МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт–Петербургский государственный университет  
аэрокосмического приборостроения»

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| преподаватель |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ  по дисциплине МДК 01.01 часть 3 |
|  |
|  |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | С326 |  |  |  | Э.С. Тигранян |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

Лабораторная работа №1

**Тема:** Создание приложения «Микропроводник».

**Цель работы**: получение практических навыков при работе с пространством имен **System.IO.**

**Задание:**

Разработать приложение «Микропроводник»,

Используя дополнительные компоненты,

* для выделенного диска необходимо выводить сведения: объем диска, свободное пространство;
* для выделенного каталога: полное название каталога, время создания каталога, корневой каталог.

При выделении файла в списке должно запускаться соответствующее приложение. Сохранить в отдельный текстовый файл имена файлов, которые открывались за последние 10 секунд работы приложения.

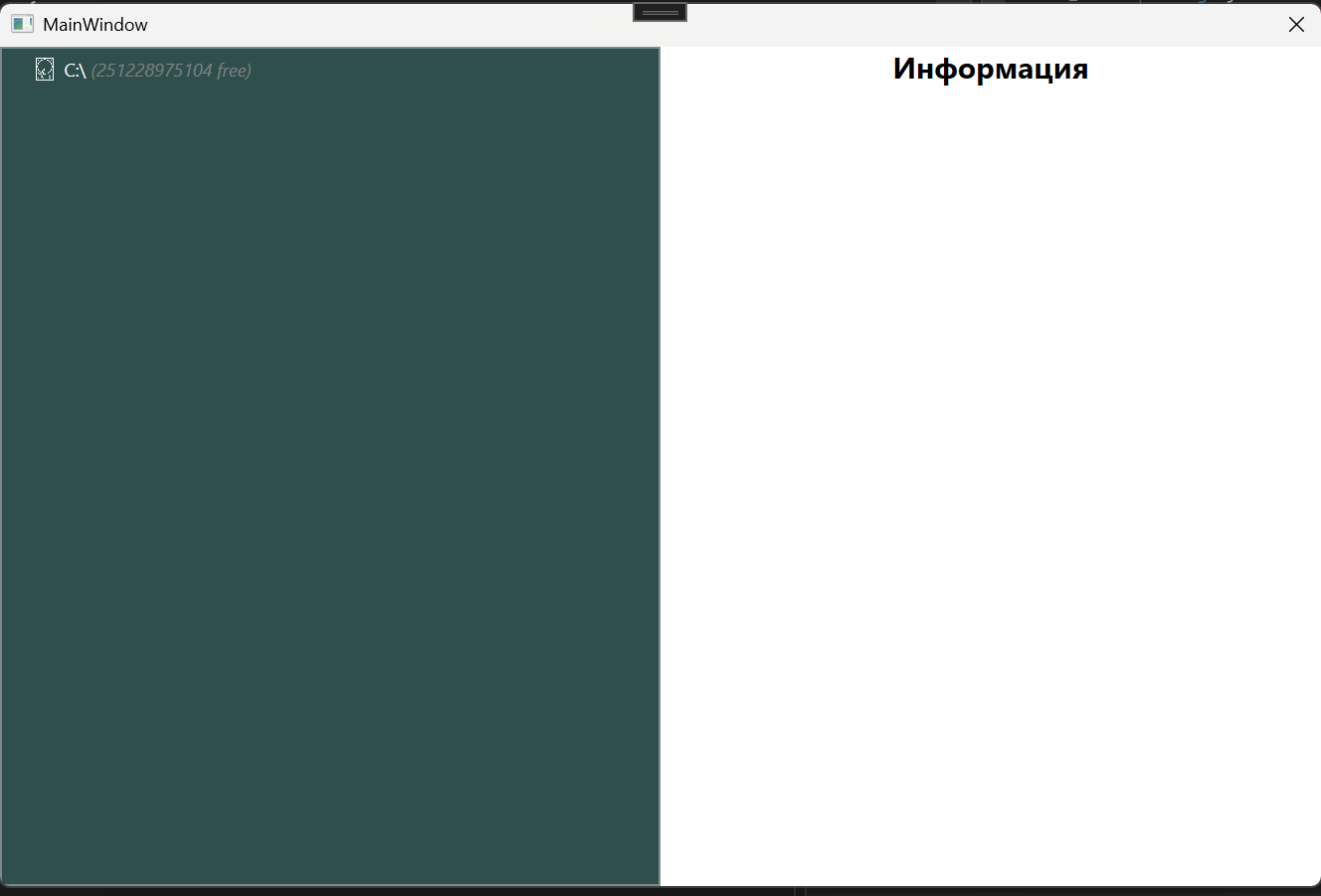


Рисунок 1 – Изначальный вид программы

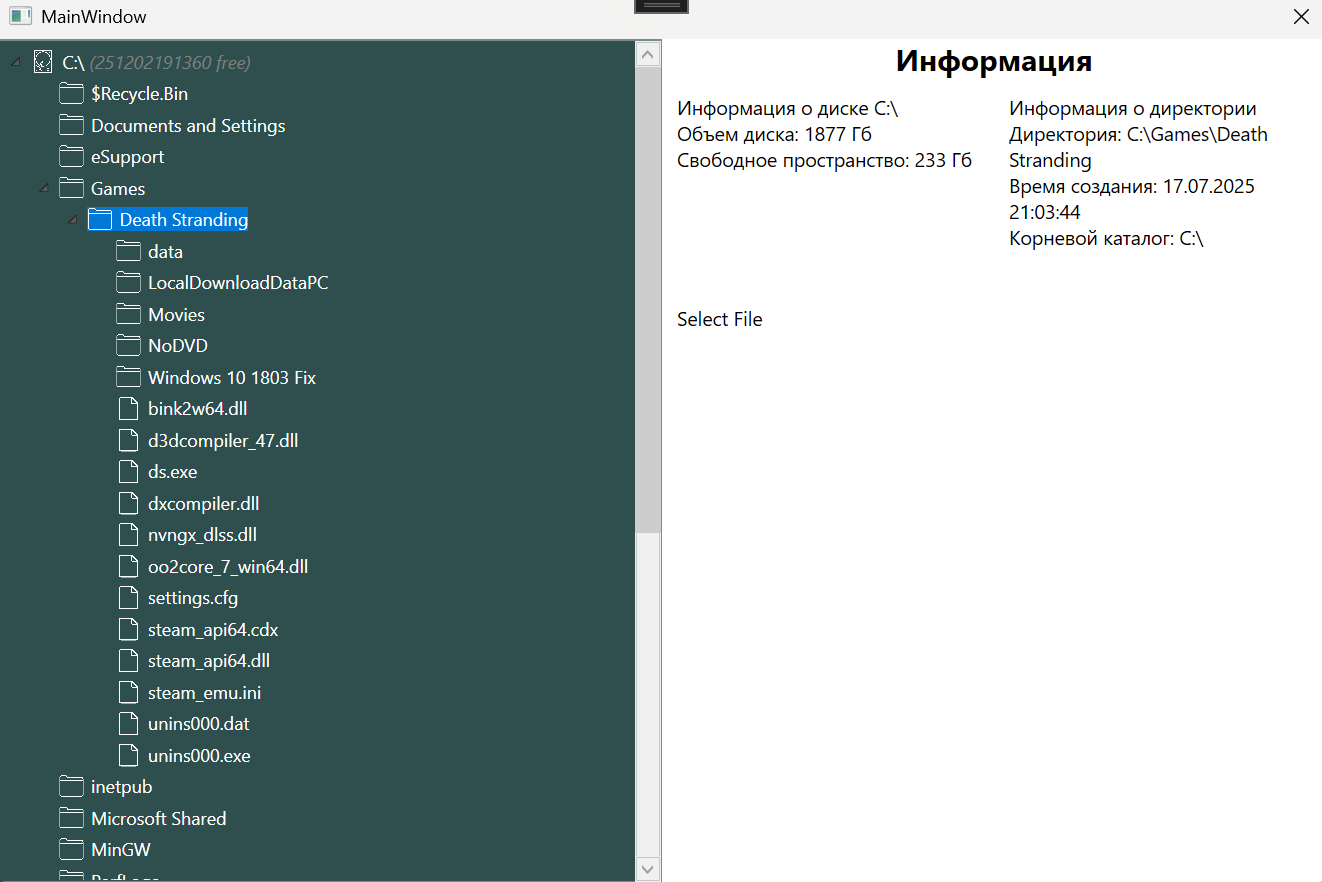


Рисунок 2 – Вид раскрытого дерева

В программе реализован предпросмотр. При веделении текстового файла идёт показ содержимого в виде текста, при веделении bin файла идёт показ содержимого в шестнадцеричном виде. При двойном нажатии на файл через операционную среду этот файл запускается. Также при выделении файла, директории, диска на правом экране выводится соответствующая информация.

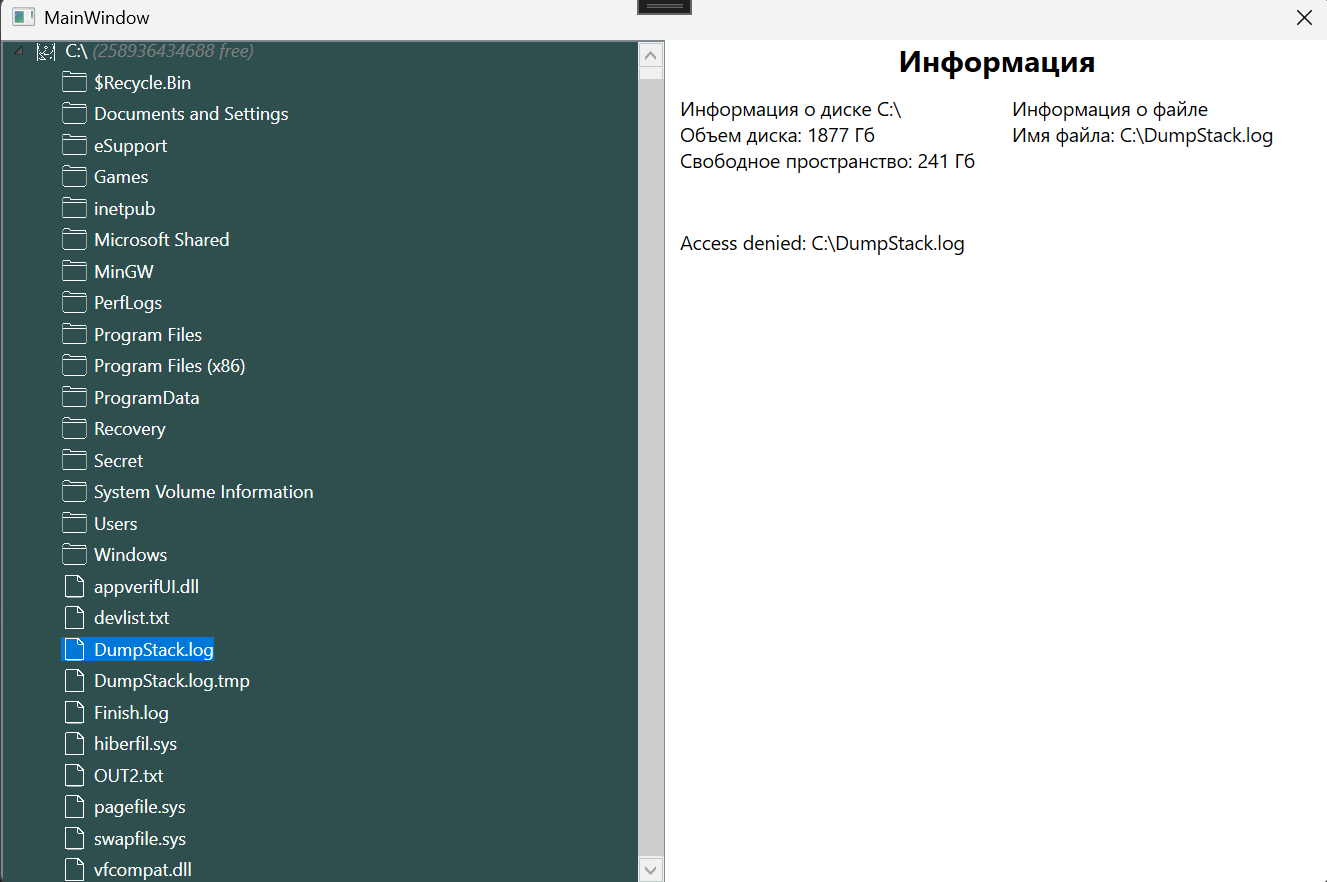


Рисунок 3 – Отказ доступа предпросмотра файла

На предпросмотр бинарного файла выделяется 500 символов

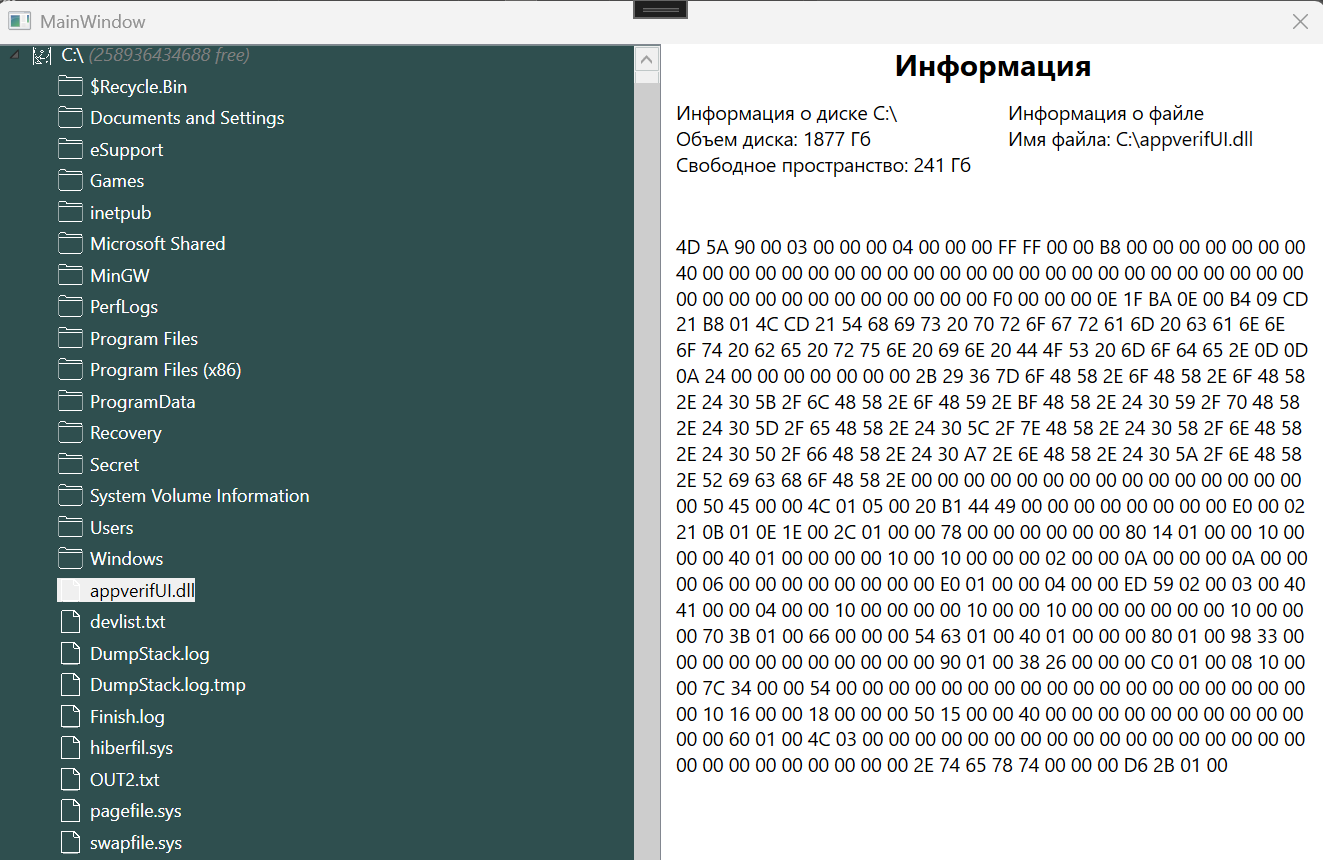


Рисунок 4 – предпросмотр бинарного файла

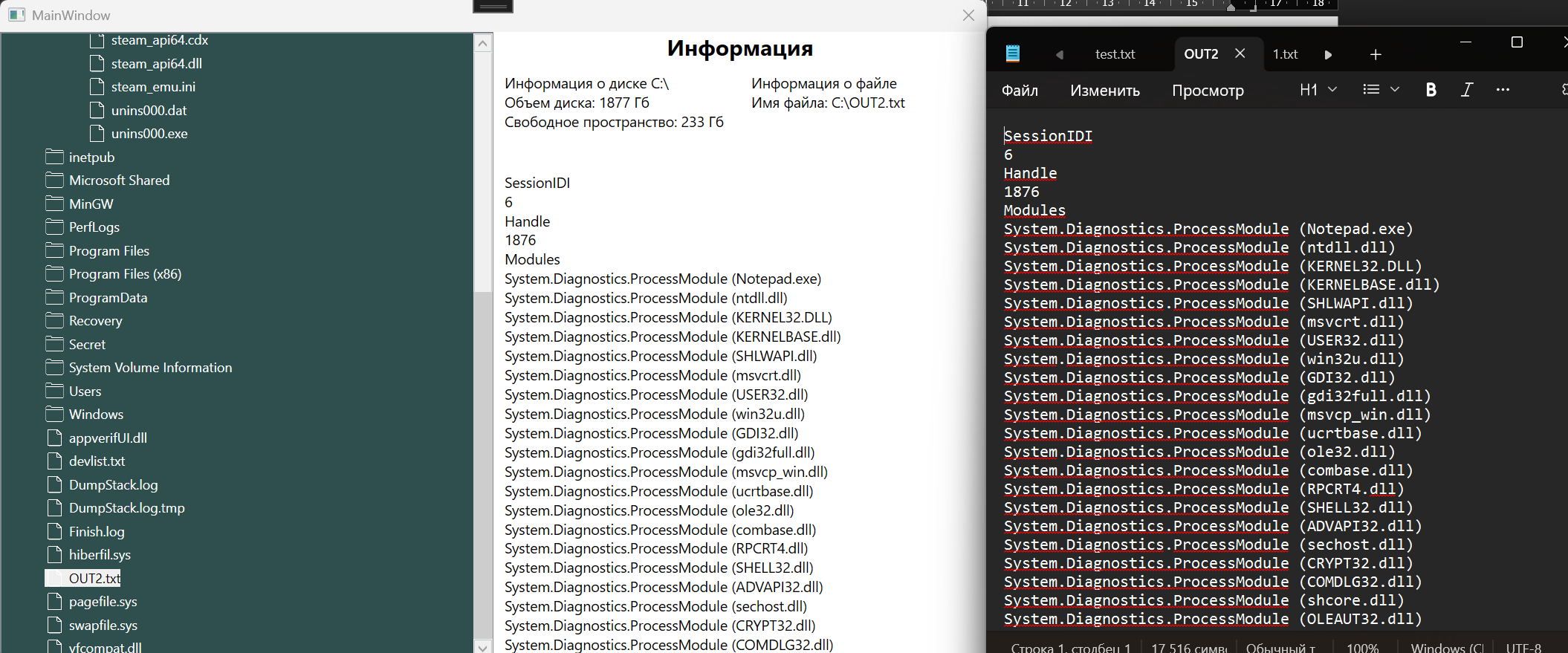


Рисунок 5 – Результат открытия и выделения файла

В программе реализован предпросмотр. При веделении текстового файла идёт показ содержимого в виде текста, при веделении bin файла идёт показ содержимого в шестнадцеричном виде. При двойном нажатии на файл через операционную среду этот файл запускается.

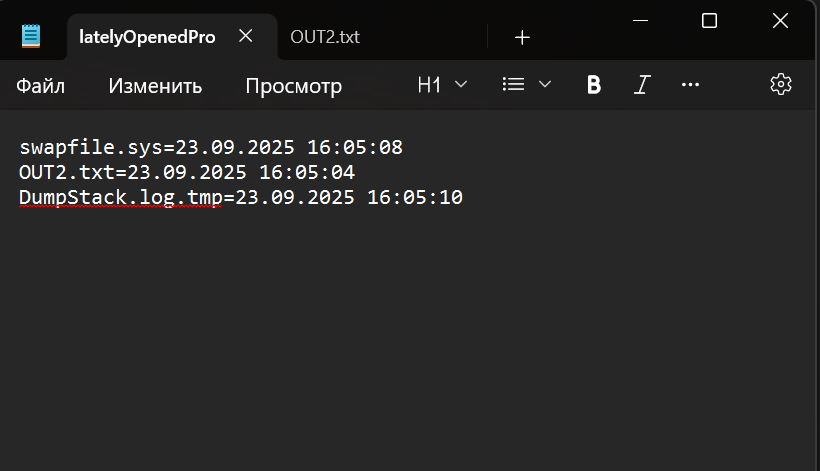


Рисунок 6 – Результат открытия программ за 10 секунд

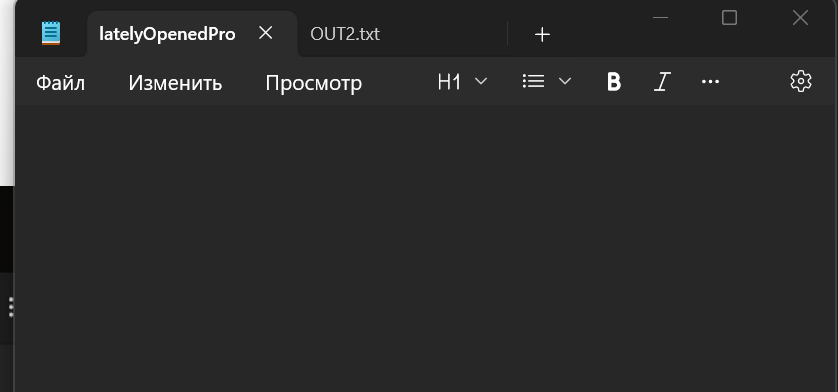


Рисунок 7 – Результат после десяти секунд

**Код программы**

Программа реализованная по паттерну MVVM, есть разделение на View – MainWindow, ViewModel – MainViewModel, Model – FileSystemObjectModel.

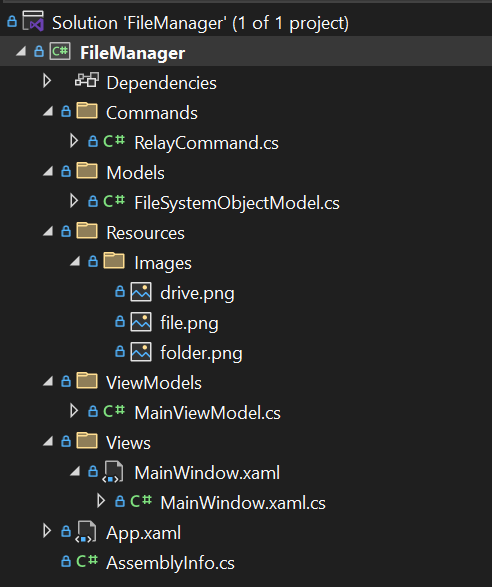


Рисунок 8 – Структура проекта

Код файла RelayCommand.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

using FileManager.Commands;

using FileManager.Models;

namespace FileManager.Commands

{

public class RelayCommand : ICommand

{

private readonly Action<object> \_execute;

private readonly Func<object, bool> \_canExecute;

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute ?? throw new ArgumentNullException(nameof(execute));

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null || \_canExecute(parameter);

}

public void Execute(object parameter)

{

\_execute(parameter);

}

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

}

}

Код файла FileSystemObjectModel.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Automation;

namespace FileManager.Models

{

public enum E\_ObjectType

{

File,

Directory,

Drive

}

public abstract class FileSystemObjectModel

{

public FileSystemObjectModel()

{

Children = new ObservableCollection<FileSystemObjectModel>();

}

public override string ToString() => Name;

public ObservableCollection<FileSystemObjectModel> Children { get; set; }

public abstract E\_ObjectType ObjectType { get; }

private string \_name;

public string Name

{

get => \_name;

set { \_name = value; }

}

private string \_fullpath;

public string FullPath

{

get => \_fullpath;

set { \_fullpath = value; }

}

private DateTime \_createDate;

public DateTime CreateDate

{

get => \_createDate;

set { \_createDate = value; }

}

}

public class File : FileSystemObjectModel

{

public override E\_ObjectType ObjectType => E\_ObjectType.File;

private long \_size; // Size in bytes

public long Size

{

get => \_size;

set { \_size = value; }

}

}

public class Directory : FileSystemObjectModel

{

public override E\_ObjectType ObjectType => E\_ObjectType.Directory;

private int \_itemAmount;

public int ItemAmount

{

get => \_itemAmount;

set { \_itemAmount = value; }

}

private long \_size; // Size in bytes

public long Size

{

get => \_size;

set { \_size = value; }

}

}

public class Drive : FileSystemObjectModel

{

public Drive(String Letter)

{

DriveInfo[] drives = DriveInfo.GetDrives();

foreach (var d in drives)

{

if (d.Name[0].ToString() == Letter)

{

this.SpaceLeft = d.AvailableFreeSpace;

this.SpaceOverall = d.TotalSize;

this.FullPath = d.Name;

break;

}

}

}

public Drive() { }

public override E\_ObjectType ObjectType => E\_ObjectType.Drive;

private long \_spaceLeft; // Size in bytes

public long SpaceLeft

{

get => \_spaceLeft;

set { \_spaceLeft = value; }

}

private long \_spaceOverall;

public long SpaceOverall

{

get => \_spaceOverall;

set { \_spaceOverall = value; }

}

}

}

Код файла MainViewModel.cs  
using FileManager.Commands;

using FileManager.Models;

using System;

using System.Collections;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Timers;

using System.Windows;

using System.Windows.Input;

using static System.Net.Mime.MediaTypeNames;

namespace FileManager.ViewModels

{

public class MainViewModel : INotifyPropertyChanged

{

private Hashtable openFiles\_openFileTime = new Hashtable();

private static System.Timers.Timer aTimer;

string filePath = "latelyOpenedPrograms.txt";

public MainViewModel()

{

RootItems = new ObservableCollection<Drive>();

LoadDrivesCommand = new RelayCommand(LoadDrives);

LoadChildrenCommand = new RelayCommand(LoadChildren);

OpenFileCommand = new RelayCommand(OpenFile);

SelectItemCommand = new RelayCommand(SelectItem);

MouseDoubleClickCommand = new RelayCommand(MouseDoubleClick);

LoadDrives(null);

SetTimer();

}

private void SetTimer()

{

aTimer = new System.Timers.Timer(1000);

aTimer.Elapsed += OnTimedEvent;

aTimer.AutoReset = true;

aTimer.Enabled = true;

}

private void OnTimedEvent(Object source, ElapsedEventArgs e)

{

Console.WriteLine("The Elapsed event was raised at {0:HH:mm:ss.fff}",

e.SignalTime);

var keysToRemove = new List<object>();

foreach (DictionaryEntry entry in openFiles\_openFileTime)

{

if (entry.Value is DateTime value)

{

if (DateTime.Now - value > TimeSpan.FromSeconds(10)) // should be 10 second

{

keysToRemove.Add(entry.Key);

}

}

}

// Remove all collected keys

foreach (var key in keysToRemove)

{

openFiles\_openFileTime.Remove(key);

}

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (DictionaryEntry entry in openFiles\_openFileTime)

{

writer.WriteLine($"{entry.Key}={entry.Value}");

}

}

}

private ObservableCollection<Drive> \_rootItems;

private FileSystemObjectModel \_selectedObject;

public ObservableCollection<Drive> RootItems

{

get => \_rootItems;

set { \_rootItems = value;

OnPropertyChanged(); }

}

private String \_selectedFilePreviewText;

public String SelectedFilePreviewText

{

get { return \_selectedFilePreviewText; }

set

{

\_selectedFilePreviewText = value;

OnPropertyChanged();

}

}

public FileSystemObjectModel SelectedObject

{

get => \_selectedObject;

set { \_selectedObject = value;

OnPropertyChanged(); }

}

private String \_driveInfo;

public String SelectedDriveInfo

{

get => \_driveInfo;

set { \_driveInfo = value; OnPropertyChanged(); }

}

private String \_directiryInfo;

public String SelectedDirectoryInfo

{

get => \_directiryInfo;

set { \_directiryInfo = value; OnPropertyChanged(); }

}

public ICommand LoadDrivesCommand { get; }

public ICommand LoadChildrenCommand { get; }

public ICommand OpenFileCommand { get; }

public ICommand SelectItemCommand { get; }

public ICommand MouseDoubleClickCommand { get; }

private void InfoUpdate(object parameter)

{

String RootDirectoryLetter = SelectedObject.FullPath[0].ToString();

Models.Drive RootDirectory = new Models.Drive(RootDirectoryLetter);

SelectedDriveInfo =

"Информация о диске " + RootDirectory.FullPath + "\n" +

"Объем диска: " + RootDirectory.SpaceOverall / 1024 / 1024 / 1024 + " Гб \n" +

"Свободное пространство: " + RootDirectory.SpaceLeft / 1024 / 1024 / 1024 + " Гб \n"

;

if (SelectedObject is Models.Directory directory)

{

SelectedDirectoryInfo =

"Информация о директории\n" +

"Директория: " + directory.FullPath + "\n" +

"Время создания: " + Convert.ToDateTime(directory.CreateDate) + "\n" +

"Корневой каталог: " + RootDirectory.FullPath + "\n"

;

}

else if (SelectedObject is Models.File file)

{

SelectedDirectoryInfo =

"Информация о файле\n" +

"Имя файла: " + file.FullPath + "\n"

;

}

else if (SelectedObject is Models.Drive drive)

{

SelectedDirectoryInfo =

"Информация о диске\n" +

"Объем диска: " + drive.SpaceOverall / 1024 / 1024 / 1024 + " Гб \n" +

"Свободное пространство: " + drive.SpaceLeft / 1024 / 1024 / 1024 + " Гб \n"

;

}

}

private void PreviewText(object parameter)

{

if (SelectedObject is Models.File file)

{

try

{

if (Path.GetExtension(file.FullPath).ToLower() == ".txt")

{

SelectedFilePreviewText = System.IO.File.ReadAllText(file.FullPath);

}

else

{

byte[] bytes = System.IO.File.ReadAllBytes(file.FullPath);

var bytesFirst500 = bytes.Take(500);

SelectedFilePreviewText = BitConverter.ToString(bytesFirst500.ToArray()).Replace("-"," ");

}

}

catch (Exception ex) {

SelectedFilePreviewText = ($"Access denied: {file.FullPath}");

}

}

else {

SelectedFilePreviewText = "Select File";

}

}

private void MouseDoubleClick(object parameter)

{

if (SelectedObject is Models.File file) {

try {

ProcessStartInfo startInfo = new ProcessStartInfo(file.FullPath);

startInfo.UseShellExecute = true;

System.Diagnostics.Process.Start(startInfo);

openFiles\_openFileTime.Add(file.Name, DateTime.Now);

}

catch {

}

}

}

private void SelectItem(object parameter)

{

InfoUpdate(null);

PreviewText(null);

}

private String GetRootDirectory(String path) => path[0].ToString();

private void LoadDrives(object parameter)

{

try

{

DriveInfo[] drives = DriveInfo.GetDrives();

foreach (DriveInfo drive in drives)

{

if (drive.IsReady)

{

RootItems.Add(

new Drive

{

Name = drive.Name,

FullPath = drive.Name,

SpaceOverall = drive.TotalSize,

SpaceLeft = drive.AvailableFreeSpace

});

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка загрузки дисков: {ex.Message}");

}

}

private void LoadChildren(object parameter)

{

if (parameter is FileSystemObjectModel item &&

System.IO.Directory.Exists(item.FullPath))

{

try

{

// Load directories

string[] dirs = System.IO.Directory.GetDirectories(item.FullPath);

foreach (string dirPath in dirs)

{

var dirInfo = new System.IO.DirectoryInfo(dirPath);

item.Children.Add(new Models.Directory

{

Name = dirInfo.Name,

FullPath = dirPath,

CreateDate = dirInfo.CreationTime,

});

}

// Load files

string[] files = System.IO.Directory.GetFiles(item.FullPath);

foreach (string filePath in files)

{

var fileInfo = new System.IO.FileInfo(filePath);

item.Children.Add(new Models.File

{

Name = fileInfo.Name,

FullPath = filePath,

CreateDate = fileInfo.CreationTime,

Size = fileInfo.Length

});

}

}

catch (UnauthorizedAccessException)

{

Console.WriteLine($"Access denied: {item.FullPath}");

}

catch (Exception ex)

{

Console.WriteLine($"Error loading children: {ex.Message}");

}

}

}

private void OpenFile(object parameter) =>

// Заглушка для реализации открытия файла

Console.WriteLine("OpenFile command executed");

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

Код файла MainWindow.xaml:

<Window x:Class="FileManager.Views.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:FileManager.Views"

xmlns:vm="clr-namespace:FileManager.ViewModels"

xmlns:models="clr-namespace:FileManager.Models"

mc:Ignorable="d"

Title="MainWindow" Height="600" Width="900"

ResizeMode="NoResize">

<Window.DataContext>

<vm:MainViewModel/>

</Window.DataContext>

<Window.Resources>

<HierarchicalDataTemplate DataType="{x:Type models:File}"

ItemsSource="{Binding Children}">

<StackPanel Orientation="Horizontal">

<Image Source="pack://application:,,,/Resources/Images/file.png" Width="16" Height="16" Margin="0,0,5,0"/>

<TextBlock Text="{Binding Name}"/>

</StackPanel>

</HierarchicalDataTemplate>

<HierarchicalDataTemplate DataType="{x:Type models:Directory}"

ItemsSource="{Binding Children}">

<StackPanel Orientation="Horizontal">

<Image Source="pack://application:,,,/Resources/Images/folder.png" Width="16" Height="16" Margin="0,0,5,0"/>

<TextBlock Text="{Binding Name}"/>

</StackPanel>

</HierarchicalDataTemplate>

<HierarchicalDataTemplate DataType="{x:Type models:Drive}"

ItemsSource="{Binding Children}">

<StackPanel Orientation="Horizontal">

<Image Source="pack://application:,,,/Resources/Images/drive.png" Width="16" Height="16" Margin="0,0,5,0"/>

<TextBlock Text="{Binding Name}"/>

<TextBlock Text="{Binding SpaceLeft, StringFormat=' ({0} free)'}"

Foreground="Gray" FontStyle="Italic"/>

</StackPanel>

</HierarchicalDataTemplate>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TreeView Grid.Column="0"

Background="DarkSlateGray"

ItemsSource="{Binding RootItems}"

SelectedItemChanged="TreeView\_SelectedItemChanged">

<TreeView.ItemContainerStyle>

<Style TargetType="{x:Type TreeViewItem}">

<Setter Property="Foreground" Value="White"></Setter>

<Setter Property="Margin" Value="0,5,0,0"></Setter>

<Setter Property="IsExpanded" Value="{Binding IsExpanded, Mode=TwoWay}"/>

<Setter Property="IsSelected" Value="{Binding IsSelected, Mode=TwoWay}"/>

<EventSetter Event="Expanded" Handler="TreeViewItem\_Expanded"/>

<EventSetter Event="MouseDoubleClick" Handler="TreeViewItem\_MouseDoubleClick"/>

</Style>

</TreeView.ItemContainerStyle>

</TreeView>

<Grid Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"></RowDefinition>

<RowDefinition Height="auto"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Grid.ColumnSpan="2" Height="auto" Text="Информация" FontWeight="Bold" FontSize="20" HorizontalAlignment="Center"></TextBlock>

<TextBlock Grid.Row="1" Grid.Column="0"

Text="{Binding SelectedDriveInfo}"

TextWrapping="Wrap"

FontSize="13"

Padding="10"

/>

<TextBlock Grid.Row="1" Grid.Column="1"

Text="{Binding SelectedDirectoryInfo}"

TextWrapping="Wrap"

FontSize="13"

Padding="10"

/>

<TextBlock Grid.Row="2" Grid.ColumnSpan="2"

Text="{Binding SelectedFilePreviewText}"

TextWrapping="Wrap"

FontSize="13"

Padding="10"

/>

</Grid>

</Grid>

</Window>

Код файла MainWindow.xaml.cs:

using System.Windows;

using System.Windows.Controls;

using FileManager.Models;

using FileManager.ViewModels;

namespace FileManager.Views

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void TreeView\_SelectedItemChanged(object sender, RoutedPropertyChangedEventArgs<object> e)

{

if (DataContext is MainViewModel viewModel && e.NewValue is FileSystemObjectModel selectedItem)

{

viewModel.SelectedObject = selectedItem;

viewModel.SelectItemCommand.Execute(null);

}

}

private void TreeViewItem\_Expanded(object sender, RoutedEventArgs e)

{

var treeViewItem = e.OriginalSource as TreeViewItem;

if (treeViewItem?.DataContext is FileSystemObjectModel item)

{

var viewModel = DataContext as MainViewModel;

if (item.Children.Count == 0)

{

viewModel?.LoadChildrenCommand.Execute(item);

}

}

}

private void TreeViewItem\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if(DataContext is MainViewModel viewModel)

{

viewModel.MouseDoubleClickCommand.Execute(null);

}

}

}

}