**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра** **ВТ**

**ОТЧЕТ**

**по лабораторной работе №4**

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

**Тема: «Двоичные деревья»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0322 |  | Руссу В.А. |
| Преподаватель |  | Пестерев Д.О. |

Санкт-Петербург

2022

**Постановка задачи**

Реализовать структуру данных двоичное дерево поиска и следующие методы: нахождение минимума, нахождение максимума, прямой (preorder), центрированный (inorder) и обратный обход (postorder) по дереву, поиск элемента, нахождение следующего и предыдущего элемента, удаление элемента, обход в ширину. Реализовать визуализацию дерева. Указать теоретическую временную сложность для всех операций.

С помощью реализованной структуры данных написать программу, позволяющую преобразовать запись из префиксной/инфиксной/постфиксной нотации в префиксную/инфиксную/постфиксную нотацию.

**Описание реализуемых функций**

Insert – вставка элемента;

FindMin – нахождение минимума;

FindMax – нахождение максимума;

printPreorder – прямой обход по дереву;

printInorder – центрированный обход по дереву;

printPostorder – обратный обход по дереву;

Search – поиск элемента;

Successor – нахождение следующего элемента;

Predecessor – нахождение предыдущего элемента;

Remove – удаление элемента;

printWidth – обход в ширину;

printTree – визуализация дерева;

construct – инициализация дерева выражений;

preorder – выражение выводится в префиксной форме;

postorder – выражение выводится в постфиксной форме;

inorder – выражение выводится в инфиксной форме.

**Теоретическая временная сложность**

h – высота дерева. N – количество узлов в дереве.

Insert – сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

FindMin – сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

FindMax – сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

printPreorder – сложность O(N), так как рекурсия применяется для каждого узла в дереве.

printInorder – сложность O(N), так как рекурсия применяется для каждого узла в дереве.

printPostorder – сложность O(N), так как рекурсия применяется для каждого узла в дереве.

Search - сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

Successor - сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

Predecessor - сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

Remove - сложность O(h), так как дерево отсортировано, рекурсия вызывается по узлам только в нужном направлении.

printWidth - сложность O(N), так как рекурсия применяется для каждого узла в дереве.

printTree - сложность O(N), так как рекурсия применяется для каждого узла в дереве.

Construct - сложность O(N), так как рекурсия применяется для каждого символа в строке.

preorder - сложность O(N), так как рекурсия применяется для каждого символа в строке.

postorder - сложность O(N), так как рекурсия применяется для каждого символа в строке.

inorder - сложность O(N), так как рекурсия применяется для каждого символа в строке.

**Пример работы программы**

В примере (Рисунок 1) показана работа всех функций для бинарного дерева. Также на рисунке 1 было введено выражение в инфиксной форме, которое было преобразовано в постфиксную и префиксную формы.

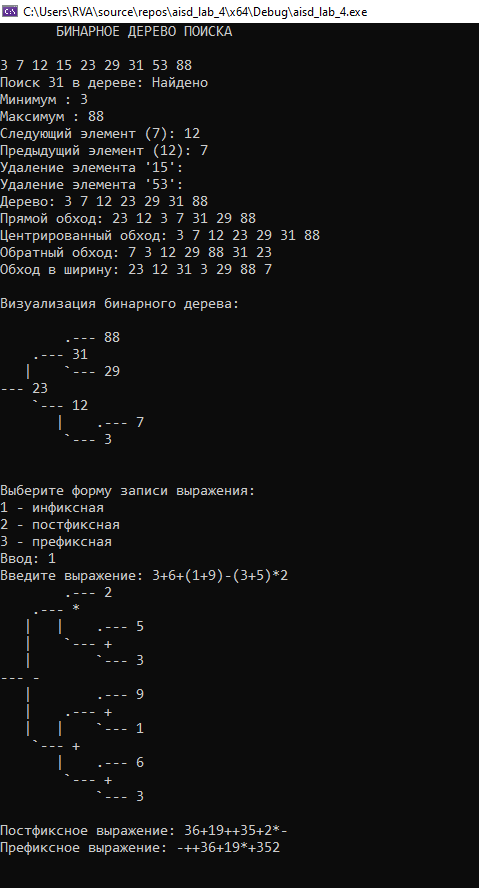


Рисунок 1

На рисунке 2 введено выражение в постфиксной форме, которое было преобразовано в инфиксную и префиксную формы.

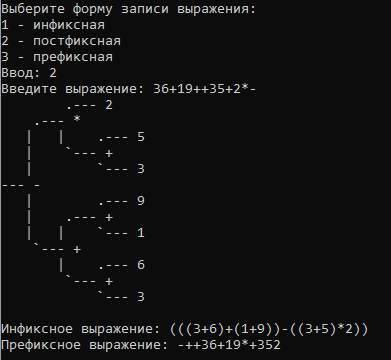


Рисунок 2

На рисунке 3 введено выражение в префиксной форме, которое было преобразовано в инфиксную и постфиксные формы.

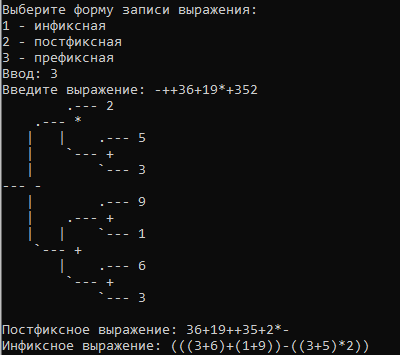


Рисунок 3

**Выводы**

В ходе выполнения данной лабораторной работы было реализовано бинарное дерево поиска и дерево арифметических выражений. В бинарном дереве поиска были реализованы следующие методы: нахождение минимума, нахождение максимума, прямой (preorder), центрированный (inorder) и обратный обход (postorder) по дереву, поиск элемента, нахождение следующего и предыдущего элемента, удаление элемента, обход в ширину. Запись арифметических выражений мы научились преобразовывать из префиксной/инфиксной/постфиксной нотации в префиксную/инфиксную/постфиксную нотацию.

Ссылка на GitHub

https://github.com/RVA-t/aisd\_lab\_4.git