Лабораторная работа 6 по С#

Цель работы. Освоить основные концепции и практические навыки работы с файлами и каталогами, включая создание, чтение, запись, удаление и перемещение файлов. Понять, как применять рефлексию для динамического получения информации о типах, методах и свойствах объектов. Научиться сериализовать и десериализовать объекты в различные форматы. Изучить методы архивирования файлов и каталогов

Задание 1

Console.WriteLine("Лабораторная 6 по С#");

DirectoryInfo directoryInfo = new DirectoryInfo(@"D:\Лабораторная работа 6 по С#");

directoryInfo.Create();

directoryInfo.CreateSubdirectory(@"dir1\dir2");

directoryInfo.CreateSubdirectory(@"dir3\dir4");

directoryInfo.CreateSubdirectory(@"dir5\dir6");

FileInfo fileInfo = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\dog.jpg");

fileInfo.Create();

FileInfo fileInfo1 = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\cat.jpg");

fileInfo1.Create();

FileInfo fileInfo2 = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\frog.jpg");

fileInfo2.Create();

string str = @"D:\Лабораторная работа 6 по С#";

if (Directory.Exists(str))

{

Console.WriteLine("Подкаталоги:");

string[] strArray = Directory.GetDirectories(str);

foreach (string str2 in strArray)

{

Console.WriteLine(str2);

}

}

string str1 = @"D:\Лабораторная работа 6 по С#\dir1\dir2";

if (Directory.Exists(str1))

{

Console.WriteLine("Файлы:");

string[] strArray = Directory.GetFiles(str1);

foreach (string str2 in strArray)

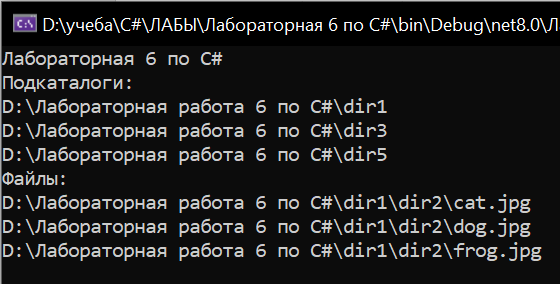
{

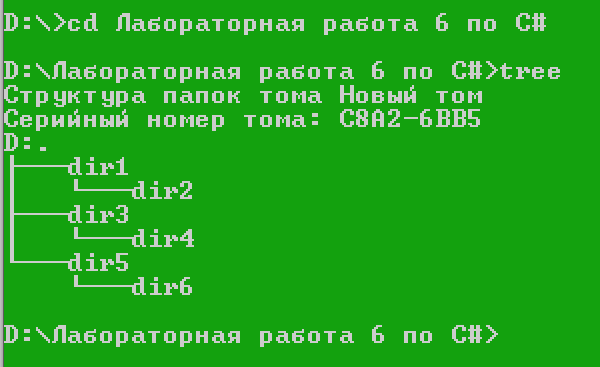
Console.WriteLine(str2);

}

}

Console.ReadKey();





Console.WriteLine("Лабораторная 6 по С#");

DirectoryInfo directoryInfo = new DirectoryInfo(@"D:\Лабораторная работа 6 по С#");

directoryInfo.Create();

directoryInfo.CreateSubdirectory(@"dir1\dir2");

directoryInfo.CreateSubdirectory(@"dir3\dir4");

directoryInfo.CreateSubdirectory(@"dir5\dir6");

//FileInfo fileInfo = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\dog.jpg");

//fileInfo.Create();

//FileInfo fileInfo1 = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\cat.jpg");

//fileInfo1.Create();

//FileInfo fileInfo2 = new FileInfo(@"D:\Лабораторная работа 6 по С#\dir1\dir2\frog.jpg");

//fileInfo2.Create();

string str = @"D:\Лабораторная работа 6 по С#";

if (Directory.Exists(str))

{

Console.WriteLine("Подкаталоги:");

string[] strArray = Directory.GetDirectories(str);

foreach (string str2 in strArray)

{

Console.WriteLine(str2);

}

}

string str1 = @"D:\Лабораторная работа 6 по С#\dir1\dir2";

if (Directory.Exists(str1))

{

Console.WriteLine("Файлы:");

string[] strArray = Directory.GetFiles(str1);

foreach (string str2 in strArray)

{

Console.WriteLine(str2);

FileInfo fileInfo3 = new FileInfo(str2);

string yearMonth = fileInfo3.CreationTime.ToString("yyyy-MM");

string yearMonthDirectory=Path.Combine(@"D:\Лабораторная работа 6 по С#", yearMonth);

Directory.CreateDirectory(yearMonthDirectory);

if (Directory.Exists(yearMonthDirectory))

{

string filePath=Path.Combine(yearMonthDirectory, fileInfo3.Name);

fileInfo3.MoveTo(filePath);

Console.WriteLine($"Файл {fileInfo3.Name} перемещен");

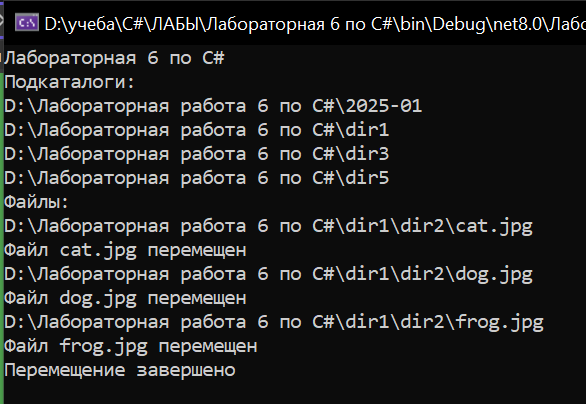
}

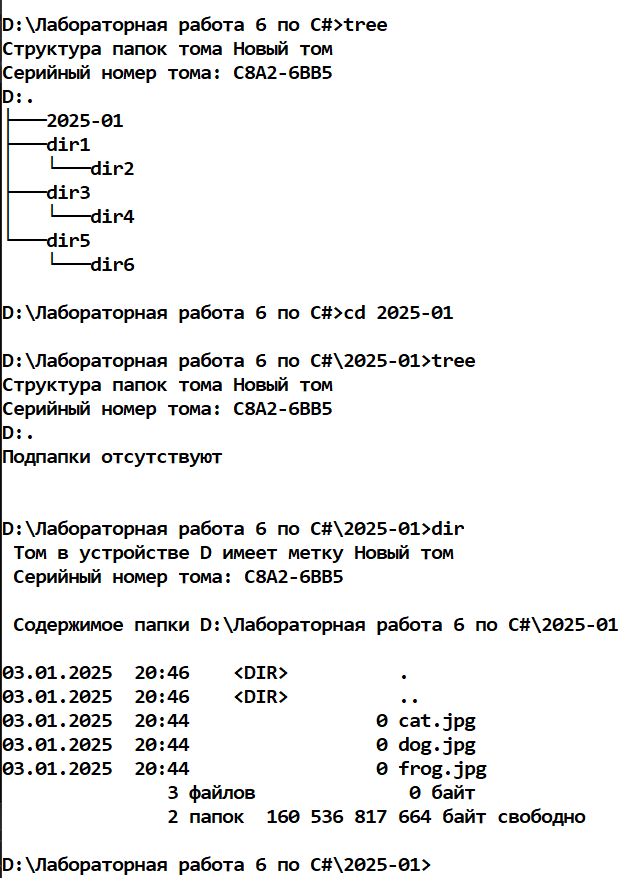
}

Console.WriteLine("Перемещение завершено");

}

Console.ReadKey();





Вариант 2

string str = @"D:\Лабораторная работа 6 по С#\2025-01";

Console.WriteLine(str);

if (Directory.Exists(str))

{

string[] strings=Directory.GetFiles(str);

foreach (string s in strings)

{

FileInfo fileInfo = new FileInfo(s);

string extension = Path.GetExtension(s);

extension = extension.Substring(1);

string directoryPath= Path.Combine(str, extension);

if (!Directory.Exists(directoryPath))

{

Directory.CreateDirectory(directoryPath);

}

if (Directory.Exists(directoryPath))

{

string fileMove = Path.Combine(directoryPath, fileInfo.Name);

fileInfo.MoveTo(fileMove);

Console.WriteLine($"Файл {fileInfo} перемещен");

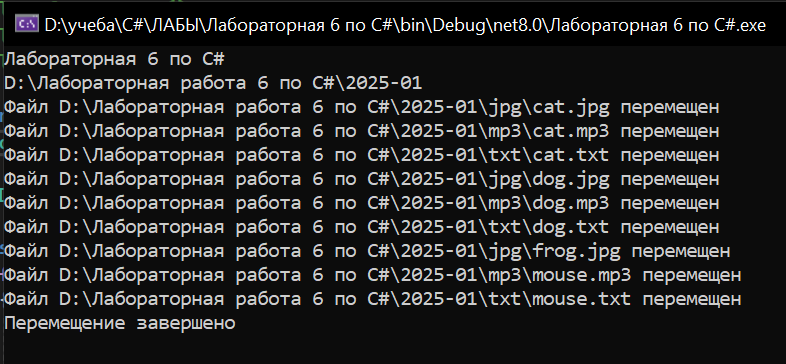
}

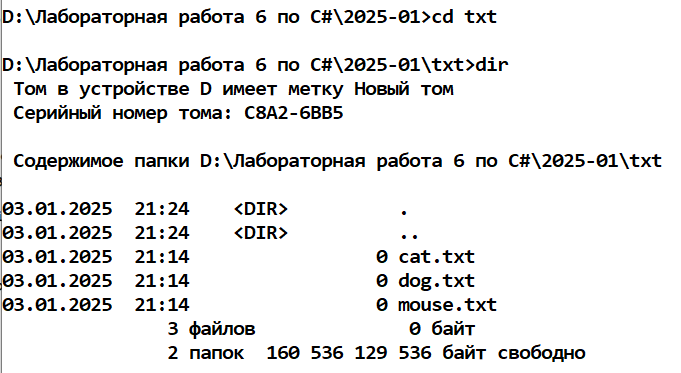
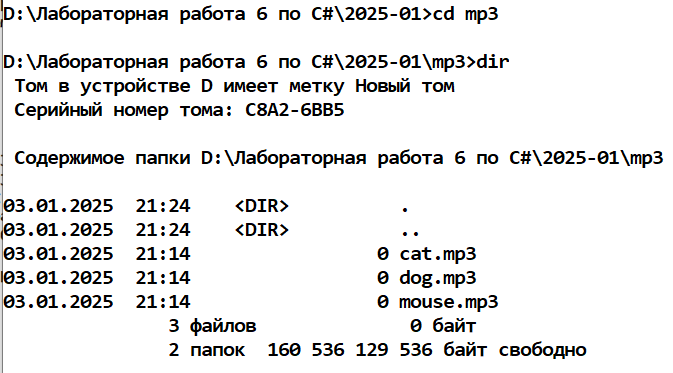
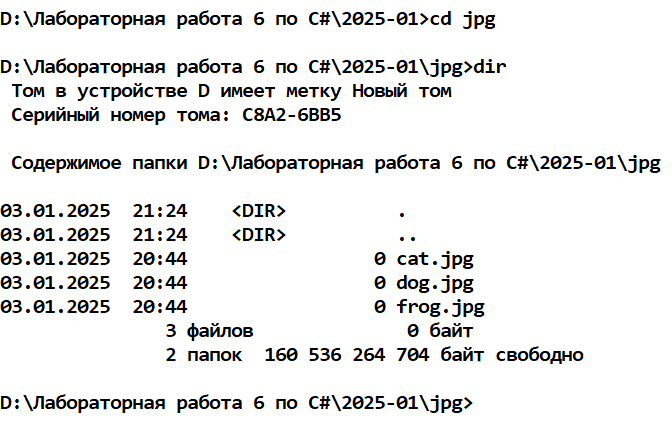
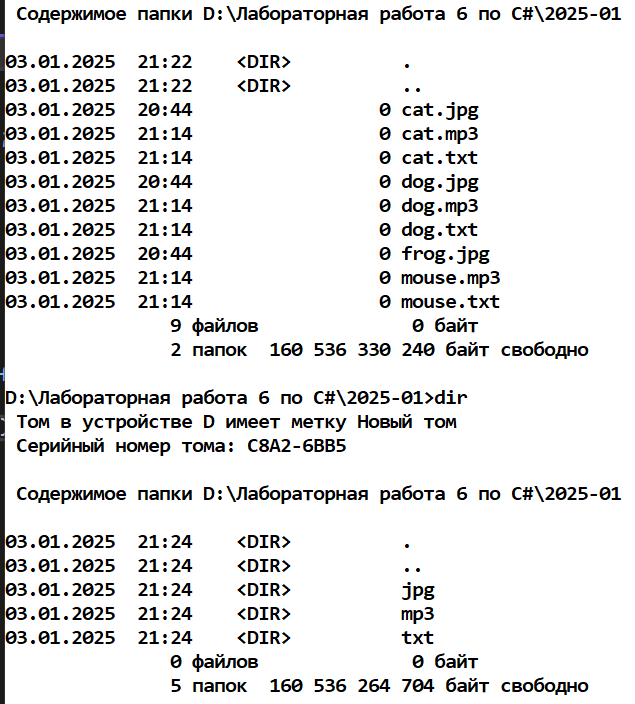
}

Console.WriteLine("Перемещение завершено");

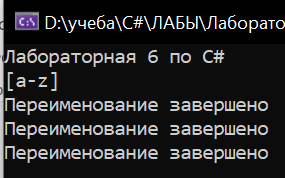
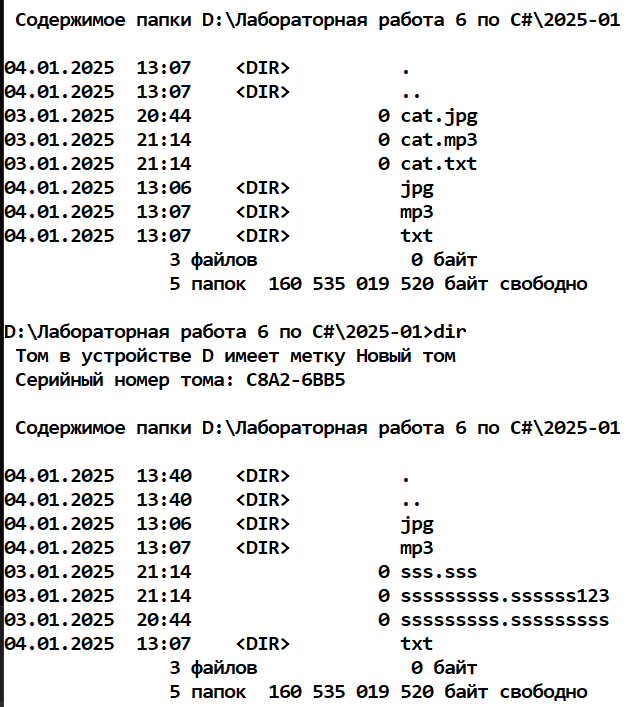
}

Console.ReadKey();





Вариант 3



string str = @"D:\Лабораторная работа 6 по С#\2025-01";

string reg;

reg=Console.ReadLine();

Regex regex=new Regex(reg);

if (Directory.Exists(str))

{

string[] strings=Directory.GetFiles(str);

foreach (string s in strings)

{

try

{

FileInfo fileInfo=new FileInfo(s);

if(Regex.IsMatch(fileInfo.Name, reg))

{

string NewName = Regex.Replace(fileInfo.Name, reg, "s");

string NewPath=Path.Combine(str, NewName);

if (File.Exists(NewPath))

{

throw new Exception($"Файл {NewName} уже существует");

}

File.Move(fileInfo.FullName, NewPath);

Console.WriteLine("Переименование завершено");

}

}

catch(Exception ex)

{

Console.WriteLine($"Exception: {ex.Message}");

}

}

}

Console.ReadKey();

Задание 2

using Лабораторная\_6\_задание\_2;

Console.WriteLine("Лабораторная 6 задание 2");

Dog myDog = new Dog

{

Name = "Rex",

Age = 5,

Breed = "German Shepherd"

};

// Сохранение информации о типе класса

Dog.WriteTypeInfo("type\_info.txt");

// Сохранение объекта в бинарный файл

myDog.SaveToBinary("dog\_info.bin");

// Загрузка объекта из бинарного файла

Dog loadedDog = Dog.LoadFromBinary("dog\_info.bin");

Console.WriteLine($"Loaded Dog: {loadedDog.Name}, Age: {loadedDog.Age}, Breed: {loadedDog.Breed}");

// Сериализация в JSON

myDog.SerializeToJson("dog\_info.json");

// Десериализация из JSON

Dog deserializedDog = Dog.DeserializeFromJson("dog\_info.json");

Console.WriteLine($"Deserialized Dog: {deserializedDog.Name}, Age: {deserializedDog.Age}, Breed: {deserializedDog.Breed}");

Console.ReadKey();

namespace Лабораторная\_6\_задание\_2

{

[Serializable]

internal class Animal

{

public string Name { get; set; }

public int Age { get; set; }

public virtual void Speak()

{

Console.WriteLine("Подает голос");

}

}

}

namespace Лабораторная\_6\_задание\_2

{

internal class Dog:Animal

{

public string Breed { get; set; }

public override void Speak()

{

Console.WriteLine("Гав! Гав!");

}

// Статический метод для записи информации о типе класса (рефлексия)

public static void WriteTypeInfo(string fileName)

{

Type type = typeof(Dog);

using (StreamWriter writer = new StreamWriter(fileName))

{

writer.WriteLine($"Class Name: {type.Name}");

writer.WriteLine($"Base Type: {type.BaseType.Name}");

writer.WriteLine($"Namespace: {type.Namespace}");

writer.WriteLine($"Properties:");

foreach (PropertyInfo prop in type.GetProperties())

{

writer.WriteLine($"- {prop.Name} : {prop.PropertyType}");

}

}

}

// Экземплярный метод для сохранения в бинарный файл

public void SaveToBinary(string fileName)

{

using (FileStream fs = new FileStream(fileName, FileMode.Create))

{

using (BinaryWriter writer = new BinaryWriter(fs))

{

writer.Write(Name);

writer.Write(Age);

writer.Write(Breed);

}

}

}

// Метод для чтения информации из бинарного файла

public static Dog LoadFromBinary(string fileName)

{

using (FileStream fs = new FileStream(fileName, FileMode.Open))

{

using (BinaryReader reader = new BinaryReader(fs))

{

Dog dog = new Dog

{

Name = reader.ReadString(),

Age = reader.ReadInt32(),

Breed = reader.ReadString()

};

return dog;

}

}

}

// Метод для сериализации объекта в JSON

public string SerializeToJson(string fileName)

{

string jsonString = JsonSerializer.Serialize(this);

File.WriteAllText(fileName, jsonString);

return jsonString;

}

// Метод для десериализации объекта из JSON

public static Dog DeserializeFromJson(string fileName)

{

string jsonString = File.ReadAllText(fileName);

return JsonSerializer.Deserialize<Dog>(jsonString);

}

}

}

