» на калькуляторе с RPN и содержимое регистров стека после нажатия каждой клавиши (в предположении, что изначально стек был полностью обнулён).

Иногда калькуляторы с RPN имеют ещё один дополнительный операционный регистр, в который после выполнении операции сохраняется предыдущее содержимое регистра X. При необходимости это значение можно извлечь с помощью специальной команды. Скобки в RPN не нужны, так как операции выполняются в порядке ввода.

**RPN** функционально аналогична обычной инфиксной записи со скобками, но при этом вычисления тех же самых выражений требуют меньшего числа нажатий на клавиши. Практика показывает, что научиться применять RPN достаточно просто, но для эффективного использования калькулятора с обратной бесскобочной логикой необходима предварительная тренировка и постоянное поддержание навыков. Среди обычных инженерных калькуляторов использование RPN является редкостью; из зарубежных можно назвать несколько моделей фирмы HP, из советских — единственную модель «Электроника Б3-19М» (в настоящее время не выпускается). Более популярна RPN в программируемых калькуляторах, в том числе благодаря достигаемому с её использованием сокращению размеров программ: для калькулятора с небольшим объёмом программной памяти экономия буквально одной-двух команд иногда определяет, поместится ли крайне необходимая программа в памяти, или её придётся сокращать, жертвуя возможностями и удобством использования.

## Описание существующих разработок

На данный момент существует множество уже разработанных приложений по этой тематике. Практически все из них предоставляют похожий функционал:

* Возможность ввода математического выражения, отображающегося в отведённом текстовом блоке;
* Возможность ввода стандартных математических операций, цифр, скобок, а также знака равенства;
* Получение результата после ввода итогового выражения, зависящего от реализованной логики операций в используемом на данный момент калькуляторе. Результат записывается также в отведённом текстовом блоке, предварительно очищенном от отображённого в нём математического выражения .

Примеры:

[](https://commons.wikimedia.org/wiki/File:Casio_HL-820ER_(1).jpg?uselang=ru)

Рис. 2 . Простой калькулятор Casio HL-820ER



Рис. 3 . Бухгалтерский калькулятор Casio DS-20

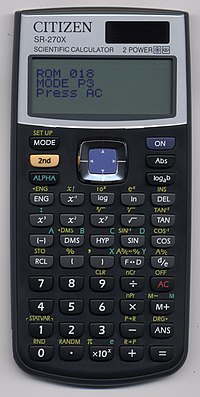


Рис. 4 . Современный инженерный

непрограммируемый калькулятор.



Рис. 5 . Программный, встроенный в Windows 7, калькулятор .

# Проектная часть

## Диаграмма прецедентов

В данном разделе содержится диаграмма прецедентов для приложения .

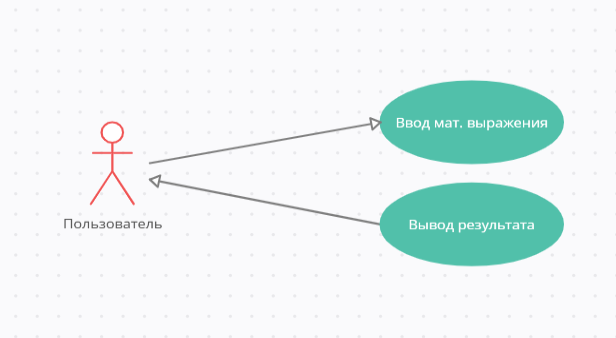


Рис. 6 . Диаграмма прецедентов программы.

## Выбор инструментов

Выбрана была среда разработки Visual Studio, язык программирования C#, платформа WPF (XAML) .

Выбор состоял из выбранного C# (WPF), устаревшего C++ (SFML), а также Java . Требовались современные технологии, что подразумевает выбор лучшего / одного из лучших языков программирования в данном направлении . C# является таковым . WPF самый популярный способ написания десктопных приложений под Windows . Платформа работает на языке XAML, что, в свою очередь, отличает от прочих вариантов, в которых подключаются сторонние библиотеки, подчёркивает углублённость WPF в данном направлении .

### Язык программирования

Для написания данного приложения использовались: высокоуровневый язык программирования C#, а также технология WPF, входящая в .NET Framework, для создания визуального интерфейса десктопного приложения .

**Язык программирования**

**C#** — [*объектно-ориентированный*](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)[*язык программирования*](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании **[Microsoft](https://ru.wikipedia.org/wiki/Microsoft" \o "Microsoft)** под руководством [Андерса **Хейлсберга**](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) и Скотта **Вильтаумота**как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework). Впоследствии был стандартизирован как [ECMA](https://ru.wikipedia.org/wiki/ECMA)-334 и [ISO](https://ru.wikipedia.org/wiki/ISO)/[IEC](https://ru.wikipedia.org/wiki/IEC) 23270.

**Особенности языка программирования:**

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java" \o "Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы,  [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80),  [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F)  с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Delphi (язык программирования)), [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) и, в особенности, [Java](https://ru.wikipedia.org/wiki/Java" \o "Java) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%8)).

C# разрабатывался как язык программирования прикладного уровня для [CLR](https://ru.wikipedia.org/wiki/Common_Language_Runtime) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C# . Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на [VB.NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET), [J#](https://ru.wikipedia.org/wiki/Visual_J%E2%99%AF) и др.

**Платформа разработки**

Windows Presentation Foundation (**WPF**) — аналог **WinForms**, система для построения клиентских приложений [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows) с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [**.NET Framework**](https://ru.wikipedia.org/wiki/.NET_Framework) (начиная с версии [3.0](https://ru.wikipedia.org/wiki/.NET_Framework_3.0)), использующая язык [**XAML**](https://ru.wikipedia.org/wiki/XAML).

**WPF** предустановлена в [*Windows Vista*](https://ru.wikipedia.org/wiki/Windows_Vista) ([.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) 3.0), [*Windows 7*](https://ru.wikipedia.org/wiki/Windows_7) ([.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) 3.5 SP1), [*Windows 8*](https://ru.wikipedia.org/wiki/Windows_8) (.NET Framework 4.0 и 4.5), [*Windows 8.1*](https://ru.wikipedia.org/wiki/Windows_8.1) (.NET Framework 4.5.1) и [*Windows 10*](https://ru.wikipedia.org/wiki/Windows_10) (.NET Framework 4.7). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в [браузере](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) приложений.

**Особенности платформы:**

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык [XAML](https://ru.wikipedia.org/wiki/XAML) (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является [DirectX](https://ru.wikipedia.org/wiki/DirectX" \o "DirectX), в отличие от [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms), где используется [GDI](https://ru.wikipedia.org/wiki/GDI)/[GDI+](https://ru.wikipedia.org/wiki/GDI%2B). Производительность [WPF](https://ru.wikipedia.org/wiki/WPF) выше, чем у [GDI+](https://ru.wikipedia.org/wiki/GDI%2B) за счёт использования аппаратного ускорения графики через [DirectX](https://ru.wikipedia.org/wiki/DirectX" \o ").Также существует урезанная версия [CLR](https://ru.wikipedia.org/wiki/CLR), называющаяся [WPF/E](https://ru.wikipedia.org/wiki/WPF/E), она же известна как [Silverlight](https://ru.wikipedia.org/wiki/Silverlight" \o "Silverlight).

[**XAML**](https://ru.wikipedia.org/wiki/XAML) представляет собой язык декларативного описания интерфейса, основанный на [**XML**](https://ru.wikipedia.org/wiki/XML). Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на [*элементы управления*](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, [Button](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0_(%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC)" \o "Кнопка (элемент интерфейса программ)) (кнопка).

Для работы с WPF требуется любой [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-совместимый язык. В этот список входит множество языков:

[**C#**](https://ru.wikipedia.org/wiki/C_Sharp), [**F#**](https://ru.wikipedia.org/wiki/F_Sharp), [**VB.NET**](https://ru.wikipedia.org/wiki/VB.NET), [**C++**](https://ru.wikipedia.org/wiki/Visual_C%2B%2B), [**Ruby**](https://ru.wikipedia.org/wiki/IronRuby), [**Python**](https://ru.wikipedia.org/wiki/IronPython), [**Delphi** **(Prism)**](https://ru.wikipedia.org/wiki/Oxygene_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), **[Lua](https://ru.wikipedia.org/w/index.php?title=Lua.NET&action=edit&redlink=1" \o "Lua.NET (страница отсутствует))** и многие другие. Для полноценной работы может быть использована как **[Visual Studio](https://ru.wikipedia.org/wiki/Visual_Studio" \o "Visual Studio)**, так и **[Expression Blend](https://ru.wikipedia.org/wiki/Expression_Blend" \o "Expression Blend)**. Первая ориентирована на программирование, а вторая — на дизайн и позволяет делать многие вещи, не прибегая к ручному редактированию [**XAML**](https://ru.wikipedia.org/wiki/XAML). Примеры этому — анимация, стилизация, состояния, создание элементов управления и так далее.

### Среда разработки программного обеспечения

**Visual Studio** это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

**Visual Studio** включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии **[IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense" \o "IntelliSense)** и возможностью простейшего [**рефакторинга** кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3).

Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как, например, [Subversion](https://ru.wikipedia.org/wiki/Subversion" \o "Subversion) и [Visual SourceSafe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe" \o "Microsoft Visual SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server" \o "Team Foundation Server)).

**Visual Studio** включает один или несколько компонентов из следующих:

* [**Visual Basic .NET**](https://ru.wikipedia.org/wiki/Visual_Basic_.NET), а до его появления — **[Visual Basic](https://ru.wikipedia.org/wiki/Visual_Basic" \o "Visual Basic)**
* [**Visual C++**](https://ru.wikipedia.org/wiki/Microsoft_Visual_C%2B%2B)
* [**Visual C#**](https://ru.wikipedia.org/wiki/C_Sharp) (включён начиная с Visual Studio .NET)
* [**Visual F#**](https://ru.wikipedia.org/wiki/F_Sharp) (включён начиная с Visual Studio 2010)

Многие варианты поставки также включают:

* [**Microsoft SQL Server**](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) либо [**Microsoft SQL Server Express**](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server#SQL_Server_Express_Edition)

В прошлом в состав **Visual Studio** также входили продукты:

* [**Visual InterDev**](https://ru.wikipedia.org/wiki/Visual_InterDev)
* [**Visual J++**](https://ru.wikipedia.org/wiki/Visual_J%2B%2B)
* [**Visual J#**](https://ru.wikipedia.org/wiki/Visual_J_Sharp)
* [**Visual FoxPro**](https://ru.wikipedia.org/wiki/Visual_FoxPro)
* [**Visual Source Safe**](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe) — файл-серверная [система управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) .

## Проектирование сценария

В данном разделе приведен сценарий использования пользователем:



Рисунок 7 . Сценарий использования сайта.

При запуске программы откроется главный экран, на которой представлены все элементы ввода мат. выражения и вывода результата .

Далее пользователь может ввести нужное ему для вычисления математическое выражение, используя визуальные кнопки, или же клавиши на клавиатуре компьютера . При нажатии на клавишу ***“Enter”*** или же клика мышью на визуальную кнопку ***“=”*** в текстовом поле, ранее используемом для ввода выражения, заранее очищенном программой, появляется результат вычислений .

## Диаграммы классов

В данном разделе располагается диаграмма классов программы

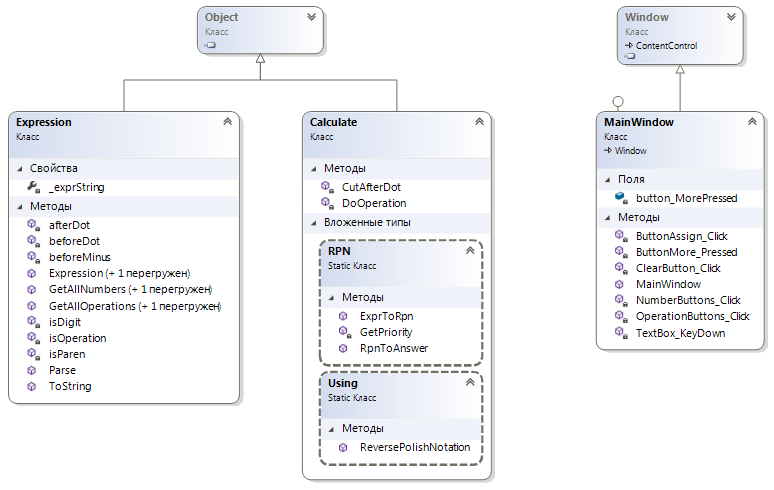


Рис. 8. Диаграмма классов программы .

## Описание главного модуля.

Главный модуль состоит из класса **MainWindow**. Он выполняется при запуске приложения. В данном модуле создаётся окно приложения .

Листинг 1. Главный модуль (MainWindow.xaml.cs)

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Calculator

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private bool button\_MorePressed = false;

#region More Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "more" .

\*/

#endregion

private void ButtonMore\_Pressed(object sender, RoutedEventArgs e)

{

double colCount = (double)grid1.ColumnDefinitions.Count;

if (!button\_MorePressed)

{

ButtonMore.Content = "back";

var newColumn = new ColumnDefinition();

grid1.ColumnDefinitions.Add(newColumn);

colCount = (double)grid1.ColumnDefinitions.Count;

Width \*= colCount / (colCount - (double)1);

button\_MorePressed = true;

}

else

{

ButtonMore.Content = "more";

grid1.ColumnDefinitions.RemoveAt((int)(colCount - 1));

colCount = (double)grid1.ColumnDefinitions.Count;

Width \*= colCount / (colCount + (double)1);

button\_MorePressed = false;

}

}

#region '=' Click:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "=" .

\*/

#endregion

private void ButtonAssign\_Click(object sender, RoutedEventArgs e)

{

string expression = textBox.Text;

double result = 0;

try

{

result = Math.Calculate.Using

.ReversePolishNotation(expression);

textBox.Text = result.ToString();

}

catch (InvalidOperationException)

{

MessageBox.Show("Input expression is not currect!");

}

}

#region Numbers Buttons:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии цифровых

\* визуальных кнопок .

\*/

#endregion

private void NumberButtons\_Click(object sender, RoutedEventArgs e)

{

textBox.Text +=

((Button)e.OriginalSource).Content.ToString();

((Button)e.OriginalSource).Focusable = false;

}

#region Clear Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "AC" .

\*/

#endregion

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

textBox.Text = "";

}

#region Operations Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальных кнопок,

\* отвечающих за хранение операций математических выражений .

\*/

#endregion

private void OperationButtons\_Click(object sender, RoutedEventArgs e)

{

textBox.Text += $" {((Button)e.OriginalSource).Content} ";

}

private void TextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

ButtonAssign\_Click(sender, e);

}

}

}

}

## Описание спецификаций к модулям

Всего разработано 2 модуля:

* **MainWindow** – главный модуль, отвечающий за логику UI-элементов;
* **Math** – модуль, отвечающий за логику вычислений (**RPN**) .

Взаимодействие между модулями можно определить по диаграммам классов из раздела 2.4.

## Описание модулей

**MainWindow** – главное окно, включающее в себя следующие UI-элементы:

* **TextBox** – текстовый элемент, отвечающий за ввод выражения, вывод результата;
* **Buttons** – кнопки, отвечающие за изменение содержимого **TextBox**’а (цифры, операции и т. д.) .

**Math** – подключаемая библиотека, включает в себя всю логику вычислений калькулятора, основанную на Обратной Польской Нотации (*Reverse Polish Notation*) . Состоит из:

* **class** Calculate:

(отвечает за вычисление математических выражений)

* + **class** Using:

(отвечает за выбор способа вычисления математических выражений)

* + - ReversePolishNotation(…)

(отвечает за выбор Обратной Польской Нотации, вызывает все необходимые функции)

* + **class** RPN:

(отвечает за вычисление выражения с помощью Обратной Польской Записи)

* + - GetPriority(…)

(возвращает приоритет принимаемой операции)

* + - ExprToRpn(…)

(парсит принимаемое выражение в RPN)

* + - RpnToAnswer(…)

(вычисляет принимаемую RPN)

* + DoOperation(…)

(вычисляет принимаемое действие выражения)

* + CutAfterDot(…)

(сокращает число после точки / запятой)

* **class** Expression:

(тип данных пользовательский (Математическое Выражение))

* + Expression(…)

(конструктор класса (перегрузка))

* + ToString(…)

(переопределённая функция, возвращающая выражение в строке)

* + GetAllNumbers(…)

(возвращает все числа заданного выражения (перегрузка))

* + GetAllOperations(…)

(возвращает все операции заданного выражения (перегрузка))

* + Parse(…)

(парсит строку в тип данных данного класса)

* + beforeMinus(…)

(выполняет определённые действия с числом перед знаком минус)

* + beforeDot(…)

(выполняет определённые действия с числом перед точкой / запятой)

* + afterDot(…)

(выполняет определённые действия с числом после точки / запятой)

* + isOperation(…)

(определяет, является ли принимаемый символ операцией)

* + isDigit(…)

(определяет, является ли принимаемый символ цифрой)

* + isParen(…)

(определяет, является ли принимаемый символ скобкой) .

Рассмотрим функцию RpnToAnswer, и построим блок схему:

Листинг 2. Функция *RpnToAnswer*, отвечающая за вычисление принимаемой строки, заранее изменённой методами Parse (из строки в Expression) и ExprToRpn (из Expression в Reverse Polish Notation) (Math.dll):

public static double RpnToAnswer(string rpn)

{

string currentNumber = "";

var result = new Stack<double>();

for (int i = 0; i < rpn.Length; i++)

{

if (GetPriority(rpn[i]) == 0)

{

while (GetPriority(rpn[i]) == 0 && i < rpn.Length)

{

currentNumber += rpn[i];

if (i + 1 < rpn.Length)

{

if (GetPriority(rpn[i + 1]) == 0)

{

i++;

}

else break;

}

else break;

}

var nfi = new NumberFormatInfo()

{

NumberDecimalSeparator = "."

};

result.Push(double.Parse(currentNumber, nfi));

currentNumber = "";

}

else if (GetPriority(rpn[i]) > 0)

{

double rValue = result.Pop();

double lValue = result.Pop();

result.Push(double.Parse(CutAfterDot(

DoOperation(lValue, rpn[i], rValue))));

}

}

return result.Pop();

}

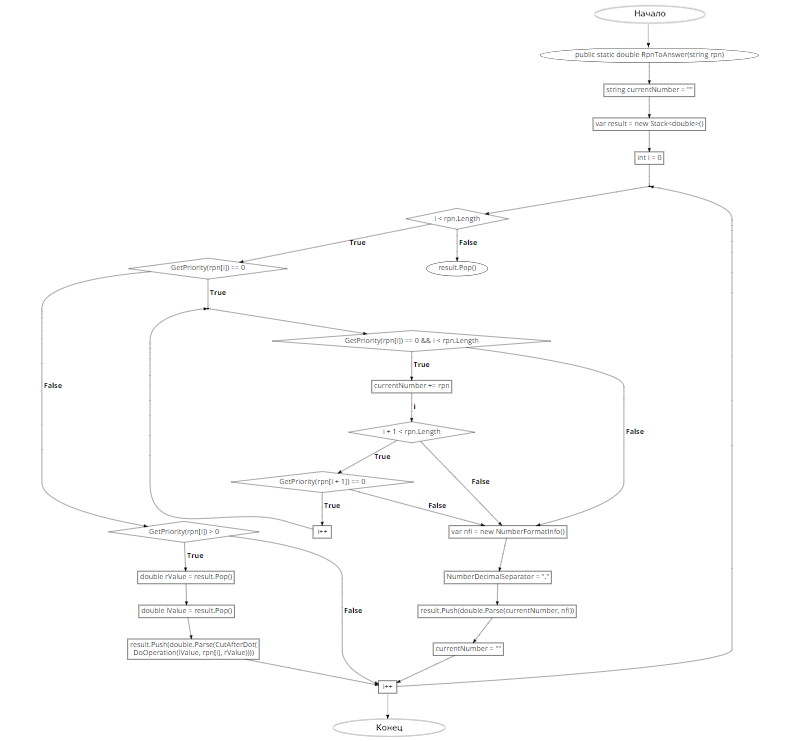


Рис. 9. Блок-схема функции RpnToAnswer, класса RPN .

## Описание тестовых наборов модулей

В некоторых модулях производится проверка данных на корректность, чтобы сообщить пользователю о проблеме и предотвратить аварийное завершение программы

**Тест 1. Проверка на правильность ввода выражения .**

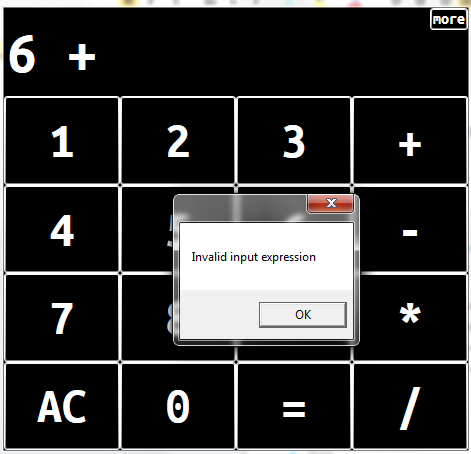


Рис. 10. Invalid input expression.

# Эксплуатационная часть

## Руководство оператора

### Назначение программы

Разработанная программа позволяет удобно, а самое главное - правильно считать длинные математические выражения, с которыми у обычных калькуляторов, как правило, возникают трудности из-за выбора при разработке альтернативной логики операций .

### Условия выполнения программы

Любой компьютер, выпущенный не раньше 2009 г, с OC Windows 7, 8, 8.1, 10 .

### Выполнение программы

При запуске приложение сразу отображает главное, едиственное окно Калькулятора:



Рисунок 11. Запуск Калькулятора .

Вы можете ввести любое выражение, которое позволяют ввести UI-элементы окна, а затем кликнуть мышью на визуальную кнопку “*=*”, или нажать на клавиатуре на клавишу “*Enter*”:

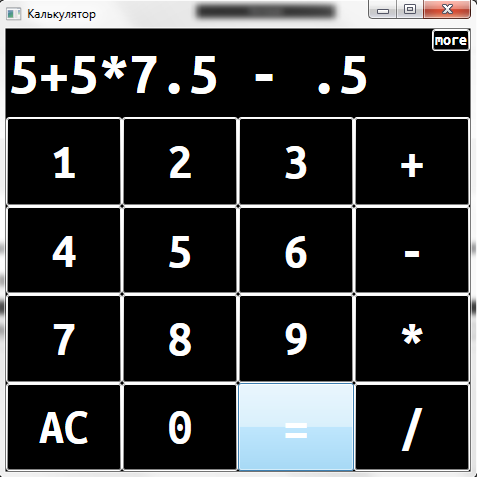


Рисунок 12. Ввод математического выражения .

Далее получаем результат:



Рисунок 13. Результат вычислений .

## To-Do лист

* Добавить визуальные кнопки скобок, в данный момент их можно использовать только с клавиатуры .
* Добавить в расширенную версию калькулятора (нажатие визуальной кнопки “more”)



Рисунок 14. Кнопка “more” .

дополнительные операции / функции вместо кнопок “?”:

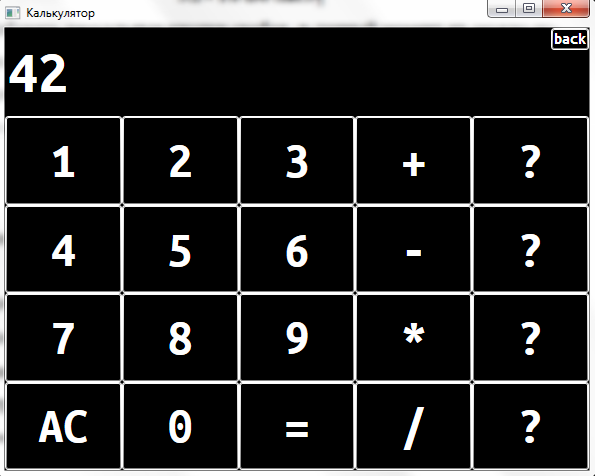


Рисунок 15. Расширенная версия приложения (доработать) .

* Добавить функцию отображения графиков и решения уравнений .
* После графиков добавить форму авторизации / регистрации .
* Заняться рефакторингом кода программы, после окончательного завершения разработки функционала .

# Заключение

Данный проект позволяет высчитывать длинные математические выражения, получая верный результат . Также приложение обладает личным парсером введённой строки, который обрабатывает лишние введённые символы (опечатки) пользователей .

# Список литературы и интернет ресурсов

1. Wikipedia (Калькулятор, C#, WPF, Visual Studio)

* https://ru.wikipedia.org/wiki/Калькулятор
* <https://ru.wikipedia.org/wiki/C_Sharp>
* <https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation>
* https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio

1. Microsoft (C#, WPF, Visual Studio)

* <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>
* <https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netframeworkdesktop-4.8>
* https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019

1. Metanit (C#, WPF)

* <https://metanit.com/sharp/tutorial/1.1.php>
* <https://metanit.com/sharp/wpf/1.php>

**Приложение 1. MainWindow.xaml.cs**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Calculator

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private bool button\_MorePressed = false;

#region More Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "more" .

\*/

#endregion

private void ButtonMore\_Pressed(object sender, RoutedEventArgs e)

{

double colCount = (double)grid1.ColumnDefinitions.Count;

if (!button\_MorePressed)

{

ButtonMore.Content = "back";

var newColumn = new ColumnDefinition();

grid1.ColumnDefinitions.Add(newColumn);

colCount = (double)grid1.ColumnDefinitions.Count;

Width \*= colCount / (colCount - (double)1);

button\_MorePressed = true;

}

else

{

ButtonMore.Content = "more";

grid1.ColumnDefinitions.RemoveAt((int)(colCount - 1));

colCount = (double)grid1.ColumnDefinitions.Count;

Width \*= colCount / (colCount + (double)1);

button\_MorePressed = false;

}

}

#region '=' Click:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "=" .

\*/

#endregion

private void ButtonAssign\_Click(object sender, RoutedEventArgs e)

{

string expression = textBox.Text;

double result = 0;

try

{

result = Math.Calculate.Using.ReversePolishNotation(expression);

textBox.Text = result.ToString();

}

catch (InvalidOperationException)

{

MessageBox.Show("Invalid input expression");

}

}

#region Numbers Buttons:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии цифровых

\* визуальных кнопок .

\*/

#endregion

private void NumberButtons\_Click(object sender, RoutedEventArgs e)

{

textBox.Text += ((Button)e.OriginalSource).Content.ToString();

((Button)e.OriginalSource).Focusable = false;

}

#region Clear Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальной кнопки "AC" .

\*/

#endregion

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

textBox.Text = "";

}

#region Operations Button:

/\*

\* Функция, отвечающая за действия,

\* выполняемые при нажатии визуальных кнопок,

\* отвечающих за хранение операций математических выражений .

\*/

#endregion

private void OperationButtons\_Click(object sender, RoutedEventArgs e)

{

textBox.Text += $" {((Button)e.OriginalSource).Content} ";

}

private void TextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

ButtonAssign\_Click(sender, e);

}

}

}

}

**Приложение 2. Math.cs(в проекте используется преобразованная в**

**подключаемую библиотеку (Math.dll) версия)**

using System.Collections.Generic;

using System.Globalization;

namespace Math

{

// Отвечает за вычисление математических выражений:

public static class Calculate

{

// Отвечает за выбор способа вычисления математических выражений:

public static class Using

{

// Отвечает за выбор Обратной Польской Нотации:

public static double ReversePolishNotation

(string expression)

{

expression = Expression.Parse(expression);

expression = RPN.ExprToRpn(expression);

return RPN.RpnToAnswer(expression);

}

}

public static class RPN

{

// Возвращает приоритет принимаемой операции:

private static int GetPriority(char operation)

{

if (operation == '\*' || operation == '/') return 3;

else if (operation == '-' || operation == '+')

return 2;

else if (operation == '(') return 1;

else if (operation == ')') return -1;

else if (int.TryParse(operation.ToString(),

out int result) || operation == '.')

{

return 0;

}

else

{

return -10;

}

}

// Парсит принимаемое выражение в Обратную Польскую Нотацию:

public static string ExprToRpn(string expression)

{

if (expression.Length != 0)

{

string \_current = "";

Stack<char> \_stack = new Stack<char>();

int \_priority;

for (int i = 0; i < expression.Length; i++)

{

\_priority = GetPriority(expression[i]);

if (\_priority == 0)

\_current += expresion[i];

if (\_priority == 1)

\_stack.Push(expression[i]);

if (\_priority > 1)

{

\_current += ' ';

while (\_stack.Count != 0)

{

if (GetPriority(\_stack.Peek()) >= \_priority)

{

\_current += \_stack.Pop();

}

else break;

}

\_stack.Push(expression[i]);

}

if (\_priority == -1)

{

\_current += ' ';

while (GetPriority(\_stack.Peek()) != 1)

{

\_current += \_stack.Pop();

}

\_stack.Pop();

}

}

while (\_stack.Count != 0)

{

\_current += \_stack.Pop();

}

return \_current;

}

return "";

}

// Вычисляет принимаемую Обратную Польскую Нотацию:

public static double RpnToAnswer(string rpn)

{

string currentNumber = "";

var result = new Stack<double>();

for (int i = 0; i < rpn.Length; i++)

{

if (GetPriority(rpn[i]) == 0)

{

while (GetPriority(rpn[i]) == 0 && i <

rpn.Length)

{

currentNumber += rpn[i];

if (i + 1 < rpn.Length)

{

if (GetPriority

(rpn[i + 1]) ==0)

{

i++;

}

else break;

}

else break;

}

var nfi = new NumberFormatInfo()

{

NumberDecimalSeparator = "."

};

result.Push(double.Parse(currentNumber, nfi));

currentNumber = "";

}

else if (GetPriority(rpn[i]) > 0)

{

double rValue = result.Pop();

double lValue = result.Pop();

result.Push(double.Parse(CutAfterDot

(DoOperation(lValue, rpn[i], rValue))));

}

}

return result.Pop();

}

}

// Вычисляет единственное принимаемое выражение:

private static double DoOperation(

double lValue, char operation, double rValue)

{

switch (operation)

{

case '+':

return lValue + rValue;

case '-':

return lValue - rValue;

case '\*':

return lValue \* rValue;

case '/':

return lValue / rValue;

default:

return 0;

}

}

// Сокращает число после запятой .

private static string CutAfterDot(double number)

{

string numStr = number.ToString();

if (numStr.Contains(","))

{

if (numStr.Substring(

numStr.IndexOf(',') + 1).Length > 4)

{

numStr = numStr.Substring(0, numStr.IndexOf(',') + 6);

if (int.Parse(

numStr[numStr.Length - 1].ToString()) > 5 &&

int.Parse(

numStr[numStr.Length - 2].ToString()) != 9)

{

int lastDigit = int.Parse(

numStr[numStr.Length - 2].ToString()) + 1;

numStr = numStr.Remove(numStr.Length - 2);

numStr += lastDigit.ToString();

}

else

{

return numStr.Remove(numStr.Length - 1);

}

}

}

return numStr;

}

}

class Expression

{

#region Expression into a String:

/\*

\* Cвойство класса Expression,

\* отвечающее за хранение выражения в

\* строковом формате .

\*/

#endregion

private string \_exprString { get; set; } = "";

public Expression() { }

public Expression(string exprString)

{

\_exprString = Parse(exprString);

}

public override string ToString()

{

return \_exprString;

}

#region Get all Numbers(Queue) from Expression:

/\*

\* Возвращает все числа заданного выражения

\* В формате QUEUE .

\*/

#endregion

public Queue<double> GetAllNumbers()

{

var allNumbers = new Queue<double>();

string currentNumber = "";

for (int i = 0; i < \_exprString.Length; i++)

{

if (\_exprString[i] == '.' || isDigit(\_exprString[i]))

{

for (; i < \_exprString.Length &&

(\_exprString[i] == '.' ||

isDigit(\_exprString[i])); i++)

{

currentNumber += \_exprString[i];

}

var nfi = new NumberFormatInfo()

{

NumberDecimalSeparator = "."

};

double number = double.Parse(

currentNumber, nfi);

allNumbers.Enqueue(number);

currentNumber = "";

}

}

return allNumbers;

}

#region Get all Numbers(Stack) from Expression:

/\*

\* Возвращает все числа заданного выражения

\* В формате STACK .

\*/

#endregion

public static Stack<double> GetAllNumbers(string expr)

{

var allNumbers = new Stack<double>();

string currentNumber = "";

for (int i = 0; i < expr.Length; i++)

{

if (expr[i] == '.' || isDigit(expr[i]))

{

for (; i < expr.Length &&

(expr[i] == '.' || isDigit(expr[i])); i++)

{

currentNumber += expr[i];

}

var nfi = new NumberFormatInfo()

{

NumberDecimalSeparator = "."

};

double number = double.Parse(

currentNumber, nfi);

allNumbers.Push(number);

currentNumber = "";

}

}

return allNumbers;

}

#region Get all Operations(Queue) from Expression:

/\*

\* Возвращает все операции заданного выражения

\* В формате QUEUE .

\*/

#endregion

public Queue<char> GetAllOperations()

{

var allOpers = new Queue<char>();

foreach (var ch4r in \_exprString)

{

if (isOperation(ch4r))

{

allOpers.Enqueue(ch4r);

}

}

return allOpers;

}

#region Get all Operations(Queue) from Expression:

/\*

\* Возвращает все операции заданного выражения

\* В формате STACK .

\*/

#endregion

public static Stack<char> GetAllOperations(string expr)

{

var allOpers = new Stack<char>();

foreach (var ch4r in expr)

{

if (isOperation(ch4r))

{

allOpers.Push(ch4r);

}

}

return allOpers;

}

/\*||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||\*/

// Парсит строку в тип данных Expression:

public static string Parse(string str)

{

string acceptedString = "";

// if string isn't empty:

if (str.Length != 0)

{

foreach (var ch4r in str)

{

if (ch4r != ' ')

{

acceptedString += ch4r;

}

}

str = acceptedString;

acceptedString = "";

// take each char from string:

for (byte i = 0; i < str.Length; i++)

{

// if char is a digit(1, 2, 3, ...):

if (isDigit(str[i]))

{

acceptedString += str[i];

}

// if char is an operation (+, -, \*, /, ...):

else if (isOperation(str[i]))

{

if (str[i] == '-')

{

beforeMinus(str, i,

ref acceptedString);

}

else

{

acceptedString += $" {str[i]} ";

}

}

// if char is a dot or a virgule:

else if (str[i] == '.' || str[i] == ',')

{

beforeDot(str, i, ref acceptedString);

afterDot(str, i, ref acceptedString);

}

// if char is a paren ('(', ')'):

else if (isParen(str[i]))

{

acceptedString += str[i];

}

// if char is a white space:

else if (str[i] == ' ')

{

if (str[i + 1] == ' ')

{

while (str[i] == ' ')

{

i++;

}

}

else

{

acceptedString += str[i];

}

}

else

{

continue;

}

}

// remove all white spaces from start and end of acceptedString:

acceptedString = acceptedString.Trim();

// if at start of string is not a digit:

if (acceptedString[0] == '.' || acceptedString[0] == ',' || isOperation(acceptedString[0]))

{

acceptedString = isOperation(acceptedString[0]) ? $"0 {acceptedString}" : $"0{acceptedString}";

}

}

return acceptedString;

}

// Выполняет определённые действия с числом перед знаком минус:

private static void beforeMinus(string str, int minusIndx,

ref string acceptedString)

{

if (acceptedString.Length == 0)

{

acceptedString += $"0 { str[minusIndx] } ";

return;

}

for (int i = acceptedString.Length - 1; i >= 0; i--)

{

if (isParen(acceptedString[i]))

{

acceptedString += $"0 { str[minusIndx] } ";

break;

}

else if (isDigit(acceptedString[i]))

{

acceptedString += $" { str[minusIndx] } ";

break;

}

else if (isOperation(acceptedString[i]))

{

if (acceptedString[i - 1] == ' ')

{

acceptedString.Remove(i - 1, 2);

}

else

{

acceptedString.Remove(i, 1);

}

}

else

{

continue;

}

}

}

// Выполняет определённые действия с числом перед точкой или запятой:

private static void beforeDot(string str, int dotIndx,

ref string acceptedString)

{

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

{

if (isDigit(str[i]))

{

break;

}

else if (isOperation(str[i]))

{

acceptedString += '0';

break;

}

}

}

// Выполняет определённые действия с числом после точки или запятой:

private static void afterDot(string str, int dotIndx,

ref string acceptedString)

{

for (int i = dotIndx + 1; i < str.Length; i++)

{

if (isDigit(str[i]))

{

acceptedString += '.';

break;

}

else if (isOperation(str[i]))

{

break;

}

}

}

// Определяет, является ли принимаемый символ операцией:

private static bool isOperation(char ch)

{

char[] regularItems = { '+', '-', '\*', '/' };

foreach (var item in regularItems)

{

if (ch == item) return true;

}

return false;

}

// Определяет, является ли принимаемый символ цифрой:

private static bool isDigit(char ch)

{

return (ch >= 48 && ch <= 57) ? true : false;

}

// Определяет, является ли принимаемый символ скобкой:

private static bool isParen(char ch)

{

if (ch == '(' || ch == ')')

{

return true;

}

return false;

}

}

}