|  |  |
| --- | --- |
|  | **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**

Студент **Паламарчук А.Н.**

Группа **ИУ7-33Б**

Предмет **Типы и структуры данных**

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

|  |  |
| --- | --- |
| Студент | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Паламарчук А.Н.** |
| Преподаватель | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Никульшина Т. А.** |
| Преподаватель | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Барышникова М. Ю.** |

*2023 г.*

# Условие задачи

Построить дерево в соответствии со своим вариантом задания. Вывести его на экран в виде дерева. Реализовать префиксный, инфиксный и постфиксный обходы дерева. Сравнить эффективность алгоритмов вычисления выражения.

# Техническое задание

Ввести значения переменных: от A 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.

## Входные данные

* Команда (число) из меню программы (см. меню программы) – обозначение необходимой операции.

## Выходные данные

* Таблицы эффективности
* Файл в DOT формате
* Файл .png с деревом
* Выражение, полученное префиксным обходом
* Выражение, полученное инфиксным обходом
* Выражение, полученное постфиксным обходом
* Результат вычисления выражения

## Возможные аварийные ситуации

* Некорректный исходный файл
* Ошибка открытия файла
* Отсутствие исходного файла
* Некорректная команда

## Описание внутренних структур данных

typedef struct tree\_node

{

const char \*name;

int val;

// родитель

struct tree\_node \*parent;

// меньшие

struct tree\_node \*left;

// большие

struct tree\_node \*right;

} tree\_node\_t;

**Константы**

#define COUNT\_TESTS 100

#define FILE\_DOT "graph.gv"

#define FILE\_PNG "graph.png"

**Используемые функции**

int fill\_tree(struct val\_nodes \*tree);

Заполнение дерева значениями пользователя

void export\_to\_dot(FILE \*f, const char \*tree\_name, struct tree\_node \*tree);

Перевод дерева в DOT формат

void apply\_pre(tree\_node\_t \*tree, void (\*f)(tree\_node\_t\*, void\*), void \*arg);

Префиксный обход дерева

void apply\_in(tree\_node\_t \*tree, void (\*f)(tree\_node\_t\*, void\*), void \*arg);

Инфиксный обход дерева

void apply\_post(tree\_node\_t \*tree, void (\*f)(tree\_node\_t\*, void\*), void \*arg);

Постфиксный обход дерева

void print\_name\_node(tree\_node\_t \*a, void \*b);

Распечатывает название узла

int evaluate\_postfix\_expression(tree\_node\_t \*node);

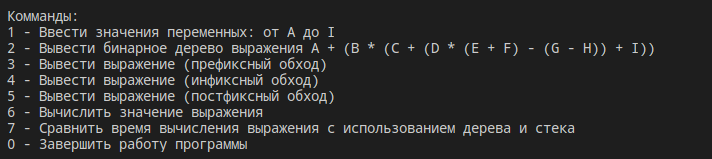
Вычисляет значение выражения

int cmp\_tree\_vs\_stack(struct val\_nodes tree);

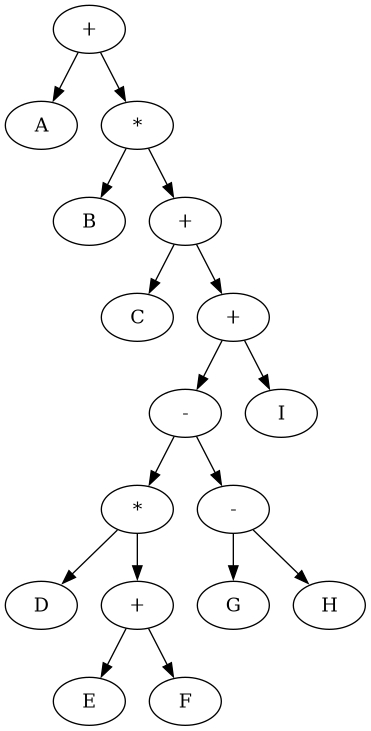
Выводит таблицы сравнения

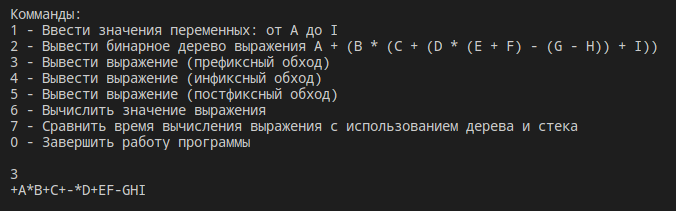
**Меню программы**

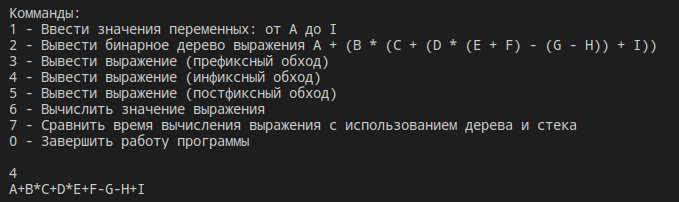
Программа обрабатывает нужную команду:

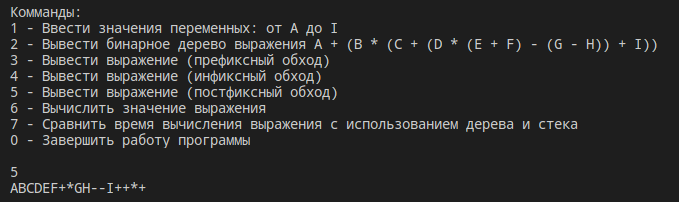
****

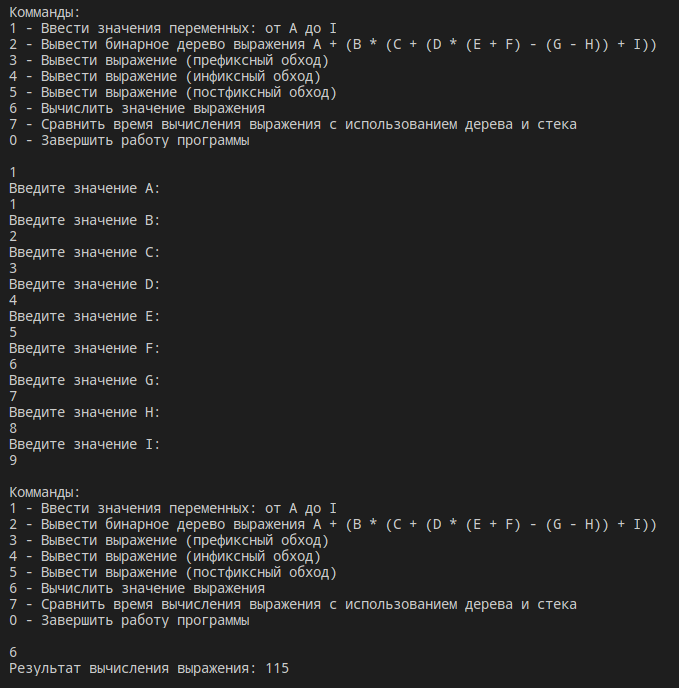
**Пример работы программы**

****





****

****

**Производительность**

# C:\Users\777\Downloads\image (2).png

# Выводы по проделанной работе

Деревья следует использовать в случаях, когда нужен быстрый поиск элементов (бинарное дерево), а также для хранения данных в иерархической форме. Кроме того, имеется существенный недостаток, который заключается в том, что при удалении требуется перестроение дерева, которое выполняется рекурсивно и требует определенных ресурсов.

В моей реализации вычисление значения выражения с помощью дерева занимает меньше времени, чем вычисление при помощи стека.

# Контрольные вопросы

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

*Дерево* – это нелинейная структура данных, используемая для представления иерархических связей (один к нескольким).

Динамически под каждый узел связного списка.

**2. Какие бывают типы деревьев?**

* *Двоичное дерево* - иерархическая структура данных, в которой каждый узел имеет не более двух потомков.
* *Красно-чёрное дерево* - один из видов самобалансирующихся двоичных деревьев поиска, гарантирующих логарифмический рост высоты дерева от числа узлов и позволяющее быстро выполнять основные операции дерева поиска: добавление, удаление и поиск узла.
* *АВЛ-дерево* - сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

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

Обход дерева, включение, исключение и поиск узлов

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

Двоичное дерево, для которого выполняются следующие дополнительные условия.

* оба поддерева - левое и правое - являются двоичными деревьями поиска;
* у всех узлов *левого* поддерева произвольного узла X значения ключей данных *меньше либо равны*, нежели значение ключа данных самого узла X;
* у всех узлов *правого* поддерева произвольного узла X значения ключей данных *больше*, нежели значение ключа данных самого узла X.