**Отчет по лабораторной работе 6**

**По предмету «Типы и структуры данных»**

Студент ИУ7-32Б

Герасименко Екатерина

**Цель работы:** построить дерево, вывести его на экран в виде дерева, реализовать основные операции работы с деревом: обход дерева, включение, исключение и поиск узлов, сбалансировать дерево, сравнить эффективность алгоритмов сортировки и поиска в зависимости от высоты деревьев и степени их ветвления; построить хеш-таблицу и вывести ее на экран, устранить коллизии, если они достигли указанного предела, выбрав другую хеш-функцию и реструктуризировав таблицу; сравнить эффективность поиска в сбалансированных деревьях, в двоичных деревьях поиска (ДДП) и в хеш-таблицах. Сравнить эффективность реструктуризации таблицы для устранения коллизий с эффективностью поиска в исходной таблице.

**Условие задачи:**

В текстовом файле содержатся целые числа. Построить ДДП из чисел файла. Вывести его на экран в виде дерева. Сбалансировать полученное дерево и вывести его на экран. Построить хеш-таблицу из чисел файла. Использовать метод цепочек для устранения коллизий. Осуществить поиск введенного целого числа 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:**

Текстовый файл с данными. Максимальное количество сравнений в хеш-таблице.

**Выходные данные:**

Вывод в консоль обычного дерева/сбалансированного дерева, созданной хеш-таблицы.

**Функция программы:**

Создание обычного/сбалансированного дерева, хеш-таблицы из файла. Поиск элементов в этих структурах.

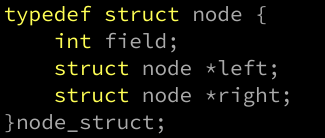
**Обращение к программе** осуществляется через консоль.

**Аварийные ситуации:**

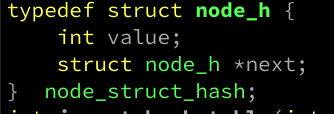
Ошибки выделения памяти, некорректный ввод.

**Структуры данных:**

Узел дерева:

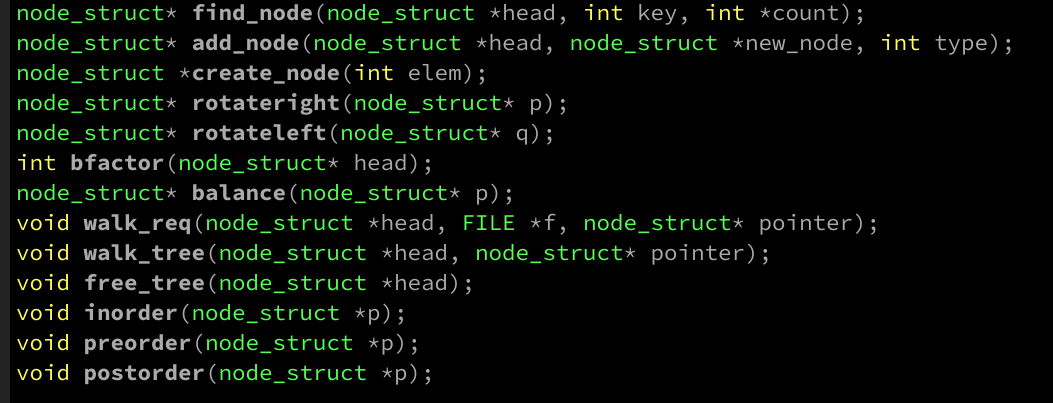
****

Узел хеш-таблицы:

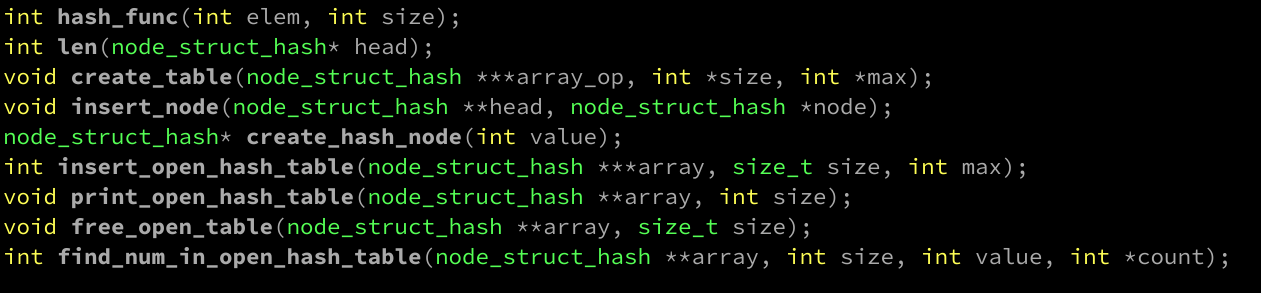
****

**Функции:**

Функции работы с деревьями:

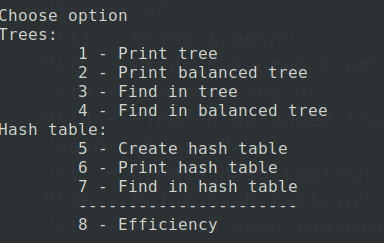


Функции работы с хеш-таблицей:

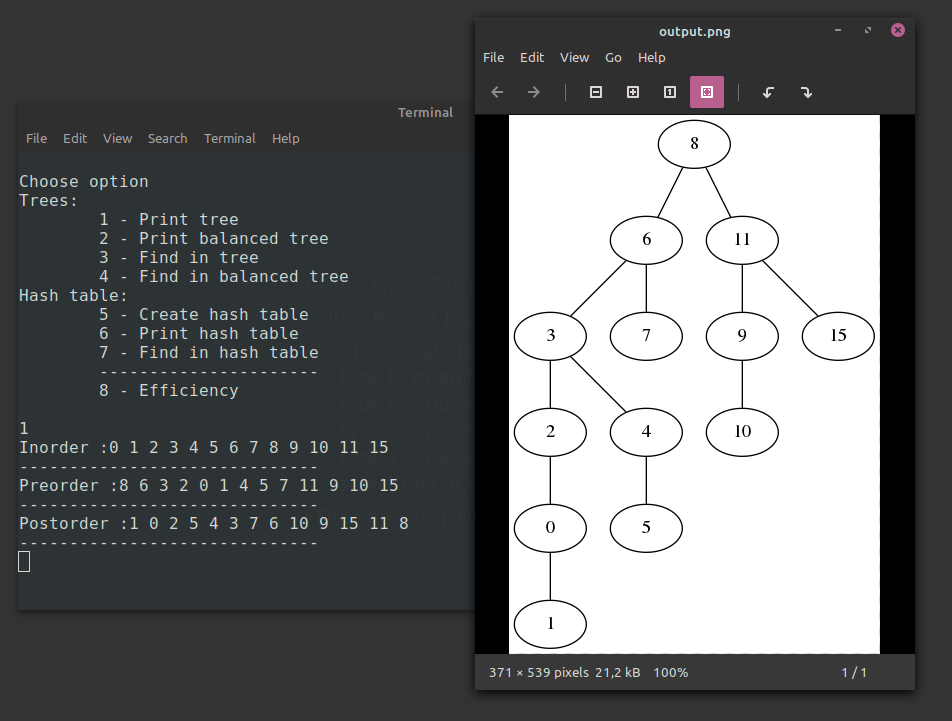
****

**Интерфейс:**

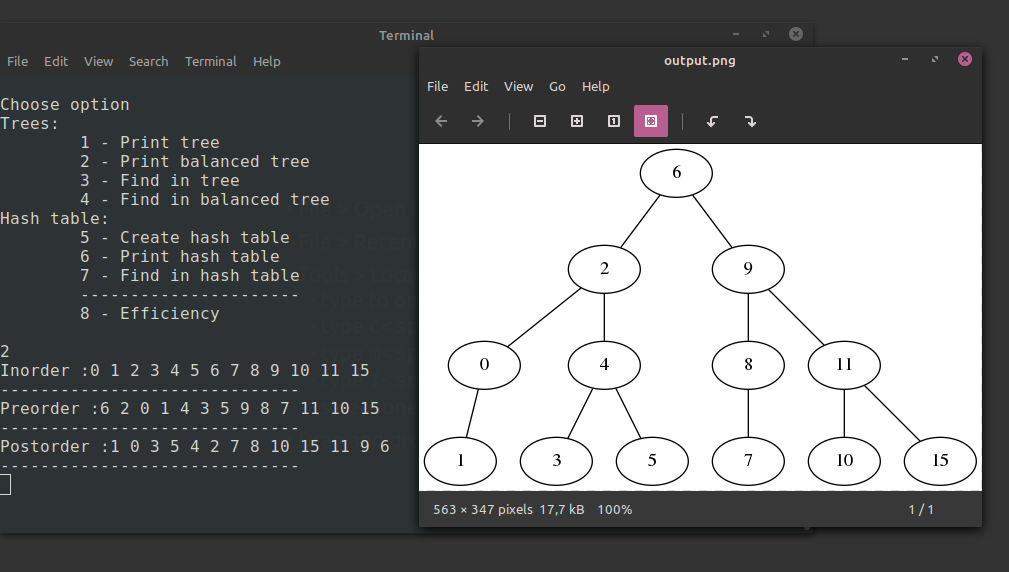
Меню:



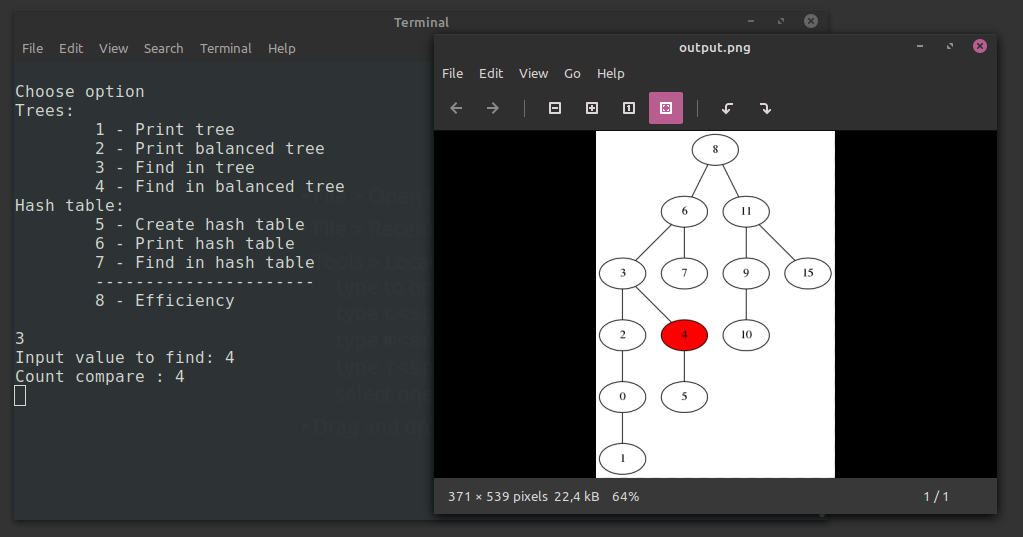
Вывод обычного дерева, включая обходы:



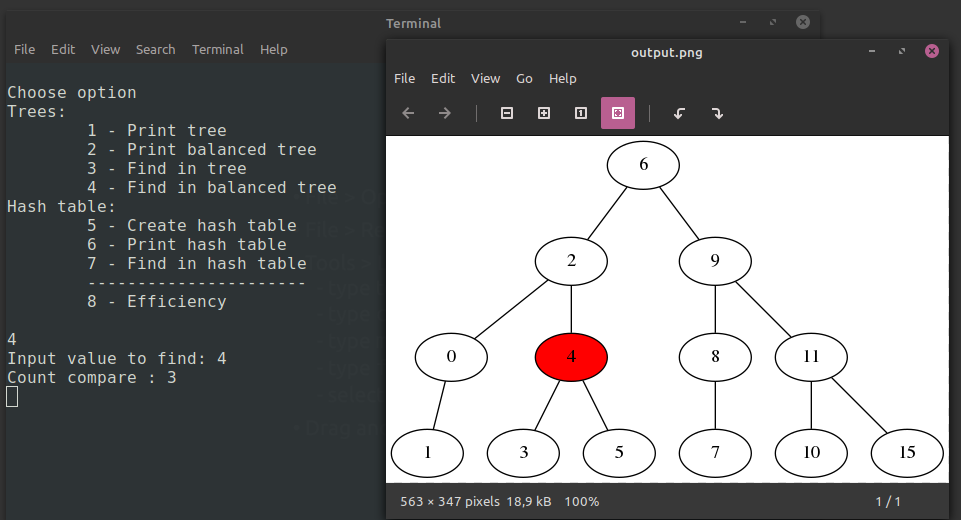
Вывод сбалансированного дерева, включая обходы:



Поиск в обычном дереве:

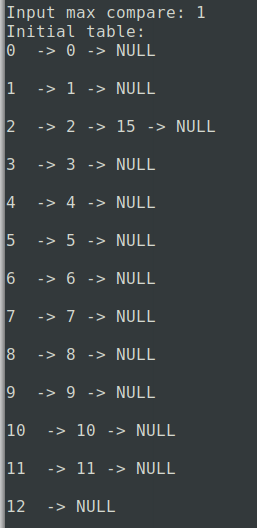


Поиск в сбалансированном дереве:



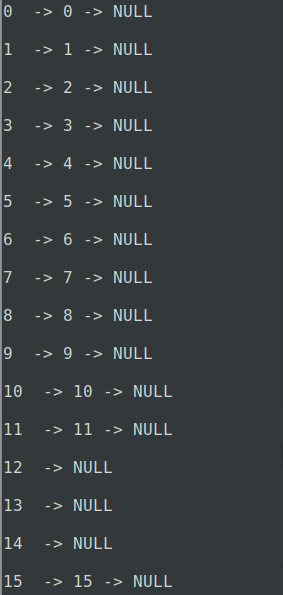
Хеш-таблица(метод цепочек):

Изначально за размер, подаваемый в хеш-функцию, принимается количество чисел в файле. Хеш-функция берет остаток от деления значения на размер.



Если максимальное количество сравнений превышает введенное значение, то мы реструктурируем таблицу, вызывая хеш-функцию с новыми параметрами. Алгоритм для рестуктуризации хеш-функции заключается в том, что мы берем размерности степени двойки и если данная размерность не подходит то мы берем следующую степень двойки. И так пока нас не устроит введенное количество сравнений.

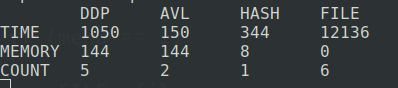
Реструктурированная таблица:



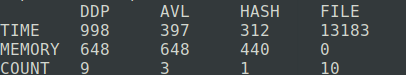
**Анализ эффективности (по памяти и времени):**

Хеш-таблица длины 11.

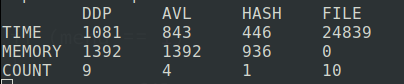
6 элементов в файле:



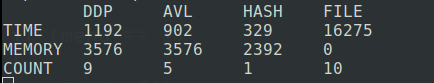
27 элементов в файле:



60 элементов в файле:



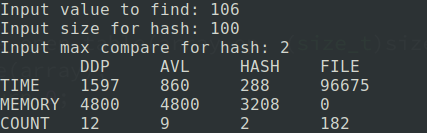
200 элементов в файле:



**Вывод:**

Самой оптимальной структурой по времени для поиска данных является хеш-таблица. Подобранные хорошо изначально параметры для хеш-таблицы позволяет уменьшить временные затраты на ее перегенерацию и в дальнейшем уменьшить среднее количество сравнений при поиске. Поиск в сбалансированном дереве быстрее, чем в обычном.

В файле из 200 чисел хеш выигрывает в 3 раза по времени, в 1,5 раза по памяти и в 4,5 раза по количеству итераций.



**Контрольные вопросы:**

**1.Что такое дерево?**

Дерево – нелинейная структура данных, которая используется для представления иерархических связей «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 (№ 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 стандартные операции возможны над деревьями?**

Обход, поиск, добавление и удаление элемента.

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

Дерево двоичного поиска – дерево, в котором все левые потомки «моложе» предка, а все правые – «старше». Это свойство выполняется для любого узла, включая корень.

**5. Чем отличается идеально сбалансированное дерево от АВЛ дерева?**

АВЛ-дерево — сбалансированное по высоте двоичное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.

Идеально сбалансированное: для каждой его вершины количество вершин в левом и правом поддереве различаются не более чем на 1.

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

Алгоритм поиска одинаковый.

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

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

**8. Что такое коллизии? Каковы методы их устранения.**

Может возникнуть ситуация, когда разным ключам соответствует одно значение хеш-функции, то есть, когда h(K1) = h(K2), в то время как K1 ≠ K2. Такая ситуация называется коллизией. Для устранения коллизий нужно выбрать другую хеш функцию (или эту же с другими параметрами) и реструктуризировать таблицу.

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

При большом количестве коллизий.