**Лабораторная №6**

**Янова Даниэлла, ИУ7-43(бывшая 33)**

**Цель задачи:**

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

**Постановка задачи**

5 вариант(19 в списке). Построить частотный словарь 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.

**Входные данные:**

Пункт меню

1 - Input data from file

2 - Output tree

3 - Search in tree

4 - Search in file

5 - Obhod

6 - Compare time

7 - Delete element

0 – Exit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Choise option:

**Выходные данные:**

Cчитывание слов в дерево с файла, имя которого вводит пользователь

Вывод дерева

Поиск слова в дереве двоичного поиска и в файле

Добавление слова в дерево и в файл, если его нигде нет

Постфиксный, инфиксный, префиксный обход слов

Сравнение по времени(тикам) поиска слов в дереве и в файле

**Аварийные ситуации**

Ошибка выделения памяти

Неправильный ввод выбора пункта меню

**Используемые структуры данных**

Узел дерева

struct Tree\_list{

struct Node Data; //данные узла

struct Tree\_list \*Left; //ссылка на левый под-узел

struct Tree\_list \*Right; //ссылка на правый под-узел

};

Структура описания дерева

struct Node{

char Word[MAX\_WORD\_SIZE]; //слово

int Count; //кол-во повторов слова в файле

};

**Использование деревьев двоичного поиска и файлов для поиска слов**

SEARCH

file tree

0004 bytes 73373 1524(4814%)

0170 bytes 77761 2712(2867%)

0820 bytes 159384 2791(5710%)

1688 bytes 227341 2967(7664%)

2119 bytes 237392 3271(7257%)

Дерево двоичного поиска регулируется по старшинству детей и родителя, из-за чего поиск по нему идет легче и быстрей, чем по идеально сбалансированному дереву. Поэтому поиск по такому дереву происходит за примерно одно и то же время(~2700 тиков).

Время поиска по файлу при этом увеличивается по мере увеличения текста в файле. И также следует обратить внимания на то, что поиск в файле превышает поиск в дереве во много раз. Отсюда следует вывод о том, что рациональнее использовать деревья.

**Ответы на контрольные вопросы:**

1. Что такое дерево?

Дерево – это нелинейная структура данных, используемая для представления иерархических связей, имеющих отношение «один ко многим».

1. Как выделяется память под представление деревьев?

Динамически (так как дерево – структура данных, рационально использующая память, под дерево в любой момент времени выделено ровно столько памяти, сколько требуется для его представления на данный момент)

1. Какие бывают типы деревьев?

По характеру хранения для бинарных деревьев:

Деревья двоичного поиска

Идеально сбалансированные

АВЛ деревья (сбалансированные деревья двоичного поиска)

1. Какие стандартные операции возможны над деревьями?

Добавление

Удаление

Поиск

Обход дерева – постфиксный, инфиксный, префиксный

1. Что такое дерево двоичного поиска?

Дерево двоичного поиска – дерево, в котором для любого узла соблюдается правило:

Левый ребенок меньше отца

Правый ребенок больше отца