****

|  |  |
| --- | --- |
|  | **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 О ЛАБОРАТОРНОЙ РАБОТЕ №6 ПО ТиСД**

Обработка деревьев

Группа ИУ7-34Б

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

|  |  |  |
| --- | --- | --- |
| Студент |  | Гареев Георгий Антонович |
| Преподаватель |  | Барышникова Марина Юрьевна |

## Описание условия задачи

Техническое задание

Построить двоичное дерево поиска, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Определить количество вершин дерева, содержащих слова, начинающиеся на указанную букву. Выделить эти вершины цветом. Сравнить время поиска начинающихся на указанную букву слов в дереве и в файле.

Исходные данные

Номер команды меню.

Описание задачи, реализуемой программой

Строит дерево по вводимым значениям и реализует базовые функции для работы с ним.

Способ обращения в программе

Пользователю выдается меню, в котором он должен выбрать номер функции 0-3:

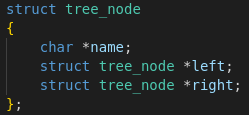
* 0 – выход из программы
* 1 – чтение слов из файла.
* 2 – добавление слова в дерево.
* 3 – удаление слова из дерева.
* 4 – поиск слова в дереве.
* 5 – вывод слов из дерева в инфиксном порядке.
* 6 – поиск слов, начинающихся с введенного символа.
* 7 – запись дерева в файл для последующей визуализации.
* 8 – сравнение времени поиска слов по символу в дереве и в файле в зависимости от кол-ва слов.

Описание возможных аварийных ситуаций и ошибок пользователя

1. Динамическая память не сможет выделится.
2. Введенный путь к файлу неправильный.
3. Попытка удалить элемент их пустого дерева.

Описание внутренних СД

Под вершину дерева выделяется структура следующего содержания:



Char \*name – указатель на слово (содержание вершины);

Struct tree\_node \*left(right) – указатель на левое (правое) поддерево.

## Описание алгоритма

При выборе 0 пункта меню (выход из программы) программа завершается с освобождением всей использованной динамической памятью.

При выборе 1 пункта меню пользователь должен ввести путь к читаемому файлу, после чего слова из файла добавляются пустое дерево, если файл нечитабелен или не существует – выдается сообщение об ошибке.

При выборе 2 пункта меню пользователь вводит слово, и оно добавляется в дерево, если такое уже существует, то ничего не происходит.

При выборе 3 пункта меню пользователь вводит слово, и оно удаляется из дерева, если такого слова нет, то ничего не происходит.

При выборе 4 пункта меню пользователь вводит слово, и в дереве происходит его поиск, в качестве результат выводится сообщение о его присутствии в дереве.

При выборе 5 пункта меню слова из дерева выводятся на экран в инфиксном порядке.

При выборе 6 пункта меню пользователь вводит символ, после чего происходит поиск слов, начинающихся с данного символа.

При выборе 7 пункта меню происходит запись дерева в файл ‘.gv’ для последующей визуализации с помощью плагина graphviz.

При выборе 8 пункта меню происходи сравнение времени поиска слов по символу в дереве и в файле в зависимости от кол-ва слов.

## Тесты

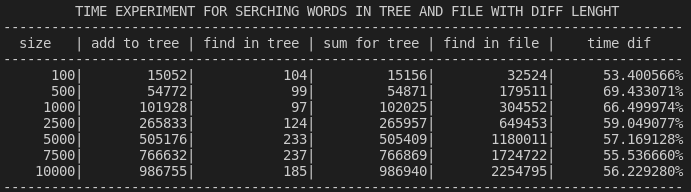
Позитивные

1. Вывод пустого дерева
2. Поиск в пустом дереве
3. Удаление и добавление различных ветвей
4. Сравнение времени

Негативные

1. Неправильный путь к файлу
2. Удаление из пустого дерева

## Оценка эффективности



Сверху представлена таблица времени (измеряется в тиках процессора) для поиска с помощью дерева и в самом файле при различном кол-ве слов. Как можем заметить, при поиске в ддп используется минимальное кол-во времени в сравнении с поиском в самом файле.

Однако для того, чтобы использовать само ддп, необходимо добавить слова в дерево, для чего требуется значительно больше времени, чем для поиска. Но даже так, суммарное время добавления в ддп и поиска в нем меньше, чем поиск в файле в среднем на 50-70%. Это связано с тем, что при поиске в файле мы считываем все слова и проверяем их первый символ на соответствие введенному, а в ддп алгоритм ищет только в тех поддеревьях, где слово может находится, а не обходит его полностью.

## Вывод

Для данной задачи, несмотря на кол-во слов и кол-во подходящих под критерий, реализация с помощью ддп значительно выигрывает по скорости работы, чем непосредственный поиск в неотсортированном файле.

Однако стоит заметить, что для ддп требуется динамическая память, и чем больше кол-во слов, тем больше памяти требуется для его хранения. При поиске в файле память требуется лишь под текущее читаемое слово, что делает этот способ значительно выгоднее, если доступная память критически мала.

## Ответы на контрольные вопросы

1. **Что такое дерево?**

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

1. **Как выделяется память под представление деревьев?**

Для каждой вершины динамически выделяется память, хранящая значение вершины и указатели на вершины-потомки, а ее адрес указывается в родительской вершине, что делает такую структуру ограниченной только допустимой памятью.

1. **Какие бывают типы деревьев?**

Дерево с базовым типом Т определяется рекурсивно либо как пустая структура (пустое дерево), либо как узел типа Т с конечным числом древовидных структур этого же типа, называемых поддеревьями.

Деревья могут быть сбалансированными (если кол-во вершин в левом и правом поддеревьях отличаются не больше, чем на 1). Могут быть бинарными (если кол-во поддеревьев у каждой вершины равно 2), они же в свою очередь могут быть ддп, если все значения в вершинах левого поддерева меньше значения родительской вершины, а правом – больше.

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

Добавление элемента в дерево, удаление элемента из дерева, поиск элемента дереве, обход дерева.

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

Дерево двоичного поиска – это такое 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.