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»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«Обработка деревьев»

по курсу «Типы и структуры данных»

Студент: Шимшир Эмирджан Османович

Группа: ИУ7-33Б

Студент _____ Шимшир Э.О.

подпись, дата *фамилия, и.о.*

Преподаватель _____ Барышникова М.Ю.

подпись, дата *фамилия, и.о.*

Оценка _____

Условие задачи

Вариант 3

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.

Построить бинарное дерево, 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.

Техническое задание

Входные данные

Для корректной работы программы нужно заполнить с клавиатуры или из файла бинарное дерево

Выходные данные

Вывод дерева на экран и результатов измерения времени при удалении элементов из дерева и из файла

Задачи, реализуемые программой

1. Добавление новых слов в бинарное дерево
2. Вывод дерева на экран
3. Поиск слов в бинарном дереве и их вывод на экран
4. Удаление слов из бинарного дерева и из файла

Допущения

Слова необходимо вводить на английском языке

Длина вводимых слов должна быть не более 20 символов

Пробелы и знаки препинания не допускаются

При поиске необходимо ввести 1 символ

Описание внутренних структур данных

Структура для хранения узла бинарного дерева:

```
typedef struct node_t
{
    char word[MAX_STR_LEN + 1];
    int height;

    struct node_t *left;
    struct node_t *right;
} node_t;
```

char word[MAX_STR_LEN + 1]; - хранящееся слово
int height; - глубина элемента
*struct node *left;* - указатель на левого потомка
*struct node *right;* - указатель на правого потомка

Память под описанные выше структуры данных выделяется динамически

Описание меню и функций программы

```
emir_shimshir@mbp-emir ~/D/g/B/lab_06 (lab_06)> ./app.exe
Программа для работы с бинарным деревом
Шимшир Эмирджан ИУ7-33Б В-24

(Слова необходимо вводить на английском языке)
(Длина вводимых слов должна быть не более 20 символов)
(Пробелы и знаки препинания не допускаются)
(При вводе буквы необходимо вводить 1 символ)


Пункты меню:
1 - Добавить в бинарное дерево слова из файла
2 - Добавить в бинарное дерево слово, введенное с клавиатуры
3 - Вывести бинарное дерево в консоли
4 - Вывести бинарное дерево на картинке
5 - Вывести все слова в дереве, начинающиеся на указанную букву
6 - Удалить из дерева слово, введенное с клавиатуры
7 - Удалить из дерева и файла все слова, начинающиеся на указанную букву и сравнить время удаления слов
0 - выход из программы
Введите пункт меню:
```

1. Пользователь выбирает пункт меню

2. Пользователь добавляет слова в дерево (пункты 1, 2)

3. Пользователь выводит дерево на экран (пункты 3, 4)

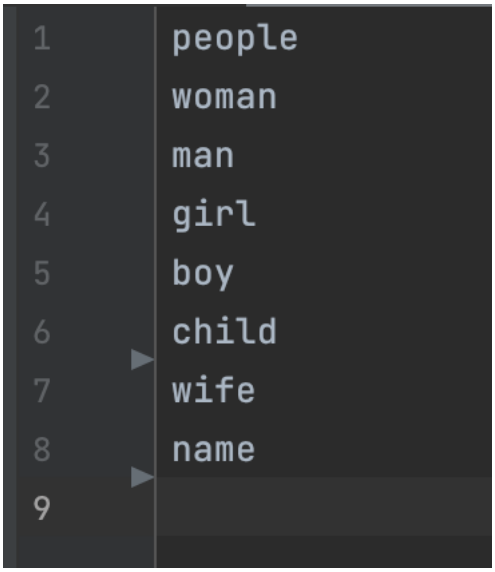
4. Пользователю выводятся слова в дереве на указанную букву (пункт 5)

5. Пользователь удаляет заданное слово (пункт 6)

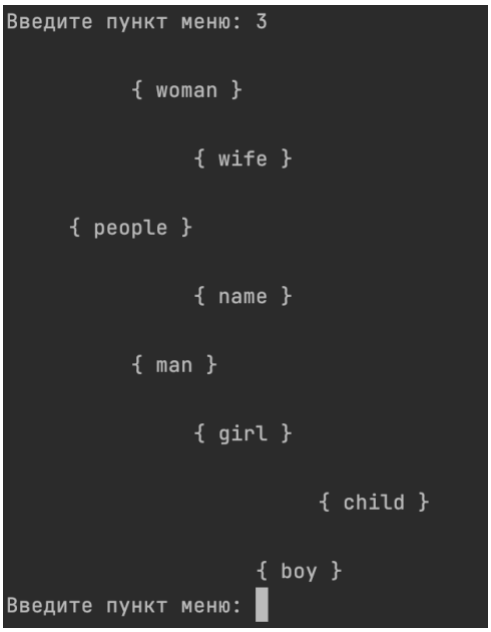
6. Пользователь удаляет слова в дереве и файле на указанную букву, также ему доступны результаты измерения времени (пункт 7)

Пример вывода бинарного дерева на экран

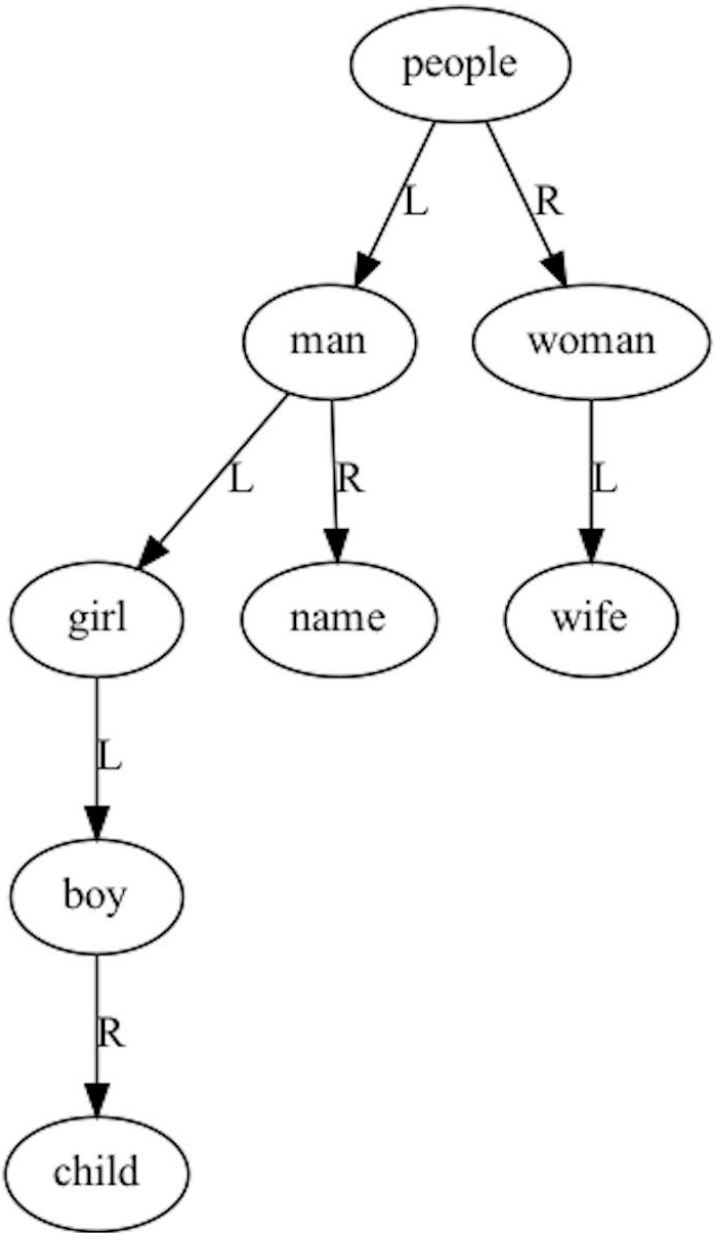
Исходный файл с данными:



Вывод бинарного дерева в консоль:



Вывод бинарного дерева в виде *.png* картинки (graphviz):



Описание функций

```c
/*
* Функция очищает дерево
*
* Принимает указатель на корень дерева
*/
void free_tree(node_t *tree);
```

```c
/*
* Функция создает узел дерева
*
* Принимает слово и глубину, возвращает указатель на новый узел
*/
node_t *tree_init_node(char *word, int height);
```

```c
/*
* Функция добавляет новый узел в дерево
*
* Принимает указатель на корень, слово, указатель на глубину, указатель на
количество сравнений
*/
node_t *tree_insert_node(node_t *tree, char *word, int *height, int *comp);
```

```c
/*
* Функция печатает дерево на экран
*
* Принимает указатель на корень и количество отступов
*/
void print_tree(node_t *tree, int place);
```

```c
/*
* Функция удаляет узел из дерева
*
* Принимает указатель на указатель на корень, слово и функцию для поиска
*/
static int delete_node(node_t **tree, char *word, search_t search)
```

Описание алгоритма и исследование полученных результатов

Описание алгоритма

В данной работе я использовал 3 вида обхода деревьев, префиксный для записи в формат *graphviz*, постфиксный для очищения дерева, и инфиксный для вывода результатов поиска. Добавление элемента я реализовал через рекурсию, а поиск элементов по слову целиком и первой букве через цикл. Алгоритм удаления подробно задокументирован в исходном коде.

Так как дерево не балансируется, сложность поиска в нем зависит от данных, а именно от порядка добавления элементов в дереве. В лучшем случае сложность поиска может оказаться **O(log2(N))** (в случае, если дерево идеально сбалансированно), а в худшем **O(N)** (в случае добавления упорядоченных элементов, так дерево выродится в односвязный список)

Полученные результаты

Результаты измерения времени удаления слов из дерева и файла при различном количестве исходных элементов. Для каждого случая проводилось 5 экспериментов. При проведении экспериментов программа была скомпилирована без оптимизаций (**-O0**), внешние задачи отсутствовали.

Размер узла дерева в байтах: 48

Время в микросекундах

Количество элементов	**Дерево**		**файл**
	Хорошо сбалансировано	**Вырождается в список**	
10	1	1	62
50	1	2	243
100	2	4	420
500	7	11	2331
1000	17	42	5432

Выводы из таблицы измерений:

Мы можем заметить, что время удаления слов из файла растет линейно с увеличением количества элементов, это согласуется с теорией так как при удалении слов из файла я считываю все слова из файла и записываю их в массив. Также мы видим, что удаление в хорошо сбалансированном дереве быстрее удаления в дереве, вырожденном в списке, особенно при больших размерностях, что также согласуется с теорией.

Тестирование

Позитивные тесты

№	Входные данные	Действия и выходные данные	Результат
1	пункт = 0	Завершение программы	Код возврата - 0
2	пункт = 1 Ввод валидного файла	Добавление данных в дерево	Ожидание следующего пункта
3	Ключ = 2 Ввод валидного слова	Добавление данных в дерево	Ожидание следующего пункта
4	Ключ = 3	Вывод дерева в консоли	Ожидание следующего пункта
5	Ключ = 4	Вывод дерева в .png	Ожидание следующего пункта
6	Ключ = 5 Ввод валидной буквы	Вывод всех слов на заданную букву	Ожидание следующего пункта
7	Ключ = 6 Ввод валидного слова	Удаление слова из дерева	Ожидание следующего пункта
8	Ключ = 7 Ввод валидного файла и буквы	Из матрицы и файла удаляются все слова на заданную букву	Ожидание следующего пункта

Негативные тесты

№	Входные данные	Выходные данные	Результат
1	Ключ = 3 Пустое дерево	Ошибка, пустое дерево	Ожидание следующего пункта
2	Ключ = 4 Пустое дерево	Ошибка, пустое дерево	Ожидание следующего пункта
3	Ключ = 5 Пустое дерево	Ошибка, пустое дерево	Ожидание следующего пункта
4	Ключ = 6 Пустое дерево	Ошибка, пустое дерево	Ожидание следующего пункта
5	Ключ = 8	Ошибка, ввода пункта меню	Ожидание следующего пункта
6	Ключ = 1 Ввод несуществующего файла	Ошибка, неверно введено имя файла	Ожидание следующего пункта

Контрольные вопросы

1. Что такое дерево?

Дерево — структура данных (рекурсивная), используемая для представления иерархических связей (один ко многим)

2. Как выделяется память под представление деревьев?

Память выделяется как для связанного списка, то есть под каждый узел отдельно.

3. Какие бывают типы деревьев?

N-арное дерево, сбалансированное дерево, бинарное дерево, бинарное дерево поиска, дерево AVL, красно-чёрное дерево

4. Какие стандартные операции возможны над деревьями?

Поиск по дереву, обход дерева, добавление элемента в дерево, удаление элемента из дерева

5. Что такое дерево двоичного поиска?

Двоичное дерево поиска (ДДП) — двоичное дерево
В нем для каждого узла выполняется условие, что правый потомок больше или равен родителю, а левый потомок строго меньше родителя (или наоборот)

Вывод

Мною были исследованы различные операции для работы с бинарными деревьями, обходы, поиски, методы вывода, добавления и удаления элементов. Также я исследовал время работы данных алгоритмов и убедился в том, что сложность алгоритмов несбалансированных деревьев сильно зависит структуры дерева.