Министерство образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет»

Кафедра «Информационные технологии и автоматизированные системы»

Отчёт

по летней практической работе.

Тема: Синтаксический разборщик математических конструкций.

Выполнил:

Cтудент группы ИВТ-20-2б

Тедеев Александр Зурабович

Проверил:

Доцент кафедры ИТАС

Мухин Олег Игоревич

Пермь 2021

**Список исполнителей**

1. Студент группы ИВТ-20-2б Тедеев Александр Зурабович.

**Реферат**

Сведения об отчете:

21 страница

24 скриншота

3 источника

Ключевые слова:

выражение, конструкция, анализ, вычисление, результат.

**Содержание**

[Введение 5](#_Toc79666057)

[1. Цель 5](#_Toc79666058)

[2. Задачи 5](#_Toc79666059)

[Основная часть 6](#_Toc79666060)

[1. Интерфейс. 6](#_Toc79666061)

[2. Визуальные эффекты. 6](#_Toc79666062)

[3. Программная реализация. 10](#_Toc79666063)

[Заключение 20](#_Toc79666064)

[Источники и литература 21](#_Toc79666065)

# **Введение**

## 1. Цель.

1.1. Разработать программу, которая сможет разбирать математические выражения и решать их.

## 2. Задачи.

* 1. Разработать понятный интерфейс для программы.
  2. Разработать алгоритм по анализу вводимого выражения.
  3. Разработать алгоритм по решению поступившего выражения.
  4. Проверить работу данной программы.

# **Основная часть**

## 1. Интерфейс.

1.1.В качестве интерфейса было решено реализовать окно калькулятора. Каждый человек пользовался калькулятором, поэтому работа с такой программой не вызовет никаких сложностей, даже без инструкции.

1.2. Для реализации интерфейса была выбрана программа Qt (результат представлен на рисунке 1), так как программа была написана средствами языка программирования C++.

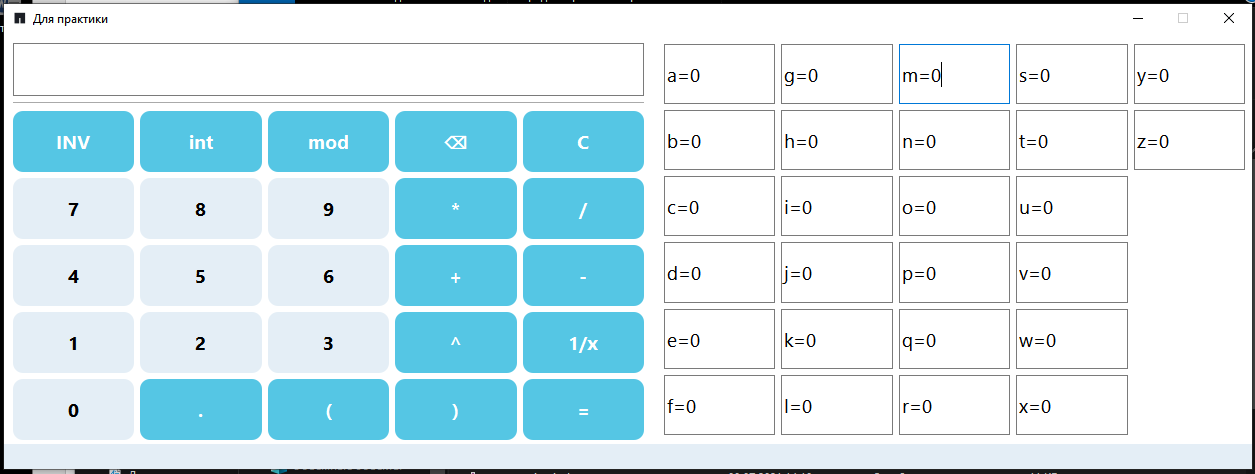


рисунок 1.

## 2. Визуальные эффекты.

2.1. При нажатии каждая кнопка меняет свой цвет для удобного отслеживания нажатий. Резульат представлен на рисунках 2, 3.

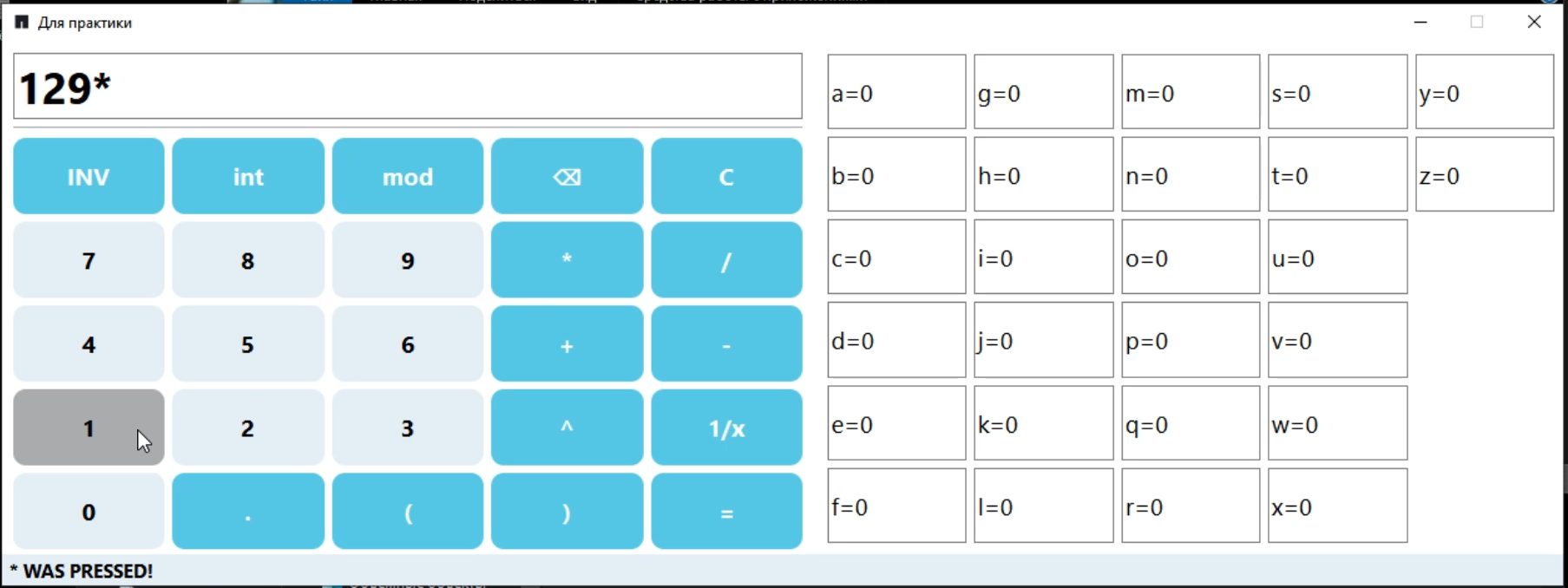


рисунок 2.

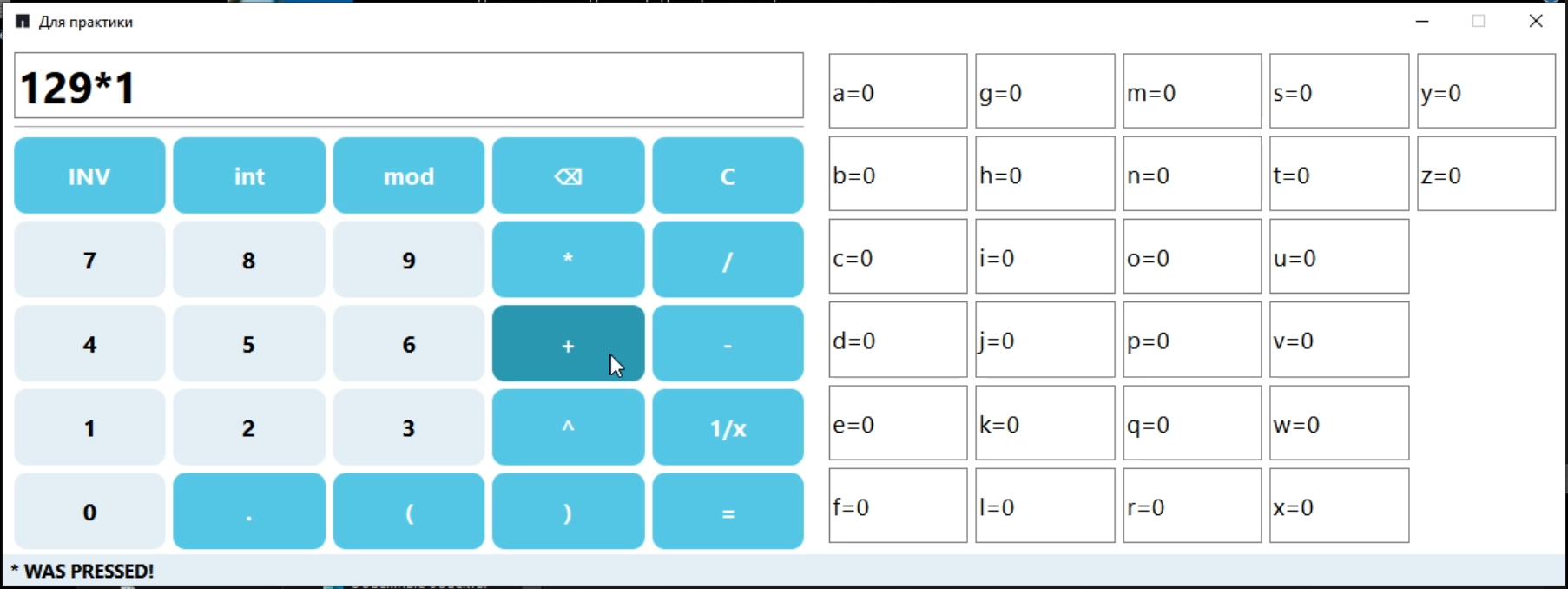


рисунок 3.

2.2. При поступлении длинного выражения, не входящего в ширину окна, размеры отображаемого выражения уменьшаются. Результат представлен на рисунке 4.

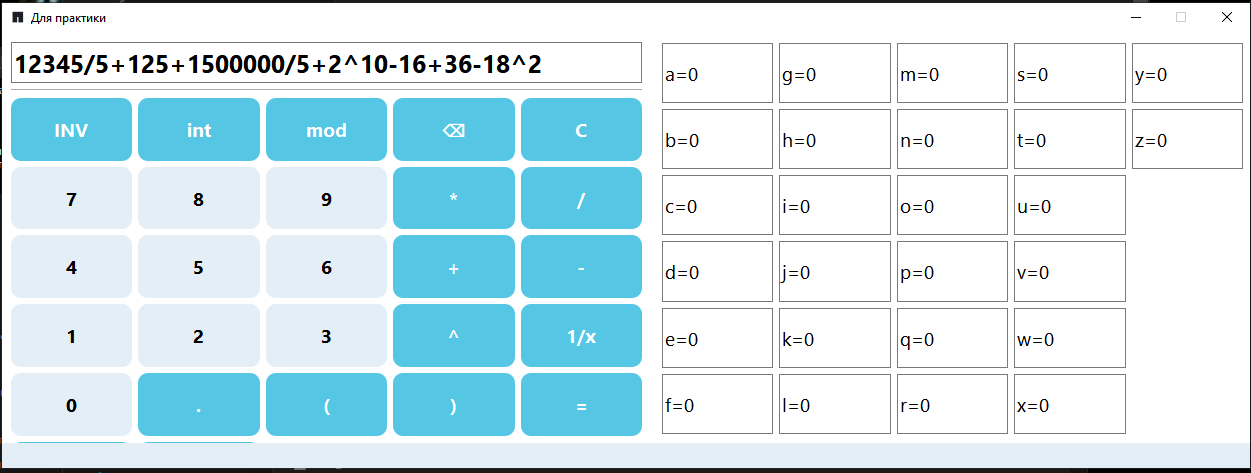


рисунок 4.

2.3. В случае уменьшения текущего выражения размер шрифта возвращаются к изначальному. Результат представлен на рисунке 5.

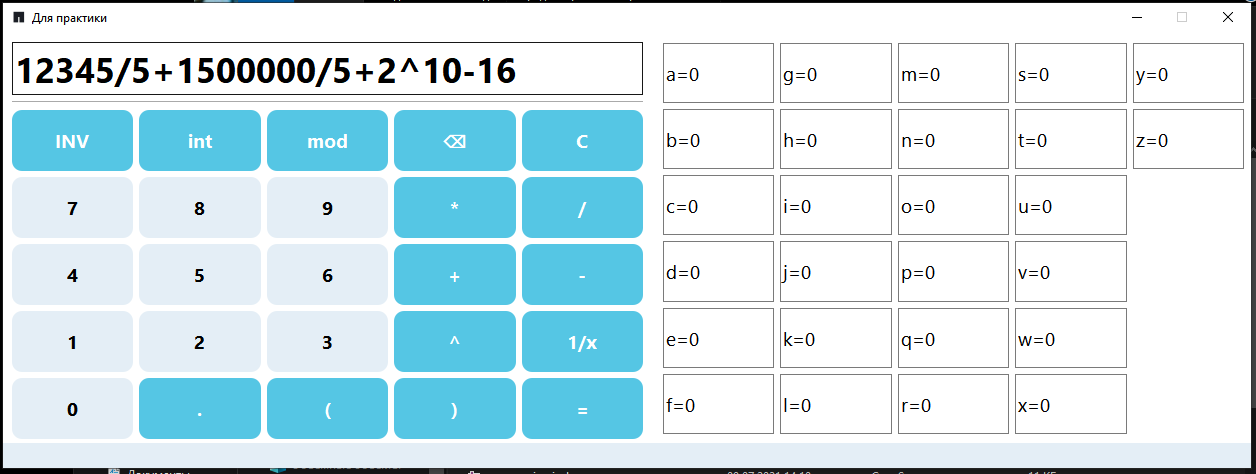


рисунок 5.

2.4. Кроме изменения цвета кнопок при их нажатии конкретно для ввода операций, таких как сложение, вычитание, умножение, деление, изменения сообщаются в, так называемом, статус баре внизу слева. Результат представлен на рисунках 2, 3, 4.

2.5. После нажатия знака равенства происходит анализ конструкции и подсчёт выражения. Действия, которые были выполнены для нахождения результата, выводятся в другом открывающемся окне. Результат представлен на рисунке 6.

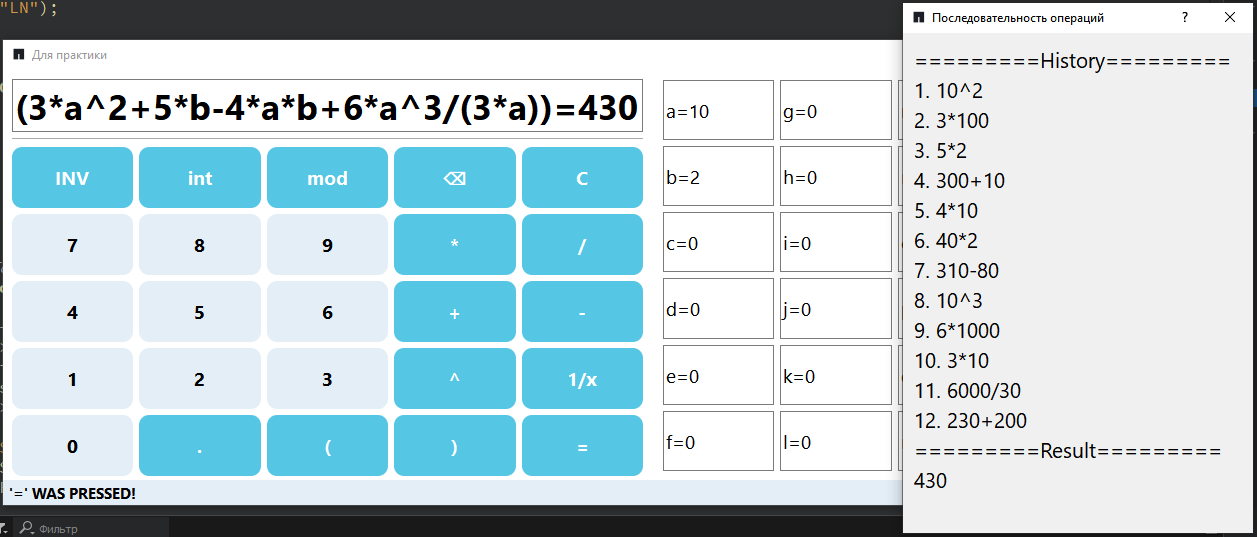


рисунок 6.

2.6. После нажатия кнопки INV раскрывается панель с дополнительными операциями (рисунки 7, 8, 9, 10).

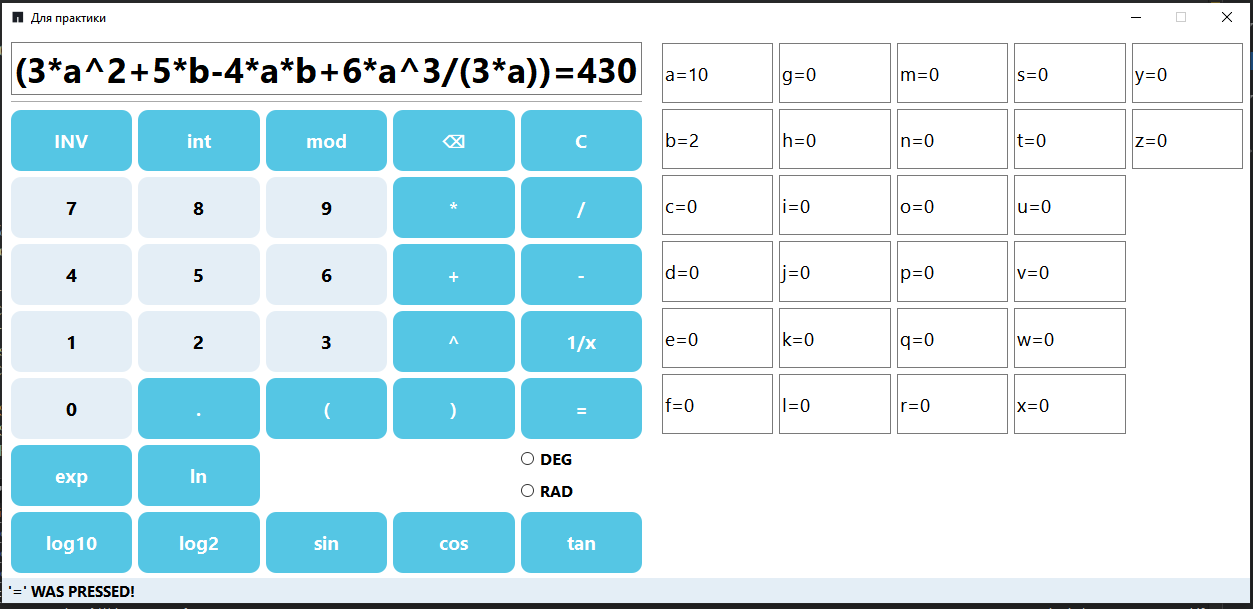


рисунок 7.

2.7. При вводе некорректных конструкций во втором окне пользователю выводятся ошибки (рисунок 8).

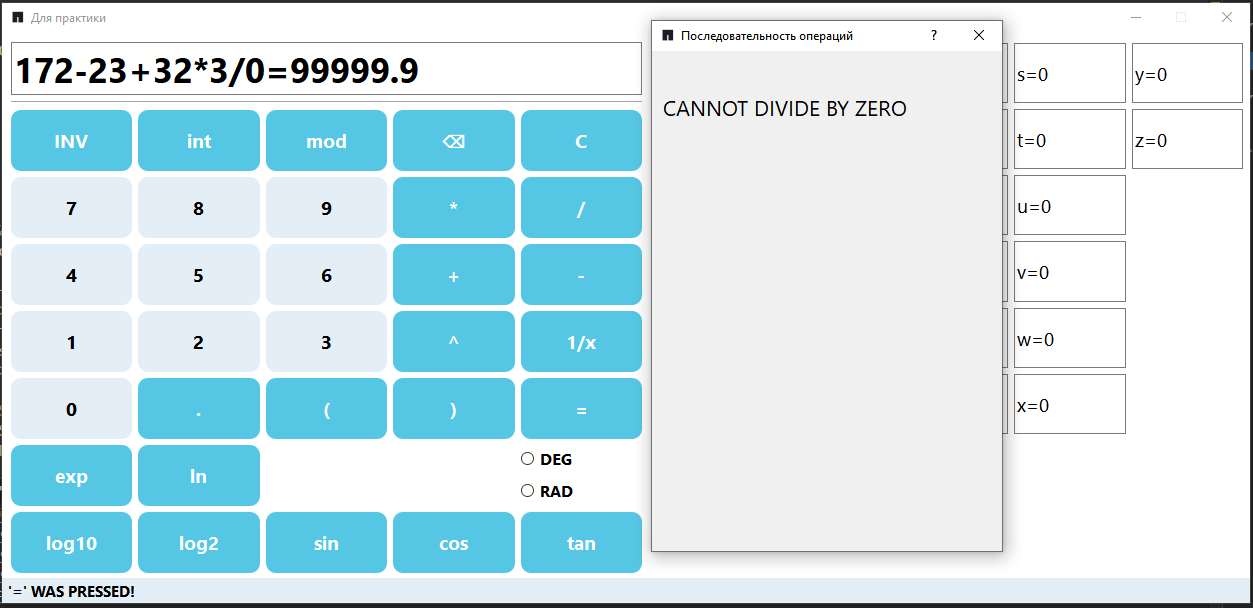


рисунок 8.

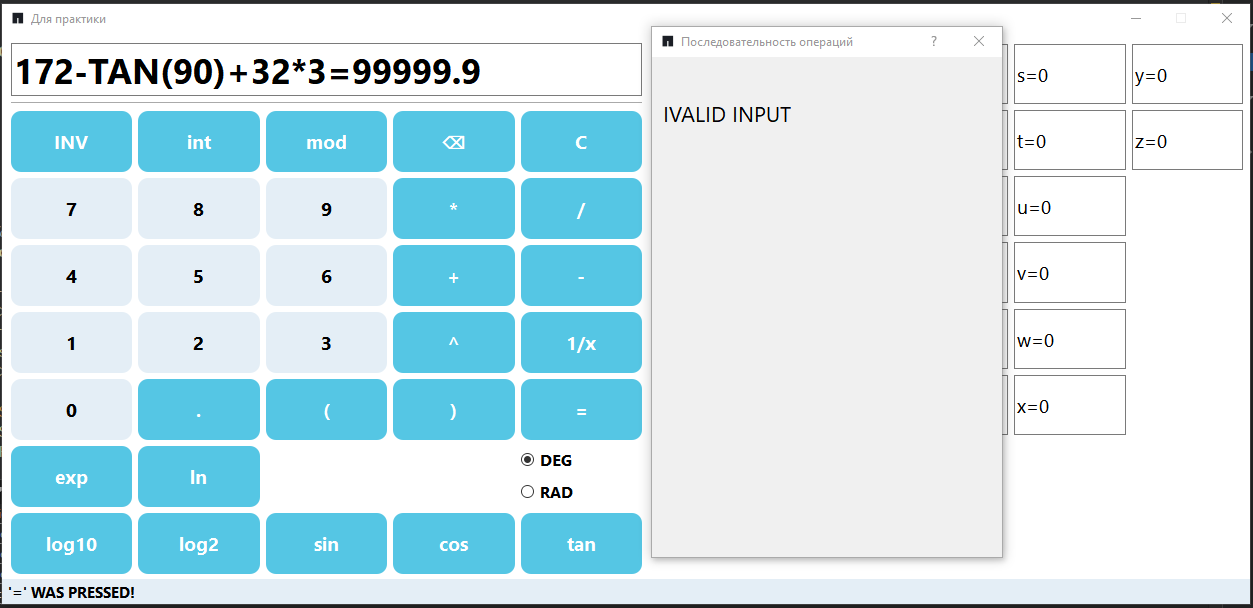


рисунок 9.

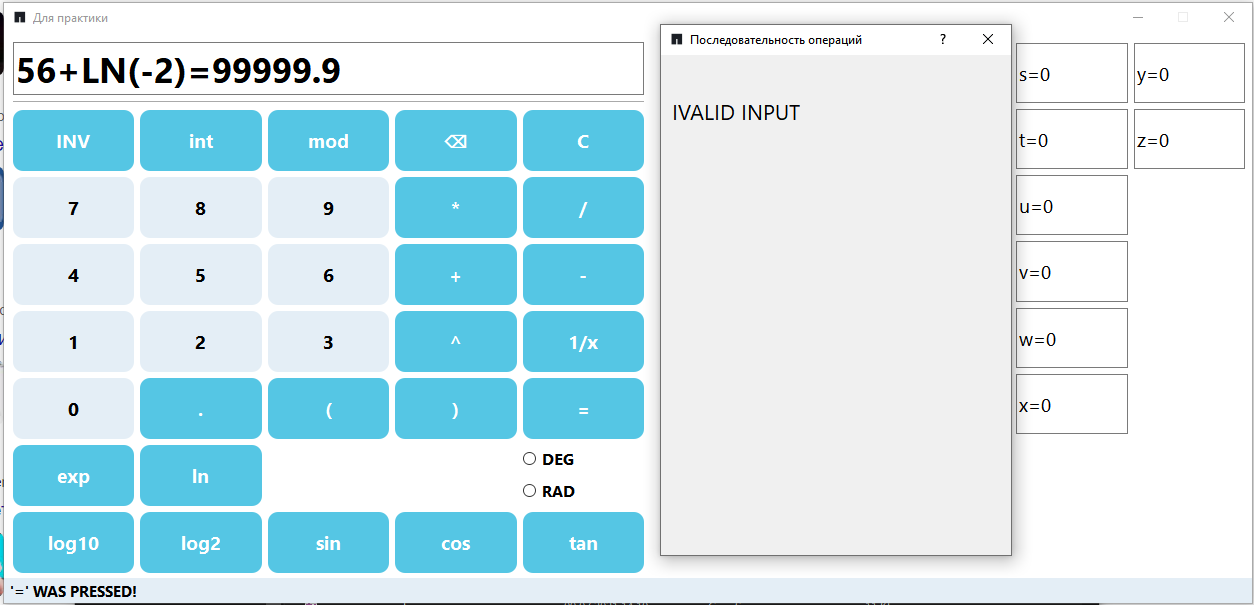


рисунок 10.

## 3. Программная реализация.

3.1. Для составления кода на C++ необходимо иметь готовый алгоритм:

3.1.1. Поступающее в компьютер выражение не должно содержать только цифры, знаки операций и круглые скобки, так как другие символы будут приводить к «не читаемости» выражения. Код на C++ представлен на рисунке 12.

3.1.2. Если выражение не содержит лишних знаков, то полученное выражение преобразуется в обратную польскую запись, пример представлен на рисунке 17. Реализация в коде (функция toPostfix()) представлена на рисунке 18.

3.1.3. Произвести вычисления над полученным выражением в обратной польской записи (функция Calculate()).

3.1.4. Вывести на экран последовательность действий, выполненных над выражением, для нахождения решения введённой конструкции. Реализация в коде – функции Calculate() и PrintExprInTheWin(), представлена на рисунках 20-22 и 23-24 соответственно.

3.2. Поступающая конструкция рассматривается после нажатия знака равно, когда вызывается событие после нажатия этой кнопки (рисунок 11).

3.3. Решение запускается только когда выражение не содержит символы кириллицы, выражение непустое, когда выражение не содержит знака =.

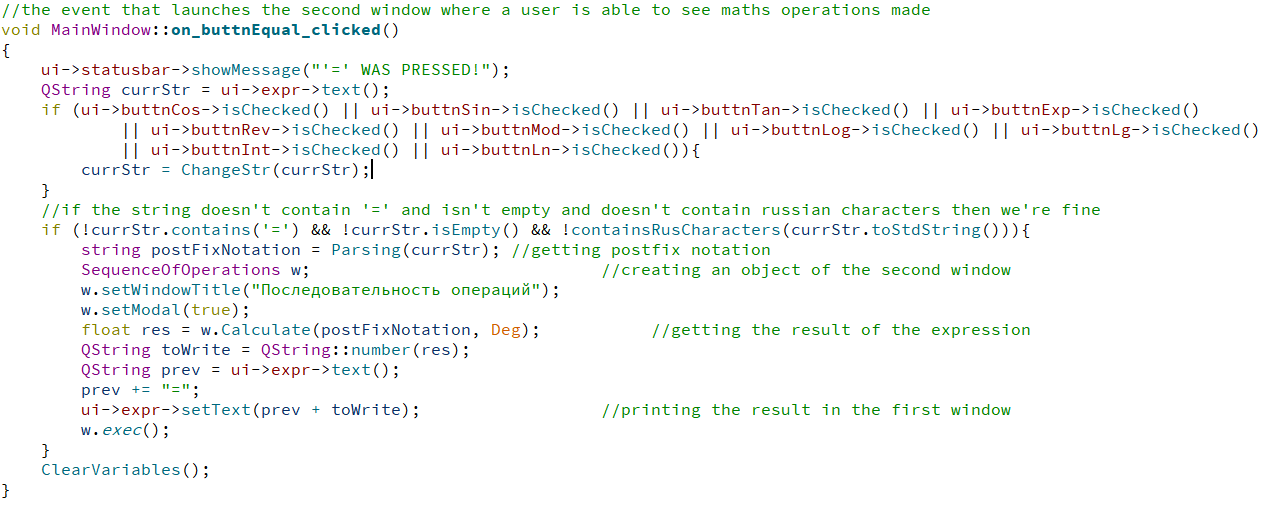


рисунок 11.

3.4. В функции Check() проверяется каждый символ выражения, если найдена хотя бы одна буква, то функция возвращает значение false в функцию Parsing(), реализация представлена на рисунке 8.

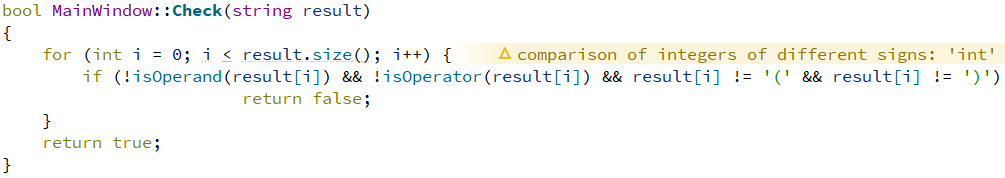


рисунок 12 – функция проверки поступившего выражения.

3.5. В функции Parsing (рисунок 13) производится запуск функции Check() в тернарном операторе; в случае, если выражение содержит латинские, то функция вызывает функцию convertString() (рисунок 9), иначе возвращает преобразованное выражение с помощью функции toPostfix() (рисунок 10).

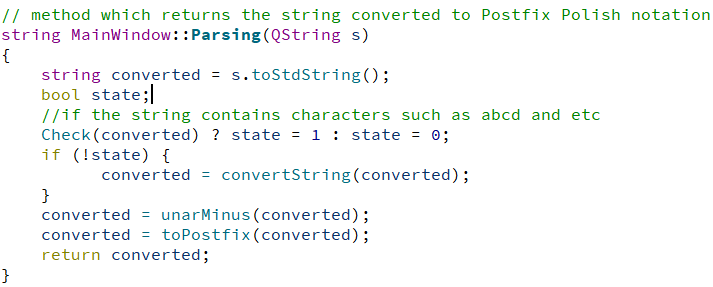


рисунок 13.

3.6. В функции convertString() (реализация на рисунках 14-16) происходит замена строки с буквами на строку с числами, например, 5a-2b -> 5\*1-2\*3, где значения берутся из таблицы рядом с калькулятором (рисунки 1-10).

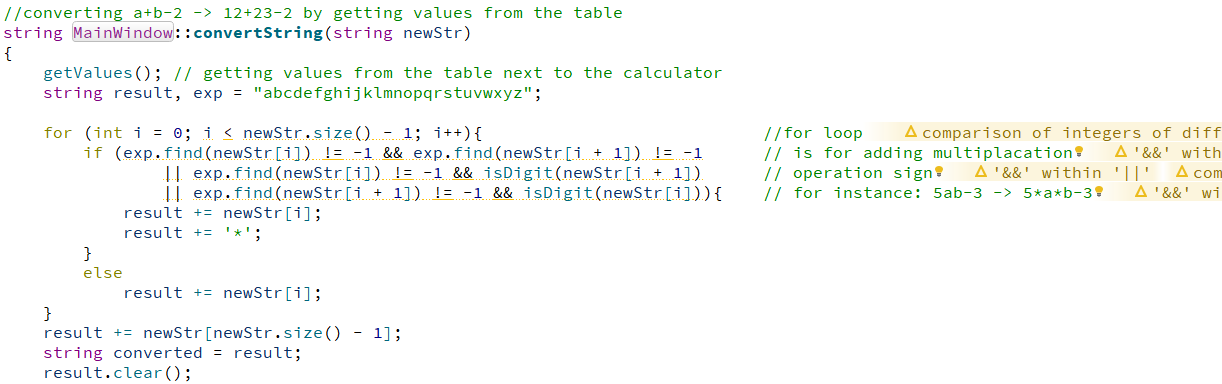
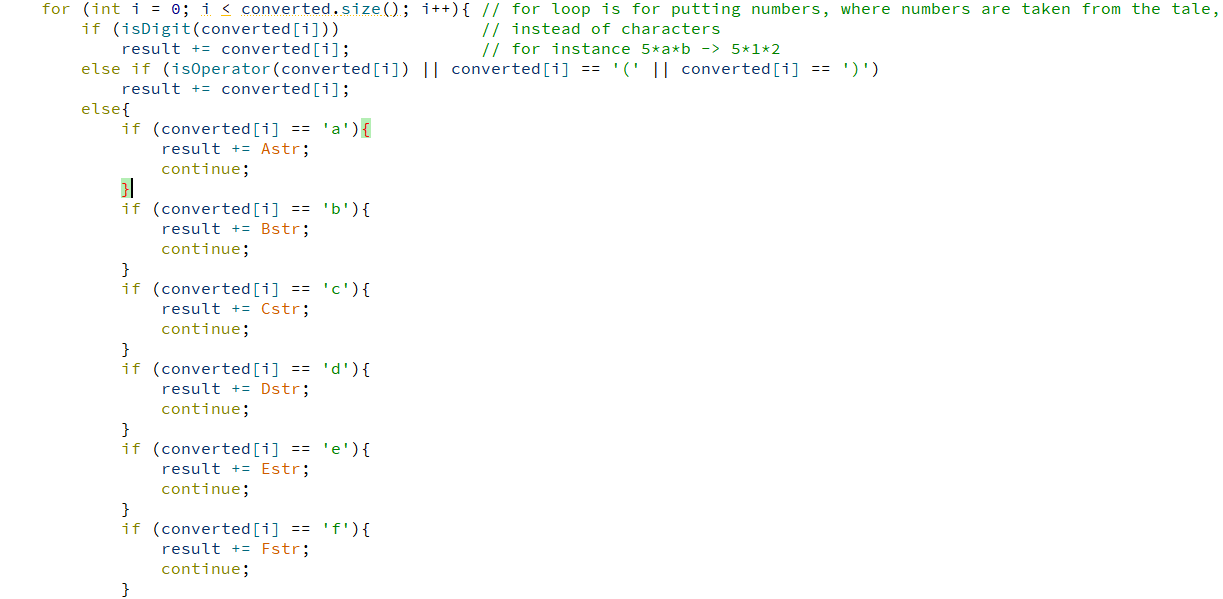
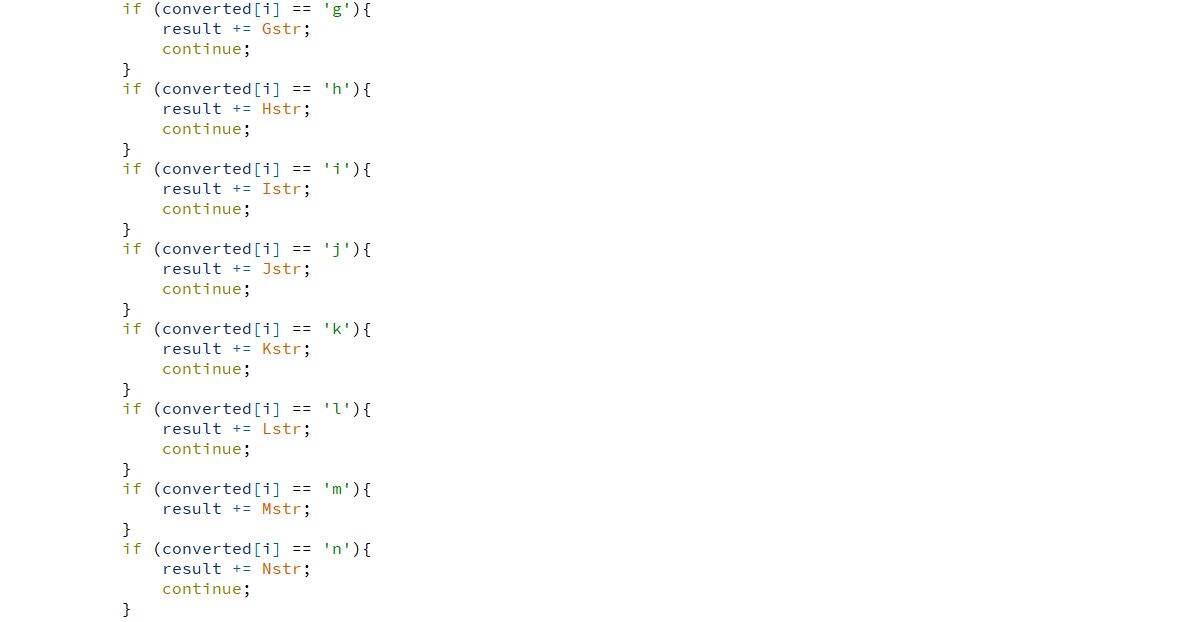


рисунок 14.





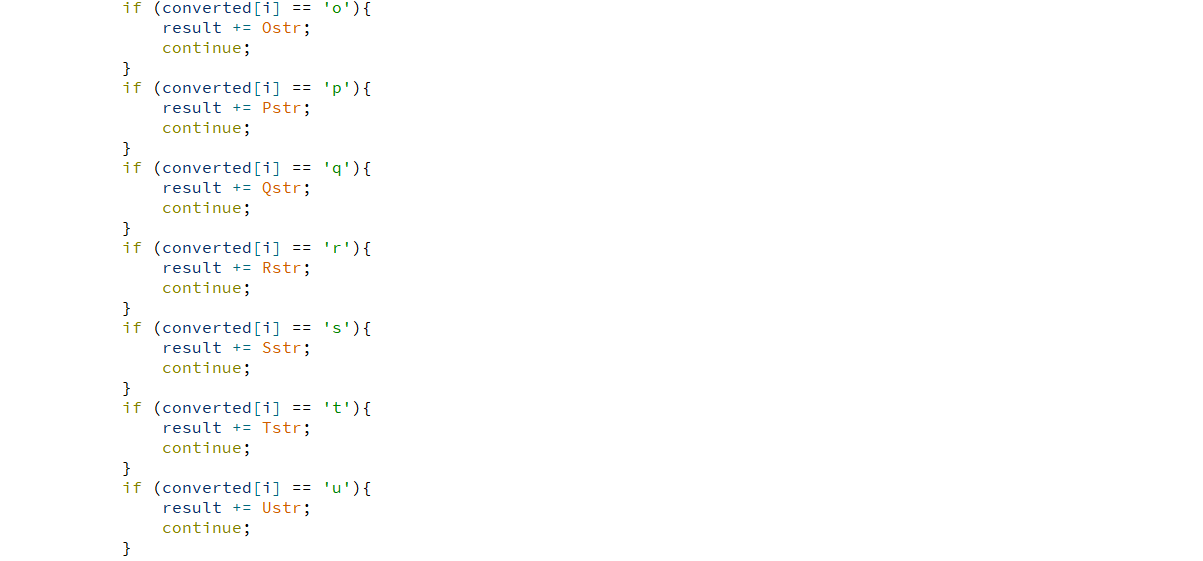


рисунок 15.



рисунок 16.



рисунок 17 – наглядное сравнение преобразованного выражения с изначальным.

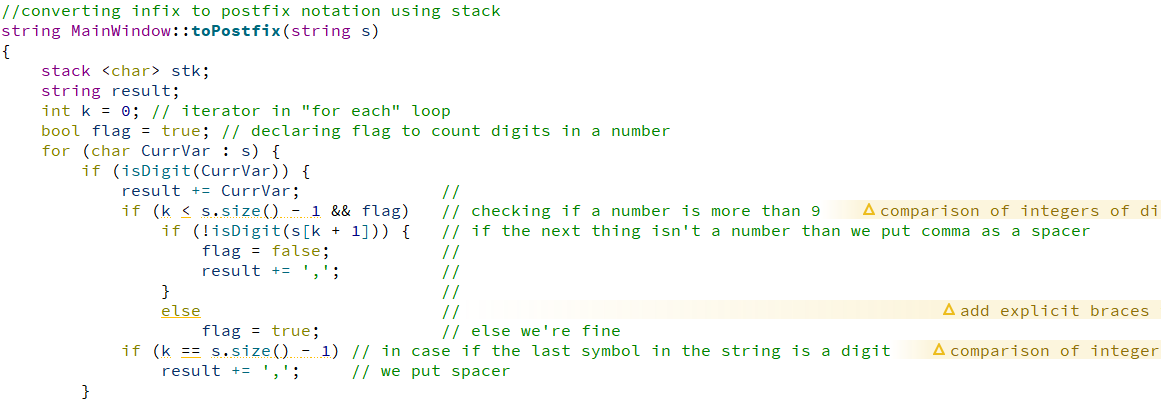
3.7. В функции toPostfix() производится перевод выражения в постфиксную нотацию. В цикле проверяется каждый символ:

3.7.1. Если символ – цифра, то она добавляется в выходную строку и отделяется запятой. В случае, если число состоит из нескольких цифр, то запятая не ставится, пока не найдётся последняя цифра данного числа.

3.7.2. Если символ – оператор (+, -, \*, /), то проверяется приоритетность в функции Priority() (рисунок 19). Если приоритетность оператора выше приоритетности оператора из стека, то данный оператор добавляется в стек; иначе все операторы из стека приоритетность которых выше, чем у данного оператора помещаются в выходную строку из стека, а данный оператор помещается в стек.

3.7.3. Если символ – это открывающая круглая скобка, то помещается в стек.

3.7.4. Если символ – закрывающая круглая скобка, то все операторы из стека помещаются в выходную строку, пока не встречается символ открывающей строки.





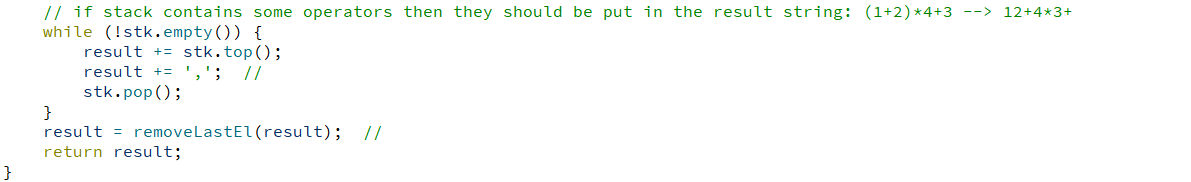


рисунок 18 – преобразование в обратную польскую запись.

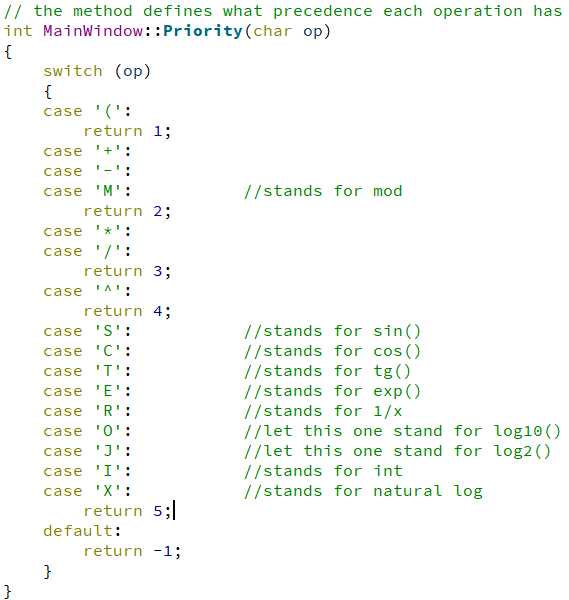


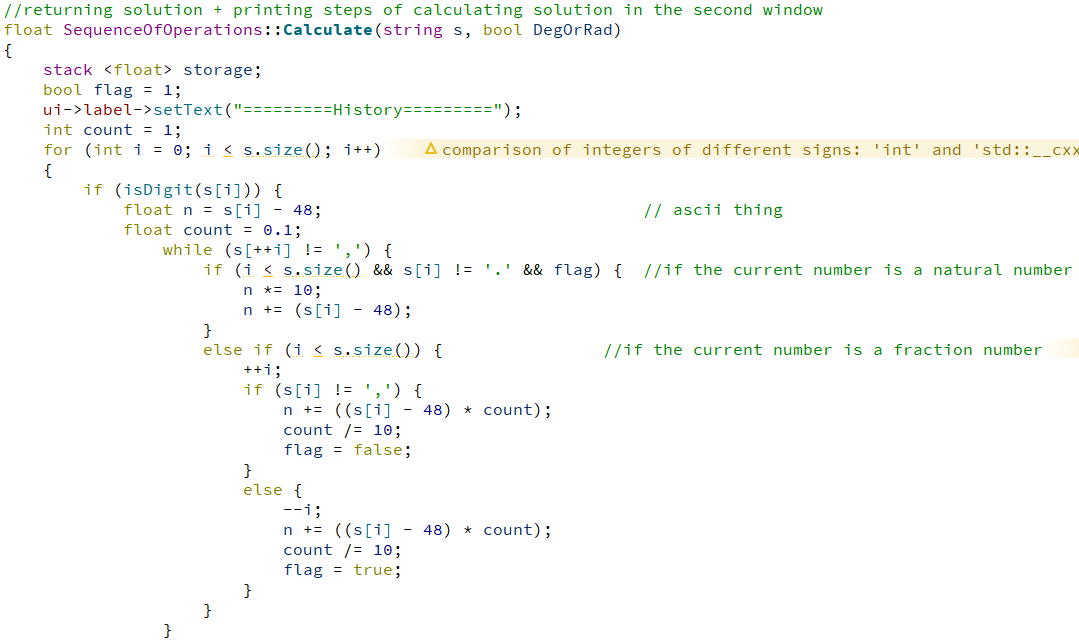
рисунок 19.

3.8. В функции Calculate() (рисунки 20-22) происходит вычисление по постфиксной нотации. Происходит проход по строке с проверкой каждого символа:

3.8.1. Если символ – это цифра, то происходит проход далее по строке, пока не встретится разделитель – запятая. Далее число помещается в стек.

3.8.2. Если символ – это оператор, то достаются два числа из стека и в зависимости от оператора выполняется операция и выводятся на экран два числа и операция, производимая над ними, (рисунки 23-24).

3.8.3. В случае, если никаких ошибок не найдено, то возвращается результат всех действий.



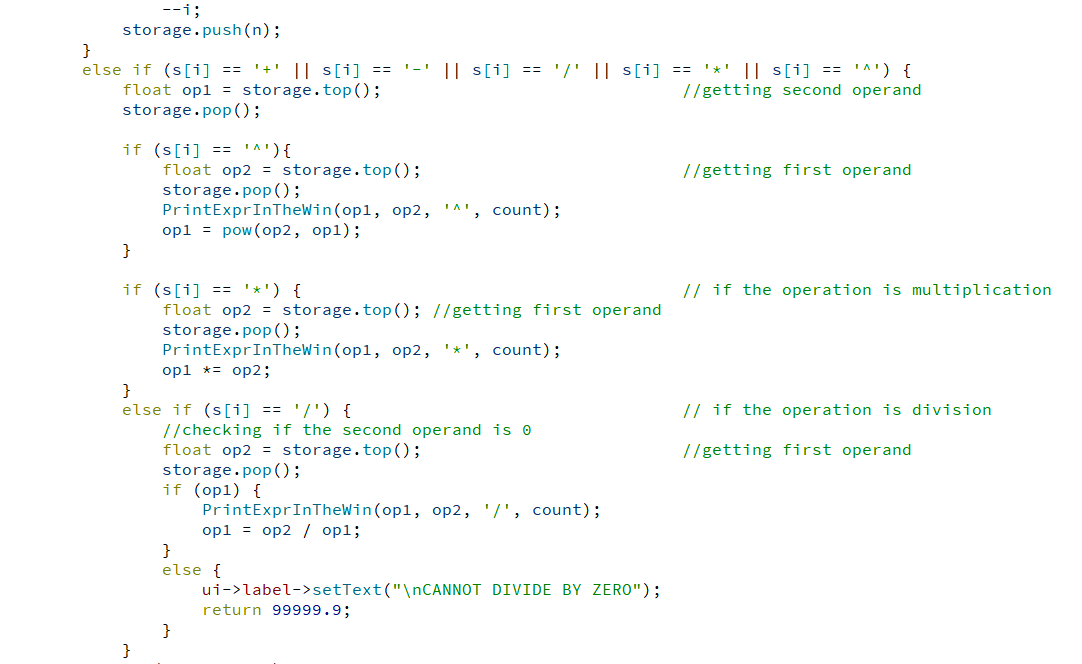


рисунок 20.



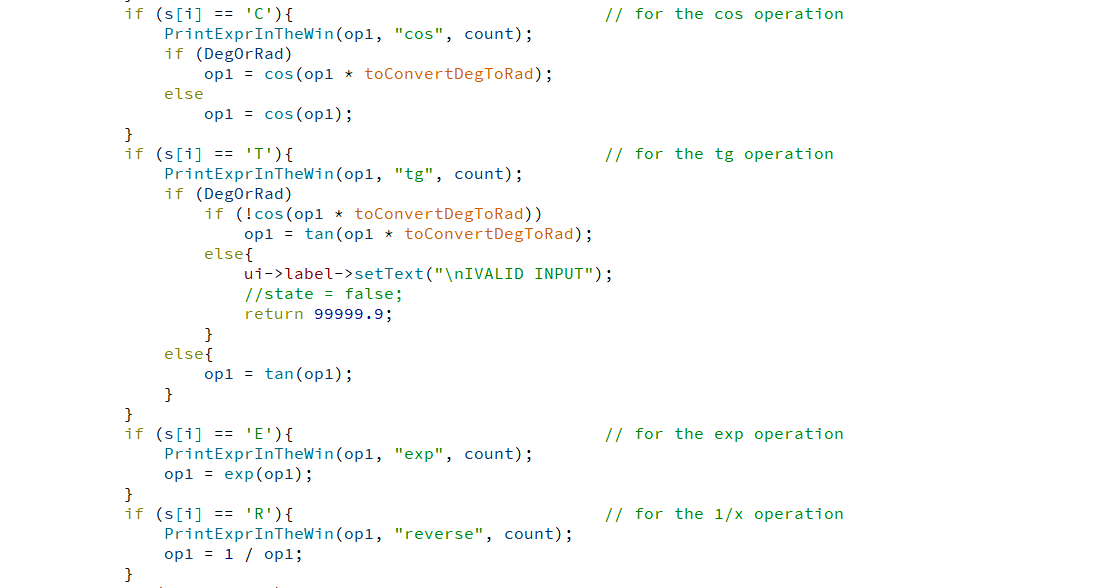


рисунок 21.



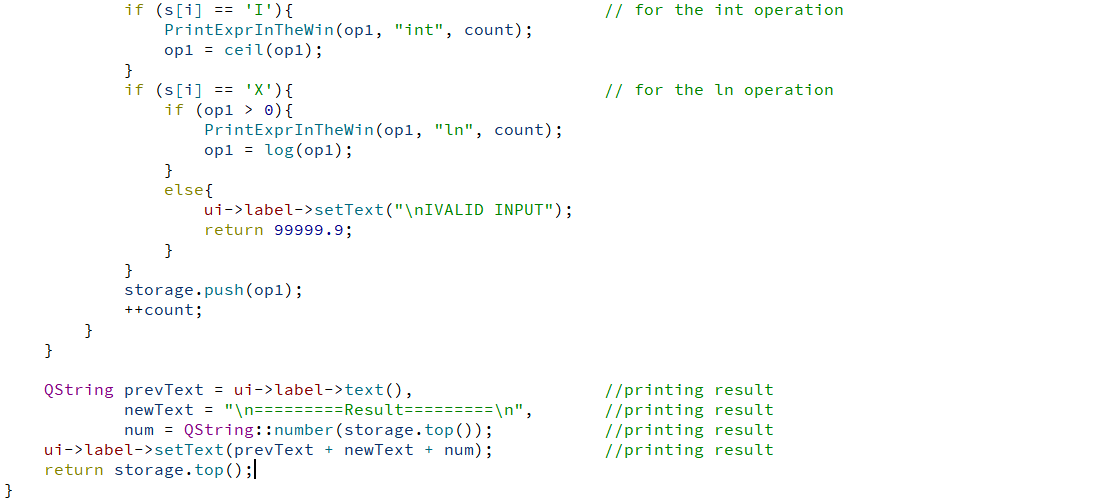


рисунок 22 – получение решения из обратной польской записи.

3.9. Печать на экран маленького выражения.

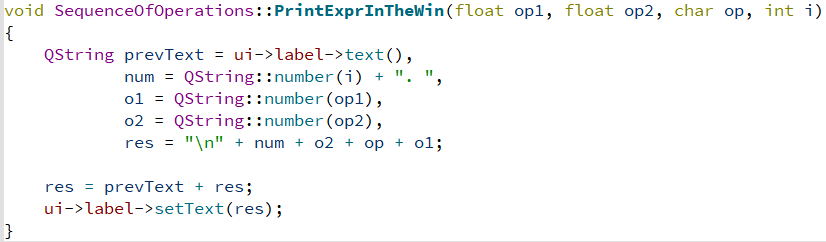


рисунок 23 – печать действия во второе окно.

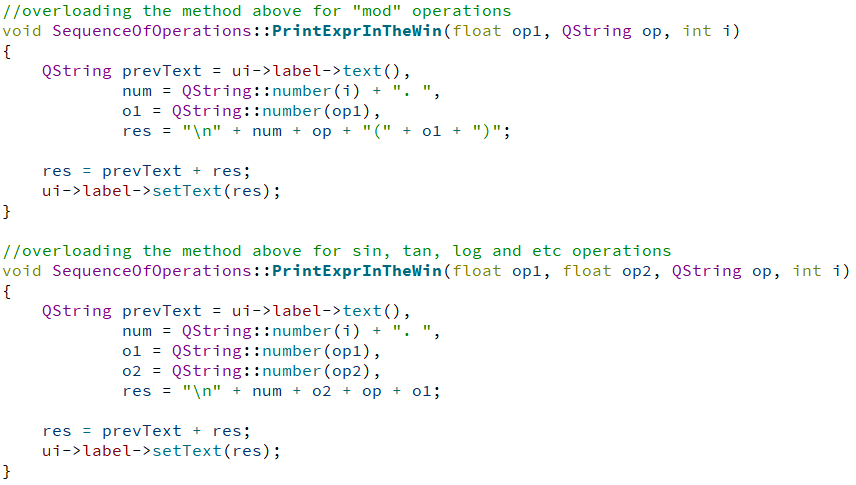


рисунок 24 – перегрузки функции PrintExprInTheWin.

# **Заключение**

Решение поставленных задач помогло осознать действия разборщика математических выражений и построить понятную и эффективную программу для демонстрации последовательных действий по вычислению поступившего выражения.

# **Источники и литература**

1. Гост - <https://docs.cntd.ru/document/1200026224>
2. Обратная польская запись - <https://habr.com/ru/post/100869/>
3. Сборка проекта - <https://evileg.com/ru/post/161/>