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

**Отчет по лабораторной работе**

# «Вычисление арифметических выражений»

**Выполнил**:

студент группы 3822Б1ПМ1

Зазнобин Петр Викторович

Нижний Новгород

2023

Содержание

[Введение 3](#_Toc529541653)

[1. Постановка задачи 4](#_Toc529541654)

[2. Руководство пользователя 5](#_Toc529541655)

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

[3.2. Описание алгоритмов 6](#_Toc529541657)

[4. Результаты экспериментов 7](#_Toc529541658)

[Заключение 8](#_Toc529541659)

[Литература 9](#_Toc529541660)

[Приложение 10](#_Toc529541661)

# Введение

Иногда бывают ситуации, когда нужно посчитать значение выражения с параметрами и сравнить результаты при разном их значении. Именно для такой проблемы был создан данный калькулятор, который может высчитывать значения выражений с параметром или без. В первом случае, позволяет пересчитывать результат при разных введенных значениях.

# Постановка задачи

**Цель данной работы** — разработка структуры данных Стек и ее использование для расчета арифметических выражений с использованием обратной польской записи (постфиксной формы).

Для достижения цели, были поставлены и решены некоторые задачи:

1. Разработка интерфейса шаблонного класса TStack.
2. Реализация методов шаблонного класса TStack.
3. Разработка интерфейса класса TPostfix для работы с постфиксной формой.
4. Реализация методов класса TPostfix.
5. Разработка и реализация тестов для классов TStack и TPostfix на базе Google Test.
6. Публикация исходных кодов в личном репозитории на GitHub.

# Руководство пользователя

В начале работы, программа описывает некоторые базовые правила работы с ней и просит ввести выражение, которое нужно посчитать:

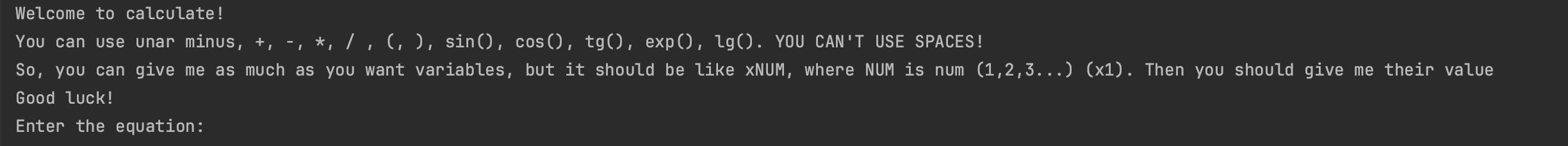


Рис. Начальный экран с правилами

Т.о. пользователь может использовать все базовые бинарные операции (за исключением подсчета остатка), унарный минус и некоторые функции (sin(), cos(), lg()**(==ln())**, exp()). В программе есть возможность вводить бесчисленное число переменных, однако важно, чтобы они были определенного формата, т.е. вида «xNum», как например х1, х2, х3 и тд. В программе запрещено использовать пробелы. На вход можно подавать целые числа и числа с плавающей запятой. Нельзя опускать операции (т.е. нельзя написать «2(3+2)», где подразумевается «2\*(3+2)»). Если программа заметит ошибку в веденном выражении, она напишет об этом в консоль и укажет индекс элемента введенной строки (считается с 0), в котором произошла ошибка. После ввода выражения, в случае наличия параметров, программа потребует пользователя ввести их значения. После ввода значений будет выведен ответ и будет предложено пересчитать значение выражения с другими значениями параметров. От выбора можно отказаться, введя «N»

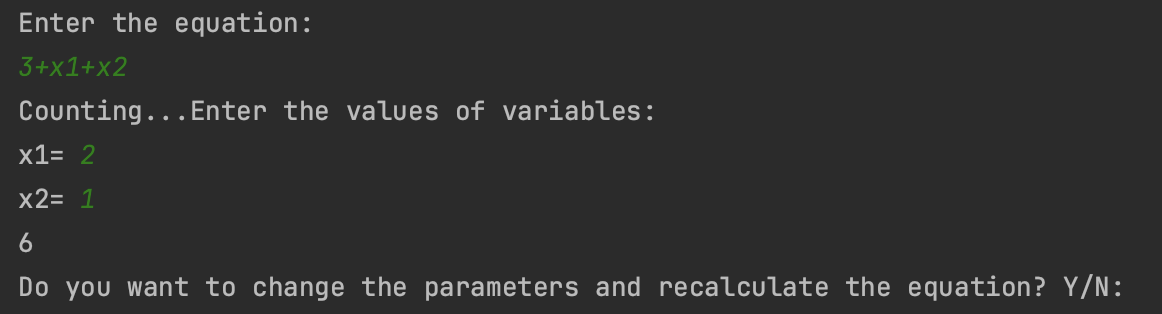


Рис. Пример работы с программой

1. **Руководство программиста**

## Описание структуры программы

Структура проекта:

* gtest — библиотека Google Test.
* samples — каталог с пользовательским приложением. Содержит реализацию основного приложения.
* test — каталог с проектом с модульными тестами. Содержит в себе файлы в gтестами для стека и основной программы.
* include – каталог с хедерами для класса DStack и Arithmetic.
* src – каталог с реализацией класса Arithmetic.
* sln - каталог с файлами с решениями (solution) для Microsoft Visual Studio 2010 и 2012.
* README.md — информация о проекте.
* Служебные файлы
  + .gitignore — перечень расширений файлов, игнорируемых Git при добавлении файлов в репозиторий.

## Описание алгоритмов

Для реализации стека была использована методика создания с стека. А именно были создан конструктор, который в самом начале выделяет доступную память для стека (capacity) методы «IsFull» и «isEmpty». Были добавлены методы «push» для добавления элемента в стек, с возможностью увеличения сopacity в случае isFull (метод «resize»), «pop» для извлечения из стека верхнего элемента, «current» для просмотра верхнего элемента стека «del» для удаления всех элементов стека без освобождения памяти, «size» для просмотра реального размера стека.

Для создания непосредственно калькулятора был использован алгоритм перевода инфиксного математического выражения в вид обратной польской записи (постфиксный вид).

Это было реализовано с помощью стека и вектора лексем. Первым делом, в методе «Parse», строка разбивается на значимые части (числа, операции, функции). Эти части заносятся в отдельную ячейку вектора. Затем, в методе «toPostfix», происходит обработка вектора (занесение каждой ячейки в вектор «postfix») по определенным правилам, с учетом приоритета операций. Числа же в формате string конвертируются в тип double (с помощью метода «StringToDouble») и также заносятся в итоговый вектор. Все это происходит так, чтобы в конце получилась запись исходного математического выражения в формате обратной польской записи.

Была реализована проверка на правильность записи исходного выражения пользователем.

Был создан метод «GetVars» для получения значений параметров и последующего занесения их в словарь (map), где в качестве ключа была строка названия параметра, а значением, непосредственно введенное пользователем число.

Был реализован метод «Calculate» для подсчета математического выражения из формы обратной польской записи.

# Результаты экспериментов

# 

# 

# 

# 

# 

# 

# Заключение

Было реализовано приложение, которое может корректно вычислять математические выражения и учитывать значения параметров, введенные пользователем, если такие имеются.

# Литература

* + - 1. ReadMe (из кейса GitHub)
      2. Лекции Александра Владимировича Сысоева