



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 ПО ДИСЦИПЛИНЕ: ТИПЫ И СТРУКТУРЫ ДАННЫХ

“Обработка деревьев”

Студент **Зернов Георгий Павлович**

Группа **ИУ7-34Б**

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

Студент _____ **Зернов Г.П.**

Преподаватель _____ **Никульшина Т.А.**

Оглавление

Условие задачи	3
Техническое задание	4
Реализуемая в программе задача	4
Входные данные	4
Выходные данные	5
Способ обращения к программе	5
Возможные аварийные ситуации или ошибки пользователя	5
Внутренние структуры и типы данных	6
Алгоритм программы	6
Функции программы	8
Тесты	9
Анализ эффективности	12
Вывод	14
Контрольные вопросы	15

Условие задачи

Построить бинарное дерево поиска, в вершинах которого находятся слова из текстового файла (предполагается, что исходный файл не содержит повторяющихся слов). 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

Реализуемая в программе задача

Цель работы: получить навыки применения двоичных деревьев, реализовать основные операции над деревьями: обход деревьев, включение, исключение и поиск узлов.

Входные данные

В случае выбора опции в меню целое число, соответствующее одной из указанных опций:

- 1 – Загрузить дерево из файла
- 2 – Добавить элемент в дерево
- 3 – Найти элемент в дереве
- 4 – Удалить элемент из дерева
- 5 – Удалить из дерева элементы, начинающиеся на заданную букву
- 6 – Вывести дерево в файл
- 7 – Анализ эффективности
- 8 – Выход

В случае загрузки дерева из файла:

- Имя файла – строка.

В случае вывода дерева в файл:

- Имя файла – строка.

В случае добавления элемента в дерево:

- Значение элемента – строка.

В случае поиска или удаление из дерева:

- Значение обрабатываемого элемента – строка.

В случае удаления из дерева элементов, начинающихся на заданную букву:

- Начальная буква – непробельный символ.

Выходные данные

1. В случае поиска элемента в дереве – сообщение о наличии/отсутствии элемента в дереве.
2. Вывести дерево в файл – файл содержащий дерево в dot формате.
3. Анализ эффективности – выведенные на экран результаты измерения времени обхода дерева для сбалансированного и вырожденного дерева и удаления слов, начинающихся на заданную букву в файле и вырожденном и сбалансированном дереве.

Способ обращения к программе

Чтобы обратиться к программе, пользователю необходимо запустить файл app.exe.

Возможные аварийные ситуации или ошибки пользователя

- Введено невалидное значение опции – вывод на экран информирующего сообщения о неверном вводе и предоставление пользователю возможности повторно ввести значение.
- Введено невалидное имя файла - вывод на экран информирующего сообщения о неверном вводе и предоставление пользователю возможности повторно ввести имя файла.
- Введено невалидное значение добавляемого элемента – вывод на экран информирующего сообщения о неверном вводе и предоставление пользователю возможности повторно ввести значение.
- Введено невалидное значение символа, по которому происходит удаление - вывод на экран информирующего сообщения о неверном

вводе и предоставление пользователю возможности повторно ввести значение.

- Ввод из файла, который невозможно открыть на чтение или содержит невалидную информацию – вывод информирующего сообщения.
- Вывод в файл, который невозможно открыть на запись – вывод информирующего сообщения.
- Попытка добавления уже существующего узла в дерево – вывод соответствующего сообщения об ошибке.
- Нехватка памяти при её выделении – соответствующее сообщение об ошибке и остановка текущего процесса с возвратом программы в состояние до его начала.

Внутренние структуры и типы данных

Структура, представляющее тип данных “бинарное дерево поиска”.

- `struct tree_node_t *left` – указатель на корень левого поддерева.
- `struct tree_node_t *right` – указатель на корень правого поддерева.
- `char *data` – указатель на строку (значение узла).

Листинг структуры:

```
typedef struct tree_node_t
{
    struct tree_node_t *left;
    struct tree_node_t *right;
    char *data;
} tree_node_t;
```

Алгоритм программы

- Общий алгоритм программы:

1. Запуск основного цикла программы.

2. Вывод меню.
3. Запрос у пользователя опции для выполнения.
4. Выполнение выбранной опции.
5. Переход к новой итерации цикла или завершение основного цикла при соответствующем выборе пользователя.

- Алгоритм добавления узла в дерево:

1. Если дерево пустое – возврат узла.
2. Иначе – если текущий элемент меньше вставляемого – рекурсивный запуск алгоритма на правом поддереве, а если больше – на левом.

- Алгоритм удаления элемента из дерева:

1. Если дерево пустое – возврат дерева.
2. Иначе – если текущий элемент меньше искомого – рекурсивный запуск алгоритма на правом поддереве, а если больше – на левом.
3. Иначе если и правое и левое поддерево не пусты – замещаем значение текущего узла, значением минимального узла правого поддерева и удаляем этот узел.
4. Иначе если одно из поддеревьев не пусто заменяем текущий узел этим поддеревом.
5. Иначе удаляем текущий узел.

- Алгоритм поиска элемента в дереве:

1. Если элемент искомый – возврат элемента.
2. Иначе – если текущий элемент меньше искомого – рекурсивный запуск алгоритма на правом поддереве, а если больше – на левом.

- Алгоритм удаления из дерева по первой букве:

1. Если дерево пустое – завершаем работу.

2. Если первая буква текущего элемента больше необходимой – рекурсивный запуск алгоритма на левом поддереве, если меньше – на правом, иначе – на обоих.
3. Если первая буква текущего элемента равна необходимой – удаление узла.

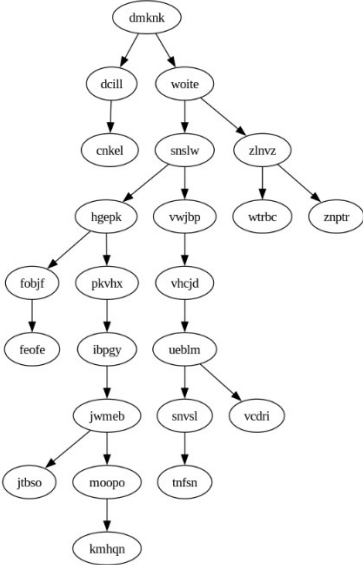
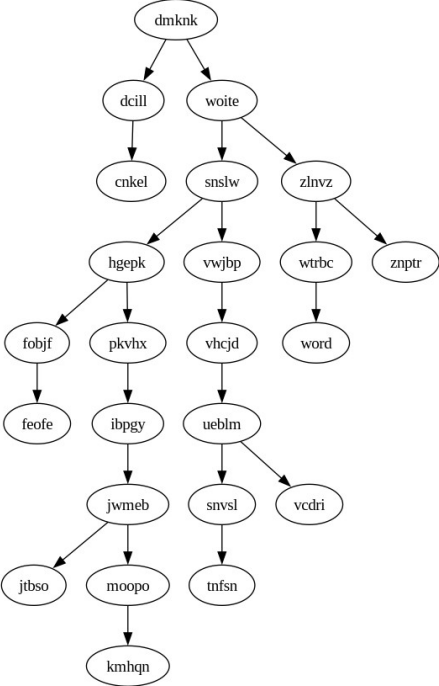
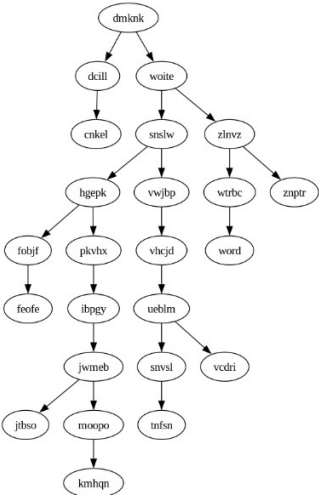
Функции программы

- `tree_node_t* create_node(char *data)` – создаёт узел по содержимому, возвращает указатель на узел.
- `void destroy_tree(tree_node_t *tree)` – освобождает выделенную под дерево память.
- `tree_node_t *minimum(tree_node_t *tree)` – возвращает указатель на минимальный узел в поддереве.
- `tree_node_t *maximum(tree_node_t *tree)` – возвращает указатель на максимальный узел в поддереве.
- `tree_node_t* insert(tree_node_t *tree, tree_node_t *node, int cmp(void *, void *))` – добавляет в бинарное дерево узел по компаратору, возвращает указатель на дерево с добавленным узлом.
- `tree_node_t *delete(tree_node_t *tree, char *data, int cmp(void *, void *))` – удаляет узел из дерева по значению и компаратору, возвращает указатель на изменённое дерево.
- `tree_node_t *search(tree_node_t *tree, void *data, int cmp(void *, void *))` – ищет узел в дереве по значению и компаратору, возвращает указатель на найденный узел.
- `void apply_pre(tree_node_t *tree, void f(tree_node_t*, void*), void *arg)` – применяет функцию к дереву, используя префиксный обход.
- `void apply_in(tree_node_t *tree, void f(tree_node_t*, void*), void *arg)` – применяет функцию к дереву, используя инфиксный обход.

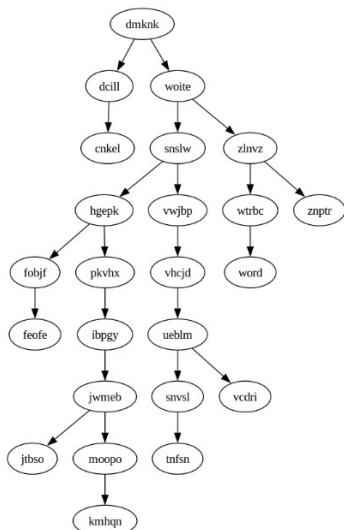
- `void apply_post(tree_node_t *tree, void f(tree_node_t*, void*), void *arg)` – применяет функцию к дереву, используя постфиксный обход.
- `void to_dot(tree_node_t *tree, void *param)` – экспортирует поддереву в dot формат.
- `void export_to_dot(FILE *file, const char *tree_name, tree_node_t *tree)` – экспортирует дерево в dot формат.
- `void delete_by_first_letter(tree_node_t **tree, char smb, int cmp(void *, void *))` – удаляет из дерева узлы по первому символу значения и компаратору.
- `int time_measurement(void)` – проводит анализ времени.
- `void print_menu(void)` – выводит меню.
- `int main(void)` – выполняет основной алгоритм программы.

Тесты

Тестовый случай	Ожидаемый результат
Введена невалидная опция	Соответствующее сообщение об ошибке, возможность снова ввести требуемое значение.
Введено невалидное имя файла ввода	Соответствующее сообщение об ошибке, возможность снова ввести имя файла.
Чтение из файла ввода невозможно	Соответствующее сообщение об ошибке.
Введено невалидное имя файла вывода	Соответствующее сообщение об ошибке, возможность снова ввести имя файла.
Запись в файл вывода невозможна	Соответствующее сообщение об ошибке.
Введено невалидное слово	Соответствующее сообщение об ошибке, возможность снова ввести имя.
Введена невалидная буква	Соответствующее сообщение об ошибке, возможность снова ввести букву.
Попытка добавить в дерево уже существующий элемент.	Соответствующее сообщение об ошибке.
Попытка удалить несуществующий	Соответствующее сообщение об ошибке.

элемент.	
Открытие файла с последовательностью слов	Успешное открытие файла
Вывод дерева в файл	Успешный вывод дерева в dot формате
<p>Добавление слова в дерево</p> <p>Слово: word</p> <p>Дерево:</p> 	<p>Полученное дерево:</p> 
<p>Поиск слова в дереве</p> <p>Слово: word</p> <p>Дерево:</p> 	Слово найдено
<p>Поиск слова в дереве</p> <p>Слово: notaword</p>	Слово не найдено

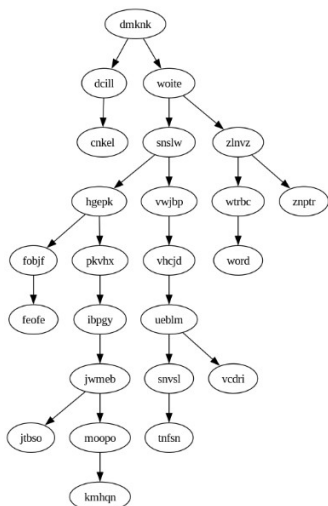
Дерево:



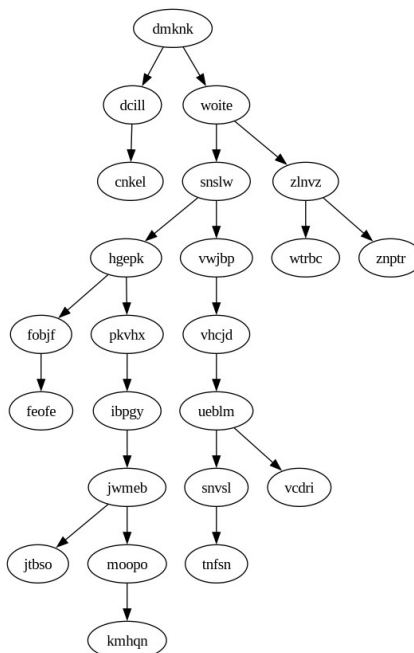
Удаление слова из дерева

Слово: word

Дерево:



Полученное дерево:

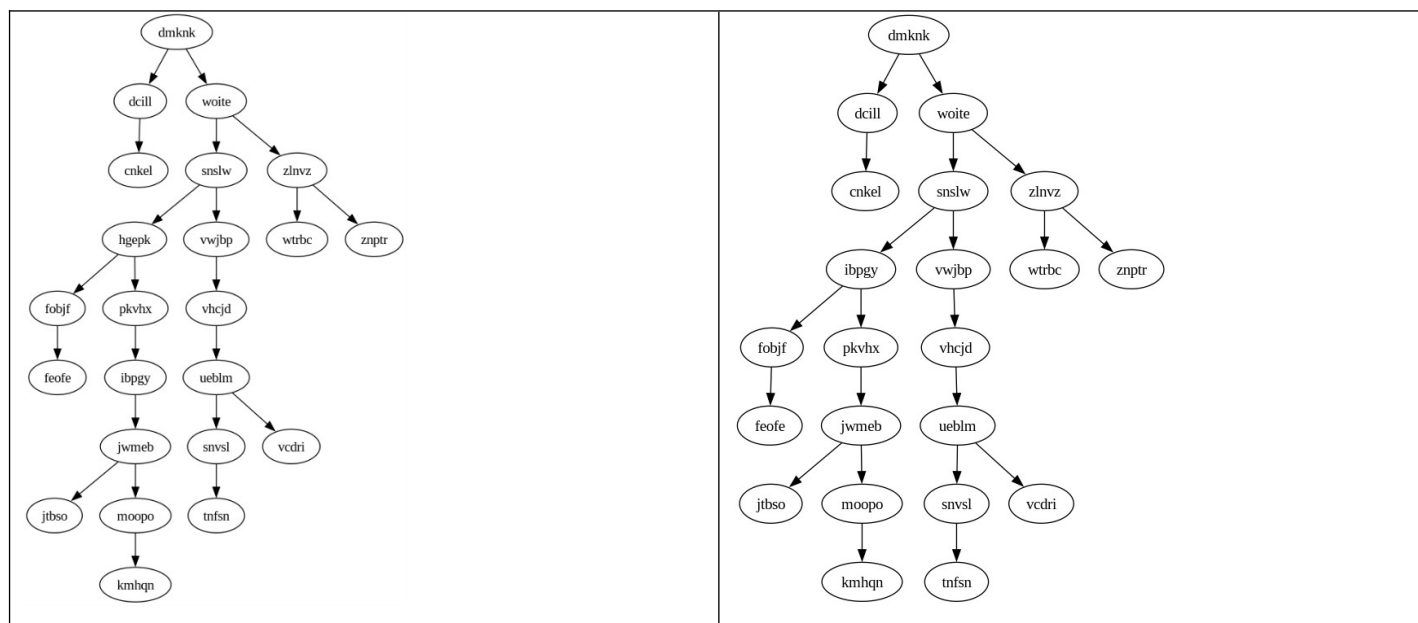


Удаление слов по первой букве из дерева

Буква: h

Дерево:

Полученное дерево:



Анализ эффективности

Производится на файлах, содержащих слова длины 5, состоящих из прописных букв латинского алфавита. Для каждого размера файла производится 1000 замеров и считается среднее.

Сбалансированное дерево:

Размер файла (слов)	Обход дерева (нс)	Удаление из дерева (нс)	Удаление из файла (нс)	Размер дерева (байт)	Размер файла (байт)
1000	1809	110	4670	30000	6000
2000	3318	120	9370	60000	12000
5000	7449	350	23030	150000	30000
10000	15440	670	46320	300000	60000
20000	34854	1230	91860	600000	120000

Вырожденное дерево:

Размер файла (слов)	Обход дерева (нс)	Удаление из дерева (нс)	Удаление из файла (нс)	Размер дерева (байт)	Размер файла (байт)
1000	1660	110	4660	30000	6000
2000	3227	300	9390	60000	12000
5000	8047	450	23200	150000	30000
10000	16091	1380	46320	300000	60000
20000	32275	2530	92970	600000	120000

Заметим, что операции в файле значительно медленнее, но файл менее затратен по памяти, в том числе и к стеку. Также время операций на файле стабильно и не зависит от его структуры в отличие от дерева.

Вывод

Время обхода дерева не зависит от формы дерева (так как при любой форме дерева количество рекурсивных вызовов одинаково), что подтверждается замерами.

Удаление из дерева зависит от формы дерева и работает значительно быстрее на сбалансированном дереве (так как в среднем меньше рекурсивных вызовов, а также в случае строгого неравенства можно не рассматривать одно из поддеревьев), что также подтверждается замерами времени.

Операции на бинарном дереве значительно быстрее чем операции на файле, но дерево занимает значительно больше места по памяти. Скорость операций в дереве зависит от его структуры. Также в дереве сложно хранить одинаковые элементы (их наличие также снижает эффективность дерева), а рекурсивная реализация дерева требовательна к размеру стека.

Контрольные вопросы:

Что такое дерево?

Связный граф, не содержащий циклы.

Как выделяется память под представление деревьев?

Память под представление деревьев выделяется динамически для каждого узла.

Какие бывают типы деревьев?

- Бинарное дерево – дерево, в котором каждый узел имеет не более 2 потомков
- N-арное дерево – дерево, в котором каждый узел имеет не более n потомков
- Сбалансированное дерево – дерево, в котором высота поддеревьев, каждого из узлов, отличается не более чем на 1.
- Вырожденное дерево – дерево, каждый узел которого имеет не более 1 потомка.
- Двоичное дерево поиска (BST) – двоичное дерево в котором для каждого узла верно, что оба поддерева являются двоичными деревьями поиска, все элементы левого поддерева меньше или равны чем корень, а все элементы правого – больше или равны.
- AVL-дерево – сбалансированное по высоте двоичное дерево поиска.
- Красно-чёрное дерево – один из видов самобалансирующихся бинарных деревьев поиска.
- Куча – двоичное дерево, отвечающее свойству кучи (в максимальной куче все элементы поддеревьев узла меньше или равны чем значение узла, в минимальной – наоборот)

Какие стандартные операции возможны над деревьями?

Обход дерева, поиск элемента в дереве, включение элемента в дерево, исключение элемента из дерева.

Что такое дерево двоичного поиска?

Двоичное дерево поиска — двоичное дерево, для которого выполняются следующие свойства:

- оба поддерева — являются двоичными деревьями поиска
- у всех узлов левого поддерева произвольного узла значения ключей данных меньше либо равны, значения ключа данных этого узла
- у всех узлов правого поддерева произвольного узла значения ключей данных больше либо равны, значения ключа данных этого узла