НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
"___"_____2016 р.

Дипломна робота

першого (бакалаврського) рівня вищої освіти
(першого (бакалаврського), другого (магістерського))
зі спеціальності 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код та назва напряму підготовки)

на тему: Програма порівняння та синхронізації каталогів файлів

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-22
(шифр групи)

Куц Михайло Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, Британ О.А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль доцент, Британ О.А. ___
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший (Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту
Куцу Михайлу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Програма порівняння та синхронізації каталогів файлів

керівник проекту (роботи) Бритов Олексій Анатолійович, доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__»_____ 2016 р. №_____

2. Строк подання студентом проекту (роботи) 14.06.2016

3. Вихідні дані до проекту (роботи) : програма для порівняння каталогів з можливістю порівняння окремих файлів, збереження сесій, формування звітів.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Провести огляд існуючих рішень для синхронізації каталогів файлів і порівняння файлів;
2. Провести аналіз існуючих алгоритмів, розробка алгоритму;
3. Написати програмний продукт, що буде працювати без попереднього встановлення і запускатись виконавчим файлом;
4. Тестування програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   1. Принцип WPF – плакат.
   2. Інфраструктура .Net – плакат.
   3. Архітектура програми – плакат.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Розробка алгоритму	28.02.2016	
4	Дослідження .Net framework	10.03.2016	
5	Дослідження технології WPF та .Net framework	15.03.2016	
6	Розробка програми	25.03.2016	
7	Тестування програми	25.04.2016	
8	Оформлення дипломної роботи	31.05.2016	
9	Отримання допуску до захисту та подача роботи в ДЕК	14.06.2016	

Студент _____ Куц М. С.
(підпис) (ініціали, прізвище)

Керівник роботи _____ Бритов О. А.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Куца Михайла Сергійовича

на тему «Програма порівняння та синхронізації каталогів файлів»

Дана дипломна робота присвячена розробці програмного продукту для порівняння та синхронізації каталогів файлів. Метою роботи є розробка програми для роботи з файлами та каталогами.

В роботі розглянуто загальну структуру організації каталогів файлів в операційній системі Windows, розроблено алгоритм для порівняння каталогів файлів включаючи їх ієрархію, розроблено алгоритм для порівняння вмісту окремих файлів, проведено тестування алгоритму порівняння файлів, проведено аналіз результатів та на їх основі створено програмний продукт. Проведено тестування програмного продукту.

Загальний обсяг роботи: 93 сторінки, 45 рисунків, 11 таблиць, 14 бібліографічних найменувань.

Ключові слова: операційна система, рекурсія, каталог файлів, файлова система.

АННОТАЦИЯ

бакалаврской дипломной работы Куца Михаила Сергеевича

на тему «Программа сравнения и синхронизации каталогов файлов»

Данная дипломная работа посвящена разработке программного продукта для сравнения и синхронизации каталогов файлов. Целью работы является разработка программы для работы с файлами и каталогами.

В работе рассмотрена общая структура организации каталогов файлов в операционной системе Windows, разработан алгоритм для сравнения каталогов файлов включая их иерархию, разработан алгоритм для сравнения содержимого отдельных файлов, проведено тестирование алгоритма сравнения файлов, проведен анализ результатов и на их основе создан программный продукт. Проведено тестирование программного продукта.

Общий объем работы: 93 страницы, 45 рисунков, 11 таблиц, 14 библиографических наименований.

Ключевые слова: операционная система, рекурсия, каталог файлов, файловая система.

ANNOTATION

a bachelor’s degree work of Kuts Milhail

entitled " Program for merging(comparing) files and folders"

This project is devoted to the research of software tools for for comparing and synchronizing file`s directories. The aim is to develop a program to work with files and directories.

The project covers the following: overall structure of directories of files in the operating system Windows; developed algorithm for directories comparison; developed algorithm for file comparison; carried analysis of the result; software tasting.

Total volume of work: 93 pages, 45 figures, 11 tables, 14 bibliographic titles.

Keywords: operating system, recursion, file directory, file system.

ЗМІСТ

Перелік умовних позначень

.Net - програмна технологія, запропонована фірмою Microsoft, для розробки програм і веб-застосунків.

WPF - Windows Presentation Foundation — графічна (презентаційна) підсистема в складі .NET Framework.

XAML - eXtensible Application Markup Language - декларативна мова розмітки.

XML - Extensible Markup Language – стандарт побудови мов розмітки ієрархічно структурованих даних

C# - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET..

NTFS – New Technology File System — стандартн файлової системи для сімейства операційних систем Microsoft Windows NT.

FAT – File Allocation Table – стандарт файлової системи.

ОС – операційна систем.

ВСТУП

Задача ефективного порівняння каталогів файлів є дуже актуальною, адже часто виникає проблема об’єднання чи копіювання одного каталогу файлів в інший, надаючи при цьому широкі можливості для маніпулювання різними файлами під час об’єднання.

Організація папок в ОС Windows може бути доволі складною, адже ця операційна система підтримує ієрархічну структуру організації папок[12]. Це означає, що при порівнянні чи об’єднанні двох папок потрібно буде переглянути всю ієрархію папок та вміст цих папок. Таке порівняння вручну може зайняти дуже багато часу, особливо, враховуючи час який знадобиться на порівняння вмісту файлів.

Також необхідність об’єднання папок часто може виникати при над масштабними проектами, де може виникати потреба зберегти стару версію проекти, а потім об’єднати цю версію з новою.

Отже, ієрархічна структура папки може бути складною та доволі глибокою, може містити багато файлів на кожному з рівнів, що може сильно заплутати користувача і для нього буде дуже складно об’єднати ці дві папки згідно до його потреб та вимог. Тому потрібна програма, що порівнювала б усі файли у всій ієрархічній структурі і формувала б звіт про порівняння для подальшого об’єднання.

Метою даної роботи є проведення глибокого аналізу та порівняння різноманітних методів та технологій призначених для об'єднання папок, та розробка програмного продукту, що відповідатиме поставленим до нього вимогам та вирішуватиме наступні задачі:

1. Робота без попереднього встановлення.

3. Порівняння окремих файлів та порівняння директорій

4. Порівняння побітове файлів, з можливістю заміщення одного файлу іншим за вибором користувача

5. Порівнювати всі директорії і файли по всій ієрархії

6. Застосовувати фільтри до порівнюваних файлів

7. Зберігати та відновлювати сеанси.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Аналіз існуючих рішень і алгоритмів

2. Розробка алгоритму

3. Розробка структури програмного продукту.

4. Аналіз та вибір програмних засобів для реалізації.

5. Розробка програми згідно до вимог.

1 ВІДОМІ РІШЕННЯ ТА ТЕХНОЛОГІЇ

1.1 Платні рішення

1.1.1 Beyond Compare 3

1.1.1.1 Загальний опис

Домашня сторінка: http://www.scootersoftware.com/

Програма для порівняння файлів і папок, в тому числі архівованих або розташованих на віддаленому сервері. За допомогою Beyond Compare 3 також можна порівнювати ключі реєстру, зображення, MP3-теги, синхронізувати каталоги, редагувати вихідний код і виконувати безліч інших завдань.

Beyond Compare зручний за рахунок використання вкладок в інтерфейсі: це дозволяє працювати з декількома сесіями паралельно і в одному вікні. Крім того, можна створювати робочі простору (workspaces) - колекції відкритих сесій.

Важливі відмінності цього продукту від більшості аналогів - наочність порівняння і зручність редагування у всіх доступних режимах. Скажімо, порівняння бінарних файлів передбачає також правку за допомогою HEX-редактора. При роботі з зображеннями відмінності відображаються як пікселі, не кажучи вже про повноцінне редагування файлів. MP3-файли можна порівнювати по метаданих. Підтримка додаткових форматів забезпечується плагінами.

В режимі порівняння файлів використовується традиційний двухпанельний інтерфейс (Рисунок. 1.1). Колір шрифту дозволяє відзначити важливі відмінності (червоне виділення) і менш істотні (синє виділення), фоновий колір також несе функціональне значення. Для редагування тексту наявний повноцінний редактор вихідного коду з підтримкою відповідних форматів (див. Tools → File Formats).

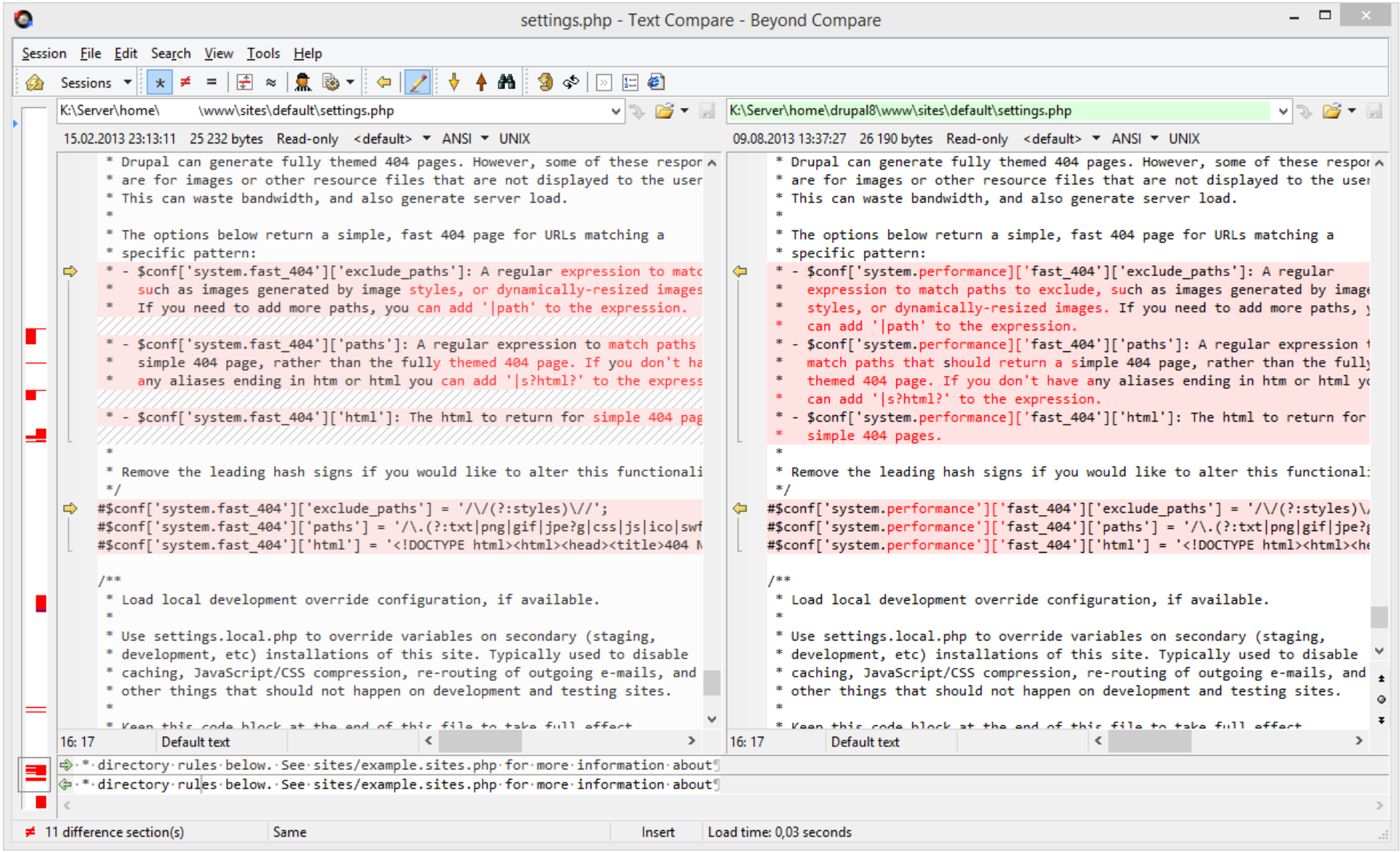

settings.php - Text Compare - Beyond Compare
Session File Edit Search View Tools Help
Sessions
K:\Server\home\ \www\sites\default\settings.php
15.02.2013 23:13:11 25 232 bytes Read-only <default> ANSI UNIX
K:\Server\home\drupal8\www\sites\default\settings.php
09.08.2013 13:37:27 26 190 bytes Read-only <default> ANSI UNIX
* Drupal can generate fully themed 404 pages. However, some of these respon
* are for images or other resource files that are not displayed to the user
* This can waste bandwidth, and also generate server load.
*
* The options below return a simple, fast 404 page for URLs matching a
* specific pattern:
* - $conf['system.fast_404']['exclude_paths']: A regular expression to matc
* such as images generated by image styles, or dynamically-resized images
* If you need to add more paths, you can add '|path' to the expression.
* - $conf['system.fast_404']['paths']: A regular expression to match paths
* simple 404 page, rather than the fully themed 404 page. If you don't ha
* any aliases ending in htm or html you can add '|s?html?' to the express
* - $conf['system.fast_404']['html']: The html to return for simple 404 pag
*
* Remove the leading hash signs if you would like to alter this functionali
*/
#$conf['system.fast_404']['exclude_paths'] = '/\/(?:styles)\//';
#$conf['system.fast_404']['paths'] = '/\.(?:txt|png|gif|jpe?g|css|js|ico|swf
#$conf['system.fast_404']['html'] = '<!DOCTYPE html><html><head><title>404 N
/**
* Load local development override configuration, if available.
*
* Use settings.local.php to override variables on secondary (staging,
* development, etc) installations of this site. Typically used to disable
* caching, JavaScript/CSS compression, re-routing of outgoing e-mails, and
* other things that should not happen on development and testing sites.
*
* Keep this code block at the end of this file to take full effect
* - $conf['system.performance']['fast_404']['exclude_paths']: A regular
* expression to match paths to exclude, such as images generated by image
* styles, or dynamically-resized images. If you need to add more paths,
* can add '|path' to the expression.
* - $conf['system.performance']['fast_404']['paths']: A regular expression
* match paths that should return a simple 404 page, rather than the fully
* themed 404 page. If you don't have any aliases ending in htm or html yo
* can add '|s?html?' to the expression.
* - $conf['system.performance']['fast_404']['html']: The html to return for
* simple 404 pages.
#$conf['system.performance']['fast_404']['exclude_paths'] = '/\/(?:styles)\/
#$conf['system.performance']['fast_404']['paths'] = '/\.(?:txt|png|gif|jpe?g
#$conf['system.performance']['fast_404']['html'] = '<!DOCTYPE html><html><he
16: 17 Default text
* directory rules below. See sites/example.sites.php for more information about
* directory rules below. See sites/example.sites.php for more information about
11 difference section(s) Same Insert Load time: 0,03 seconds

Рисунок 1.1 - Інтерфейс програми BeyondCompare, порівняння текстових файлів[1]

При порівнянні каталогів використовується такий самий двухпанельний вид відображення (Рисунок 1.2). Для показу відмінностей застосовуються маркування та значки, значення яких легко виявити в документації. Можна порівнювати директорії як побайтово (повільно), так і за вказаними властивостями, таким як дата або розмір (швидкий метод).

Відображення даних регулюється за допомогою файлового фільтра (вивід зазначених типів файлів) або фільтра відображення (показ змін). Всі необхідні інструменти винесені на панель інструментів праворуч від вкладок. В меню Actions знаходяться команди, пов'язані з синхронізацією і порівнянням. Управління відображенням є в меню View.

У порівнянні каталогів можуть бути задіяні різні джерела - не тільки локальні, але також і віддалені каталоги, архіви (підтримуються формати RAR, ZIP, 7z і т. д.), знімки файлової структури (snapshots). Завдяки попереднього перегляду, можна заздалегідь ознайомитися з результатом синхронізації.

Рутинні завдання автоматизуються за допомогою скриптів. У Beyond Compare є підтримка командного рядка і регулярних виразів. Звіт доступний для друку або виведення в форматах HTML і TXT.

Beyond Compare являє собою багатофункціональне рішення для синхронізації, злиття, редагування різних типів даних. Нарівні з функціональністю, дуже зручний вкладковий інтерфейс з підтримкою сесій і робочих просторів. В результаті, програма хороша при роботі як з малими, так і з досить складними проектами.

Плюси:

- Зручний інтерфейс
- Робота з архівами і віддаленими каталогами
- Можливості автоматизації
- Доступні фільтри і режими відображення

Мінуси: ціна, так як ліцензія коштує близько 60$ для одного користувача[1], що робить її доволі дорогою для використання.

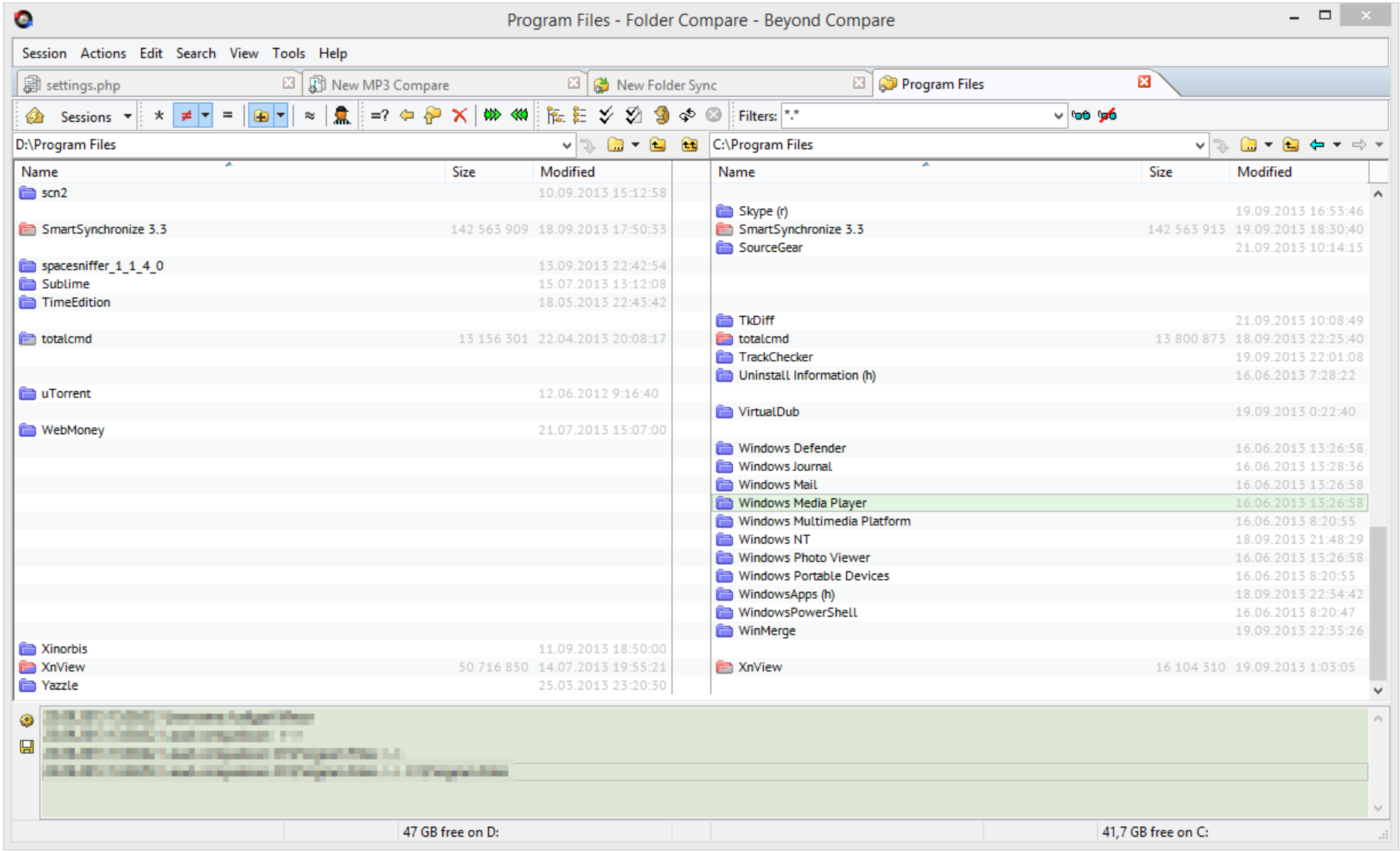


Рисунок 1.2 - Інтерфейс програми BeyondCompare при порівнянні директорій[1]

1.1.2 Compare ++

1.1.2.1 Загальний опис

Домашня сторінка: http://cmpp.coodesoft.com/

Compare ++ являє собою програму для порівняння директорій і текстових файлів в ОС Windows. Позиціонується, перш за все, як інструмент для програмістів і веб-розробників: завдяки розпізнаванню функцій, класів, документ з вихідним кодом легко структурується, зручний в навігації, редагуванні.

Оболонка Compare ++ дуже проста і нічим не перевантажена. Збереження сесій не передбачено, замість цього можна скористатися історією. Вкладочний інтерфейс дозволяє працювати з декількома проектами одночасно.

Серед основних можливостей - порівняння файлів, каталогів і тристороннє злиття. При порівнянні даних можна швидко перемкнутися в один з двох режимів, орієнтованих на роботу з текстом або кодом. У другому випадку, на панелі Function View виводиться список класів, до кожного з яких легко перейти в документі і провести з ним потрібні дії (наприклад, злиття) парою кліків миші. Взагалі кажучи, Function View - не така і рідкість для редакторів вихідного коду, але для програм даної категорії це одиничний випадок.

З інших особливостей редагування: передбачена нумерація рядків, підсвічування синтаксису (C / C ++, Java, C #, Javascript, CSS і мн. Ін.), В тому числі парних дужок. Всі внесені в документах правки маркуються: текст був змінений, видалений або доданий. Результат порівняння можна експортувати в diff-, txt- або html-файл і, опціонально, відправити на email.

Порівняння директорій здійснюється без попереднього перегляду. Синхронізація вмісту архівів і з віддаленим сервером не передбачена, зате можна зберігати знімки файлової структури і застосовувати фільтри для файлів і каталогів. На панелі інструментів також можна виявити додаткові режими відображення. Порівняння здійснюється за властивостями файлу - датою модифікації та розміру.

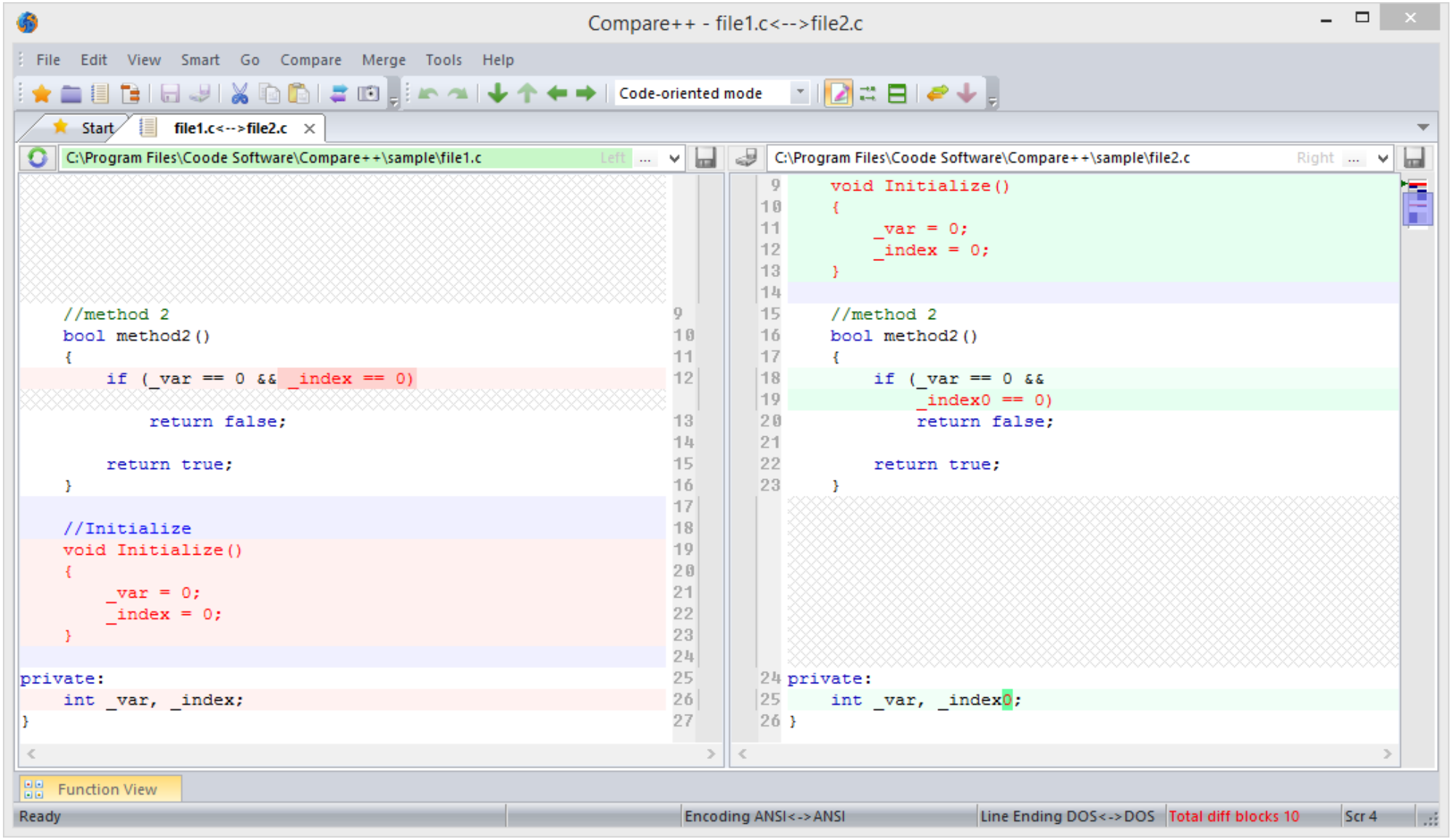


Рисунок 1.3 - Інтерфейс Compare++[2]

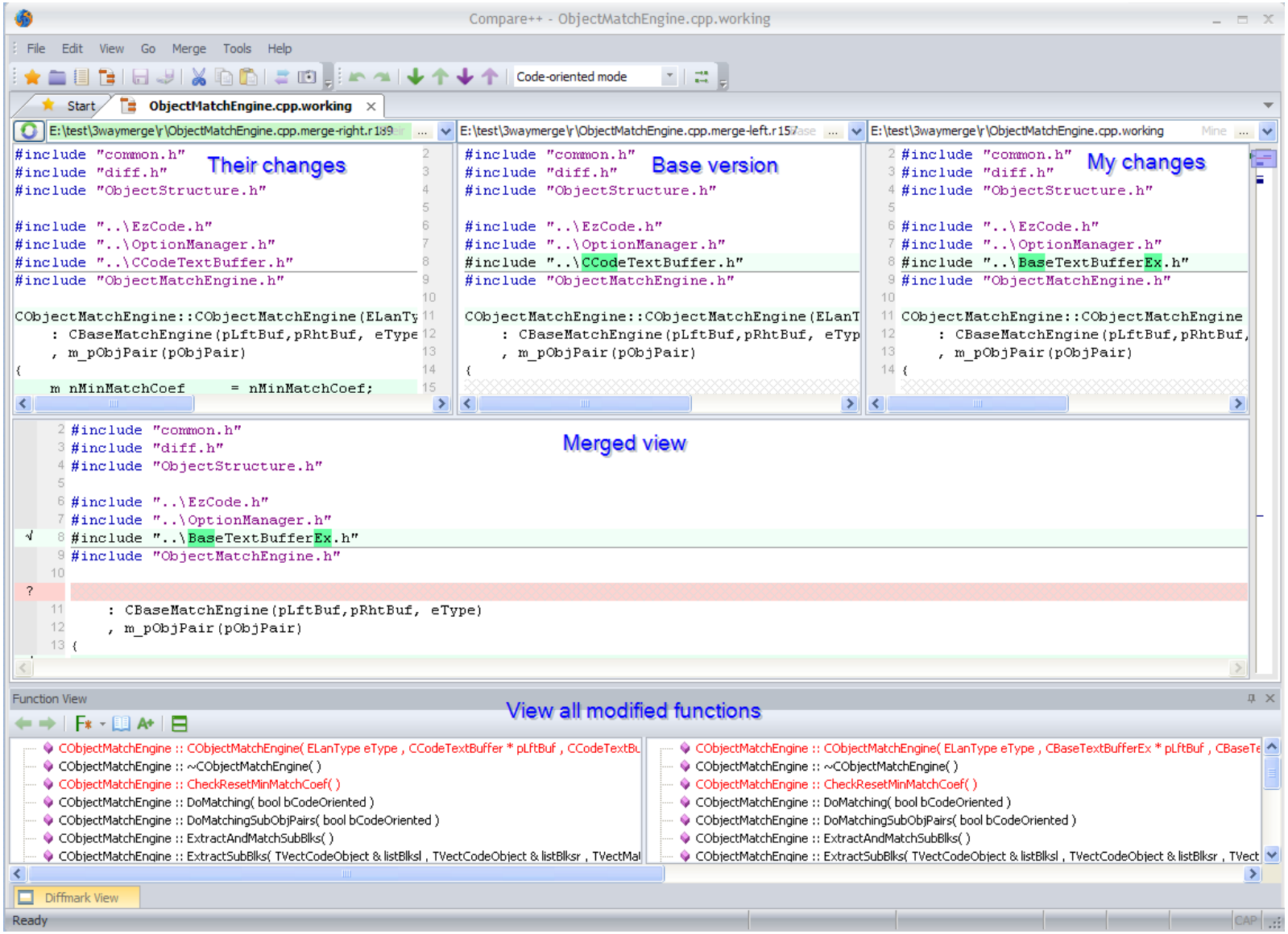


Рисунок 1.4 - Тристороннє злиття Compare++[2]

Передбачені деякі зручні особливості:

- Інтеграція в контекстне меню Провідника Windows
- Зручний і підлаштовуємий призначений для користувача інтерфейс
- Інтеграція з іншими продуктами, такими як система управління версіями з інтерфейсом командного рядка
- Багатомовний інтерфейс

Compare++ можна розглядати як добре настроюваний редактор вихідного коду, з функціями порівняння і синхронізації. І навпаки - як програму для синхронізації з редакторськими функціями. У зв'язці з інтеграційними можливостями, програма буде корисна як доповнення до IDE-середовищі.

Плюси:

- Зручна робота з вихідним кодом
- Тристороннє злиття
- Створення diff-патчів і звітів

Мінуси:

- Відсутня повноцінна підтримка сесій
- Відсутність розширень

1.1.3 Araxis Merge

1.1.3.1 Загальний опис

Домашня сторінка: http://www.araxis.com/merge/

Araxis Merge - програма для візуального порівняння, злиття і синхронізації папок. Вбудований редактор розпізнає різні формати документів: вихідний код, веб-сторінки, XML, PDF, Microsoft Office, зображення і т. п. Також в Merge передбачена інтеграція з популярними системами управління версіями і іншими середовищами розробки.

У Merge задіяний вкладковий інтерфейс, підтримується збереження сеансів і робочих просторів з усіма параметрами в окремий файл. Стрічка Ribbon розділена на секції, завдяки цьому розташування команд легко запам'ятовується, інструменти для роботи з текстом завжди під рукою. Всі дії,

пов'язані з редагуванням і навігацією по тексту, доступні на стрічці. Панель інструментів ретельно налаштовується тільки в Mac OS, можливі й інші відмінності між версіями Merge, в залежності від платформи. Розташування панелей легко змінити на вертикальне чи горизонтальне, можна додати додаткові інформаційні колонки. Таким чином, програма зручна, її інтерфейс продуманий до дрібниць.

Merge підтримує 4 режими порівняння. Додаткові режими (порівняння зображень і довічних даних) не такі цікаві через відсутність інструментів редагування. Тому далі мова буде йти про порівняння файлів і директорій.

Текстовий редактор підтримує підсвічування синтаксису, нумерацію рядків. При порівнянні, крім змін, зазначених відповідним кольором, зручно відстежувати зв'язки між рядками за допомогою сполучних ліній (Linking lines). Навівши курсор на відповідний блок, можна застосувати для нього операцію злиття з сусіднім файлом (функція Point-and-click merging). Тут вловлюється аналогія зі згаданою в першій частині огляду програмою SmartSynchronize. У документ можна додавати закладки і коментарі. Експорт звітів, з занесенням всіх відмінностей, здійснюється в форматах DIFF, HTML, HTML-слайдшоу і XML.

Другий основний режим роботи Araxis Merge - порівняння і синхронізація каталогів. Сильна сторона цього інструменту - підтримка різних джерел: віртуальна файлова система, мережеві диски, проекти та інші джерела. При порівнянні, доступні дві панелі з відображенням структури каталогів, також нескладно активувати режим тристоронньої злиття.

Конфігурація фільтрів для директорій і файлів доступна в розділі Filters налаштувань. Вони діляться на візуальні (висновок тільки потрібних даних) і фільтри вибору (вибір файлів за заданими критеріями).

У переліку доступних операцій - об'єднання папок, побайтово порівняння і порівняння за розміром і датою. При автосінхронізації файли з конфліктами не обробляються і відкладаються для прийняття користувачем рішення.

Потенційні можливості Araxis Merge значно зростають, якщо брати в розрахунок інтеграцію з Mercurial, Git, Subversion, Perforce .

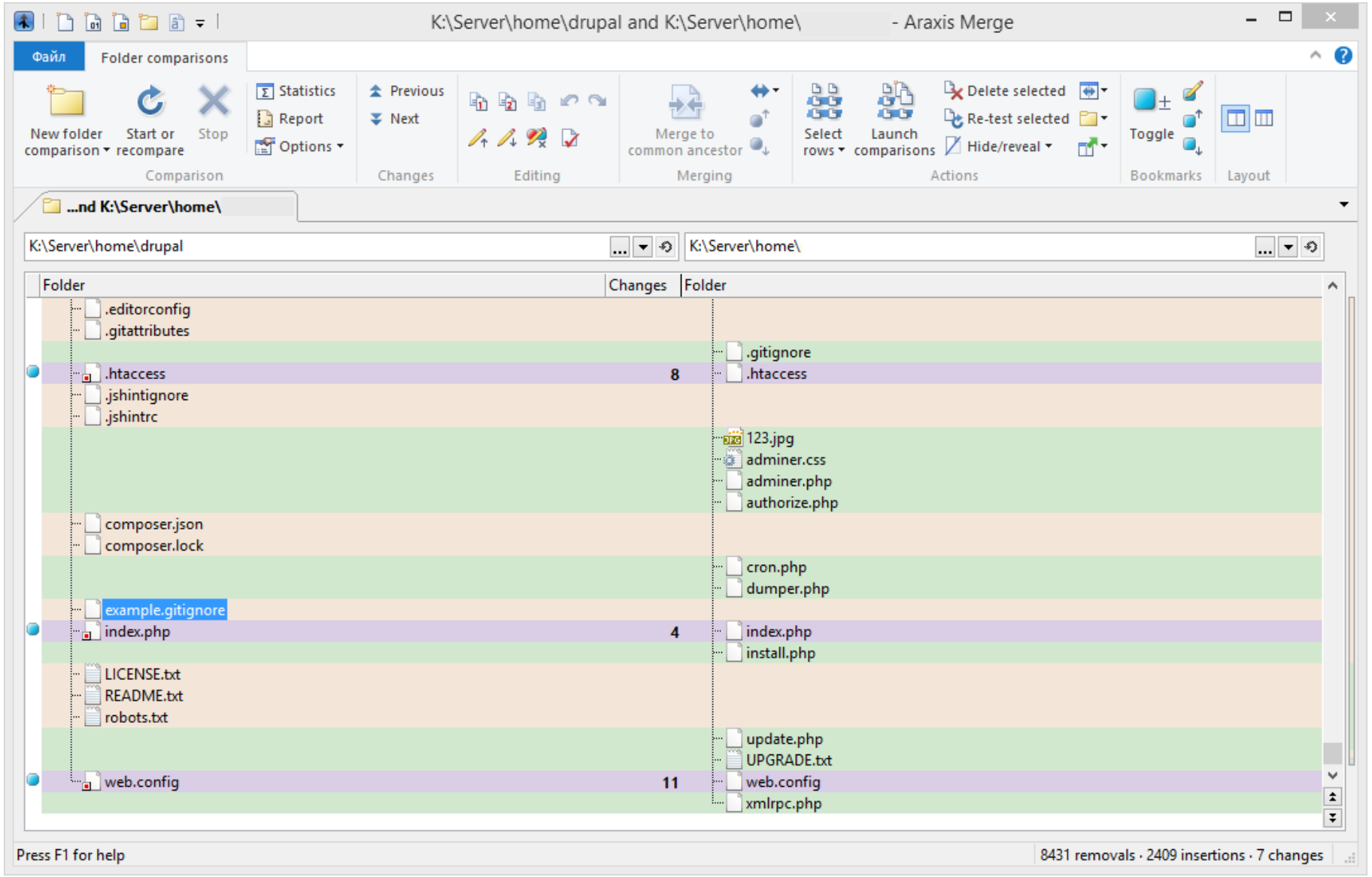

K:\Server\home\drupal and K:\Server\home\ - Araxis Merge
Файл
Folder comparisons
New folder comparison
Start or recompare
Stop
Statistics
Report
Options
Comparison
Previous
Next
Changes
Editing
Merge to common ancestor
Merging
Select rows
Launch comparisons
Delete selected
Re-test selected
Hide/reveal
Actions
Toggle
Bookmarks
Layout
...nd K:\Server\home\
K:\Server\home\drupal
K:\Server\home\
Folder
Changes
Folder
.editorconfig
.gitattributes
.gitignore
.htaccess
8
.htaccess
.jshintignore
.jshintrc
123.jpg
adminer.css
adminer.php
authorize.php
composer.json
composer.lock
cron.php
dumper.php
example.gitignore
index.php
4
index.php
install.php
LICENSE.txt
README.txt
robots.txt
update.php
UPGRADE.txt
web.config
11
web.config
xmlrpc.php
Press F1 for help
8431 removals · 2409 insertions · 7 changes

Рисунок 1.5 - Araxis Merge порівняння директорій[3]

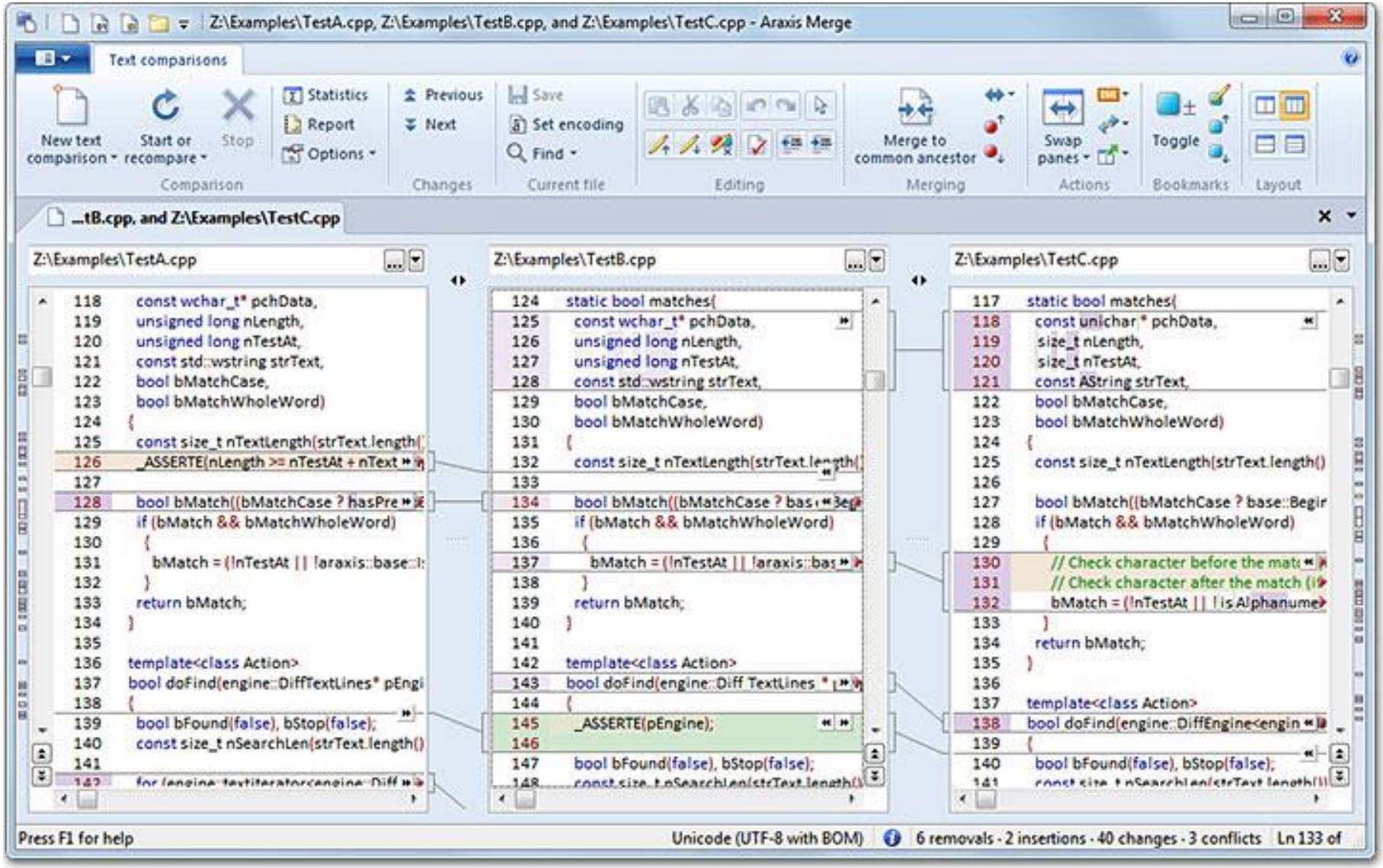


Рисунок 1.6 - Araxis Merge порівняння файлів Cpp[3]

Araxis Merge - програма з добре продуманим інтерфейсом, оптимальним набором інструментів для порівняння, синхронізації і злиття даних. З побажань - не вистачає додаткових режимів порівняння даних, в поєднанні зі зручним редагуванням, як у випадку з Beyond Compare.

Плюси:

- Інтеграція з середовищами розробки
- Зручне візуальне порівняння
- Перемикання відображення
- Статистика та звітність

1.1.4 UltraCompare Professional

1.1.4.1 Загальний опис

Домашня сторінка: http://www.ultraedit.com/products/ultracompare.html

UltraCompare дозволяє порівнювати текстові файли, документи Word, виконавчі файли, каталоги і архіви, локальні, віддалені директорії і знімні носії. Підтримуються автоматична синхронізація, тристороннє злиття, пошук дублікатів та інші допоміжні операції.

Інтерфейс UltraCompare передбачає як сесії, так і робочі простори. Він зручний, не в останню чергу, за рахунок швидкого доступу до файлових операцій: наприклад, можна вибрати для порівняння потрібні файли прямо з бокової панелі. Тут же, в лівій панелі, можна створити фільтр. Додаткові опції можна знайти в розділі Ignore Options налаштувань.

На вибір надані наступні режими роботи з даними: порівняння тексту і папок (плюс тристоронє злиття для обох видів), двійкових файлів, синхронізація папок. Зміна режиму здійснюється через меню Mode. Вибравши його, можна вказати джерела для лівої і правої панелей, перетягнувши їх у вікно і натиснувши кнопку Go.

При порівнянні тексту, рядки з відмінностями маркуються фоновим кольором і позначаються символами поруч з нумерацією. Сполучні лінії дозволяють відстежити зв'язок між двома документами і допомагають при злитті, відповідні команди доступні в розділі меню Merge. Для редагування

вихідного коду рекомендується вдатися до інструментарію програми UltraEdit. Виникає питання: наскільки доцільно оплачувати одну з додаткових можливостей, якщо вартість цього редактора вище, ніж UltraCompare.

Для рекурсивного і не рекурсивного порівняння доступні локальні, віддалені і знімні джерела, також підтримуються ZIP-, RAR-, JAR- архіви. Результати порівняння відображаються по обидві панелі (джерело - одержувач). Порівняння здійснюється за розміром і віком або вмістом файлів.

Користувач має можливість створювати правила синхронізації (rules): заміна файлів, видалення застарілих елементів, копіювання та інші. Можлива синхронізація за розкладом. Підтримується командний рядок, в наявності інтеграція з додатками для контролю версій: AnkhSVN Perforce, QVCS, Subversion, TortoiseCVS, TortoiseSVN і іншими.

Інструментарій UltraCompare включає в себе найбільш затребувані режими порівняння, злиття і синхронізації, дозволяючи задіяти різні джерела даних. Широкі інтеграційні можливості за допомогою командного інтерфейсу. Начебто, все на користь розробнику, але редактор без підсвічування синтаксису - це явний недолік.

Плюси:

- Порівняння архівів і віддалених каталогів
- Інтеграція з додатками для контролю версій
- Тристороннє порівняння

Мінуси:

- Обмеження вбудованого редактора

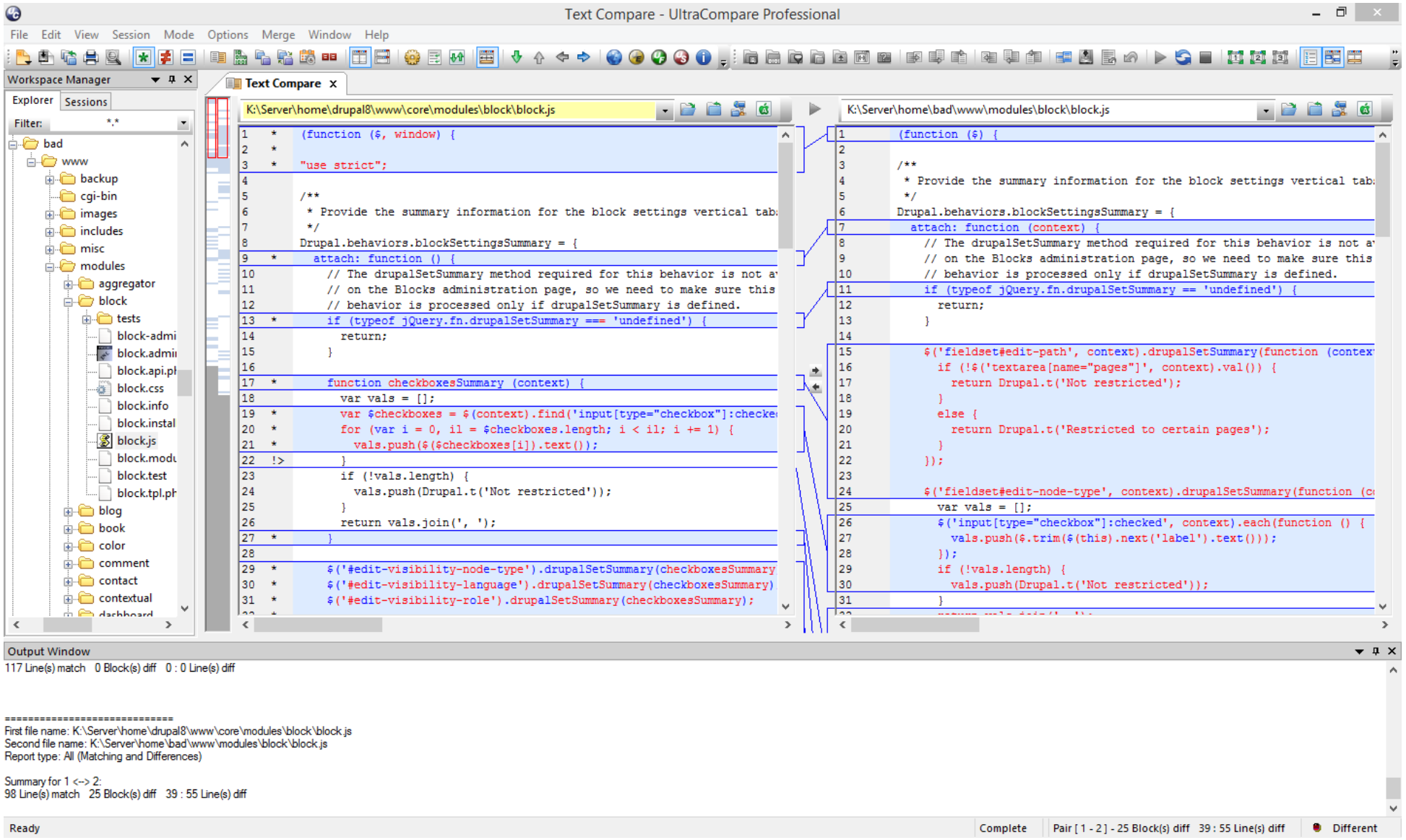


Рисунок 1.7 - UltraCompare порівняння JS файлів[4]

1.1.5 ExamDiffPro

1.1.5.1 Загальний опис

Домашня сторінка: http://www.prestosoft.com/edp_examdiffpro.asp

Програма ExamDiff призначена для візуального порівняння текстових, двійкових файлів і директорій. У довгому списку основних функцій можна виявити підтримку плагінів. Завдяки їм, можлива робота з такими форматами, як XML, HTML, PDF, MS Excel, Word, PowerPoint і іншими.

Очевидно, що для редагування коду ExamDiff відкриває широкі можливості, як по функціональності, так і по зручності. Меню View дозволяє налаштувати відображення панелей на розсуд користувача, панель інструментів повністю настроюється.

Редактор підтримує підсвічування синтаксису, нумерацію, в документі можна створювати закладки. Передбачено різні способи навігації по документу: власне, навігаційна панель, випадаючий список, синхронна прокрутка. Багато дій виконуються без зайвих, як часто трапляється, маніпуляцій: наприклад, для переходу до налаштувань колірної схеми достатньо клікнути по маркуванням в статусному рядку, для додавання виключення - поставити прапорець Skip біля списку, а через контекстне меню відкривається доступ до команд плагінів.

Серед додаткових можливостей для синхронізації директорій - підтримка віддалених джерел, архівів. У ExamDiff можна створювати знімки для подальшого порівняння, застосовувати фільтри для різних груп файлів (в настройках є ряд налаштувань), застосовувати фільтри по часу і розміру.

Існує безкоштовна версія ExamDiff. У ній немає тристороннього злиття, порівняння директорій і двійкових файлів.

ExamDiffPro містить необхідні інструменти для редагування, адаптований для роботи з великими документами, вихідним кодом, дозволяючи швидко орієнтуватися по документу і переходити в потрібну ділянку файлу.

Плюси:

- Гнучка настройка
- Розширені функції редагування
- Підтримка розширень

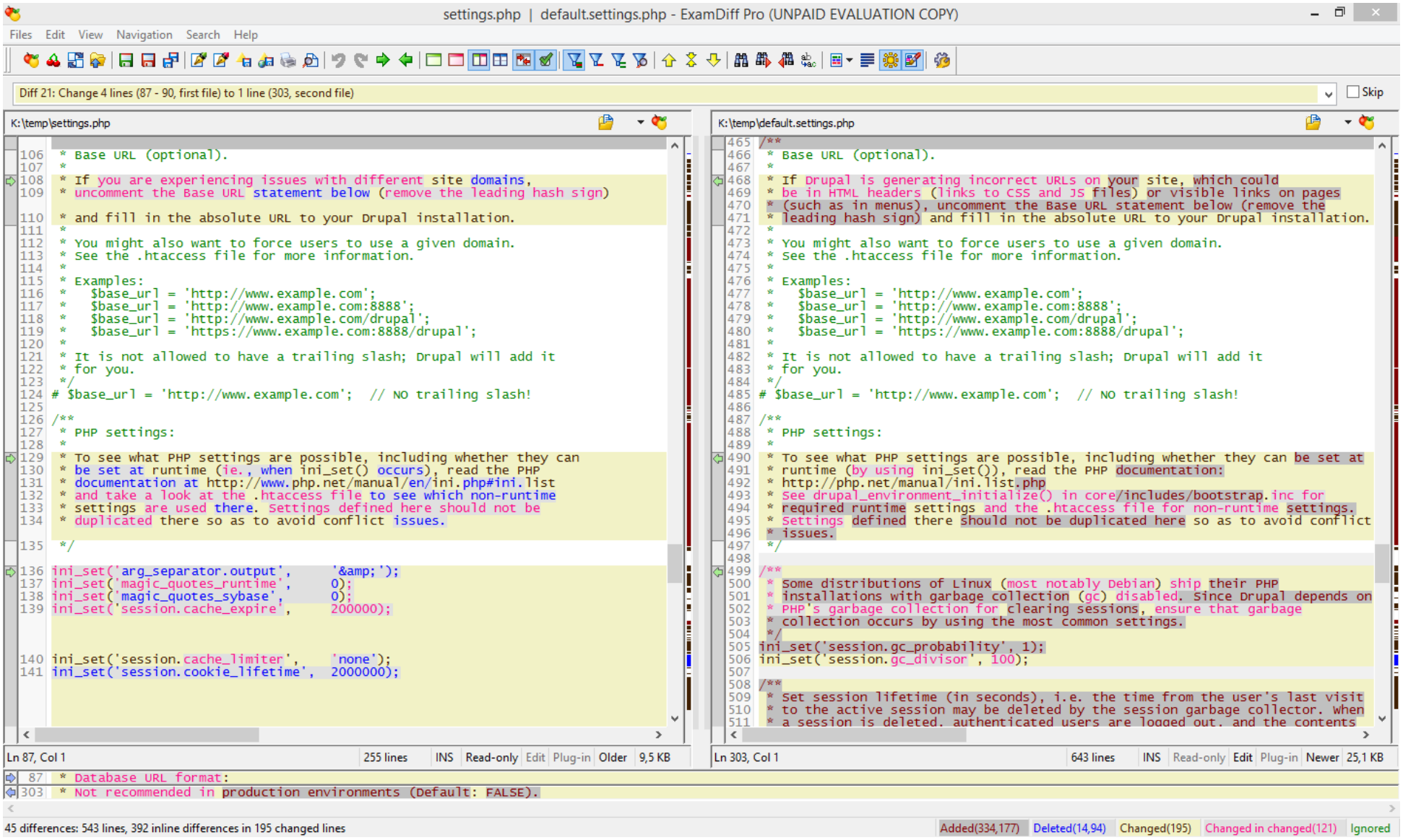


Рисунок 1.8 - ExamDiffPro порівняння PHP файлів[5]

1.2 Безкоштовні рішення

1.2.1 SmartSynchronize

1.2.1.1 Загальний опис

Домашня сторінка: http://www.syntevo.com/smartsynchronize/

Програма SmartSynchronize являється безкоштовною для некомерційного використання, для комерційного використання потрібно придбати ліцензію.

Має вже звичайний для таких програм інтерфейс. Дозволяє зручно порівнювати як директорії так і окремі файли.

Дозволяє проводити тристороннє злиття. Також SmartySynctonize вбудований в програму SmaryGit – зручний Git інструмент, що є також безкоштовним для некомерційної розробки.

Також варто відмітити, що в SmartySynctonize відсутні проблеми з різними кодуваннями.

Невеликим недоліком редактору можна назвати відсутність підсвітки синтаксису для мов програмування.

SmartySynctonize включає в себе основні необхідні інструменти для редагування, адаптовані для роботи з директоріями, не підсвічує синтаксис коду, вбудований в SmartGit.

Плюси:

- тристороннє злиття;
- немає проблем з кодуваннями;
- крім файлів, може порівнювати директорії.

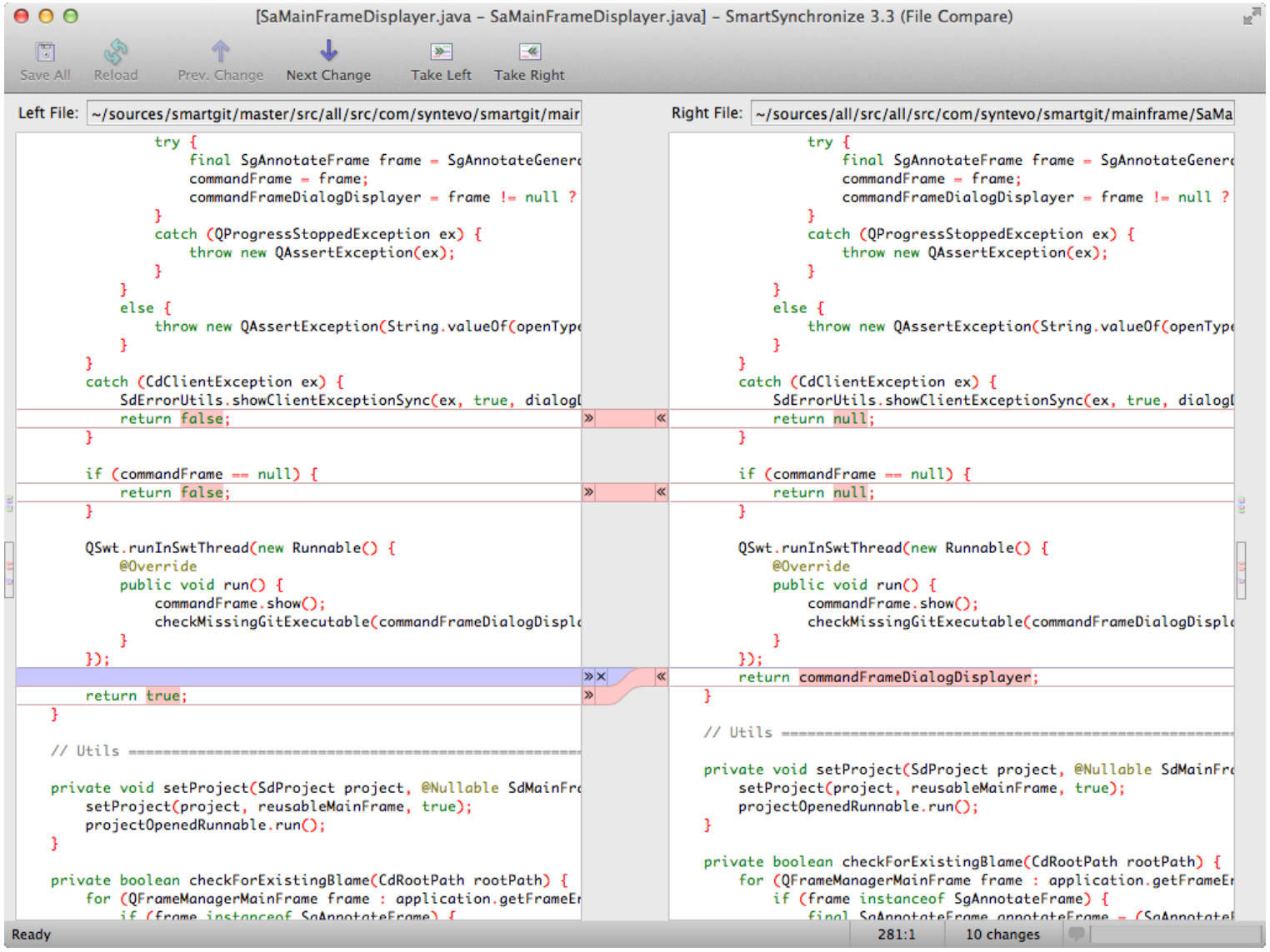


Рисунок 1.9 - інтерфейс програми SmartSynchronize[6]

1.2.2 WinMerge

1.2.2.1 Загальний опис

Домашня сторінка: http://www.winmerge.org

WinMerge - безкоштовне ПЗ з відкритим вихідним кодом для порівняння і синхронізації файлів і директорій, яке розповсюджується на умовах ліцензії GNU GPL для Microsoft Windows[7].

Основні можливості:

- Візуальне порівняння і синхронізація текстових файлів
- Гнучкий редактор з підсвічуванням синтаксису, можливістю показу номерів рядків і автоматичним перенесенням тексту.
- Підтримує формати текстових файлів DOS, UNIX і Макінтош
- порівняння каталогів
- Вміє створювати патчі
- Підсвічування синтаксису для багатьох мов програмування, таких як C / C ++, Java, Pascal, Basic, JavaScript, Python
- Підтримка декількох кодових сторінок
- Інтеграція з системами контролю версій TortoiseSVN (Subversion), TortoiseHg (Mercurial), Microsoft Visual SourceSafe і Rational ClearCase

Програма WinMerge створена виключно для Windows OS є доволі зручною у використанні. Має широку інтеграцію в системи контролю версій.

Плюси:

- Open Source;
- ніяких проблем з кодуваннями;
- підсвічування синтаксису вбудоване;

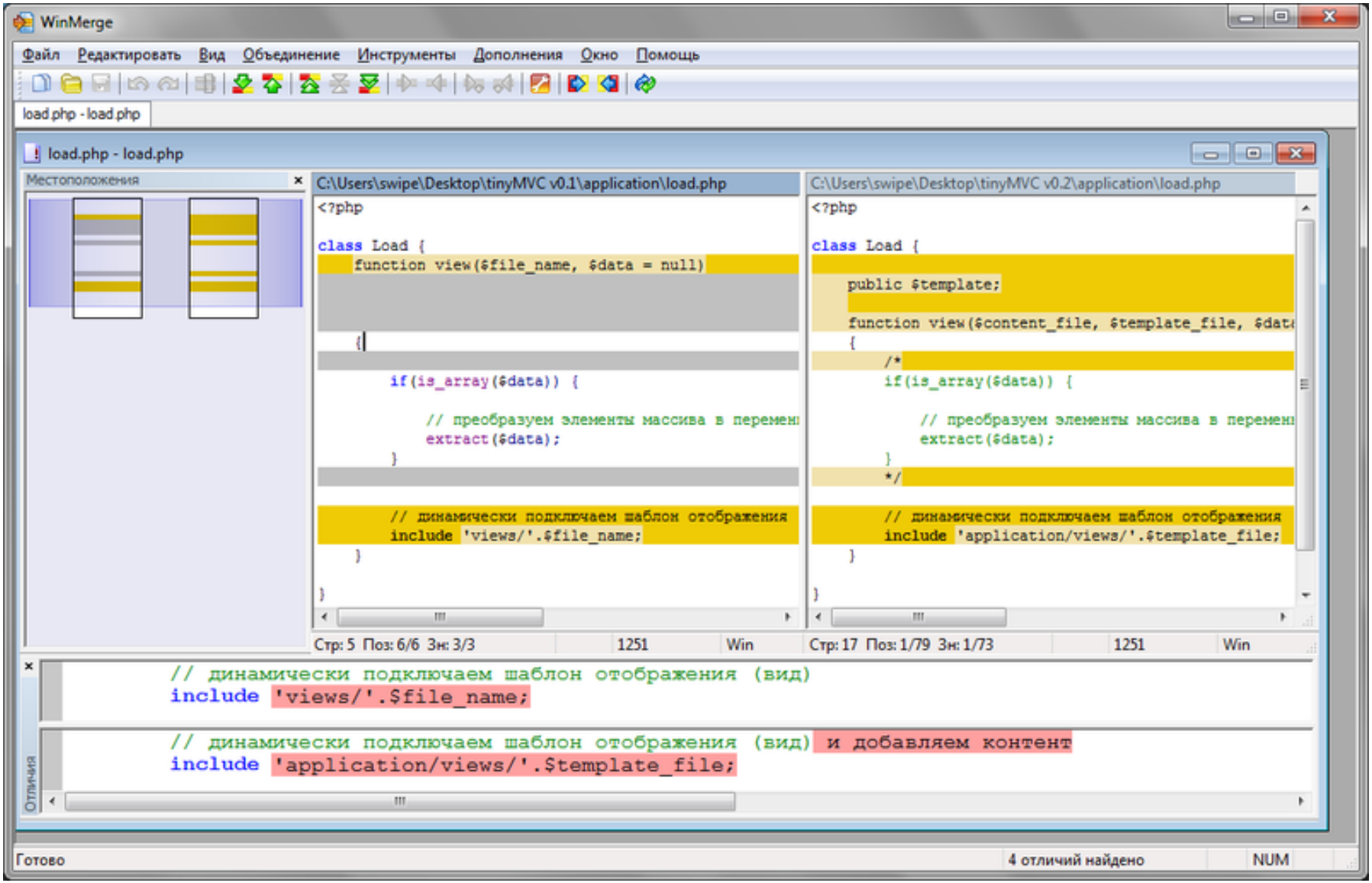


Рисунок 1.10 - Інтерфейс WinMerge[7]

1.2.3 Meld

1.2.3.1 Загальний опис

Домашня сторінка: http://meldmerge.org/

Meld - безкоштовна комп'ютерна програма для порівняння вмісту текстових файлів або каталогів. Основні функції:

- Порівняння двох-трьох файлів або каталогів.
- Створення файлів довідки (англ. Patch file) з описом відмінностей між файлами.
- Робота з системами керування версіями Git, Subversion, Mercurial, Bazaar і CVS.
- Вкладковий інтерфейс.

Доволі зручний в роботі інтерфейс. При перегляді відмінностей між двома файлами доволі легко відміняти зміни зажавши клавішу Shift і натискаючи курсором миші на ділянки змін.

Програма Meld не містить в собі нічого особливого, що вирізняло б її з поміж схожих програм.

Плюси:

- GPLv2;
- двостороннє і тристороннє злиття файлів;
- порівняння директорій;
- підсвічування синтаксису (при встановленому GtkSourceView).

Мінуси:

- для установки під Windows потрібно встановити Python, GTK +, Glib, GtkSourceView, що не кожному сподобатися.

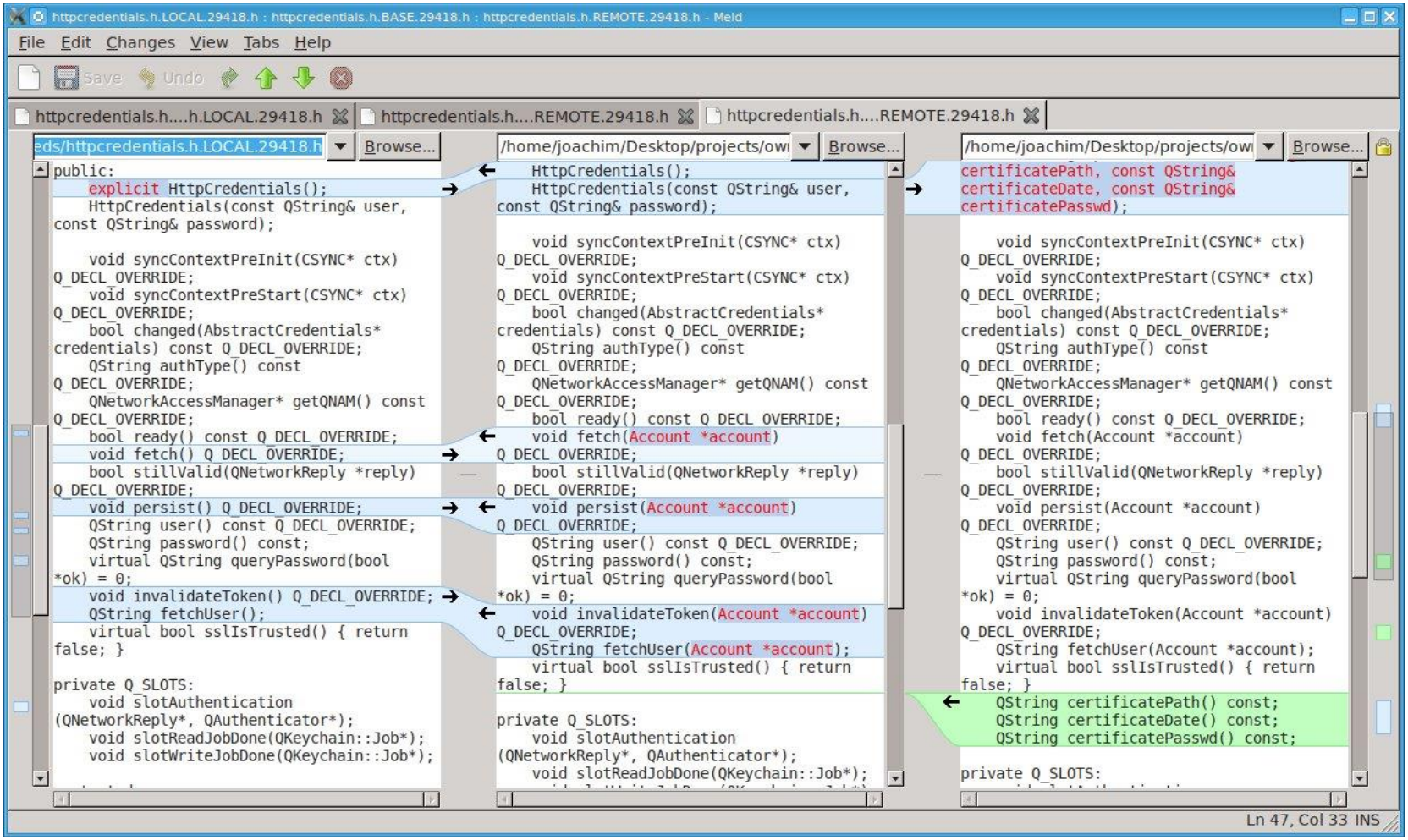


Рисунок 1.11 - Інтерфейс Meld[8]

1.2.4 Diffuse

1.2.4.1 Загальний опис

Домашня сторінка: http://diffuse.sourceforge.net/

Diffuse - безкоштовна програма для порівняння вмісту текстових файлів або каталогів. Ця програма не дуже популярна.

Має звичайний, для такого роду програм, інтерфейс. Підтримує багатостороннє злиття файлів, ця особливість виділяє її з схожих програм.

Diffuse має велику глубину відмін змін(Undo), що є важливим плюсом для програми. Програма не дозволяє порівнювати директорії.

Програма Diffuse є доволі зручною для порівняння файлів завдяки великій глибині відмін і багатосторонньому злитті.

Плюси:

- GPL;
- підтримка 2-way, 3-way і n-way (довільну кількість файлів) злиття;
- підсвічування синтаксису;
- відмінно працює з UTF-8;
- необмежена глибина відмін (Undo);
- зручна навігація по коду.

Мінуси:

- неможливість порівнювати директорії.

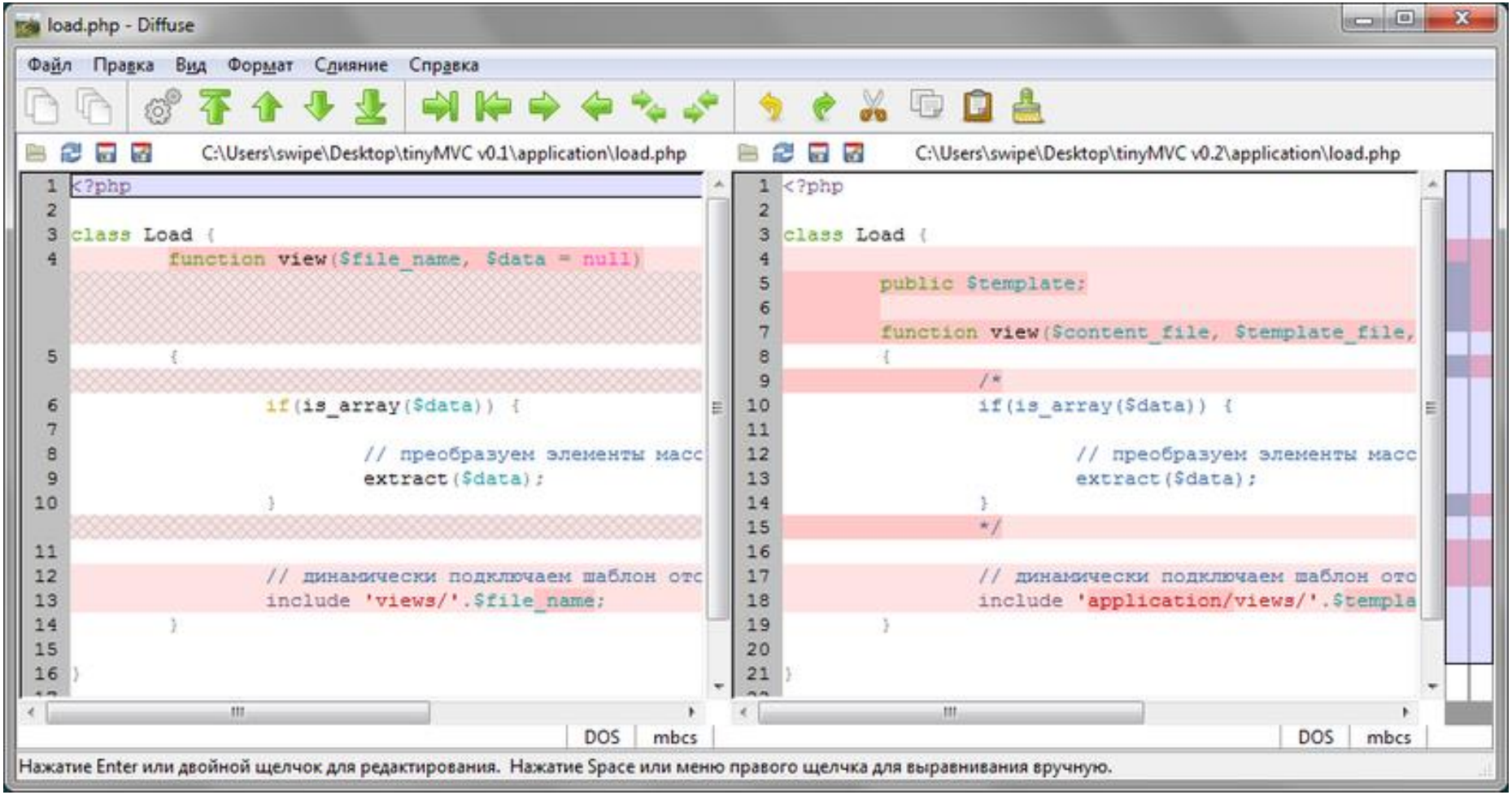


Рисунок 1.12 - Інтерфейс програми Diffuse[9]

1.2.5 Perforce P4 Merge

1.2.5.1 Загальний опис

Домашня сторінка: https://www.perforce.com/product/

Diffuse - безкоштовна програма для порівняння файлів, картинок. Має звичайний, для такого роду програм, інтерфейс. Підтримує тристороннє злиття файлів.

Програма Perforce P4 Merge нічим особливо не вирізняється з-поміж конкурентів. Загалом програма є доволі посередньою, але той факт що вона безкоштовна, робить її доволі конкурентноспроможною.

Плюси:

- підтримка 2-way, 3-way злиття;
- порівняння картинок;
- підсвічування синтаксису;
- відмінно працює з UTF-8;

Мінуси:

- неможливість порівнювати директорії.

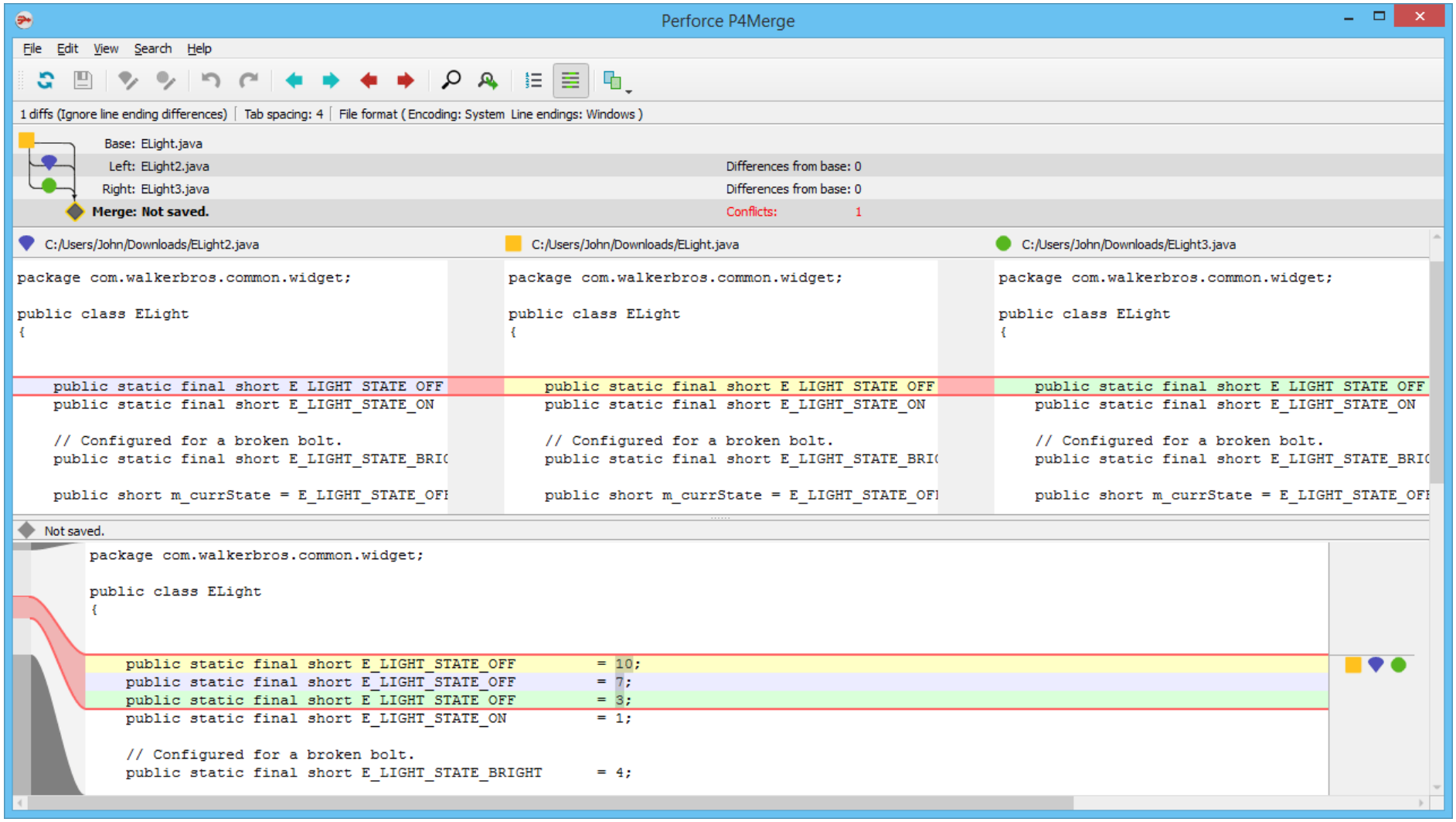


Рисунок 1.13 - Інтерфейс PerforceP4Merge[10]

1.3 Порівняння рішень у вигляді таблиці

Таблиця 1.1 - Порівняння існуючих рішень 1-ша частина

Програма	Розробник	Вартість	Платформи	Експорт	Підтримка CVS
SmartSynchronize	syntevo GmbH	Безкоштовна для неком. використ.	Linux, Mac OS, Windows	HTML	+
WinMerge	Dean Grimm	Безкоштовна	Windows	CSV, HTML, XML, табулированный	+
Meld	Stephen Kennedy	Безкоштовна	BSD, Solaris, Linux, Mac OS, Windows	DIFF	+
Diffuse	Derrick Moser	Безкоштовна	Windows, Mac OS, Linux, BSD	–	+
Perforce P4 Merge	Perforce	Безкоштовна	Windows, Mac OS, Linux, Sun Solaris	–	–
Beyond Compare 3	Scooter Software	$30+	Windows, Linux	XML, HTML, CSV, TXT, Unix Patch	+
Compare++	Coode Software	$29,95	Windows	DIFF, TXT, HTML	+
Araxis Merge	Araxis LTD.	€99+	Mac OS, Windows	DIFF, HTML, HTML-слайдшоу, XML	+
UltraCompare Professional	IDM Computer Solutions, Inc	$49,95	Windows	TXT, DIFF-отчет	+

ExamDiff Pro	PrestoSoft	$34,99	Windows	UNIX, HTML, Diff	+

Таблиця 1.2 - Порівняння існуючих рішень 2-га частина

Програма	**Порівняння**					**Підсвітка синтаксису**	**Сесії**
	Локальних директорій	**Віддалених директорій / архівів**	**Тристороннє**	**Фалів**	**Інші види**		
SmartSynchronize	+	–/–	+	+	–	+	+ (історія, профілі)
WinMerge	+	–/–	–	+	Двійкові (exe) файли	+	+ (проекти)
Meld	+	–/–	+	+	–	+	–
Diffuse	–	–/–	+	+	Багатостороннє злиття	+	–
Perforce P4 Merge	–	–/–	+	+	Зображення	–	–
Beyond Compare 3	+	+/+	+	+	Двійкові файли, зображення, MP3, файли Реєстру	+	+
Compare++	+	–/–	+	+	–	+	+ (історія)
Araxis Merge	+	+/+	+	+	Двійкові файлы, Зображення	+	+
UltraCompare Professional	+	+/+	+	+	Двійкові файли	–	+
ExamDiff Pro	+	+/+	+	+	Двійкойві файли	+	+

1.4 Аналіз завдання. Розробка кінцевого завдання.

Злиття файлів та папок дуже часто використовують в системах контролю версіями, це є основною операцією, яка примиряє кілька змін, внесених у версії набору файлів. Найчастіше, це необхідно, коли можливе створення декількох версій файлу. Коли дві (чи більше версій) файлу зливаються, результат являє собою єдиний набір файлів, який містить усі набори змін.

У деяких випадках злиття може виконуватися автоматично. В інших випадках, людина повинна вирішити, що саме файли повинні містити.

Види злиття:

- Автоматичне злиття:

автоматичне злиття забезпечується програмно. Наприклад, Вікіпедія дозволяє двом користувачам редагувати одну і ту ж статтю одночасно; коли останній учасник зберігається, зміни об'єднуються в статті замість перезапису попереднього набору змін.

- Ручне злиття:

при ручному злитті користувач сам повинен пересортувати (можливо, за допомогою інструментів програмного забезпечення) зміни в файлах, що порівнюються. Наприклад, якщо два файли трохи відрізняються і користувач хоче обрати найкращі зміни з обох, це як правило, може бути досягнуто шляхом об'єднання файлів вручну, вибираючи потрібні зміни з обох джерел. Ручне злиття також потрібно, коли автоматичне злиття працює в конфлікт змін; наприклад, далеко не всі інструменти автоматичного злиття можуть об'єднати дві зміни в одному і тому ж рядку коду. У цих випадках, системи контролю версій звертаються до користувача, щоб вказати результат злиття.

Алгоритми злиття:

- Тристороннє злиття:

тристороннє злиття виконується після автоматизованого аналізу різниці між файлом "A" і файлом "B" також беручи до уваги вихідного файлу "C". Це грубий метод злиття, але широко застосовується, оскільки вимагає тільки одного загального предка, щоб відновити ті зміни, які повинні бути об'єднані.

Тристороннє злиття шукає секцій, які є однаковими в двох з трьох файлів. В цьому випадку є два варіанти розподілу, а також версія, яка

знаходиться в загального предка "C" відкидається, в той час як версія, яка відрізняється зберігається на виході. Розділ однаковий в "A" і "C" виводить змінену версію в "B", і також розділ, який є тим же самим в "B" і "C", виводить версію в "A".

Розділи, які відрізняються у всіх трьох файлах позначені як конфліктна ситуація і залишається для користувача, щоб вирішити.

- Рекурсивне тристороннє злиття:

рекурсивне тристороннє злиття створює віртуального предка шляхом злиття неунікальних предків в першу чергу. Алгоритм рекурсивно об'єднує їх. Оскільки існує кінцеве число версій в історії, процес гарантовано в кінцевому підсумку припиниться. Цей метод використовується інструментом контролю версій Git. (Рекурсивна реалізація злиття Git також обробляє інші незручні випадки)

Рекурсивне тристороннє злиття може використовуватися тільки в тих ситуаціях, коли є інформація про спільне походження DAG (орієнтований ациклічний граф) похідних, які будуть об'єднані. Отже, воно не може бути використано в тих випадках, коли не можна визначити предків.

- Плетене злиття:

це алгоритм, який не використовує загального предка для двох файлів. Замість цього, він відстежує, як поодинокі лінії були додані і видалені в похідних версіях файлу, а також робить об'єднаний файл по цій інформації.

Для кожного рядка в похідних файлах, плетене злиття збирає наступну інформацію: які рядки перед ним, які слідують, і чи був він видалений на якомусь етапі історії. Якщо будь-який похідний рядок був видалений в якийсь момент, віе не повинен бути присутнім в об'єднаній версії.

Кінцеве завдання:

1. Операційна система - Windows.

2. Виконуваний файл запускається без установки.

3. Два режими роботи програми: порівняння окремих файлів, порівняння директорій

4. Порівняння побітове файлів, з можливістю заміщення одного файлу іншим за вибором користувача

5. При порівнянні директорій порівнюються всі файли в них

6. При порівнянні директорій мати можливість додати фільтри на файли

7. Передбачити можливість декількох вкладок і збереження сеансу.

8. Розробити механізм управління вкладками (закриття, створення нових)

9. При порівнянні папок мати можливість вибору папки для злиття(серед цих двох)

1.5 Висновки до розділу 1

В цьому розділі було розглянуто існуючі рішення та зроблено висновки про їх актуальність та функціонал, проведено аналіз платних та безкоштовних рішень з порівняльною характеристикою у вигляді таблиці. Було проведено аналіз завдання та розроблено кінцеве завдання.

2 ФАЙЛОВА СИСТЕМА

2.1 Що таке файлова система

В обчислювальній техніці, файлова система — це спосіб організації даних, який використовується операційною системою для збереження інформації у вигляді файлів на носіях інформації. Також цим поняттям позначають сукупність файлів та директорій, які розміщуються на логічному або фізичному пристрої. Створення файлової системи відбувається в процесі форматування.

В залежності від організації файлів на носії даних, файлові системи можуть поділятись на:

- Ієрархічні файлові системи — дозволяють розміщувати файли в каталоги;
- Плоскі файлові системи — не використовують каталогів;
- Кластерні файлові системи — дозволяють розподіляти файли між кількома однотипними фізичними пристроями однієї машини;
- Мережеві файлові системи — забезпечують механізми доступу до файлів однієї машини з інших машин мережі;
- Розподілені файлові системи — забезпечують зберігання файлів шляхом їх розподілу між кількома машинами мережі;

Без файлової системи інформація, розміщена на певному накопичувачі, являла б собою один великий файл, що робить неможливим пошук та отримання необхідної інформації з нього. Є багато різних видів файлових систем. Кожен з них має різну структуру, логіку, властивості швидкості, гнучкості, безпеки, розміри і багато іншого. Деякі файлові системи були

розроблені для використання на конкретних пристроях. Наприклад, файлова система ISO 9660 розроблена спеціально для оптичних дисків.

Windows підтримує три типи файлових систем: NTFS, FAT32 і старішу файлову систему FAT (також її називають FAT16), яка рідко використовується[14].

2.2 Файлова система FAT32

FAT32 (File Allocation Table - таблиця розташування файлів) — це файлова система, що підтримує томи (логічні диски) обсягом до 8 ТБ і використовує для зберігання файлів менші фрагменти диска, ніж файлова система FAT16. Це збільшує вільний простір на диску. Файлова система FAT32 не підтримує диски, менші за 512 МБ.

Файлова система FAT32 була вперше реалізована в операційній системі Windows 95 OEM Service Release 2 (OSR2)[11].

2.2.1 Директорії

Дані директорій організовуються в 32-байтові записи. Тоді в одному секторі розміщується рівно 16 записів, і вони не перетинають межу сектора.

Є чотири типи 32-байтових записів:

- Нормальний запис з коротким ім'ям файлу.
- Запис з довгим іменем файлу.
- Не використовується. Перший байт - 0xE5
- Кінець директорії. Перший байт нульовий.

Записи типу "Не використовується" з'являються при видаленні файлів. Коли щось видаляється, перший байт стрічки запису просто замінюється на 0xE5, і потім місце використовується при додаванні нового запису. Записи, що не починаються з 0xE5, чи 0x00 і є вмістом каталогу. Структура такого запису зображена на рисунку 2.1

Поле	Зміщення	Розмір
Коротка назва файлу	0x00	8+3 байт
Байт атрибутів	0x0B	8 біт
Номер першого кластеру	0x14	32 біти
Розмір файлу	0x1C	32 біти

Рисунок 2.1 – структура запису

Розширення файлу завжди зберігається в файлах з 9 по 11. Якщо назва файлу коротша за вісім символів, то вільні байти заповнюються прогалинами (0x20). Найбільший розмір файлу 4Гб, через те, що розмір файлу зберігається в 32-розрядному полі. Структура байту атрибутів зображена на рисунку 2.2

Біт	Функція
0 (найменш значущий біт)	Тільки читання
1	Прихований
2	Системний
4	Субдиректорія (Вказівник показує на кластер з 32 байтовими записами)
5	Архівний файл
6, 7 (найбільш значущий біт)	Не використовуються, і завжи мають бути нульовими.

Рисунок 2.2 – структура байту атрибутів

2.2.2 Таблиця розміщення файлів

FAT32 названа так саме тому, що записи в цій таблиці 32-розрядні. Таблиця FAT - це просто великий масив беззнакових цілих, де кожен запис відповідає за свій кластер, і вказує номер наступного кластера. Якщо файл вміщується в один кластер, або кластер є останнім, то в таблиці FAT для цього кластера записане значення 0xFFFFFFFF[11].

2.3 Файлова система NTFS

NTFS (New Technology File System - «файлова система нової технології») - стандартна файлова система для сімейства операційніх систем Microsoft Windows NT.

NTFS замінила файлову систему FAT, яка використовувалася в MS-DOS і попередніх до Windows NT версіях Microsoft Windows. NTFS підтримує систему метаданих і використовує спеціалізовані структури даних для зберігання інформації про файли для поліпшення продуктивності, надійності і ефективності використання дискового простору. NTFS має вбудовані можливості розмежовувати доступ до даних для різних користувачів і груп користувачів, а також призначати квоти (обмеження на максимальний обсяг дискового простору, займаний тими чи іншими користувачами). NTFS використовує систему журналювання для підвищення надійності файлової системи. У Файловій системі NTFS відсутнє розділення на атрибути.

Розрізняють декілька версій NTFS: v1.2 використовується в Windows NT 3.51 і Windows NT 4.0, v3.0 поставляється з Windows 2000, v3.1 — з Windows XP і Windows Server 2003. Іноді останні версії позначають як v4.0, v5.0 і v5.1 відповідно до версій Windows NT, з якими вони поставляються.

2.3.1 Структура NTFS

На початку тому знаходиться завантажувальний запис тому (Volume Boot Record), в якому міститься код завантаження Windows, інформація про томи (зокрема, тип файлової системи), адреси системних файлів. Завантажувальний запис займає зазвичай 8 КБ (16 перших секторів). Структура зображена на рисунку 2.3.

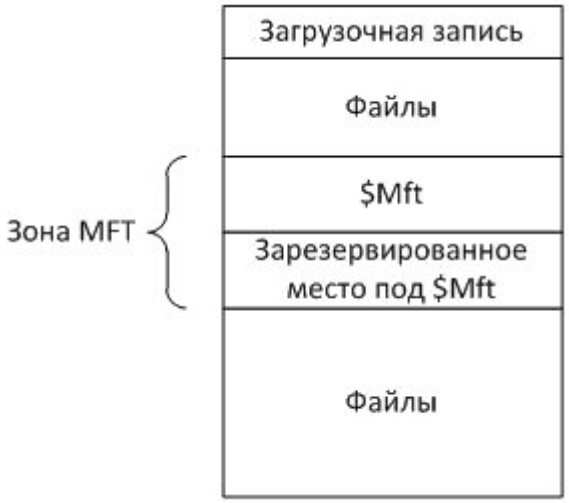


Рисунок 2.3 – структура NTFS

В певній галузі тому (адреса початку цієї області вказується в завантажувальному запису) розташована основна системна структура NTFS - головна таблиця файлів (Master File Table, MFT). У записах цієї таблиці міститься вся інформація про розташування файлів на томі, а невеликі файли зберігаються прямо в записах MFT.

Важливою особливістю NTFS є те, що вся інформація, як для користувача, так і системна, зберігається у вигляді файлів. Імена системних файлів починаються зі знаку "$". Наприклад, запис завантаження тому міститься в файлі $ Boot, а головна таблиця файлів - в файлі $ Mft. Така організація інформації дозволяє однаково працювати як з даними користувача, так і з системними даними на томі.

Оскільки MFT є найважливішою системною структурою, до якої при операціях з томом найбільш часто відбуваються звернення, вигідно зберігати файл $ Mft в безперервній області логічного диска, щоб уникнути його фрагментації (розміщення в різних областях диска), отже, підвищити швидкість роботи з ним . З цією метою при форматуванні тому виділяється безперервна область, звана зоною MFT (MFT Zone). У міру збільшення головної таблиці файлів, файл $ Mft розширюється, займаючи зарезервоване місце в зоні.

Решта місця на томі NTFS відводиться під файли - системні і призначені для користувача.

2.4 Порівняння файлових систем NTFS і FAT

Для сучасних версій Windows найкраще використовувати файлову систему NTFS. Вона має багато переваг над файловою системою FAT32[14], наприклад:

- можливість автоматично виправляти деякі помилки диска, які не може виправити FAT32.
- покращена підтримка жорстких дисків великої місткості.
- кращий захист, оскільки можна використовувати дозволи та шифрування для обмеження доступу до вказаних файлів певним користувачам.

2.5 Висновки до розділу 2

Файлова система — це базова структура, яку комп’ютер використовує для впорядкування даних на диску. Windows підтримує три типи файлових систем: NTFS, FAT32 і старішу файлову систему FAT (також її називають FAT16), яка рідко використовується.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

Під час написання даної дипломної роботу було розроблено програмний продукт, який дозволяє порівнювати файли (порівняння текстових файлів, визначення та підсвічування відмінностей, можливість підсвічування синтаксису), порівнювати каталоги файлів за їхнім вмістом та їх злиття.

3.1 Вибір засобів розробки

Програму створено у вигляді desktop application(настільного додатку). Для розробки додатку була обрана об’єктно-орієнтована мова програмування C#. Для створення інтерфейсу було використано мову розмітки XAML. Додаток розроблено на програмній платформі .NET Framework, використовуючи технологію WPF в середовищі MS Visual Studio 2015 Community.

Цей вибір технологій дозволяє забезпечити можливість розширення функціоналу в майбутньому, розробити прозору структуру програми, забезпечити її швидку роботу без особливих вимог до апаратного забезпечення.

3.1.1 .NET Framework

Microsoft .NET — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків[13].

Багато в чому є продовженням ідей та принципів, покладених в технологію Java.

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU [14].

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор.

3.1.2 Технологія WPF

Windows Presentation Foundation (WPF) — графічна підсистема в складі .NET Framework що має пряме відношення до XAML. WPF разом з .NET Framework вбудована в Windows Vista, а також доступна для установки в Windows XP Service Pack 2 і Windows Server 2003[14].

Це перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро, яке повинне замінити GDI і GDI+, використовувані на попередній Windows-платформі. WPF є високорівневим об'єктно-орієнтованим функціональним

шаром (англ. framework), що дозволяє створювати двовимірні та тривимірні інтерфейси.

3.1.3 Мова розмітки XAML

XAML (скорочення від Extensible Application Markup Language) є мовою розмітки, яку використовують для створення екземплярів об'єктів .NET. Хоча мова XAML — це технологія, що може бути застосовна до багатьох різних предметних областей, її головне призначення — конструювання інтерфейсів користувачів WPF[14]. Інакше кажучи, документи XAML визначають розташування панелей, кнопок та інших елементів керування, що становлять вікна в застосунку WPF. Малоймовірно, що вам доведеться писати код XAML вручну. Замість цього ви використовуєте інструмент, що генерує необхідний код XAML.

3.2 Порівняння та злиття папок

3.2.1 Розробка алгоритму

Папки в ОС Windows мають ієрархічну структуру. Це означає, що порівнянні папок потрібно враховувати усі дочірні папки і виводити звіт по порівнянню для усієї ієрархії. В звіт вважаю за потрібне вносити інформацію про конфліктуючі файли, тобто файли, які мають однакове ім’я. Для таких файлів проводиться порівняння. Спочатку порівнюється розмір, а при необхідності (якщо розмір однаковий) проводиться порівняння хешу файлів.

Після виконання порівнянь формується звіт, який представлений у вигляді таблиці даних.

3.2.1.1 Алгоритм порівняння файлів

При виборі між порівнянням по хешу чи побайтовим порівнянням надано перевагу порівнянню по хешу(за допомогою хеш фукції).

В першу чергу хеш-функції є математичною функцією, яка перетворює обсяг даних великого розміру в набагато менший набір даних. Цей набір є відображенням актуальних даних, тому він ідеально підходить для порівняння даних. Шанси генерації двох однакових хешів для різних файлів практично неможливо. Крихітна зміна в файлі призводить до досить великих і непередбачуваних змін в сформованому хеш.

Таким чином, хеш файлу є хеш-представленням файлу. Хеш на вигляд являє собою довгий числовий або буквено-цифровий вираз, такі як, наприклад:

- b7404b4dd5e4d1b67869226dcbc2da09
- 29-B4-1C-B3-54-F3-14-19-16-EE-0D-6A-F5-73-56-9F-DA-3F-D5-47

Проведено аналіз швидкодії для двох методів: побайтове порівняння та порівняння хешу. Результат на рисунку 3.1.

Як видно з результатів, немає практично ніякої різниці між методами порівняння для невеликих файлів, але коли було порівняно два великих файли, близько 700MB кожен, можна чітко бачити різницю між методами. Побайтовий методом виконувався близько 27 секунд, щоб завершити порівняння великих файлів, в той час як метод порівняння хеш файлу займає близько 18 секунд.

Це порівняння ясно показує, що метод порівняння по хешу працює швидше, і з урахуванням практично неможливого створення однакових хешів для різних файлів можна сказати, що цей метод є кращим.

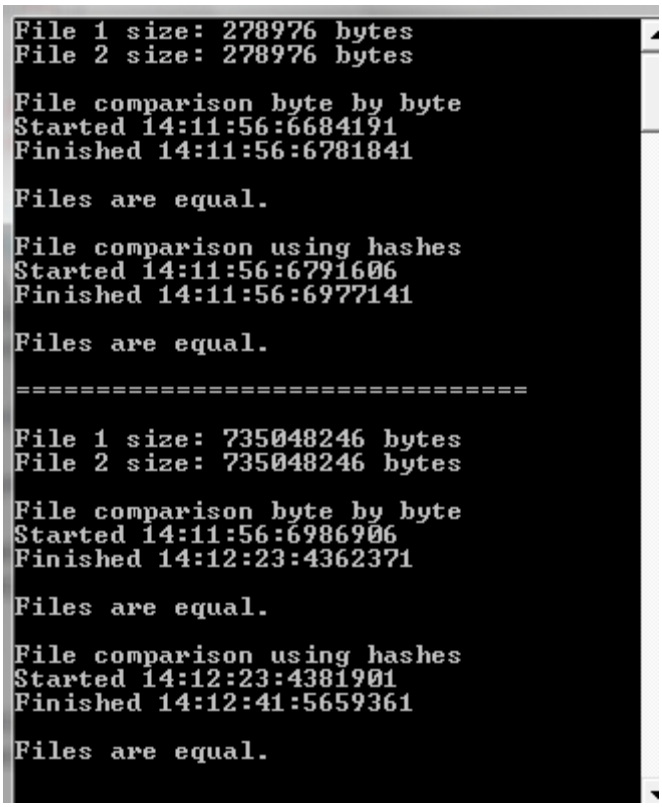


Рисунок 3.1 – результат порівняння

Програмна реалізація виглядає наступним чином:

```
public static bool CompareFileHashes(string fileName1, string fileName2)
{
   // Create an instance of System.Security.Cryptography.HashAlgorithm
   HashAlgorithm hash = HashAlgorithm.Create();

   // Declare byte arrays to store our file hashes
   byte[] fileHash1;
   byte[] fileHash2;
```

```
    // Open a System.IO.FileStream for each file.
    // Note: With the 'using' keyword the streams
    // are closed automatically.
    using (FileStream fileStream1 = new FileStream(fileName1, FileMode.Open),
                fileStream2 = new FileStream(fileName2, FileMode.Open))
    {
       // Compute file hashes
       fileHash1 = hash.ComputeHash(fileStream1);
       fileHash2 = hash.ComputeHash(fileStream2);
    }

    return BitConverter.ToString(fileHash1) == BitConverter.ToString(fileHash2);
}
```

3.2.1.2 Алгоритм вилучення даних з папок

При порівнянні папок потрібно враховувати всі дочірні папки та їх вміст. Щоб здійснити таку вибірку потрібно використати рекурсивну функцію. В основі цієї функції буде прохід по вмісту папки, при потраплянні на дочірню папку відбудеться рекурсивний виклик цієї функції вже для дочірньої папки, що дасть змогу пройтися по всій ієрархії папок.

Це рішення є загальним і універсальним незалежно від типу чи розміру ієрархічної організації папки.

Рисунок 3.2 – ієрархія папок

По рисунку 3.2, де зображена тестова ієрархія, можна описати принцип роботи алгоритму. При зчитуванні даних з «TestFolder» потрапляємо в папку «1» і починаємо зчитування даних з папки «1», таким самим чином потрапляємо в папку «1.1» і потім в «1.2», після цього продовжуємо роботу з «TestFolder» і зчитуємо дані спочатку з папки «2», а потім з папки «3».

Програмна реалізація виглядає наступним чином:

```
public FolderItem(DirectoryInfo folder, string dirPattern = "*", Container.FolderPos
ComparePos = Container.FolderPos.First)
{
  foreach (var file in Folder.GetFiles(DirPattern))
     SubFiles.Add(new FileItem(file));
   SubFolders = new List<FolderItem>();
   foreach (var dir in Folder.GetDirectories())
  //recursion
     SubFolders.Add(new FolderItem(dir, dirPattern: dirPattern));
}
```

3.2.1.3 Алгоритм формування звіту порівняння

Порівнюючи папки необхідно звертати увагу лише на файли з однаковим іменем, саме такі файли потребують контролю користувача. Файли з однаковим іменем порівнюються спочатку по довжині і потім, якщо необхідно, по хешу. Якщо файли однакові по вмісту, то вони також не потребують особливого контролю користувача, але будуть включені в звіт порівняння для більшої інформативності. Найбільшої уваги потребують файли, що мають однакову імя і різний вміст, вони потребують контролю користувача, який повинен явно вказати який з них зберегти(перший, другий чи обидва).

3.2.1.4 Порівняння за шаблоном

При порівнянні папок є можливість ввести шаблон для порівняння. Шаблон має вигляд строки, яка задає критерії до імен порівнюваних файлів. Для створення такого шаблону достатньо використовувати всього два символи: «?» і «*». Символ «?» означає один будь-який символ, а символ «*» визначає будь-яку послідовність символів. Приведемо приклад використання цих символів:

- «*Text?.txt» - цей вираз означає, що будуть обрані лише ті файли в яких після будь-якої послідовності символів буде слово «Text» потім один будь-який символ, а завершуватись має на «txt». Для цього виразу підійдуть наступні імена файлів: «SomeText1.txt», «1TextR.txt»; а наступні не відповідатимуть до цього виразу: «SomeText123.txt», «OtherFile.exe», «123Text.txt».
- «file?.exe» - цей вираз означає, що будуть обрані лише ті файли в імені яких після слова «file» буде один будь-який символ, а завершуватись має на «exe». Для цього виразу підійдуть наступні імена файлів: «file1.exe»,

«fileP.exe»; а наступні не відповідатимуть до цього виразу: «filePort.exe», «OtherFile.exe», «file1.txt».

3.2.2 Опис функціоналу

Програма надає можливість обрати 2 будь-які папки для порівняння за допомогою діалогового вікна, що зображено на рисунку 3.3. Потрібно обрати дві папки для порівняння. Після вибору папки з’являється базова інформація про папку (ім’я, час останньої зміни, кількість папок та файлів на першому рівні ієрархії), що показано на рисунку 3.4.

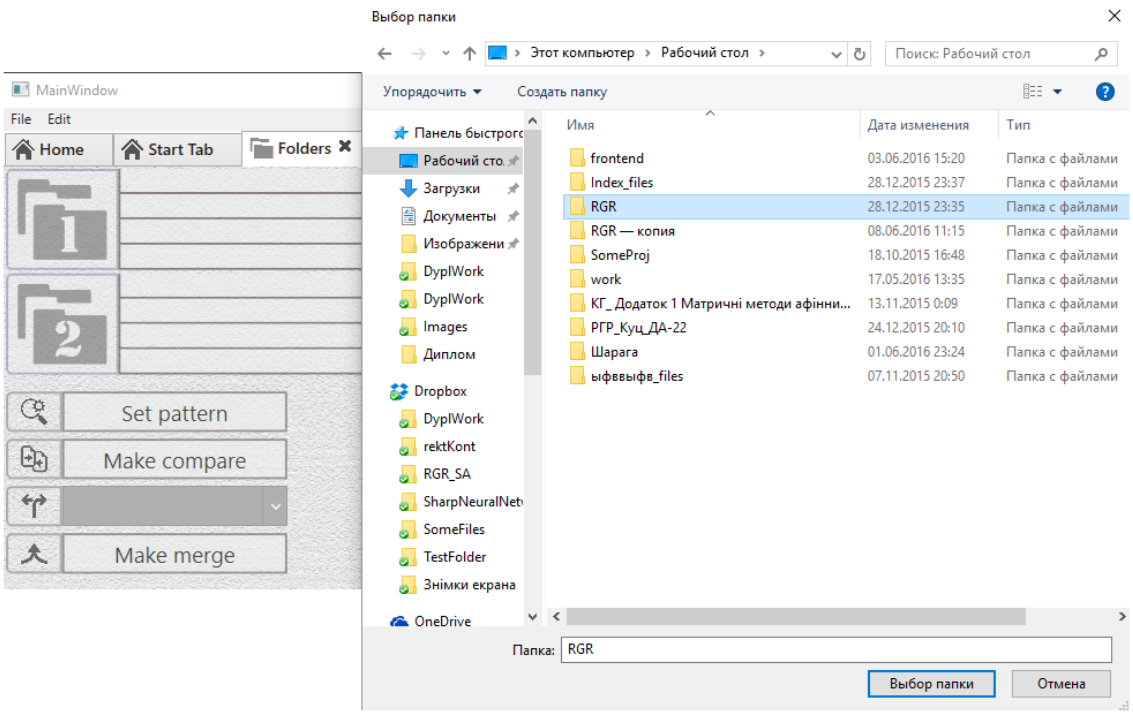


Рисунок 3.3 – діалог для вибору папки

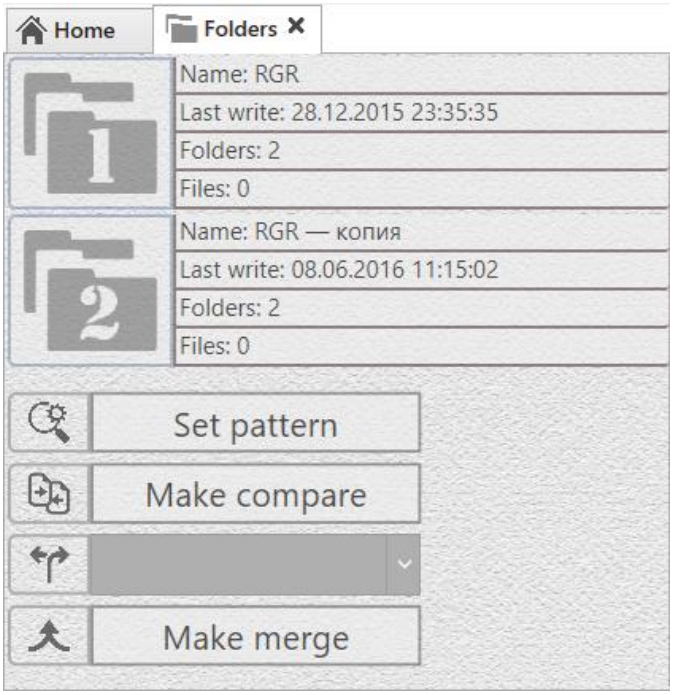
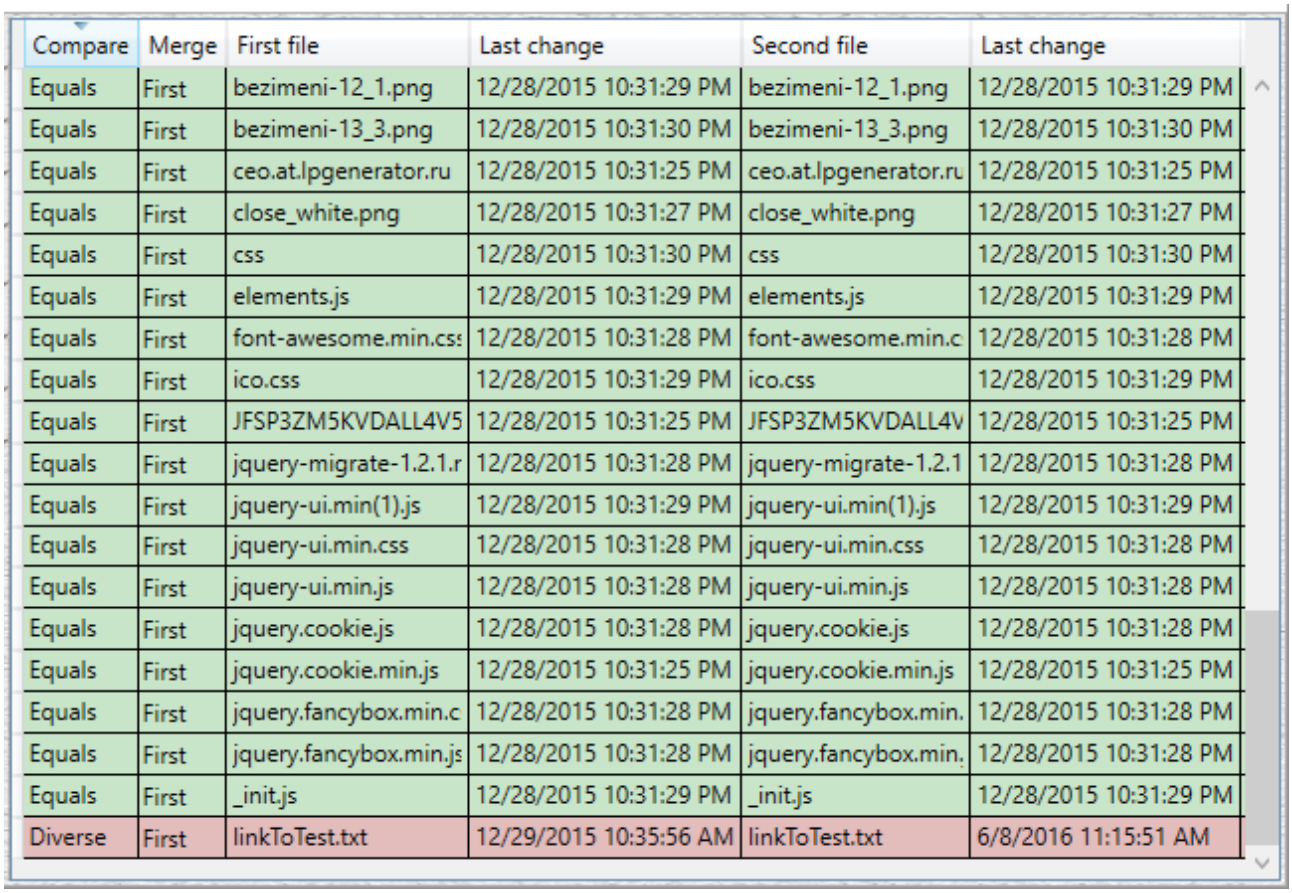


Рисунок 3.4 – інтерфейс програми про порівнянні папок

Програма дозволяє здійснити порівняння папок з виведенням звіту про порівняння до таблиці. Як вже зазначалося, порівняння відбувається по всій ієрархії папки, що дозволяє отримати повну інформацію про відмінності в порівнюваних файлах. До звіту про порівняння потравляють лише файли з однаковим ім’ям та місцем в ієрархії цих папок, адже само вони є конфліктними при злитті папок. Оформлення звіту програмою показано на рисунку 3.5.



Compare	Merge	First file	Last change	Second file	Last change
Equals	First	bezimeni-12_1.png	12/28/2015 10:31:29 PM	bezimeni-12_1.png	12/28/2015 10:31:29 PM
Equals	First	bezimeni-13_3.png	12/28/2015 10:31:30 PM	bezimeni-13_3.png	12/28/2015 10:31:30 PM
Equals	First	ceo.at.lpgenerator.ru	12/28/2015 10:31:25 PM	ceo.at.lpgenerator.ru	12/28/2015 10:31:25 PM
Equals	First	close_white.png	12/28/2015 10:31:27 PM	close_white.png	12/28/2015 10:31:27 PM
Equals	First	css	12/28/2015 10:31:30 PM	css	12/28/2015 10:31:30 PM
Equals	First	elements.js	12/28/2015 10:31:29 PM	elements.js	12/28/2015 10:31:29 PM
Equals	First	font-awesome.min.css	12/28/2015 10:31:28 PM	font-awesome.min.c	12/28/2015 10:31:28 PM
Equals	First	ico.css	12/28/2015 10:31:29 PM	ico.css	12/28/2015 10:31:29 PM
Equals	First	JFSP3ZM5KVDALL4V5	12/28/2015 10:31:25 PM	JFSP3ZM5KVDALL4V	12/28/2015 10:31:25 PM
Equals	First	jquery-migrate-1.2.1.r	12/28/2015 10:31:28 PM	jquery-migrate-1.2.1	12/28/2015 10:31:28 PM
Equals	First	jquery-ui.min(1).js	12/28/2015 10:31:29 PM	jquery-ui.min(1).js	12/28/2015 10:31:29 PM
Equals	First	jquery-ui.min.css	12/28/2015 10:31:28 PM	jquery-ui.min.css	12/28/2015 10:31:28 PM
Equals	First	jquery-ui.min.js	12/28/2015 10:31:28 PM	jquery-ui.min.js	12/28/2015 10:31:28 PM
Equals	First	jquery.cookie.js	12/28/2015 10:31:28 PM	jquery.cookie.js	12/28/2015 10:31:28 PM
Equals	First	jquery.cookie.min.js	12/28/2015 10:31:25 PM	jquery.cookie.min.js	12/28/2015 10:31:25 PM
Equals	First	jquery.fancybox.min.c	12/28/2015 10:31:28 PM	jquery.fancybox.min.	12/28/2015 10:31:28 PM
Equals	First	jquery.fancybox.min.js	12/28/2015 10:31:28 PM	jquery.fancybox.min.	12/28/2015 10:31:28 PM
Equals	First	_init.js	12/28/2015 10:31:29 PM	_init.js	12/28/2015 10:31:29 PM
Diverse	First	linkToTest.txt	12/29/2015 10:35:56 AM	linkToTest.txt	6/8/2016 11:15:51 AM

Рисунок 3.5 – звіт про порівняння

Як видно з рисунку 3.5, файли підсвічуються зеленим якщо результатом їх порівняння є «Equals» (це означає, що файли абсолютно однакові по вмісту), цей результат не є конфліктним, адже неважливо який з файлів буде збережений під час злиття папок так як ці файли абсолютно ідентичні. Також, як можна помітити, є файли підсвічені червоним кольором адже результат їхнього порівняння є «Diverse» (це означає, що файли не є однаковими по вмісту), цей результат є конфліктним, тому користувач в колонці «Merge» має обрати який файл він хоче зберегти при злитті, доступними є опції «First»(перший файл), «Second»(другий файл) і «Both»(обидва файли).

Також у користувача є можливість обрати в яку з обраних папок він бажає провести злиття. Серед доступних опцій є також можливість задання шаблону для порівняння файлів, що дозволить порівнювати та проводити злиття конкретних файлів, що відповідають шаблону.

3.2.3 Реалізація

Робота з папками винесена в окремий модуль програми, що забезпечує можливість розширення і цього модуля і програми в майбутньому. Структура програмної частини роботи з папками показана на рисунку 3.6, і створена за допомогою стандартних засобів середовища розробки MS Visual Studio.

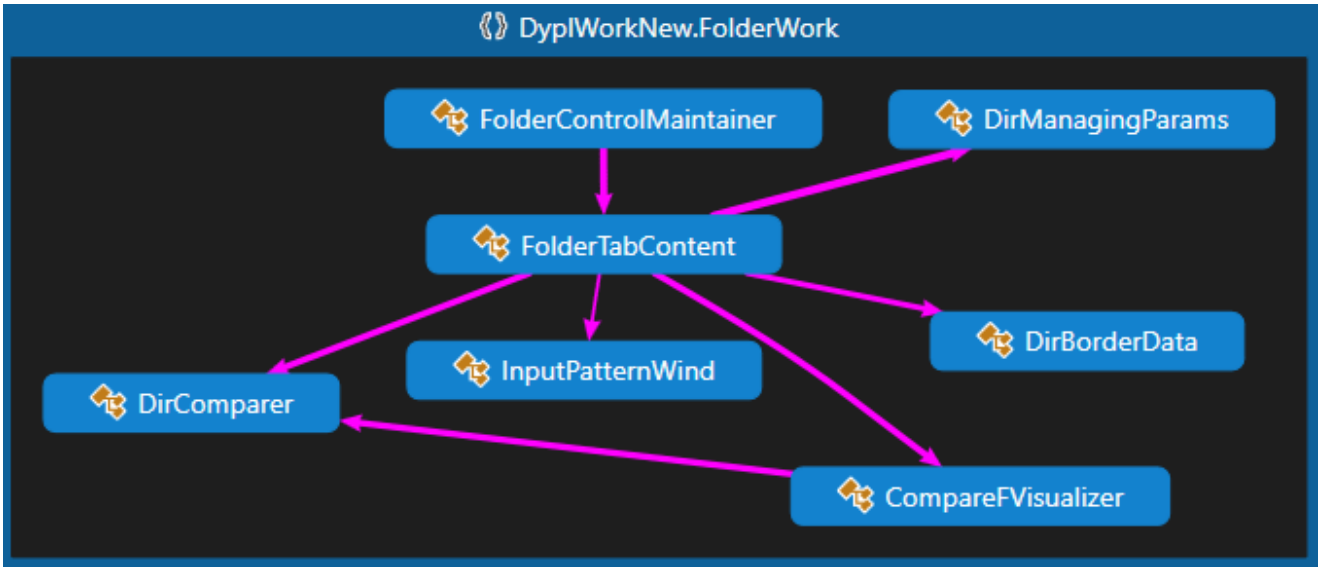


Рисунок 3.6 – структура програми

Клас «FolderControlMaintainer» є управляючим класом для роботи з папками, він відповідає за створення, відновлення сеансів та видалення вкладок для роботи з папками. Основним його методом є метод «CreateTabContent()», що відповідає за створення екземпляру класу «FolderTabContent». Структура класу зображена на рисунку 3.7.

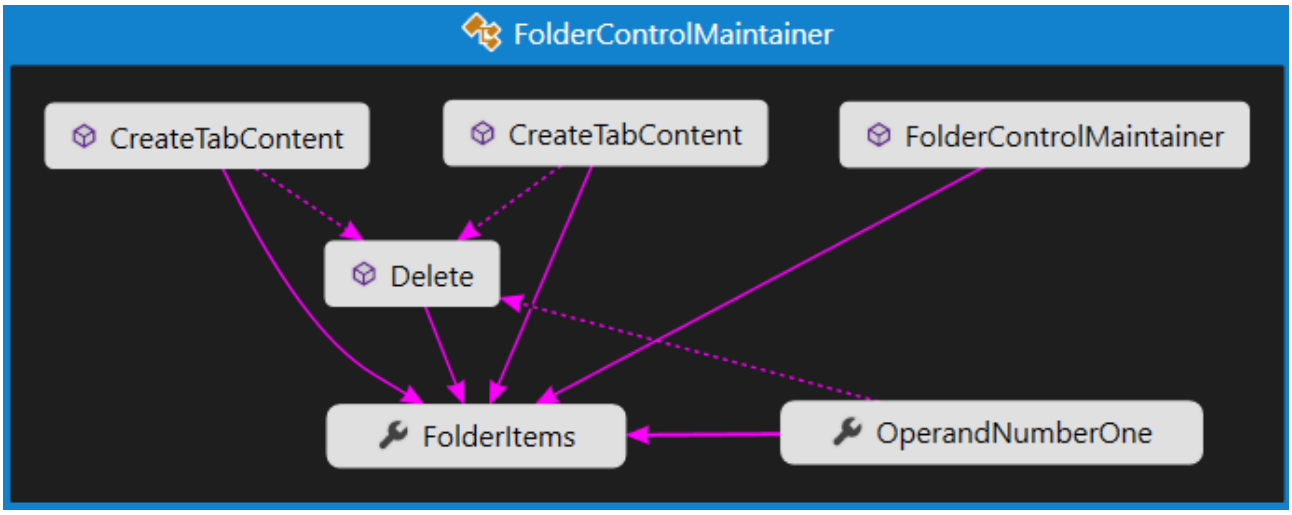


Рисунок 3.7 – структура класу «FolderControlMaintainer»

Дуже важливим класом також є клас «FolderTabContent». Цей клас відповідає за побудову інтерфейсу користувача і обробку запитів, подій, вводу даних. Передбачає механізм відновлення сеансу, відновлюючи інтерфейс згідно до збереженого сеансу. Структура класу доволі об’ємна, тому зображувати її на малюнку немає необхідності.

Важливим класом, що відповідає за порівняння змісту папок є «DirComparer». Цей клас задіяний в рекурсивному алгоритмі для порівняння папок. Працює цей алгоритм наступним чином: при порівнянні папок, якщо зустрічаються папки з однаковим іменем, рекурсивно створюється новий екземпляр класу «DirComparer» який порівнює вже ці дочірні папки, і так поки всі папки з однаковим іменем не будуть порівняні. Структура класу «DirComparer» наведена на рисунку 3.8.

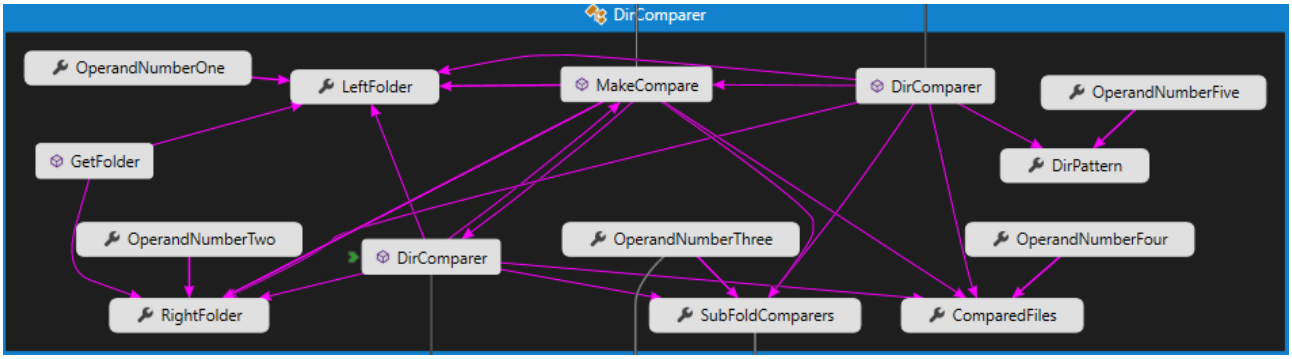


Рисунок 3.8 – структура класу «DirComparer»

Звіт про порівняння формується на основі екзеплярів класу «ReportItem». Ці екземпляри містять в собі всю необхідну інформацію про результат порівняння для користувача. Звіт складається з колекції екземплярів класу «ReportItems», яка за допомогою механізму DataBinding (прив’язки даних) пов’язується з таблицею. Цей механізм дозволяє двухсторонній зв’язок між таблицею і колекцією, це означає, що при зміні даних в таблиці буде оновлено колекцію даних. Це доволі зручний механізм, який дозволив спростити програмну реалізацію взаємодії з користувачем. Структура класу «ReportItem» зображена на рисунку 3.9.

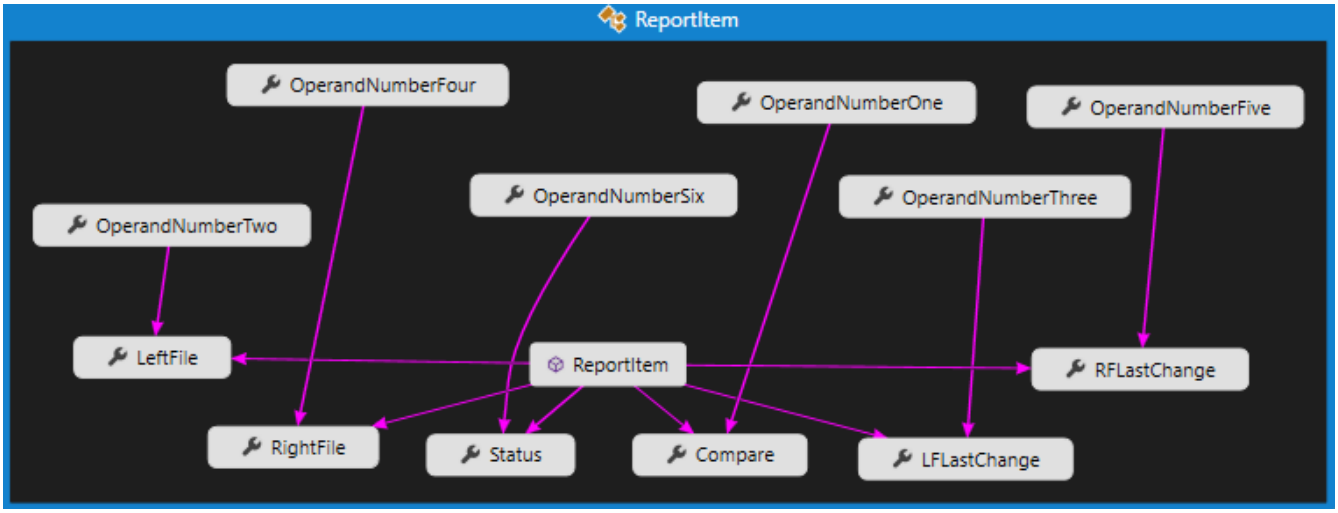


Рисунок 3.9 – структура класу «ReportItem»

Також варто приділити увагу допоміжним класам, що об’єднані окремий простір імен «DyplWorkNew.FolderWork.FolderClasses». Загальна структура зображена на рисунку 3.10. Ці класи призначені для роботи з папками та файлами, обробки інформації про файли і папки.

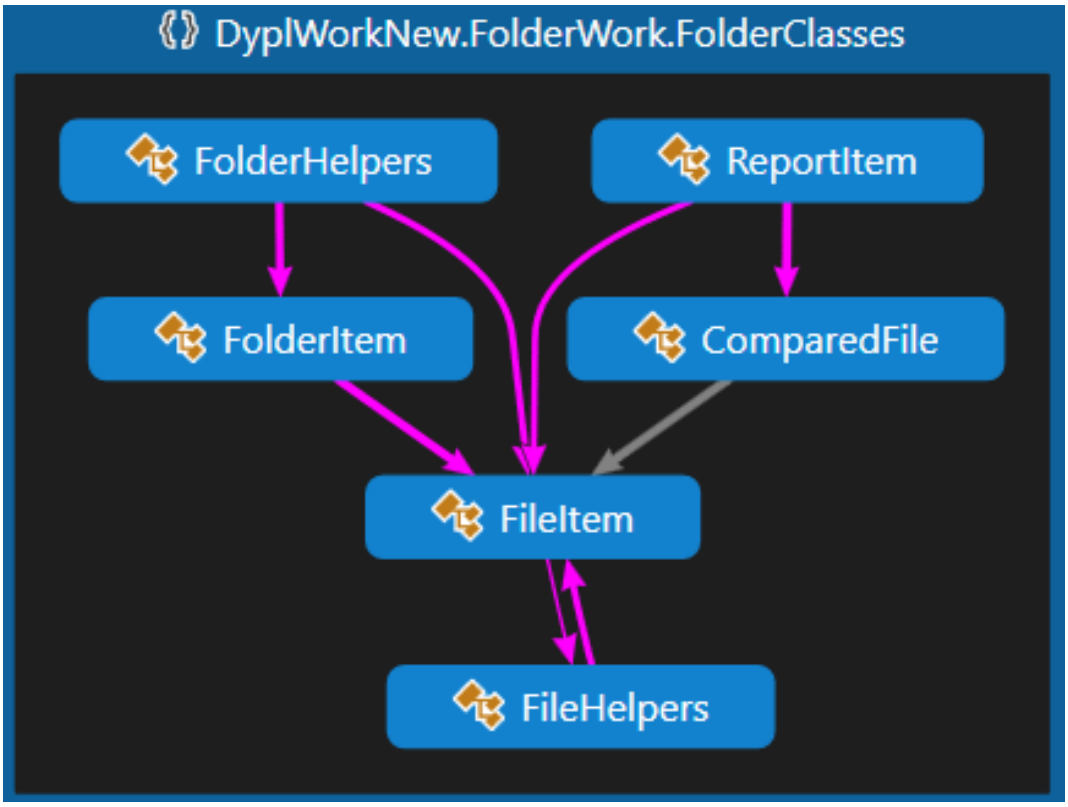


Рисунок 3.10 – структура роботи з папками

3.3 Порівняння текстових файлів

Часто виникає задача порівняти 2 файли по їх вмісту і відслідкувати відмінності між ними. В програмі реалізований модуль для порівняння файлів з можливістю підсвічування синтаксису. Цей модуль зручний для майбутнього розширення, дозволяє просто створювати нові правила для підсвічування синтаксису.

3.3.1 Розробка алгоритму

3.3.1.1 Алгоритм порівняння тексту

Для порівняння та виділення різниці між двома файлами було обрано наступний алгоритм. Припустимо, у нас є текст (початковий текст): «АА ЇЇ КВ ЗЗ УУ ДД» і його модифікована копія (Змінений текст): «АА ЇЇ ХХ ЗЗ ДХ УУ» і нам потрібно виділити відмінності. Алгоритм вирішення цього завдання наступний:

- кожному рядку дамо унікальне ім'я (індекс), щоб простіше було працювати складемо таблицю (таблиця 3.1):

Таблиця 3.1 – таблиця індексів строк

АА	1
ЇЇ	2
КВ	3
ЗЗ	4
УУ	5
ДД	6
ХХ	7
ДХ	8

- тепер переведемо рядки відповідно до цієї таблиці:
  1) «АА ЇЇ КВ ЗЗ УУ ДД» => 1 2 3 4 5 6
  2) «АА ЇЇ ХХ ЗЗ ДХ УУ» => 1 2 7 3 8 5
- далі використовується алгоритм вибору найбільшої спільної послідовності (longest common subsequence, LCS) отримаємо наступну таблицю(таблиця 3.2):

Таблиця 3.2 – строки відображені по індексам

Перша строка	1 2 3 4 5 6
Друга строка	1 2 7 3 8 5
Результат	1 2 3 5

Як видно з таблиці 3.2, «1 2 3 5» - найдовша загальна частина (ця частина однакова з урахуванням, що десь щось вставили, десь видалили)

- далі по черзі переглядаємо обидві рядки, і якщо, скажімо, в одному рядку символ з'явився, а в іншій немає, значить він був вставлений і так далі(Таблиця 3.3).

Таблиця 3.3 – результат порівняння

Строка №1	Строка №2	Спільна частина	Результат порівняння
1	1	1	Спільний
2	2	2	Спільний
3	7	3	7 помічається как вставлена строка і "зсовуємо" 2-гу строку нагору
3	3	3	Спільний
4	8	5	Не спільний, так як відсутній в загальній строці, помітимо як змінену
5	5	5	Спільний
6			Видалена, так як є в першій і відсутня в другій строці

3.3.1.1 Відображення відмінностей

Для відображення відмінностей використовуються регулярні вирази та XML файли з правилами підсвічування та виділення певного тексту згідно до критеріїв.

Для побудови XML файлів з правилами підсвічування відповідно до синтаксису використовується XSD Schema. При використанні XSD XML парсер може перевірити не тільки правильність синтаксису XML документа, але також його структуру, модель змісту і типи даних. Такий підхід дозволяє об'єктно-орієнтованим мовам програмування легко створювати об'єкти в пам'яті, що, безсумнівно, зручніше ніж розбирати XML як звичайний текстовий файл. Крім того, XSD легко розширюється, і дозволяє підключати вже готові словники для опису типових задач. Варто також згадати про те, що в XSD є вбудовані засоби документування, що дозволяє створювати самодостатні документи, які не потребують додаткового опису.

3.3.2 Опис функціоналу

Програма надає можливість обрати будь-які два файли, використовуючи фільтри по типу файла. Після вибору файлів автоматично виконається порівняння їх вмісту. Після чого результат порівняння відобразиться в інтерфейсі користувача.

Вибір файлів виконується за допомогою виклику файлового діалогу (рисунок 3.11). Після вибору файла, його вміст завантажиться до відповідного текстового поля, де вже користувач зможе наглядно аналізувати відмінності між файлами (рисунок 3.12).

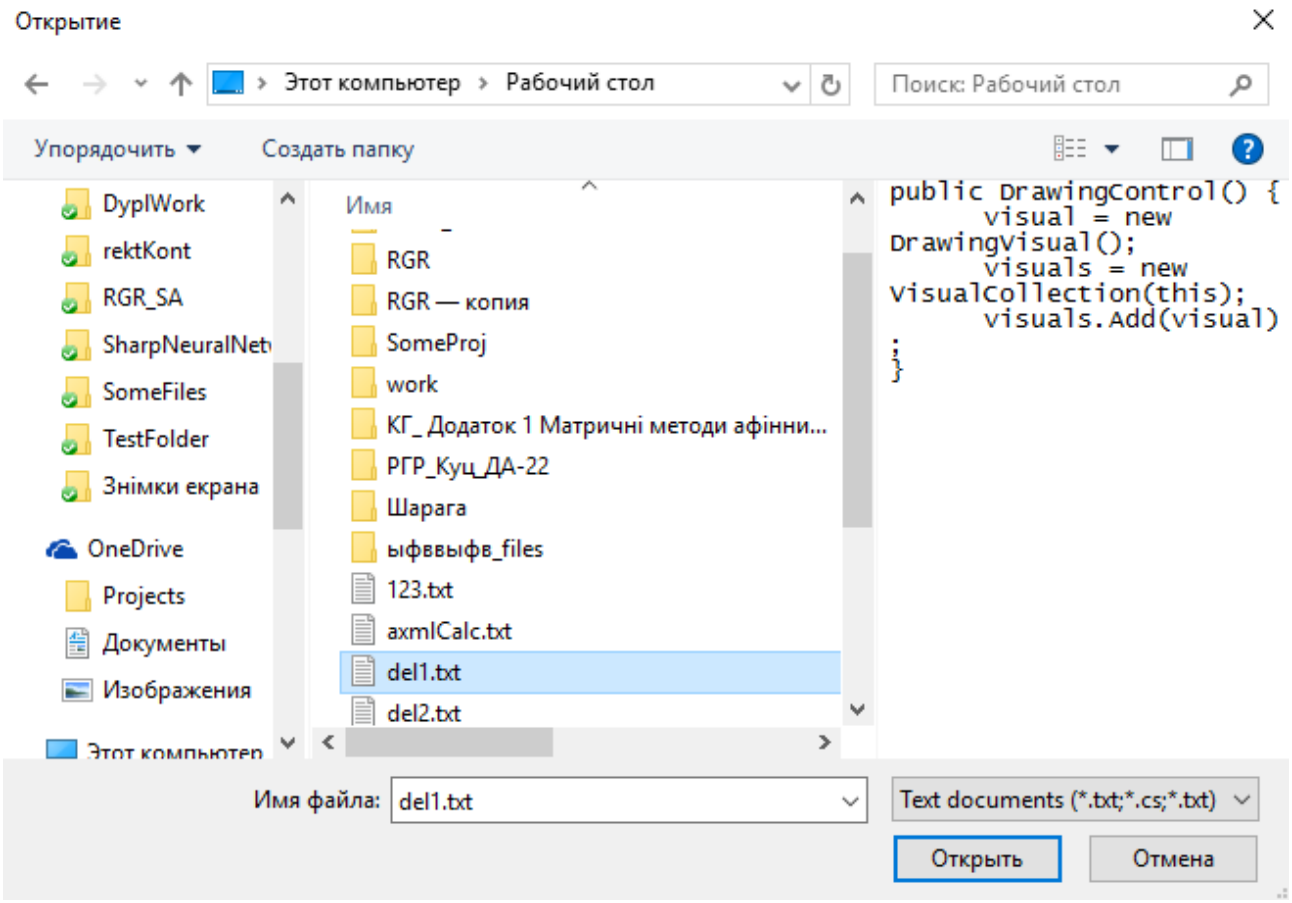


Рисунок 3.11 – файловий діалог

Для прикладу порівняємо два текстових файли. Текст першого з них буде наступним: «public DrawingControl() {

visual = new DrawingVisual();

visuals = new VisualCollection(this);

visuals.Add(visual);

}».

Текст другого файлу буде не дуже суттєво відрізнятися від першого:

«public DrawingControl(int NewParam) {

visual = new DrawingVisual();

NewParam = 2;

visuals = new VisualCollection(this);

NewParam += 3;

}».

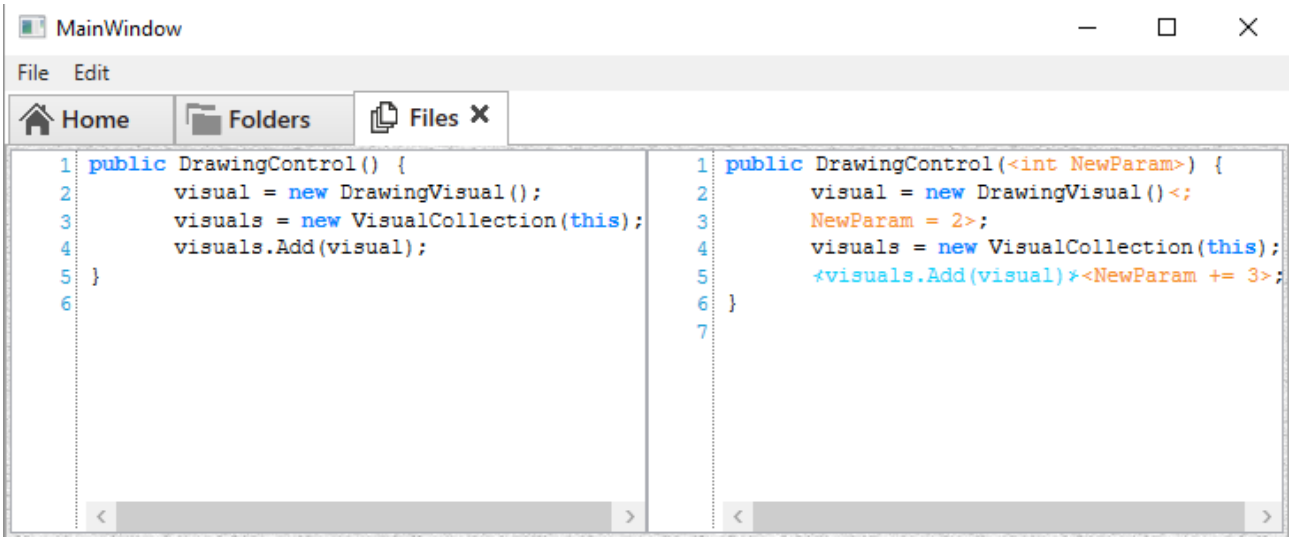


Рисунок 3.12 – відображення змін

Як видно з рисунку 3.12 всі відмінності коректно відобразились. Блакитним кольором підсвічується те, що було видалено, а помаранчевим – те, що додалось в порівнянні з першим файлом.

3.3.3 Реалізація

Всі класи та файли, пов’язані з порівнянням тексту файлів та визначенням відмінностей, винесені в простір імен «DyplWorkNew.FilesWork».

Клас «FileControlMaintainer» є управляючим файлом для роботи з файлами. Він відповідає за створення інтерфейсу користувача і за відновлення сеансів. Його структура зображена на рисунку 3.13.

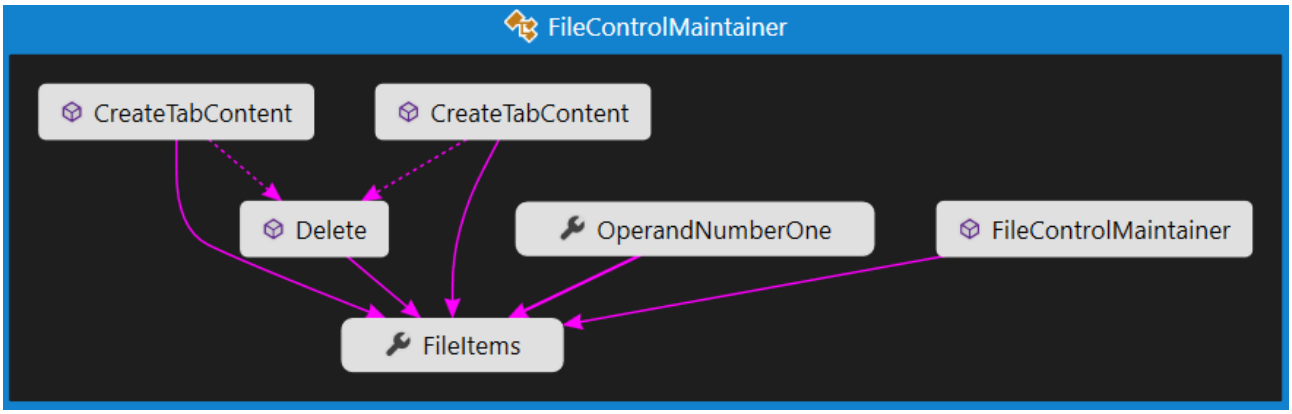


Рисунок 3.13 – структура «FileControlMaintainer»

Важливим класом також є клас «FileTabContent». Цей клас відповідає за побудову інтерфейсу користувача і обробку запитів, подій, вводу даних. Передбачає механізм відновлення сеансу, відновлюючи інтерфейс згідно до збереженого сеансу. Структура класу доволі об’ємна, тому зображувати її на малюнку немає необхідності.

Важливою частиною порівняння текстових файлів є підсвічування змін і синтаксису. За це відповідає клас «SyntaxHighlightBox», який наслідується від стандартного класу «TextBox» і описує необхідні методи для організації підсвічувань. Загальна структура цього класу і допоміжних класів зображена на рисунку 3.14.

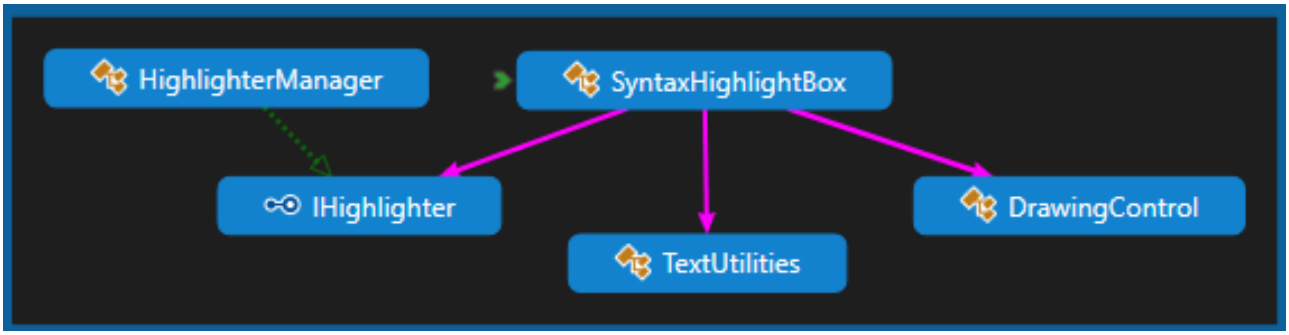


Рисунок 3.14 – структура «SynyaxHighlightingBox»

3.4 Відновлення сеансів

Також програма дозволяє відновлювати сеанси, тобто можна відновити всі дані, що були в останньому сеансі. Ця функція реалізована за допомогою DataContract сериалізації. Для використання цього механізму потрібно усі необхідні властивості класів винести в додаткові властивості, з для яких вказати необхідний атрибут «DataMember», а для самого класу потрібно вказати атрибут «DataContract».

DataContract сериалізація це не єдиний спосіб зберегти інформацію програми, ще можна використати XML сериалізацію. Проте XML сериалізація має більше властивостей і можливостей, які в даній програмі не потрібні, але ці додаткові можливості сповільнюють процес сериалізації, тому було обрано DataContract сериалізацію. В коді це виглядає наступним чином:

3.5 Висновки до розділу 3

В цьому розділі проведено вибір та аналіз засобів розробки, описано ключові особливості засобів розробки. Було розроблено алгоритм для порівняння папок та файлів, проведено аналіз та обрано найбільш оптимальні алгоритми. Також було реалізовано алгоритм для порівняння вмісту текстових файлів. В розділі проведено опис функціоналу для порівняння і об’єднання папок та для порівняння текстових файлів. Також було реалізовано механізм збереження сеансів.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Порівняння та об’єднання папок без фільтру

Для тестування створено дві тестові папки «Тест1» та «Тест2», вони мають одну папку з однаковим іменем «HTML files» (рисунок 4.1)

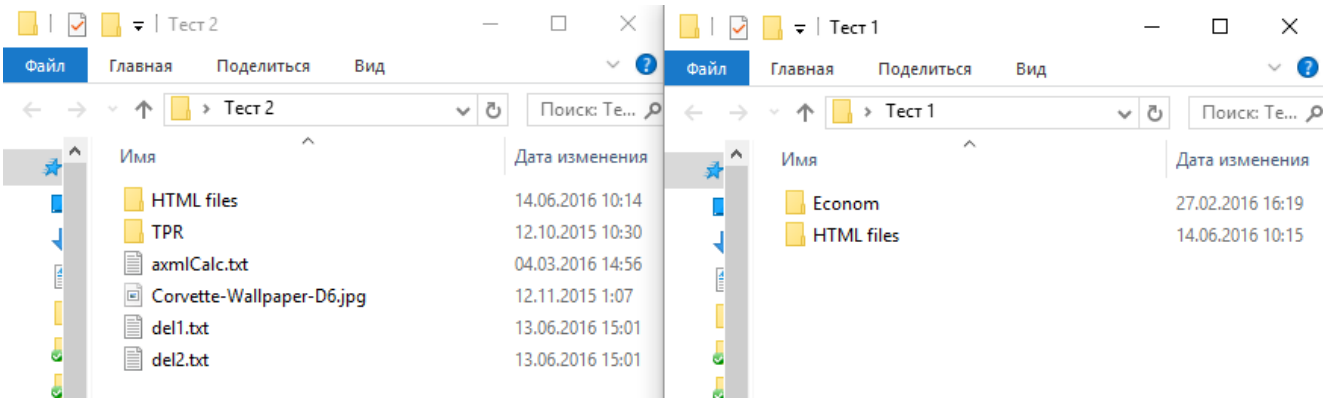


Рисунок 4.1 – тестові папки «Тест1» і «Тест2»

Папка «HTML files» містить в собі 3 файли (рисунок 4.2), де 2 файли JS трохи відрізняються (змістом, але розмір однаковий) в папках «Тест1» та «Тест2», отже програма повинна розпізнати їх як конфліктні і підсвітити червоним кольором. Відмінності між файлами зображені на рисунку 4.3 та рисунку 4.4.

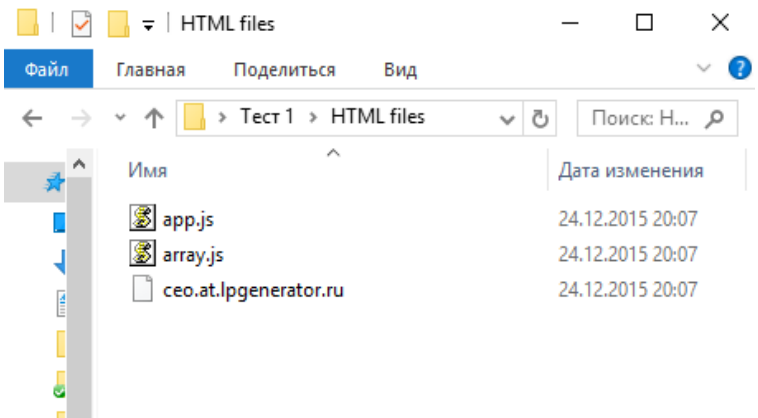


Рисунок 4.2 – вміст папки «Html files»

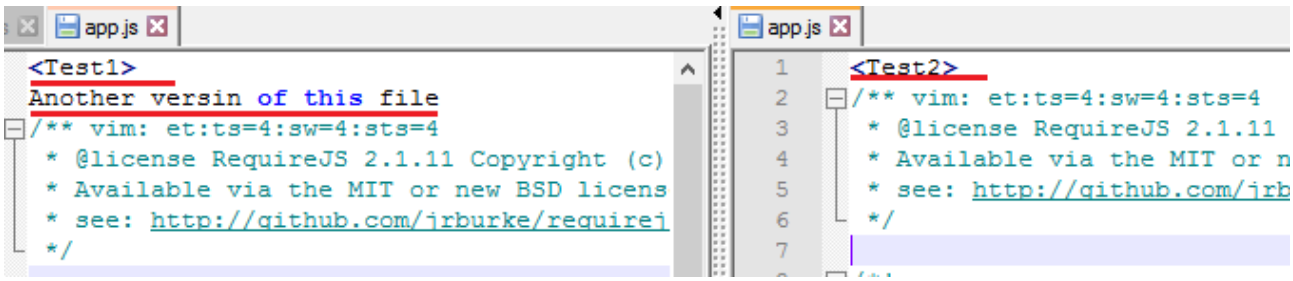


Рисунок 4.3 : Різниця між файлами «app.js»

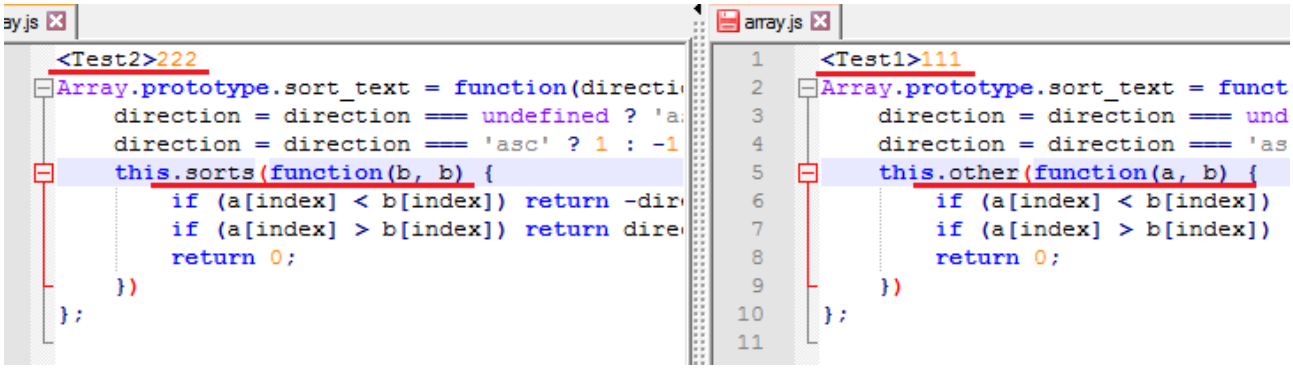


Рисунок 4.4 : Різниця між файлами «array.js»

Спочатку оберемо в програмі ці 2 папки:

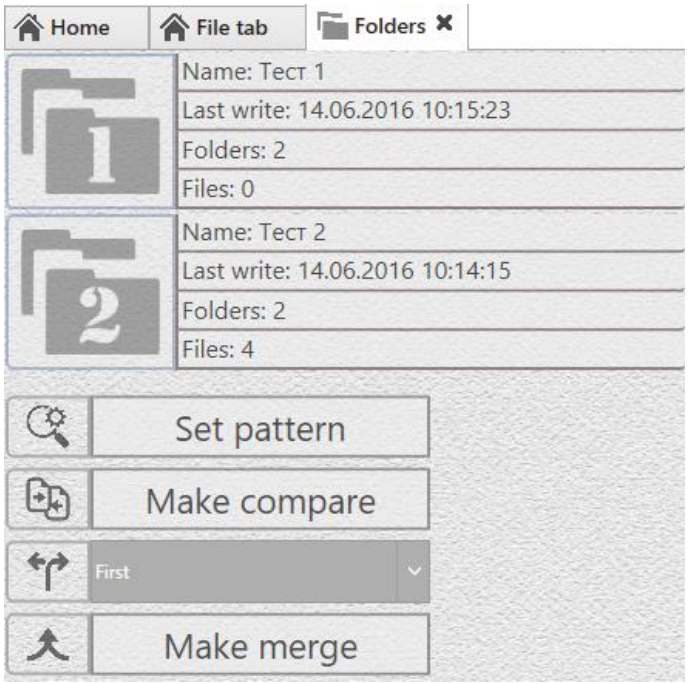


Рисунок 4.5 – вкладка порівняння файлів

Після чого натиснемо кнопку «Make compare», виведеться звіт про порівняння у вигляді таблиці, як вже зазначалося, маємо 3 файли з однаковим іменем, отже в таблиці повинно бути 3 записи, причому 2 з них конфліктні.

Compare result	Merge	First file	Last change	Second file	Last change
Diverse	First	app.js	12/24/2015 8:07:13 PM	app.js	6/14/2016 10:17:43 AM
Diverse	First	array.js	12/24/2015 8:07:15 PM	array.js	6/14/2016 10:17:29 AM
Equals	First	ceo.at.lpgenerator.ru	12/24/2015 8:07:11 PM	ceo.at.lpgenerator.ru	12/24/2015 8:07:11 PM

Рисунок 4.6 – звіт про порівняння

Як видно з рисунку 4.6, порівняння відбулося правильно.

Тепер перевіримо коректність об’єднання папок. Об’єднувати будемо в папку «Тест1», причому файл «app.js» оберемо з папки «Тест1», а файл «array.js» оберемо з папки «Тест2», відмінності яких вже були продемонстровані вище. Результат об’єднання цих файлів на рисунку 4.7.

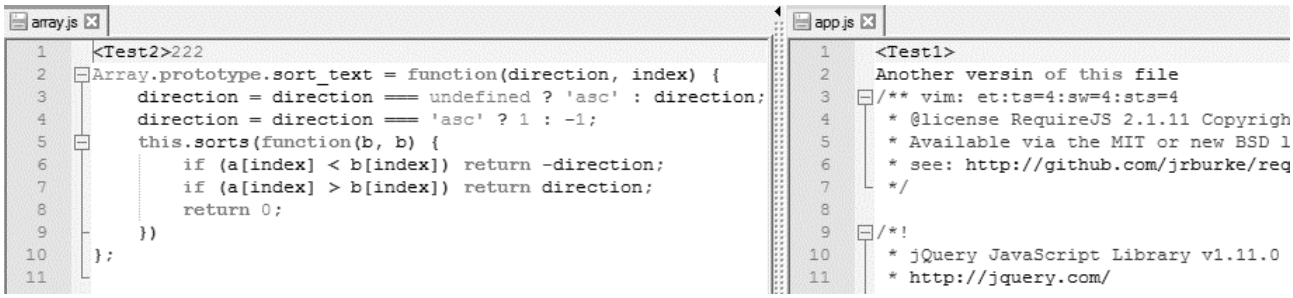


Рисунок 4.7 – демонстрація коректності файлів

Як видно файл «array.js» був успішно замінений файлом «array.js» з папки «Тест2», а файл «app.js» залишився без змін. Отже об'єднання папок відбулося успішно. Також всі файли та папки, яких не було в папці «Тест1», але які були в папці «Тест2», теж успішно перенеслися. Як видно з рисунку 4.8, в папку «Тест1» додалася папка «TPR» та ще 4 файли з папки «Тест2».

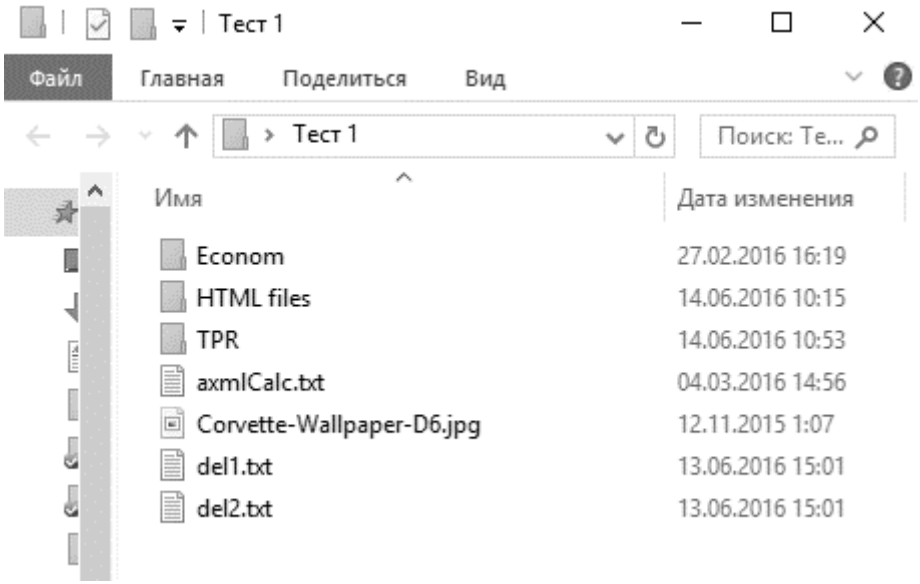


Рисунок 4.8 – папка «Тест1» після синхронізації

4.2 Порівняння та об’єднання папок з фільтром

Повернемо папки до початкового стану, як зображено на рисунку 4.1. Так само запустимо програму та оберемо ці 2 папки, «Тест1» та «Тест2», я зображено на рисунку 4.5. Порівнювати будемо лише текстові файли, тобто в фільтр портібно задати строку «*.txt». Також в папку «Текст1» скопіюємо 2 текстових файли «del1.txt» та «del2.txt» і трохи змінимо їх текст. Проведемо порівняння цих папок, результат порівняння видно на рисунку 4.9.

Compare result	Merge	First file	Last change	Second file	Last change
Diverse	First	del1.txt	6/14/2016 12:00:12 PM	del1.txt	6/13/2016 3:01:22 PM
Diverse	First	del2.txt	6/14/2016 12:00:19 PM	del2.txt	6/13/2016 3:01:55 PM

Рисунок 4.9 – звіт про порівняння

Як бачимо на цей раз файли з папки «HTML files» не порівнювались, адже не відповідають фільтру. Тепер об’єднаємо ці дві папки зберігши обидва файли з папки «Текст2». Під час об’єднання з другої папки мають перенестися лише файли з розширенням «.txt». Як бачимо з рисунку 4.10, перенеслися лише

файли з розширенням «.txt», а зображення з розширенням «.jpg» не перенеслося при об’єднанні.

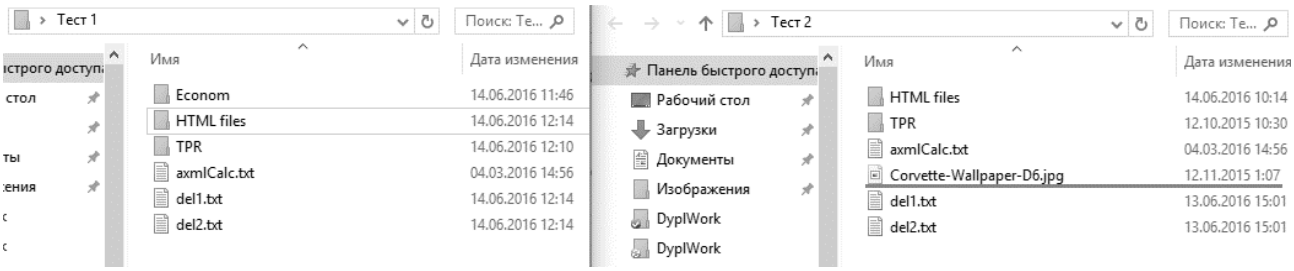


Рисунок 4.10 – результат синхронізації папок

4.3 Висновки до розділу 4

В цьому розділі було проведено тестування програмного продукту. В першій частині було проведено тестування порівняння та об’єднання двох папок без використання фільтру. Результати порівняння були правильними та очікуваними, об’єднання папок також пройшло успішно. В другій частині розділу проведено тестування об’єднання та порівняння файлів з використанням фільтрів. Результат цього тестування також був вірним, що є показником коректної роботи програми.

Оскільки тестування пройшло успішно і помилок виявлено не було, можна вважати, що програма працює коректно.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для порівняння, злиття та аналізу директорій та файлів. Інтерфейс користувача був розроблений за допомогою мови програмування C# у середовищі розробки Microsoft Visual Studio 2015 Community. Інтерфейс користувача створений за допомогою технології Windows Presentation Foundation.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Windows, без попереднього встановлення продукту за допомогою виконуючого файлу.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі

оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

5.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір оптимальної СКБД;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування C#;

б) мова програмування Java;

в) мова програмування C++;

г) мова програмування Python;

Функція F_2:

а) Microsoft SQL Serve;

б) Oracle.

б) MySQL.

Функція F_3:

а) інтерфейс користувача, створений за технологією WPF;

б) інтерфейс користувача, створений за технологією JavaFX.

в) інтерфейс користувача, створений за технологією Qt Quick

5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5.1).

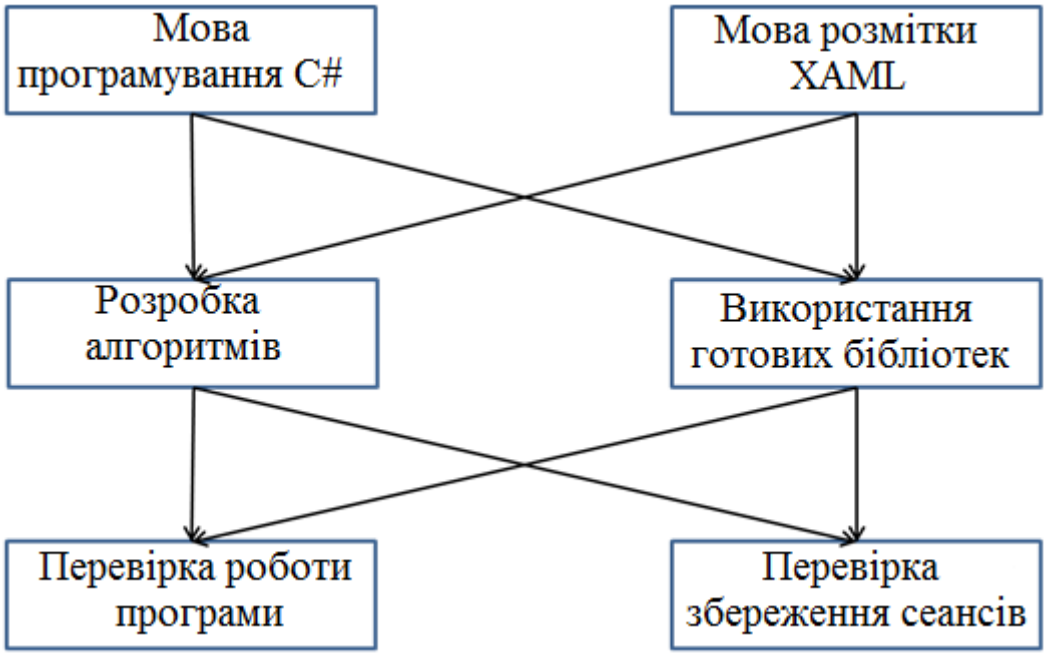


Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1 – Позитивно- матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Займає менше часу при написанні коду	Не кросплатформений
	Б	Кросплатформений	Низька швидкодія
	В	Кросплатформений	Висока швидкодія
	Г	Кросплатформений	Низька швидкодія
F2	А	Більш дешева вартість корпоративної ліцензії	Необхідність додаткової інсталяції, низький рівень користувацької підтримки
	Б	Подовжений термін користувацької підтримки	Більш висока вартість корпоративної ліцензії. Необхідність додаткової інсталяції
	В	Безкоштовна ліцензія	Вільна система керування реляційними базами даних
F3	*А*	Простота створення	Відсутність кросплатформеності
	Б	Простота створення,	Відсутність кросплатформеності,
	В	Відносно не просто в реалізації	Кросплатформений

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант б) і г) має бути відкинутий.

Функція *F2:*

Вибір СКБД не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) б) та в) гідними розгляду.

Функція *F3:*

Оскільки, програмний продукт реалізується на мові C#, використовуємо варіант А як єдиний можливий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а

2. F1а – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2 Обґрунтування системи параметрів ПП

5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

Таблиця 5.2 – Результати ранжування параметрів

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000

Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки запитів користувача	X3	мс	1000	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

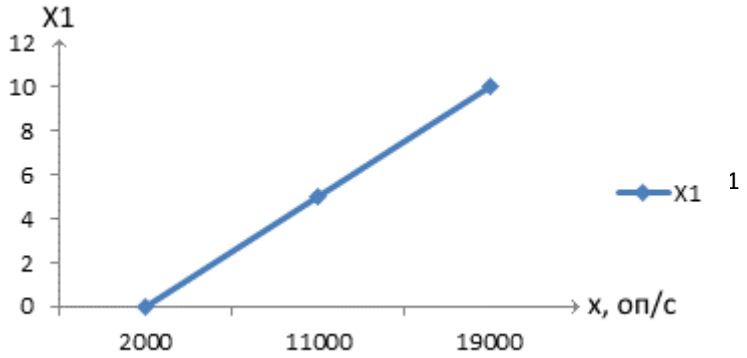


Рисунок 5.2 – X1, швидкодія мови програмування

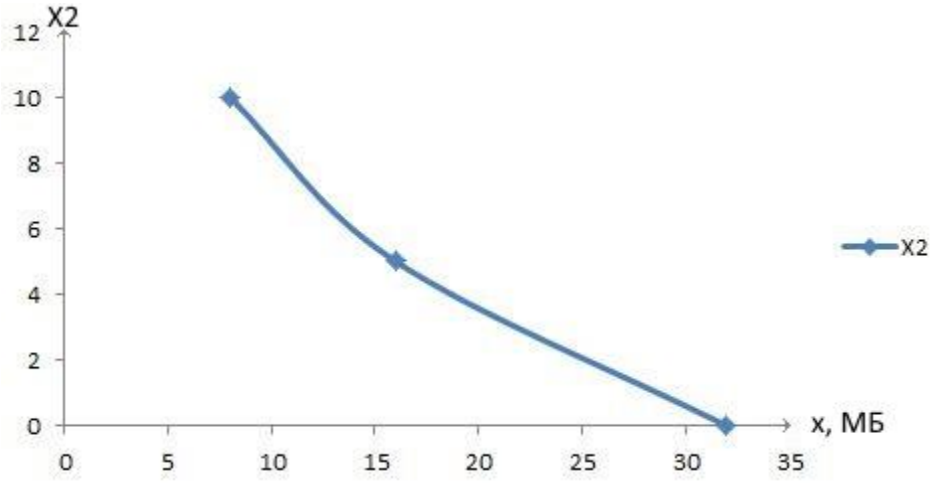


Рисунок 5.3 – X2, об’єм пам’яті для збереження даних

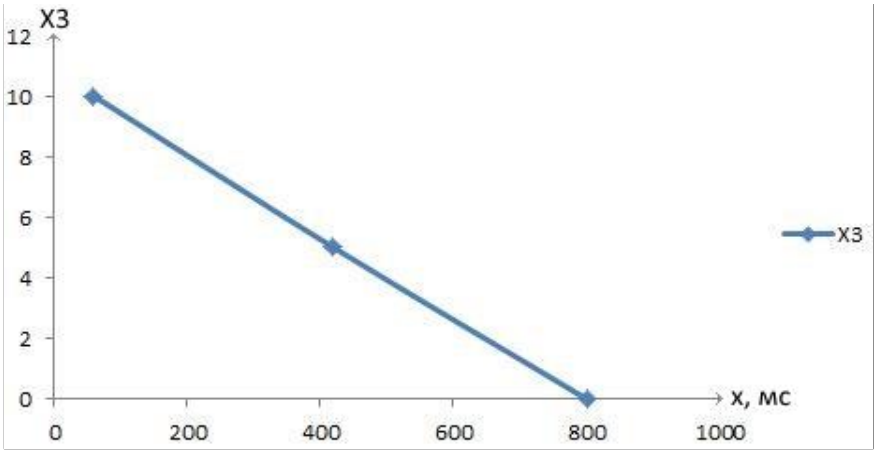


Рисунок 5.4 – X3, час виконання запитів користувача

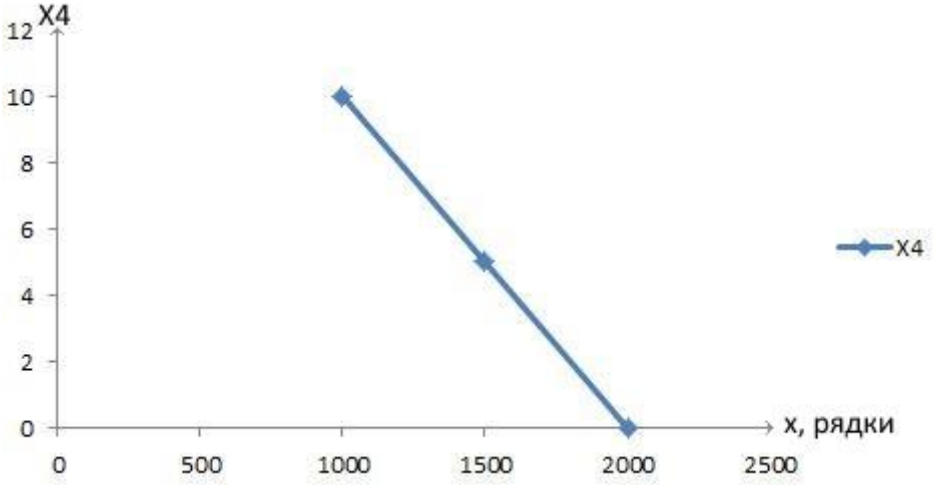


Рисунок 5.5 – X4, потенційний об’єм програмного коду

5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,57
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,04
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,26

	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105, \qquad (5.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25., \qquad (5.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75., \qquad (5.3)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67\,, \qquad (5.4)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j, \\ 0.5 \text{ при } X_i < X_j \end{cases} \qquad (5.5)$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{в}i} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij}., \qquad (5.6)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j., \qquad (5.7)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,221	23,15	0,216	100	0,208
X2	1,5	1,0	0,5	1,5	4,5	0,301	26,35	0,277	124,25	0,272
X3	1,5	1,5	1,0	1,5	5,5	0,334	33,25	0,339	156	0,353
X4	0,5	0,5	0,5	1,0	2,5	0,186	14,35	0,255	64,75	0,161
Всього:					16	1	98	1	445	1

5.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1000 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} , , \qquad (5.8)$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,208	0,751
F2(X2)	А	16	3,4	0,272	0,86
F3(X3,X4)	А	800	2,4	0,353	0,843
	Б	80	1	0,161	0,161

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], , \qquad (5.9)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0.751+ 0.86 + 0.843 = 2.434

K_{K2} = 0.751+ 0.86 + 0.161= 1.672

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.10)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1329{,}54$ людино-годин;

$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1346.53$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 5000 грн., один фінансовий аналітик з окладом 6000 грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} \text{грн.}, \qquad (5.11)$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_ч = \frac{5000 + 5000 + 6000}{3 \cdot 21 \cdot 8} = 31.74 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \qquad (5.12)$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 31.74 \cdot 1328.64 \cdot 1.2 = 50605.24$ грн.

II. $С_{ЗП} = 31.74 \cdot 1345.52 \cdot 1.2 = 51248.16$ грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 50605.24 * 0.22 = 11133.15$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 51248.16 * 0.22 = 11274.59$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 5000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 5000 \cdot 0,2 = 12000$ грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 12000 \cdot (1 + 0.2) = 14400$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0,22 = 14400 * 0,22 = 3168$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 10000 \cdot 0.05 = 575$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 0,9 \cdot 1.56 = 373.74$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 14400 + 3168 + 2875 + 575 + 373.74 + 6700 = 28091.74$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 28091.74 / 1706,4 = 16.46$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т \qquad (5.13)$$

I. $С_М = 16.46 * 1328.64 = 21869.41$ грн.;

II. $С_М = 16.46 * 1345.52 = 22147.25$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 50605.24 * 0,67 = 33905.51$ грн.;

II. $С_Н = 51248.16 * 0,67 = 34336.26$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н \qquad (5.14)$$

I. $С_{ПП} = 50605.24 + 3168 + 21869.41 + 33905.51 = 109548.16$ грн.;

II. $С_{ПП} = 51248.16 + 3168 + 22147.25 + 34336.26 = 110899.67$ грн.;

5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj}, \quad (5.15)$$

$К_{ТЕР1} = 2.434 / 109548.16 = 0.\ 22 \cdot 10^{-4}$;

$К_{ТЕР2} = 1.672 / 110899.67 = 0.15 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,\ 22 \cdot 10^{-4}$.

5.6 Висновки до розділу 5

Було проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП. Було обрано найбільш оптимальний варіант зважаючи на складність реалізації, швидкодію.

В другій частину ФВА розрахунки були присвячені вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації проводилось на основі: коефіцієнту ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу, що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,22 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – C#;

– СКБД MS SQL Server;

– інтерфейс користувача, створений за технологією WPF.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

Дипломна робота присвячена дослідженню методів порівняння та об’єднання папок, порівняння файлів, розробці програмного продукту для об’єднання і порівняння папок і файлів, створення звітів про порівняння папок, відображення відмінностей між файлами.

У результаті виконання дипломної роботи були отримані такі результати:

1. Здійснено аналіз файлових систем, що використовуються в ОС Windows, проведено їх порівняння.
2. Проаналізовано та обґрунтовано вибір засобів розробка програмного продукту, проведено аналіз технології WPF, платформи .Net.
3. Проаналізовано алгоритми порівняння файлів.
4. Запропоновано використовувати порівняння по хешу при порівнянні файлів. Це значно скоротило час, потрібний для порівняння файлів в порівняння з побайтовим порівнянням.
5. Реалізовано алгоритм порівняння текстових файлів.
6. Реалізовано механізм збереження сеансів.
7. Реалізовано алгоритм порівняння та об’єднання каталогів файлів.
8. Виконано тестування програмного продукту.
9. Виконано економічний аналіз розробки.

В роботі також зроблено детальний опис алгоритмів, що були використані в програмі, було описано функціонал для порівняння текстових файлів та функціонал для порівняння та об’єднання папок. Було розроблено зручний інтерфейс користувача, що передбачає управління вкладками (видалення та створення нових) , формування звітів про порівняння та підказки при невірних

кроках користувача. Було розроблено механізм збереження сеансів, який автоматично запам’ятовує стан програми при її закритті з можливістю завантажити останній сеанс при старті роботи програми.

Щоб перевірити працездатність програми було проведено тестування програмного продукту. Було зроблено тест роботи програми без використання фільтрів, де було перевірено коректність порівняння папок, вірність формування звіту по порівнянню та перевірено механізм об’єднання папок. Також проведено тестування роботи програми з використанням фільтрів, під час якого було перевірено коректність роботи фільтрів під час порівняння папок, формуванню звітів по порівнянню та об’єднанню папок.

Для подальшого розвитку необхідно удосконалити структуру програми провести рефакторинг коду, що значно спростить дописування нових модулів. Необхідно розробити ефективний спосіб візуалізації відмінностей у вмісті файлів, що допоможе значно розширити можливості по управлінню змінами в файлах. Також необхідно проаналізувати можливість розробки системи контролю версій та розробити ефективний алгоритм для управління версіями. Тоді програмний засіб можливо буде використовувати в якості складової систем контролю версій. Можна проаналізувати можливість підключення до віддалених сховищ даних, тоді програму можна буде використовувати для хмарних сховищ інформації, що є доволі цікавою та актуальною можливістю.

Основна потенційна галузь застосування – системи контролю версій. Використовуючи розробку процес спростити процес синхронізації папок та файлів, зручний та зрозумілий інтерфейс робить програму простою у користуванні, що теж є плюсом.

ПЕРЕЛІК ПОСИЛАНЬ

1. Офіційний сайт компанії ScooterSoftware – Режим доступу : http://www.scootersoftware.com/ – Дата доступу : 04.05.2016.
2. Офіційний сайт компанії CodeSoft – Режим доступу : http://cmpp.coodesoft.com/ – Дата доступу : 04.05.2016
3. Офіційний сайт компанії Araxis – Режим доступу : http://www.araxis.com/merge/ – Дата доступу : 05.05.2016
4. Офіційний сайт компанії Ultraedit – Режим доступу : http://www.ultraedit.com/ - Дата доступу : 05.05.2016
5. Офіційний сайт компанії prestoSoft – Режим доступу : http://www.prestosoft.com/ - Дата доступу : 05.05.2016
6. Офіційний сайт компанії syntEvo – Режим доступу : http://www.syntevo.com/ - Дата доступу : 06.05.2016
7. Офіційний сайт програмного продукту WinMerge – Режим доступу : http://www.winmerge.org - Дата доступу : 07.05.2016
8. Офіційний сайт програмного продукту MeldMerge – Режим доступу : http://meldmerge.org/ - Дата доступу: 09.05.2016
9. Офіційний сайт програмного продукту Diffuse – Режим доступу : http://diffuse.sourceforge.net/ - Дата доступу: 09.05.2016
10. Офіційний сайт компанії Perforce – Режим доступу : https://www.perforce.com/ - Дата доступу: 09.05.2016
11. Remzi H., Andrea C./ Operating Systems: Three Easy Pieces/ H. Remzi, C. Andrea// Arpaci-Dusseau Books, 2014 – P. 52 – 60.
12. Гордеєв А. В.// Операционные системы: Учебник для вузов. — 2-е изд./ А. В. Гордеєв. – СПб.: Питер, 2007. – С 416.
13. Скит Д. /C# для профессионалов: тонкости программирования, 3-е издание, // Д. Скит. — М.: «Вильямс», 2014. — С 608.
14. Сайт підтримки компанії Micrisoft – Режим доступу : https://support.microsoft.com/ - Дата доступу: 01.06.2016