Федеральное государственное автономное образовательное учреждение  
высшего образования

«Нижегородский государственный университет

им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

Отчёт по лабораторной работе

**Вычисление арифметических выражений**

**Выполнил:**

студент ИТММ гр. 0826-1

Поляков М.О.

**Проверил:**

ассистент каф. МОСТ ф-та ИТММ

Кустиков В.Д.

Нижний Новгород

2015 г.

Оглавление

[Постановка задачи 3](#_Toc438494940)

[Руководство программиста 4](#_Toc438494941)

[Описание структуры программы 4](#_Toc438494942)

[Описание структур данных 5](#_Toc438494943)

[Описание алгоритмов 6](#_Toc438494944)

[Заключение 7](#_Toc438494945)

[Литература 8](#_Toc438494946)

[Приложения 9](#_Toc438494947)

# Постановка задачи

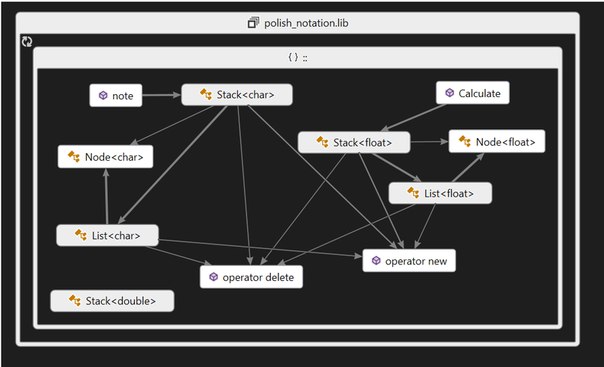
Разработать программу, выполняющую вычисление арифметического выражения с вещественными числами и преобразования данного выражения в общем виде в польскую форму записи. Выражение в качестве операндов может содержать переменные и вещественные числа. Допустимые операции известны: +, -, /, \*. Допускается наличие знака "-" в начале выражения или после открывающей скобки. Программа должна выполнять предварительную проверку корректности выражения и сообщать пользователю вид ошибки и номер позиции строки, в которых были найдены ошибки. Все функции должны тестироваться с помощью Google test-ов.

# Руководство программиста

## Описание структуры программы

Ядром проекта является библиотека polish\_notation, схему которой вы видите на Рисунке 1. Стрелка в схеме означает, что выбранный класс использует функционал класса, на который ссылается стрелка.

Рисунок 1. Схема зависимостей классов polish\_notation



## Описание структур данных

Входные данные хранятся в виде строки (string). В результате обработки Parser-ом мы получаем массив лексем.

Corrector хранит все найденные ошибки в стеке ошибок (Stack<Error>), который потом выводится в консоль.

* **calculate**

Объявления – calculate.h

Реализация – calculate.cpp

Содержит функцию note и функции Calculate, note

* **stack**

Объявление, реализация – stack.h

Представляет собой обыкновенный стек. Реализован при помощи шаблонов.

Приватные поля:

\* Реализованы два конструктора: пустой и с размером стека

Обладаем методами Put (извлекает и удаляет элемент с верхушки стека), Peek (показывает верхний элемент без удаления), Push (добавляет элемент в верхушку стека). Проверка на переполнение/пустоту осуществляется методами IsFull/IsEmpty.

* **list**

Объявление, реализация –list.h

Представляет собой обыкновенный стек. Реализован при помощи шаблонов.

Приватные поля:

\* Реализованы два конструктора: пустой и с ссылкой на список

Обладаем методами Search (поиск элемента в списке), erase (удаление элемента из списка), InsertFirst (добавляет элемент в начало списка), InsertLast (добавляет элемент в конец списка), InsertAfter (добавляет элемент после какого-либо элемента списка), InsertBefore (добавляет элемент перед каким-либо элементом списка). Возврат указателя на первый элемент getFirst.

## Описание алгоритмов

Рассмотрим пример: (b+a) \* c

Первая лексема – открывающая скобка, она отправляется в стэк. Далее идёт переменная б, она тоже отправляется в стэк, затем приходит лексема а так как операнды по приоритету польской формы стоят выше операций. И, наконец, приходит +. Далее, когда приходит закрывающая скобка, данные выгружаются из стэка в другой, буферный стэк, и приходит в пустой стэк операнд с затем операция \*. После они тоже выгружаются в буферный стэк и начинается вычисления с помощью функции Calculate, которая просит нас ввести значения переменных и вычисляет по польской форме.

# Заключение

В итоге мы имеет совокупность классов, которые позволяют вычислять некоторые арифметические выражения, контролировать корректность ввода и позволяют проводить вычисления над одним выражением, но с разными переменными. Решение такой простой задачи сильно упрощает жизнь людям в самых разных областях деятельности: от вычисления школьных выражений, до сложных инженерных расчётов. Каждый класс покрыт набором юнит тестов в изоляции. Кроме этого, написаны интеграционные тесты, проверяющие корректность работы несколько классов вместе. Получены навыки работы со стеком, синтаксического разбора выражения, анализа на корректность и вычисления выражения.

# Литература

1. <http://habrahabr.ru/post/100869/>
2. Лекции В.П.Гергель

# Приложения