



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

## **Отчет по лабораторной работе №6** **«ОБРАБОТКА ДЕРЕВЬЕВ»**

Студент Козырных Александр

Группа ИУ7 – 32Б

Преподаватель Силантьева А. В.

Вариант 6

2023 год.

## Оглавление

<u>ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ.....</u>	<u>3</u>
<u>ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....</u>	<u>3</u>
<u>НАБОР ТЕСТОВ.....</u>	<u>4</u>
<u>ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ.....</u>	<u>4</u>
<u>ОЦЕНКА ЭФФЕКТИВНОСТИ.....</u>	<u>5</u>
<u>ОПИСАНИЕ АЛГОРИТМА.....</u>	<u>5</u>
<u>ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....</u>	<u>6</u>

## ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ

Ввести значения переменных: от А до I. Построить и вывести на экран бинарное дерево следующего выражения:  $A + (B * (C + (D * (E + F) - (G - H)) + I))$ . Написать процедуры постфиксного, инфиксного и префиксного обхода дерева и вывести соответствующие выражения на экран. Подсчитать результат. Используя «польскую» запись, ввести данное выражение в стек. Сравнить время вычисления выражения с использованием дерева и стека.

## ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

### Входные данные:

Пункты меню и числовые данные.

### Выходные данные:

Двоичное дерево, результат расчета выражения с помощью дерева/стека.  
Время вычисления.

### Обращение к программе:

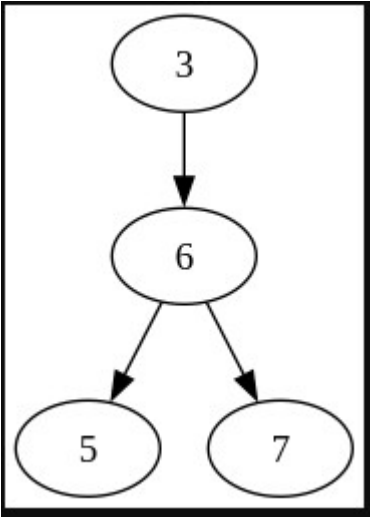
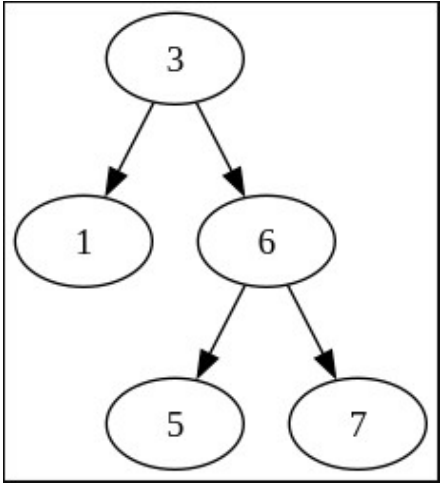
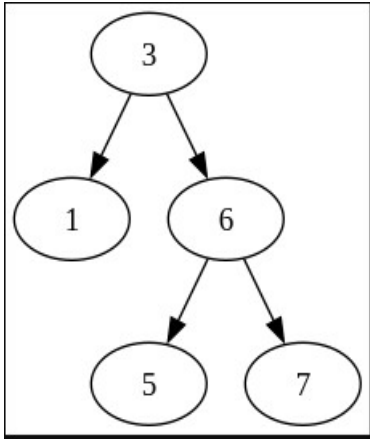
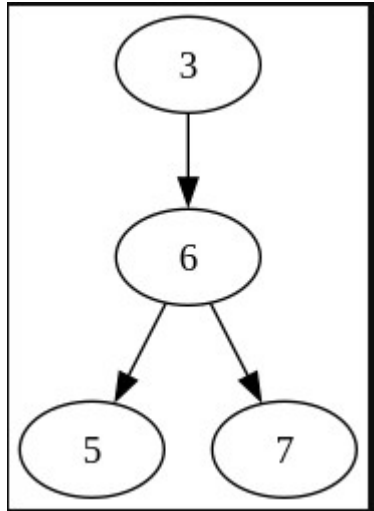
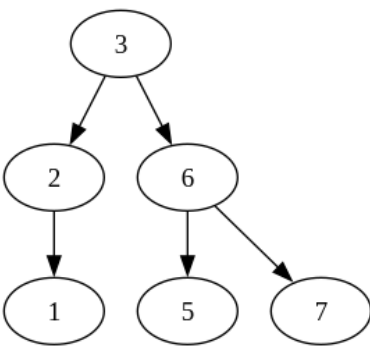
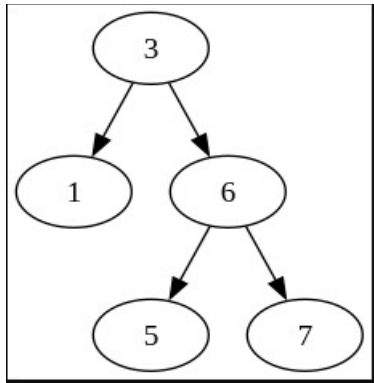
Запускается через терминал командой: ./app.exe.

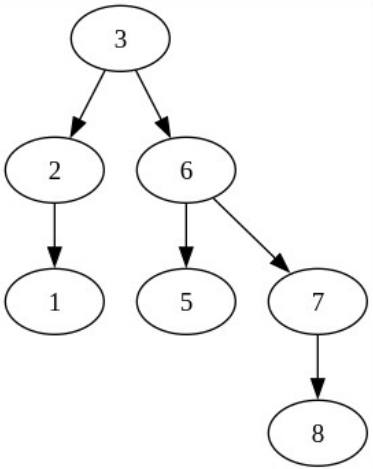
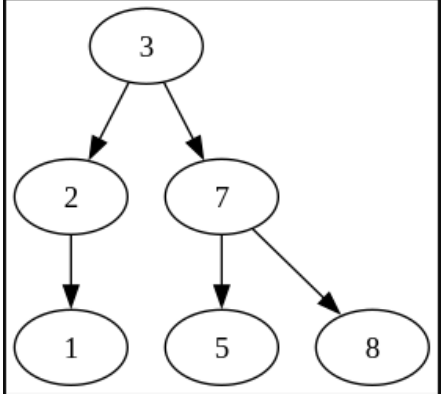
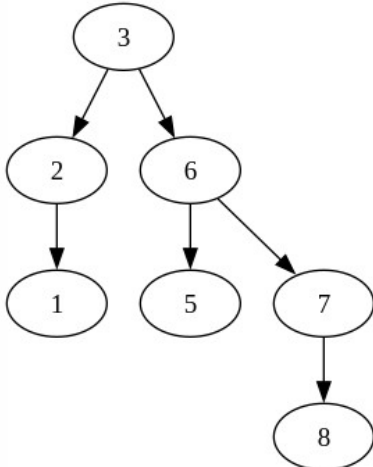
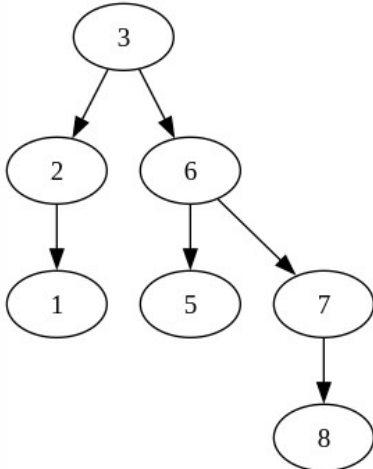
### Аварийные ситуации:

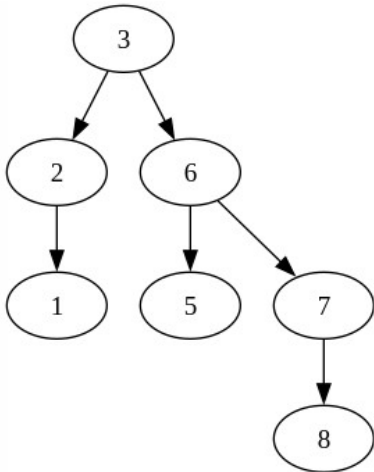
1. Некорректные данные переменной/пункта меню.
2. Ошибка выделения памяти.

## НАБОР ТЕСТОВ

№	Название теста	Пользовательский ввод	Вывод
1	Корректный ввод переменных	1 1 2 3 4 5 6 7 8 9	Выражение

2	Добавление узла в дерево	 <p>+ 1</p>	
3	Удаление листа	 <p>-1</p>	
4	Удаление узла с 1 потомком	 <p>-2</p>	

5	Удаление узла с двумя потомками	 <p style="text-align: center;">-6</p>	
6	Префиксный обход дерева		3 2 1 6 5 7 8
7	Инфиксный обход дерева		1 2 3 5 6 7 8

8	Постфиксный обход дерева	 <pre> graph TD     3((3)) --&gt; 2((2))     3 --&gt; 6((6))     2 --&gt; 1((1))     6 --&gt; 5((5))     6 --&gt; 7((7))     7 --&gt; 8((8)) </pre>	1 2 5 8 7 6 3
9	Некорректный ввод переменной	1.0	
10	Некорректный ввод переменной	a	Некорректно выбран пункт меню
11	Ошибка памяти	----	Ошибка выделения памяти! или Произошла непредвиденная ситуация
12	Попытка вставить существующую вершину	...	Такая уже есть
13	Попытка удалить несуществующую вершину	...	Такой вершины нет

## ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ

```
5 /*typedef на узел дерева*/  
4 typedef struct node tree_node_t;  
3  
2 struct node {  
1     tree_node_t *left;  
0     tree_node_t *right;  
9     int value;  
8     char option;  
7     /*  
6     left - Левый потомок текущего узла  
5     right - Правый потомок текущего узла  
4     value - числовое значение узла  
3     option - операция над числом (для вычисления выражения)  
2     */  
1 };
```

```
5 typedef struct node_t node_t;  
4 struct node_t  
3 {  
2     /* Узел стека  
1     * prev - Указатель на предыдущий узел  
2     * type - целочисленное значение  
1     */  
2     int type;  
3     node_t *prev;  
4 };  
5  
6  
7 typedef struct list_stack_t  
8 {  
9     /*  
10    * Стек на листе  
11    *  
12    * index - Размер текущего стека  
13    * top - Указатель на вершину (первый) узел стека  
14    */  
15    int index;  
16    node_t *top;  
17 } list_stack_t;
```

## ОЦЕНКА ЭФФЕКТИВНОСТИ (МС)

	Бинарное дерево	Стек на листе	Линейное дерево
Подсчёт выражения	0.077	0.251	----
Добавление узла  И  Удаление узла	0.09	0.032	7.83

Тесты проводились 10000 раз. Для тестов про удаление и добавление узлов использовалось количество тестов (10000) в виде количества узлов. Как можем заметить, скорость работы двоичного дерева для расчета выражения в  $\sim 4$  раза быстрее стека. Однако скорость вставки И опустошения (полной очистки) у разветвленного дерева в  $\sim 3$  раза медленнее.

Скорость работы над выражением у дерева выше потому, что оно работает с бинарными операциями, и такому дереву не нужно постоянно переставлять свои элементы, просто пройти по нему 1 раз. Стек в свою очередь постоянно убирает, считает и обратно вставляет выражения, что замедляет скорость его работы.

Однако скорость добавления и удаления легко объясняется. В то время когда у стека скорость удаления и добавления  $O(1)$ , у дерева же обе операции имеют ассимптотическую сложность  $O(\log_2(n))$ .

## ОПИСАНИЕ АЛГОРИТМА

1. После запуска программы пользователю предлагается ввести пункт меню
2. В зависимости от пункта меню вводятся числа или выводится результат.



## ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

### 1. Что такое дерево?

Дерево – это структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».

### 2. Какие бывают типы деревьев?

Деревья двоичного поиска, AVL-деревья, двоичные.

### 3. Какие стандартные операции возможны над деревьями?

Поиск по дереву, обход дерева, добавление/удаление элемента (узла) из дерева.

### 4. Что такое дерево двоичного поиска?

Двоичное дерево поиска (ДДП) — двоичное дерево, в котором для каждого узла выполняется условие, что левый узел меньше текущего, а правый больше. Узлы с одинаковыми значениями не допускаются.

## Вывод

Скорость вычисления выражения деревом в 4 раза быстрее стека, однако скорость добавления и удаления в 3 раза медленнее. Но преимущества двоичного дерева состоит в том, что доступ к любому его элементу осуществимо за  $O(\log_2(n))$ , когда как у стека это  $O(n)$  (причем помноженный на два).