

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

Студентка гр. 1373

Таканаев Т.Ш.

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

1. Односвязный список

Для реализации структуры односвязного списка был создан класс `LinkedList`.
В экземпляре класса имеем:

1) Структура узла.

Реализованные функции класса:

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

2. Динамический массив

Для реализации динамического массива был создан класс `ArrayList`. В экземпляре класса имеем:

- 1) Ссылку на обобщенный массив данных;
- 2) Количество элементов в массиве;
- 3) Переменную, хранящую размер массива в данный момент времени;

Для расширения размера массива применяется функция `resize()`. Она создает новый массив нужной нам размерности.

Реализованные функции класса:

- 1) Конструктор;
- 2) Добавление элемента в массив по индексу;
- 3) Добавление элемента массива без индекса;
- 4) Получение элемента по индексу;
- 5) Удаление элемента по индексу;
- 6) Вывод всех элементов массива;
- 7) Получение размера массива.

3. Стек

Для реализации стека был создан класс `Stack`. В экземпляре класса имеем:

- 1) Ссылку на обобщенный массив данных;
- 2) Переменную, хранящую индекс элемента, находящегося на вершине стека;
- 3) Переменную, хранящую вместимость стека;

Реализованные функции класса:

- 1) Запись элемента на вершину стека;
- 2) Получение элемента с вершины стека с удалением;
- 3) Получение элемента с вершины стека без удаления;
- 4) Получение размера стека;
- 5) Проверка стека на пустоту;
- 6) Проверка стека на заполненность;

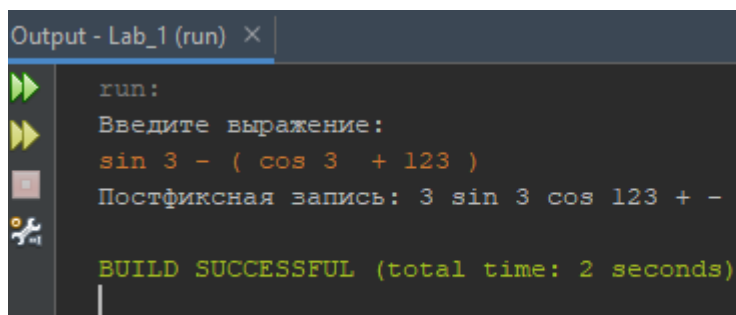
4. Алгоритм сортировочной станции

Опишем алгоритм перевода выражения из инфиксного вида в постфиксный на примере операндов и операторов.

Далее, прежде всего идет проверка. Если токен невалидный, то выводится ошибка и программа прекращает работу. 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.

На рисунке 1 представлен набор выражений поданных в алгоритм сортировочной станции и полученные результаты.



```
Output - Lab_1 (run) ×
run:
Введите выражение:
sin 3 - ( cos 3 + 123 )
Постфиксная запись: 3 sin 3 cos 123 + -

BUILD SUCCESSFUL (total time: 2 seconds)
```

Рисунок 1 – демонстрация работы реализованного алгоритма

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.