**Работа с файлами**

Задание 1 Основные операции с файлами.

Листинг программы:

string filePath = "example.txt";

string copyPath = "example\_copy.txt";

string newDirectory = "NewDirectory";

string renamedFilePath = Path.Combine(newDirectory, "familiya.io");

// 1. Создать файл, записать в него текст, прочитать и вывести в консоль.

File.WriteAllText(filePath, "Hello, world!");

Console.WriteLine(File.ReadAllText(filePath));

// 2. Проверить существование файла перед его удалением.

if (File.Exists(filePath))

{

File.Delete(filePath);

Console.WriteLine("Файл удален.");

}

// 3. Получить информацию о файле (размер, даты).

File.WriteAllText(filePath, "Sample text for file.");

FileInfo fileInfo = new FileInfo(filePath);

Console.WriteLine($"Размер: {fileInfo.Length} байт, Дата создания: {fileInfo.CreationTime}");

// 4. Скопировать файл и убедиться, что копия существует.

File.Copy(filePath, copyPath);

Console.WriteLine($"Копия файла существует: {File.Exists(copyPath)}");

// 5. Переместить файл в новую директорию.

Directory.CreateDirectory(newDirectory);

File.Move(copyPath, renamedFilePath);

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

// 6. Переименовать файл в файл familiya.io (уже сделано в предыдущем шаге).

// 7. Обработать ошибку при удалении несуществующего файла.

try

{

File.Delete("non\_existent\_file.txt");

}

catch (FileNotFoundException e)

{

Console.WriteLine($"Ошибка: {e.Message}");

}

// 8. Сравнить два файла по размеру.

long originalSize = new FileInfo(filePath).Length;

long copiedSize = new FileInfo(renamedFilePath).Length;

Console.WriteLine($"Размер оригинального файла и копии одинаковый: {originalSize == copiedSize}");

// 9. Удалить все файлы в папке, соответствующие определенному шаблону.

foreach (var file in Directory.GetFiles(newDirectory, "\*.ii"))

{

File.Delete(file);

Console.WriteLine($"Удален файл: {file}");

}

// 10. Вывести список всех файлов в заданной директории.

var files = Directory.GetFiles(newDirectory);

Console.WriteLine("Файлы в директории:");

foreach (var f in files)

{

Console.WriteLine(f);

}

// 11. Запретить запись в файл и попытаться записать в него.

var attributes = File.GetAttributes(renamedFilePath);

File.SetAttributes(renamedFilePath, attributes | FileAttributes.ReadOnly);

try

{

File.AppendAllText(renamedFilePath, "Trying to write to a read-only file.");

}

catch (UnauthorizedAccessException e)

{

Console.WriteLine($"Ошибка: {e.Message}");

}

// 12. Проверить доступные права к файлу.

Console.WriteLine($"Доступные права к файлу {renamedFilePath}:");

Console.WriteLine($"Чтение: {File.Exists(renamedFilePath) && (File.GetAttributes(renamedFilePath) & FileAttributes.ReadOnly) == 0}");

Console.WriteLine($"Запись: {(File.GetAttributes(renamedFilePath) & FileAttributes.ReadOnly) == 0}");

Console.WriteLine($"Выполнение: {File.Exists(renamedFilePath)}");

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

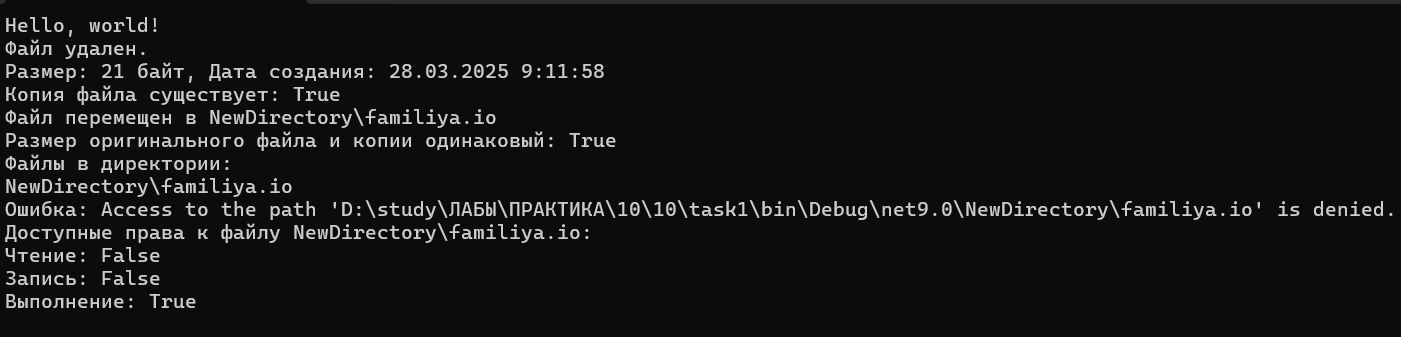


Рисунок 1.1 – Результат работы программы