



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 **«Деревья»**

Студент

Городский Юрий Николаевич

Группа

ИУ7 – 32Б

Оглавление

Условие задачи.....	3
Техническое задание.....	3
Функции программы.....	3
Аварийные ситуации:.....	4
Обращение к программе.....	4
Структура данных.....	4
Тесты.....	5
Замерный эксперимент.....	10
Контрольные вопросы.....	11
Вывод.....	12

Условие задачи

Цель работы – получить навыки применения двоичных деревьев

Задание:

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.

Техническое задание

В файловой системе каталог файлов организован в виде бинарного дерева.

Каждый узел обозначает файл, содержащий имя и атрибуты файла, в том числе и дату последнего обращения к файлу. Написать программу, которая обходит дерево и удаляет из него все файлы, последнее обращение к которым происходило до определенной даты. Вывести исходные и измененные деревья в виде дерева.

Сравнить время удаления в дереве, построенном по алфавиту, со временем перестроения дерева по дате обращения и удаления в нем.

Входные данные:

1. **Номер команды:** целое число в диапазоне от 0 до 9
2. **Элемент дерева:** имя файла и дата последнего изменения в секундах с 1970 года
3. **Прототип файла для удаления:** имя файла и дата последнего изменения в секундах с 1970 года

Выходные данные:

1. Состояние дерева на текущий момент
2. Данные замерного эксперимента в виде графика и файлов с данными

Функции программы

1. Ввести элемент дерева

2. Удалить узел дерева
3. Перестроить дерево по дате
4. Перестроить дерево по имени
5. Вывести состояние дерева
6. Обход дерева сверху вниз
7. Обход дерева снизу вверх
8. Обход дерева слева направо
9. Замерный эксперимент
10. Выйти

Аварийные ситуации:

1. Некорректный ввод команды (введено не число или число не находится в диапазоне от 0 до 9): сообщение «Неверная команда».
2. Введено не число при вводе даты узла или прототипа: сообщение «Ошибка ввода»
3. Не удалось открыть файл при записи замерного эксперимента: сообщение «Ошибка работы с файлом».

Обращение к программе

Запуск через терминал (./app.exe).

Структура данных

```
// Узел дерева
typedef struct tree_type
{
    char *name; // имя файла
    time_t date; // дата изменения
    struct tree_type *left; // Левый потомок
    struct tree_type *right; // Правый потомок
} tree_t;
```

Тесты

Таблица 1: Негативные тесты

Описание	Входные данные	Выходные данные
Некорректная команда (число)	10	«Неверная команда»
Некорректная команда (число)	-1	«Неверная команда»
Некорректная команда (не число)	a	«Неверная команда»
Неверно введена дата файла (не число)	1 abc.txt a	«Ошибка ввода»

Таблица 2: Позитивные тесты

Описание	Ввод	Вывод
Ввод узла	1 abcd.txt 12	«Операция завершена»

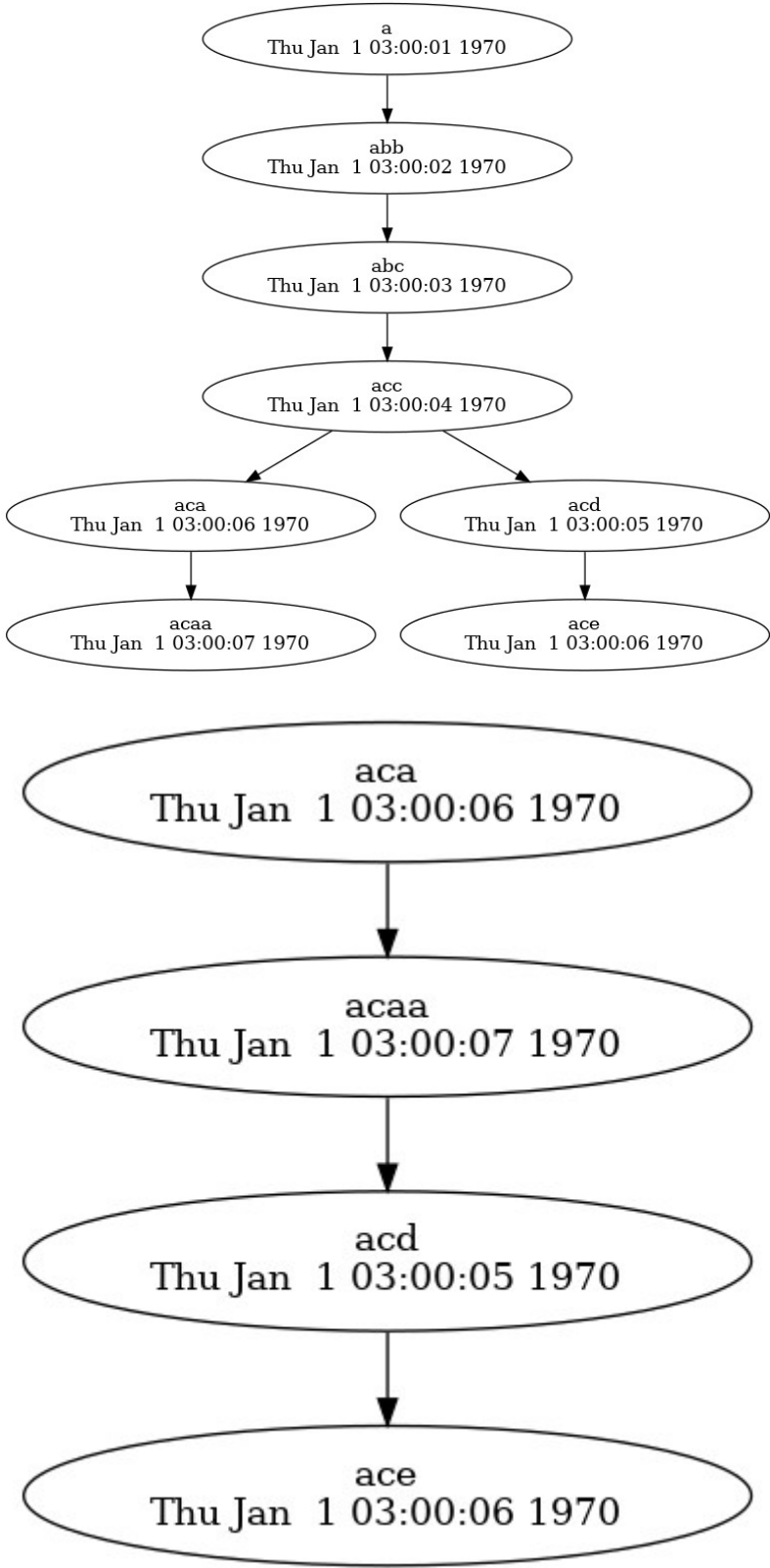
Таблица 3: Примеры работы

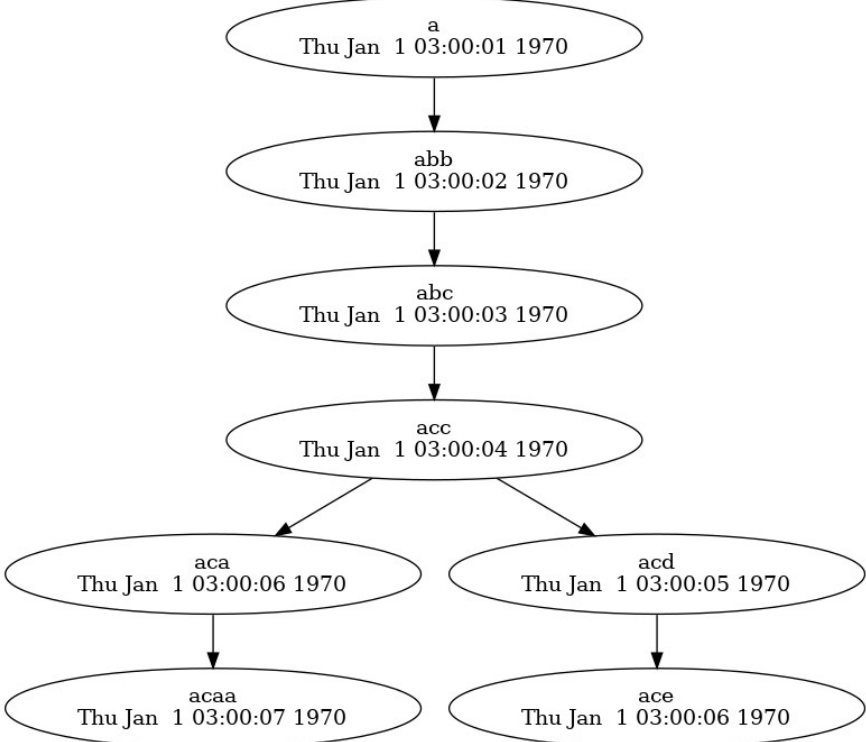
Описание	Ввод	Вывод
Ввод дерева	1 a 1 1 abb 2 1 abc 3 1 acc 4 1 acd 5 1 ace 6 1 aca 6 1 acaa 7 3	<pre>graph TD; a["a Thu Jan 1 03:00:01 1970"] --> abb["abb Thu Jan 1 03:00:02 1970"]; abb --> abc["abc Thu Jan 1 03:00:03 1970"]; abc --> acc["acc Thu Jan 1 03:00:04 1970"]; acc --> aca["aca Thu Jan 1 03:00:06 1970"]; acc --> acd["acd Thu Jan 1 03:00:05 1970"]; aca --> acaa["acaa Thu Jan 1 03:00:07 1970"]; acd --> ace["ace Thu Jan 1 03:00:06 1970"];</pre>

Перестроения	{Ввод дерева} 4 3 5 3	<pre> graph TD a1([a Thu Jan 1 03:00:01 1970]) --> abb1([abb Thu Jan 1 03:00:02 1970]) abb1 --> abc1([abc Thu Jan 1 03:00:03 1970]) abc1 --> acc1([acc Thu Jan 1 03:00:04 1970]) acc1 --> aca1([aca Thu Jan 1 03:00:06 1970]) aca1 --> acd1([acd Thu Jan 1 03:00:05 1970]) aca1 --> acaa1([acaa Thu Jan 1 03:00:07 1970]) acaa1 --> ace1([ace Thu Jan 1 03:00:06 1970]) a2([a Thu Jan 1 03:00:01 1970]) --> abb2([abb Thu Jan 1 03:00:02 1970]) abb2 --> abc2([abc Thu Jan 1 03:00:03 1970]) abc2 --> acc2([acc Thu Jan 1 03:00:04 1970]) acc2 --> aca2([aca Thu Jan 1 03:00:06 1970]) acc2 --> acd2([acd Thu Jan 1 03:00:05 1970]) aca2 --> acaa2([acaa Thu Jan 1 03:00:07 1970]) acd2 --> ace2([ace Thu Jan 1 03:00:06 1970]) </pre>
--------------	--------------------------------------	---

Удаление до
даты

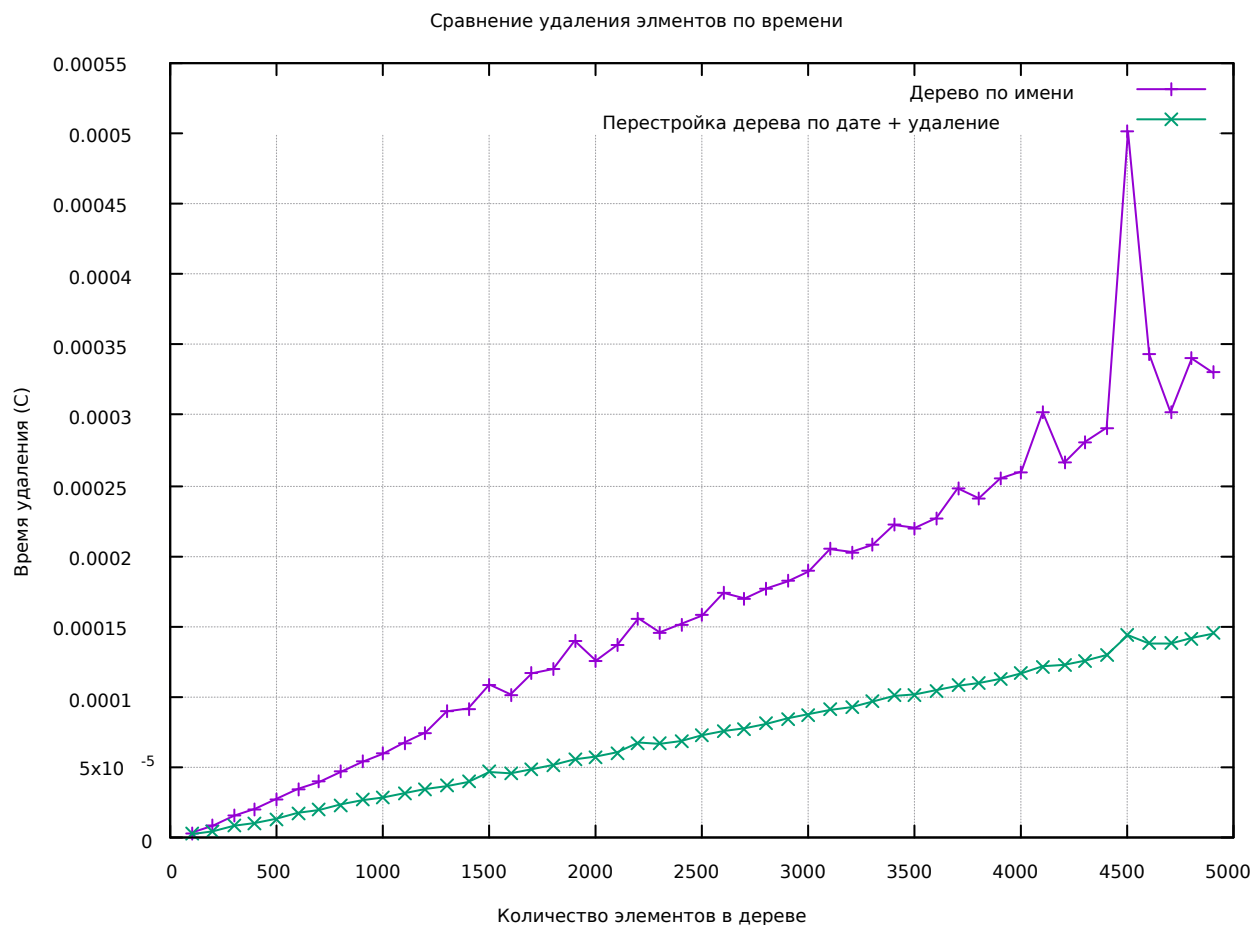
{Ввод
дерева}
3
2
abc
4
3



Обходы	{Ввод дерева} 3 6(сверху вниз) 7 (снизу вверх) 8 (слева направо)	 <pre> graph TD a([a Thu Jan 1 03:00:01 1970]) --> abb([abb Thu Jan 1 03:00:02 1970]) abb --> abc([abc Thu Jan 1 03:00:03 1970]) abc --> acc([acc Thu Jan 1 03:00:04 1970]) acc --> aca([aca Thu Jan 1 03:00:06 1970]) acc --> acd([acd Thu Jan 1 03:00:05 1970]) aca --> aaaa([aaaa Thu Jan 1 03:00:07 1970]) acd --> ace([ace Thu Jan 1 03:00:06 1970]) </pre> <p> a abb abc acc aca aaaa acd ace Операция завершена ... aaaa aca ace acd acc abc abb a Операция завершена ... a abb abc aca aaaa acc acd ace Операция завершена </p>
--------	--	--

Замерный эксперимент

Замеры проводились для деревьев с числом элементов от 100 до 5000 с шагом в 100 элементов. Замер с каждым числом повторялся 25 раз и бралось среднее значение. При замере с перестроением время перестроения тоже учитывалось. При генерации дерева для обеспечения случайности имен, имена файлов имеют три буквы из английского алфавита, что обеспечивает число возможных комбинаций гораздо большее чем число элементов в дереве. Чтобы при увеличении числа элементов удалялось примерно одна и та же часть узлов от общего количества время изменения имеет значение 0 или 1, что позволяет при каждом эксперименте удалить примерно половину элементов.



Как видно, перестроение дерева по дате и удаление из нового дерева оказывается быстрее удаления из дерева, построенного по имени.

Контрольные вопросы

1. Что такое дерево?

Дерево – нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один к 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. Можно использовать списки для упрощенной работы с данными, когда элементы требуется добавлять и удалять.

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

Главным преимуществом деревьев является возможность реализовать быстрые алгоритмы связанные с сравнением элементов. Однако такое преимущество реализуется только при построении дерева по полю, по которому ведется операция.