|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 6 |

**Название:**

Деревья

**Дисциплина:** Типы и структуры данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-36Б |  | | А.А. Жаворонкова |
|  | (Группа) | |  | (И.О. Фамилия) |
|  |  | |  |  |
| Преподаватель: | Никульшина Т. А. | |  |  |

Москва, 2022

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.

Описание ТЗ

*Описание исходных данных и результатов*

*Исходные данные:* пункт меню (см. описание задачи), элементы дерева – строки.

*Результаты:* дерево, найденный элемент дерева, элементы, начинающиеся на заданную букву, таблица сравнения времени поиска.

*Описание задачи, реализуемой программой*

Меню:

1) Загрузить данные из файла

2) Вывести дерево

3) Добавить элемент

4) Удалить элемент

5) Найти элемент

6) Определить количество вершин дерева, содержащих слова, начинающиеся на указанную букву

7) Сравнить время поиска в дереве и файле

8) Вывести меню

0) Выйти из программы

*Способ обращения к программе*

Вызов программы происходит через терминал (main.exe). Дальнейшая работа с программой выполняется при помощи меню, выводимого на экран.

*Описание возможных ошибок пользователя*

* Добавление элемента, уже присутствующего в дереве
* Добавление / удаление элемента в неинициализированное дерево
* Удаление / поиск несуществующего элемента

Описание внутренних структур данных

Дерево представляется списком, каждый элемент которого описывается следующим типом:

typedef struct node

{

    char \*word;

    struct node \*left;

    struct node \*right;

} node\_t;

**word –** указатель на строку (данные элемента)  
**left –** указатель на элемент больше данного  
**right –** указатель на элемент меньше данного

Набор тестов с указанием проверяемого параметра

|  |  |  |
| --- | --- | --- |
| Входные данные | Выходные данные | Что проверяется |
| Добавление существующего элемента | «Элемент уже существует» | Обработка ошибки добавления существующего элемента |
| Удаление несуществующего элемента | «Элемент не найден» | Обработка ошибки удаления не существующего элемента |
| Поиск элемента | Выделение цветом найденного элемента | Поиск элемента |
| Поиск по первой букве | Выделение цветом найденных элементов | Поиск слов, начинающихся на указанную букву |
| Вывод дерева | «Дерево еще не считано» | Вывод не считанного дерева |

Описание алгоритма

Для поиска слов, начинающихся с указанной буквы, используется префиксный обход дерева: сначала проходятся все вершины левого поддерева, затем все вершины правого.  
Для удаления узла с указанным ключом сначала происходит его поиск. В случае, если узел найден, то он удаляется. Если удаляемый узел является концевым, то просто удаляется ссылка на него. Если удаляемый узел имеет одного потомка, в этом случае переадресуется ссылка на этого потомка. Если удаляемый узел имеет двух потомков, то на его место ставится самый левый потомок из правого поддерева.  
Операция включения элемента в дерево разбивается на три этапа: включение узла в пустое дерево, поиск корня для добавления нового узла, включение узла в левое или правое поддерево.

Временная эффективность

|  |  |  |
| --- | --- | --- |
| Количество элементов | Дерево | Файл |
| 10 | 167 | 6633 |
| 50 | 322 | 8025 |
| 100 | 853 | 17786 |
| 500 | 2735 | 50343 |

Затраты памяти

|  |  |  |
| --- | --- | --- |
| Количество элементов | Дерево | Файл |
| 10 | 240 | 64 |
| 50 | 1200 | 422 |
| 100 | 2400 | 832 |
| 500 | 12000 | 3950 |

Вывод

Поиск элемента в дереве выполняется быстрее, чем в файле, потому что в дереве нет необходимости просматривать каждый элемент, так как все элементы больше текущего расположены справа, меньше – слева. Однако по памяти файл выигрывает у дерева.

Ответы на вопросы

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

2. Как выделяется память под представление деревьев?  
 Память выделяется для каждого элемента отдельно, то есть под каждую структуру.

3. Какие бывают типы деревьев?

Красно-черное дерево, неупорядоченное, бинарное дерево.

4. Какие стандартные операции возможны над деревьями?  
 Обход дерева, поиск по дереву, включение в дерево, исключение из дерева.

5. Что такое дерево двоичного поиска? Дерево двоичного поиска – это бинарное дерево, в котором все левые потомки моложе предка, а все правые – старше.