**Звіт до Лабораторної роботи №3**

**Кучугурної Олександри Юріївни**

**ПЗ-22-3**

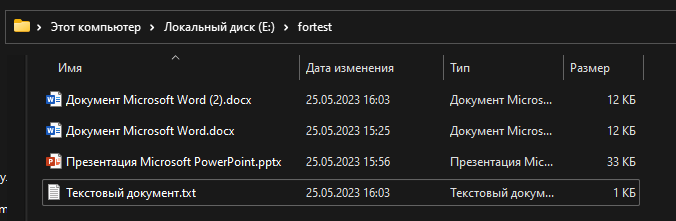
Тема: Робота з файловою системою у мовах програмування високого рівня

Варіант 12. Синхронізувати цифрові в двох каталогах за датою (залишити свіжіші).

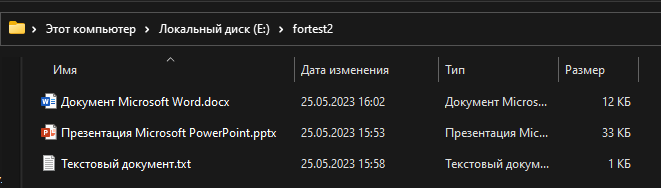
В якості мови програмування високого рівня я обрала С#.

Заздалегідь створюємо 4 каталоги для синхронізації: E:\fortest, E:\fortest2, E:\fortest3, E:\fortest4.

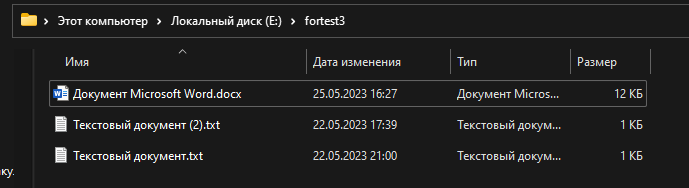
E:\fortest:



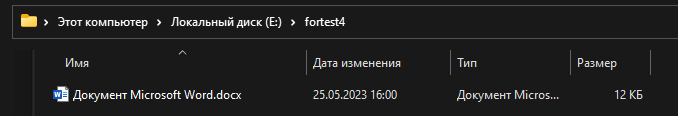
E:\fortest2:



E:\fortest3:



E:\fortest4:

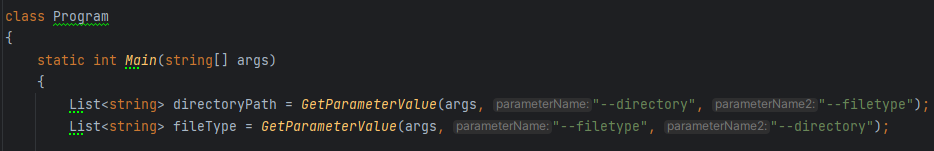


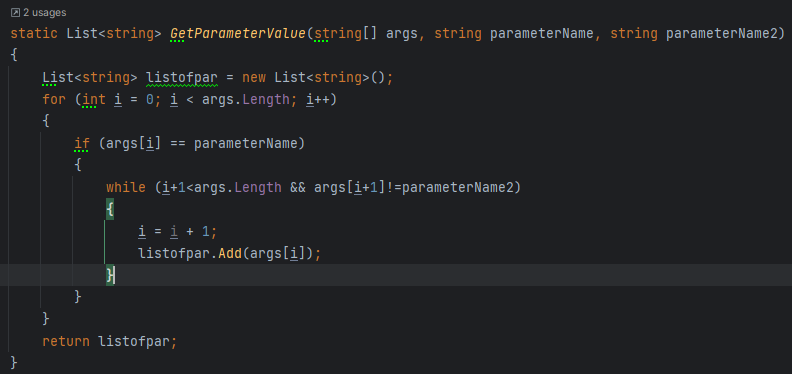
Працюємо з вказаним каталогом (передається як параметр утиліти) та з вказаними файлами за шаблоном (наприклад, \*.exe). В командному рядку це виглядає так:



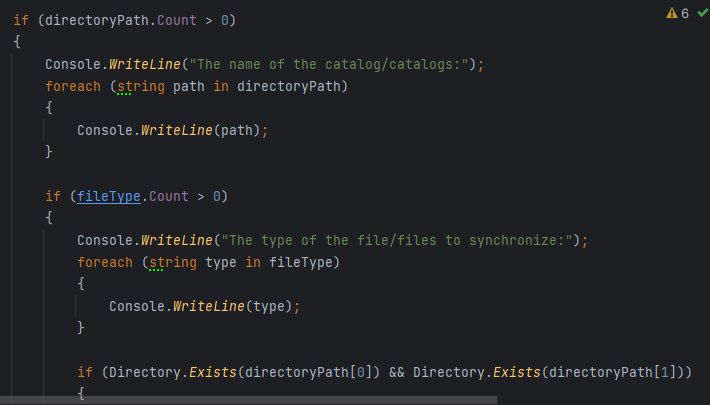
Тобто, перед викликом необхідного каталогу спочатку вказуємо утиліту –directory, потім можна вводити будь-яку кількість каталогів. Так само з шаблонами файлів: спочатку пишемо –filetype і потім вводимо необхідні типи файлів для синхронізації.

Передбачається зміна кількість параметрів командного рядка:

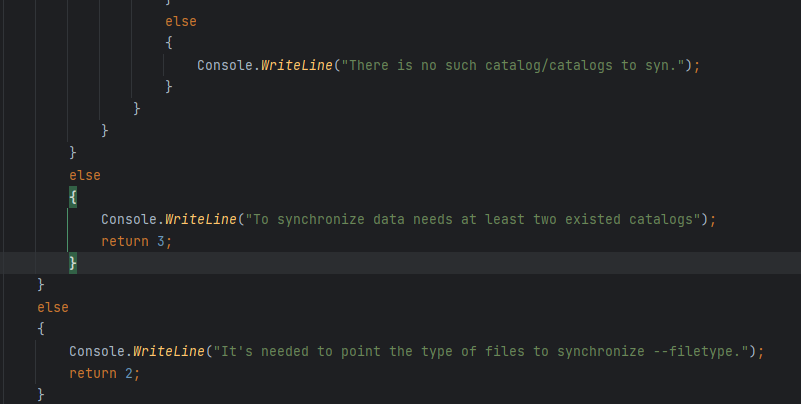


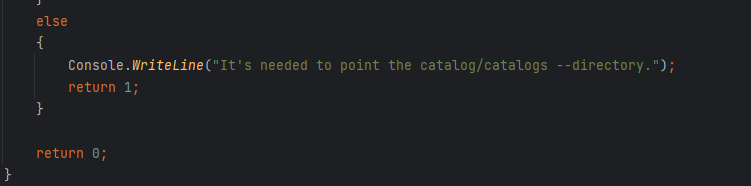


Потім з підказками перевіряємо коректність введених даних та виводимо задані каталоги та типи:



Повідомляємо, якщо щось пішло не так та формуємо коди завершення:



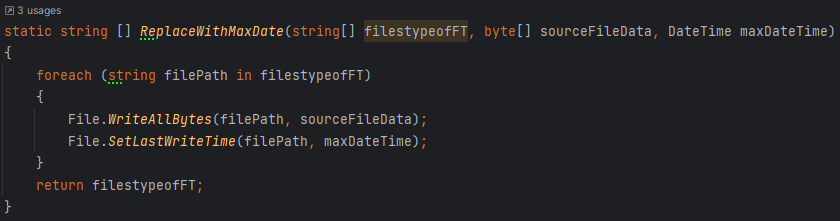


Тепер, для синхронізації цифрових в двох каталогах за датою, спочатку створюємо метод, що буде синхронізувати всі файли одного типу за датою в одному каталозі, враховуючи їх атрибути (прихований, тільки для читання, архівний):

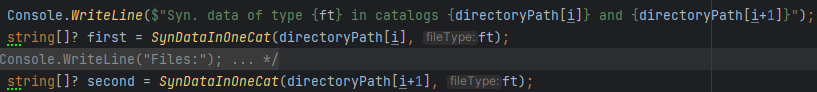


В цьому методі знаходимо найсвіжіший файл заданого типу і замінюємо інші файли каталогу на найсвіжіший. Для цього використовуємо ще два нових методи:

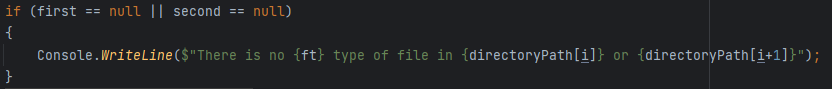




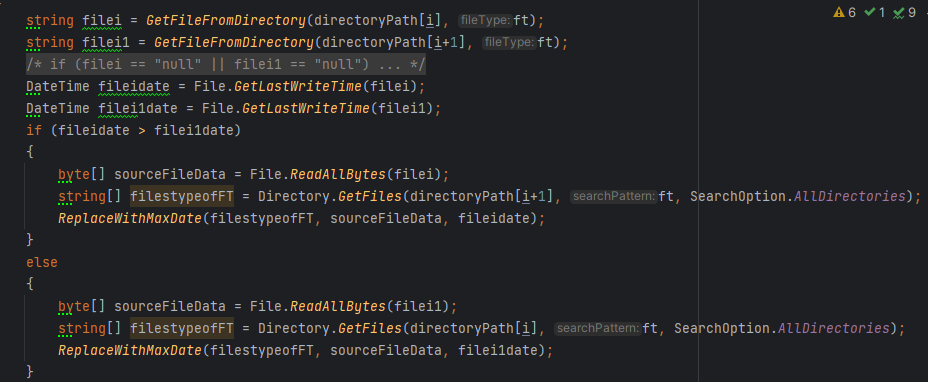
Виконуємо метод в методі Main (при цьому перед виконанням синхронізації у вигляді підказки повідомляємо користувачу, які каталоги будуть синхронізовані):



І перевіряємо, чи взагалі були файли необхідного типу в каталогах:

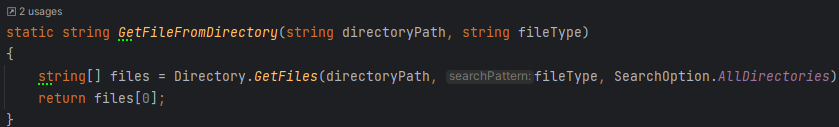


Якщо хоча б один з двох каталогів не мав заданого типу файлу, то виводиться повідомлення. В іншому випадку виконується вже синхронізація між двома каталогами:



Розглянемо по черзі, як синхронізуються дані в двох каталогах.

Спочатку отримуємо по одному файлу з кожного каталогу (використовуючи метод GetFileFromDirectory):



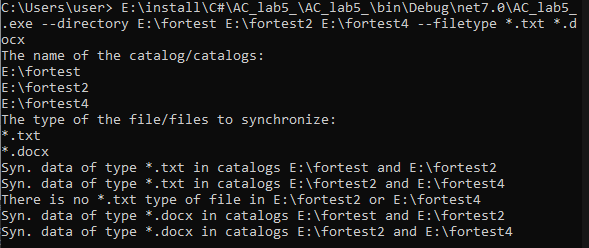
Після синхронізації методом SynDataInOneFile всі файли заданого типу у файлі будуть мати однакове значення (а саме значення найбільш свіжішого), і після перевірки на null у списку синхронізованих файлів буде як мінімум один файл, отже метод GetFileFromDirectory повертає саме перший файл каталогу.

Потім отримуємо час оновлення кожного з двох файлів та перевіряємо їх, якщо перший – найсвіжіший, тоді дані цього файлу синхронізуємо з даними файлів другого каталогу, і навпаки, якщо другий файл виявився свіжіший.

Ці дії виконуємо для кожного заданого типу файлу.

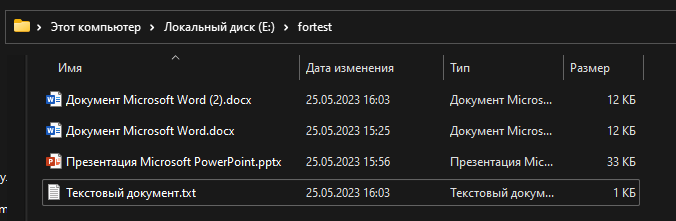


Тепер перевіримо роботу програми у командній строчці:

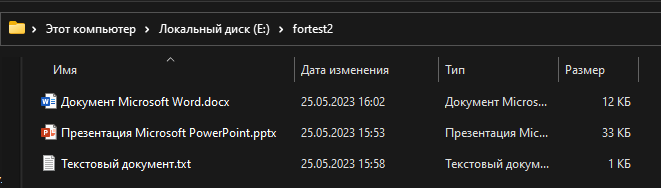


Було:

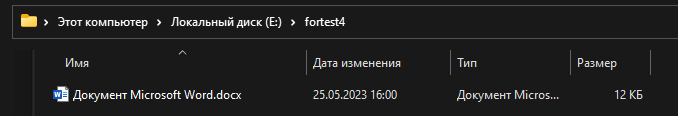
E:\fortest:



E:\fortest2:

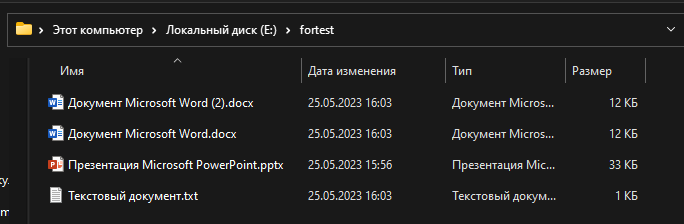


E:\fortest4:

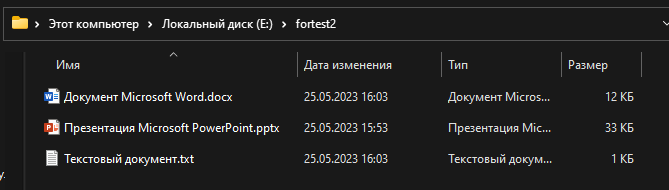


Стало:

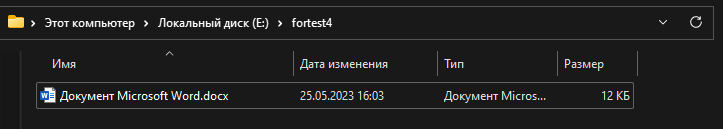
E:\fortest:



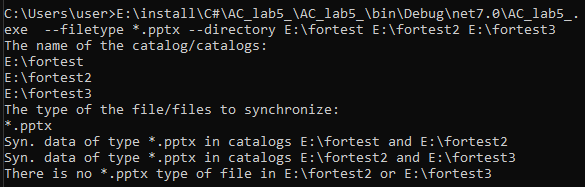
E:\fortest2:



E:\fortest4:

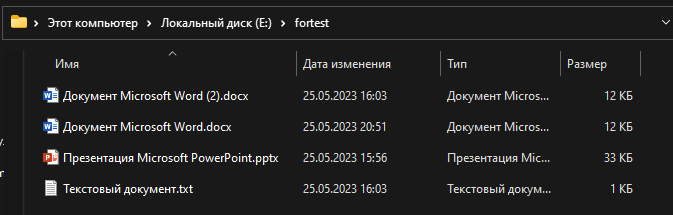


Ще один приклад:

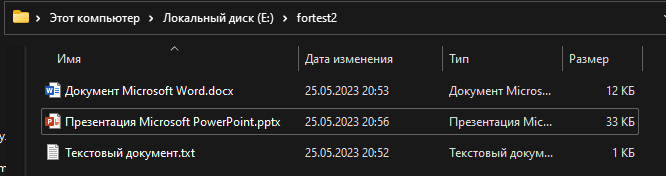


Було:

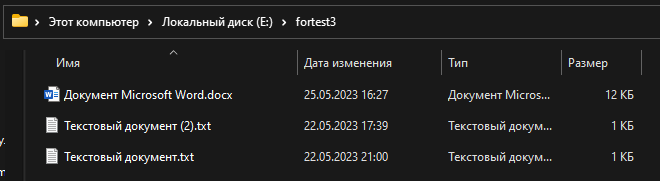
E:\fortest:



E:\fortest2:

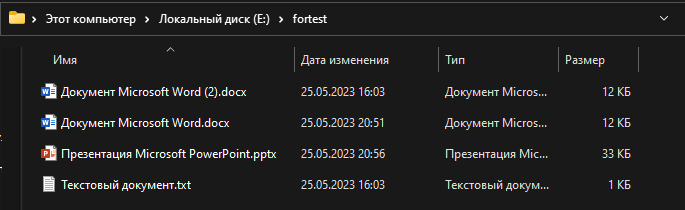


E:\fortest3:

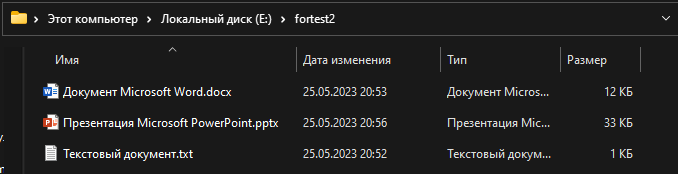


Стало:

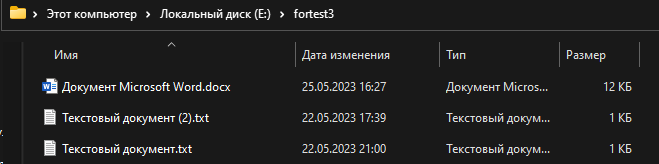
E:\fortest:



E:\fortest2:



E:\fortest3:



Весь код програми:

namespace AC\_lab5\_;  
  
using System;  
using System.Collections.Generic;  
using System.IO;  
  
class Program  
{  
 static int *Main*(string[] args)  
 {  
 List<string> directoryPath = *GetParameterValue*(args, "--directory", "--filetype");  
 List<string> fileType = *GetParameterValue*(args, "--filetype", "--directory");  
  
 if (directoryPath.Count > 0)  
 {  
 Console.*WriteLine*("The name of the catalog/catalogs:");  
 foreach (string path in directoryPath)  
 {  
 Console.*WriteLine*(path);  
 }  
  
 if (fileType.Count > 0)  
 {  
 Console.*WriteLine*("The type of the file/files to synchronize:");  
 foreach (string type in fileType)  
 {  
 Console.*WriteLine*(type);  
 }  
  
 if (Directory.*Exists*(directoryPath[0]) && Directory.*Exists*(directoryPath[1]))  
 {  
 foreach (var ft in fileType)  
 {  
 for (int i = 0; i + 1 < directoryPath.Count; i++)  
 {  
 if (Directory.*Exists*(directoryPath[i]) && Directory.*Exists*(directoryPath[i + 1]))  
 {  
 Console.*WriteLine*($"Syn. data of type {ft} in catalogs {directoryPath[i]} and {directoryPath[i+1]}");   
 string[]? first = *SynDataInOneCat*(directoryPath[i], ft);  
 // Console.WriteLine("Files:");  
 // string[] files = Directory.GetFiles(directoryPath[i]);  
 // foreach (string s in files)  
 // {  
 // Console.WriteLine(s);  
 // }  
 string[]? second = *SynDataInOneCat*(directoryPath[i+1], ft);  
 if (first == null || second == null)  
 {  
 Console.*WriteLine*($"There is no {ft} type of file in {directoryPath[i]} or {directoryPath[i+1]}");  
 }   
 else  
 {  
 string filei = *GetFileFromDirectory*(directoryPath[i], ft);  
 string filei1 = *GetFileFromDirectory*(directoryPath[i+1], ft);  
   
   
 DateTime fileidate = File.*GetLastWriteTime*(filei);  
 DateTime filei1date = File.*GetLastWriteTime*(filei1);  
 if (fileidate > filei1date)  
 {  
 byte[] sourceFileData = File.*ReadAllBytes*(filei);  
 string[] filestypeofFT = Directory.*GetFiles*(directoryPath[i+1], ft, SearchOption.*AllDirectories*);  
 *ReplaceWithMaxDate*(filestypeofFT, sourceFileData, fileidate);  
 }  
 else  
 {  
 byte[] sourceFileData = File.*ReadAllBytes*(filei1);  
 string[] filestypeofFT = Directory.*GetFiles*(directoryPath[i], ft, SearchOption.*AllDirectories*);  
 *ReplaceWithMaxDate*(filestypeofFT, sourceFileData, filei1date);  
 }  
 }  
 }  
 else  
 {  
 Console.*WriteLine*("There is no such catalog/catalogs to syn.");  
 }  
 }  
 }  
 }  
 else  
 {  
 Console.*WriteLine*("To synchronize data needs at least two existed catalogs");  
 return 3;  
 }  
 }  
 else  
 {  
 Console.*WriteLine*("It's needed to point the type of files to synchronize --filetype.");  
 return 2;  
 }  
 }  
 else  
 {  
 Console.*WriteLine*("It's needed to point the catalog/catalogs --directory.");  
 return 1;  
 }  
  
 return 0;  
 }  
 static List<string> *GetParameterValue*(string[] args, string parameterName, string parameterName2)  
 {  
 List<string> listofpar = new List<string>();  
 for (int i = 0; i < args.Length; i++)  
 {  
 if (args[i] == parameterName)  
 {  
 while (i+1<args.Length && args[i+1]!=parameterName2)  
 {  
 i = i + 1;  
 listofpar.Add(args[i]);  
 }  
 }  
 }  
 return listofpar;  
 }  
  
 static string []? *SynDataInOneCat*(string directoryPath, string fileType)  
 {  
 string[] filestypeofFT = Directory.*GetFiles*(directoryPath, fileType, SearchOption.*AllDirectories*);  
 if (filestypeofFT.Length > 0)  
 {  
 DateTime maxDateTime = DateTime.*MinValue*;  
 string maxDate = *FindMaxDate*(filestypeofFT, ref maxDateTime);  
 byte[] sourceFileData = File.*ReadAllBytes*(maxDate);  
 return *ReplaceWithMaxDate*(filestypeofFT, sourceFileData, maxDateTime);  
 }  
 else  
 {  
 return null;  
 }  
 }  
 static string *FindMaxDate*(string [] filestypeofFT, ref DateTime maxDateTime)  
 {  
 DateTime[] dateTimes = new DateTime[7];  
  
 int IndOfMax=0;  
  
 for (int i = 0; i < filestypeofFT.Length; i++)  
 {  
 dateTimes[i] = File.*GetLastWriteTime*(filestypeofFT[i]);  
 if (dateTimes[i] > maxDateTime)  
 {  
 maxDateTime = dateTimes[i];  
 IndOfMax = i;  
 }  
 }  
 return filestypeofFT[IndOfMax];  
 }  
 static string [] *ReplaceWithMaxDate*(string[] filestypeofFT, byte[] sourceFileData, DateTime maxDateTime)  
 {  
 foreach (string filePath in filestypeofFT)  
 {  
 File.*WriteAllBytes*(filePath, sourceFileData);  
 File.*SetLastWriteTime*(filePath, maxDateTime);  
 }  
 return filestypeofFT;  
 }  
 static string *GetFileFromDirectory*(string directoryPath, string fileType)  
 {  
 string[] files = Directory.*GetFiles*(directoryPath, fileType, SearchOption.*AllDirectories*);  
 return files[0];  
 }  
}