**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«Московский энергетический институт»**

**Кафедра математического и компьютерного моделирования**

**«Математическое обеспечение ЭВМ»**

Лабораторная работа №1

**"Работа с таблицей идентификаторов"**

Вариант №11

Выполнил: Симаков А.М.

Группа: А-16-20

Преподаватель: Князев А.В.

**Москва 2021г**

**Постановка задачи**

Общая

Разработать функции для включения описания идентификатора в таблицу, поиска по имени и исключения описания идентификатора из таблицы.

Описание идентификатора состоит из имени идентификатора и его атрибута.

Разработать программу, демонстрирующую использование указанных

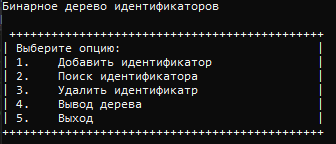
функций. Программа должна быть разработана как консольное приложение на языке C# в среде Visual Studio.

Индивидуальная

Способ представления таблицы – бинарное дерево.  
Метод поиска – бинарный поиск.

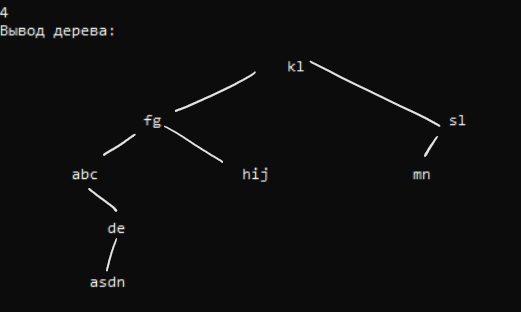
**Описание работы программы**

В консоль выводится окно опций.

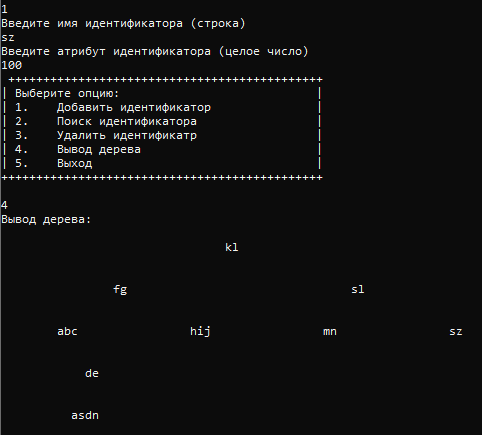


Пользователь выбирает желаемую опцию.

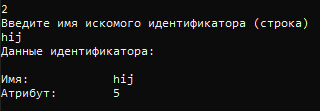
Выбор 4 – вывод дерева (на картине изображены вспомогательные ребра)



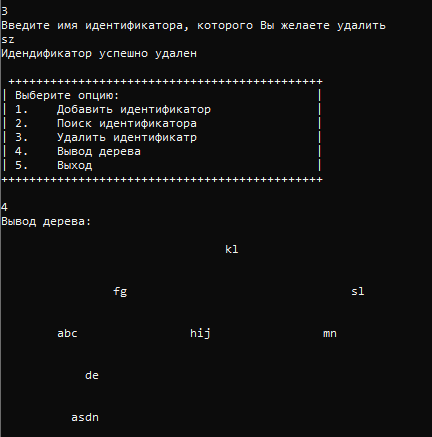
Выбор 1 – добавление нового идентификатора.



Выбор 2 – поиск идентификатора.



Выбор 3 – удаление идентификатора.



Выбор 6 – Выход.

**Алгоритмы выполнения основных операций**

Алгоритм добавления идентификатора

Если дерево пусто – новый узел корневой.

Опускаем флаг

Пока флаг опущен:

Если идентификатор текущего узла равен искомому

То сообщаем, что такой идентификатор уже есть в дереве

Если идентификатор текущего узла меньше искомого

Если левая ветвь тупиковая, то добавляем новый узел слева, поднимаем флаг

Иначе переходим по левой ветви

Если идентификатор текущего узла больше искомого

Если правая ветвь тупиковая, то добавляем новый узел справа, поднимаем флаг

Иначе переходим по правой ветви

Алгоритм бинарного поиска имени идентификатора

Если дерево пусто – выводим сообщение об этом.

Пока мы ничего не возвращаем:

Если идентификатор текущего узла равен искомому

То возвращаем текущий узел

Если идентификатор текущего узла меньше искомого

То переходим по левой ветви

Если идентификатор текущего узла больше искомого

То переходим по правой ветви

Если ветви текущего узла тупиковые возвращаем null

Алгоритм удаления узла

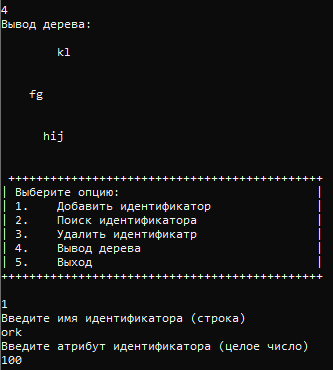
Запускаем алгоритм поиска имени идентификатора, возвращающий узел node

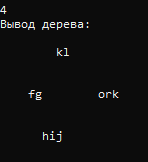
Если левая и правая ветви тупиковые, то ставим предыдущему null на такой ветви, что введет к текущей

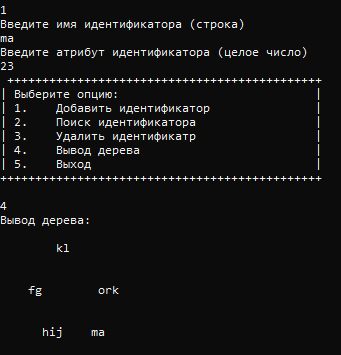
Иначе ставим предыдущему null на такой ветви, что введет к текущей и переходим в самый левый узел после перехода по правой ветви и прикрепляем к ней переход по левой ветви

**Тесты**

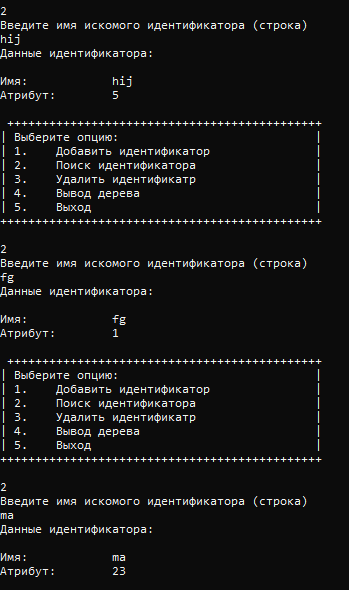
1. Добавление узлов



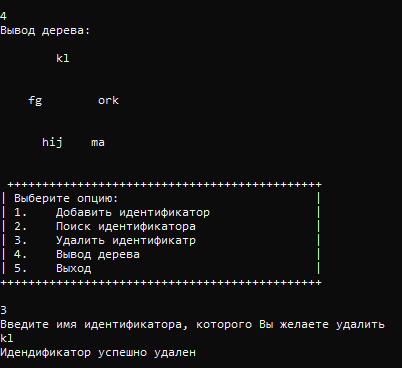


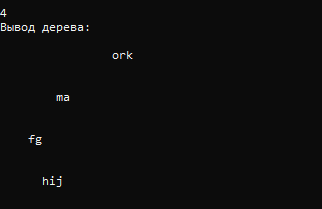


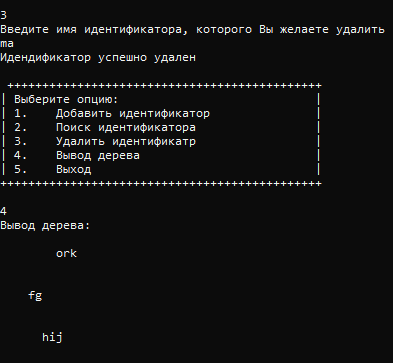
1. Поиск узлов



1. Удаление узлов

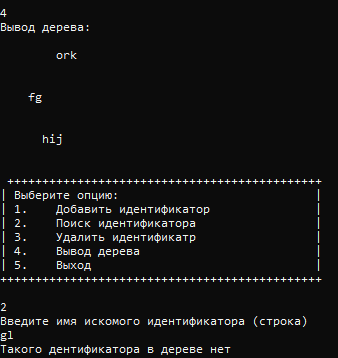


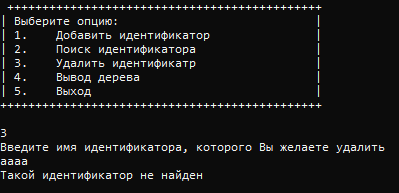




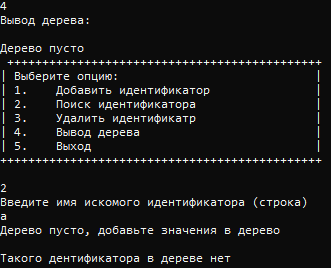
1. Ошибки

Поиск и удаление





Пустое дерево



**Листинг программы**

Program

using System;

namespace Lab1\_D

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Бинарное дерево идентификаторов\n");

string choice;

BinaryThree three = new BinaryThree();

//three.Add("kl", 0);

//three.Add("fg", 1);

//three.Add("hij", 5);

for (; ; )

{

Console.WriteLine(" +++++++++++++++++++++++++++++++++++++++++++++");

Console.WriteLine("| Выберите опцию: |");

Console.WriteLine("| 1.\tДобавить идентификатор |");

Console.WriteLine("| 2.\tПоиск идентификатора |");

Console.WriteLine("| 3.\tУдалить идентификатр |");

Console.WriteLine("| 4.\tВывод дерева |");

Console.WriteLine("| 5.\tВыход |");

Console.WriteLine("++++++++++++++++++++++++++++++++++++++++++++++\n");

choice = Console.ReadLine();

string newId;

int atribute;

switch (choice)

{

case "1":

Console.WriteLine("Введите имя идентификатора (строка)");

newId = Console.ReadLine();

Console.WriteLine("Введите атрибут идентификатора (целое число)");

atribute = Convert.ToInt32(Console.ReadLine());

three.Add(newId, atribute);

break;

case "2":

Console.WriteLine("Введите имя искомого идентификатора (строка)");

newId = Console.ReadLine();

Node node = three.Search(newId);

if (node == null)

Console.WriteLine("Такого дентификатора в дереве нет");

else

node.Output();

break;

case "3":

Console.WriteLine("Введите имя идентификатора, которого Вы желаете удалить");

newId = Console.ReadLine();

three.DeleteNode(newId);

break;

case "4":

Console.WriteLine("Вывод дерева:\n");

three.ThreeOutput();

break;

case "5":

return;

}

}

}

}

}

Node.cs

using System;

namespace Lab1\_D

{

class Node

{

public Node() { }

public Node(string id, int atribute)

{

ID = id;

Atribute = atribute;

}

public string getID()

{

return ID;

}

public void changeID(string id)

{

ID = id;

}

public int getAtribute()

{

return Atribute;

}

public Node getLeft()

{

return left;

}

public void changeLeft(Node tnode)

{

left = tnode;

}

public Node getRight()

{

return right;

}

public void changeRight(Node tnode)

{

right = tnode;

}

public void Output()

{

Console.WriteLine("Данные идентификатора:\n");

Console.WriteLine("Имя:\t\t" + ID);

Console.WriteLine("Атрибут:\t" + Atribute + "\n");

}

private string ID;

private int Atribute;

private Node left;

private Node right;

}

}

BinaryThree

using System;

namespace Lab1\_D

{

class BinaryThree

{

public void Add(string ID, int atribute)

{

Node tnode = new Node(ID, atribute);

if (this.root == null)

{

root = tnode;

}

else

{

Node search = root;

bool f = false;

while (!f)

{

if (search.getID() == ID)

{

Console.WriteLine("Такой элемент уже есть в дереве\n");

return;

}

if (String.Compare(tnode.getID(), search.getID()) < 0)

{

if (search.getLeft() == null)

{

search.changeLeft(tnode);

f = true;

}

else

{

search = search.getLeft();

}

}

else

{

if (search.getRight() == null)

{

search.changeRight(tnode);

f = true;

}

else

{

search = search.getRight();

}

}

}

}

}

public Node Search(string id)

{

if (root == null)

{

Console.WriteLine("Дерево пусто, добавьте значения в дерево\n");

return null;

}

Node node = root;

for (; ; )

{

if (String.Compare(node.getID(), id) == 0)

return node;

if (node.getLeft() == null && node.getRight() == null)

return null;

if (node.getLeft() != null)

if (String.Compare(node.getID(), id) > 0)

node = node.getLeft();

if (node.getRight() != null)

if (String.Compare(node.getID(), id) < 0)

node = node.getRight();

}

}

private int rec(Node tnode)

{

int r = 1;

int k = 1;

if (tnode.getLeft() != null)

{

r += rec(tnode.getLeft());

}

if (tnode.getRight() != null)

{

k += rec(tnode.getRight());

}

if (r > k)

return r;

else

return k;

}

public int Length()

{

if (root == null)

return 0;

return rec(root);

}

private void writeNode(Node tnode, int k)

{

while (k != 0)

{

Console.Write(" ");

k--;

}

Console.Write(tnode.getID());

}

private void writeEmpty(int k)

{

while (k != 0)

{

Console.Write(" ");

k--;

}

}

public void ThreeOutput()

{

if (root == null)

{

Console.WriteLine("Дерево пусто");

return;

}

int k = 0;

int n = 1;

int length = this.Length();

Node[] mass0 = new Node[30];

Node[] mass = new Node[30];

writeNode(root, Convert.ToInt32(Math.Pow(2, length)));

mass0[0] = root;

Console.WriteLine("\n\n");

while (n < length)

{

k = Convert.ToInt32(Math.Pow(2, n));

int i = 0;

int j = 0;

while (i < k)

{

if (mass0[j] == null)

mass[i] = null;

else

mass[i] = mass0[j].getLeft();

i++;

if (mass0[j] == null)

mass[i] = null;

else

mass[i] = mass0[j].getRight();

i++;

j++;

}

for (int m = 0; m < k; m++)

mass0[m] = mass[m];

if (mass[0] != null)

writeNode(mass[0], Convert.ToInt32(Math.Pow(2, length - n)));

else

writeEmpty(Convert.ToInt32(Math.Pow(2, length - n)));

for (int m = 1; m < k; m++)

{

if (mass[m] != null)

writeNode(mass[m], Convert.ToInt32(Math.Pow(2, length - n + 1)));

else

writeEmpty(Convert.ToInt32(Math.Pow(2, length - n + 1)));

}

Console.WriteLine("\n\n");

n++;

}

}

public void DeleteNode(string id)

{

if (root == null)

{

Console.WriteLine("Дерево пусто, добавьте значения в дерево\n");

return;

}

Node change = new Node();

if (root.getID() == id)

{

if (root.getLeft() == null && root.getRight() == null)

{

this.Clean();

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

if (root.getLeft() == null)

{

root = root.getRight();

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

if (root.getRight() == null)

{

root = root.getLeft();

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

change = root.getRight();

while (change.getLeft() != null)

change = change.getLeft();

change.changeLeft(root.getLeft());

root = root.getRight();

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

Node search = root;

Node prevsaerch = new Node();

bool branch = false; //false = left; true = right

while (id != search.getID())

{

prevsaerch = search;

if (String.Compare(id, search.getID()) < 0)

{

if (search.getLeft() == null)

{

Console.WriteLine("Такой идентификатор не найден\n");

return;

}

else

{

search = search.getLeft();

branch = false;

}

}

else

{

if (search.getRight() == null)

{

Console.WriteLine("Такой идентификатор не найден\n");

return;

}

else

{

search = search.getRight();

branch = true;

}

}

}

if (search.getLeft() == null && search.getRight() == null)

{

if (branch)

prevsaerch.changeRight(null);

else

prevsaerch.changeLeft(null);

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

if (search.getLeft() == null)

{

if (branch)

prevsaerch.changeRight(search.getRight());

else

prevsaerch.changeLeft(search.getRight());

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

if (search.getRight() == null)

{

if (branch)

prevsaerch.changeRight(search.getLeft());

else

prevsaerch.changeLeft(search.getLeft());

Console.WriteLine("Идендификатор успешно удален\n");

return;

}

change = search.getRight();

while (change.getLeft() != null)

change = change.getLeft();

change.changeLeft(search.getLeft());

if (branch)

prevsaerch.changeRight(search.getRight());

else

prevsaerch.changeLeft(search.getRight());

Console.WriteLine("Идендификатор успешно удален\n");

}

public void Clean()

{

root = null;

}

private Node root;

}

}