



**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)**

## Лабораторная Работа №6 «Деревья» Вариант №6

Студент **Шахнович Дмитрий Сергеевич**

Группа **ИУ7-22Б**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент Шахнович Д.С.

Оценка \_\_\_\_\_

2023 թ.

## Описание условия задачи

Построить дерево в соответствии со своим вариантом задания. Вывести его на экран в виде дерева. Реализовать основные операции работы с деревом: обход дерева, включение, исключение и поиск узлов. 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.

## **Техническое задание**

### **Исходные данные:**

Для вычислений данными являются 9 целых чисел А, В... I, которые применяются для вычисления выражения из условия задачи. Во время работы программы данными могут быть символы, которую будет обрабатываться в дереве.

### **Выходные данные:**

Программа должна выдавать схему нынешнего дерева в виде графа, информации о наличии отдельного символа в дереве, различные обходы по дереву, выводить результат вычисления заданного выражения и выводить информацию о сравнении различных вариантов реализации вычисления выражений.

### **Описание задания:**

Реализация основных методов бинарных деревьев, а также сравнение скорости вычисления выражения на дереве и стеке.

### **Способы обращения к программе:**

Запуск программы через терминал, затем управление программой с помощью меню. Пункты меню:

- 1 — Добавить символ в дерево.
- 2 — Найти символ в дереве.
- 3 — Удалить символ из дерева
- 4 — Префиксный обход дерева
- 5 — Инфиксный обход дерева
- 6 — Постфиксный обход дерева
- 7 — Вывести изображение дерева
- 8 — Изменить значения переменных в выражении
- 9 — Провести вычисления деревом
- 10 — Провести вычисления статическим стеком
- 11 — Провести вычисления списком-стеком
- 12 — Сравнить реализации
- 0 — Выход.

### **Аварийные ситуации:**

- 1) Ввод несуществующей команды в меню;  
Сообщение: «Ошибка: Некорректная команда.»
- 2) Ошибка ввода/вывода;  
Сообщение: «Ошибка функций ввода/вывода.»
- 3) Переполнение статического стека;  
Сообщение: «Ошибка: стек переполнен.»
- 4) Неудачная попытка работы с файлом  
Сообщение: «Ошибка при работе с файлом.»
- 5) Неудачная попытка выделения памяти;  
Сообщение: «Ошибка выделения памяти.»
- 6) Ввод литералов или чисел вне запрашиваемого диапазона;

Сообщение: «Ошибка: Некорректный формат ввода.»

7) Удаление элемента дерева, которого не существует

Сообщение: «Ошибка: Элемент не найден.»

8) Добавление существующего элемента в дерево

Сообщение: «Ошибка: Элемент уже существует.»

9) Деление на ноль в дереве

Сообщение: «Ошибка: Деление на ноль.»

## Описание структур данных

```
/// @brief Структура узла бинарного дерева выражения
struct btree_node
{
    btree_node *parent; /// Указатель на родителя узла, для корня - NULL
    btree_node *left; /// Указатель на левого потомка
    btree_node *right; /// Указатель на правого потомка
    int data_id; /// Переменная для определения информационной части
    узла.
    /// Если VALUE_ID, то хранится число, Если OPERATION_ID, то хранится
    символ операции
    union
    {
        int value;
        char op;
    } data; /// Информационная часть узла
};
```

```
/// @brief Узел стека в виде односвязного списка
struct stack_node_t
{
    int data; /// Значение узла
```

```
stack_node_t *next; /// Указатель на следующий(более близкий к  
начальному) элемент стека.  
};
```

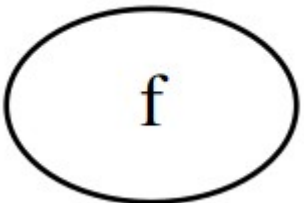
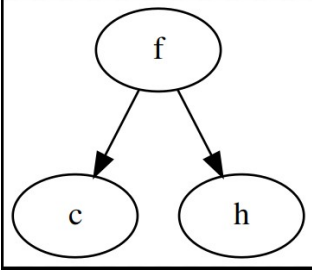
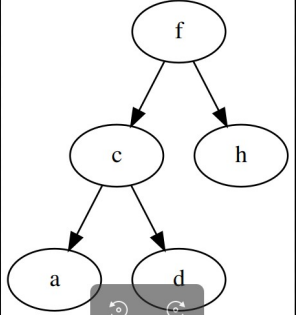
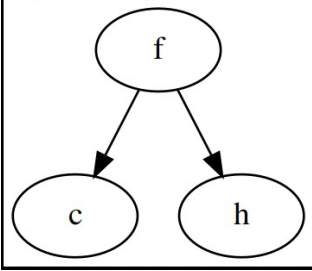
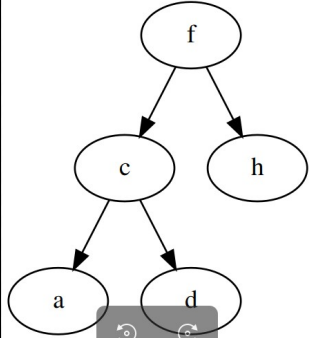
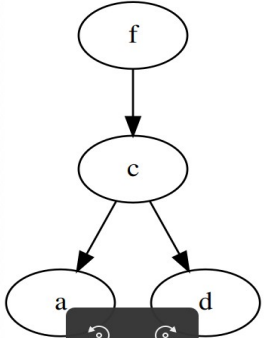
```
/// @brief Реализация стека с помощью статического массива  
struct static_stack_t  
{  
    int *head_ptr; /// Указатель на голову стека  
    int *end_ptr; /// Указатель на конец статического массива  
    int arr[STATIC_STACK_SIZE]; /// Статический массив  
};
```

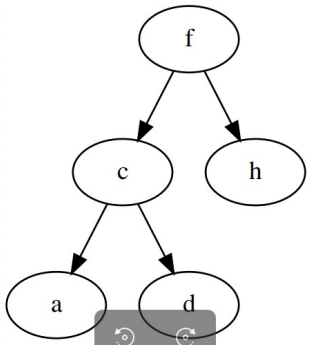
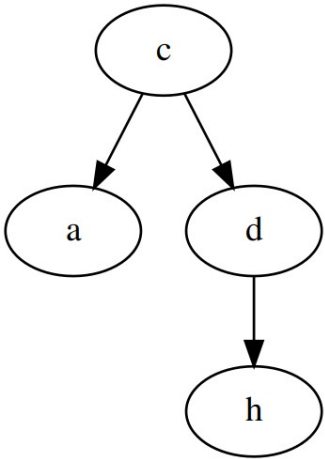
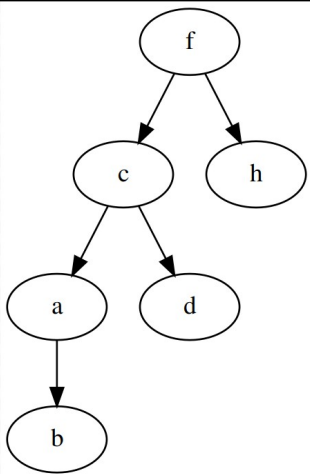
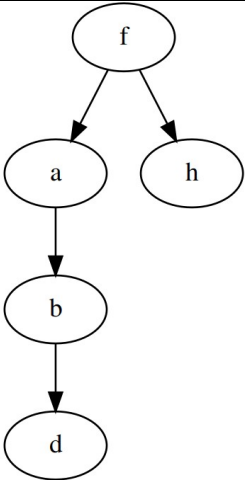
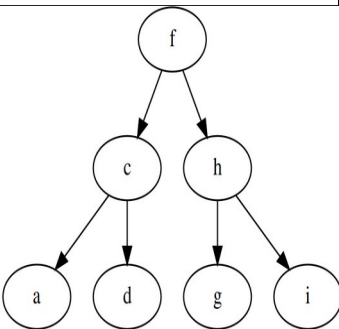
## Описание алгоритма

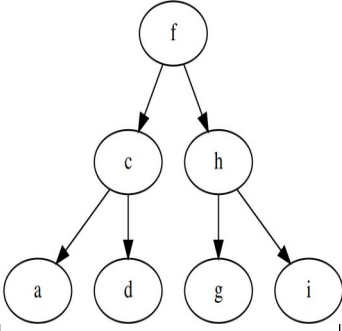
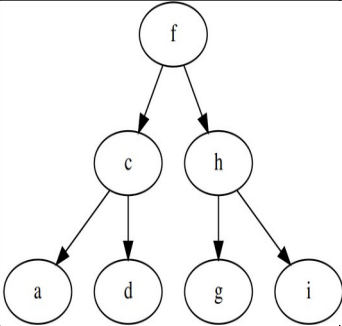
1. Вывести пользователю меню и ожидать ввода номера команды;
2. В зависимости от выбора пункта провести операцию с текущим деревом символов или провести вычисления на дереве выражений.

При возникновении ошибок или завершения пункта вернуться к выбору пункта меню.

# Тестовые данные

Позитивные тесты			
№	Описание	Вход	Выход
1	Добавить символ в пустое дерево	Граф: Пустой f	
2	Добавить символы в дерево с несколькими элементами	 a d	
3	Найти существующий символ в дереве	 h	Символ найден в дереве.
4	Удалить лист дерева	 h	

5	Удалить корень дерева		
6	Удалить потомка из середины дерева	 <p>c</p>	
7	Префиксный обход дерева		fcadhgi

8	Инфиксный деревя	обход		acdfghi
9	Постфиксный деревя	обход		adcgihf
10	Посчитать выражение(всеми способами) стандартными переменными	со	A... = 1, 2, 3, 4, 5, 6, 7, 8, 9	115
11	Посчитать выражение(всеми способами) переменные	изменив	A... = -2, 4, 2, 6, 12, -2, 1, 3, 4	270
Негативные тесты				
1	Ввести некорректный код в меню.	16	Ошибка: Некорректная команда.	
2	При запросе числа ввести литерал.	sda	Ошибка: Некорректный формат ввода.	
3	При запросе символа ввести несколько	sdw	Ошибка: Некорректный формат ввода.	
4	Добавить символ в	Дерево: «а»	Ошибка: Элемент уже	



	дерево, где уже есть этот символ	a	существует.
5	Удалить не существующий в дереве символ	Дерево: «a» b	Ошибка: Элемент не найден.
6	Вывести обход у пустого дерева	Дерево: пустой	Пустое дерево
7	Задать выражение в дереве с делением на ноль	10 / 0	Ошибка: Деление на ноль.

## Замеры эффективности

Замеры выполнения операции проводились 1000000 раз для каждого метода хранения со случайными значениями чисел. В качестве результата бралось среднее.

### Время выполнения вычислений

Время вычисления деревом, нс	Время в списке-стеке, мкс	Время в статическом стеке, нс
252	192	493

Как видно самым быстрым оказался метод вычисления статическими стеком, вероятно из-за того, что это единственный метод со статическим выделением памяти. Самым медленным был вариант со стеком-списком.

## Ответы на вопросы

1. Что такое дерево? Как выделяется память под представление деревьев?

Дерево — нелинейная структура данных, которая используется для иерархических связей. Любое дерево определяется либо как пустое, либо как узел, к которому присоединены деревья, называемые поддеревьями. При этом у каждого поддерева может быть только одна связь с узлом из более высокого дерева.

Память под деревья выделяется динамически, отдельно под каждый узел дерева, из-за чего узлы дерева могут находиться в разных участках памяти.

2. Какие бывают типы деревьев?

Деревья можно разделить по количеству узлов-потомков у одного узла. Если их число не превышает 2, то такое дерево называется двоичным. Если двоичное дерево построено так, что все левые потомки меньше данного, а все правые — больше, то такое дерево называется двоичным деревом поиска.

Также среди деревьев можно выделить самобалансирующиеся бинарные деревья, в которых разница между высотами левой и правой ветвей не больше 1. Одной из разновидностью самобалансирующихся деревьев является красно-черное дерево, в котором с помощью особых правил достигается самобалансировка дерева, при этом каждый узел имеет цвет — красный или черный.

3. Какие стандартные операции возможны над деревьями?

Для деревьев стандартными операциями определяют:

- Включение элемента в дерево
- Исключения элемента из дерева
- Поиск элемента в дереве
- Обход дерева(прохождения по всем его узлам)

#### 4. Что такое дерево двоичного поиска?

Деревом двоичного поиска называется двоичное дерево, в котором в левом поддереве узла находятся элементы меньше данного, а в правом — большие. Такое расположение позволяет проводить операции поиска, добавления и удаления из бинарного дерева за  $O(\log n)$ , где  $n$  — глубина дерева.

## **Выводы**

Во многих компьютерных системах для моделирования используется структура данных — дерево. Особенно часто используется его вариация — дерево двоичного поиска. Примером такой системы может быть решение арифметических выражений, которое как показал эксперимент, может быть быстрее чем с использованием стека. Однако при использовании дерева программисту требуется следить за правильностью всех указателей, использующихся в структуре.