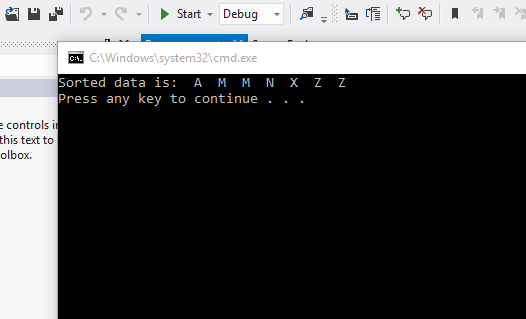
Student: Brian Johnston

Class: COP2362

Instructor Hamilton

Assignment 4-3

Print screen



Code – Binary Tree

Tree.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace BinaryTree

{

public class Tree<TItem> where TItem : IComparable<TItem>

{

public TItem NodeData { get; set; }

public Tree<TItem> LeftTree { get; set; }

public Tree<TItem> RightTree { get; set; }

public Tree(TItem nodeValue)

{

this.NodeData = nodeValue;

this.LeftTree = null;

this.RightTree = null;

}

public void Insert(TItem newItem)

{

TItem currentNodeValue = this.NodeData;

if (currentNodeValue.CompareTo(newItem) > 0)

{

// Insert the item into the left subtree

if (this.LeftTree == null)

{

this.LeftTree = new Tree<TItem>(newItem);

}

else

{

this.LeftTree.Insert(newItem);

}

}

else

{

// Insert the item into the right subtree

if (this.RightTree == null)

{

this.RightTree = new Tree<TItem>(newItem);

}

else

{

this.RightTree.Insert(newItem);

}

}

}

public string WalkTree()

{

string result = "";

if (this.LeftTree != null)

{

result = this.LeftTree.WalkTree();

}

result += String.Format(" {0} ", this.NodeData.ToString());

if (this.RightTree != null)

{

result += this.RightTree.WalkTree();

}

return result;

}

}

}

BinaryTreeTEst

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using BinaryTree;

namespace BinaryTreeTest

{

class Program

{

static void Main(string[] args)

{

Tree<int> tree1 =new Tree<int>(10);

tree1.Insert(5);

tree1.Insert(11);

tree1.Insert(5);

tree1.Insert(-12);

tree1.Insert(15);

tree1.Insert(0);

tree1.Insert(14);

tree1.Insert(-8);

tree1.Insert(10);

tree1.Insert(8);

tree1.Insert(8);

string sortedData = tree1.WalkTree();

Console.WriteLine("Sorted data is: {0}", sortedData);

Tree<string> tree2 = new Tree<string>("Hello");

tree2.Insert("World");

tree2.Insert("How");

tree2.Insert("Are");

tree2.Insert("You");

tree2.Insert("Today");

tree2.Insert("I");

tree2.Insert("Hope");

tree2.Insert("You");

tree2.Insert("Are");

tree2.Insert("Feeling");

tree2.Insert("Well");

tree2.Insert("!");

sortedData = tree2.WalkTree();

Console.WriteLine("Sorted data is: {0}", sortedData);

}

}

}

Build Tree

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using BinaryTree;

namespace BuildTree

{

class Program

{

static void Main(string[] args)

{

Tree<char> charTree = null;

InsertIntoTree<char>(ref charTree, 'M', 'X', 'A', 'M', 'Z', 'Z', 'N');

string sortedData = charTree.WalkTree();

Console.WriteLine("Sorted data is: {0}", sortedData);

}

static void InsertIntoTree<TItem>(ref Tree<TItem> tree, params TItem[] data) where TItem : IComparable<TItem>

{

foreach (TItem datum in data)

{

if (tree == null)

{

tree = new Tree<TItem>(datum);

}

else

{

tree.Insert(datum);

}

}

}

}

}