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**

**«Преобразование арифметических выражений в обратную польскую запись»**

**Выполнил:** студент группы 381706-1

Денисов Владислав Львович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc705622)

[2. Постановка задачи 4](#_Toc705623)

[3. Руководство пользователя 5](#_Toc705624)

[4. Руководство программиста 6](#_Toc705625)

[4.1 Описание структуры программы 6](#_Toc705626)

[4.2 Описание структур данных 6](#_Toc705627)

[4.3 Описание алгоритмов 7](#_Toc705628)

[5. Заключение 10](#_Toc705629)

[6. Литература 11](#_Toc705630)

# Введение

Лабораторная работа направлена на разработку системы для преобразования арифметических выражений в обратную польскую запись и вычисления их значений с ее помощью.

Обратная польская запись (англ. Reverse Polish notation, RPN) – форма записи математических и логических выражений, в которой операнды расположены перед знаками операций.

Пример:

* Исходное выражение:
* Выражение в обратной польской записи:

# Постановка задачи

В рамках лабораторной работы ставится задача эффективной реализации системы нахождения значений арифметических выражений путем их преобразования в обратную польскую запись.

Для работы будут разработаны:

* класс строки, для получения исходного выражения с консоли,
* функция преобразования введенного пользователем арифметического выражения в обратную польскую запись,
* функция вычисления значения арифметического выражения по его обратной польской записи,

Программное решение будет выглядеть следующим образом:

1. Набор перечисленных выше функций для работы с арифметическим выражением.
2. Вспомогательный класс строки – TString.
3. Класс для обработки исключений – TException, которые могут возникнуть при выполнении различных операций.
4. Программа, демонстрирующая работу разработанной библиотеки.
5. Набор автоматических тестов с использованием Google C++ Testing Framework.

# Руководство пользователя

В качестве примера использования разработанной библиотеки предлагается следующее.

Пользователю предлагается ввести арифметическое выражение. Программа подтверждает то, что выражение введено верно путем его отображения на консоли.

Производится преобразования выражения в обратную польскую запись. Результат преобразования выводится на консоль.

Выполняется вычисление значения введенного арифметического выражения по построенной обратной польской записи. Результат вычислений отображается пользователю (в случае вещественного числа, производится округление до 3 знака после запятой).

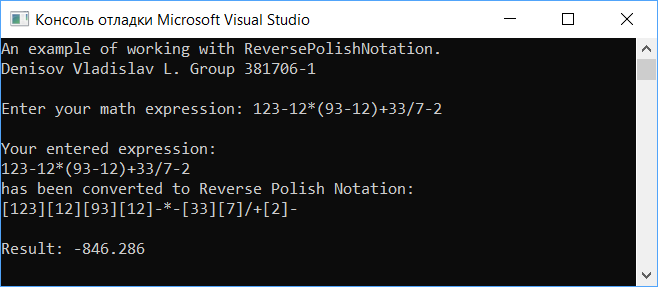


Рисунок 1 Пример работы демонстрационной программы.

# Руководство программиста

## 4.1 Описание структуры программы

Программа состоит из следующих модулей:

* Модуль Polish. Содержит пример использования полиномов. Реализация в файле *main\_Polish.cpp.*
* Модуль PolishLib – статическая библиотека. Содержит файлы *Polish.h* и *Polish.cpp*, в которых описаны объявления и реализация всех необходимых функций для перевода арифметического выражения в обратную польскую запись, а также нахождения его значения с помощью полученной записи.

Кроме того, модуль содержит файлы *MyString.h* и *MyString.cpp*, содержащие интерфейс и реализацию класса строки TString, разработанной с целью удобства при построении обратной польской записи.

* Модуль PolishTest. Содержит суммарно 28 тестов, описанных в файле *PolishTest.cpp* и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.

## Описание структур данных

#### Класс TException – класс исключений.

Класс содержит 1 **private** поле *std::string msg* – переменная, хранящая сообщение об ошибке в виде строки.

И содержит 2 **public** элемента:

*TException(std::string \_str) : msg(\_msg)* – конструктор с одним параметром.

*void Print()* – метод отображения ошибки на консоль.

#### Класс TString – вспомогательный класс строки.

Рассмотрим класс *TString* подробно.

**Элементы класса, объявленные со спецификатором protected:**

char\* str – указатель на область в памяти, где хранится строка.

int length – длина строки.

**Элементы класса, объявленные со спецификатором public:**

TString() – конструктор по умолчанию.

TString(TString &string) – конструктор копирования.

TString(char\* \_str) – конструктор преобразования типа.

~TString() – деструктор.

int GetLength() – метод, позволяющий получить длину строки.

void DeleteStr() – метод, позволяющий удалить строку.

TString operator+(TString &string) – оператор сложения двух строк.

TString& operator=(TString &string) – оператор присваивания.

char& operator[](int i) – оператор 0-based индексации.

**Дружественные функции:**

friend std::istream& operator>>(std::istream &is, TString &string) – перегруженный оператор ввода строки с помощью консоли.

friend std::ostream& operator<<(std::ostream &os, const TString &string) – перегруженный оператор вывода строки на консоль.

#### Блок для работы с арифметическим выражением.

Рассмотрим его более подробно. Необходимые для работы функции описаны в файле *Polish.h.* Таковыми являются:

int GetPriority(const char operation) – метод, возвращающий приоритет текущей операции. Если операции не существует, то бросается исключение.

bool IsOperation(char symbol) – метод, позволяющий узнать, является ли текущий символ операцией или нет.

TQueue<char> ConvertToPol(TString str) – метод, выполняющий построение обратной польской записи для переданного в качестве параметра выражения.

double Calculate(TQueue<char> queue) – метод вычисления результата арифметического выражения по его обратной польской записи.

## Описание алгоритмов

**Перевод выражения в обратную польскую запись.**

Введем приоритет для допустимых в вычислениях операций:

* ‘\*’ и ‘/’ – приоритет 3,
* ‘+’ и ‘—’ – приоритет 2,
* ‘(’ и ‘)’ – приоритет 1.

Для хранения операций будем использовать стек *stack*, результат перевода выражения в обратную польскую запись будем записывать в очередь *queue*. Исходное выражение хранится в строке *str*.

**Алгоритм:**

* + - 1. Начинаем просмотр введенного арифметического выражения слева направо. Пункты 2-4 повторяются до тех пор, пока вся строка не будет обработана или не будет вызвано исключение.

1. Если текущий символ *str* – цифра, то добавляем ее в очередь *queue*. Причем заметим, что если это первая цифра числа, то в очередь перед ней добавим ‘[’. После последней цифры ставим ‘]’. Квадратные скобки позволяет нам разграничить различные числа при вычислении значения выражения.
2. Если текущий символ строки *str* – операция или левая скобка, то помещаем его в *stack*.

Однако обратим внимание на то, что если текущий символ является операцией, и она имеет больший приоритет, чем операция на вершине стека *stack*, то новая операция добавляется в стек.

Иначе выполняем извлечение элементов из стека *stack* и переносим их в очередь *queue*. Извлечение элементов продолжаем либо до опустошения стека*,* либо до того, как приоритет новой операции не станет больше, чем у операции на вершине стека. Только после этого новая операция добавляется в стек.

1. Если текущий символ строки *str* – правая скобка, то извлекаем элементы из стека *stack* и добавляем их в очередь *queue*, пока стек не станет пустым или не встретится левая скобка. Левая скобка из стека удаляется.
2. Если вся строка *str* обработана и в стеке *stack* остались элементы, то выполняем их извлечение и добавляем в очередь *queue*. На этом алгоритм завершается.

**Вычисление значения выражения с помощью обратной польской записи.**

Создадим стек, в который будем помещать операнды-числа, путем их построения с помощью введенных ограничителей в виде квадратных скобок.

**Алгоритм:**

1. Выполним проверку на то, что первый элемент в очереди не является какой-либо операцией. Если все успешно, то выполняем пункты 2 и 3, пока очередь не будет полностью обработана. Если проверка провалилась, то бросаем исключение.
2. Если очередной элемент очереди является ‘[’, то необходимо выполнить построение числа. Берем элементы из очереди, пока не встретим ‘]’. Число строим посредством добавления к нему очередного разряда на каждой итерации цикла. Полученное число помещаем в стек.
3. Если взятый из очереди элемент оказался какой-либо операцией. Выполняем извлечение из стека двух чисел. Первое – правый операнд, второе – левый. Выполняем действие, которое требует обрабатываемая операция. Результат помещаем в стек.
4. Если после завершения обработки очереди в стеке остались элементы, то обратная польская запись была построена неверна – бросаем исключение.

# Заключение

В результате лабораторной работы была разработана библиотека, реализующая небольшой инструмент, позволяющий построить обратную польскую запись для какого-либо допустимого арифметического выражения и затем с помощью нее вычислить его значение.

Были разработаны и доведены до успешного выполнения тесты, разработанные для данного программного проекта с использованием Google C++ Testing Framework.

Программное решение было продемонстрировано с помощью нахождения значения для введенного пользователем арифметического выражения. Описание примера работы с программой было представлено в разделе «Руководство пользователя».

# Литература

1. Википедия: свободная электронная энциклопедия: на русском языке [Электронный ресурс] // URL: https://ru.wikipedia.org/wiki/Обратная\_польская\_запись