



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 Тузов Даниил Александрович

Группа ИУ7 – 32Б

Преподаватель Барышникова Марина Юрьевна  
Силантьева Александра Васильевна

## **Оглавление**

<b><u>ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ.....</u></b>	<b><u>3</u></b>
<b><u>ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ.....</u></b>	<b><u>3</u></b>
<b><u>ОПИСАНИЕ СТРУКТУР ДАННЫХ.....</u></b>	<b><u>4</u></b>
<b><u>ОПИСАНИЕ АЛГОРИТМА.....</u></b>	<b><u>5</u></b>
<b><u>ОПИСАНИЕ ФУНКЦИЙ.....</u></b>	<b><u>5</u></b>
<b><u>НАБОР ТЕСТОВ.....</u></b>	<b><u>7</u></b>
<b><u>ЗАМЕРЫ ВРЕМЕНИ БАЗОВЫХ ОПЕРАЦИЯ.....</u></b>	<b><u>9</u></b>
<b><u>ЗАМЕРЫ ВРЕМЕНИ ВЫПОЛНЕНИЯ ЗАДАЧИ.....</u></b>	<b><u>9</u></b>
<b><u>ВЫВОД.....</u></b>	<b><u>10</u></b>
<b><u>ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ.....</u></b>	<b><u>10</u></b>

## 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.

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.

## ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

### Входные данные:

Пользователю предлагается ввести число, отвечающее за действие, узел дерева (файл и его атрибуты), имя файла для вывода дерева

### Выходные данные:

DOT-файлы с полученными деревьями, время удаления, поиска, вставки и обходов, результаты обходов

### Меню:

- 1 — Вставка в дерево
- 2 — Удаление из дерева
- 3 — Вывод дерева в dot-файл
- 4 — Удаление всех файлов из дерева, дата обращения к которым меньше, чем введенная
- 5 — pre-order обход
- 6 — in-order обход
- 7 — post-order обход
- 0 — завершение программы

### Описание программы:

Программа имитирует работу с деревьями (вставку в дерево, поиск в дереве, удаление из дерева, обходы дерева)

### **Способ обращения к программе:**

Взаимодействие через консоль.

### **Аварийные ситуации:**

1. Ошибка ввода  
Код ошибки — 1
2. Ошибка памяти  
Код ошибки — 2
3. Ошибка удаления из дерева  
Код ошибки — 3
4. Ошибка вставки в дерево  
Код ошибки — 4
5. Ошибка файла  
Код ошибки — 5

## **ОПИСАНИЕ СТРУКТУР ДАННЫХ**

### Описание узла дерева

filename – название файла

access – доступ к файлу

owner – владелец файла

size – размер файла

year – год последнего обновления файла

month – месяц последнего обновления файла

day – день последнего обновления файла

hour – час последнего обновления файла

minutes – минута последнего обновления файла

left – указатель на левое поддерево (меньшие элементы, чем в текущем)

right – указатель на правое поддерево (большие элементы, чем в текущем)

```
typedef struct tree_type
```

```
{  
    char *filename;  
    char *access;  
    char *owner;  
    char *size;  
    int year;  
    int month;  
};
```

```
int day;
int hour;
int minutes;
struct tree_type *left;
struct tree_type *right;
} tree_t;
```

## ОПИСАНИЕ АЛГОРИТМА

1. Пользователь выбирает действие
2. Согласно выбранному действию вызывается соответствующая функция
  1. При вставке, удалении и поиске рекурсивно спускаемся в левое или правое поддерево (в зависимости от сравнения)
  2. При выборе варианта обхода печатается соответствующий обход
3. Выполняется логика этой функции
4. Если во время выполнения функции возникла ошибка, то выводится сообщение, поясняющее эту ошибку. Программа аварийно завершается
5. Если ошибки не возникло, возвращаемся к 1 пункту

## ОПИСАНИЕ ФУНКЦИЙ

`void description();` - функция выводит описание программы

`void menu();` - функция выводит меню пользователю

`tree_t *input_el_for_delete(int act);` - функция предлагает пользователю ввести элемент, который он хочет удалить при заданном варианте сортировки act в дереве. Возвращает элемент

`tree_t *tree_create(char *filename, char *access, char *owner, char *size, int year, int month, int day, int hour, int minute);` - функция создает узел дерева по параметрам, которые хранятся в узле и возвращает его

`void tree_free(tree_t *pt);` - функция очищает память из-под узла

`int cmp_filename(tree_t *pl, tree_t *pr);` - функция сравнивает два узла по имени файла. Возвращает результат сравнения

`int cmp_date(tree_t *pl, tree_t *pr);` - функция сравнивает два узла по дате последнего обращения. Возвращает результат сравнения

`void tree_destroy(tree_t *pt);` - функция уничтожает объект-дерево

`void tree_print(FILE *f, tree_t *pt);` - функция печатает объект-дерево в файл.

`void tree_print_dot(FILE *f, tree_t *pt, const char *name);` - функция печатает объект-дерево с именем name в dot-файл

`void pre_order(tree_t *pt);` - pre-order обход дерева

`void in_order(tree_t *pt);` - in-order обход дерева

`void post_order(tree_t *pt);` - post-order обход дерева

`tree_t *tree_find(tree_t *pt, tree_t *el, int (*comp)(tree_t *, tree_t *));` - функция поиска в дереве заданного элемента el. Возвращает найденный элемент или NULL

`tree_t *tree_insert(tree_t *pt, tree_t *el, int (*comp)(tree_t *, tree_t *));` - функция вставки в дерево заданного элемента el. Возвращает указатель на новый корень

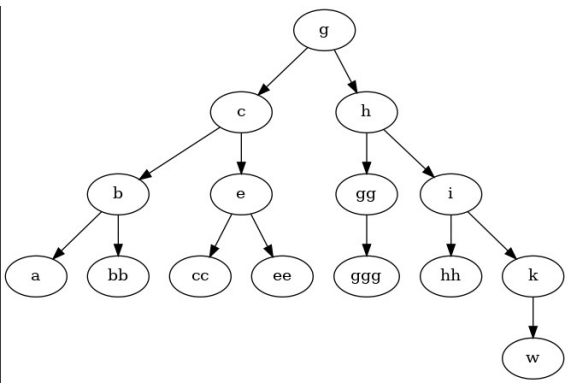
`static tree_t *del_el_tree(tree_t *pt);` - функция удаления из дерева корня(!). Возвращает указатель на новый корень

`tree_t *tree_delete(tree_t *pt, tree_t *el, int (*comp)(tree_t *, tree_t *));` - функция удаления из дерева заданного элемента el. Возвращает указатель на новый корень.

`tree_t *tree_delete_task(tree_t *pt, tree_t *el, int (*comp)(tree_t *, tree_t *));` - функция удаляет из дерева, построенного в алфавитном порядке, все элементы, дата обращения к которым меньше, чем у el. Возвращает новую голову

`tree_t *tree_delete_task_sorted(tree_t *pt, tree_t *el, int (*comp)(tree_t *, tree_t *));` - функция удаляет из дерева, построенного по дате обращения к файлам, все элементы, дата обращения к которым меньше, чем у el. Возвращает новую голову

## НАБОР ТЕСТОВ

№	Описание теста	Ввод пользователя	Ответ программы
1	Выбор действия	a	Некорректный ввод действия
2	Выбор действия	-1	Некорректный ввод действия
3	Выбор действия	8	Некорректный ввод действия
4	Выбор действия	0	Завершение программы
5	Выбор действия	1	Вставка в дерево
6	Выбор действия	2	Удаление из дерева
7	Выбор действия	3	Экспорт дерева в dot-файл
8	Выбор действия	4	Выполнение задания
9	Вставка элемента	Существующий элемент	<p>Ошибка вставки</p>  <pre> graph TD     g((g)) --&gt; c((c))     g --&gt; h((h))     c --&gt; b((b))     c --&gt; e((e))     b --&gt; a((a))     b --&gt; bb((bb))     e --&gt; cc((cc))     e --&gt; ee((ee))     h --&gt; gg((gg))     h --&gt; i((i))     gg --&gt; ggg((ggg))     i --&gt; hh((hh))     i --&gt; k((k))     k --&gt; w((w))         </pre>
10	Вставка элемента	w w w w 2023 11 28 10 39	
11	Удаление элемента	Не существующий элемент	Ошибка удаления

12	Удаление элемента	b	
13	Экспорт в файл	a.dot	Успешно экспортировано в файл
14	Выполнение задания	2023 11 11 10 9	Успешно удалены все файлы, обращение к которым было раньше 10:09 11 ноября 2023 года
15	Ввод элемента	Ввод буквы вместо года, месяца, дня, часа или минуты	Ошибка ввода
16	Ввод элемента	A b c d 1 2 3 4 5	Успешно введен элемент
17	Ввод варианта сортировки	1	
18	Ввод варианта сортировки	2	
19	Ввод варианта сортировки	Не 1 и не 2	Ошибочный вариант сортировки
20	pre-order	Граф из п.18	G c b a bb e cc ee h gg ggg i hh k



21	in-order	Граф из п.18	A b bb c cc e ee g ggg gg h hh i k
22	post-order	Граф из п.18	A bb h cc ee e c ggg gg hh k i h g

### ЗАМЕРЫ ВРЕМЕНИ БАЗОВЫХ ОПЕРАЦИЙ

В ходе эксперимента для получения результатов проводилось 2000 замеров по времени. Усредненный результат представлен в таблице

Результаты представлены в **микросекундах**, у каждого узла есть только 1 сын (правостороннее дерево)

Высота дерева	Добавление	Удаление
1	5	8
30	34	46
60	45	57
100	90	115

Результаты представлены **для тех же данных**, но дерево двоичного поиска

Высота дерева	Добавление	Удаление
1	5	8
6	9	11
7	10	12
7	10	12

### ЗАМЕРЫ ВРЕМЕНИ ВЫПОЛНЕНИЯ ЗАДАЧИ

В ходе эксперимента для получения результатов проводилось 2000 замеров по времени. Усредненный результат представлен в таблице

Результаты представлены в **микросекундах**, в качестве дерева используется идеально сбалансированное дерево с высотой 3

Сортировка по алфавиту	Сортировка по времени обращения
18	11

## ВЫВОД

В ходе лабораторной работы я познакомился с двоичным деревом поиска. Я выяснил:

- Вставка в двоичное дерево работает на 33% быстрее, чем удаление.
  - Связано это с тем, что при вставке мы просто спускаемся до листа, в который записываем элемент, а при удалении нам необходимо найти вершину
- Время вставки и удаления прямопропорционально зависит от высоты дерева и не зависит от степени ветвления
- При сортировке по дате обращения быстрее происходит удаление из дерева, чем при сортировке файлов в алфавитном порядке (порядка 40%)

## ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

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

Дерево — связный ациклический граф. В памяти деревья можно представить в виде связей с предками (указатели).

### 2. Какие бывают типы деревьев?

Бинарные (2 потомка), двоичного поиска, сбалансированные (высота каждого листа отличается не больше, чем на 1), avl, splay, red-black, декартово дерево (а еще хвойные и лиственные).

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

Вставка в дерево, удаление из дерева, поиск в дереве, pre-order обход, in-order обход, post-order обход

### 4. Что такое дерево двоичного поиска?

Деревом двоичного поиска называются двоичные деревья, у которых для каждого узла выполняется:

- В левом поддереве лежат элементы, меньшие, чем в текущем узле
- В правом поддереве лежат элементы, большие, чем в текущем узле