|  |  |
| --- | --- |
| **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 Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6**

**«ОБРАБОТКА ДЕРЕВЬЕВ»**

Студент Фам Минь Хиеу

Группа ИУ7 – 32Б

Преподаватель Никульшина Т.А.

*2022 г.*

**Цель работы:**

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

**Условие**

Построить двоичное дерево поиска, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Определить количество вершин дерева, содержащих слова, начинающиеся на указанную букву. Выделить эти вершины цветом. Сравнить время поиска начинающихся на указанную букву слов в дереве и в файле.

**Входные данные**

Номер пункта: целое число в диапазоне от 0 до 6.

**Выходные данные**

Данная программа выводит на экран ДДП в виде дерева, сравниние эффективности поиска в ДДП и в файле.

Функции программы:

Работа с программой осуществляется в консоли с помощью меню:

[1] – загружает информацию из текстового файла. Генерирует дерево двоичного поиска.

[2] – выводит на экран деревья двоичного поиска.

[3] – добавление элемента.

[4] – удаление элемента.

[5] – запрашивает у пользователя указанную букву, по которой найти все слова

[6] – Сравнить время поиска в дереве и в файле.

[0] – выход из программы.

Обращение к программе:

Запускается через ./app.exe

Аварийные ситуации:

1. Некорректный ввод номера команды.

На входе: 7.

На выходе: сообщение «Ошибка, не верно введен пункт!!!»

2. Некорректный ввод номера команды.

На входе: a.

На выходе: сообщение « Ошибка, не верно введен пункт!!!»

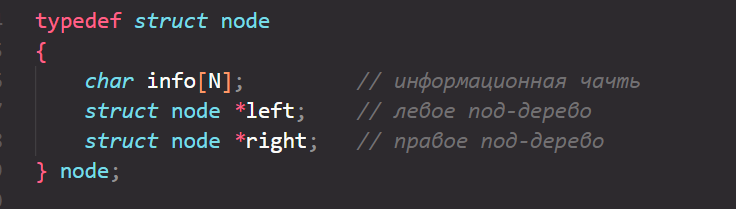
3. Пустая очередь при удалении.

На входе:

На выходе: сообщение «Пустое дерево!»

**Структуры данных и функции**

Узел дерева представляет собой структуру:



**Описание алгоритма**

1. Обход дерево либо слева направо: A,R,B (инфиксный обход) либо снизу вверх: A,B,R (постфиксный обход) сверху вниз: R,A,B (префиксный обход).
2. Включение : Начните с корня. Сравните искомый элемент с корнем, если он меньше корня, то рекурсивно вызовите левое поддерево, иначе рекурсивно вызовите правое поддерево.
3. Удаление : Если корень равен NULL, то вернуть корень (базовый случай). Если ключ меньше значения корня, то установить

root->left = deleteNode(root->left, key) Если ключ больше значения корня, то установите root->right = deleteNode(root->right, key). В противном случае проверьте если корень является листовым узлом, возвращается null. Иначе, если у него есть только левая потомка, возвращается левая потомка .Иначе, если у него есть только правая потомка, возвращается правая потомка. Иначе замените значение удаляемого узла значением самого левого узла правого поддерева

Поиск: работает так же, как алгоритм бинарного поиска

# **Набор тестов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Название теста** | **Пользователь вводит** | **Вывод** |
| 1 | Некорректный ввод команды | 1 | Ошибка, не верно введен пункт! |
| 2 | Некорректный ввод команды | A | Ошибка, не верно введен пункт! |
| 3 | Команда 1 | 1 | Файл читается |
| 4 | Команда 2 | 2 | ДДП в виде дерева |
| 5 | Команда 3 | 3 | Элемент добавлено |
| 6 | Команда 4 | 4 | Элемент удалено |
| 7 | Команда 5 | 5 | Результат поиска в ДДП |
| 8 | Команда 6 | 6 | Эффективность использования ДДП |

# **Оценка эффективности**

|  |
| --- |
| **time search in BST at lever 0 : 0.002000**  **time search in file text: 0.120000**  **use BST beats 60.000000 times**  **time search in BST at lever 1: 0.005000**  **time search in file text: 0.151000**  **use BST beats 30.200000 times**  **time search in BST at lever 2: 0.008000**  **time search in file text: 0.176000**  **use BST beats 22.000000 times**  **time search in BST at lever 3: 0.011000**  **time search in file text: 0.166000**  **use BST beats 15.098182 times** |

**Выводы**

В ходе данной работы я научился работать с таким структуром данных, как дерево двоичного поиска, поиск элемента в ДДП. Также я сравнил эффективность поиска по ДДИ и по файлу при различных степенях. Использовать двоичное дерево поиска для поиска данных намного эффективнее, чем использовать файл, так как поиск в ДДП тратим O(log(N)) времени по теории, а в файле O(N).

# **Ответы на контрольные вопросы**

**1. Что такое дерево?**

Дерево – рекурсивная структура данных, используемая для представления иерархических связей, имеющих отношение “один ко многим”.

**2. Каким выделяется память под представление деревьев?**

В виде связанного списка. Каждый лепесток – узел. Память: (sizeof(тии) + 8 \* 2) \* size\_of\_tree.

**3. Какие бывают типы деревьев?**

Дерево, двоичное дерево, двоичное дерево поиска, АВЛ дерево.

**4. Какие стандартные операции возможны над деревьями?**

Обход дерева, поиск по дереву, включение элемента в дерево, исключение элемента из дерева.

**5. Что такое дерево двоичного поиска?**

Двоичное дерево поиска – двоичное дерево, для каждого узла которого выполняется условие: левый потомок больше или равен родителю, правый потомок строго меньеш родителя (также можно наоборот).