Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 1

По дисциплине ЯПИС

За шестой семестр

Тема: «Работа с таблицей идентификаторов»

Вариант 9

Выполнил:

Студент 3 курса

Группы ИИ-16

Журавлёв

Проверил:

Слинко

Брест, 2021

*Цель работы:* изучить основные методы организации таблиц идентификаторов, получить представление о преимуществах и недостатках, присущих различным методам организации таблиц символов (идентификаторов).

Для выполнения лабораторной работы требуется написать программу, которая получает на входе набор идентификаторов, организует таблицу по заданному методу и позволяет осуществить поиск идентификатора в этой таблице.

Список идентификаторов задан в виде текстового файла. Длина идентификаторов ограничена 32 символами.

9. Таблица строится по методу дерева. Программа должна подсчитывать число выполненных сравнений при поиске.

Код программы:

**Main:**

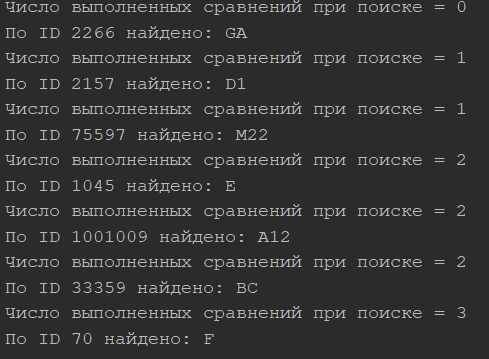
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileReader;  
  
public class Main {  
 public static void main(String[] args) throws Exception {  
 String delimiter = ", ";  
  
 HashTable hashTable = new HashTable();  
  
 BufferedReader bufferedReader = new BufferedReader(new FileReader(new File("data.txt")));  
 String data = bufferedReader.readLine();  
  
 String[] strings = data.split(delimiter);  
 for (String el : strings) {  
 hashTable.add(el);  
 }  
  
 for (String el: strings) {  
 System.*out*.println("По ID " + el.hashCode() + " найдено: " + hashTable.find(el.hashCode()));  
 }  
 }  
}

**HashTable:**

public class HashTable {  
  
 private Tree tree;  
  
 public HashTable() {  
 tree = new Tree();  
 }  
  
 public void add(String element){  
 tree.add(element);  
 }  
 public String find(Integer hash){  
 return tree.find(hash);  
 }  
  
}

**Tree:**

public class Tree {  
  
 private Node root;  
  
 public void add(String data){  
 root = add(root, data, data.hashCode());  
 }  
  
 private Node add(Node node, String data, Integer hash){  
 if(node == null){  
 node = new Node(hash, data);  
 return node;  
 }  
 if(hash < node.hash){  
 node.left = add(node.left, data, hash);  
 }  
 else if(hash > node.hash){  
 node.right = add(node.right, data, hash);  
 }  
 else{  
 throw new RuntimeException(String.*format*("Element with hash %d already exist", hash));  
 }  
 return node;  
 }  
  
 public String find(Integer hash) {  
 return find(root, hash, 0);  
 }  
  
 private String find(Node node, Integer hash, Integer count) {  
 if(node == null) {  
 return "";  
 }  
 if(node.hash.equals(hash)) {  
 System.*out*.println("Число выполненных сравнений при поиске = " + count);  
 return node.data;  
 }  
 else if(hash < node.hash){  
 return find(node.left, hash, count + 1);  
 }  
 else{  
 return find(node.right, hash, count + 1);  
 }  
 }  
  
 private class Node {  
 Integer hash;  
 String data;  
 Node left;  
 Node right;  
  
 public Node(Integer hash, String data) {  
 this.hash = hash;  
 this.data = data;  
 }  
 }  
}



Вывод: научился строить хэш-таблицы; разобрал и реализовал хэш-таблицу по методу дерева.