НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування та спеціалізованих комп‘ютерних систем

**Лабораторна робота №4  
*З дисципліни: «Операційні системи»***

«**Файлові системи**»

Студентка групи КВ-42

Брикалова Вікторія

Перевірив(ла):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2017

**Завдання**

1. Написати програму, що моделює роботу складових заданої файлової системи згідно варіанта (перелік варіантів представлений нижче у табл.4.3).

Вхідні дані студент задає самостійно з урахуванням особливостей індивідуального варіанта завдання.

1. Зробити візуалізацію роботи програми і кінцевих результатів на різних наборах вхідних даних.

**Варіант 2:**

**Побудова таблиці ідентифікаторів за методом бінарного дерева**

Дерево повинно бути ідеально сбалансованим і впорядкованим. Забезпечити можливість додаткового включення та виключення ідентифікаторів при збереженні сбалансованості.

Забезпечити можливість пошуку заданого ідентифікатора з роздруківкою шляху пошуку та відображенням вигляду дерева.

У якості ідентифікаторів використовувати три букви.

**Текст програми**

**Identifier.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OSLab4

{

class Identifier : IComparable<Identifier>

{

public string Value { get; private set; }

public Identifier(string value)

{

if (ValidateIdentifier(value) == false)

throw new ArgumentException();

Value = value;

}

private bool ValidateIdentifier(string identifier)

{

if (identifier.Length > 3)

return false;

char[] tmp = identifier.ToCharArray();

if (Char.IsDigit(tmp[0])) return false;

if (Char.IsDigit(tmp[1])) return false;

if (Char.IsDigit(tmp[2])) return false;

return true;

}

public override string ToString()

{

return Value.ToString();

}

#region IComparable

public int CompareTo(Identifier other)

{

return Value.CompareTo(other.Value);

}

#endregion

}

}

**Node.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using MindFusion.Diagramming.Wpf;

namespace OSLab4.AVLTree

{

public class AVLNode<TKey, TValue>

{

public AVLNode(TKey key, TValue value)

{

this.Key = key;

this.Value = value;

this.LeftChild = null;

this.RightChild = null;

this.Parent = null;

this.Balance = 0;

}

public AVLNode(TKey key, TValue value, AVLNode<TKey, TValue> parent)

: this(key, value)

{

this.Parent = parent;

}

public AVLNode(AVLNode<TKey, TValue> node)

: this(node.Key, node.Value, node.Parent)

{

this.LeftChild = node.LeftChild;

this.RightChild = node.RightChild;

}

public ShapeNode RBuildTree(Diagram parent)

{

ShapeNode node = new ShapeNode(parent)

{

Bounds = new Rect(new Point(), MainWindow.NodeSize),

Text = Key.ToString(),

Tag = this

};

parent.Nodes.Add(node);

if (LeftChild != null)

{

var childNode = LeftChild.RBuildTree(parent);

childNode.AttachTo(node, AttachToNode.TopLeft);

var link = new DiagramLink(parent, node, childNode);

parent.Links.Add(link);

}

if (RightChild != null)

{

var childNode = RightChild.RBuildTree(parent);

childNode.AttachTo(node, AttachToNode.TopRight);

var link = new DiagramLink(parent, node, childNode);

parent.Links.Add(link);

}

return node;

}

public TKey Key { get; set; }

public TValue Value { get; set; }

public AVLNode<TKey, TValue> Parent { get; set; }

public AVLNode<TKey, TValue> LeftChild { get; set; }

public AVLNode<TKey, TValue> RightChild { get; set; }

public int Balance { get; set; }

public override int GetHashCode()

{

return (this.Key.GetHashCode() << 16) | (this.Value.GetHashCode() & 0xFFFF);

}

//Two nodes are equal if their both keys and values match.

public override bool Equals(object obj)

{

AVLNode<TKey, TValue> node = obj as AVLNode<TKey, TValue>;

if (node == null)

{

return false;

}

else

{

if (this.Key.Equals(node.Key) && this.Value.Equals(node.Value))

{

return true;

}

else

{

return false;

}

}

}

public override string ToString()

{

return String.Format("({0} {1})", this.Key, this.Value);

}

}

}

**Tree.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using MindFusion.Diagramming.Wpf;

namespace OSLab4.AVLTree

{

public class AVLNode<TKey, TValue>

{

public AVLNode(TKey key, TValue value)

{

this.Key = key;

this.Value = value;

this.LeftChild = null;

this.RightChild = null;

this.Parent = null;

this.Balance = 0;

}

public AVLNode(TKey key, TValue value, AVLNode<TKey, TValue> parent)

: this(key, value)

{

this.Parent = parent;

}

public AVLNode(AVLNode<TKey, TValue> node)

: this(node.Key, node.Value, node.Parent)

{

this.LeftChild = node.LeftChild;

this.RightChild = node.RightChild;

}

public ShapeNode RBuildTree(Diagram parent)

{

ShapeNode node = new ShapeNode(parent)

{

Bounds = new Rect(new Point(), MainWindow.NodeSize),

Text = Key.ToString(),

Tag = this

};

parent.Nodes.Add(node);

if (LeftChild != null)

{

var childNode = LeftChild.RBuildTree(parent);

childNode.AttachTo(node, AttachToNode.TopLeft);

var link = new DiagramLink(parent, node, childNode);

parent.Links.Add(link);

}

if (RightChild != null)

{

var childNode = RightChild.RBuildTree(parent);

childNode.AttachTo(node, AttachToNode.TopRight);

var link = new DiagramLink(parent, node, childNode);

parent.Links.Add(link);

}

return node;

}

public TKey Key { get; set; }

public TValue Value { get; set; }

public AVLNode<TKey, TValue> Parent { get; set; }

public AVLNode<TKey, TValue> LeftChild { get; set; }

public AVLNode<TKey, TValue> RightChild { get; set; }

public int Balance { get; set; }

public override int GetHashCode()

{

return (this.Key.GetHashCode() << 16) | (this.Value.GetHashCode() & 0xFFFF);

}

//Two nodes are equal if their both keys and values match.

public override bool Equals(object obj)

{

AVLNode<TKey, TValue> node = obj as AVLNode<TKey, TValue>;

if (node == null)

{

return false;

}

else

{

if (this.Key.Equals(node.Key) && this.Value.Equals(node.Value))

{

return true;

}

else

{

return false;

}

}

}

public override string ToString()

{

return String.Format("({0} {1})", this.Key, this.Value);

}

}

}

**Тест**

