Отчет по Лабораторной работе №6

Дисциплина «Типы и Структуры Данных»

# Чалый Андрей

**Группа ИУ7-32Б**

Цель работы

Построить ДДП, сбалансированное двоичное дерево и хеш-таблицу по указанным данным. Сравнить эффективность поиска в сбалансированном двоичном дереве, в ДДП и в хеш-таблице используя открытую адресацию. Вывести на экран деревья и хеш-таблицу. Подсчитать среднее количество сравнений для поиска данных в указанных структурах. Произвести реструктуризацию хеш-таблицы, если среднее количество сравнений больше указанного. Оценить эффективность использования этих структур (по времени и памяти) для поставленной задачи. Оценить эффективность поиска в хеш-таблице при различном количестве коллизий.

Задание

Построить ДДП, в вершинах которого находятся слова из текстового файла. Вывести его на экран в виде дерева. Сбалансировать полученное дерево и вывести его на экран. Построить хеш-таблицу из слов текстового файла. Использовать метод цепочек для устранения коллизий. Осуществить поиск введенного слова в ДДП, в сбалансированном дереве, в хеш-таблице и в файле. Сравнить время поиска, объем памяти и количество сравнений при использовании различных (4-х) структур данных. Если количество сравнений в хеш-таблице больше указанного (вводить), то произвести реструктуризацию таблицы, выбрав другую функцию.

Входные данные

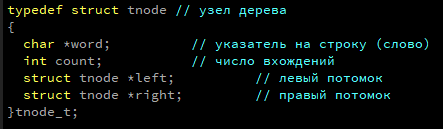
Текст в файле на английском языке, выбор пользователем меню, размер хэш-таблицы, кол-во сравнений для хэш-таблицы, слово добавления

Выходные данные

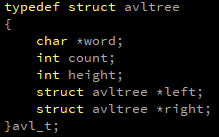
ДДП, АВЛ, хеш-таблица (метод цепочек), оценка эффективности поиска элемента (для деревьев, хеш-таблицы и файла), среднее количество сравнений.

Внутренняя структура данных

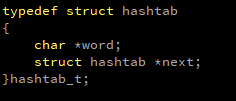
ДДП:



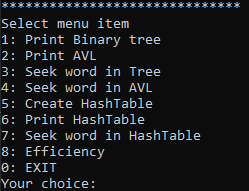
AVL:



Хеш-таблица:



Меню

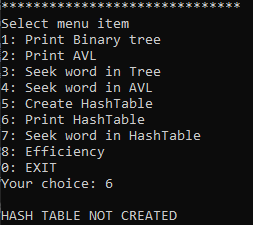


Аварийные ситуации

Аварийные ситуации в данной лабораторной связаны с неверным вводом пункта меню.

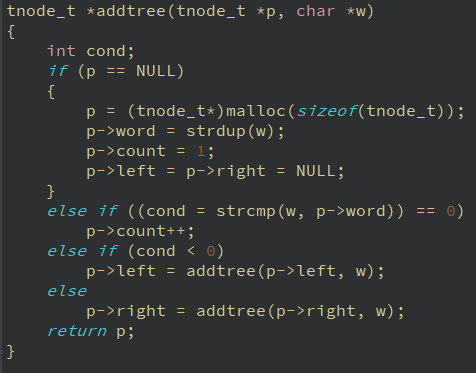
Также пользователь должен сначала заполнить хэш-таблицу, и только затем осуществлять поиск слов в ней, либо выводить ее.

Попытка вывести таблицу, не задав ее:

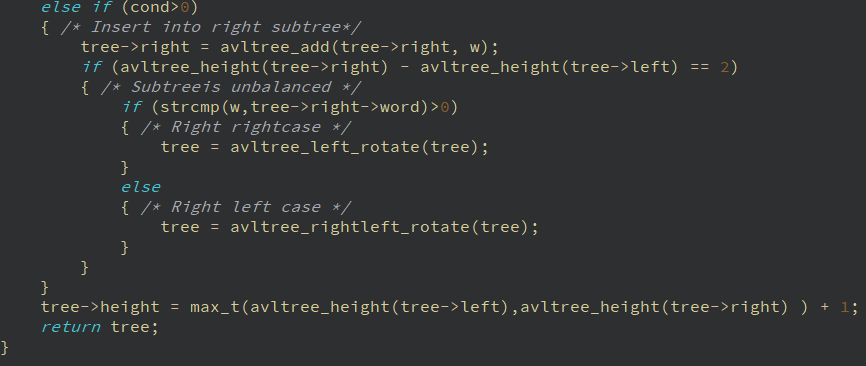
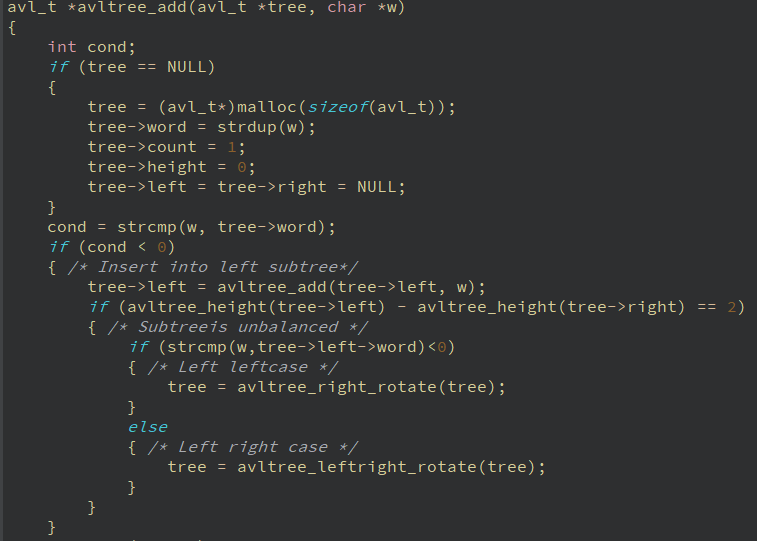


Алгоритм добавления

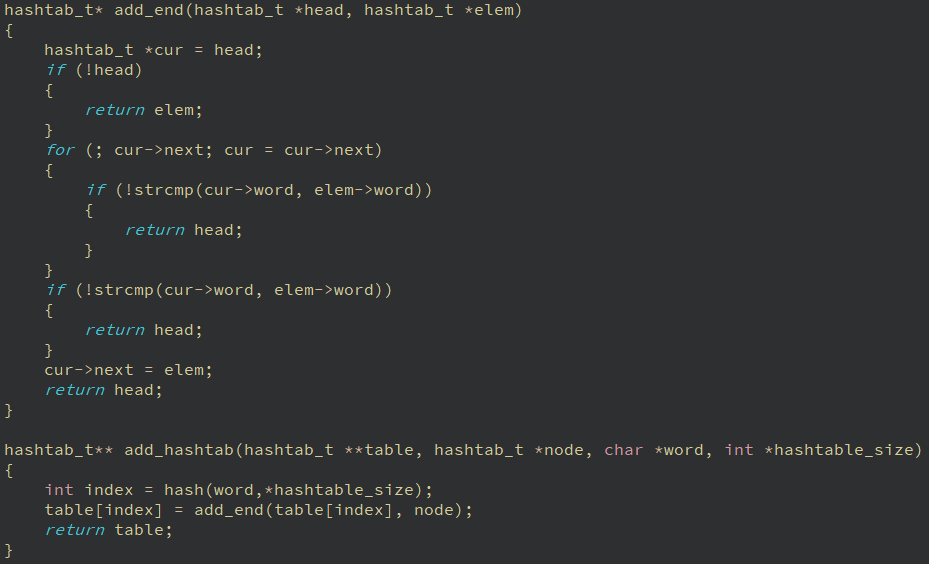
1. Дерево



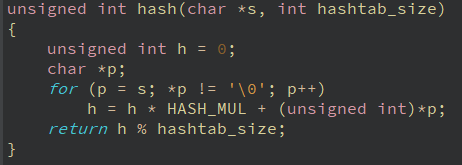
2. АVL



3. Хэш таблица

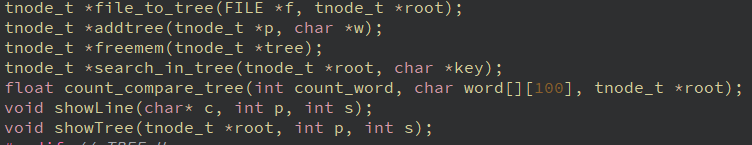


Используемая хэш – функция:

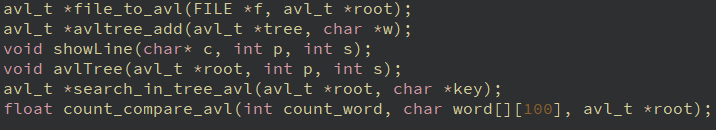


Вычисление остатка от деления ключа на размерность таблицы. Размерность таблицы передается нечетная. Когда наше среднее количество сравнений оказывается больше допустимого, происходит реструктуризация хэш-таблицы путем увеличения ее размера и заново пересчитываем hash.

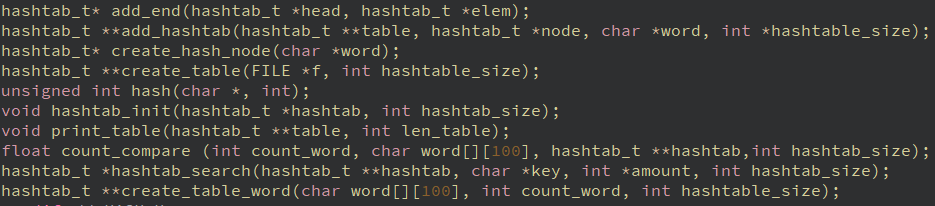
Функции для работы с ДДП



Функции для работы с АВЛ



Функции для работы с таблицей

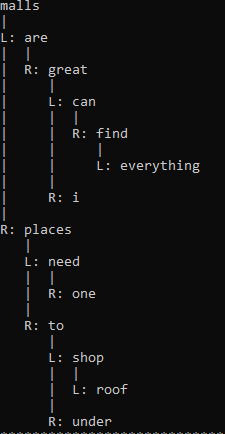


Тесты

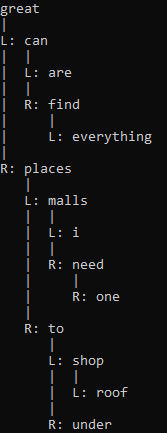
Пусть в файле следующая строка:

“Malls are great places to shop. I can find everything need under one roof.”

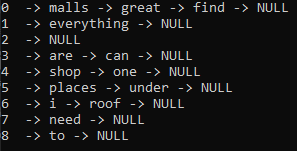
1. ДДП – вывод



1. AVL



1. ХЭШ-таблица при кол-во сравнений не больше 2



1. Поиск слова в ДДП



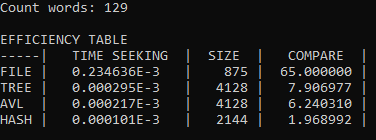
1. Поиск слова в AVL



1. Поиск слова в хэш-таблице



1. Эффективность



Оценка эффективности

В результате ХЭШ-таблица оказалась наиболее эффективной по всем параметрам (не считая эффективности по размеру по отношению к файлу). Это объясняется тем, что ХЭШ-таблица хранит только по одному указателю на каждое слово, дерево хранит два указателя (на правого и левого потомка)

По времени АВЛ на 25% эффективнее ДДП, это связано с тем, что АВЛ дерево сбалансированно, а вот ХЭШ эффективнее на 65% ДДП.

По памяти хэш-таблица эффективнее на 50 процентов, чем дерево.

Хэш-таблица имеет нормальный показатель – среднее количество сравнений 2. При таком показателе она считается эффективной.

Файл хоть и выгоден по памяти на 60 процентов, но сильно проигрывает по времени.

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?

Выделение памяти под деревья определяется типом их представления. Это может быть таблица связей с предками или связный список сыновей. Оба представления можно реализовать в виде матрицы или списка. При реализации списком память выделяется динамически, при реализации матрицей статически.

3. Какие стандартные операции возможны над деревьями?

Основные операции с деревьями: обход дерева, поиск по дереву, включение в дерево, исключение из дерева.

4. Что такое дерево двоичного поиска?

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

5. Чем отличается идеально сбалансированное дерево от АВЛ дерева?

Дерево, у которого число вершин в левом и правом поддеревьях отличается не более, чем на единицу называется идеально сбалансированным. Двоичное дерево, у каждого узла которого высота двух поддеревьев отличается не более чем на единицу называется АВЛ-деревом.

6. Чем отличается поиск в АВЛ-дереве от поиска в дереве двоичного поиска?

Поиск в АВЛ дереве имеет сложность О(log2n), в то время как в обычном ДДП сложность О(n).

7. Что такое хеш-таблица, каков принцип ее построения?

Массив, заполненный в порядке, определенным хеш-функцией, называется хеш- таблицей.

Хеш-функция – функция, которая ставит в соответствие каждому ключу индекс ячейки, где расположен элемент с этим ключом.

8. Что такое коллизии? Каковы методы их устранения.

Коллизия – ситуация, когда разным ключам соответствует одно значение хеш-функции. Существует несколько возможных вариантов разрешения коллизий: внешнее (открытое) хеширование (метод цепочек) и внутреннее (закрытое) хеширование (открытая адресация).

9. В каком случае поиск в хеш-таблицах становится неэффективен?

Поиск в хэш-таблице становится неэффективным при большом числе коллизий – сложность поиска возрастает.

При открытом хэшировании в случае, когда элемент таблицы с индексом, который вернула хеш-функция, уже занят, к нему присоединяется связный список. Таким образом, если для нескольких различных значений ключа возвращается одинаковое значение хеш- функции, то по этому адресу находится указатель на связанный список, который содержит все значения. Поиск в этом списке осуществляется простым перебором, так как при грамотном выборе хеш-функции любой из списков оказывается достаточно коротким.

При закрытом хэшировании в этом случае, если ячейка с вычисленным индексом занята, то можно просто просматривать следующие записи таблицы по порядку, до тех пор, пока не будет найден ключ K или пустая позиция в таблице.

10. Эффективность поиска в АВЛ деревьях, в дереве двоичного поиска и в хеш-таблицах.

В хэш-таблице минимальное время поиска О(1). В АВЛ дереве О(log2n). В дереве двоичного поиска О(h), где h – высота дерева.