МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Перевод выражения в постфиксную форму»**

**Выполнил:** студент группы 381706-1

Нечаева Екатерина Владимировна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc560926)

[2. Постановка задачи 4](#_Toc560927)

[3. Руководство пользователя 5](#_Toc560928)

[4. Руководство программиста 6](#_Toc560929)

[4.1 Описание структуры программы 6](#_Toc560930)

[4.2 Описание структур данных 6](#_Toc560931)

[4.3 Описание алгоритмов 7](#_Toc560932)

[5. Заключение 8](#_Toc560933)

[6. Литература 9](#_Toc560934)

# Введение

Постфиксная запись представляет собой такую запись арифметического выражения, в которой сначала записываются операнды, а затем – знак операции. Например, для выражения a + b \* c постфиксная запись будет a b c \* +. Здесь операндами операции \* будут b и c (два ближайших операнда), а операндами операции + будут а и составной операнд b c \*. Эта запись удобна тем, что она не требует скобок. Например, для выражения (a + b) \* c постфиксная запись будет a b + c \*. В этой записи не требуется ставить скобки для того, чтобы изменить порядок вычисления, зависящий от приоритета операций, как в исходном выражении.

Обратная польская запись имеет ряд преимуществ перед инфиксной записью при выражении алгебраических формул. Во-первых, любая формула может быть выражена без скобок. Во-вторых, она удобна для вычисления формул в машинах со стеками. В-третьих, инфиксные операторы имеют приоритеты, которые произвольны и нежелательны.

Алгоритм перевода в постфиксную запись обрабатывает исходную строку и строит новую из тех же символов. Кроме того, необходим еще стек – аналогичный массив, используемый для временного хранения операций.

В данной лаборторной работе требуется разработать программу, которая для вычисления выражения переводит его в постфиксную форму и считает.

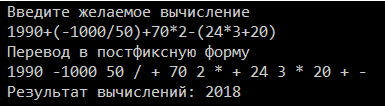
# Постановка задачи

Для реализации данной работы требуется реализовать методы:

* Конструктор-инициализатор
* Перевод в постфиксную форму
* Калькулятор, для подсчета полученного выражения
* Метод, находящий операции в выражении
* Метод, который расставляет приоритеты между операциями
* Метод, который узнает приоритет операции в стеке

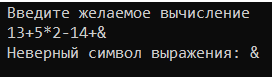
# Руководство пользователя

Программа начинает свою работы с ввода выражения, которое пользователь хочет сосчитать. После чего, она делает преобразование над выражением, а именно переводит его в постфиксную форму и выводит на экран результат вычислений.

**

*Рисунок 1. Пример использования программы.*

Если же пользователь введет неверный символ, программа выведет сообщение об ошибке.

**

*Рисунок 2. Пример вывода исключения.*

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль postfix. Содержит пример использования программы(*mainpostfix.cpp*)*.*
* Модуль libpostfix – статическая библиотека. Содержит заголовочный файл *postfix.h*, в котором описаны методы для класса TPostfixБ реализованные в исполняемом файле *postfix.cpp*.
* Модуль postfixTest. Содержит тесты, проверяющие правильность работы методов класса, описанных в файле *postfixTest.cpp*.
* Модуль libthrows – библиотека для исключений. Содержит заголовочный файл *throws.h*, в котором он и реализован.

## Описание структур данных

#### Класс TException – класс исключений.

Содержит одно защищенное поле *string str,* переменная строкового типа, которая хранит сообщение об исключении.

Содержит публичные методы

* TException(std::string \_str); //конструктор-инициализатор
* void Print(); //вывод данного исключения на консоль

#### Класс TPostfix.

Защищенные поля:

* char\* inputistr; //строка ввода
* char\* outputistr; //строка вывода

Содержит публичные методы:

* TPostfix(char\* \_inputistr); //конструктор-инициализатор
* char\* Postfix(); //перевод в постфиксную форму
* double calculate(); //считает полученною строку

Вспомогательные методы:

* bool isoperation(const char a); //проверка символа на оператор
* int Priority(const char a); //приоритет операторов
* int PriorityStack(TStack<char> &a); //приоритет операторов в стеке

## Описание алгоритмов

**Алгоритм преобразования инфиксной формы записи в постфиксную.**

1. Создать пустой стек, для хранения операторов.
2. Сканировать введенную пользователем строку слева направо:

* Если символ является операндом, то добавить его в строку вывода, увеличив при этом индекс на 1.
* Если символ является левой скобкой, положить его в стек.
* Если символ является правой скобкой, то выталкивать операторы из стека пока не будет найдена соответствующая левая скобка. Каждый оператор добавлять в строку, увеличивая каждый раз индекс на 1.
* Если символ является оператором \*, /, + или -, поместить его в стек. Однако, перед этим вытолкнуть любой из операторов, уже находящихся в стеке, если он имеет больший или равный приоритет, и добавить его в результирующую строку.

1. Когда входная строка будет полностью обработана, проверить стек. Любые операторы, всё ещё находящиеся в нём, следует вытолкнуть и добавить в конец итоговой строки.
2. Как только стек оказывается пустым, строке вывода по отслеживающему индексу в конце ставится символ ‘\0’, что означает конец строки вывода.

**Постфиксные вычисления.**

1. Создаём пустой стек для чисел.
2. Сканируем строку слева направо.

* Если символ является операндом, то преобразовываем его из строки в целое число и помещаем значение в стек.
* Если символ является оператором \*, /, + или -, то он нуждается в двух операндах. Производим выталкивание из стека дважды. Сначала вытолкнется второй операнд, а затем - первый. Выполняем арифметическую операцию и помещаем результат обратно в стек.

1. Когда входное выражение полностью обработано, его результат находится в стеке. Выталкиваем его и возвращаем в качестве ответа.

# Заключение

В ходе выполнения данной лабораторной была реализована программа, с помощью которой любое выражение можно перевести в постфиксную форму записи и сосчитать полученное выражение. Также, узнали, какие преимущества носит обратная польская запись, разобравшись на примере. Рассмотрели алгоритмы, позволяющие провести такую операцию и выполнить подсчет выражения.

Также были разработаны тесты, проверяющие корректную работу методов данных классов, с помощью Google C++ Testing Framework.

# Литература

1. Постфиксная запись [Электронный ресурс]

<http://natalia.appmat.ru/c%26c%2B%2B/postfisso.html>

1. Алгоритм преобразования в постфиксную форму [Электронный ресурс]

<https://aliev.me/runestone/BasicDS/InfixPrefixandPostfixExpressions.html>

1. Обратная польская запись [Электронный ресурс]

<https://habr.com/ru/post/100869/>