**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Вычисление арифметических выражений**

| Студент гр. 1301 |  | Ахметзянов Д.А. |
| --- | --- | --- |
| Преподаватель |  | Пестерев Д.О. |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

| Студент Ахметзянов Д.А. | | |
| --- | --- | --- |
| Группа 1301 | | |
| Тема работы: Вычисление арифметических выражений | | |
| Исходные данные:  Строка с арифметическим выражением | | |
| Содержание пояснительной записки:  “Формулировка задания”, “Описание алгоритма”, “Пример выполнения”, “Код программы” | | |
|  | | |
| Дата выдачи задания: 14.11.2022 | | |
| Дата сдачи реферата: | | |
| Дата защиты реферата: | | |
| Студент |  | Ахметзянов Д.А. |
| Преподаватель |  | Пестерев Д. |

**АННОТАЦИЯ**

Считывается идентификатор нотации в которой записано выражение и само выражение. В зависимости от нотации выражение преобразуется в двоичное дерево с операндами в листьях и операторами в узлах. Затем для каждого оператора рекурсивно вычисляется значение в его поддереве. Результат всего выражения будет находится в корне дерева.

**SUMMARY**

The identifier of the notation in which the expression is written and expression itself is read. Depending on the notation expression is being converted to a binary tree with operands in it’s leafs and operators in nodes. Then for each operator value is recursively calculated for it’s subtree. Result of the whole expression will be in the root of the tree.

**СОДЕРЖАНИЕ**

[**Формулировка задания.**](#_heading=h.oc18e82qb35r) **4**

[**Описание алгоритма.**](#_heading=h.30j0zll) **4**

[**Пример выполнения.**](#_heading=h.3znysh7) **5**

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Формулировка задания.

Реализовать вычисление арифметических выражений с

бинарными операциями {+,-,\*,/,^} и числовыми операндами,

записанных в префиксной/инфиксной/постфиксной записи.

# Описание алгоритма.

По выражению в зависимости от формы записи строится бинарное дерево, в котором в листьях хранятся операнды, а в остальных узлах операторы.

1. Префиксная запись.

Если первое выражение в строке это число то узлу переданному в функцию присваивается его значение.

Иначе если в строке первым символом является оператор то создается узел для него создаются правый и левые потомки.

Для них функция вызывается рекурсивно с урезанным выражением (длина выражения обработанного выражения возвращается из функции).

1. Постфиксная запись.

Аналогично префиксной с отличием только в том, что обработка строки идет с конца.

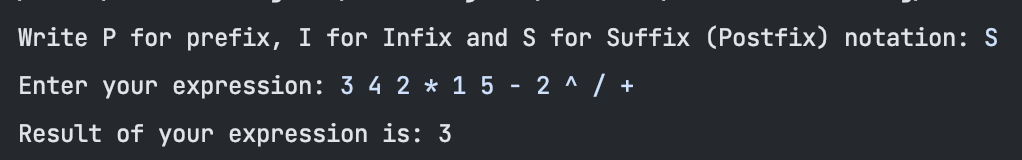
1. Инфиксная запись.

Парсисится первый операнд, если после этого строка осталось пустой то возвращается узел со значением этого операнда, иначе парсится оператор, для него создается узел, первый операнд вставляется в его левого потомка, а правый получается из рекурсивного вызова функции с оставшейся частью выражения.

Для того чтобы учитывать порядок выполнения выражений при первом проходе обрабатываются только операторы + и -, то есть выражения вида 2 \* 2 будут помещены в узел как единое целое, затем для таких узлов происходит обработка операций \* и /, и в конце операции ^.

Для получения результата выражения, начиная с корня дерева у узлов вызывается метод computeValue, который применяет содержащийся в узле операнд к результатом того же метода своих левого и правого потомков.

# Пример выполнения.



# Код программы

<https://github.com/DamirAhm/AADS/tree/master/Kursach>