Министерство образования Российской Федерации

Пермский Национальный Исследовательский Политехнический Университет

Кафедра «Информационные технологии и автоматизированные системы»

Учебно-исследовательская работа

Отчёт по практической работе

Тема: «Синтаксический разборщик математических конструкций»

Выполнил:

студент группы ИВТ-20-2б

Тедеев Александр Зурабович

Проверил:

доцент кафедры ИТАС

Мухин О. И.

Пермь 2021

**Содержание**

[1. Введение 3](#_Toc76143852)

[1.1 Цель 3](#_Toc76143853)

[1.2 Задачи 3](#_Toc76143854)

[2. Основная часть 4](#_Toc76143855)

[1.1 Интерфейс 4](#_Toc76143856)

[2.2 Программная реализация 8](#_Toc76143857)

[3. Вывод 14](#_Toc76143858)

# **1. Введение**

## **1.1 Цель**

Написать программу, которая сможет разбирать математические выражения и решать их.

## **1.2 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.

# **2. Основная часть**

## **1.1 Интерфейс**

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

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

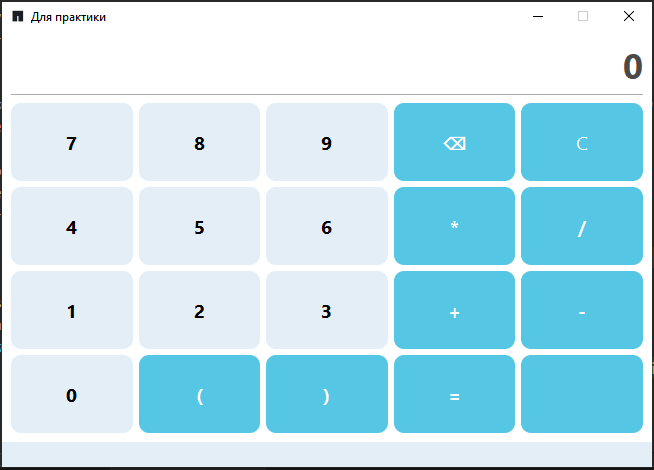


рисунок 1.

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

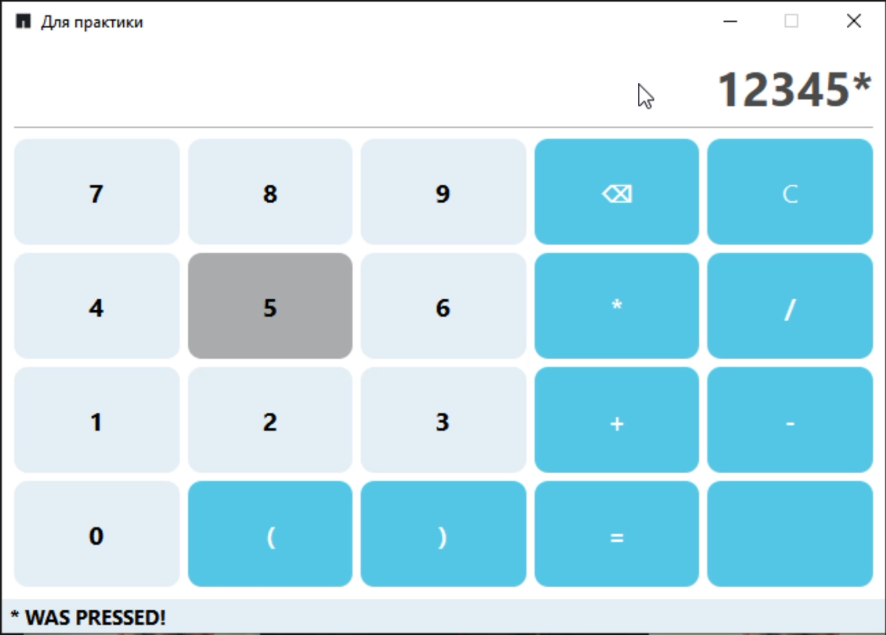


рисунок 2.

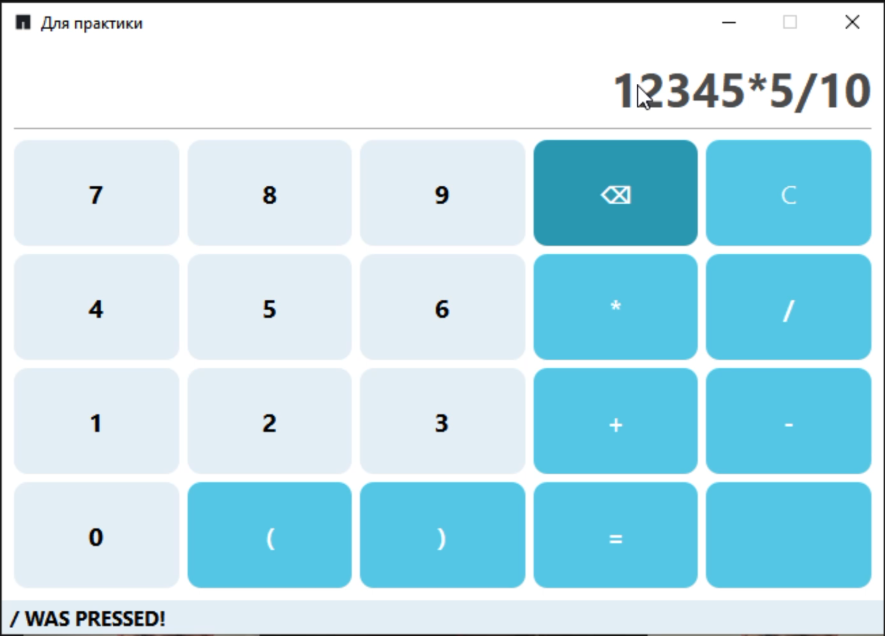


рисунок 3.

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

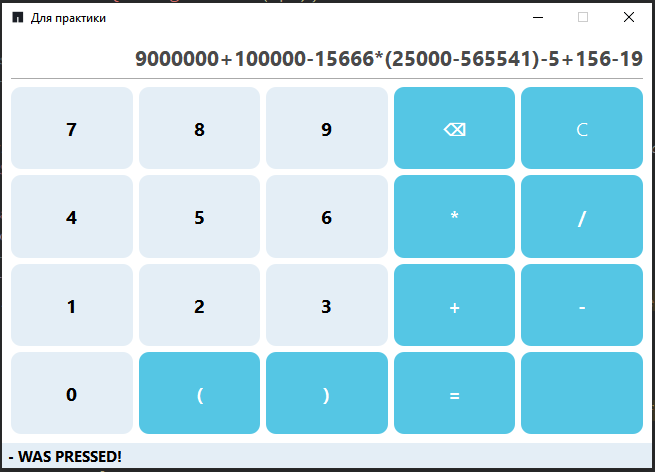


рисунок 4.

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

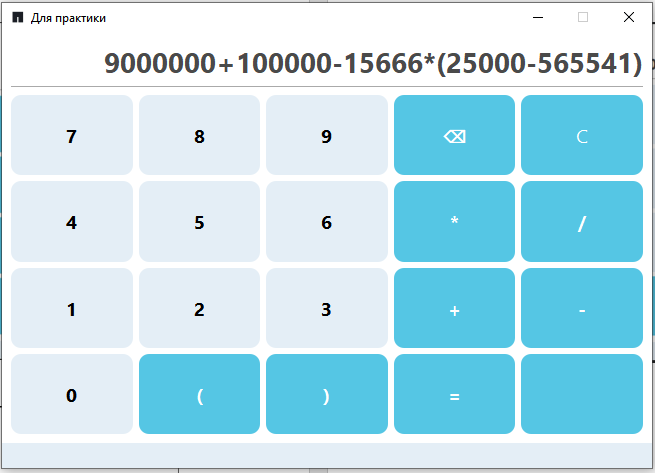


рисунок 5.

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.

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.

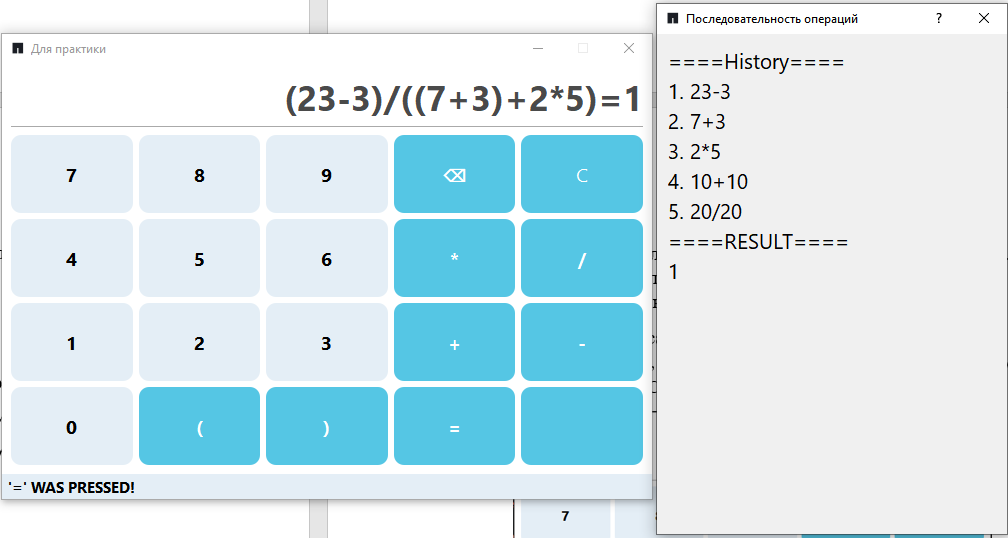


рисунок 6.

## **2.2 Программная реализация**

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

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 7.
2. Если выражение не содержит лишних знаков, то полученное выражение преобразуется в обратную польскую запись, пример представлен на рисунке 9. Реализация в коде (функция toPostfix()) представлена на рисунке 9.
3. Произвести вычисления над полученным выражением в обратной польской записи (функция Calculate()).
4. Вывести на экран последовательность действий, выполненных над выражением, для нахождения решения введённой конструкции. Реализация в коде – функции Calculate() и PrintExprInTheWin(), представлена на рисунках 12, 13.

1.1. В функции Check() проверяется каждый символ выражения, если найден хотя бы один не подходящий, то функция возвращает значение false в функцию Parsing(), реализация представлена на рисунке 8.

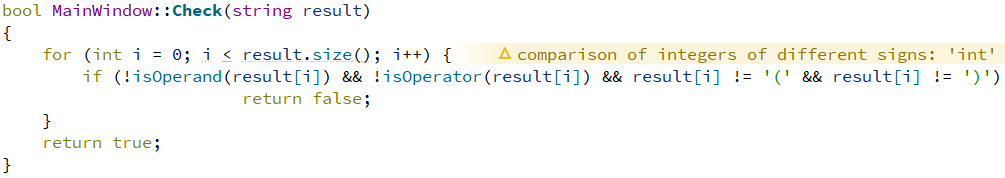


рисунок 7 – функция проверки поступившего выражения.

1.2. В функции Parsing (рисунок 8) производится запуск функции Check() в тернарном операторе; в случае, если выражение содержит неправильные символы, например, латинские, то функция возвращает строку с ошибкой, иначе возвращает преобразованное выражение с помощью функции toPostfix() (рисунок 10).

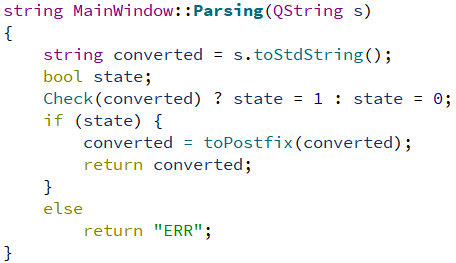


рисунок 8.



рисунок 9 – наглядное сравнение преобразованного выражения с изначальным.

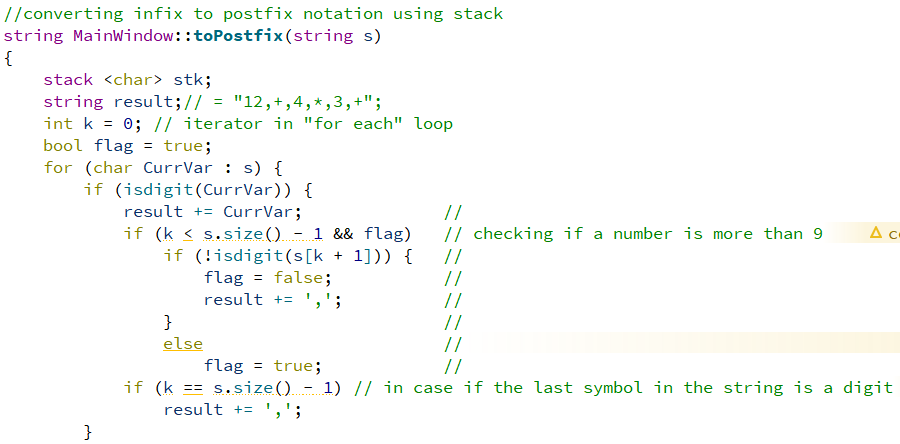
1.3. В функции toPostfix() производится перевод выражения в постфиксную нотацию. В цикле проверяется каждый символ:

1.3.1. 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.

1.3.2. Если символ – оператор (+, -, \*, /), то проверяется приоритетность в функции Priority() (рисунок 11). Если приоритетность оператора выше приоритетности оператора из стека, то жанный 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.

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

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





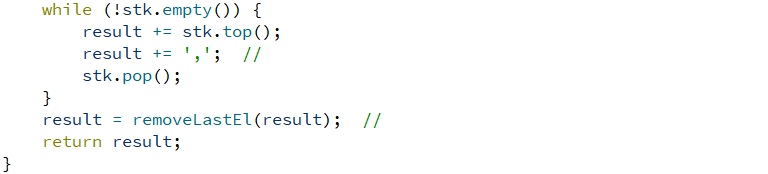


рисунок 10 – преобразование в обратную польскую запись.

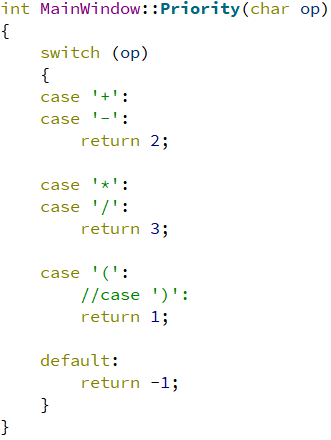


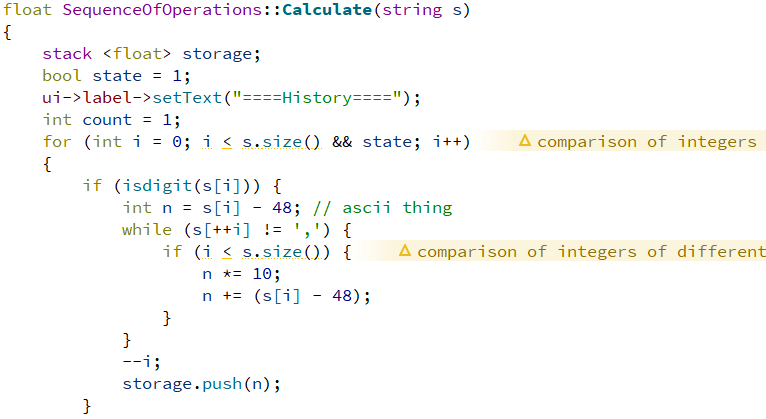
рисунок 11.

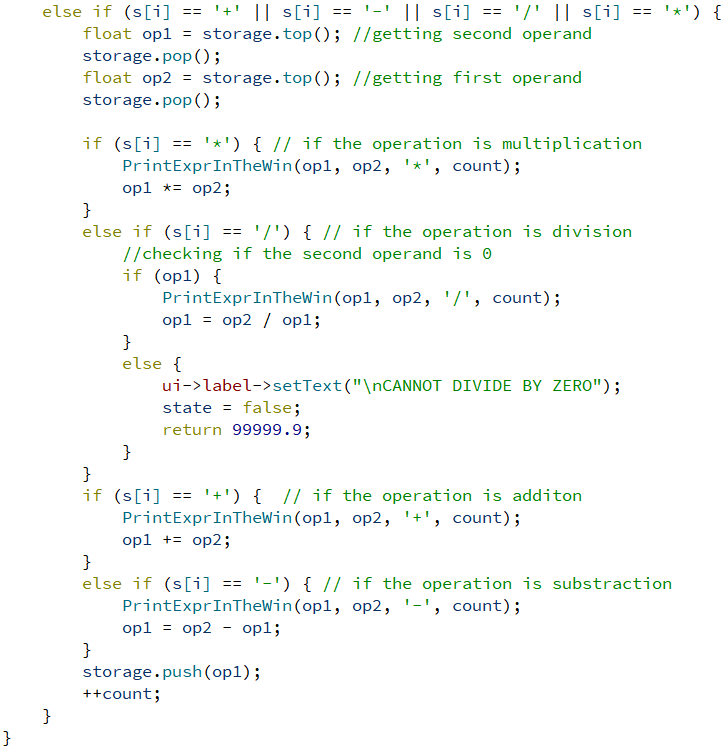
1.4. В функции Calculate() (рисунок 12) происходит вычисление по постфиксной нотации. Происходит проход по строке с проверкой каждого символа:

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

1.4.2. 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, (рисунок 13).

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





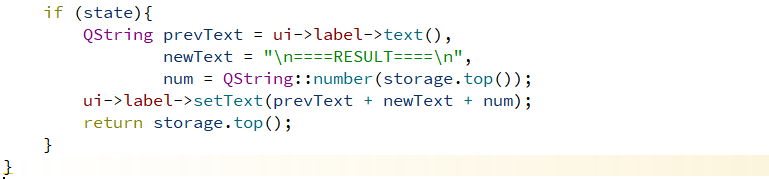


рисунок 12 – получение решения из обратной польской записи.

1.5. Печать на экран маленького выражения.

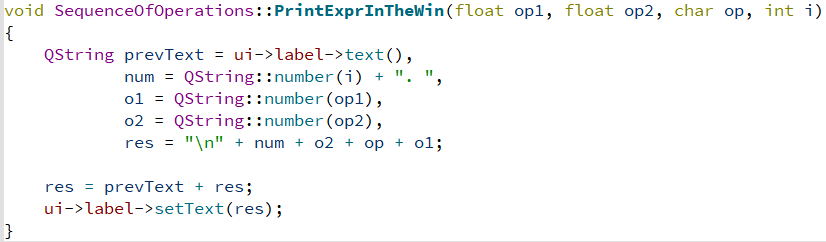


рисунок 13 – печать действия во второе окно.

# **3. Вывод**

Программа работает эффективно и стабильно, так как все предложенные задачи были обдуманы и решены.