МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе №4

# по дисциплине «Теория множеств и алгоритмы на графах».

Тема: «Работа с деревьями»

Выполнил: Винникова Е.В., МВА-122

Проверил: Кузьмина Т. М.

Москва 2024

**Цель работы:** изучить библиотечные классы языка С#, предназначенные для работы с деревьями и научится их использовать в программе.

**Задание**

Общее задание.

На форму помещен пустой компонент treeView1 и 3 кнопки. При нажатии на первую кнопку в компоненте treeView1 появляется дерево, определенное вариантом. Задание состоит из 4 частей. В первой части указывается обработчик, какого события должен быть создан. Вторая часть задания определяет работу второй кнопки, третья часть задания – работу третьей кнопки. Четвертая часть задания определяет дерево, которое выводится при нажатии первой кнопки.

Первая часть задания. Работа с событиями свертывания, развертывания и выделения узлов.

2. В момент развертывания узла, его имя выводится на форму.

Вторая часть задания. Это задание определяет работу первой кнопки

1. выводит на форму имя выделенного узла.

Третья часть задания. Кнопка, которая выполняет, (если возможно, если возможно, то выводит сообщение об этом) следующее действие:

2. На форму выводит имя родителя выделенного узла.

Четвертая часть задания. Вариант дерева представлен на рис. 1.

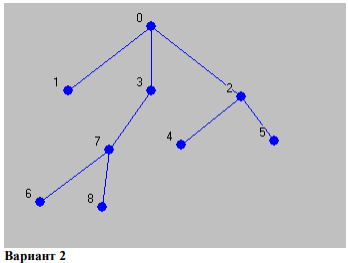


Рис. 1. Вариант дерева.

**Теория:**

Программа использует элемент управления TreeView в WPF для отображения иерархической структуры данных в виде дерева. TreeView представляет собой набор узлов, где каждый узел может содержать дочерние элементы. Основной узел называется корневым, а узлы без дочерних элементов — листьями.

События играют важную роль в управлении деревом. Например, событие Expanded отслеживает разворачивание узла, а SelectedItem позволяет получить текущий выбранный узел. Для взаимодействия с пользователем программа отображает сообщения о действиях в текстовом поле.

Динамическое добавление обработчиков событий к узлам происходит с помощью рекурсивного метода, который проходит по всем уровням дерева и применяет нужные действия. Также программа позволяет просматривать родительские элементы выбранного узла и получать информацию о текущем состоянии дерева.

Взаимодействие с пользователем обеспечивается кнопками, которые вызывают различные действия, такие как построение дерева, отображение выбранного узла и вывод родительского узла. Вся информация отображается в текстовом поле, обеспечивая наглядность работы приложения.

**Программа:**

Код:

namespace Algos4Lab2

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void btnBuildTree\_Click(object sender, RoutedEventArgs e)

{

TreeView.Items.Clear();

TreeViewItem root = new TreeViewItem { Header = "0" };

TreeViewItem node1 = new TreeViewItem { Header = "1" };

TreeViewItem node2 = new TreeViewItem { Header = "2" };

TreeViewItem node3 = new TreeViewItem { Header = "3" };

root.Items.Add(node1);

root.Items.Add(node2);

root.Items.Add(node3);

TreeViewItem node7 = new TreeViewItem { Header = "7" };

node3.Items.Add(node7);

TreeViewItem node6 = new TreeViewItem { Header = "6" };

node7.Items.Add(node6);

TreeViewItem node8 = new TreeViewItem { Header = "8" };

node7.Items.Add(node8);

node2.Items.Add(new TreeViewItem { Header = "4" });

node2.Items.Add(new TreeViewItem { Header = "5" });

AttachEvents(root);

TreeView.Items.Add(root);

}

private void AttachEvents(TreeViewItem item)

{

if (item == null) return;

item.Expanded -= TreeViewItemExpanded;

item.Expanded += TreeViewItemExpanded;

foreach (object child in item.Items)

{

if (child is TreeViewItem childItem)

{

AttachEvents(childItem);

}

}

item.Loaded += (s, e) =>

{

foreach (object child in item.Items)

{

if (child is TreeViewItem childItem)

{

AttachEvents(childItem);

}

}

};

}

private void TreeViewItemExpanded(object sender, RoutedEventArgs e)

{

if (e.OriginalSource == sender && sender is TreeViewItem item)

{

txtMessages.Text += $"Развернуто: {item.Header}\n";

}

e.Handled = true;

}

private void btnShowSelected\_Click(object sender, RoutedEventArgs e)

{

if (TreeView.SelectedItem is TreeViewItem selectedItem)

{

txtMessages.Text += $"Текущий выбранный узел: {selectedItem.Header}\n";

}

else

{

txtMessages.Text += "Узел не выбран.\n";

}

}

private void btnShowParent\_Click(object sender, RoutedEventArgs e)

{

if (TreeView.SelectedItem is TreeViewItem selectedItem)

{

TreeViewItem parent = FindParent(TreeView, selectedItem);

if (parent != null)

{

txtMessages.Text += $"Родитель узла {selectedItem.Header}: {parent.Header}\n";

}

else

{

txtMessages.Text += $"Узел {selectedItem.Header} не имеет родителя.\n";

}

}

else

{

txtMessages.Text += "Узел не выбран.\n";

}

}

private TreeViewItem FindParent(ItemsControl parent, TreeViewItem child)

{

foreach (object item in parent.Items)

{

if (item is TreeViewItem treeItem)

{

if (treeItem.Items.Contains(child))

{

return treeItem;

}

TreeViewItem found = FindParent(treeItem, child);

if (found != null)

{

return found;

}

}

}

return null;

}

}

}

Рисунок программы представлен на рис. 2.

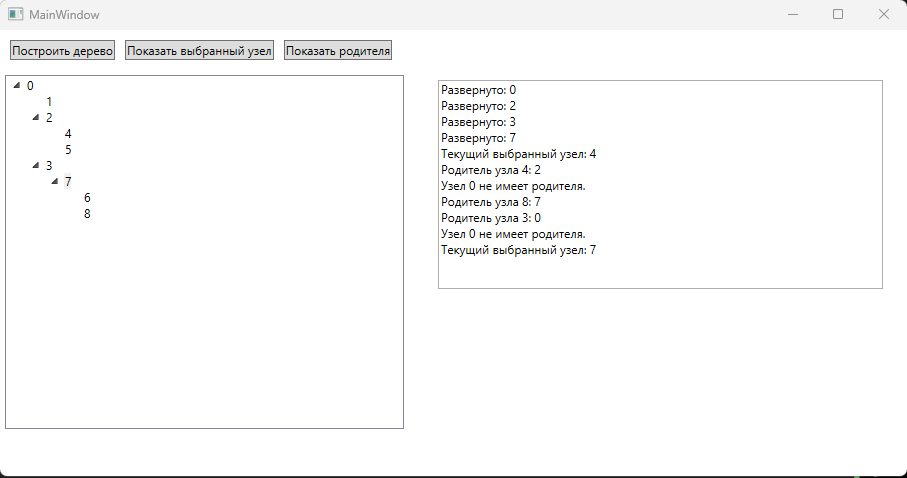


Рис.2. Программа.