

**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

\_\_\_\_\_

Злобин С.В.

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) Динамический массив (вектор)**

Для создания динамического массива создаём класс MyVector

Стандартный конструктор MyVector()

Перегруженный MyVector(int newsize)

Деструктор ~MyVector()

Методы класса:

- 1) Обращение по индексу int& operator[](int num)
- 2) Индекс по значению int find(int val)
- 3) Добавление по индексу void add(int num, int val)
- 4) Удаление по индексу void delete\_a(int num)
- 5) Вставка в конец void push\_back(int val)
- 6) Получение ссылки на элемент int\* get\_reference(int num)
- 7) Получение размера массива int get\_size()
- 8) Вывод данных массива на экран void print\_data()

### **2) Односвязный список**

Для создания односвязного списка создаём класс MyList

В классе создаём ещё один класс MyNode

Стандартный конструктор MyList()

Перегруженный MyList(int new\_size)

Деструктор ~MyList()

Методы класса:

- 1) Обращение по индексу int& operator[](int num)
- 2) Индекс по значению int find(int val)
- 3) Добавление по индексу void add(int num, int val)
- 4) Удаление по индексу void delete\_element(int num)
- 5) Создание списка void create\_list(int new\_size)
- 6) Получение размера списка int get\_size()

- 7) Вывод на экран данных списка `void print_data()`
- 8) Удаление списка `void delete_list()`
- 9) Вставка в начало `void push_front(int val)`
- 10) Вставка в конец `void push_back(int val)`
- 11) Получение ссылки на элемент `MyNode* get_reference(int num)`

### 3) Стек

Для создания стека создаём класс `MtStack`

В нём создаём ещё один класс `MyNode`

Реализация стека является реализацией двусвязного списка без некоторых функций

Стандартный конструктор `MyStack()`

Деструктор `~MyStack`

Методы класса:

- 1) Снятие со стека `void pop()`
- 2) Добавление на стек `void push(string newData)`
- 3) Значение вершины стека без его снятия `string peak()`
- 4) Вывод данных стека на экран `void print_data()`

### 4) Алгоритм сортировочной станции

1. Пока есть ещё символы для чтения:

- Читаем очередной символ.
- Если символ является числом или постфиксной функцией (например, **!** — факториал), добавляем его к выходной строке.
- Если символ является префиксной функцией (например, **sin** — синус), помещаем его в стек.
- Если символ является открывающей скобкой, помещаем его в стек.
- Если символ является закрывающей скобкой:
  - До тех пор, пока верхним элементом стека не станет открывающая скобка, выталкиваем элементы из стека в выходную строку. При этом открывающая скобка удаляется из стека, но в выходную строку не добавляется.

- Если существуют разные виды скобок, появление непарной скобки также свидетельствует об ошибке. Если какие-то скобки одновременно являются функциями (например, [x] — целая часть), добавляем к выходной строке символ этой функции.
- Если символ является бинарной операцией **o1**, тогда:
  1. пока на вершине стека префиксная функция...
    - **ИЛИ** операция на вершине стека приоритетнее **o1**;
    - **ИЛИ** операция на вершине стека левоассоциативная с приоритетом как у **o1**;
    - выталкиваем верхний элемент стека в выходную строку.
  2. помещаем операцию **o1** в стек.
  3. Когда все символы входной строки перебраны, выталкиваем все символы из стека в выходную строку.

### Демонстрация работы программы.

```

Enter an expression: 7 * cos ( 5 + 2 ) + 3
Your answer: 7 5 2 + cos * 3 +

Enter an expression: 7 + ln ( 2 )
Error! Token 'ln' is invalid

Enter an expression: 5 + ( 6 * 3
Error! Extra left bracket

Enter an expression: 5 * ( 1 - 3 * 5 ) - 2 )
Error! Extra right bracket

```

### Вывод.

В результате работы были реализованы следующие структуры данных:

- 1) Динамический массив
- 2) Односвязный список
- 3) Стек

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