|  |  |
| --- | --- |
|  | **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 ПО ПРЕДМЕТУ "ТИПЫ И СТРУКТУРЫ ДАННЫХ" №6**

Студент \_Золотухин Алексей Вячеславович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_\_\_ИУ7-34Б\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_Золотухин А. В.\_\_\_\_

*подпись, дата фамилия, и.о.*

Принял \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_Силантьева А. В. \_

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва, 2021**

**Задача:**

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

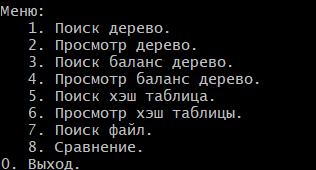
таблицы, выбрав другую функцию.

**Входные данные**

Число сравнений при превышении которого реструктуризируется хэш-таблица.

Файл со словами.  
Программа создаёт 3 структуры данных на основе файла. После чего на входе программы вводится номер команды, интересующей пользователя – 2 или 4 — вывод деревьев на экран, 6 — вывод хеш-таблицы, 1, 3, 5, 7 — поиск элемента в 4 структурах данных, 8 — анализ эффективности созданных структур данных.

Искомое слово.



**Выходные данные**

В результате программа выводит результат выполнения выбранного пункта.

Реализованная программа представляет собой консольное приложение, предоставляющее пользователю возможность изучить работу структур, а также сравнить их эффективность.

**Структуры данных**

В программе было реализовано 3 структуры данных: двоичное дерево поиска, хеш-таблица, Файл.

#define STR\_LEN 50

**ДДП:**

typedef struct tree

{

    char s[STR\_LEN];

    int height;

    struct tree \*left;

    struct tree \*right;

} tree\_t;

**Хеш-таблица:**

typedef struct hash

{

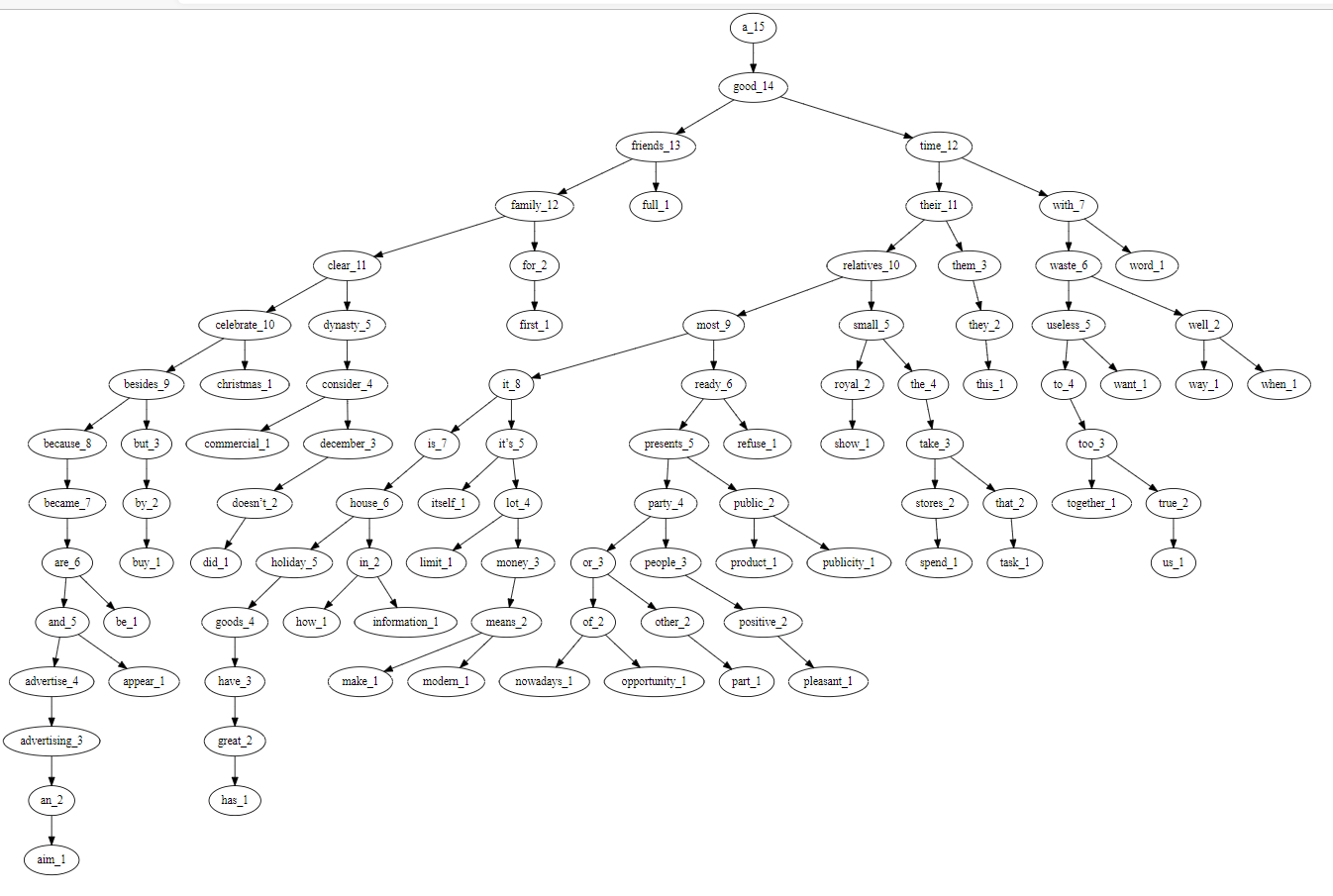
    char s[STR\_LEN];

    struct hash \*next;

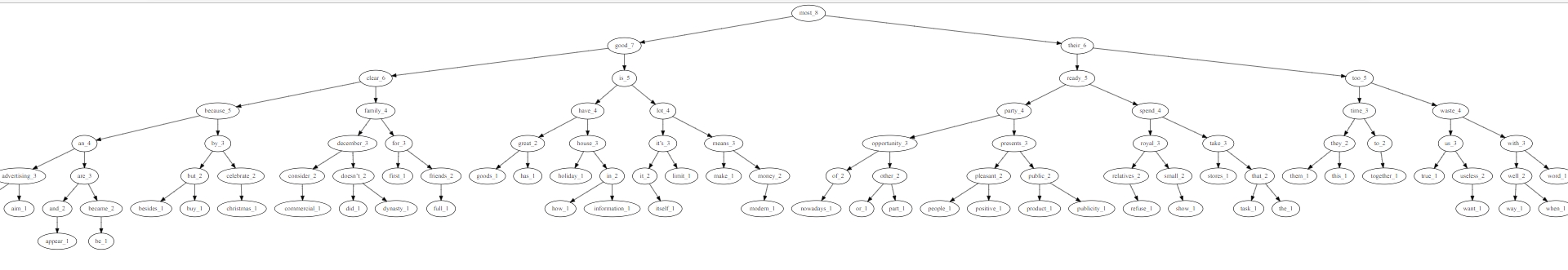
}   hash\_t;

**Тесты**

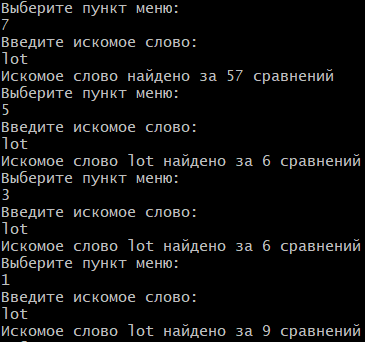
1. Вывод бинарного дерева



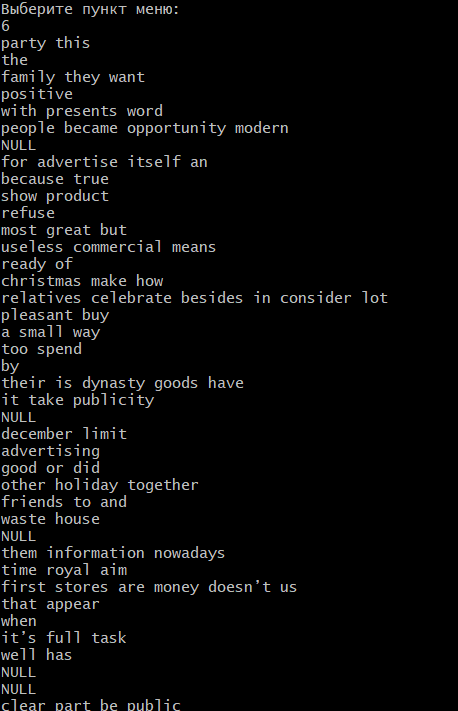
1. Вывод АВЛ-дерева



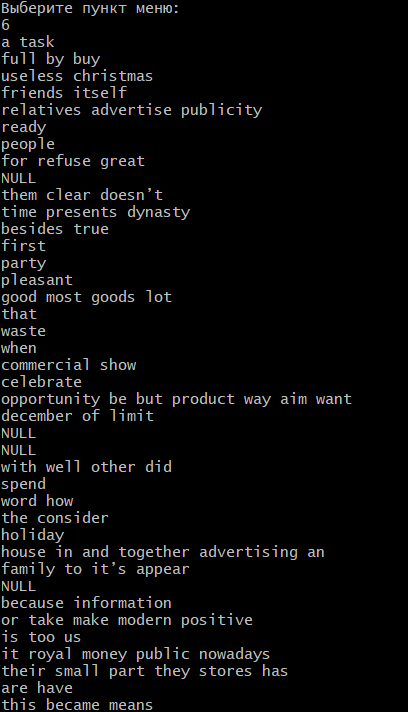
1. Поиск элемента в четырех структурах



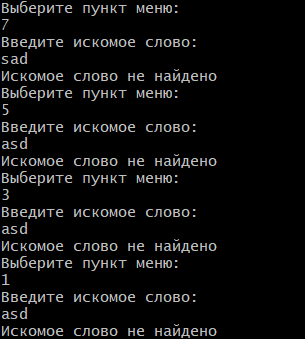
1. Исходная хэш-таблица



1. Реструктуризация хеш-таблицы

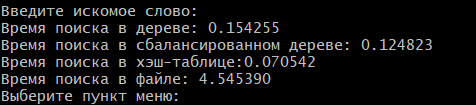


1. Поиск несуществующего элемента



**Анализ эффективности**

Операция поиска

****

|  |  |  |
| --- | --- | --- |
| Структура данных | Время  (в микросекундах) | Эффективность относительно верхнего  (в процентах) |
| Файл | 4.545 | - |
| Бинарное дерево | 0.154 | 2851% |
| АВЛ-дерево | 0.124 | 24% |
| Хеш-таблица | 0.071 | 75% |
| Реструк. Хэш-таблица | 0.06 | 20% |

Сравнения

|  |  |  |
| --- | --- | --- |
| Структура данных | Количество сравнений | Эффективность относительно верхнего  (в процентах) |
| Файл | 47.5 | - |
| Бинарное дерево | 8.5 | 458% |
| АВЛ-дерево | 5.77 | 47% |
| Хеш-таблица | 3.34 | 72% |
| Реструк. Хэш-таблица | 2.21 | 51% |

Затраты по памяти

|  |  |  |
| --- | --- | --- |
| Структура данных | Память (в байтах) | Эффективность  относительно верхнего  (в процентах) |
| Файл | 670 | - |
| Бинарное дерево | 6768 | -910% |
| АВЛ-дерево | 6768 | 0% |
| Хеш-таблица | 6248 | 8% |
| Реструк. Хэш-таблица | 6344 | -2% |

**Заключение**

В результате работы были реализованы четыре структуры данных.

С точки зрения временных затрат хеш-таблица оказалась эффективней остальных на поиске. АВЛ-дерево «обыгрывает» ДДП при поиске.

Файл обладает минимальными затратами памяти. Таким образом, выбор структуры зависит от поставленной задачи и средств, которые программист готов на неё затратить.

**Ответы на контрольные 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?***

В памяти деревья можно представить в виде связей с предками (еще их называют родителями); связного списка потомков (сыновей) или структуры данных.

***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?***

Если при добавлении узлов в дерево мы будем их равномерно располагать слева и справа, то получится дерево, у которого число вершин в левом и правом поддеревьях отличается не более, чем на единицу. Такое дерево называется идеально сбалансированным.   
 Адельсон-Вельский и Ландис сформулировали менее жесткий критерий сбалансированности таким образом: двоичное дерево называется сбалансированным, если у каждого узла дерева высота двух поддеревьев отличается не более чем на единицу. Такое дерево называется АВЛ-деревом.

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

Абсолютно идентичен.

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

То есть необходимо создать такую функцию, по которой можно вычислить этот индекс. Такая функция называется хеш-функцией (от англ. To hash - крошить, рубить) и она ставит в соответствие каждому ключу ki индекс ячейки j, где расположен элемент с этим ключом, таким образом: h(ki) = j, если j=(1,m), где j принадлежит множеству от 1 до m, а m. – размерность массива.

***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.

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

Когда хэш-функция дает слишком много коллизий, и для поиска нужного элемента нам приходится все больше и больше сравнивать.

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

См. п. Анализ эффективности