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 3821Б1ПМ2

Сятов Н. А.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2021

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

В данной лабораторной работе передо мной стояла задача написать программу на языке С++, которая будет принимать на ввод строку виде арифметического выражения, которое преобразовывалось в обратную польскую запись, вместо переменных подставлялись значения, заданные пользователем, и находился результат в типе double. Для реализации требуется написать класс стек, класс Postfix, в котором одно из полей – массив указателей абстрактного класса Lexem, от которого наследуются классы операций, операндов и переменных. Также проверить корректность работы с помощью тестирования на основе google test.

# Метод решения

Для решения данной задачи я реализовал класс Stack. Также, чтобы эффективно реализовать функцию для нахождения решения, я сделал абстрактный класс Lexem, чтобы использовать виртуальные функции. От абстрактного класса я отнаследовал 3 других: Operation, Operand, Var.

Основной класс – TPostfix. В нём есть 2 массива указателей Lexems\*: один хранит инфиксную форму, другой - постфиксную. В нём происходит преобразование в обратную польскую запись и нахождение результата с подстановкой значений переменных.

***Обратная польская запись*** – метод записи арифметических выражений, в которой не нужно учитывать приоритеты операций (поэтому в ней нет скобок). В такой записи намного проще обрабатывать данные.

В такой записи операция идёт после операндов и операции выполняются последовательно, после нахождения результата одной операции, результат вставляется вместо операндов и этой операции (5 4 \* 9 - → 20 9 -).

Чтобы преобразовать инфиксную форму в постфиксную используется Stack.

ПРИМЕР

Инфиксная форма: 5\*4-(6+7)

Перевод в постфиксную с помощью стэка:

|  |  |  |
| --- | --- | --- |
|  | Постфиксная форма (заполняется постепенно) | Стэк |
| 1 | 5 | |  | | --- | |  | |  | |  | |  | |  | |
| 2 | 5 | |  | | --- | |  | |  | |  | |  | | \* | |
| 3 | 54 | |  | | --- | |  | |  | |  | |  | | \* | |
| 4 | 54\* | |  | | --- | |  | |  | |  | |  | | - | |
| 5 | 54\* | |  | | --- | |  | |  | |  | | ( | | - | |
| 6 | 54\*6 | |  | | --- | |  | |  | |  | | ( | | - | |
| 7 | 54\*6 | |  | | --- | |  | |  | | + | | ( | | - | |
| 8 | 54\*67 | |  | | --- | |  | |  | | + | | ( | | - | |
| 9 | 54\*67+ | |  | | --- | |  | |  | |  | |  | | - | |
| 10 | 54\*67+- | |  | | --- | |  | |  | |  | |  | |  | |

5\*4-(6+7) → 54\*67+-

# Руководство пользователя

Пользователю предлагается ввести арифметические выражение соблюдая несколько правил: нельзя использовать пробелы, переменные – одна буква (верхнего или нижнего регистра), константы можно вводить в виде десятичных дробей или в экспоненциальной форме записи. Если в выражении присутствует ошибка(лишняя скобка, неправильная запись чисел и т. д.), то программа выводит сообщение об ошибке и примерно указывает на ту лексему, в которой программа видит ошибку. Если ввод правильный, то выводится инфиксная форма, постфиксная и результат.

# Описание программной реализации

Папки проекта:

gtest – библиотека Google test

samples – каталог с пользовательским приложением

test – каталог с тестами

include – каталог с заголовочными файлами:

arithmetic.h – содержит прототипы классов (Lexems, Operation, Operand, Var, TPostfix) и реализацию класса исключений lexException и прототипы глобальных функций.

stack.h – содержит реализацию класса Stack.

src – содержит файл arithmetic.cpp с реализацией классов из заголовочного файла arithmetic.h

sln – каталог с файлами и решениями (solution) для Microsoft Visual Studio 2010.

Функции:

1. Виртуальные функции:
   1. virtual std::string whatis() – возвращает название класса(Operation, Operand, Var).
   2. virtual size\_t prioritet() – возвращает приоритет операции.
   3. virtual void ToDo(Stack<double>& S) – в классе Operand кладёт операнд в стэк, конвертирую из типа string в double; в классе Operation берёт из стэка операнды, выполняет операцию и кладёт результат в стэк; в классе Var если переменная не была инициализирована ранее, то просит пользователя задать значение переменной и сохраняет в глобальной map, далее кладёт в стэк значение данной переменной.
   4. virtual std::string show() – возвращает значение лексемы.
2. Методы класса TPostfix:
   1. void init\_infix() – инициализирует поле Lexems\* infix\_form.
   2. void init\_postfix() – переводит инфиксную форму в постфиксную и инициализирует Lexems\* postfix\_form.
   3. void correctChecker(const size\_t& i, const size\_t index) – проверяет на корректность введённых данных.
   4. double resolve() – находит значение выражения.
   5. std::string get\_infixLexem() – возвращает инфиксную форму в виде строки.
   6. std::string get\_postfixLexem() – возвращает постфиксную форму в виде строки.
3. Глобальные функции:
   1. bool isOperand(const char& lexema) noexcept – проверяет, операнд это или нет.
   2. bool isOperator(const char& lexema) noexcept – проверяет, оператор это или нет.
   3. bool isVar(const char& lexema) noexcept – проверяет, оператор это или нет.
   4. double convert(std::string lexema) – конвертирует сроку в число.

# Подтверждение корректности

Для подтверждения корректности в программе использовались 54 теста на основе google test. Из них: 17 тестов на проверку корректности стэка, 11 тестов на проверку классов Operation, Operand, Var, 14 тестов на проверку класса TPostfix, 12 тестов на проверку глобальных функций.

# 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.