Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

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

Институт информационных технологий, математики и механики

**Отчёт по лабораторной работе**

# **«Вычисление арифметических выражений»**

**Выполнил**:

Студент группы 3821Б1ПМ2

Борисов С.А.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

Содержание

[1. Постановка задачи 3](#_Toc529541654)

[2. Руководство пользователя 4](#_Toc529541655)

[3.1. Описание структуры программы 6](#_Toc529541656)

[3.2. Описание алгоритмов 8](#_Toc529541657)

[4. Результаты экспериментов 9](#_Toc529541658)

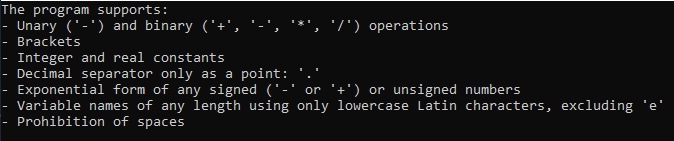
[Заключение 1](#_Toc529541659)0

# Постановка задачи

Цель данной работы - разработка на языке С++ структуры данных Стек и её использование для расчёта арифметических выражений с использованием обратной польской записи (постфиксной формы), описать алгоритм и программную реализацию, а также разработать и реализовать тесты для Стека и класса арифметических выражений на базе Google Test.

# Руководство пользователя

В начале программа выводит список поддерживаемых лексем в арифметическом выражении:



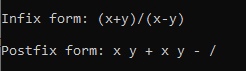
Изображение 1: вывод списка поддерживаемых лексем

Далее программа предлагает пользователю ввести арифметическое выражение в инфиксной форме:

IMG_256

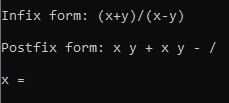
Изображение 2: ввод инфиксной формы

Если пользователь ввёл корректное выражение, то на экран выведется представление этого выражения в обратной польской записи



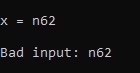
Изображение 3: вывод представления арифметического выражения в обратной польской записи

Далее, если выражение содержит переменные, то программа предложит пользователю ввести их значения



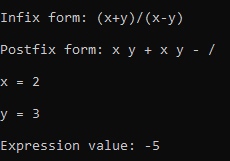
Изображение 4: ввод значения переменной

Если пользователь ввёл некорректные значения переменных, то на экран выведется фраза «Bad input: <введённое значение>»

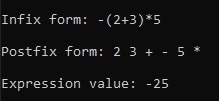


Изображение 5: сообщение о некорректном введённом значении переменной

Если же пользователь ввёл корректные переменные или если выражение не содержало переменных, то программа выведет значение данного выражения

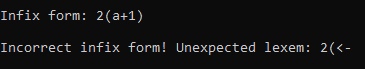


Изображение 6: вывод значения выражения с переменными



Изображение 7: вывод значения выражения без переменных

Если пользователь ввёл некорректное выражение, то программа выведет фразу «Incorrect infix form!» с указанием на ошибку



Изображение 8: сообщение об ошибке в введённой инфиксной форме с указанием ошибки

1. **Руководство программиста**

## Описание структуры программы

Программа содержит два основных класса:

Stack:

При создании экземпляра этого класса, принимающего в конструкторе размер стека, для него создаётся динамический массив T\* pMem. Также класс содержит поля int top (индекс вершины стека) и size\_t size (размер стека) и следующие методы:

• current\_size - возвращает текущее количество элементов в стеке

• clear - очищает стек

• isEmpty - возвращает true, если стек пустой и false в противном случае

• isFull - возвращает true, если стек заполнен и false в противном случае

• push - кладёт новый элемент на вершину стека (если стек к этому моменту уже заполнен, то выполняется перевыделение памяти)

• oversize - приватный метод, перевыделяющий память для стека

• pop - возвращает и удаляет элемент на вершине стека (если к этому моменту стек пуст, то выбрасывается исключение)

• show\_top - возвращает элемент на вершине стека без его удаления (если к этому моменту стек пуст, то выбрасывается исключение)

Также класс Stack содержит деструктор, очищающий поле pMem.

TArithmetic:

Данный класс содержит в себе внутренний абстрактный класс Lexem, а также его класс-наследник Operation и абстрактный класс-наследник Operand. От последнего наследуются ещё два класса: Constant и Variable.

Класс Lexem и его классы-наследники содержат методы Get (возвращает лексему), Value (возвращает значение лексемы, в случае операции - приоритет, константы - саму константу в типе данных double, переменной - значение переменной), what (возвращает информацию о том, чем именно является данная лексема). Также класс Lexem содержит чисто виртуальный метод SetValue (устанавливает значение лексемы).

Класс TArithmetic содержит следующие поля:

• string infix - хранит в себе принятое в конструкторе арифметическое выражение в инфиксной форме

• Lexem\*\* lexem - массив указателей, содержащий лексемы из инфиксной формы в соответствующем порядке

• Lexem\*\* postfix - массив указателей, содержащий лексемы из массива указателей lexem, но уже в порядке, соответствующем обратной польской записи и, соответственно, без скобок

• int lexems\_count - счётчик поступивших лексем из инфиксной формы

• int operations\_count - счётчик операций в арифметическом выражении

• int postfix\_count - счётчик лексем, поступивших в постфиксную форму

Класс TArithmetic содержит следующие методы:

1. Приватные:

• find\_operation - возвращает true, если лексема - операция и false в противном случае

• isDigit - возвращает true, если лексема - цифра и false в противном случае

• isAlpha - возвращает true, если лексема - символ латиницы в нижнем регистре и false в противном случае

• find\_error - возвращает в виде string информацию об ошибках в инфиксной форме

• parser - разбивает инфиксную форму на отдельные лексемы

• toPostfix - переводит инфиксную форму в постфиксную

• check\_input - вызывается после ввода пользователем значения выражения и возвращает true, если значение корректное и false в противном случае

1. Публичные:

• GetPostfix - возвращает в виде string постфиксную форму арифметического выражения

• Calculate - вычисляет выражение в постфиксной форме и возвращает результат в виде doube

Класс TArithmetic также содержит деструктор, в котором сначала удаляются элементы массива lexem, а потом сам массив lexem и массив postfix.

## Описание алгоритмов

В данной реализации используются два основных алгоритма:

1. Перевод из инфиксной формы записи арифметического выражения в постфиксную:

- Если текущая лексема - операция:

- если стек операций не пустой:

- если текущая лексема - закрывающая скобка:

- вынимаем из стека операций элементы и переводим их в постфиксную форму до тех пор, пока не встретим открывающую скобку

- вынимаем из стека операций открывающую скобку

- если текущая лексема - не открывающая скобка и не унарный минус:

- вынимаем из стека операций элементы и переводим их в постфиксную форму до тех пор, пока не встретим операцию с приоритетом ниже, чем у текущей операции

- если текущая операция не закрывающая скобка, кладём её в стек операций

- если текущая лексема - не операция, то переводим её в постфиксную форму

- оставшиеся в стеке операции вынимаем и переводим в постфиксную форму до тех пор, пока стек не станет пустым

1. Вычисление арифметического выражения в обратной польской записи:

- Если текущая лексема - константа или переменная:

- кладём её значение в стек

- если текущая лексема - операция:

- если текущая лексема - унарный минус:

- вынимаем из стека значение, применяем к нему операцию унарного минуса и кладём получившееся значение в стек

- если текущая операция - бинарная:

- вынимаем из стека два значения и применяем к ним текущую операцию, учитывая при этом порядок операндов в случае разности и деления и кладём получившееся значение в стек

- пройдя по всем операциям, в стеке останется один элемент - значение арифметического выражения

# Результаты экспериментов

Для проведения экспериментов были разработаны и реализованы тесты на базе Google Test. Классы Stack и TArithmetic успешно проходят все тесты.

# Заключение

В ходе данной лабораторной работы были реализованы на языке С++ структура данных Стек и класс арифметических выражений, использующий Стек, были описаны используемые алгоритмы и программная реализация, а также разработаны и реализованы тесты для Стека и класса арифметических выражений, которые, по результатам экспериментов, были успешно пройдены.