Отчет по лабораторной работе №4

Выполнил:

студент группы ИВТ-19-1

Полянский Илья Игоревич

**Цель работы:**

Запрограммировать калькулятор, аналог встроенного калькулятора Windows

**Задание.**

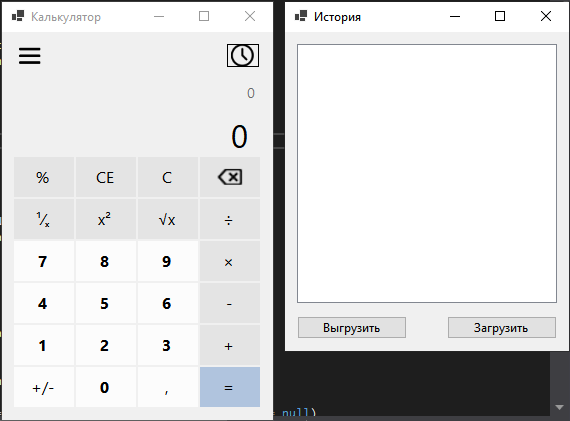
1. Калькулятор пишется на С# (Windows form application).
2. Калькулятор должен вычислять результаты суммирования, вычитания, деления и умножения, использования встроенных функций (типа возведения в квадрат, вычисление квадратного корня, обратной величины, логарифма, тригонометрических функций).

Дополнить калькулятор окном журнала операций. В журнале операций должны отображаться операции в сеансе, с возможностью сохранения в текстовом файле и считывания с него.

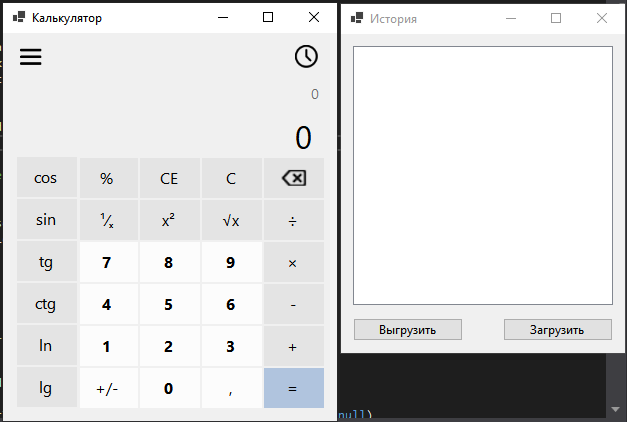
Написание программы осуществлялось в среде программирования MS Visual Studio 2019 при помощи API Windows Forms.

Работа программы

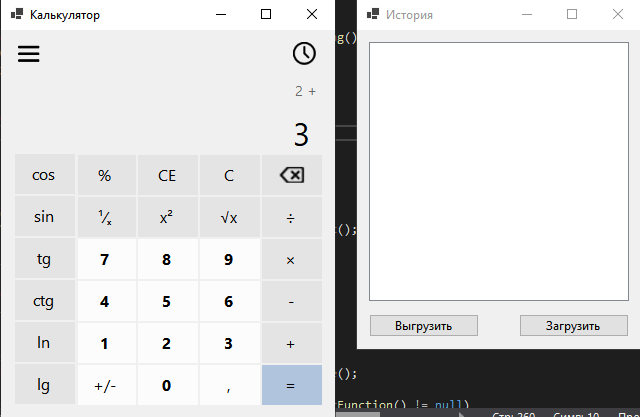
Начальное состояние:



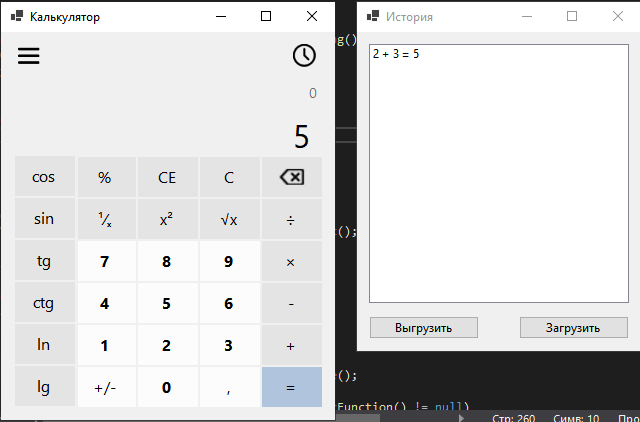
Выдвижное «инженерное меню»



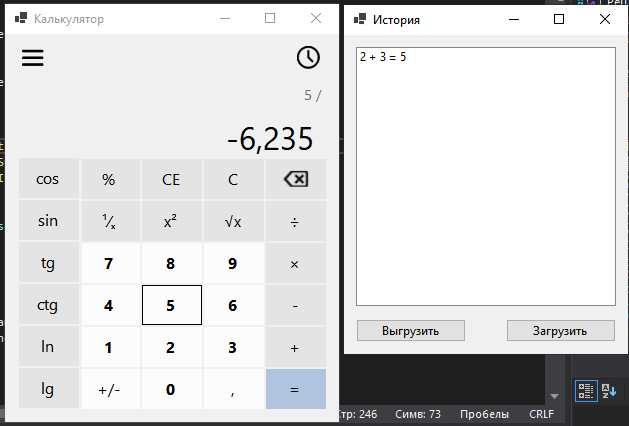
Вводим выражение



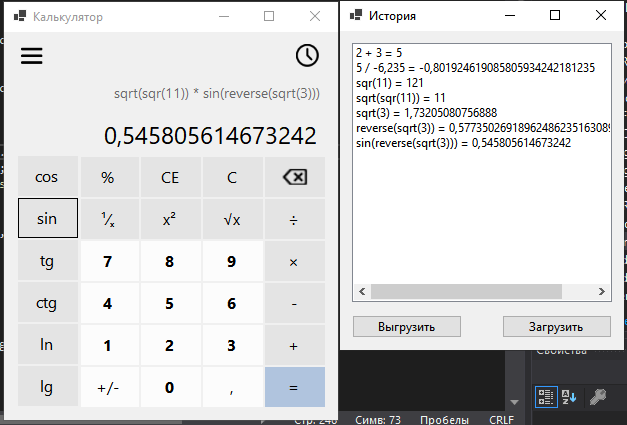
Нажимаем кнопку «=», получаем результат



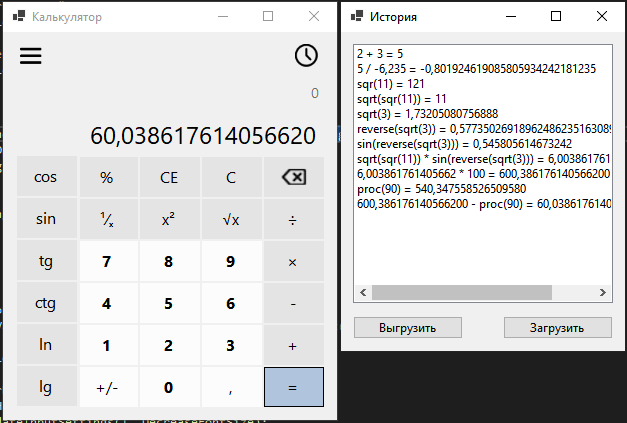
Получив результат, можно сразу нажать на один из операторов и продолжить вычисления, где левым операндом будет результат предыдущего выражения. Тут же продемонстрируем результат работы кнопок «+/-» и «,»



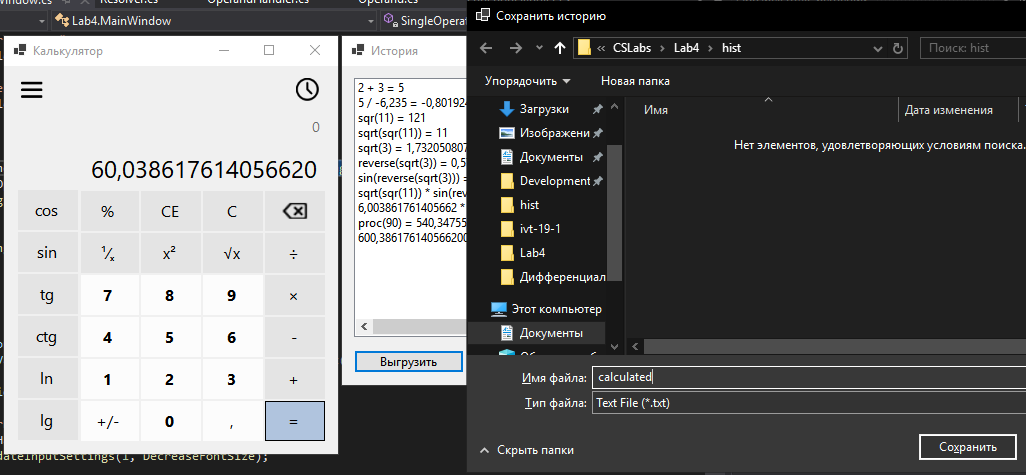
Введем более сложное выражение. Сначала над числом 11 выполняем операцию возведения в квадрат, получается 121, результат заносится в историю. Затем возьмем квадратный корень от получившегося выражения – это 11, результат заносится в историю. Затем вводим знак умножения и вводим значение второго операнда – 3, берем от него квадратный корень, выполняем операцию reverse, то есть 1/x, и от этого берем синус



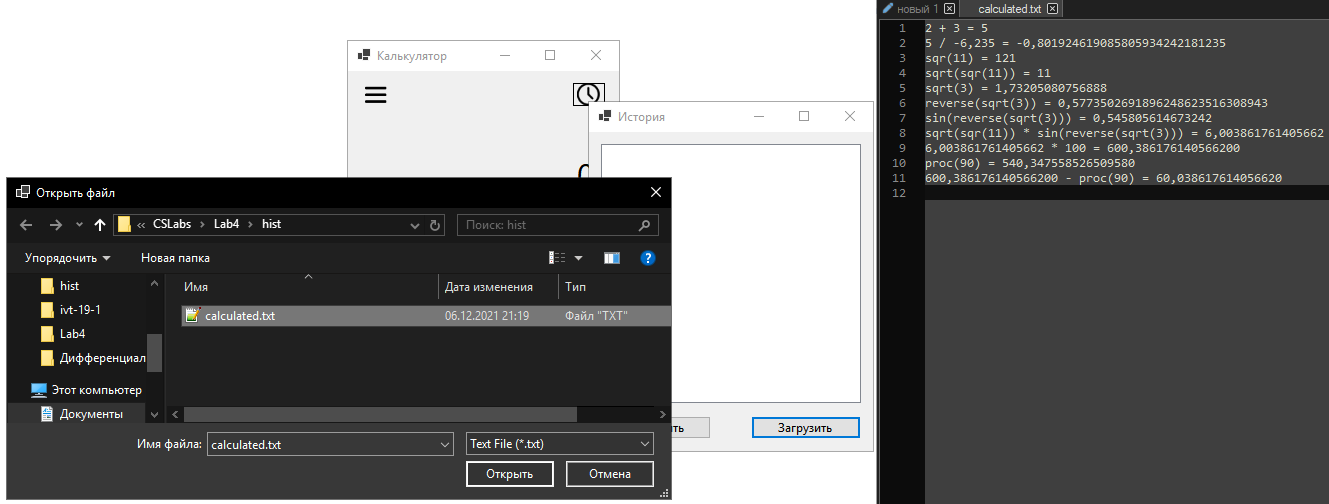
Посчитаем результат. Помножим его на 100, а затем отнимем из него 90%



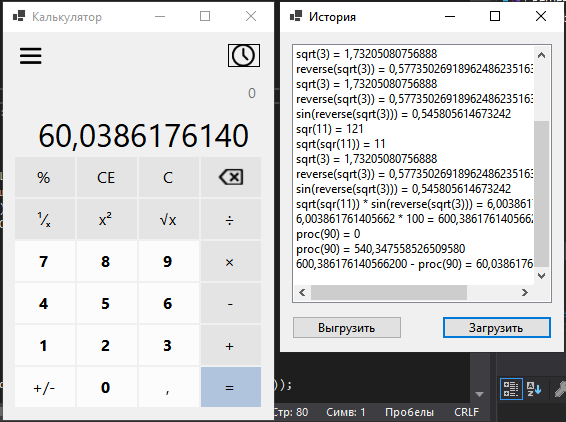
Сохраним историю на диск



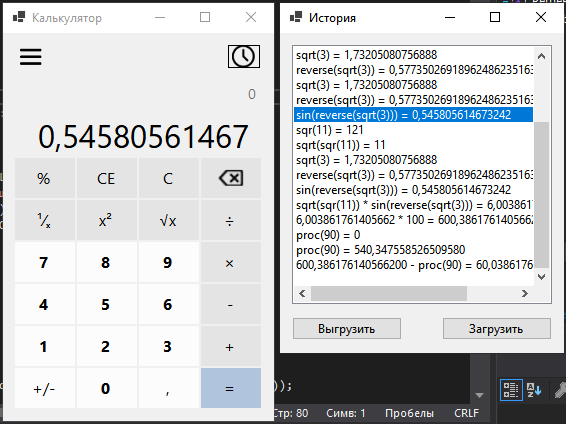
Затем перезапустим приложение и загрузим историю из файла



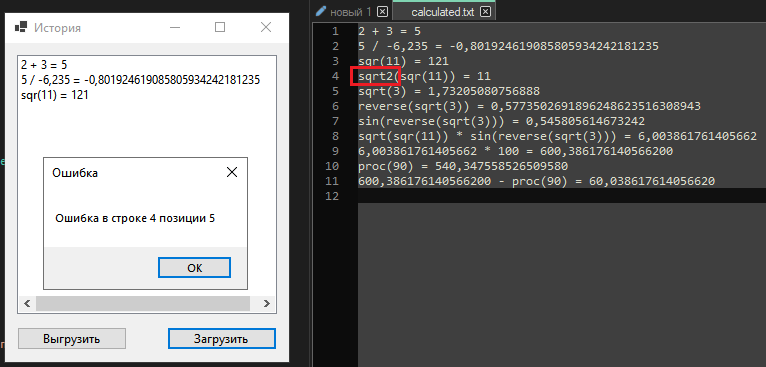
История загрузилась. Однако записей стало больше, т.к. каждая строка рассматривается как отдельное выражение, в то время как используя интерфейс мы сначала вводим число, потом над этим числом берем какую-то функцию, потом над этой функцию берем другую функцию и т.д. Тут же, например, рассматривая строки sqr(11) и sqrt(sqr(11)), мы сначала посчитаем sqr(11), занесем запись в историю, затем перейдем ко второй строке, в ней сначала снова посчитаем sqr(11), занесем это в историю, и только затем посчитаем sqrt(sqr(11)) и занесем это в историю. Из двух строк вышло 3 записи



Также по элементам истории можно переключаться

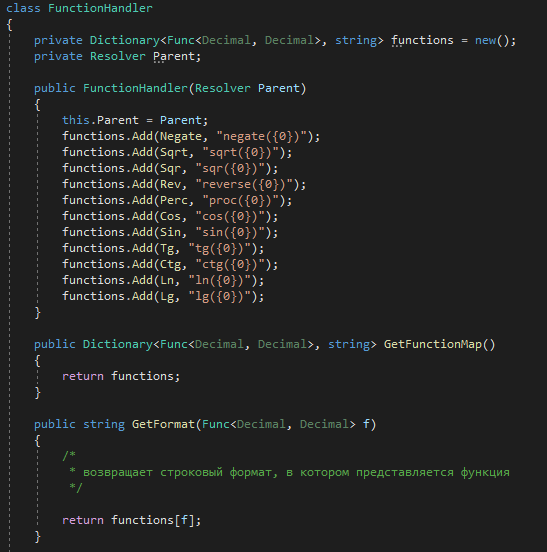


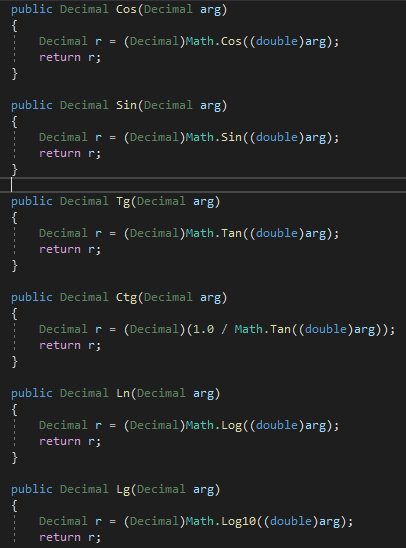
Допустим ошибку и попробуем загрузить такой файл



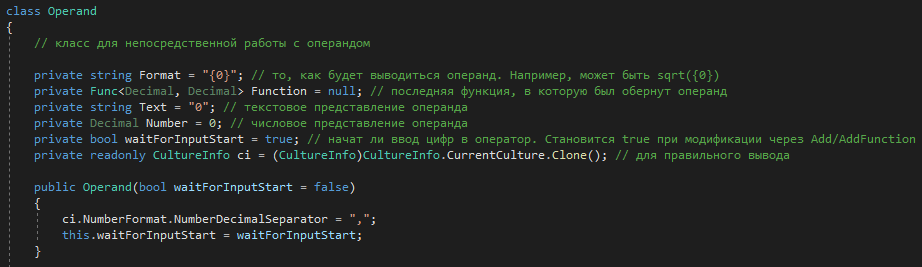
Код программы

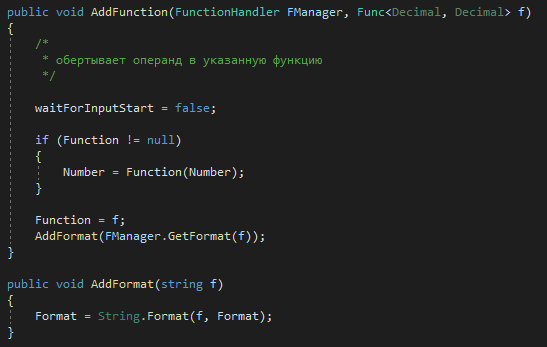
Код класса, отвечающего за функции, применяемые над операндами. Каждой функции соответствует какое-то строковое представление

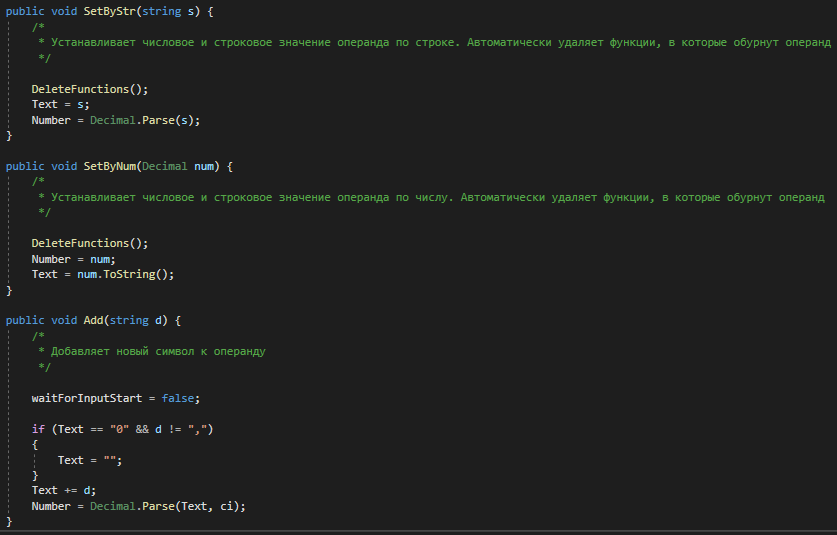




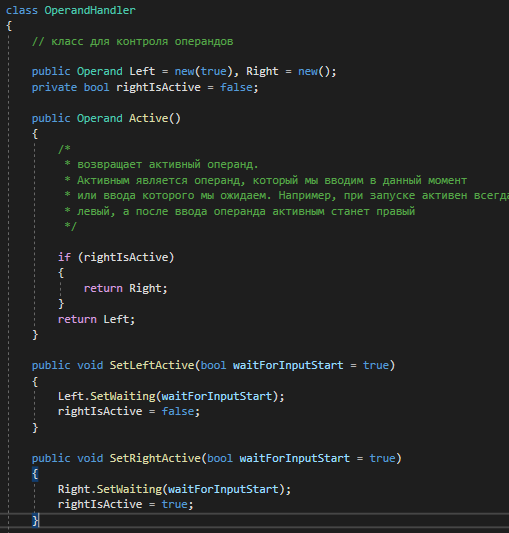
Код класса, отвечающего за хранение состояний оператора, основные методы



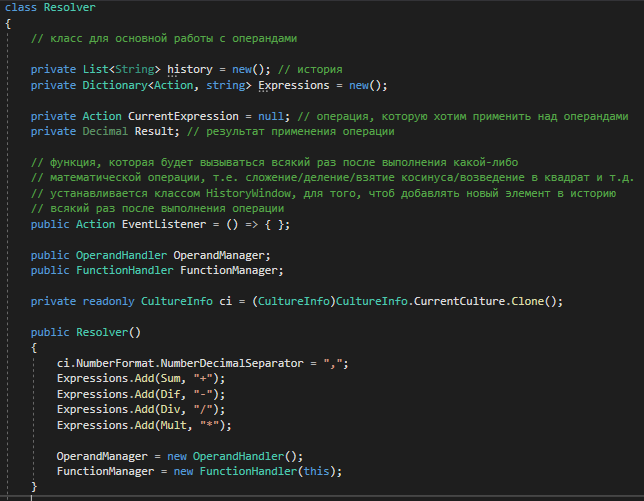


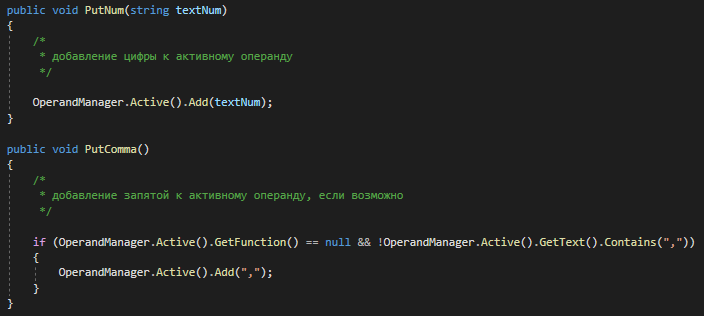


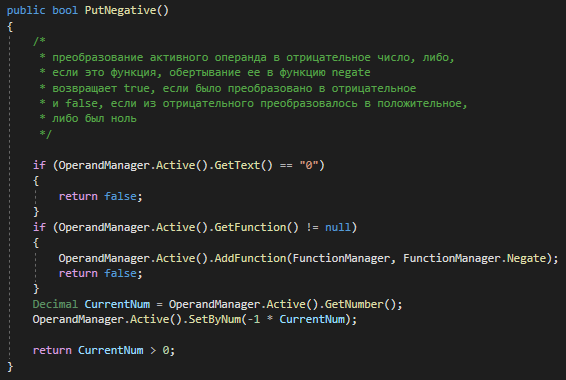
Класс, отвечающий за хранение информации о том, с каким операндом мы сейчас работаем – левым или правым



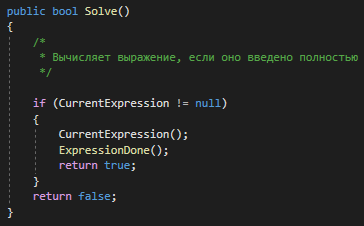
Главный класс, в котором модифицируются операнды, они оборачиваются в функции, над ними выполняются операции. Основные методы:

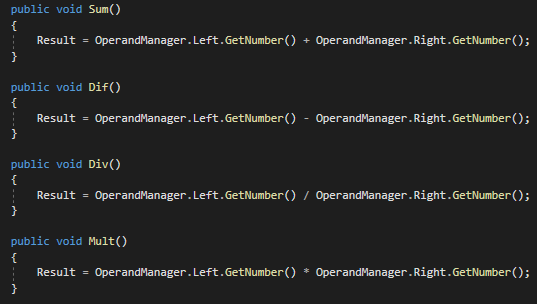


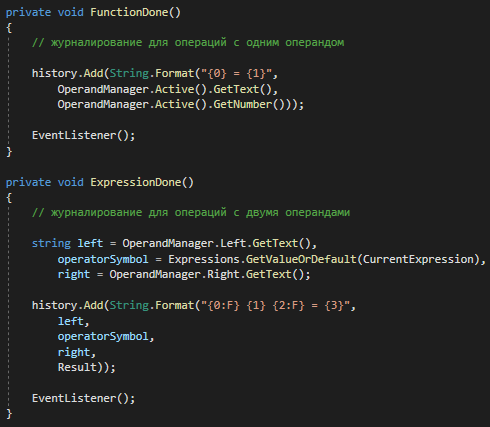


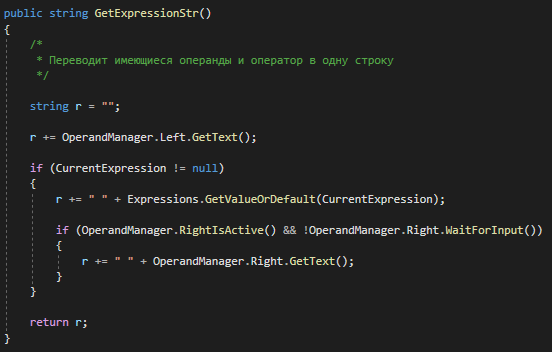






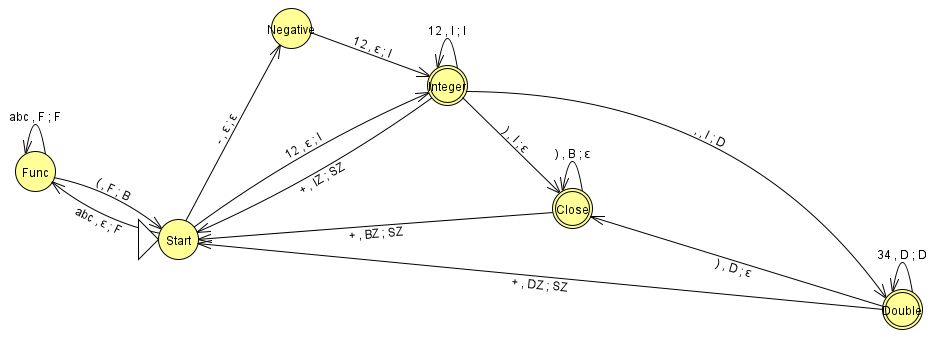






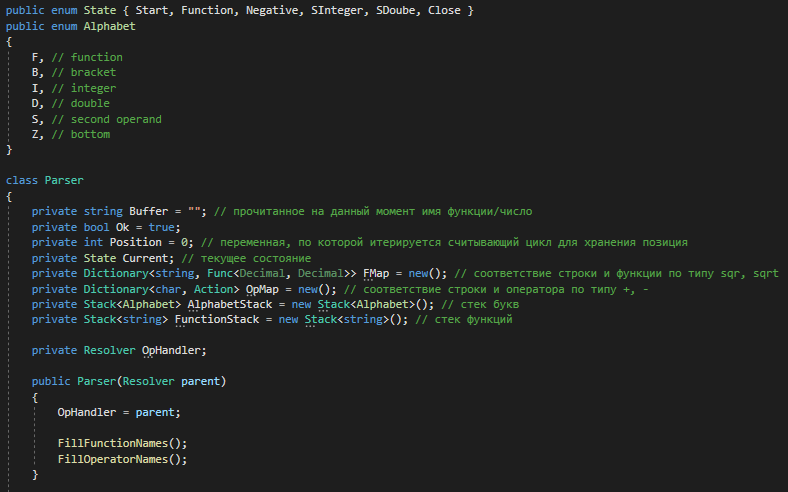
Следующий класс представляет собой аналог анализатора из третьей лабораторной. Он работает по принципу магазинного автомата и обрабатывает строку, приводя ее к объекту, с которым может взаимодействовать программа.

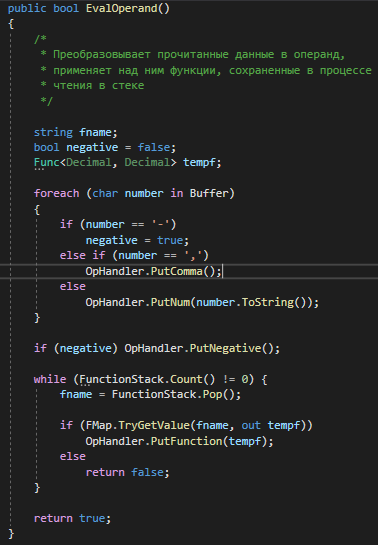
Схема автомата, по которому строился механизм работы класса:

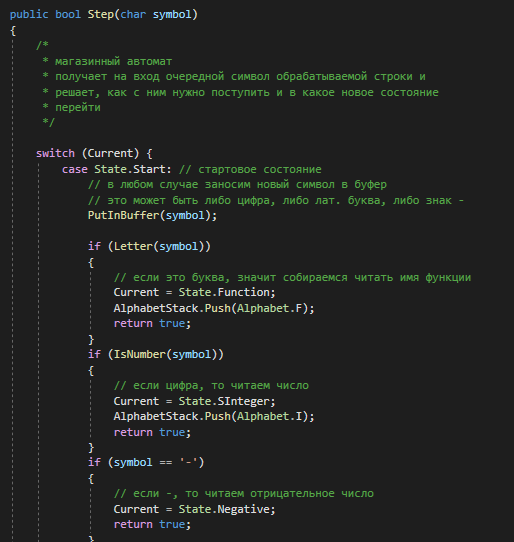


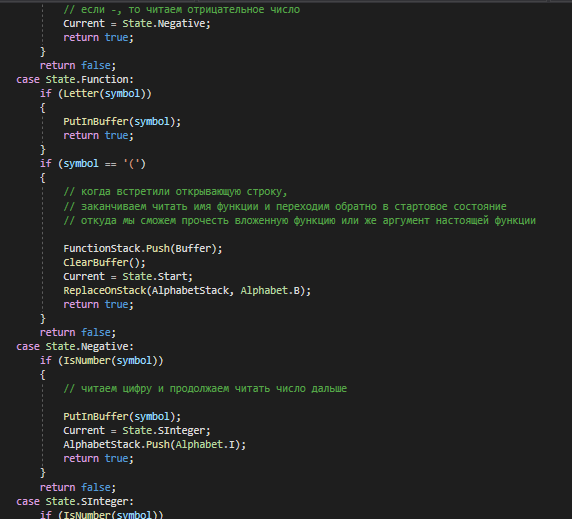
Если изначально (Start) мы начинаем читать буквы, то должно быть это названием функции, поэтому переходим в состояние Func, положив на стек букву F(function), где дочитываем названием до тех пор, пока не встретим открывающую скобку, тогда мы убираем со стека букву F и кладем букву B(bracket), затем переходим обратно в Start. После открывающей скобки может идти название следующей (вложенной) функции, тогда алгоритм повторяется. Либо это может быть цифра/знак минуса. Если это знак минуса, то мы просто считываем его и переходим в состояние Integer, положив на стек букву I(integer). Если это цифра, то считаем ее, кладем на стек букву I и переходим в состояние Integer. Оттуда мы считываем остальные цифры. Если встретили символ «,», то переходим в состояние Double, заменив на стеке букву I на букву D(double). Прочитав закрывающую скобку из Double или из Integer, мы удаляем соответствующую им букву со стека и переходим в состояние Close, где мы читаем все закрывающие скобки и удаляем символы B со стека. Из состояний Integer, Close и Double, прочитав символ оператора мы попадаем обратно в состояние Start. Если до этого момента все было нормально, то сейчас на стеке должно остаться IZ, BZ или DZ. Мы меняем их на SZ. Символ S(second) будет сигнализировать о том, что мы читаем второй операнд. Затем мы считываем второй операнд. Успешными исходами считаются те, где на стеке остались буквы **ISZ / BSZ / DSZ / BZ / IZ / DZ.**

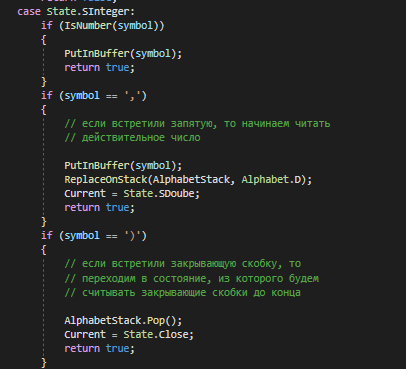
Код класса и основные методы:

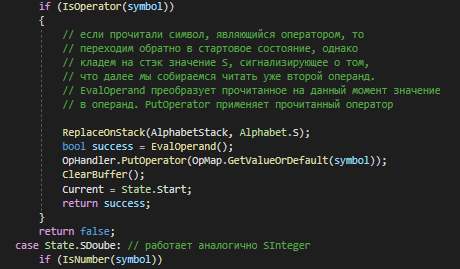


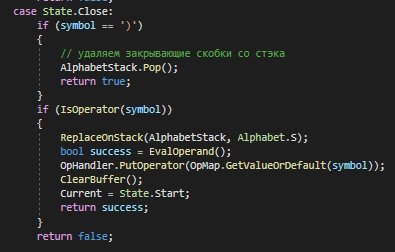


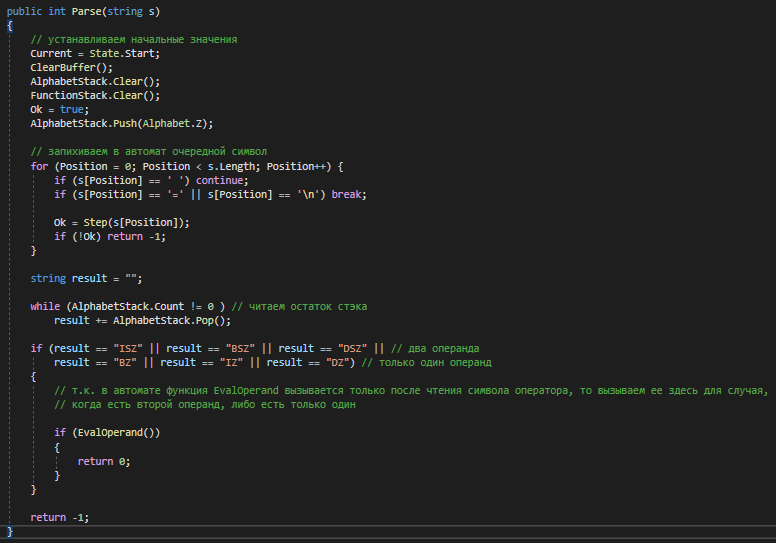




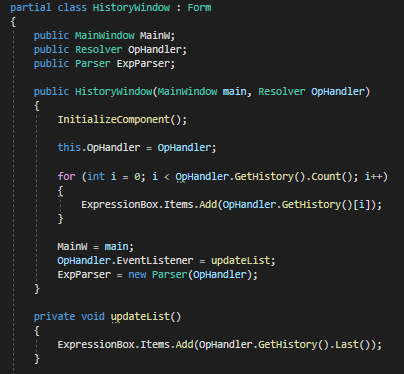


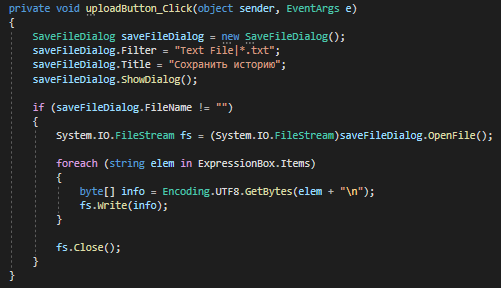


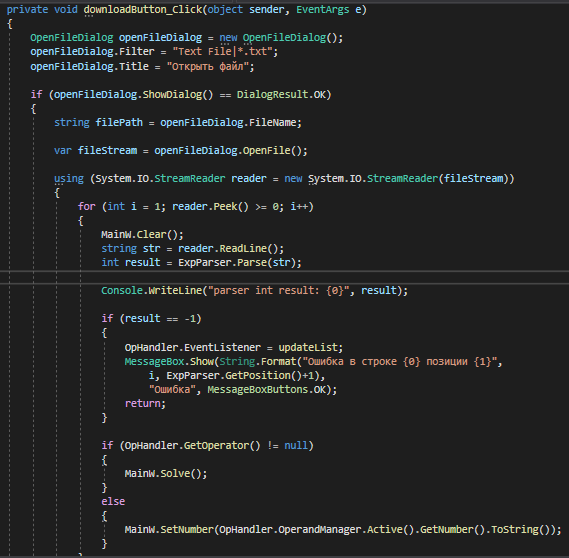


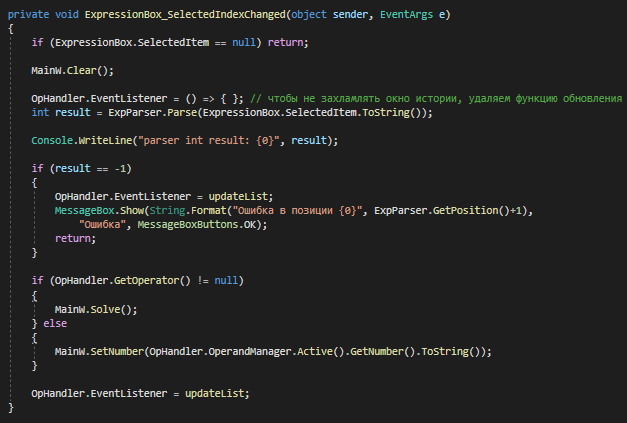


Класс окна истории:









Класс главного окна, основные методы:

