



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»

Вариант 5

Студент: Писаренко Дмитрий Павлович

Группа: ИУ7-34Б

Студент

Писаренко Д.П.

подпись, дата

фамилия, и.о.

Преподаватель

Рыбкин Ю.А.

подпись, дата

фамилия, и.о.

Цель работы

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

Условие задачи

Построить частотный словарь (слово – количество повторений) из слов текстового файла в виде дерева двоичного поиска. Вывести его на экран в виде дерева. Осуществить поиск указанного слова в дереве и в файле. Если слова нет, то (по желанию пользователя) добавить его в дерево и, соответственно, в файл. Сравнить время поиска слова в дереве и в файле.

Техническое задание

Исходные данные

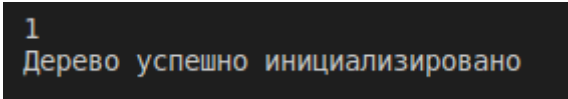
Выбор действия: целое число от 0 до 7.

0. Выход из программы

Не требует ввода от пользователя.

1. Инициализировать дерево из файла

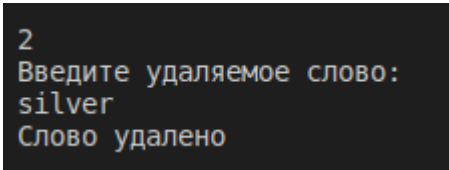
Данные читаются из указанного файла в бинарное дерево. При успешном чтении пользователю сообщается о нем, иначе – программа завершается с ненулевым кодом возврата.



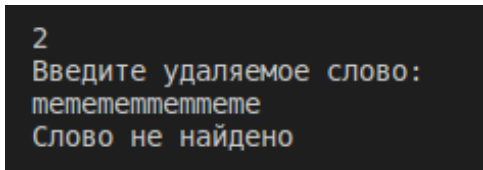
```
1
Дерево успешно инициализировано
```

2. Исключить узел

Пользователь вводит слово, которое хочет удалить. Если такое слово найдено – слово удаляется и на экран выводится “Слово удалено”, в ином случае – на экран выводится “Слово не найдено”.



```
2
Введите удаляемое слово:
silver
Слово удалено
```



```
2
Введите удаляемое слово:
тетететтетете
Слово не найдено
```

3. Найти указанное слово

Пользователь вводит слово, которое ищется в дереве. При нахождении на экран выводится “Слово НАЙДЕНО в дереве”, в ином случае на экран выводится “Слово НЕ НАЙДЕНО в дереве” и пользователю предлагается добавить его с помощью ввода “1”, после добавления на экран выводится “Слово добавлено в файл и дерево”.

```
3
Введите слово, которое нужно найти:
gip
Слово НАЙДЕНО в дереве
```

```
3
Введите слово, которое нужно найти:
ботонка
Слово НЕ НАЙДЕНО в дереве
Если хотите добавить его - введите 1, если нет - любое целое число
1
Слово добавлено в файл и дерево
```

4. Вывести дерево

При помощи скрипта, языка описания графов DOT и программы Graphviz получаем выведенное дерево “tree.png”.

```
#!/bin/sh
dot -Tpng tree.gv -o tree.png
gio open tree.png
```

Скрипт print_tree.sh

5. Сравнить время поиска в дереве и файле

Пользователю предлагается ввести размерность дерева (10/100/1000/10000 слов). После этого для данного количества слов сравнивается время поиска в дереве и файле.

```
5
Выберите размерность (1 - 10, 2 - 100, 3 - 1000, 4 - 10000):
3
Затраченное время для файла - 46.04 мс
Затраченное время для дерева - 0.31 мс
```

6. Сравнить время добавления в дереве и файле

Пользователю предлагается ввести размерность дерева (10/100/1000/10000 слов). После этого для данного количества слов сравнивается время добавления в дереве и файле.

```
6
Выберите размерность (1 - 10, 2 - 100, 3 - 1000, 4 - 10000):
3
Затраченное время для файла - 0.03 мс
Затраченное время для дерева - 0.35 мс
```

7. Сравнить время удаления в дереве и файле

Пользователю предлагается ввести размерность дерева (10/100/1000/10000 слов). После этого для данного количества слов сравнивается время удаления в дереве и файле.

```
7
Выберите размерность (1 - 10, 2 - 100, 3 - 1000, 4 - 10000):
3
Затраченное время для файла - 50.11 мс
Затраченное время для дерева - 0.41 мс
```

Структуры данных

```
typedef struct binary_tree
{
    struct branch *head;
    int size;
} binary_tree_t;
```

`struct branch *head` – указатель на корень дерева
`int size` – количество элементов

Описание полей структуры `binary_tree_t`

```
typedef struct branch
{
    char *word;
    struct branch *parent;
    struct branch *left;
    struct branch *right;
} branch_t;
```

`char *word` – слово
`struct branch *parent` – родительский узел вершины
`struct branch *left` – левый узел вершины
`struct branch *right` – правый узел вершины

Способ обращения к программе

Работа с программой осуществляется с помощью консоли.

Сборка осуществляется с помощью команды **make release**

Запуск выполняется с помощью команды **./app.exe**

Дальнейшая работа производится с помощью меню:

```
o dimasxt@dimasxt-VirtualBox:~/BMSTU-TaDS/lab_06$ ./app.exe

1. Инициализировать дерево из файла
2. Исключить узел
3. Найти указанное слово
4. Вывести дерево
5. Сравнить время поиска в дереве и файле
6. Сравнить время добавления в дереве и файле
7. Сравнить время удаления в дереве и файле
0. Выход

█
```

Тестирование

Позитивные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 1 Ключ = 3 Слово, которое нужно найти: gir	Сообщение: Слово НАЙДЕНО в дереве	Ожидание следующего ключа
2	Ключ = 1 Ключ = 3 Слово, которое нужно найти: аааа 1	Сообщение: Слово НЕ НАЙДЕНО в дереве ... Слово добавлено в файл и дерево	Ожидание следующего ключа
3	Ключ = 1 Ключ = 2 Удаляемое слово: аааа	Сообщение: Слово не найдено	Ожидание следующего ключа
4	Ключ = 1 Ключ = 2 Удаляемое слово: silver	Сообщение: Слово удалено	Ожидание следующего ключа
5	Ключ = 1 Ключ = 4	Дерево в файле .png	Ожидание следующего ключа
6	Ключ = 5 Размерность: 3	Время для поиска в дереве и файле	Ожидание следующего ключа
7	Ключ = 0	Отсутствуют	Завершение программы

Негативные тесты

#	Входные данные	Выходные данные	Результат
1	Ключ = 123	Сообщение: Номер меню - целое число от 0 до 5	Код ошибки 2
2	Ключ = 5 Ключ = 123	Сообщение: Размерность - число от 1 до 4	Код ошибки 8
3	Ключ = 1 Некорректное название файла	Сообщение: Файл не найден	Код ошибки 5
4	Ключ = dfwfd	Сообщение: Номер меню - целое число	Код ошибки 1

Таблицы с результатами измерения времени

Время замерялось при 1000 выполнениях функций.

Время в таблицах указано в мс.

Поиск

Размерность	Дерево	Файл
10	0.06	1.22
100	0.11	10.85
1.000	0.33	40.08
10.000	0.83	361.5

Добавление

Размерность	Дерево	Файл
10	0.07	0.03
100	0.14	0.03
1.000	0.38	0.04
10.000	0.63	0.03

Удаление

Размерность	Дерево	Файл
10	0.08	1.2
100	0.13	23.53
1.000	0.41	83.22
10.000	1.3	951.61

Контрольные вопросы

1. Что такое дерево?

Дерево – нелинейная структура данных, которая используется для представления иерархических связей «один ко многим». Дерево с базовым типом T определяется рекурсивно: это либо пустая структура (пустое дерево), либо узел типа T с конечным числом древовидных структур того же типа – поддеревьев.

2. Как выделяется память под представление деревьев?

Выделение памяти под деревья определяется типом их представления. Это может быть таблица связей с предками (№ вершины - № родителя), или связный список сыновей. Оба представления можно реализовать как с помощью матрицы, так и с помощью списков. При динамическом представлении деревьев (когда элементы можно удалять и добавлять) целесообразнее использовать списки – т.е. выделять память под каждый элемент динамически.

3. Какие бывают типы деревьев?

N -арное дерево, сбалансированное дерево, бинарное дерево, бинарное дерево поиска, дерево AVL, красное-черное дерево, 2-3 дерево.

4. Какие стандартные операции возможны над деревьями?

Обход, поиск, добавление и удаление элемента.

5. Что такое дерево двоичного поиска?

Дерево двоичного поиска – дерево, в котором все левые потомки «моложе» предка, а все правые – «старше».

Вывод

В ходе выполнения лабораторной работы была освоена обработка деревьев. Экспериментальным путем доказано, что поиск и удаление в дереве выполняются в несколько раз, чем в файле (в 20-800 раз в зависимости от размерности: чем больше размерность, тем больше преимущество дерева). Добавление же быстрее работает в файле: оно не зависит от размерности, так как файл открывается на дозапись.