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»
Вариант 6

Студент Кладницкий А. Б.

Преподаватель Силантьева А. В.

Группа ИУ7 – 32Б

2022 г.

1. **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. Сравнить эффективность алгоритмов сортировки и поиска в зависимости от высоты деревьев и степени их ветвления.

2. **ТЗ**

   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.

   1. Исходные данные:
      - Пункт меню
      - Значения переменных
      - Данные для добавления/поиска/удаления
      - Меню:
        1. Enter variables and set expression
        2. Set custom expression
        3. Count expression
        4. Insert into tree
        5. Search in tree
        6. Delete from tree
        7. Show tree
        8. Print tree in prefix form
        9. Print tree in infix form
        10. Print tree in postfix form
        11. Print menu again
        12. Check efficiency vs stack
        13. Check efficiency of different sizes
        0. Exit
   2. Результирующие данные:
      - В зависимости от пункта меню:
        - Подсчитанное выражение
        - Сообщение об успешности удаления/добавления данных
        - Вывод дерева заданным обходом
        - Вывод дерева графически
        - Результаты замеров времени
   3. Задача программы:
      - Построение двоичного дерева
      - Подсчет двоичного дерева
      - Замер эффективности разных алгоритмов подсчета выражения

4. Способ обращения к программе:
    - Запуск через терминал (./app.exe)
5. Возможные ошибки:
    - Ошибка выделения динамической памяти
    - Ошибка чтения из потока
    - Ошибка чтения целого числа
    - Использование определенных пунктов меню без предварительной подготовки

3. Описание внутренних структур данных

Тип данных `tree_t`:

```
typedef struct tree_node tree_t;
struct tree_node
{
  char *data;
  tree_t *right;
  tree_t *left;
};
```

Тип содержит в себе данные, хранящиеся в узле дерева, и указатели на его левого и правого потомков.

4. Алгоритм

1. Считать пункт меню
2. Выполнить действие в соответствии с ним
3. Для создания дерева выражения:
    i. Разбить выражение на 2
    ii. Добавить их и знак в дерево
    iii. Продолжать для обоих выражений, пока они не сведутся к числам

5. Набор тестов

	Описание	Входные данные	Результат
1.	Неверный пункт меню	q, 100, -5 и т.д.	EXIT_WRONG_MENU_NUM
2.	Ошибка выделения динамической памяти	???	EXIT_ALLOCATE
3.	Неверное целое число для переменной A..I	Ввод переменных, 1er34o, 66.3 и т.д.	EXIT_BAD_INT
4.	Пустой ввод	-	EXIT_WRONG_READ
5.	Подсчет незаполненного дерева	Незаполненное дерево, запрос подсчета	EXIT_CANT_COUNT
6.	Ошибка чтения числа для поиска/вставки/удаления	1hg12, 7.82, wjkgh и т.д.	EXIT_BAD_INT
7.	Ввод переменных	Пункт меню 1 Значения переменных	Введенные переменные
8.	Расчет заданного выражения	Пункт меню 2	Результат подсчета выражения
9.	Вставка/удаление/поиск заданного числа в дереве	Пункт меню 4-6, число	Сообщение об успешности вставки/удаления/поиска числа
10.	Вывод дерева в графическом виде	Пункт меню 7	Дерево в графическом виде
11.	Вывод дерева в префиксном/инфиксном/постфиксном обходе	Пункты меню 8-10	Вывод дерева в заданном обходе
12.	Сравнение эффективности при подсчете стека и дерева	Пункт меню 12	Результаты замеров
13.	Сравнение эффективности при обходе дерева в зависимости от ветвления и высоты	Пункт меню 13	Результаты замеров

6. Оценка эффективности

Эффективность поиска в дереве в зависимости от высоты и ветвления:

(во сколько раз обработка дерева с одной ветвью быстрее, чем со всеми)

Время (мкс):

Высота	Одна ветвь	Полное ветвление	Эфф., раз
10	56	99	1.77
15	74	766	10.35
20	92	12454	135.37
25	112	336240	3002.14

Эффективность подсчета заданного выражения через дерево относительно стека:

(во сколько раз подсчет выражения на дереве быстрее, чем на стеке)

Время (мкс):

Бин. дерево	Стек	Эфф., раз
105	778	7.41

7. **Вывод**

   Подсчет выражения быстрее при помощи бинарного дерева. Скорость поиска в дереве снижается с увеличением его высоты и ветвления.

8. **Контрольные вопросы**

   1. Что такое дерево?

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

   2. Как выделяется память под представление деревьев?

      Рекурсивно. В структуре хранятся данные и указатели на левую и правую ветви, после этого (при необходимости) выделяется память под левого и правого потомков (если они есть)

   3. Какие бывают типы деревьев?
      - Дерево двоичного поиска
      - Двоичное дерево
      - АВЛ дерево
      - Красно-черное дерево

   4. Какие стандартные операции возможны над деревьями?
      - Обход дерева (префиксный, инфиксный, постфиксный)
      - Вставка узла
      - Удаление узла
      - Поиск узла

   5. Что такое дерево двоичного поиска?

      Дерево двоичного поиска – это такое дерево, в котором все левые потомки моложе предка, а правые – старше. Это свойство называется характеристическим свойством дерева двоичного поиска и выполняется для любого узла, включая корень.