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 Нисуев Нису Феликсович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна

2023г.

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.

Описание технического задания

Построить бинарное дерево поиска, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Удалить все слова, начинающиеся на указанную букву. Сравнить время удаления слов, начинающихся на указанную букву, в дереве и в файле.
Допущение: В файлах нет повторяющихся слов

Входные данные:

1. **Номер команды**: целое число в диапазоне $\{-1\} \cup [1; 5]$.
2. **Дополнения к таблице:** строковое или целочисленное поле (в зависимости от команды)

Выходные данные:

1. Результат выполнения команды.
2. Сообщение об ошибке.

Обращение к программе:

Запуск через терминал (./target/app.exe | make run)

Аварийные ситуации:

1. Неверная команда
2. Неверный пользовательский ввод
3. Обращение к пустому файлу или дереву

Описание структуры данных

```
/// @brief Строка
typedef char *string;

/// @brief word_tree_t - дерево двоичного поиска слов
typedef struct leaf{
    string word;              // Слово
    struct leaf *left;        // Левый потомок
    struct leaf *right;       // Правый потомок
} word_tree_t;
```

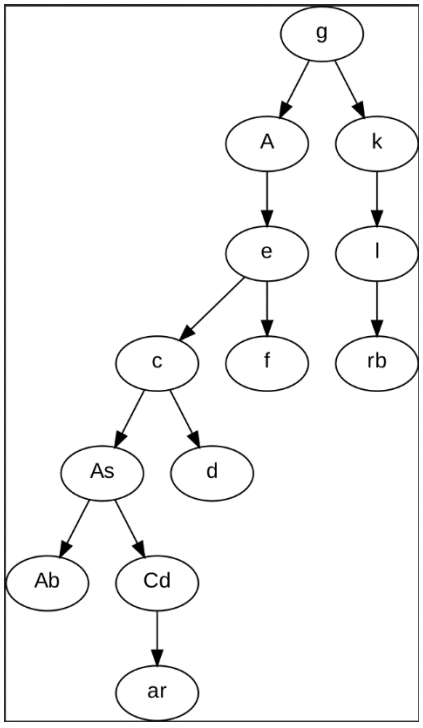
Набор тестов

№	Название теста	Пользовательский ввод	Вывод
Негативные тесты			
1	Некорректные пункт меню	99	ERROR: Incorrect action
2	Некорректный пункт подменю 1	1 3	ERROR: Incorrect action
3	Некорректный пункт подменю 2	4 5	ERROR: Incorrect action
4	Некорректный файл	1 2 Notfile.txt	ERROR: File can’t be opened
5	Пустой файл	1 2 Empty.txt	ERROR: File is empty
6	Пустое дерево	{Дерево пустое}	ERROR: Tree is empty

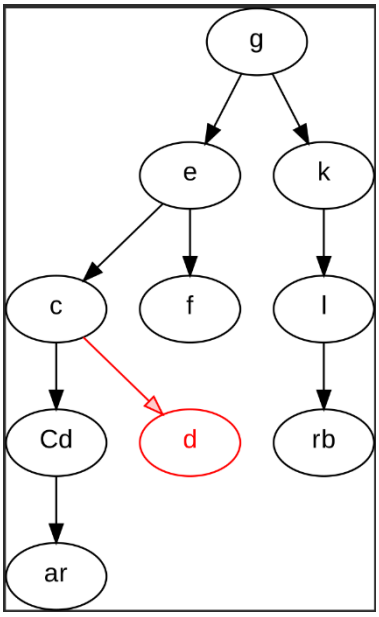
		{Пункты: 2, 3, 4, 5}	
7	Не введено слово для поиска	2 ‘\n’	ERROR: Incorrect input
8	Число вместо буквы	4 1	ERROR: Incorrect input
9	Строка вместо буквы	4 vnwpv	ERROR: Incorrect input
10	Генерация недопустимого числа строк	1 2 200000	ERROR: Incorrect input
11	Генерация отрицательного Количества строк	1 2 -1	ERROR: Incorrect input
Позитивные тесты			
1	Загрузка из сгенерированного файла	1 2 10	{Дерево из 10 случайных слов} Data successfully loaded
2	Считывание дерева из нормального файла	1 2 Realfile.txt	{Дерево из слов в файле} Data successfully loaded
3	Поиск слова в дереве (Слово в дереве есть)	2 {word}	Word “{word}” is founded
4	Поиск слова в дереве (Слова в дереве нет)	2 {word}	Word “{word}” is not founded

5	Удаление слов из дерева (Есть слова начинающиеся на введенную букву)	3 n	Successfully deleted <n> words beginning on “n”
6	Удаление слов из дерева (нет слов начинающихся на введенную букву)	3 n	Words beginning on “n” not founded
7	Вывод дерева	5	{Вывод дерева с помощью dot}
8	Обход дерева	4 1	{Обход дерева}

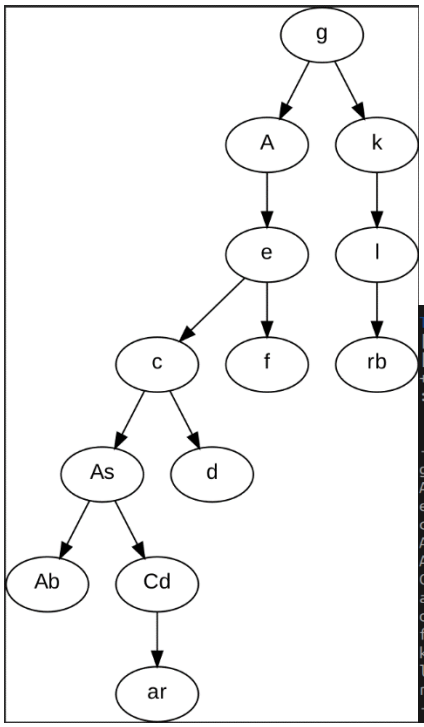
УДАЛЕНИЕ СЛОВ:




```
Input first letter of deleting words: A
Succesfully deleted 3 words beginnig on "A"
```



Обходы дерева:




```
Traversal tree
|----> 1. Preorder traversal
|----> 2. Inorder traversal
+----> 3. Postorder traversal
: 1

--Traversal--
g
A
e
c
As
Ab
Cd
ar
d
f
k
l
rb
--End traversal--
```

```
Traversal tree
|----> 1. Preorder traversal
|----> 2. Inorder traversal
+----> 3. Postorder traversal
: 2

--Traversal--
A
e
c
As
Ab
Cd
ar
d
f
g
k
l
rb
--End traversal--
```

```
Traversal tree
|----> 1. Preorder traversal
|----> 2. Inorder traversal
+----> 3. Postorder traversal
: 3

--Traversal--
A
e
c
As
Ab
Cd
ar
d
f
k
l
rb
g
--End traversal--
```

Оценка эффективности

Замеры производятся 1000 раз. На 1000 случайных словах (Big data), На 1000 словах в которых нет удаляемых (No words for delete), На 1000 словах при которых дерево становится несбалансированным (Linear).

	Tree	File
Big data		
Time, ticks	16371	814402
Memory, b	24000	8000
Linear		
Time, ticks	3861	695694
Memory, b	24000	8000
No words for delete		
Time, ticks	1448	502517
Memory, b	24000	8000

Контрольные вопросы

1.Что такое дерево?

Дерево – это нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим». Дерево с базовым типом T определяется рекурсивно либо как пустая структура (пустое дерево), либо как узел типа T с конечным числом древовидных структур этого же типа, называемых поддеревьями.

2.Как выделяется память под представление деревьев?

Способ выделения памяти под деревья определяется способом их представления в программе. С помощью матрицы или списка может быть реализована таблица связей с предками или связный список сыновей. Целесообразно использовать списки для упрощенной работы с данными, когда элементы требуется добавлять и удалять, т. е. выделять память под каждый элемент отдельно.

3. Какие стандартные операции возможны над деревьями?

Основные операции с деревьями: обход дерева, поиск по дереву, включение в дерево, исключение из дерева. Обход вершин дерева можно осуществить следующим образом:

- сверху вниз (префиксный обход)
- слева направо (инфиксный обход)
- снизу вверх (постфиксный обход)

4.Что такое дерево двоичного поиска?

Дерево двоичного поиска – дерево, в котором все левые потомки моложе предка, а все правые – старше. Это свойство называется характеристическим свойством дерева двоичного поиска и выполняется для любого узла, включая корень. С учетом этого свойства поиск узла в двоичном дереве поиска можно осуществить, двигаясь от корня в левое или правое поддерево в зависимости от значения ключа поддерева.

Вывод

Основным преимуществом деревьев является возможная высокая эффективность реализации основанных на нем алгоритмов поиска и сортировки. Также из дерева быстрее происходит удаление чем в файле, но при этом затрачивается больше памяти