**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра Информационные Системы**

# ОТЧЕТ

**по лабораторной работе №1**
**по дисциплине «Алгоритмы и структуры данных»**
**Тема: Списочные структуры**

Студент гр. 1373 _____ Котов И.И.

Преподаватель _____ Бондаренко Б.Е.

Санкт-Петербург
2022

## Цель работы.

Реализовать следующие структуры: односвязный список, динамический массив и стек. Стек можно реализовать как на базе списка, так и отдельно. Использовать стек для реализации алгоритма сортировочной станции. Разрешённые символы в исходном выражении: +, -, *, /, ^, sin, cos, (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Для упрощения разбиения входной строки на токены разрешается отделять каждый символ пробелом.

# Ход работы

Всю работу выполнял на языке программирования Python.

## Реализация структур данных:

### 1. Односвязный список

Для реализации структуры односвязного списка был создан класс Node с полями data – данные элемента списка, и next - ссылка на следующий элемент. Также класс содержит методы для удобной реализации списка.

Сам класс односвязного списка LinkedList содержит следующие методы:

1) Вставка элемента в конец списка;
2) Вставка элемента по индексу;
3) Удаление элемента по индексу;
4) Получение элемента по индексу;
5) Получение размера списка;
6) Вывод всех значений элементов списка.

### 2. Динамический массив

Для реализации динамического массива был создан класс DynamicArray с полями: размер массива (size), текущее количество элементов в массиве (countElements), сам массив (array).

Реализованные методы класса:

1) Создание массива;
2) Изменение размера массива;
3) Добавление элемента в конец массива;
4) Вставка элемента на индекс;
5) Удаление элемента по индексу;
6) Возвращение элемента по индексу;
7) Получение размера массива.

В реализации данной структуры ключевым моментом было следующее: если при добавление элемента в массив не хватало исходного заданного размера, то надо изменить размер массива, то есть создать новый размером больше, перенести все элементы со старого в него, продолжить добавление.

### 3. Стек

Для реализации стека был создан класс Stack. Данный класс был реализован с помощью односвязного списка. Список реализованных функций для данного класса получился следующим:

1) Добавление элемента на вершину стека;
2) Удаление элемента с вершины стека;
3) Получение элемента с вершины стека;
4) Получение размера стека;
5) Вывод всех элементов стека;
6) Реверс элементов.

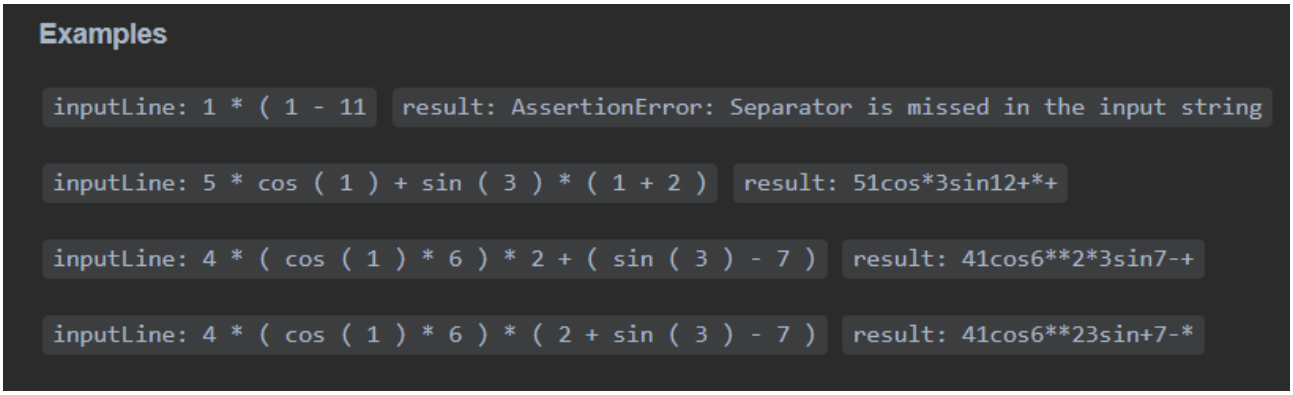
## Алгоритм сортировочной станции

За счет использования пробелом перед каждым из токенов легко отделять элементы друг от друга.

Сначала идет проверка на количество скобок, далее проверка на валидность токена. Если токен – число, то он сразу записывается в выходную строку. Иначе – проверяется стек на наличие других токенов:

Например, если в стеке находится оператор «+», а текущий токен «*», то «+» будет снят со стека и записан в выражение. Так будет до тех пор, пока приоритет текущего токена выше тех, что лежат на стеке.

Примеры ввода и вывода программы:




## Выводы

В результате работы были реализованы следующие структуры данных:

1) Односвязный список;
2) Динамический массив;
3) Стек.

На основе реализованных структур, был написан алгоритм сортировочной станции, переводящий выражение из префиксной нотации в постфиксную.