Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 382003-1

Измайлов И.Р.

Нижний Новгород

2021г.

**Содержание**

[Постановка задачи 3](#_Toc91708562)

[Алгоритм выполнения 4](#_Toc91708563)

[Руководство пользователя 5](#_Toc91708564)

[Описание программной реализации 7](#_Toc91708565)

[Приложение 9](#_Toc91708566)

# Постановка задачи

Передо мной была поставлена задача – повторить весь материал по языку программирования С++, который мы успели пройти на момент окончания 3 семестра. На основе этих знаний написать программу, которая бы реализовывала вычисление арифметических выражений.

Арифметическое выражение – выражение, в котором операндами являются объекты, над которыми выполняются арифметические операции.

Например (1+10)\*(100-45+78)/45 – инфиксная запись

Существует так называемая обратная польская форма записи, не содержащая скобок. Операторы следуют после собственных операндов. Для данного примера постфиксная запись будет иметь следующий вид:

1 10 + 100 45 – 78 + \* 45 /

Допустимые операции: +, -, /, \*. Допускается наличие унарного минуса.

И для вычисления арифметических выражений нам необходимо преобразовать инфиксную запись в постфиксную.

Предполагается, что вычисления, преобразования будут производиться с использованием структуры данных Стек.

Стек – структура данных, представляющий из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека. Говорят, что стек устроен по принципу FILO(first in - last out, «первым пришел – последним вышел»).

# Алгоритм выполнения

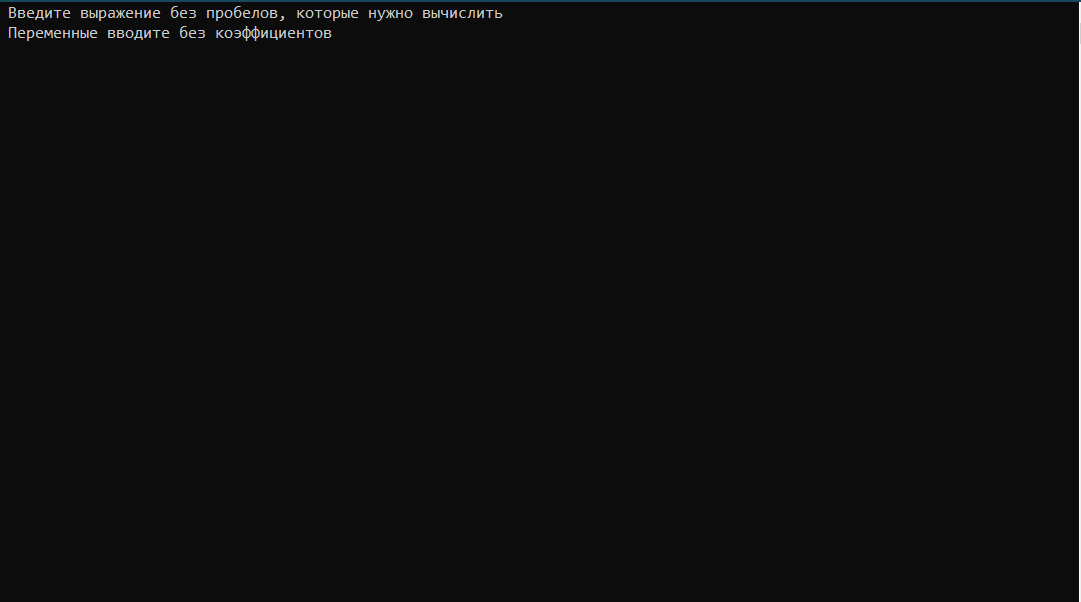
1. Разбиение исходного арифметического выражения на лексемы (т.е. выделить операнды, операции, переменные)
2. Проверка корректности выражения:

* правильность расстановки скобок (с использованием стека (TStack) для хранения индексов скобок).
* пропущены ли операнды или знаки операций
* недопустимые символы

1. Перевод выражения в постфиксную (польскую) запись
2. Вычисление выражения по постфиксной записи

# Руководство пользователя

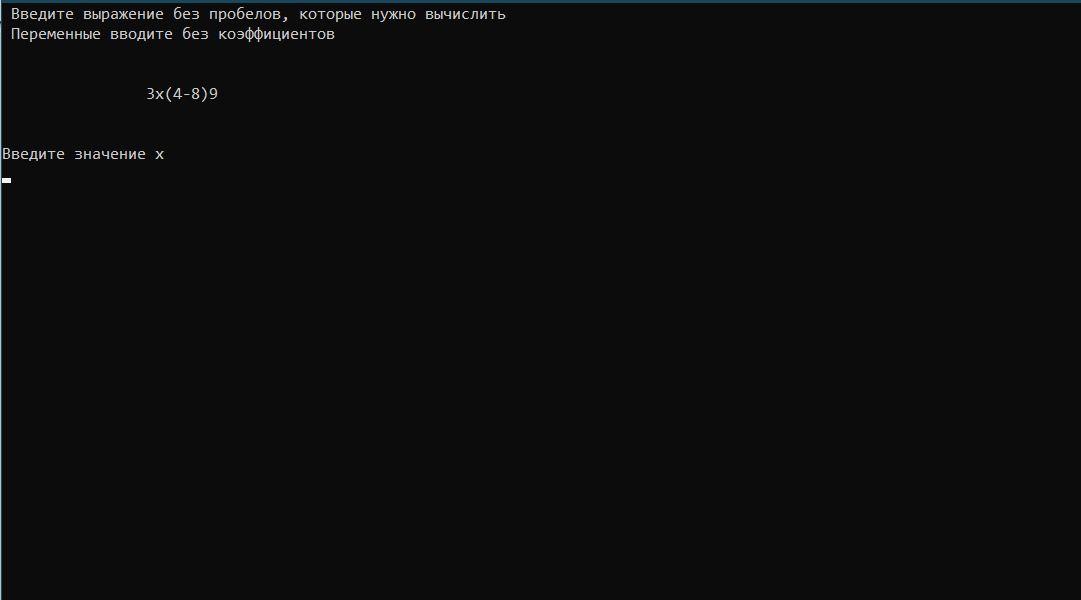
Пользователю на стартовой странице предоставляется возможность ввести выражение. Операторы и операнды записываются без пробелов. Неизвестные параметры принимаются без коэффициентов. То есть запись(x1, при разбиении на лексемы будет представлен в виде x\*1) .

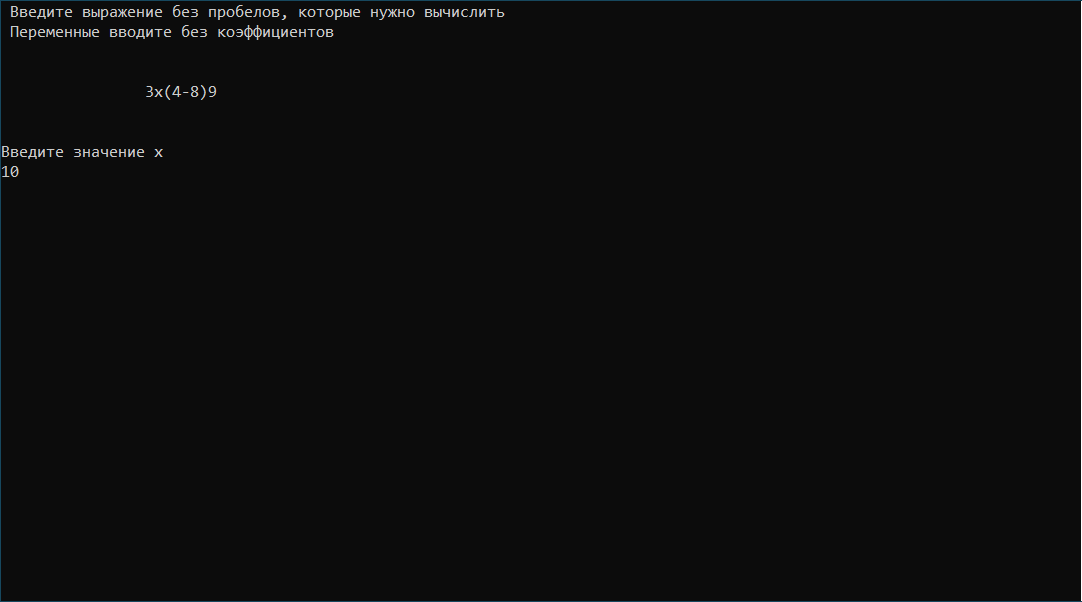


Допускается вариант не ставить знак «умножение» перед скобкой и после нее. Разрешимый вариант:

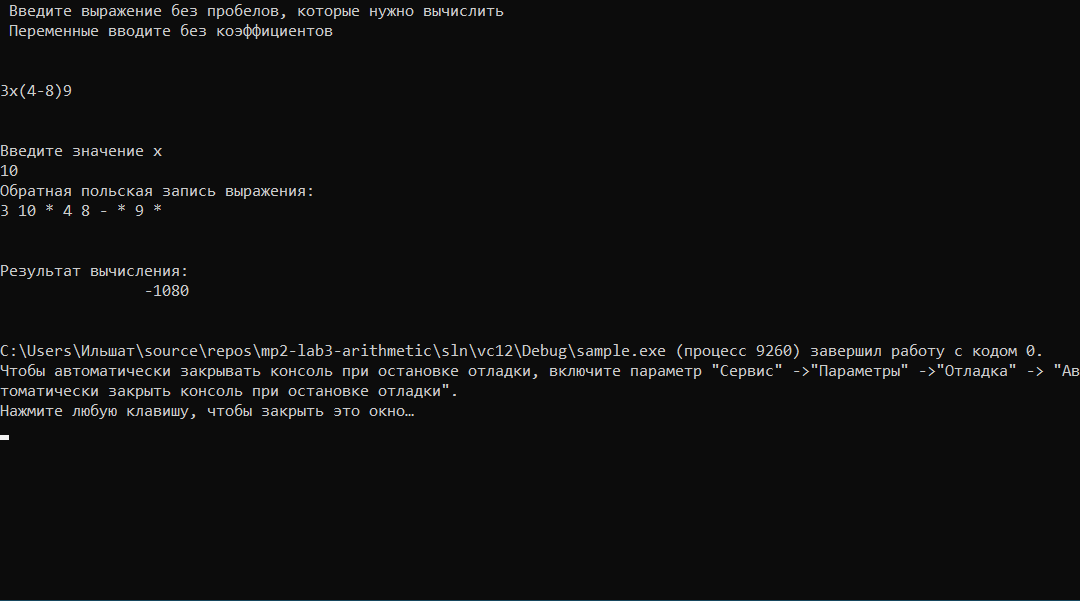
3x(4-8)9 будет автоматически представлен в виде 3\*x\*(4-8)\*9

Так как у нас содержится неизвестная величина x, то программа предложит ввести значение для x. Для наглядности зададим х значение 10.





После того, как пользователь присвоил неизвестной величине значение 10, на экране появится запись исходного выражения в обратной польской записи и результат вычисления.



# Описание программной реализации

Для хранения лексем используется класс “lexeme”, которая может хранить значение(какое то число) или операцию(+,-,/,\*, унарный минус). Если isV = true, то это означает, что в объекте класса lexeme хранится какое-то значение. Если isV= false, то, соответственно – операнд.

**Подтверждение корректности**

Проверка корректности алгоритма осуществляется с помощью gtests.

# Приложение

double arithmetic::calculate(lexeme\* arrt, int kol)

{

Stack<lexeme> s;

lexeme t;

double x1, x2;

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

{

if (arrt[i].isValue())

{

s.push(arrt[i]);

}

else

{

if (arrt[i].getOp() == "+")

{

x2 = s.pop().getValue();

x1 = s.pop().getValue();

t = x1 + x2;

}

if (arrt[i].getOp() == "-")

{

x2 = s.pop().getValue();

x1 = s.pop().getValue();

t = x1 - x2;

}

if (arrt[i].getOp() == "\*")

{

x2 = s.pop().getValue();

x1 = s.pop().getValue();

t = x1 \* x2;

}

if (arrt[i].getOp() == "/")

{

x2 = s.pop().getValue();

x1 = s.pop().getValue();

t = x1 / x2;

}

if (arrt[i].getOp() == "un-")

{

x2 = s.pop().getValue();

t = -1 \* x2;

}

s.push(t);

}

}

return s.pop().getValue();

}