Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: **«МЕТОД ТА ПРОГРАМНА РЕАЛІЗАЦІЯ БЛОЧНОГО ПОРІВНЯННЯ ФАЙЛІВ»**

Виконав: студент 2 курсу, групи 1КІ-18м
спеціальності
123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)

Гудименко Олександр Олександрович
(прізвище та ініціали)
Керівник доц. каф. ОТ, к.т.н. Савицька Л.А.
(прізвище та ініціали)
Рецензент доц. каф. МБІС, к.т.н. Поплавський А.В.
(прізвище та ініціали)

м. Вінниця - 2020 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки 12 – Інформаційні технології
Спеціальність 123 – Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри Т. Б. Мартинюк

_____
"____" _____20___року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Гудименку Олександру Олександровичу

1. Тема проекту (роботи) Метод та програмна реалізація блочного порівняння файлів
Керівник проекту (роботи) Савицька Людмила Анатоліївна, к.т.н., доц. каф. ОТ
затверджені наказом вищого навчального закладу від "___"_____20__року №___
2. Строк подання студентом проекту (роботи)_____
3. Вихідні дані до проекту (роботи) список технічної літератури, аналіз, вивчення та дослідження процесів синхронізації та порівняння файлів чи папок, технічне завдання на магістерську роботу.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз сучасних методів та засобів порівняння та синхронізації файлів. Керування процесом передавання та контроль достовірності даних. Аналіз порівняння даних за методом CRC16. Задача порівняння файлів. Огляд програмних засобів для порівняння та подальшої синхронізації файлів. Огляд мережевих сервісів для порівняння та синхронізації файлів. Аналіз отриманих даних. Метод та програмна реалізація блочного порівняння файлів. Загальна схема методу блочного порівняння файлів. Процес порівняння файлів. Процеси підрахунку та порівняння хеш-кодів за методом CRC 16. Побудова користувацького інтерфейсу для програми блочного порівняння файлів. Етапи створення програми блочного порівняння файлів. Тестування і перевірка правильності роботи програми. Класи та функції програми блочного порівняння файлів. Економічна доцільність розробки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Приклад розрахунку контрольної суми методом CRC16. Алгоритм розрахунку контрольної суми методом CRC16. Результат аналізу в GoodSync. овнішній вигляд Viceversa Pro та Allway Sync. Інтерфейс Freefilesync та налаштування кліента в SugarSync. Оцінка конкурентів. Схематичне представлення роботи методу блочного порівнння файлів. Схематичне представлення роботи процесу порівнння файлів. Схематичне розширене представлення роботи методу блочного порівнння файлів. Блок схема роботи алгоритму процесів підрахунку та порівняння хеш-кодів за методом CRC 16. Тестування і перевірка правильності роботи програми.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Технічний розділ	Савицька Л.А.		
Економічний розділ	Глущенко Л. Д.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
	Огляд існуючих підходів до розв'язання задачі		
	Аналіз сучасних методів та засобів порівняння та синхронізації файлів		
	Узагальнення інформації		
	Розробка методу та програмної реалізації блочного порівняння файлів		
	Узагальнення інформації		
	Проведення економічних розрахунків		
	Оформлення пояснювальної записки до дипломної роботи		
	Оформлення додатків та графічного матеріалу		

Студент _____ Гудименко О.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Савицька Л.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дана магістерська кваліфікаційна робота присвячена розробці та програмній реалізації методу блочного порівняння файлів. Цей програмний засіб дозволить порівнювати файли на персональному комп'ютері за допомогою застосування механізму CRC-кодів.

Високий рівень вирішення поставленої задачі досягнуто за рахунок використання сучасної мови програмування Java. Можливості Java дозволяють інтегрувати в програму кілька варіантів порівняльних функцій: індивідуальну, групову, глибинне порівняння із контролем змісту, а також визначення унікальних файлів. Це зробить програму зручним інструментом програміста та керівника великими спільними проектами.

В даній магістерській роботі виконано дослідження і аналіз сучасних методів та засобів порівняння та синхронізації файлів, розглянуто існуючі методи вирішення задачі порівняння файлів, технологічний ланцюжок роботи методу блочного порівняння файлів та розроблено новий метод блочного порівняння файлів, запропоновано ключові процеси роботи методу блочного порівняння файлів, алгоритм роботи процесів підрахунку та порівняння хеш-кодів за методом CRC 16, який дозволяє відстежувати зміни у вмісті файлу.

REVIEW

This master's qualification work is devoted to the development and program implementation of the method of block comparison of files. This software tool will allow you to compare files on a personal computer using the mechanism of CRC codes.

The high level of the solution to this task was achieved through the use of modern Java programming language. Java capabilities allow you to integrate several options of comparative functions into the program: individual, group, in-depth comparison with content control, as well as the definition of unique files. This will make the program a convenient tool for the programmer and the manager with great collaborative projects.
In this master's thesis the research and analysis of modern methods and means of comparison and synchronization of files was made, existing methods for solving the problem of file comparison, the technological chain of work of the method of block comparison of files were considered, and a new method of block comparison of files was developed, the key processes of the method of block comparison of files, algorithm The processes of counting and comparing hash codes using the CRC 16 method, which allows you to track changes in the contents of the file.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Останні десятиріччя означені швидким розвитком науки і техніки у переважній кількості галузей людської діяльності [1-3]. На території України все більше стають популярними та набувають розвитку компанії з іноземним капіталом, діяльність яких пов’язана з outsource-розробкою програмних засобів [4-7]. В загальному випадку, процедура порівняння файлів широко використовується в галузі програмування, де ведеться спільна робота над великими проектами і є необхідним часто відслідковувати зміни, які вносять в робочі файли проектів різні користувачі і команди, тому швидкість відслідковування цих змін під час синхронізації файлів та (або) каталогів є **актуальним** завданням – щоб протягом порівняння не були внесені повторні зміни [8-9].

Якщо над файлом чи папкою працюють двоє або декілька осіб (часто – груп осіб), пересилаючи цей файл (папку, архів) один одному для перегляду чи коригування, часто важко встановити, які саме зміни було внесені до нової версії (копії) файлу або папки чи архіву [10-12].

Іноді трапляється, що працівники редагують один файл одночасно (наприклад, файл бази даних). У такому випадку може трапитися так, що зміна буде внесена у один і той самий рядок чи стовпчик. Таким чином виникне специфічна задача, яка полягає у такому: якщо наперед невідомо чи були внесені зміни у однакові частини документа, редактори самі можуть знищити результати власної праці перезаписуванням цих результатів роботою іншого редактора, отриманою як файл різниці.

Відомий алгоритм роботи методу CRC 16 [13-16] дозволяє звести процедуру порівняння файлів до порівняння хеш-кодів, які вираховуються як строкове представлення контрольних кодів для блоків визначеного розміру, на які поділяється вміст файлу.

Предметом даної магістерської роботи є програмна реалізація порівняння файлів та каталогів за допомогою хеш-коду за методом CRC 16 [17-19], який дозволяє відстежувати зміни змісту файлу без витрат на рядкове чи двійкове порівняння. Саме такий метод дозволяє виконувати швидке порівняння достатньо великих обсягів даних.

Отже, задача блочного порівняння файлів має важливий прикладний характер, і для її розв’язання достатньо класичних методів, що і спричинює можливість застосування нових засобів сучасних інформаційних технологій з метою досягнення кращих показників аналізу та порівняння вмісту великих масивів файлів.

Тема магістерської кваліфікаційної роботи є актуальною, оскільки завжди існує потреба у організації порівняння великих масивів файлів, а використання спеціалізованого програмного забезпечення дозволить пришвидшити процес такого порівняння. Запропонований метод та програмна реалізація блочного порівняння файлів будуть корисним для державних та приватних компаній, особливо за умов, що їх діяльність тісно пов’язана із галузями ІТ-індустрії.

Використання сучасних інформаційних технологій, також відомих методів для розв'язання даної поставленої задачі є актуальним, оскільки такий підхід надає широкі можливості з метою модифікацій і налаштування нових різних методів під дану задачу блочного порівняння, а отже – дозволяє досягти кращого, у порівнянні з традиційними існуючими методами, результат.

Таким чином, розв’язання задачі блочного порівняння файлів та папок забезпечить подальше впровадження інформаційних технологій у галузь організації документообігу, особливо, під час спільної роботи над великими проектами.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри обчислювальної техніки Вінницького національного технічного університету; у рамках Національної стратегії розвитку освіти в Україні на період до 2021 р. (наказ Президента № 334/2012 від 25.06.2013 р.); плану наукової та навчально-методичної роботи кафедри обчислювальної техніки.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є пришвидшення порівняння достатньо великих обсягів файлів.

Для досягнення поставленої у роботі мети необхідно розв’язати такі завдання:

− провести аналіз сучасних методів та засобів порівняння та синхронізації файлів;

− розглянути існуючі методи вирішення задачі порівняння файлів та постановка вимог та обрати й обґрунтувати вибір методу, що задовольняв би меті даної магістерської кваліфікаційної роботи;

− розглянути існуючі способи керування процесом передавання та контроль достовірності даних;

− запропонувати технологічний ланцюжок роботи методу блочного порівняння файлів та розробити метод блочного порівняння файлів згідно мети магістерської кваліфікаційної роботи;

− розробити ключові процеси роботи методу блочного порівняння файлів та на його основі цього розробити програмний засіб;

− виконати програмну реалізацію запропонованого методу блочного порівняння файлів;

− провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – це сучасні процеси спільної роботи над проектами на підприємстві із використанням сучасних інформаційних технологій.

Предмет дослідження – це методи та програмні засоби блочного аналізу та порівняння файлів.

Методи дослідження. Дослідження, виконані під час роботи над кваліфікаційною магістерською роботою, ґрунтуються на теоретико-множинних підходах і принципах кросплатформеного підходу для розбиття вмісту файлів на блоки та підрахунку хеш-кодів кожного блоку; структурному проектуванні програмного забезпечення – для реалізації кросплатформеного програмного забезпечення блочного порівняння файлів; методах об'єктно-орієнтованого програмування – для реалізації методу блочного порівняння файлів та розробки відповідного програмного забезпечення.

Наукова новизна одержаних результатів полягає в такому:

− вперше запропоновано метод блочного порівняння файлів, без витрат на рядкове чи двійкове порівняння. Запропонований у даній роботі метод блочного порівняння файлів дозволяє пришвидшити достатньо великих обсягів даних;

− вдосконалено процес порівняння двох файлів для програми блочного порівняння файлів за рахунок застосування можливостей хеш-кодів із використанням методу CRC 16, який дозволяє відстежувати зміни у вмісті файлу.

Практичне значення одержаних результатів полягає у такому:

1. Розроблено новий метод блочного порівняння файлів.

2. Вдосконалено процес порівняння двох файлів за рахунок застосування можливостей хеш-кодів із використанням методу CRC 16.

3. Розроблено алгоритм роботи процесів підрахунку та порівняння хеш-кодів за методом CRC 16, який дозволяє відстежувати зміни у вмісті файлу.

4. Знайшов подальшого розвитку технологічний ланцюжок роботи методу блочного порівняння файлів.

5. Розроблено програмний засіб для блочного порівняння файлів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів та інформаційних технологій під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів дослідження з результатами, що отримані під час впровадження розробленого програмного засобу.

Особистий внесок магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ЗАСОБІВ ПОРІВНЯННЯ ФАЙЛІВ

1.1 Керування процесом передавання та контроль достовірності даних

Спосіб комунікації між двома вузлами в процесах комунікації називають режимами передачі. Розрізняють такі режими передачі даних:

- симплексний;
- дуплексний;
- напівдуплексний.

1.1.1 Режими передачі

При симплексному (simplex) режимі передачі даних приймач і передавач зв'язуються лінією зв'язку, якою інформація передається в одному (і тільки в одному) напрямку. Передавальний вузол в симплексному режимі передачі даних повністю займає канал. Приклади: радіомовлення, телемовлення [21].

Напівдуплексний (half duplex) режим передачі даних допускає передачу даних в двох напрямах, але в різні моменти часу. Два вузли зв'язуються таким каналом зв'язку і передачі даних, який дозволяє їм почергово (увага, не одночасно) передавати інформацію та дані. Для зміни напряму передачі, зазвичай, застосовується передача спеціального сигналу і отримання на нього підтвердження.

Дуплексний або повнодуплексний (duplex, full duplex) режим передачі даних дозволяє одночасно передавати інформацію в обох двох напрямах. У простому випадку для процесу дуплексного зв'язку застосовується дві лінії зв'язку (перша – пряма і друга – зворотна), але існують такі рішення, що дозволяють підтримувати дуплексний режим на єдиній лінії (скажімо, обидва вузли передачі даних можуть одночасно передавати, а з сигналу, що ними приймається, віднімати власні відомі дані). Відомо, що дуплексний режим може бути симетричним або асиметричним.

1.1.2 Асинхронний, синхронний, ізохронний і плезіохронний режими передачі даних

Для послідовної передачі даних достатньо однієї лінії зв’язку, по якій можуть послідовно передаватися біти даних. При цьому, приймач даних має вміти розпізнавати, де починається і де закінчується сигнал, що відповідає кожному біту даних. Інакше кажучи, передавач і приймач мають вміти синхронізуватися [22]. У випадку, якщо якість синхронізації під час передачі даних низька (відхилення під час

передачі одного біта досягає декількох відсотків), і тоді вже застосовується асинхронний (asynchronous) режим передачі інформації: виконується узгодження синхронних генераторів на початку передачі кожного байта. Передача байту починається зі спеціального, так званого, старт-біта, потім йдуть біти даних, а за ними, ймовірно, біт парності. Після передачі усіх бітів даних передається власне стоп-біт. Обидва, і старт-біт і стоп-біт завжди мають визначене значення:

- старт-біт кодується логічним нулем «0»;
- стоп-біт – логічною одиницею «1».

Між власне передачею стоп-біта одного байта і старт-біта наступного байта проходить довільний час. При цьому швидкість передачі даних в асинхронному режимі обмежена. Це сотні кілобіт в секунду (стандартні швидкості: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 біт/с).

Якщо синхронізація досить якісна (скажімо, застосовується додаткова лінія, по якій передаються синхросигнали), то можна передавати потік інформаційних даних без додаткової синхронізації окремих байтів. Такий режим називають синхронним режимом (synchronous). Тоді передача бітів даних передує і закінчується видачею в канал передачі символу синхронізації. За відсутності даних взагалі, передавач повинен постійно передавати в канал ці символи синхронізації. У випадку ізохронної (isochronous) передачі даних, відправка кадрів даних відбувається в певні (відомі і приймачу і відправнику) моменти (відліки) часу. При цьому дані, що є передаваними одним вузлом А з постійною швидкістю, поступатимуть до приймача В з тією ж швидкістю. Ізохронна передача даних необхідна, скажімо, для доставлення цифрованого відеозображення чи звуку. Плезіохронна (plesiochronous) передача даних вимагає також внутрішньої синхронізації вузлів від кожного з джерел із номінально співпадаючими частотами. Сам термін “плезіохронна” означає “майже синхронна” передача, оскільки частоти джерел точно не можуть співпадати і не співпадають, а з часом накопичується розбіжність, яка компенсується деякою вставкою фіктивних даних.

1.1.3 Контроль достовірності даних під час передачі їх по каналу зв’язку

При передачі даних засобами складних та складених мережа, що мають лінії зв'язку з відповідними перешкодами часто, застосовують відповідні заходи для забезпечення і контролю достовірності інформації (тобто, відсутності помилок) [23].

Контроль достовірності передачі даних по каналу зв'язку вимагає введення деякої надлишковості в передавану інформацію. Така надлишковість може вводитися:

- на рівні окремих символів або груп символів;
- на рівні передаваних кадрів.

У свої надмірні поля передавач А розміщує деякий код, що обчислюється по певним правилам з корисної інформації, що передається. Приймач Б порівнює цей отриманий код із тим значенням, що він обчислив самостійно. Виявлена невідповідність значень свідчить про деяке спотворення інформації, а от збіг свідчить про високу ймовірнсть правильної передачі даних. Ймовірнсть виявлення такої помилки залежить від обраної схеми контролю даних і також, співвідношення розмірів даних інформаційного і контрольного полів. Тут найненадійніша схема – це є контроль паритету (досить проста схема). А найекономічніша схема тут – дублювання інформації.

При цьому, найнадійніше виявлення спотворень чи неспівпадінь дає CRC-контроль, реалізація якого дійсно складніша, але накладні витрати, як не дивно, менші (типові - 2 байти на 4 Кбайт даних). Також, між ними знаходиться схема з обчисленням та розрахунком контрольної суми.

Пр цьому процес дублювання інформації зводиться до процесу повторення одного й того ж елементу двічі. Приймач А лише перевіряє збіг чи незбіжність копій. Така копія іноді представляється в, так званому, інверсному вигляді. Тоді дублювання з метою якісної організації процесу, застосовують лише для окремих коротких елементів кадру, для яких контроль іншими способами є незручним. І тоді дублювання призводить до 100% накладних витрат та на практиці застосовується, скажімо, в байті стану кадру Token Ring.

Спеціалізований контроль паритету (parity check) використовує контрольний біт, що доповнює число одиничних бітів контрольованого елементу до парного (це є even parity, тобто, контроль парності) або непарного (це є odd parity, тобто, контроль непарності) значення, і залежить від прийнятої
го узгодження.

Цей контроль паритету дозволяє програмно виявляти тільки помилки непарної кратності, при цьому спотворення двох, чотирьох і так далі біт контрольованої області залишаться досі непоміченими. При такому підході, контроль може бути

«горизонтальним» або «вертикальним». Дід час горизонтального контролю деякий біт паритету «Р» вводиться до кожного символ, що передається – і цей вид контролю широко застосовується в послідовних інтерфейсах RS-232C, а також, інших, тоді при цьому накладні витрати складають 1 біт на кожен байт.

А от контрольні суми і CRC-коди успішно застосовуються для контролю цілого кадру (або навіть його декількох полів). Для цього в самому кадрі визначається спеціальне окреме поле для контрольного значення коду, це зазвичай 1, 2 або 4 байти. Тоді контрольна сума застосовується для кадрів довжиною кратною одному байту (тобто, одному слову).

Поняття контрольна сума (checksum) зазвичай обчислюється як сума і доповнення суми усіх байтів кадру до нуля (або ж значення FF – усіх двійкових одиниць). Процес підсумовування виконується по модулю, що відповідає розрядності поля деякого контрольного коду. При цьому приймач А, підсумувавши по тому ж самому модулю усі байти даних або слова кадру, включно з контрольними кодами, має отримати те ж саме число (це буде 0 або FF). Отож, контрольна сума є найпростішим засобом контролю цілісності кадру даних, і ймовірнсть виявлення помилок – тут складає до 99,6 %.

Надмірний циклічний контроль, скорочено CRC (англ., Cyclic Redundancy Check) використовує дещо складніший алгоритм. Вибирається деякий породжуючий поліном, розрядністю більшою за поле, що виділене для контрольного коду. Отож, для звиичайного 16-бітового CRC рекомендується поліном x16+x12+x5+1 (10001000000100001b). Тоді кадр (який є довгим двійковим числом) з обнуленим полем CRC ділиться на цей поліном, і тоді залишок від цього ділення (звісно, по модулю розрядності поля CRC) надалі переноситься в поле CRC передаваних даних. Приймач Б ділить прийнятий ним кадр даних (разом з цим полем CRC) на той же вказаний поліном і, відповідно, порівнює залишок від ділення з деяким еталоном (наприклад, нулем або іншим числом). Якщо ж ці числа співпали, тоді ухвалюється рішення про коректність даного кадру. Відомий 16-бітовий CRC код дозволяє просто виявляти такі помилки (тобто, спотворення групи сусідніх бітів) з ймовірністю до 99,9984%. І він же, 16-розрядний CRC-код застосовується в багатьох мережевих протоколах усіх протокольних стеків.

Ще один метод контролю – ECC-контроль (англ., Error Correction Code), дозволяє не лише виявити, але і часто – виправляти деякі помилки. У складених мережах ECC практично не застосовується, оскільки він вимагає ще більшої надлишковості (поясними: він застосовується в пристроях зберігання даних, де у випадку помилки немає можливості повторної передачі даних).

У загальному випадку, керування потоком передачі даних (англ., data flow control) є засобом узгодження темпу (швидкості, розмір вікна тощо) передачі даних в протокольному стекові з можливостями приймача. При цьому, бітові швидкості приймачів і передавачів завжди (!) повинні співпадати (якщо інакше, тоді зовсім неможливе коректне декодування прийнятих даних взагалі), але, звісно, можливі ситуації, коли передавач А передає інформацію в такому темпі, неприйнятному для приймача Б. При цьому вхідний буфер приймача Б надто переповнюється і частина передаваної інформації тимчасово втрачається. Засоби керування таким потоком даних дозволяють приймачу Б подати передавачу А деякий сигнал на припинення або продовження передачі (зменшення и збільшення розміру вікна у протоколі TCP). Саме такі засоби вимагають наявності, так званого, зворотнього каналу передачі.

З метою контролю передавання та отримання інформації приймачем Б застосовують квитування (handshaking – рукостискання) – відсилання спеціального повідомлення про отримання кадру (пакету). На кожний прийнятий кадр приймач Б відповідає коротким кадром-підтвердженням. Також, може бути підтвердження на групу кадрів. Тоді у випадку отримання коректного кадру посилається таке позитивне підтвердження ACK (ACKnowledge), а у випадку помилкової ситуації – негативне NACK (Negative ACKnowledge), або іноді – відсутність підтвердження взагалі протягом певного часу. Все ж, при отриманні NACK передавач А відразу повторно посилає кадр для отримувача Б. За умови відсутності підтвердження протягом певного часу тайм-ауту (timeout) передавач А також виконує повторну додаткову передачу даних, але на очікування в цьому випадку вже втрачається деякий час. Таке підтвердження можна і потрібно використовувати і для керування потоком даних – відсилання підтвердження є і сигналом готовності. Недолік такого підходу – темп відсилання кадрів природньо обмежується часом обертання маркера чи кадра по мережі.

Поняття пакетна передача даних (англ.,burst transfer), під час якої передавач А відсилає серію послідовних кадрів, на яку повинен отримати загальне підтвердження від отримувача Б. Якщо ж в самому підтвердженні є місце для переліку, так званих, хороших і поганих кадрів, то відсилати вдруге можна тільки погані кадри (невдало передані). Тоді з'являється потреба в ідентифікації цих кадрів і їх підтверджень.

Розроблено метод «ковзаючого вікна» в протоколі TCP, що є гібридом індивідуальних підтверджень та пакетної передачі одночасно. Тут передавач А посилає серію нумерованих кадрів, знаючи, що прихід підтвердження на них може затримуватися відносно номеру кадру на час оберту маркеру чи кадру по мережі. Цей час заздалегідь обраховується і визначений, і ширина так званого, «вікна» визначається тим числом кадрів, які можна відіслати на деякій вибраній швидкості каналу передачі за час оберту. Ці підтвердження, звісно, нумеруються відповідно до кадрів, і ця нумерація може бути як циклічною по модулю, що зазначений шириною самого вікна передачі. Якщо ж передавач А не отримує підтвердження на свій переданий кадр, що виходить з вікна, тоді він вважає його назавжди втраченим і повторює його передачу знову. Коли отримання NACK відбулося, тоді сама передача повторюється тільки для цього єдиного кадру. На випадок повторного передавання, передавач А має тримати в своєму буфері усі ті кадри вікна, заміщаючи лише ті, що підтверджені, новими кадрами. Цей дієвий метод дозволяє повністю використовувати пропускну спроможність каналу передачі даних незалежно від дальності передачі [24].

1.2 Аналіз порівняння даних за методом CRC16

Метод CRC (Cyclic Redundancy Check) – є одним із поширених методів розрахунку контрольних сум. В основі методу CRC лежить подача бітових послідовностей у вигляді многочленів з коефіцієнтами 0 або 1, де порядок бітів відповідає ступеню доданка (звісно, починаючи з 0-го), а значення біту відповідно – його коефіцієнту. Застосовується метод CRC таким чином [13-16].

1) Обирається перший утворюючий многочлен (назвемо його G(x)), який буде спільний для усіх вузлів мережі. Тут старший і молодший біти даного многочлену мають дорівнювати одиниці «1».

2) Надалі, послідовність усіх інформаційних бітів даного кадру формують многочлен кадру (назвемо його M(x)). Кількість бітів даного кадру

m має бути більше кількості біт утворюючого многочлену r. Ідея полягає у тому, щоб у кінець кадру додавалася така послідовність бітів, яка би разом з іншими бітами кадру ділилася по модулю 2 на утворюючий многочлен без утворення остачі.

3) Отримувач, отримавши кадр, який містить контрольну суму CRC, розділить його на многочлен G(x). Тоді ненульова остача буде означати помилку. Ділення за модулем 2 – це означає використання при діленніх двійкови арифметичних дій додавання без використання переносів, тобто, застосування віднімання без використання позичання, тобто заміняються на логічну операцію XOR (рис. 1.1).

Детальний алгоритм роботи CRC є таким:

1) При визначеному степені r многочлену G(x), додається r нульових бітів до останнього многочлену, щоб він мав у собі m+r бітів, тобто відповідав многочлену $x^r \cdot M(x)$;

2) Надалі – бітова послідовність ділиться по модулю 2, що відповідає тому многочлену $x^r \cdot M(x)$ на, відповідно, бітову послідовність утворюючого многочлену G(x);

3) Далі остача від ділення і буде тією контрольною сумою, що додається в кінець результуючого кадру.

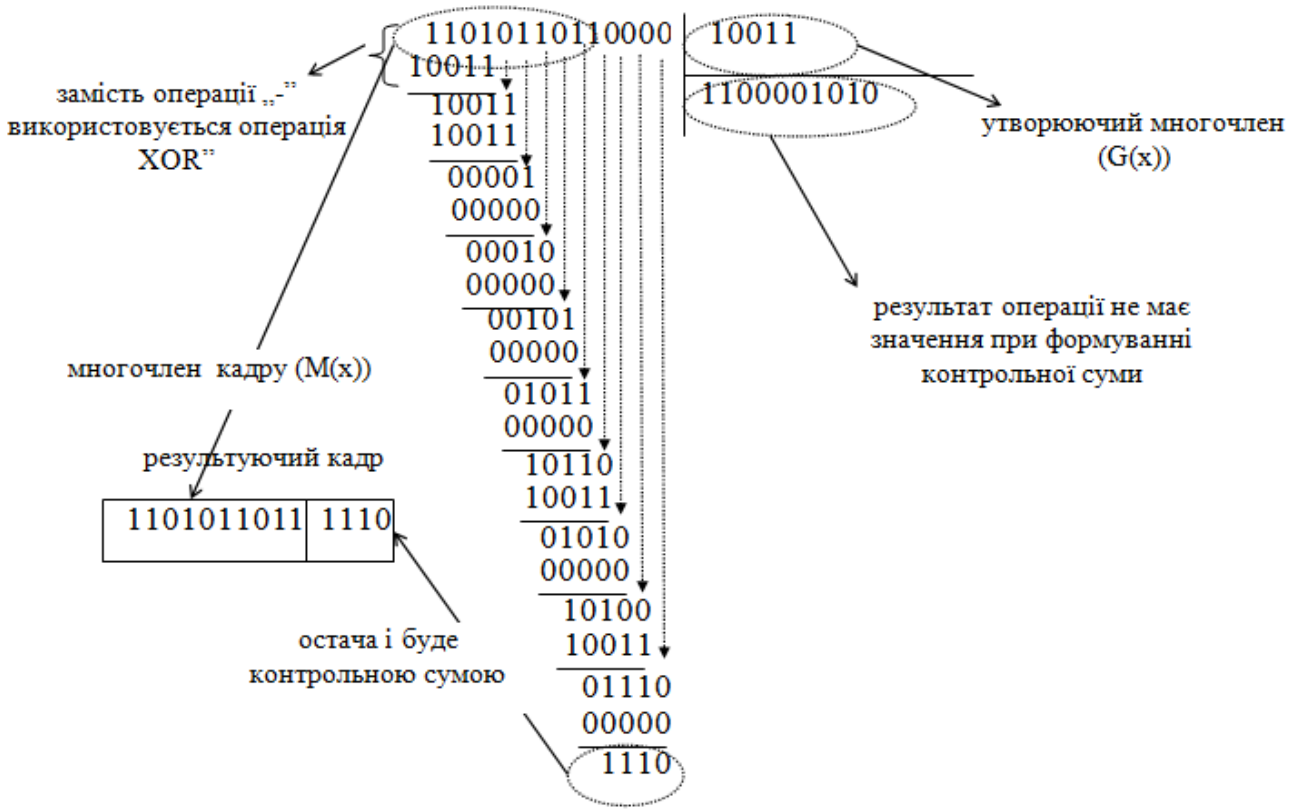


Рисунок 1.1 – Приклад розрахунку контрольної суми методом CRC16

Розглянемо такий приклад. Вираховується контрольна сума для деякого кадру із таким вмістом: 1101011011. Тоді маємо при утворюючому многочлені G(x)=x4+x+1 послідовність (10011). Відповідно, повертаючись до умови, поділимо послідовність бітів 1101011011, які доповнюються 4-ма логічними нулями на послідовність 10011. На рисунку 2.1 показано, що при такому бінарному діленні, замість операції віднімання застосовується операція виключного АБО (XOR). Результуюча остача і, відповідно, й контрольна сума буде рівною виразу 1110. В комп’ютерну мережу буде відправлений кадр такого змісту: 11010110111110.

Контрольна сума CRC застосовується в багатьох мережевих протоколах та протокольних стеках з різними значеннями утворюючих многочленів. Скажімо в Modbus широко застосовується метод CRC16 з використанням 16-бітної контрольної суми і таким утворюючим многочленом: G(x)=x15+x13+1 (тобто 10100000 00000001). Популярним, звісно. також є CRC32, який має 32-бітну контрольну суму.

Алгоритм обрахунку контрольних сум за методом CRC, що наведений вище, доволі складний, а тому на практиці часто застосовують більш простіші модифікації. Скажімо, в мережевому обладнані застосовують алгоритми із використанням так званого регістру зсуву, який апаратно значно простіше реалізовується. Також, інколи

для швидшого обрахунку використовують заздалегідь створені таблиці усіх можливих комбінацій кодів CRC для старшого і молодшого регістрів відповідно [17-19].

Алгоритм обрахунку за методом CRC, який наведений вище, як правило не застосовується напряму. На практиці розробники частіше користуються алгоритмом, який базується по закладеним заздалегідь даним готових результатів обчислень за методом CRC по будь якій з комбінацій байтів. Такий підхід прискорює процес обрахунку, що дійсно важливо під час обміну даними. Однак, для кращого розуміння цього процесу розрахунку і для можливості його постійного використання, скажімо, в лабораторних дослідах, можна скористатися алгоритмом, що наведений нижче.

Метод CRC16 використовує характеристичний поліном такого вигляду:

$$x16+x15+x2+1$$

або

1010000000000001

(A00116)

Опишемо алгоритм кроками нижче та зобразимо його графічно на рисунку 2.2.

Алгоритм розрахунку контрольної суми методом CRC16 виглядає таким чином:

1) На початку цього розрахунку виділяють слово (або два байти) під поле контрольної суми CRC і заповнюють надалі усі біти логічними одиницями «1».

2) Далі перший байт повідомлення чи кадру додадуть за функцією виключного АБО із даним полем.

3) Далі перевіряють нульовий біт на наявність одиниці «1».

4) При від’ємному результаті надалі проводять побітовий зсув коду CRC на один біт вправо, заповнюючи старший біт нулем «0».

5) При додатньому результаті зсув також проводять, але після цього додають за функцією виключного АБО характеристичний поліном, тобто, вираз A00116.

6) І таку операцію зсуву з перевіркою нульового біту «0» проводять 8 разів.

7) Ось таку процедуру проводять надалі для інших байтів повідомлення чи інформаційного потоку.

8) Результати передаються у ось такій послідовності: спочатку молодший байт, а потім старший байт.

Алгоритм роботи наведено нижче. (рис. 1.2).

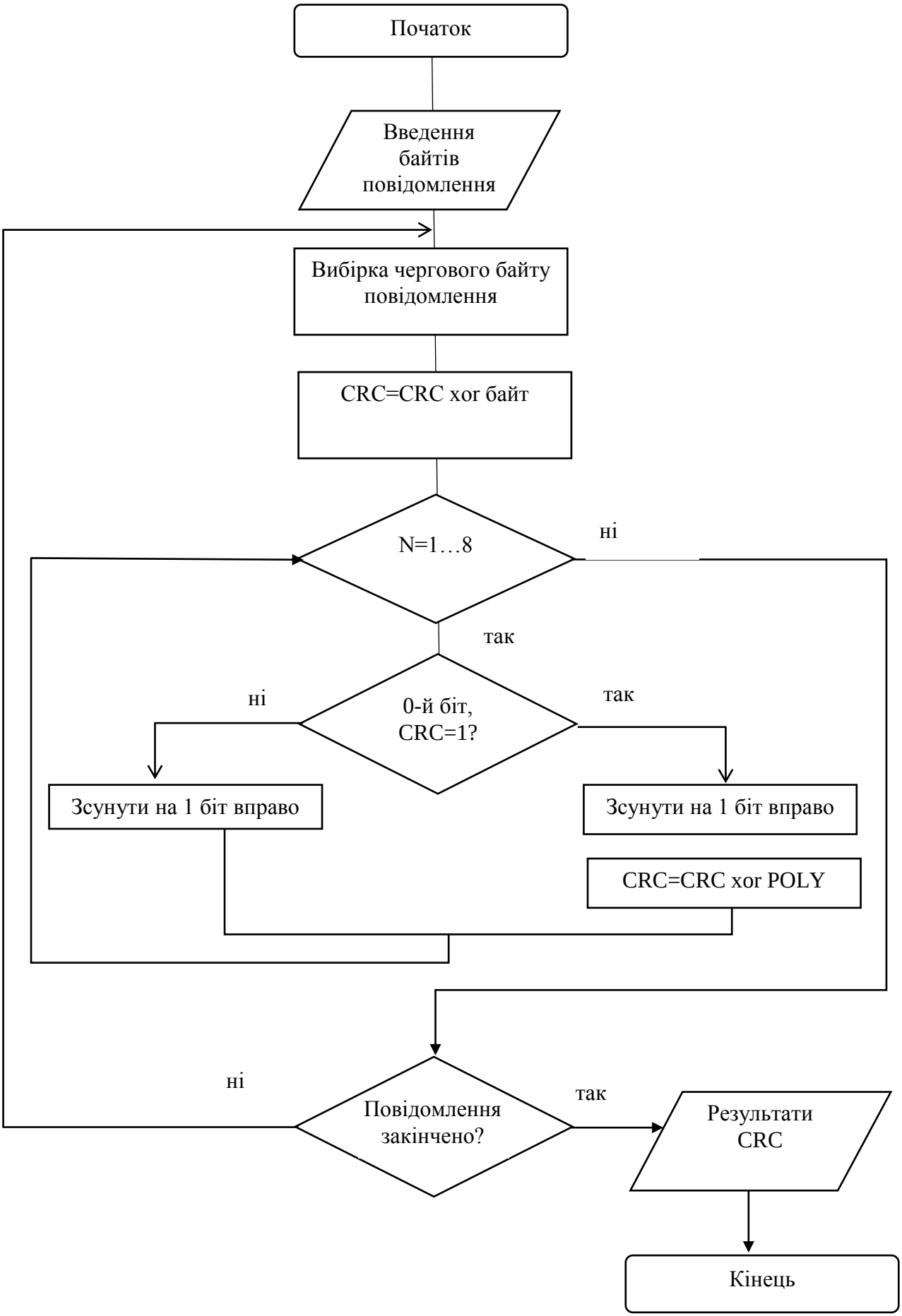


Рисунок 1.2 – Алгоритм розрахунку контрольної суми методом CRC16

1.3 Задача порівняння файлів та постановка концептуальних вимог

Багатьом користувачам, особливо, розробникам програмних засобів, сьогодні доводиться працювати не на одному, а на двох або навіть більшій кількості віддалених стаціонарних комп'ютерів (як мінімум, на робочому і домашньому) – проте на практиці це фактично означає необхідність синхронізації усіх його робочих матеріалів.

Мобільним співробітникам та віддаленим мобільним користувачам в цьому плані ще складніше, адже їм необхідний портативний пристрій, нетбук або інший мобільний комп'ютер. Таким чином, їм важливо та необхідно забезпечити синхронізацію своїх файлів ще і на їхніх мобільних пристроях, з метою уникнути проблем з версіями документів і проектів.

Зрозуміло, можна регулярно вручну (що вкрай незручно!) копіювати оновлені файли на усі комп'ютери та пристрої – робочий, домашній і мобільний. Проте це не краще рішення, оскільки операції копіювання і перезапису доведеться виконувати щодня. І такий підхід невтримно призводить до механічних помилок та великої кількості перезаписів. Набагато швидше, простіше і надійніше вдатися до застосування процесів синхронізації даних, скориставшись сучасною відповідною утилітою або онлайновим сервісом для синхронізації файлів та папок.

Відомо, що для синхронізації даних користувачі можуть застосовувати спеціалізовані утиліти та веб-сервіси, які в обох випадках відстежують вміст вказаних папок чи файлів, розташованих в різних місцях (скажімо, на двох різних віддалених пристроях чи комп'ютерах), і синхронізують ці корситувацькі дані відповідно до вибраного методу синхронізації.

Існує багато варіантів програмних засобів для синхронізації файлів. Найзручніше, якщо комп'ютери сполучені між собою безпосередньо через локальну чи корпоративну мережу, інфрачервоний порт або, в загальному, Інтернет. Тоді сам процес синхронізації даних проводиться лише за один етап, тобто, натисненням однієї (двох) кнопки у вікні відповідної програми.

У випадку, коли такого безпосереднього з'єднання не існує, то дані можуть бути синхронізованими за допомогою, так званого, пристрою-посередника, який застосовується для перенесення інформації між двома віддаленими комп'ютерами. В ролі такого пристрою може виступати флеш-носій, резервний чи зовнішній жорсткий диск, папка на ftp-сервері і так далі. У цьому випадку дані синхронізуються у декілька

етапів: 1) спочатку з одного комп'ютера файли запаковуються і 2) відсилаються на пристрій-посередник, після цього 3) на іншому комп'ютері ці дані приймаються.

Все це є справедливим у випадку користування утилітами. Стосовно веб-сервісів, варто відзначити, що під час процесів синхронізації не вимагається безпосереднього з'єднання між собою цільових пристроїв, що синхронізуються, оскільки вибрані папки на комп'ютері або ноутбуці синхронізуються із саме тими призначеними для користувача даними, що знаходяться в онлайн-сховищі.

Сформуємо концептуальні вимоги до методу порівняння з точки зору синхронізації даних:

1) Оскільки синхронізувати дані доводиться регулярно (як правило, щодня), то даний процес зручніше автоматизувати.

2) Процес автомазації процесу синхронізації даних варто проводити як наслідок аналізу. Тобто, наприклад: виконувати поцес синхронізації файлів за встановленим розкладом або при настанні певних подій:

а) при підключенні знімного диска,

б) запуску системи,

в) у випадку появи оновлень в папках, що синхронізуються.

3) За необхідності, під час обробки великих об'ємів інформації, частина файлів в процесі синхронізації розумно ігнорувати (що скоротить час, потрібний на обробку даних):

а) виключити системні файли,

б) виключити приховані файли.

1.4 Огяд програмних засобів для порівняння та подальшої синхронізації файлів

Для задоволення потреб користувачів на ринку сучасних програмних засобів для порівняння та синхронізації файлів та каталогів є досить багато утиліт та інших програм для виконання цих завднь. Цілком очевидно, що серед них є платні та безкоштовні програми, проте варто відзначити, що можна виділити цілий ряд безкоштовних продуктів з достатнім функціоналом та навіть досить високими (як для задоволення потреб рядового користувача) функціональністями. Як приклади таких подібних утиліт розглянемо тут такі програми, як Viceversa, Goodsync, Allway Sync і Freefilesync.

Ці спеціалізовані утиліти забезпечують процеси порівняння та синхронізації файлів та папок дуже швидко і є зручними у вживанні, оскільки найчастіше тут достатньо один раз налаштувати необхідні параметри для операцій та автоматизувати дані процеси – і тоді надалі програмні засоби самостійно відстежуватимуть ситуацію з відслідковування процесів порівнянн та синхронізації.

Проведення процесу синхронізації, як правило, складнощів не викликає. Спочатку створюється нове завдання на синхронізування, потім задають дві папки – початкова і цільова, і (за необхідності) визначають умови фільтрації файлів. Звісно, що можна все без розбору продублювати з однієї папки в іншу. І задача буде виконаною. А що ж робити в умовах обмеженості ресурсів? Раціональніше активізувати процес аналізу вмісту папок та файлів, тобто, запустити проце с порівняння файлів, клацнувши по кнопці Аналіз. Результат цих даних у вихідній і цільовій папках відображується на моніторі із вказанням нових, змінених чи видалених файлів (рис. 1.3).

Рисунок 1.3 - Результат аналізу в GoodSync

Потім можна мануально запустити синхронізацію файлів (кнопка Синхронізація), однак значно зручніше налаштувати програму на автоматичне виконання обохцих процесів.

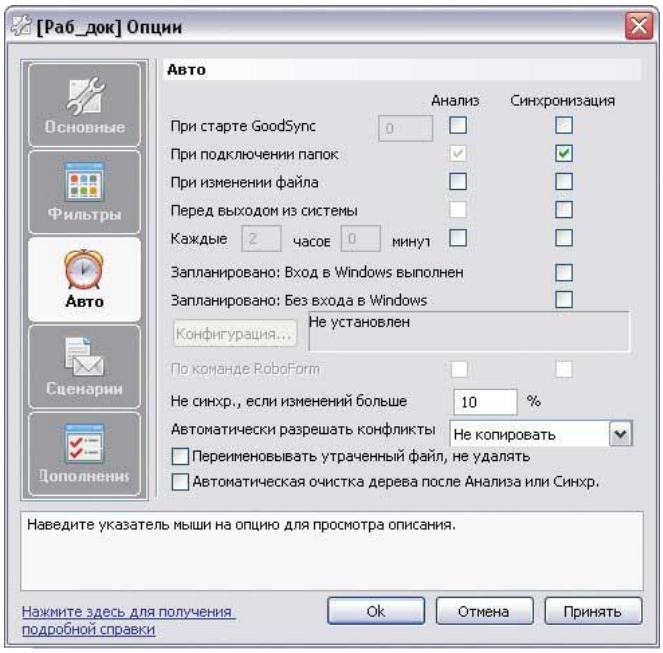


Рисунок 1.4 – Налаштування автоматичної синхронізації в GoodSync

З цією метою користувачі відкривають вбудований в утиліту планувальник і наперед визначають час процесу синхронізації або подію, при настанні якого програма повинна запускати процеси порівняння і синхронізацію файлів (див. рис. 1.4).

Час автоматичного процесу синхронізації файлів на портативний накопичувач (скажімо, флеш-накопичувач) може виникнути проблема розпізнавання носія, як диска. Для того, аби програма могла дійсно правильно розпізнати потрібний носій та диск, потрібно вручну змінити шлях до пристрою, замінивши там букву імені диска на мітку тому (=VolumeName:\folder1\folder2 – рис. 1.5). Відповідну таку мітку тому для конкретного якогось диска нескладно налаштувати у властивостях, скориставшись можливостями ОС Windows. Вживання вищевказаних налаштувань гарантує виявлення потрібного диска чи накопичувача незалежно від привласненого йому імені.

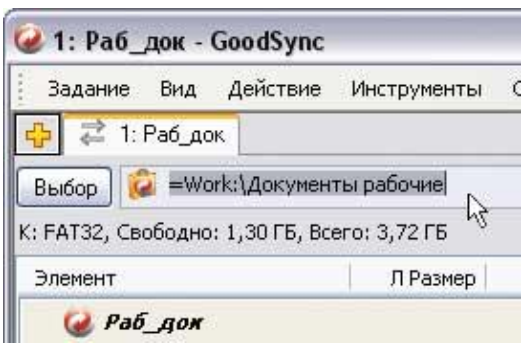


Рисунок 1.5 – Заміна букви диска міткою тому в GoodSync

Розробник програмного забезпечення компанія TGRMN Software випустила свій аналог програми порівняння та синхронізації файлів та катлогів під назвою ViceVersa. Сайт програми: http://www.tgrmn.com/. Розмір дистрибутива складає Pro – 3,4 Мбайт, Plus – 1,1 Мбайт. а Free – 708 Кбайт. Працює під керуванням: Viceversa Pro 2.5 і Viceversa Plus 2.4.2 – для ОС Windows (всі версії цієї ОС); а також, версія Viceversa Free 1.0.5 – для ОС Windows Xp/vista/7. Програма платна, її ціна така: для Pro – 59,95 дол.; для Plus – 34,95 дол.; і є також версія Free – безкоштовно

Viceversa Pro – це спеціалізована програма для порівняння та синхронізації файлів, а також для резервного копіювання і реплікації файлів і папок (рис. 1.6). З її допомогою користувач може виконувати процеси синхронізації даних між, скажімо, стаціонарними комп'ютерами, чи, наприклад, ноутбуками, або навіть файловими серверами чи якимись зовнішніми носіями. Реалізується функціонал по локальній мережі, через мережу інтернет, і, звісно, із застосуванням будь-яких зовнішніх дисків чи накопичувачів.

Рисунок 1.6 – Зовнішній вигляд Viceversa Pro

Підчас процесів порівняння та синхронізації файлів аналізуються такі параметри, як:

- розмір файлу;
- час створення файлів;
- дата створення файлів;
- контрольні суми;
- сукупність перерахованих параметрів.

Тут розробниками передбачена можливість включення та/або деяких даних підчас аналізу, скажімо, підкаталогів, а також навіть окремих файлів з врахуванням усіх їх атрибутів і масок (якщо необхідно). Допускається також і процеси синхронізації і резервного копіювання відкритих та/або заблокованих деякими додатками файлів, включно із поштовими базами Outlook і Outlook Express, документами Word, Excel і БД SQL.

Процеси порівняння та синхронізації даних відбувається вручну на вимогу або ж можуть бути виконані в автоматичному режимі – за розкладом (скажімо, щодня в якийсь певний час). З метою економії дискового простору і забезпечення захисту інформації на будь-якому носієві в даній програмі передбачений інструментарій для архівування та додаткового шифрування файлів (на вимогу).

Запропонована утиліта випускається в таких трьох редакціях:

- безкоштовною Free (http://www.tgrmn.com/free/);
- комерційний базовий Plus;
- комерційний розширеною Pro.

Можливості безкоштовної редакції програмного засобу обмежені процесами порівняння і синхронізації файлів в визначених папаках між різними накопичувачами на дисках різних типів, тоді синхронізація виробляється мануально. Редакція програми Plus дозволяє працювати з портом usb, жорсткими і мережевими дисками, а також приводом Dvd/cd, і це забезпечує можливість порівняння, синхронізації, резервування певних відкритих чи заблокованих файлів або папок і, таким чином, може бути налаштована на роботу за визначеним розкладом. У редакції Pro підтримується весь заявлений розробниками функціонал.

Розробник Siber Systems, Inc випустив свою програму під назвою Goodsync. Сайт програми знаходиться тут: http://www.goodsync.com/. Розмір дистрибутива програмного засобу такий: 7,15 Мбайт. Працює під керуванням: ОС Windows 2000/XP/Vista/7. Програма платна, її ціна: 29,95 дол.

Програмний засіб Goodsync – це зручний і простий інструмент для виконання процесів порівняння файлів із подальшим застосуванням синхронізації і резервного їх копіювання (за потреби) (рис. 1.7).

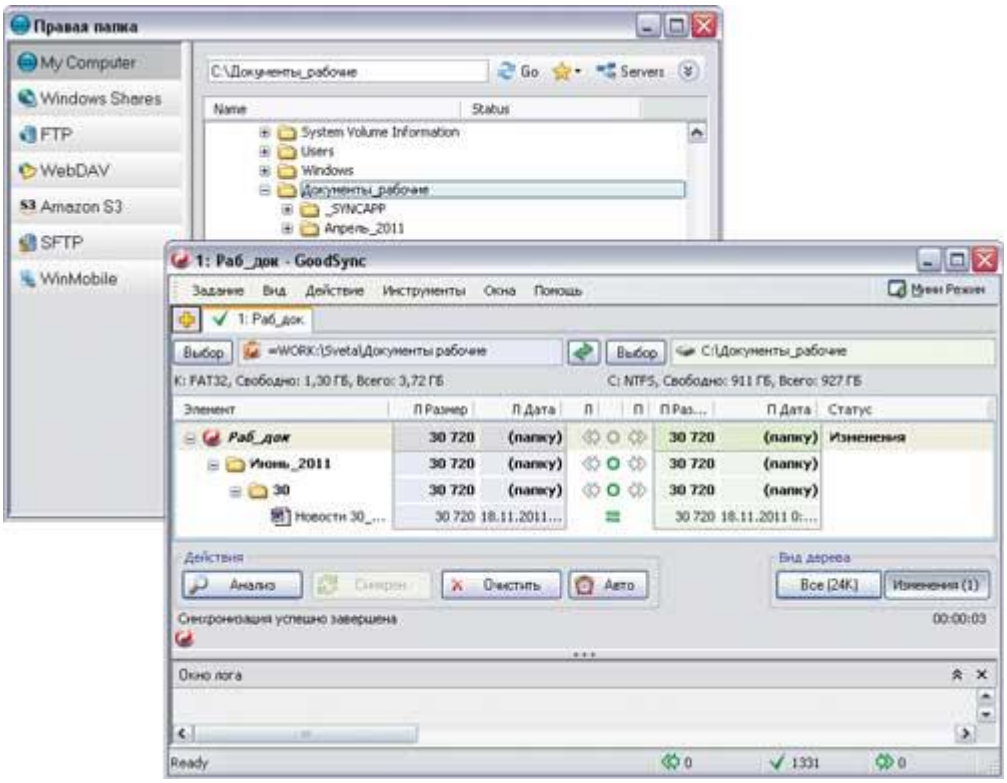


Рисунок 1.7 – Інтерфейс Goodsync

Програмний засіб дозволяє користувачам синхронізувати свої файли між звичайними і портативними комп'ютерами, зовнішніми дисками і серверами, дозволяє також виконувати резервне копіювання важливих користувацьких даних на різні носії (включно з FTP- і webdav-серверами). Розробниками, також, передбачена додаткова можливість порівнняння та синхронізації файлів між пристроями під керуванням ОС Windows Mobile Phone або Pocket PC (Windows CE) і звичайним настільним комп'ютером. Такі процеси порівняння та синхронізації можуть відбуватися безпосередньо між комп'ютерами (наприклад, у локальній мережі або через мережу інтернет з FTP-, WEBDAV- і Secure ftp-серверів тощо) або з підключенням до будь-яких зовнішніх накопичувачів (Usb-накопичувача, зовнішнього HDD-диску).

Аналіз користувацьких даних проводиться з врахуванням дати, часу модифікації файлів чи папок, їх розміру. Під час процесу порівняння файлів програмним засобом автоматично ігноруються приховані і системні файли, також можна налаштувати включення чи виключення деяких файлів з певними іменами, відповідно масці, а також, це застосовно до файлів певного розміру або навіть з певним часом зміни.

Тут передбачена можлива синхронізація заблокованих файлів із застосуванням спеціальної служби Volume Shadow Copy. З метою автоматизації процесів порівняння та синхронізації файлів та папок, сюди включений інструментарій для запуску

синхронізації за визначеним розкладом, а також при настанні визначених наперед певних подій:

- скажімо, при самому підключенні комп'ютера до локальної мережі,
- або під час підключенні зовнішнього диска до комп'ютера,
- або ж спрацьовувати цей механізм може під час запуску ОС.

Тут застосовується планувальник ОС Windows. З метою підвищення безпеки під час віддаленого аналізу файлів чи папок, додаткової синхронізації даних, тут реалізована передача файлів по зашифрованому каналу передачі даних (FTP через SSH і WEBDAV через SSL), а під час резервного копіювання – можливе застосування шифрованої файлової системи EFS (Encrypting File System).

Програма має демо, яка повністю функціональна 30 днів. Потім вона може використовуватися домашніми користувачами і некомерційними організаціями безкоштовно, але з такими обмеженнями – дозволяється створювати до 3 завдань по синхронізації, що містять не більше 100 файлів.

Розробником є організація Botkind, Inc. Сайт програми тут: http://allwaysync.com/. Розмір дистрибутива: 6,9 Мбайт. Працює під керівництвом: ОС Windows 2000/XP/2003/Vista/2008/7. Ціна програми: залежить від ліцензії: є Pro – 29,99 долл.; надано версію Free – безкоштовно (тільки для некомерційного використання).

Allway Sync – досить проста у вживанні спеціалізована утиліта, що призначена для порівняння файлів з метою їх подальшої синхронізації і резервування (рис. 1.8). Програма Allway Sync забезпечує синхронізацію файлів та папок між настольними комп’ютерами, портативними ноутбуками, також зовнішніми жорсткими дисками, також usb-носіями, звісно, що підтримує Ftp/sftp-сервери і різні онлайн сховища даних. Аналіз інформації і її оновлення виконуються по локальній мережі, через мережу інтернет і за допомогою зовнішніх накопичувачів (як то флешки, зовнішні жорсткі диски тощо).

Рисунок 1.8 – Інтерфейс Allway Sync

У програмі Allway Sync останні версії файлів, що обробляються, виявляються на основі комбінації атрибутів файлу, його розміру і часу його створення. З метою зменшення переліку аналізованих файлів програма може дозволяти включення та/або виключення об'єктів синхронізації (файлів, папок) із врахуванням місця знаходження файлу, його імені і атрибутів (лише включення та/або виключення прихованих чи системних файлів). Процеси порівняння та синхронізації можуть проводитися за вимогою і, звісно, автоматично – це коли через певний проміжок часу, під час підключення зовнішнього пристрою тощо можливим стає використання спеціалізованого планувальника задач ОС Windows.

Програма представлена в двох таких редакціях: безкоштовна Free і комерційна Pro. Безкоштовна редакція дозволяє порівнювати та синхронізувати не більше 40 тис. файлів в 30-денний термін. Також є спеціальна портативна редакція утиліти, призначена для установки на флеш-носій або зовнішній HDD-диск.

Розробник програм компанія Zenju представив Freefilesyn. Сайт програми є тут: http://freefilesync.sourceforge.net/. Розмір дистрибутива складає 9,27 Мбайтів. Працює під керуванням ОС Windows 2000/XP/Vista/7 Ціна Freefilesyn: безкоштовно.

Програмний засіб Freefilesync – це безкоштовна утиліта, призначена для виконання процесів порівняння і синхронізації файлів між різними типами комп'ютерів і зовнішніх дисків (рис. 1.9).

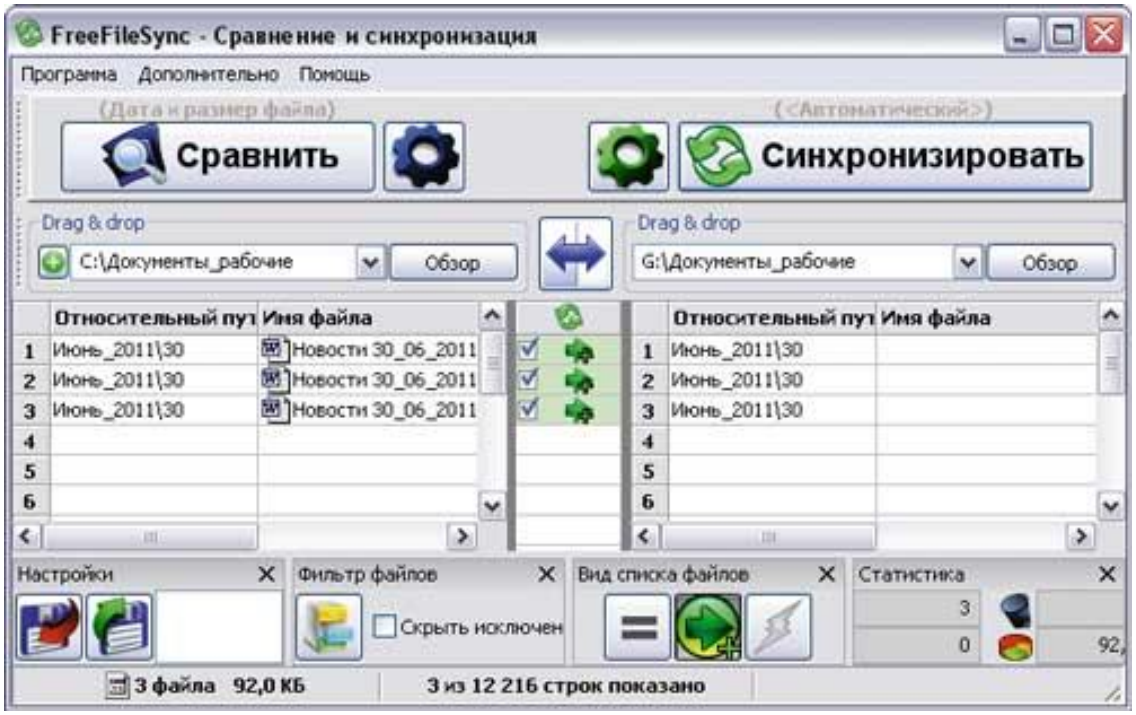


Рисунок 1.9 – Інтерфейс Freefilesync

Тут аналіз файлів проводиться з врахуванням дати їх створення і їх розміру. Під час процесів порівнянні даних ігноруються каталоги типу «\recycler» і типу «\ System Volume Information». У Freefilesyn передбачена вбудована можливість включення та/або виключення окремих файлів, типів файлів фз врахуванням їх дати створення, їх розміру і їх імен. Можливе копіювання наперед заблокованих файлів із застосуванням операційного ПЗ Windows Volume Shadow Copy Service. У Freefilesyn є інструментарій для створення пакетних завдань, причому, запуск яких можна автоматизувати через планувальник ОС Windows.Утиліта Freefilesyn поширюється за ліцензією GNU GPL, а її встановдлення можлива в 2 варіантах: стаціонарному і портативному.

1.5 Огляд мережевих сервісів для порівняння та синхронізації файлів

Сервіси, які призначені для процесів порівняння та синхронізації файлів, в світовій мережі є чимало. Частина з них презентуються як онлайнові сховища даних з можливістю проведення процесів синхронізації, інші ж призначені спеціально для порівняння файлів, а потім вже – для їх подальшої синхронізації. У даній роботі автор зупинивсяя на двох найбільш популярних сервісах для порівняння та синхронізації

файлів – Sugarsync і Dropbox, які за своїми можливостями значно випереджають багатьох конкурентів.

На відміну від утиліт, що розглянуті вище, сервіси вимагають значно більшого обсягу часу для аналізу (процес порівняння) даних і їх синхронізацію. Така різниця в часі, звичайно, є відносною і визначається досить конкретними умовами роботи в мережі Інтернет, а також, об'ємом інформації, що порівнюється та синхронізується, – тому зрозуміло, при низькій швидкості мережевого з'єднання така об’ємна операція може виконуватися досить довго. Ось чому даний спосіб є цікавим та дієвим лише у випадку наявності постійного високошвидкісного підключення до Інтернету.

Варто сказати, що сервіси володіють усім необхідним спеціалізованим функціоналом для доступу до користувацьких документів з самих різних пристроїв, які мають доступ до мережі в будь-якому місці (дім, в офіс, у відрядження тощо) і у будь-який час, що дуже актуально для мобільних і віддалених користувачів. Більш того, за певної необхідності можна дістати доступ до своїх документів навіть (!) з чужого пристрою, оскільки проаналізовані файли не лише синхронізуються на усі вказані користувачем додаткові пристрої, але і резервуються в онлайн сховищі. До того ж, вживання таких сервісів значно спрощує обмін даними. Це означає, що можна без особливих зусиль обмінюватися файлами, документами, даними з іншими співробітниками компанії, родичами, друзями тощо, з людьми, які працюють над тим же проектом, а також медіа-даними і іншими матеріалами.

Також, звернімо увагу, що для процесу веб-синхронізації даних зовсім не потрібна одночасна наявність в мережі Інтернет усіх пристроїв, що синхронізуються, одночасно, тому що в якості посередника використовуються спеціальні сервери. І тоді – процес синхронізації кожного з визначених користувачем комп'ютерів чи пристроїв проводиться під час або після підключення їх до Інтернету.

У технічному розрізі жодних труднощів під час вживання цих сервісів не виникає. Спочатку на деякому (обирає користувач) ресурсі потрібно завести свій аккаунт, потім викачати програму-клієнт і встановити її на кожному комп'ютері, на якому потрібно порівнювати, а потім і синхронізувати дані (з вказівкою одного і того ж логіну та паролю, звісно). Потім потрібно налаштувати усі параметри процесів синхронізації; у деяких таких сервісів це може виконуватися безпосередньо в ході встановлення сервісного клієнтського ПЗ. Під налаштуванням автором розуміється

деяка вказівка на папки, які надалі потрібно буде порівняти та синхронізувати між декількома пристроями, ї, можливо, там потрібно буде визначити додаткові параметри. Скажімо, в сервісі Sugarsync потрібно вибрати ім'я і іконку для швидкої ідентифікації віддаленого комп'ютера і тільки потім вказати необхідні теки (рис. 1.10). Після цього виконується завантаження файлів і даних на видалений сервер – тобто, процеси порівняння і синхронізації файлів.

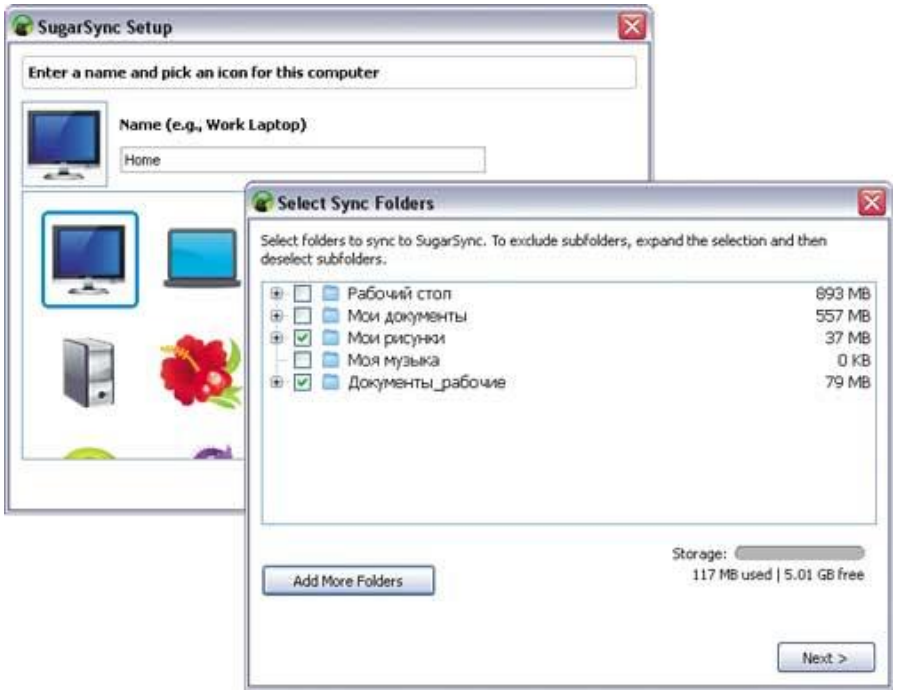


Рисунок 1.10 – Налаштування кліента в SugarSync

Подальші дії користувача напряму залежать від вибраного ним онлайн сервісу. Наприклад, на сервісі Sugarsync потрібно додатково вказати через спеціальний модуль Manage Sync Folders, між якими пристроями повинні виконуватися порівняння та синхронізація файлів (рис. 1.11). У сервісі Dropbox необхідність в такій операції відпадає, але тут доведеться регулярно копіювати дані, що порівнюються та синхронізуються, в папку Мої документи\dropbox (вона на комп'ютері користувача створюється автоматом під час встановлення клієнта). Після проведення налаштувань вміст папок (вказаних користувачем в випадку сервісу Sugarsync і папки Dropbox при використанні цього сервісу) автоматичнопорівнюватиметься та синхронізуватиметься із вказаними відповідними онлайновими серверами в обидві сторони через мережу інтернет.

Рисунок 1.11 – Визначення папок, що синхронізуються в Sugarsync

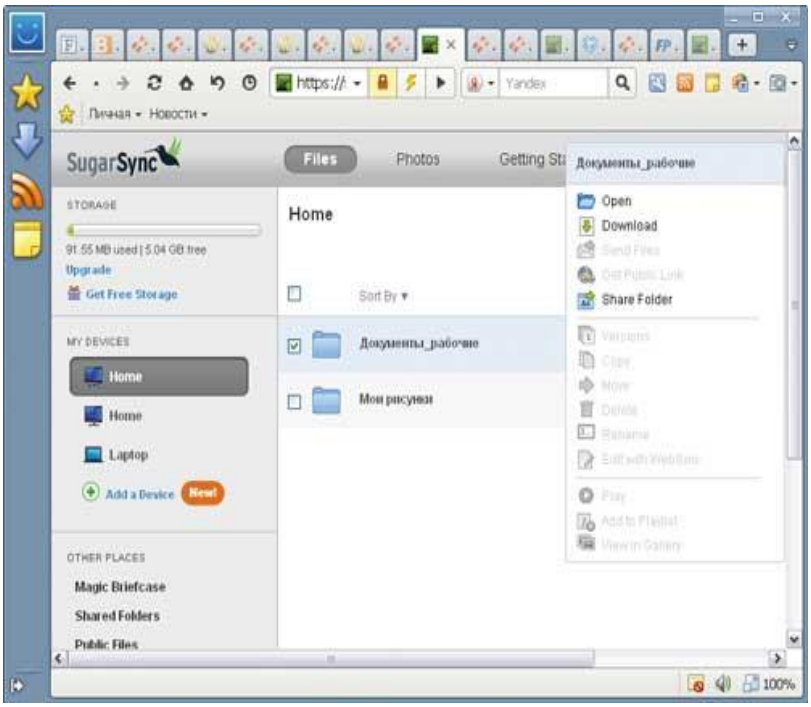


Рисунок 1.12 – Доступ в онлайн-сховище Sugarsync через веб-сервер-інтерфейс

Таким чином, усі пристрої, що синхронізуються в мережі, матимуть останні версії проаналізованих файлів. За необхідності дістати доступ до них можна буде не лише з пристроїв, що вказані, а й через онлайн-сховище (див. рис. 1.12).

Обидва вищеназвані автором сервіси можуть застосовуватися для автоматичного порівняння та синхронізації різних персональних даних і, таким чином, дозволяють синхронізувати інформацію між двома і більшою кількістю комп'ютерів і усілякими мобільними пристроями.

1.6 Аналіз отриманих даних

Нагадаємо сформовані концептуальні вимоги до процесу порівняння файлів з точки зору подальшої можливої синхронізації даних і занесемо дані у таблицю 1.1:

1) Оскільки синхронізувати дані доводиться регулярно (як правило, щодня), то даний процес зручніше автоматизувати.

2) Процес автомазації процесу синхронізації даних варто проводити як наслідок аналізу. Тобто, наприклад: виконувати поцес синхронізації файлів за встановленим розкладом або при настанні певних подій:

а) при підключенні знімного диска,

б) запуску системи.

3) За необхідності, під час обробки великих об'ємів інформації, частина файлів в процесі синхронізації розумно ігнорувати (що скоротить час, потрібний на обробку даних):

а) виключити системні файли,

б) виключити приховані файли.

Таблиця 1.1 – Характеристики програмних засобів для порівняння файлів

	Можливість порівняння внаслідок підключенні носія	Можливість порівняння внаслідок запуску системи	Можливість порівняння виявлення внаслідок оновлень	Можливість виключення системних даних	Можливість виключення прихованих даних	Можливість блочних замін у файлі	Застосування хеш-даних для операції порівняння файлів
Goodsync	+	+	+	-	-	-	-
Viceversa	+	+	+	+	-	-	-
Allway Sync	+	+	-	+	+	-	-
Freefilesync	+	+	+-	+	+	-	+-
Sugarsync	+	+	+-	+-	+	+-	+-
Dropbox	+	+	+	+	+	-	+-
Нова розробка	+	+	+	+	+	+	+

За можливістю порівняння внаслідок підключенні носія всі програмні засоби є дієвими та мають відмінні характеристики. За можливістю порівняння внаслідок запуску системи всі програмні засоби є дієвими та мають відмінні характеристики. За можливістю порівняння внаслідок виявлення оновлень одна з утиліт, а саме, Allway Sync, не задовільняє вимогам, а два програмних засоби, а саме, Freefilesync та Sugarsync виконують цю функцію не недостатньому рівні, або такому, що не відповідає вимогам (чи не передбачено у функціоналі). За параметром «Можливість виключення системних даних» – утиліта Goodsync не має цього функціоналу, а сервіс Sugarsync – виконує це за додаткових умов. За параметром «Можливість виключення прихованих даних» - утиліти Goodsync та Viceversa не мають такого функціоналу, а інші програмні засоби на достатньому рівні виконують цю задачу. За показником «Можливість блочних замін у файлі» - усі програмні засоби не мають такого

функціоналу крім нової розробки та частково – у Sugarsync, де такий підхід можна створити мануально, проте точково. За параметром «Застосування хеш-даних для операції порівняння файлів» - усі утиліти, крім частково Freefilesync, не мають цього функціоналу, а серед сервісів, що розглянуті тут – ця функціональність реалізована закритим методом.

Оцінимо числовими значеннями ці показники та занесемо дані у таблицю 1.2 та побудуємо діаграму 1.13.

Таблиця 1.2 – Оцінка характеристик

	Можливість порівняння внаслідок підключенні носія	Можливість порівняння внаслідок запуску системи	Можливість порівняння виявлення внаслідок оновлень	Можливість виключення системних даних	Можливість виключення прихованих даних	Можливість блочних замін у файлі	Застосування хеш-даних порівняння дляоперації
Goodsync	10	9	9	2	3	1	1
Viceversa	10	9	10	8	3	1	1
Allway Sync	9	9	2	8	8	1	1
Freefilesync	8	10	5	9	8	1	4
Sugarsync	10	10	6	7	8	5	6
Dropbox	10	10	9	9	9	1	7
Нова розробка	10	10	9	10	9	8	9

1.7 Висновок за розділом

Керування передаванням інформації та контроль достовірності даних є однією з ключових задач сучасного комп’ютерного діловодства. Це пояснює та водночас, локалізує проблему синхронізації даних до задачі порівняння файлів та папок.

Розглянуто 2 варіанти автоматичної синхронізації файлів та папок – за допомогою спеціалізованих синхронізуючих утиліт і за допомогою онлайн сервісів порівняння та подальшої синхронізації даних. Вибір найбільш вдалого рішення з наведених варіантів залишається за користувачем. Утиліти є більш кращими з позиції швидкості і можливості тонкого налаштування параметрів процесів порівняння та синхронізації, включно з обробленням чітко вибраних типів даних. Водночас, сервіси

відмінно забезпечують доступ до актуальних версій файлів та папок фактично з будь-якого пристрою, що має доступ до мережі.

У даній роботі розглянуто спосіб порівняння файлів за допомогою методу CRC16 і далі планується розробити програмне забезпечення із застосуванням цього методу вкупі з механізмом блочного порівняння файлів.

2 МЕТОД ТА ПРОГРАМНА РЕАЛІЗАЦІЯ БЛОЧНОГО ПОРІВНЯННЯ ФАЙЛІВ

У цьому розділі описаний метод блочного порівняння файлів, заглаьний принцип його роботи та технологічний ланцюжок, що підтримує даний метод. Також, тут наданий опис ключових процесів методу блочного порівняння файлів та детальний опис програмної реалізації методу блочного порівняння файлів, а також, тестування розробленого програмного засобу.

2.1 Загальна схема методу блочного порівняння файлів

Порівняння файлів широко застосовується в індустрії програмування, де ведеться спільна робота і необхідно часто відстежувати зміни, які вносять в робочі файли різні користувачі, тому швидкість при синхронізації каталогів (скажімо) є загальною потребою – щоб протягом порівняння не були внесені зміни. Алгоритм роботи методу CRC 16 дозволяє звести порівняння до порівняння так званих хеш-кодів, які вираховуються як строкове представлення контрольних кодів для блоків, на які поділяється зміст файлу.

Загальний принцип роботи методу блочного порівняння файлів полягає у тому, що вирішення задачі порівняння файлів призводить до подальшої синхронізації цих файлів оптимальним чином.

В результаті, зменшується час на порівняння файлів та їх синхронізацію за рахунок передачі по мережі тільки змінених блоків данх.

У загальному, опис технологічного ланцюжка, що супроводжує даний метод, є таким. На вхід аналізатора А приймається файл 1, що призначений для порівняння із подальшою можливою його синхронізацією (за потреби). Потім відбувається процес аналізу файлу 1 на цілісність, тобто, визначається, чи був він змінений чи ні (швидкий аналіз із синхронізованим на той момент часу файлом 2, який знаходиться на віддаленому ресурсі). У випадку, якщо знайдені відмінності, приймається рішення про запуск процесу розбиття файлу на блоки визначеної довжини, далі – процес перевірки хеш-кодів кожного з блоків, виявлення тих, що були змінені і передача у мережу (запуск процесу синхронізації) лише тих блоків, що мають зміни.

Схематичне представлення роботи методу блочного порівнння файлів наведено на рисунку 2.1 [20].

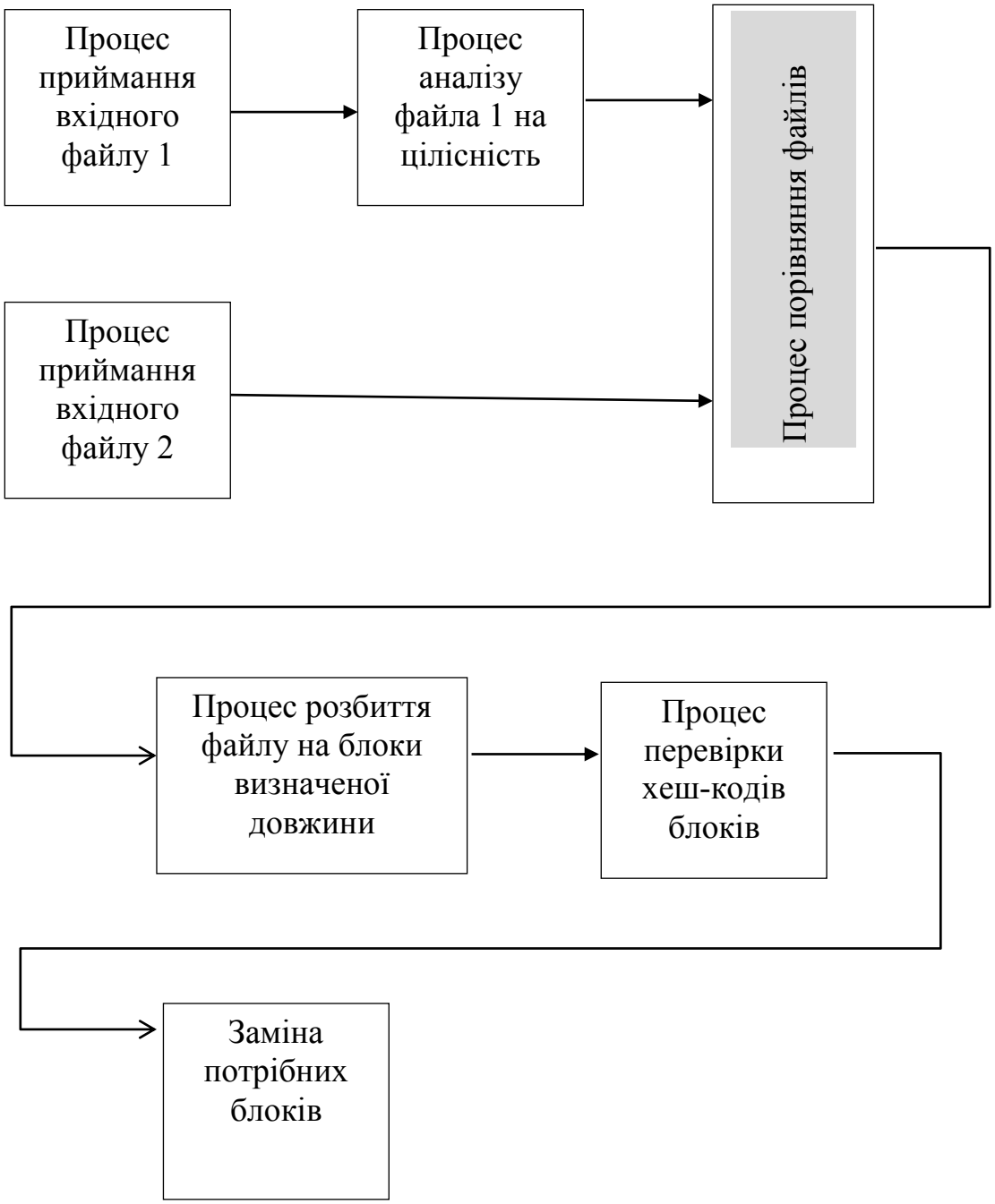


Рисунок 2.1 – Схематичне представлення роботи методу блочного порівнння файлів

Перелік процесів, що супроводжують метод блочного порівняння файлів наведено тут:

1) Процес приймання вхідного файлу 1
2) Процес аналізу файла 1 на цілісність
3) Процес приймання вхідного файлу 2
4) Процес порівняння файлів
5) Процес розбиття файлу на блоки визначеної довжини

6) Процес перевірки хеш-кодів блоків
7) Заміна потрібних блоків

2.2 Процес порівняння файлів

Предметом роботи даної програми є виконання процесу порівняння двох файлів та/або каталогів за допомогою застосування можливостей хеш-кодів із використанням методу CRC 16, який дозволяє нам відстежувати зміни у вмісті (відслідковувати модифікацію) файлу без витрат на рядкове чи двійкове порівняння. Запропонований у даній роботі метод блочного порівняння файлів дозволяє виконувати швидке порівняння достатньо великих обсягів даних [8].

Процес порівняння файлів широко застосовується в індустрії програмування, де ведеться спільна робота і необхідно часто відстежувати зміни, які вносять в робочі файли різні користувачі, тому швидкість при синхронізації і окремих файлів і цілих каталогів є нагальною потребою – потрібна швидкодія, щоб протягом самого процесу порівняння не були внесені зміни [9-10].

Розглянутий вище алгоритм роботи методу CRC 16 дозволяє звести процес порівняння файлів до порівняння, так званих, хеш-кодів, які вираховуються як строкове представлення контрольних кодів для блоків, на які поділяється вміст файлу.

Таким чином, процес порівняння двох файлів зводиться до таких кроків:

- Порівняння імен;
- Порівняння дати модифікації і розміру;
- Підрахунок хеш-кодів за методом CRC 16;
- Порівняння хеш-кодів за методом CRC 16.

Схематичне представлення роботи процесу порівнння файлів наведено на рисунку 2.2 [20].

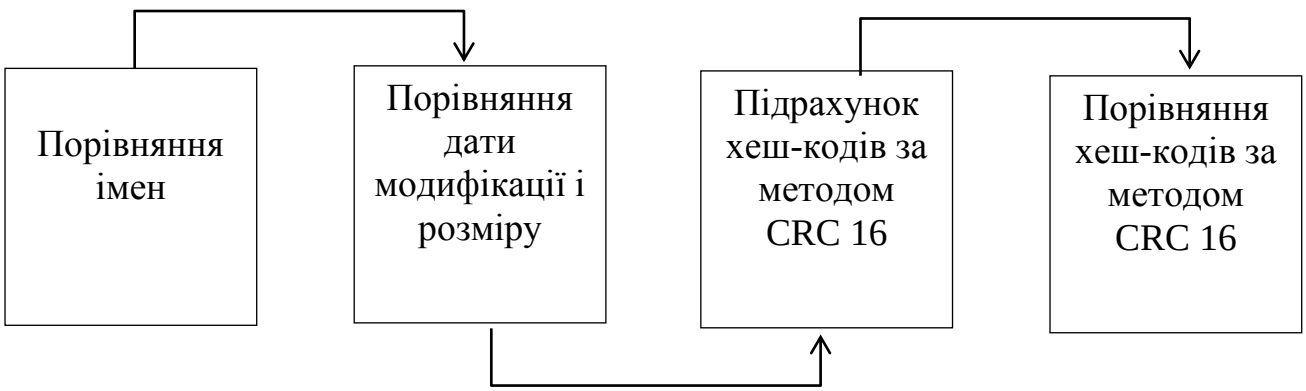


Рисунок 2.2 – Схематичне представлення роботи процесу порівнння файлів

Перевагою запропонованого процесу порівнння файлів є те, що він виконуватиметься з однаковою швидкістю для файлів будь-якого розміру, адже хеш-код має фіксовану довжину.

Це дозволяє інтегрувати в програму кілька варіантів порівняльних функцій:

- індивідуальну,
- групову,
- глибинне порівняння із контролем змісту,
- визначення унікальних файлів.

Це зробить програму зручним інструментом програміста та керівника великими спільними проектами.

Модифікуємо загальну схему роботи методу блочного порівнння файлів 2.1 до розлогої схеми роботи методу блочного порівнння файлів 2.3 із урахуванням схематичного представлення роботи процесу порівнння файлів.

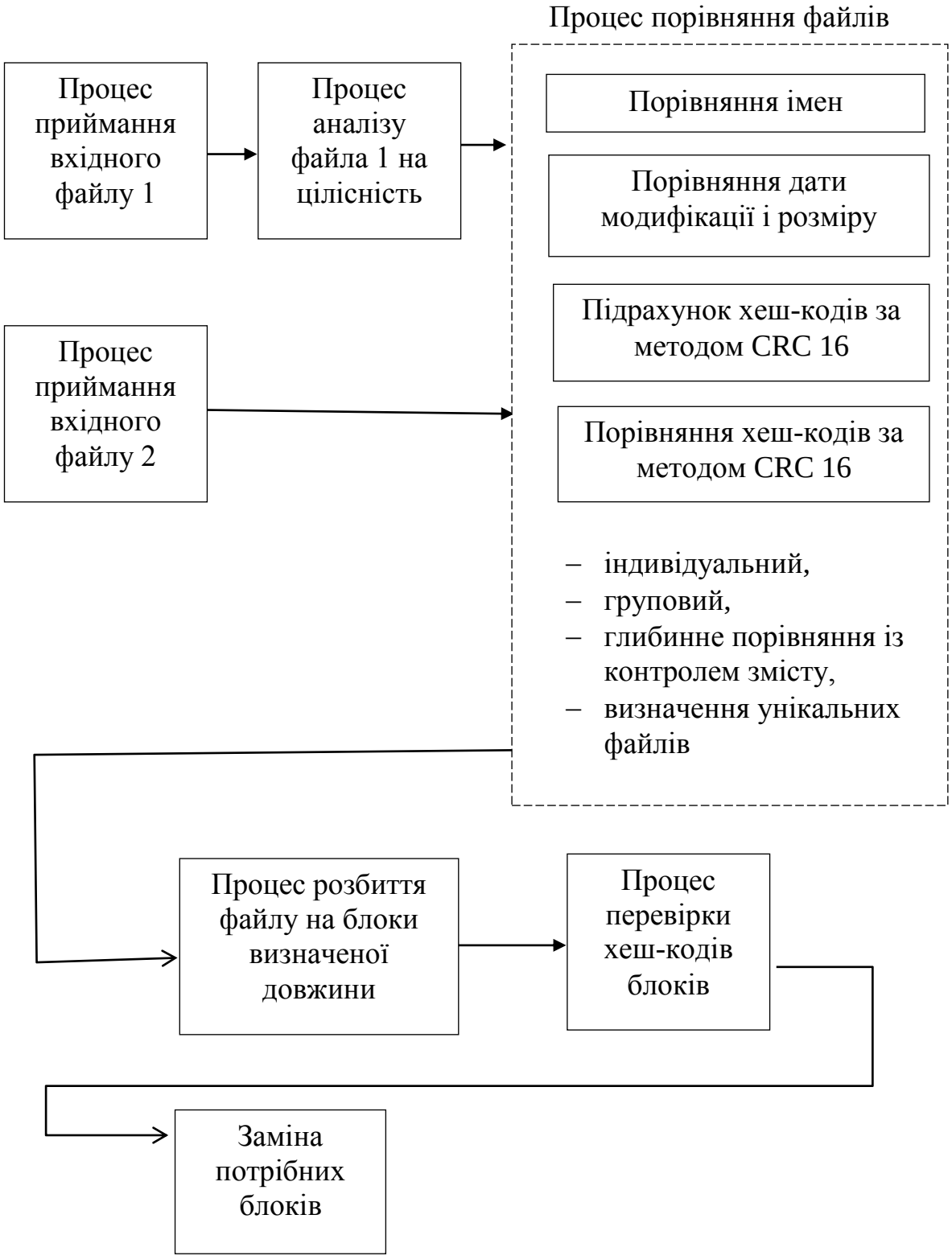


Рисунок 2.1 – Схематичне розширене представлення роботи методу блочного порівнння файлів

2.3 Процеси підрахунку та порівняння хеш-кодів за методом CRC 16

Для встановлення необхідних ресурсів програим визначимо ресурси, що необхідні для роботи процесу порівнянн файлів, тобто, відстежування змін, опис якого було наведено в попередньому підрозділі, з метою продемонструвати його реалізацію механізмами та стандартними компонентами мови програмування Java. Далі розглянемо рішення для розробки, проектування та безпосередньо створення користувацького інтерфейсу.

Оскільки на цьому етапі немає необхідності перевіряти цілісність даних, а лише визначити, чи були внесені в файл зміни, тож достатньо скористатись відомим методом CRC 16. Різний контент, вміст файлу дасть різні коди під час використання цього підходу.

Короткий екскурс в алгоритм CRC 16 покаже, що він був розроблений в 1991 році професором та науковцем Рональдом Л. Рівестом. Його помилково часто називають алгоритмом шифрування, але насправді це твердження не є дійсним. Наприклад, ключовою ознакою MD5 є те, що зашифровані дані за алгоритмом роботи відновити не вдасться. Він лише показує, чи є цілісним пакет даних, або в нього були внесені зміни – наприклад, за рахунок помилок передачі даних по каналу зв’язку, або зловмисником. Тому MD5 (Message Digest 5) правильно називати хеш-функцією.

Функція хешування, скажімо, H є відображенням, на вхід якої подається повідомлення, назвемо його M, довільної довжини, а на виході, відповідно, виходить значення h кінцевої довжини, де (2.1):

$$h = H(M). \qquad (2.1)$$

У довільному випадку хеш-значення довжини h набагато менше вихідного повідомлення M. Таким чином, у випадку застосування MD5 довжина складе h=128 біт. Хеш-функція тоді повинна задовольняти таким поставленим умовам:

1) За достатньо великим повідомленням M наша хеш-функція має швидко обчислити значення довжини h за методом CRC 16, і це значеня довжини має цілком залежати від кожного біта повідомлення M;

2) Незоротність: за значенням довжини h неможливо відновити вихідний текст повідомлення M;

3) Математично та операційно дуже важко (фактично майже неможливо) знайти два повідомлення M і M1, які дають два однакових h-значення довжини.

Відомо, що хеш-значення за методом CRC 16 є контрольною сумою вихідного повідомлення M і його називають в термінології MDC (англ. Manipulation Detection Code – тобто, код виявлення змін) або ще використовують назву MI (Message Integrity Check – тобто, перевірка цілісності повідомлення).

Пояснимо на прикладі деякої функції, яка кожному натуральному числу від 1 до 106 ставить у відповідність інше натуральне число від 1 до 1000.

Нехай у нас є деякий набір деяких вхідних даних. Для спрощення виконання і розуміння цієї задачі будемо розглядати натуральні числа від 1 до 106. І нехай є деяка зазаначена функція, у якій один параметр – це є натуральне число від 1 до 106, а на виході функції повертається значення, таке, що M – є натуральне число від 1 до 1000.

По великому рахунку, неважливо, що саме робить ця визначена функція, а важливо те, що вона кожному натуральному числу від 1 до 106 ставить у відповідність інше натуральне число від 1 до 1000.

Програмний код, реалізовує подібну залежність, наведено на лістингу 2.1:

```
public int hash(long int x)
{if (x%1000==0)
 return 1000;
return (x % 1000);}
```

Лістинг 2.1 – Програмний код реалізації хеш-функції за методом CRC 16

Слід зауважити, що за даним конкретим прикладом початкове число за хеш-значенням відновити, в принципі, можна. Однак результат буде досить неоднозначним. Якщо у якості хеш-значення виступає число 234, то вихідні числа можуть бути такими: 1234, 2234, 3234… і так далі…

Така властивість неоднозначності хеш-функції за методом CRC 16 накладає на алгоритм додаткову вимогу, яка є такою: алгоритм має бути достатньо складним, щоб фактично унеможливити варіант, при якому зміни в потоці вхідних даних були

такими, щоб результуюче хеш-значення було тим же, що і для коректного потоку. І ось саме тому мінімальна довжина хешу MD5 має скалдати 128 біт.

Для обробки на свій вхід MD5 одержує деякий символьний блок даних. Далі цей блок даних перетворюється в послідовність із нулів «0» і одиниць «1». У кожного символу в потоці даних є свій номер. Ці унікальні номери можна записати у двійковій системі числення. Таким чином, тоді кожний символ можна записати як послідовність нулів «0» і одиниць «1». Якщо послідовно цим скористатися, тоді напевно одержимо з рядка деяку нову послідовність із нулів і одиниць.

На початку вхідний потік даних «вирівнюється» таким чином, що б його довжина стала конгруентною (порівнянно) з 448 за модулем 512. Вирівнювання забезпечується в такий спосіб: до вхідного потоку додається один біт «1», а потім біти «0» доти доки, поки довжина вхідного потоку не буде рівна з 448 за модулем 512. Процес вирівнювання відбувається завжди, навіть якщо довжина вхідного потоку була вже порівнянна з 448 за модулем 512. Очевидно, що до потоку додається мінімум 1 біт, або якщо це максимум, то – 512 бітів. Отриману послідовність позначимо S.

З метою підрахування результату використовуються 4 подвійних слова (кожне по 32 біта). Ці зазначені подвійні слова ініціалізуються такими 16-вими значеннями, де першим наймолодший байт є (2.2)-(2.5):

$$A: 01234567, \tag{2.2}$$

$$B: 89ABCDEF, \tag{2.3}$$

$$C: FEDCBA98, \tag{2.4}$$

$$D: 76543210. \tag{2.5}$$

Одночасно для отримання результату використовуються такі функці (2.6)-(2.9):

$$F(x, y, z) = (x \,\&\, y) \mid (\sim x \,\&\, z), \tag{2.6}$$

$$G(x, y, z) = (x \,\&\, z) \mid (y \,\&\, \sim z), \tag{2.7}$$

$$H(x, y, z) = x \wedge y \wedge z, \tag{2.8}$$

$$I(x, y, z) = y \wedge (x \mid \sim z), \tag{2.9}$$

де

& - побітове І,

| - побітове АБО,

^ - побітове що виключає АБО,

~ - побітове заперечення,

X,Y,Z – це зазначені подвійні (32-бітні) слова.

Результати функцій, відповідно, теж є подвійні слова. Для кінцевого підрахунку застосовується ще одна функція (нехай позначимо її W). Вона обробляє свої вхідні дані особливим чином і повертає на виході результат (тобто, опис цієї функції є достатньо комплексним, оскільки відрізняєтьсязалежно від способу реалізації – скажімо, реалізація для мови програмування Java використовує свою функції синуса – і тоді виходить за рамки даної дипломної роботи). Обробка цих даних відбувається з використанням функцій F, G, H, I (2.6)-(2.9) у такій послідовності.

Запишемо ці випадки обчислень стисло у вигляді кроків узагальненого алгоритму:

1) Запам'ятовуємо перші 512 біт нашої послідовності S.
2) Тепер видаляємо перші 512 біт з послідовності S (насправді можна обійтися й без видалення, але тоді на 1 кроці треба брати не перші 512 бітів, а наступні 512 біт).
3) Викликаємо зазначену вище функцію W. ЇЇ параметри A,B,C,D – це поточні значення відповідних подвійних 32-бітних слів. Параметр T містить у собі значення збережених 512 бітів.
4) Додаємо до A A0.
5) Додаємо B=B+B0.
6) Додаємо C=C+C0.
7) Додаємо D=D+D0.
8) Якщо довжина послідовності 0, виходимо.
9) Переходимо до кроку 1.

Після виконання цього алгоритму отримаємо нов ізначення виразів A,B,C,D – це і є результат (його довжина складе 128 біт). На практиці часто можна побачити результат MD5 як послідовність із 32 16-вих чисел, скажімо, ось такий вигляд:

"a" - CC175B9C0F1B6A831C399E269772661

"abc" - 90150983cd24fb0d6963f7d28e17f72

Просимо звернути увагу, що від довжини вихідного рядка довжина MD5-коду не залежить, а записувати цей код можна як з використанням прописних літер, так і малих.

Задяки компактності, часто він застосовується для цифрових підписів та потвердження цілісності передачі даних під час вирішення багатьох задач, пов’язаних із трансфером інформації.

Блок-схема роботи алгоритму процесів підрахунку та порівняння хеш-кодів за методом CRC 16наведена на рис. 2.4.

У даній роботі використаємо 128-бітну послідовність для контролю внесення змін в файл.

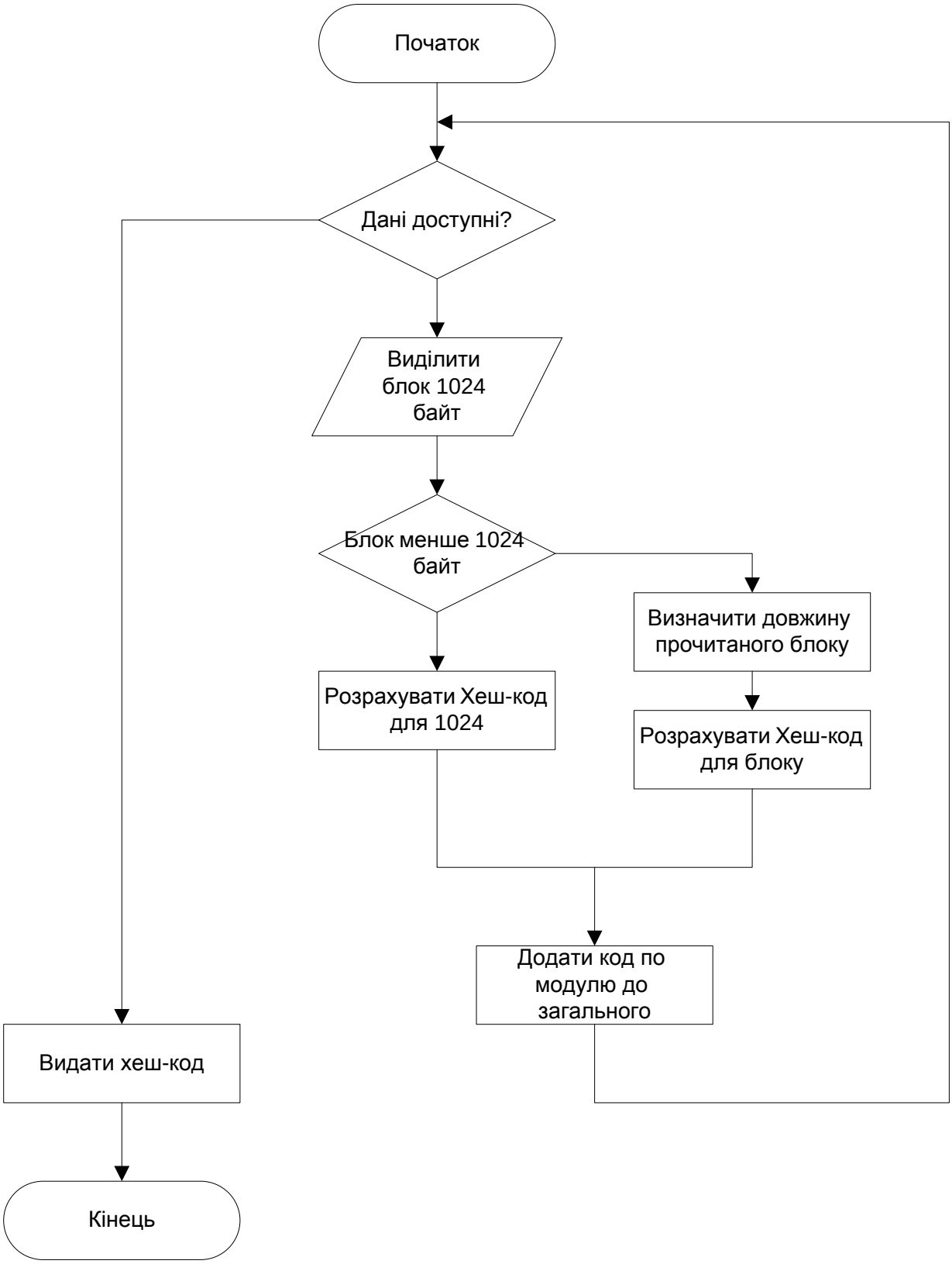


Рисунок 2.4 – Блок схема роботи алгоритму процесів підрахунку та порівняння хеш-кодів за методом CRC 16

2.4 Побудова користувацького інтерфейсу для програми блочного порівняння файлів

Спеціалізований пакет Abstract Windowing Toolkit (AWT) дозволяє досить зручно створювати програми із графічними інтерфейсами різної складності, що не залежать від конкретної платформи чи ОС. Більше того, професійне використання пакету AWT є набагато простішим й зрозумілішим за програмні інтерфейси під операційну систему Windows (наприклад, WinAPI, MFC чи ATL), Motif або OS/2. Ще в поставці пакету JDK версії 1.2 було додано досить багато нових класів і нових інтерфейсів, включно з професійною підтримкою функцій рисування, виведення на принтер, обробки графічних зображень і підтримки функції Drag and Drop, а також, Java 2D. Тепер до базового складу стандартних бібліотек мови програмування Java входить практично все, що може бути необхідно для розробки кросплатформеного графічного інтерфейсу будь-якої складності і зручності простим шляхом. На цій основі, загалом, будується бібліотека Swing – основна серед графічних компонентів технології Java [25-29].

Майже усі компоненти, що перелічені в технічній документації, які постачаються разом із даним інтегрованими середовищами розробки. Також, можна звернути увагу на програму SwingSet із стандартної поставки пакету JDK, щоб побачити компоненти бібліотеки Swing у дії.

Компоненти бібліотеки Swing є на 100% «рідними» Java-компонентами. Це означає, що вони цілком не залежать від якихось конкретних реалізацій інтерфейсних елементів на різних платформах. Це безпересно означає, що ці компоненти доступні на усіх платформах та операційних системах, де є підтримка Java 2.

У загальному випадку, усі компоненти Swing реалізовані як розширення від AWT. Відомо, що, багато традиційних компонентів AWT – такі, як: кнопки, списки, діалогові панелі, ін; у теперішній час повністю перероблені і у порівнянні із материнським пакетом, відповідно, мають більш розширений функціонал. Завдяки цьому, компоненти AWT стали значно доступнішими на різних платформах та операційих стстемах і тепер мають змогу давати додаткові можливості, які є відсутні в деяких інших графічних середовищах. На рисунку нижче показана взаємодія Swing і AWT.

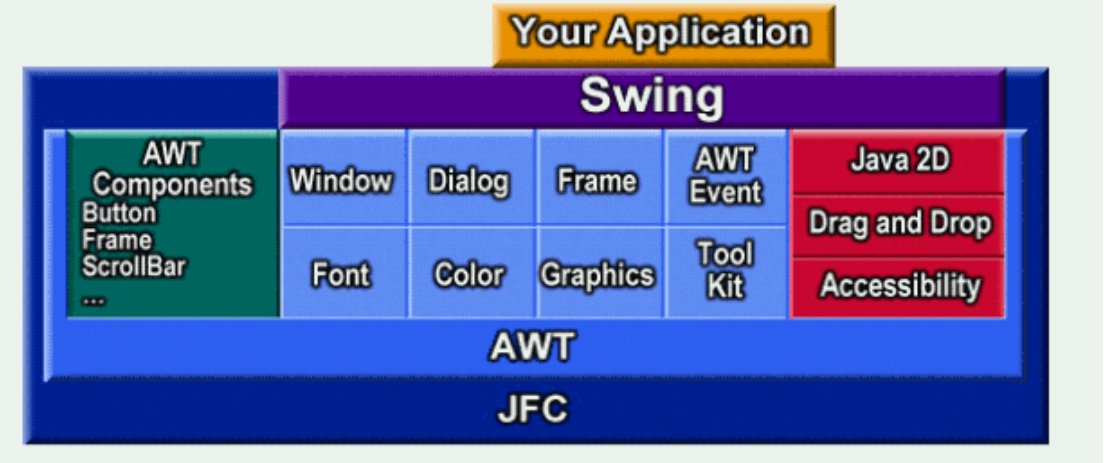


Рисунок 2.5 – AWT та Swing

Відомо, що вдосконалені компоненти Swing відіграють досить важливу роль під час створення графічних інтерфейсів різних програм. Зведемо в таблицю 2.1 переважну більшість основних компонентів і одночасно дамо коротку їх характеристику [30-33].

Таблиця 2.1 – Коротка характеристика деяких основних компонентів Swing

Компонент	Опис
Box	Контейнер загального свого призначення для організації декількох вкладених компонентів використовуючи модель «BoxLayout».
JApplet	Підклас великого класу Applet, і містить можливості панелі «JRootPane» з метою зв‘язати компоненти в єдине ціле
JButton	Універсальна кнопка, що може містити просто текст, або навіть графічне зображення, чи і те й інше

Продовження таблиці 2.1

JCheckBox	Спеціалізована кнопка з додатковою незалежною фіксацією
JCheckBoxMenuItem	Спеціалізована кнопка з незалежною фіксацією для додаткового використання в меню
JColorChooser	Додаткова компонента для вибору варіанту кольору в одній з колірних схем. Застосовується разом з javax.swing.colorchooser
JComboBox	Розширений комбінований список – рядок введення даних, з ним – випадаючий список і тоді користувач має можливість вводити текст або вибирати елемент зі списку на вибір.
JComponent	Основний кореневий елемент ієрархії спеціалізованої бібліотеки компонентів Swing. Додає специфічні розширені властивості на зразок підказок і підтримки подвійної буферизації (якщо потрібно)
JDesktopPane	Спеціальний контейнер для компонентів JInternalFrame для підтримки парадигми робочого столу «в одному вікні». Підтримує також і багатовіконний інтерфейс (MDI).
JDialog	Спеціальний контейнер для відображення буд-яких діалогових панелей
JEditorPane	Розширений текстовий редактор з новими можливостями форматування текстів зсобами об'єкту EditorKit. Додатково може відображати й редагувати текст у форматі HTML і RTF.
JFileChooser	Компонента вибору файлу та/або каталогу. Тут додатково підтримується фільтрація й можливість попереднього перегляду вмісту файлу. Застосовується разом з javax.swing.filechooser
JFrame	Спеціалізований контейнер для вікон верхнього рівня

Продовження таблиці 2.1

JInternalFrame	Спеціалізований контейнер для підтримки вкладених вікон. Схожий з JFrame і має заголовок вікна. Одночсано є незалежним вікном і відображається усередині батьківського контейнера. Часто застосовується разом з JDesktopPane.
JLabel	Спеціалізована компонента для відображення тексту, графіки чи і того і іншого
JLayeredPane	Новий контейнер, що дозволяєсвоїм дочірнім об'єктам перекривати один одного. Керує порядком накладення дочірніх об'єктів одни на інший.
JList	Зручна компонента для відображення розширеного списку з можливістю вибору. Елементами цього списку можуть бути рядки, також, графічні зображення або інші об'єкти теж.
JMenu	Спеціалтне випадаюче меню, яке базово входить до складу JMenuBar або це може бути підменю усередині іншого меню.
JMenuBar	Компонента, що відображає розмірну лінійку меню.
JMenuItem	Позначає належність до одного елемент меню
JOptionPane	Компонента, що найчастіше застосовується для швидкого відображення діалогових панелей усередині батьківського контейнера JDialog. Також, задає набір статичних методів для стандартних діалогових панелей
JPanel	Спеціалізований контейнер для групування компонентів у відповідному LayoutManager.

Продовження таблиці 2.1

JPasswordField	Поле для подальшого введення паролю (тексту), що забивається символами безпеки.
JPopupMenu	Вікно для подальшого відображення спливаючого меню. Застосовується в межахJMenu або для створення окремих спливаючих меню
JProgressBar	Компонента, що виявляє процес виконання тривалої операції
JRadioButton	Радіокнопка
JRadioButtonMenuItem	Радіокнопка для подальшого використання в меню
JRootPane	Комплексний контейнер, що зазвичай застосовується JApplet, JDialog, JFrame і JInternalFrame.
JScrollBar	Різні типи смуги прокручування
JScrollPane	Контейнер, що дозволяє усім своїм дочірнім компонентам прокручуватись вертикально/горизонтально. Додатково забезпечує підтримку фіксованих зон.
JSeparator	Компонента для промальовування горизонтальних і вертикальних обмежувачів.
JSlider	Компонента для візуального адаптивного введення цифрових значень шляхом перетягуванням повзунка
JSplitPane	Контейнер, спеціалізованого призначення, у якому відображаються два дочірніх об'єкти, причому, розмір яких можна змінювати довільно розробнику чи користувачу
JTabbedPane	Контейнер, що реалізовує спеціалізовану панель із закладками

Продовження таблиці 2.1

JTable	Компонента для цілковитого відображення таблиць із можливістю редагування їх вмісту «на льоту». Може відображати як стрічкові дані, так і будь-який інший тип текстових даних. Зазвичай, застосовується разом з javax.swing.table
JTextArea	Компонента для відображення й одночасного (за потреби) редагування багаторядкового тексту. Заснований на додатковій компоненті JTextComponent.
JTextComponent	Компонента, призначена для реалізації компонент відображення й редагування тексту. Є чільною частиною javax.swing.text
JTextField	Компонента, призначена для відображення, введення й редагування лише одного рядка тексту. Заснована в своїй основі на компоненті JTextComponent.
JTextPane	Підклас JEditorPane служить для відображення й редагування попередньо відформатованого тексту, але такого, що не є текстом у форматі HTML або RTF
JToggleButton	Розширений батьківський компонент спеціально для JCheckBox і JRadioButton.
JToolBar	Компонента, призначена для відображення розробленої панелі інструментів
JToolTip	Вікно для відображення підказок користувачу або інша пояснювальна інформація

Закінчення таблиці 2.1

JTree	Компонента для відображення і представлення деревовидної структури даних. Застосовується разом з javax.swing.tree
JViewport	Контейнер для відображення деякої окремої частини визначеного дочірнього об'єкта. Звичайно застосовується разом з JScrollPane.
JWindow	Вікно, але таке, що без заголовка, без смуг прокручування та інших елементів.

Однією з найбільш найцікавіших властивостей спеціалізованої бібліотеки Swing, які вона додає у порівнянні з AWT – це вбудована у неї можливість зміни зовнішнього вигляду компонентів, а також, інтерфейсів – за типом Pluggable Look and Feel (PL&F). Запропонована архітектура PL&F дозволяє програмісту надбудовувати зовнішній вигляд і, за потреби, окремо прорисовувати як сам вигляд, так і поведінку не лише одного компонента, а й цілої групи компонентів. Також до складу поставки Swing входить новий ряд наперед визначених стилів – Metal L&F, Motif L&F, Windows L&F. Передбачені розробниками, існують відповідні бібліотеки для окремих інтерфейсів Macintosh та інших платформ [34-37].

Для досягнення мети нашої програми блочного порівняння файлів необхідно застосувати два важливих рішення: 1) забезпечити можливість позначок кольором в списках, і 2) - користувацьке меню.

Для того, щоб програміст міг створити список із зазначеними позначками, необхідно перевизначити компоненту, що відповідає за відображення деякого елемента списку. Це DefaultListCellRenderer. І містить він лише функцію: getListCellRendererComponent, що приймає як аргумент визначене значення елемента списку. Напишемо тут просту функцію, що буде перевіряти наявність чи відсутність позначки у внутрішньому об’єкті елемента-файлу, і, в залежності від результату, присвоюватимемо фону відповідне значення (лістинг 2.2):

```
        DefaultListCellRenderer renderer =
         (DefaultListCellRenderer) super.getListCellRendererComponent(list, value,
  index, isSelected, cellHasFocus);
        if(value instanceof UltimateCompare.FileRecord)
        {
         UltimateCompare.FileRecord curRecord =
  (UltimateCompare.FileRecord)value;
         if(curRecord.marked)
          renderer.setForeground(Color.red);
         else
          renderer.setForeground(Color.black);
        }
        return renderer;
```

Лістинг 2.2 – Встановлення позначки в списку

Далі програмісту треба викликати спеціалізовану функцію rightWing.addListSelectionListener(this), яка зареєструє створений ним новий обробник відображення елементів. З метою створення меню в програмі знадобиться кілька об’єктів. Тоді винесемо службову функцію, яка буде створювати новий пункт меню за назвою і командою (лістинг 2.3).

```
private JMenuItem createMenuItem(String name, String command)
 {          JMenuItem newItem = new JMenuItem(name);
            newItem.setActionCommand(command);
            newItem.addActionListener(this);
            return newItem;
  }
```

Лістинг 2.3 – Службова функції конструювання елемента меню

Тепер ми можемо надати код для створення користувацького меню (лістинг 2.4):

```
JMenu submenu = new JMenu("Порівняння");
submenu.add(createMenuItem("Нове порівняння", "newCompare"));
submenu.add(new JSeparator());
submenu.add(createMenuItem("Просте порівняння", "simpleCompare"));
submenu.add(createMenuItem("Синхронізація", "deepCompare"));
submenu.add(createMenuItem("Знайти дублікати", "findDoubles"));
submenu.add(createMenuItem("Очистити позначки", "clearCompare"));
submenu.add(new JSeparator());
submenu.add(createMenuItem("Вихід", "quit"));
menuBar.add(submenu);
setJMenuBar(menuBar);
```

Лістинг 2.4 – Створення користувацького меню.

І нарешті, коли зроблена левова частина роботи, сконструюємо панель відображення характеристик файлу. Для цього потрібні стандартні компоненти з таблиці 2.1 (лістинг 2.5).

```
infoPane = new JPanel(new BorderLayout());
wingPane = new JPanel(new GridLayout(4, 1));
wingPane.add(new JLabel("Ім'я: "));
wingPane.add(new JLabel("Дата: "));
wingPane.add(new JLabel("Обсяг: "));
wingPane.add(new JLabel("CRC: "));
wingPane.setBorder(new EmptyBorder(0, 5, 0, 10));
infoPane.add(wingPane, BorderLayout.WEST);
wingPane = new JPanel(new GridLayout(4, 1));
wingPane.add(nameInfoRight);
wingPane.add(dateInfoRight);
wingPane.add(sizeInfoRight);
wingPane.add(contentInfoRight);
```

```
infoPane.add(wingPane, BorderLayout.CENTER);
infoPane.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5),
                                      new TitledBorder(new
EtchedBorder(EtchedBorder.RAISED), " Інфо файлу (права панель) ")));
innerPane.add(infoPane);
```

Лістинг 2.5 – Конструювання панелі характеристик файлу

Особливим тут є використання класу CompoundBorder – саме цей клас створює симпатичну рамку навколо блоку характеристик файлів. Нарешті, побудова користувацького інтерфейсу для компаратора не є складною задачею, якщо правильно і повно використовувати стандартними (таблиця 2.1) компоненти мови програмування Java.

2.5 Етапи створення програми блочного порівняння файлів

Створення програми мовою програмування Java поділяється на такі основні етапи.

1) Розробка головного класу програми.
2) Розробка користувацького інтерфейсу.
3) Реалізація основних функцій.

Головний клас нашої програми відрізняється від інших стандартних класів наявністю так званої «головної функції», яка є точкою входу в програму і виконується першою при запуску. Використаємо для створення програми інтегроване середовище розробки JDeveloper [38-42].

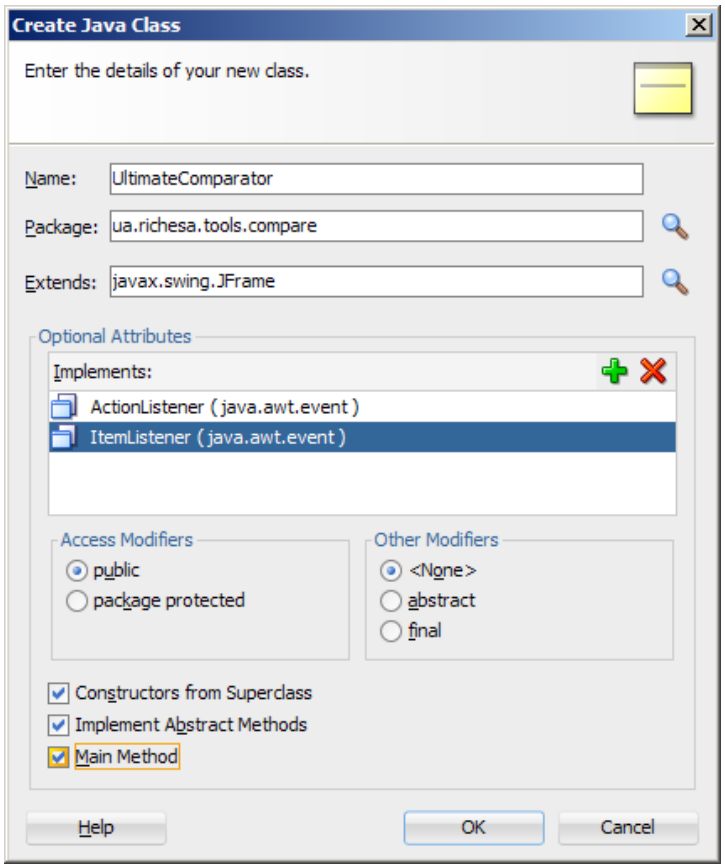


Рисунок 2.6 – Створення головного класу програми

Звернімо увагу, що в полі Extends має бути вказано унікальне ім′я батьківського класу програми (в даному випадку це клас JFrame, тобто клас головного вікна програми), а в списку Optional Attributes мають бути вказані інтерфейси, які він повинен реалізовувати.

Для поставлених у роботі цілей необхідні інтерфейси ActionListener та ItemListener, які дозволяють головному класу бути обробником подій відповідно – натискання кнопки і переключення радіокнопки.

Конструювання користувацького інтерфейсу для даної прграми блочного порівняння файлів можна вести двома способами – за допомогою спеціалізованого візуального редактора, або за допомогою застарілого методу «жорсткого кодування». Простіше, звісно – за допомогою редактора [43-46].

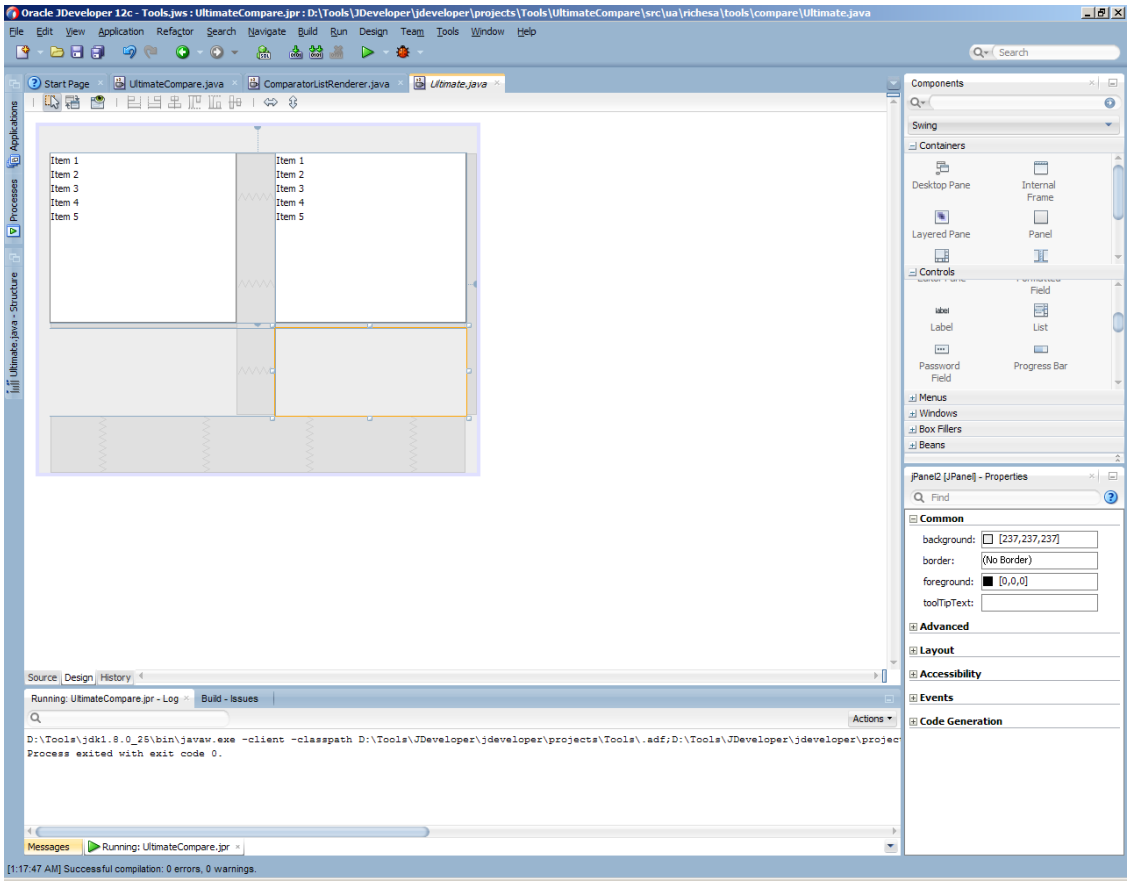


Рисунок 2.7 – Візуальний редактор форм JDeveloper

Остаточний дизайн користувацького інтерфейсу виглядатиме так:

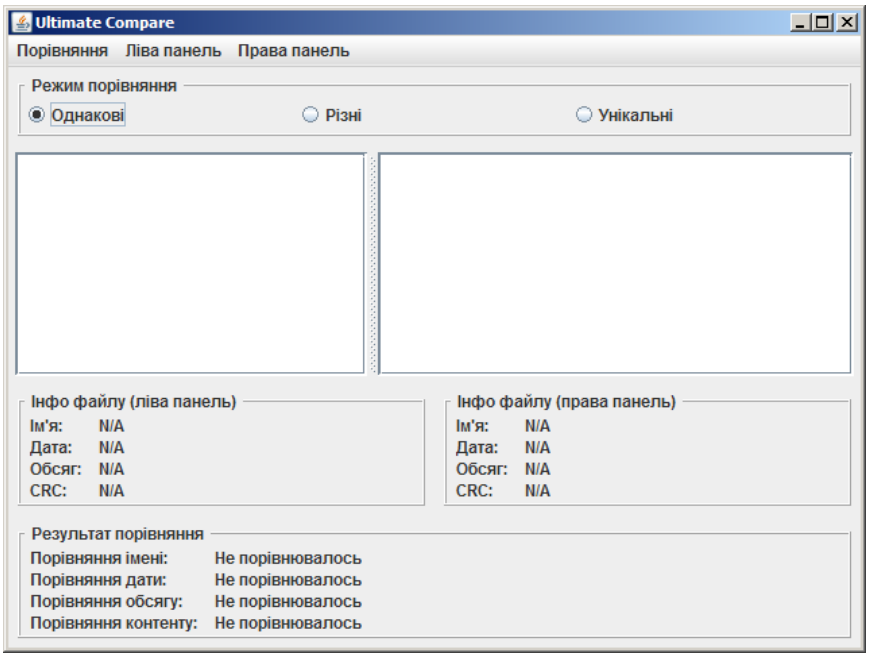


Рисунок 2.8– Дизайн користувацького інтерфейсу.

В процесі розробки програми блочного порівнянн файлів скористаємось одним із спеціальних окремих класів Java для створення нової динамічної панелі, яка здатна перемикатись між двома різними панелями, наприклад, – із текстовою зоною та компонентом-деревом. Таким чином, ми, так би мовити, економимо робочий простір програми, і саме цим досягаємо максимуму подання інформації.

Після того, як інтерфейс сконструйовано, можна додатково розподілити функції за відповідними елементами [47-54]:

Таблиця 2.2 – Співставлення функцій

Елемент меню	Функція
Нове порівняння	Очистити обидві панелі і підготувати програму до роботи
Просте порівняння	Виконати просте порівняння, по імені, даті чи розміру
Синхронізація	Виконати синхронізацію двох каталогів, підсвітити різницю
Знайти дублікати	Знайти дублікати виділеного файлу
Очистити позначки	Зняти позначки в обох панелях
Додати файл	Додає новий файл на панель
Додати каталог	Додає усі файли з вказаного каталогу на панель
Додати каталог рекурсивно	Додає усі файли із вказаного каталогу та його підкаталогів
Видалити файл	Видаляє файл

Таблиця 2.3 – Співставлення функцій (продовження)

Радіокнопка Однакові	Режим підсвітки однакових файлів
Радіокнопка Різні	Режим підсвітки різних файлів
Радіокнопка Унікальні	Режим підсвітки унікальних файлів

Реалізація усіх цих функцій в кінцевому підсумку складає простий та інтуїтивно зрозумілий графічний інтерфейс для програми блочного порівняння файлів. Вихідний текст програми наведено в додатку.

2.6 Тестування і перевірка правильності роботи програми

Перевіримо роботу програми на тестовому прикладі. Запакуємо і розпакуємо десять файлів, а потім порівняємо їх вміст за допомогою утиліти порівняння каталогів. Створимо два каталоги, Source та Dest.

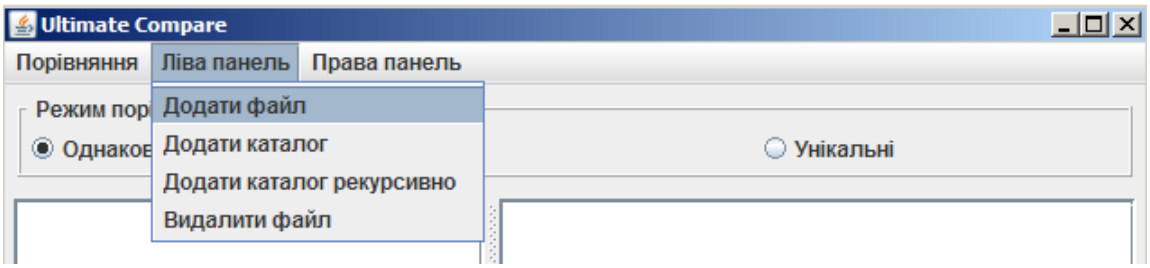


Рисунок 2.9 – Додавання файлів на панель

Спочатку додамо файли з цих каталогів на панелі. Для цього застосовується функція Додати файл, або Додати каталог, або Додати каталог рекурсивно. В даному випадку нам необхідна друга функція.

Оскільки меню Ліва панель і Права панель не відрізняються, просто повторимо цю функцію для кожної панелі і отримаємо:

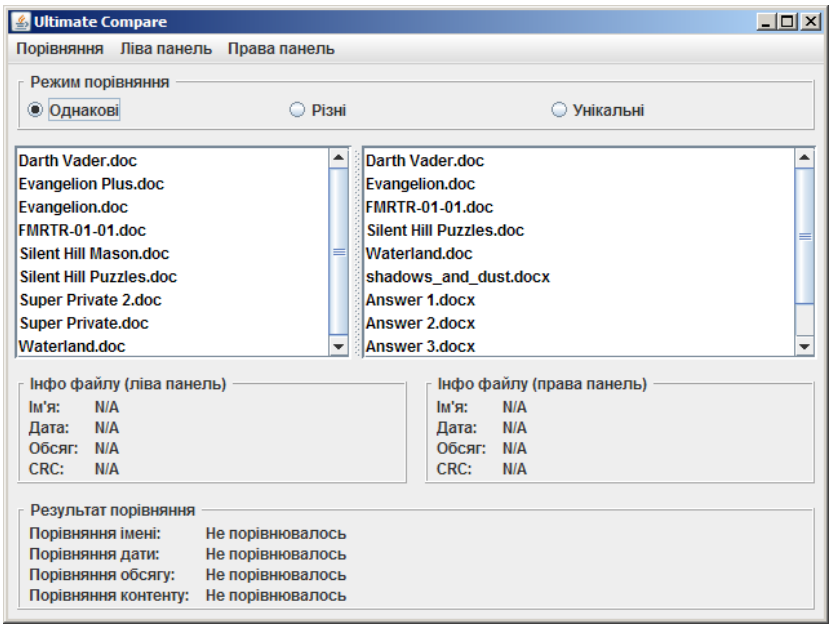


Рисунок 2.10 – Робочий простір програми

Ці два каталоги в чомусь співпадають, а в чомусь відрізняються. Виділятимем по два файли: один в правій панелі, інший в лівій. В нижніх панелях будуть відображатись їх характеристики та відмінності.

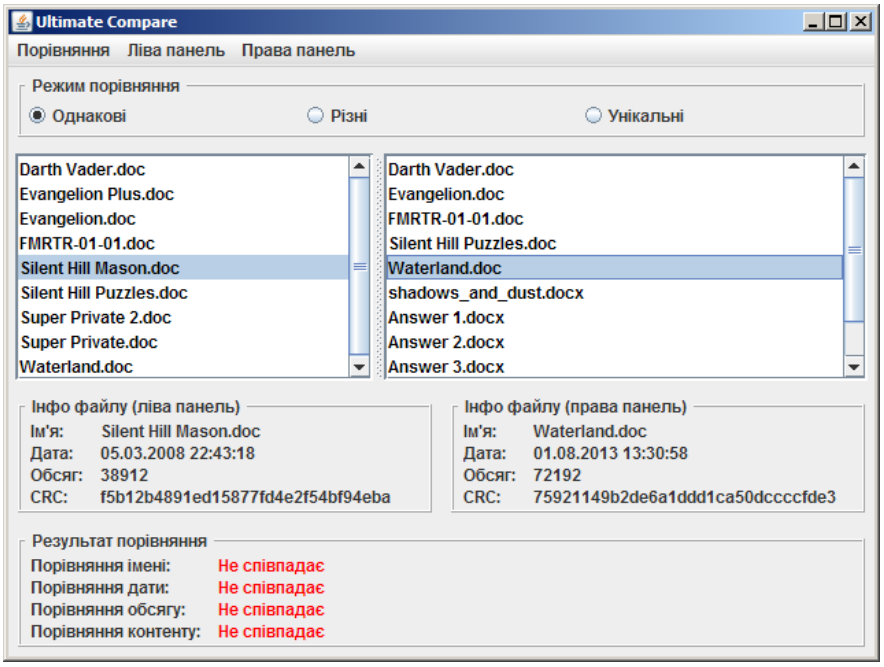


Рисунок 2.11 – Порівняння різних файлів

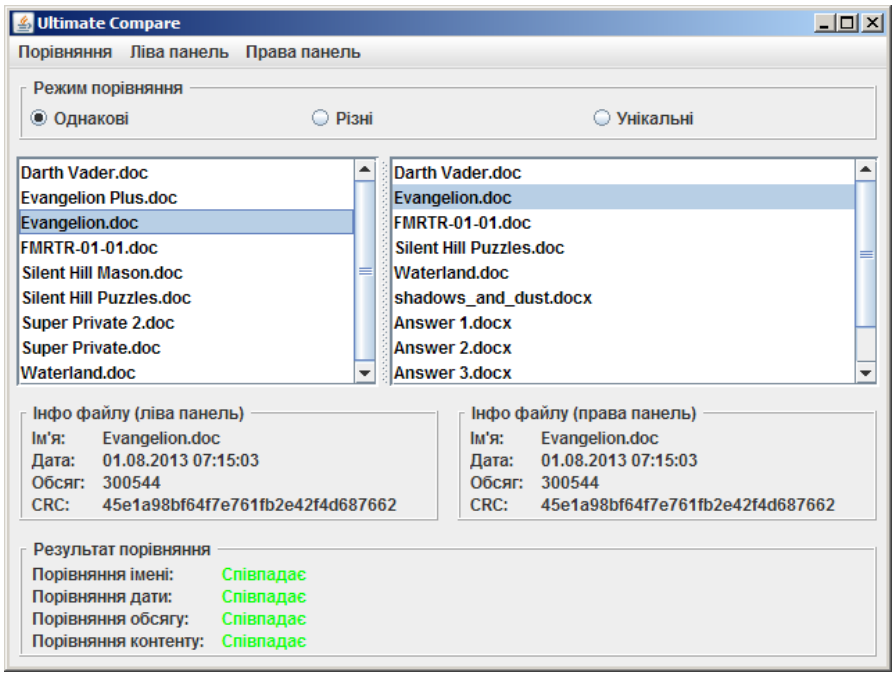


Рисунок 2.12 – Порівняння однакових файлів

Зверніть увагу, разом із характеристиками автоматично підраховується хеш-код за методом CRC 16. Це дозволяє без витрат часу швидко порівнювати зміст файлів.

Також, можна порівняти каталоги в автоматичному режимі.

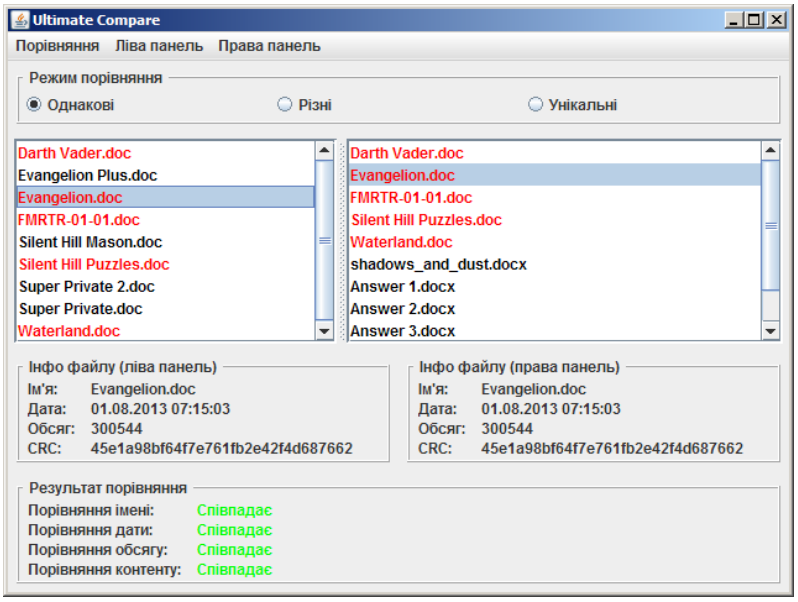


Рисунок 2.13 – Автоматичне порівняння із підсвіткою

Підсвітка в даному випадку демонструє однакові файли. За допомогою панелі перемикання режимів можна підсвітити навпаки, різні файли, або унікальні, які існують в одному екземплярі в кожному каталозі.

Таким чином, варіюючи використання функцій, можна легко відстежувати зміни в різних каталогах, таким чином, забезпечуючи коректну спільну роботу із файлами в спільному доступу.

Також, можна контролювати коректність пересланих файлів по захищеному каналі, і навіть відстежувати появу паразитних і троянських файлів.

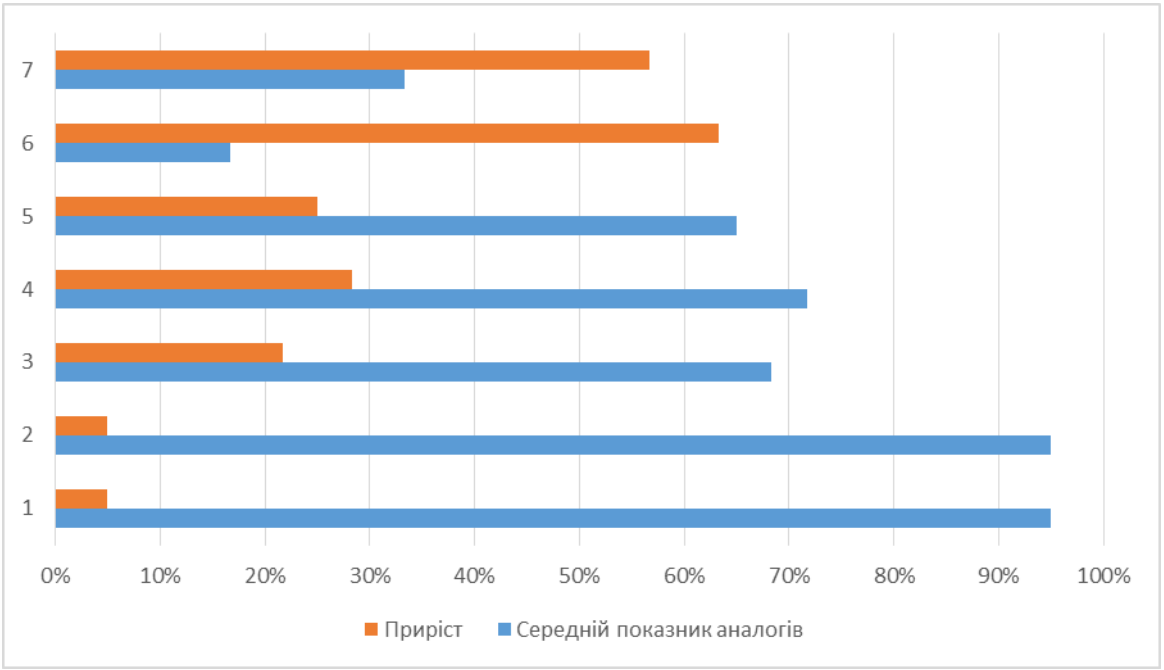
Порівняємо аналоги та конкуренти із розробленою програмою.

За можливістю порівняння внаслідок запуску системи всі програмні засоби є дієвими та мають відмінні характеристики. За можливістю порівняння внаслідок виявлення оновлень одна з утиліт, а саме, Allway Sync, не задовільняє вимогам, а два програмних засоби, а саме, Freefilesync та Sugarsync виконують цю функцію не недостатньому рівні, або такому, що не відповідає вимогам (чи не передбачено у функціоналі). За параметром «Можливість виключення системних даних» – утиліта Goodsync не має цього функціоналу, а сервіс Sugarsync – виконує це за додаткових умов. За параметром «Можливість виключення прихованих даних» - утиліти Goodsync та Viceversa не мають такого функціоналу, а інші програмні засоби на достатньому рівні виконують цю задачу. За показником «Можливість блочних замін у файлі» - усі програмні засоби не мають такого функціоналу крім нової розробки та частково – у Sugarsync, де такий підхід можна створити мануально, проте точково. За параметром «Застосування хеш-даних для операції порівняння файлів» - усі утиліти, крім частково Freefilesync, не мають цього функціоналу, а серед сервісів, що розглянуті тут – ця функціональність реалізована закритим методом.

Оцінимо числовими значеннями ці показники та занесемо дані у таблицю 2.4 та побудуємо діаграму 2.14.

Таблиця 2.4 – Оцінка характеристик

	Можливість порівняння внаслідок підключенні носія	Можливість порівняння внаслідок запуску системи	Можливість порівняння внаслідок виявлення оновлень	Можливість виключення системних даних	Можливість виключення прихованих даних	Можливість блочних замін у файлі	Застосування хеш-даних для операції порівняння файлів
Goodsync	100%	90%	90%	20%	30%	10%	10%
Viceversa	100%	90%	100%	80%	30%	10%	10%
Allway Sync	90%	90%	20%	80%	80%	10%	10%
Freefilesync	80%	100%	50%	90%	80%	10%	40%
Sugarsync	100%	100%	60%	70%	80%	50%	60%
Dropbox	100%	100%	90%	90%	90%	10%	70%
Нова розробка	100%	100%	90%	100%	90%	80%	90%
Середній показник аналогів	95%	95%	68%	72%	65%	17%	33%
Приріст	5%	5%	22%	28%	25%	63%	57%



Діаграма 2.14 – Порівняльний приріст по кожному з показників

У загальному, по ключовій позиції «Застосування хеш-даних для операції порівняння файлів» маємо приріст під час роботи даної програми на 23%.

2.7 Класи та функції програми блочного порівняння файлів

На основі класів можна створювати підкласи, які успадковують властивості та поведінку батьківських класів. Таким чином можна створити цілу ієрархію класів. Різні мови дещо по різному реалізовують механізм успадкування. Існує множинне та одинарне успадкування.

Множинне – це, коли підклас створюється на основі кількох безпосередніх батьків (як то в мові програмування C++). Одинарне успадкування – це коли клас може мати одного безпосереднього батька (мова програмування Java). Надкласи можуть мати свої надкласи, підкласи можуть також бути надкласами для певних класів.

До складу програми графічного інтерфейсу архіватора входить один основний клас і один допоміжний.

- UltimateCompare. Основний клас програми, успадкований від стандартного класу головного вікна програми JFrame, і реалізує інтерфейси ActionListener та ItemListener. Представляє собою головний клас програми, який власне і запускається при запуску програми.
- ComparatorListRenderer. Успадкований від класу DefaultListCellRenderer. Допоміжний клас, який дозволяє робити позначки кольором в елементі списку.

Кожен із класів має свої властивості та функції. В таблиці 3.5 наведені усі властивості, які має кожен із них, та їх функції:

Таблиця 2.5 – Властивості класів

Назва властивості	Тип	Значення
Клас UltimateCompare		
leftWing	JList	Ліва панель файлів
rightWing	JList	Права панель файлів
nameInfoLeft	JLabel	Ім’я файлу (вибраного в лівій панелі)
dateInfoLeft	JLabel	Дата
sizeInfoLeft	JLabel	Розмір
contentInfoLeft	JLabel	Хеш-код за методом CRC 16
nameInfoRight	JLabel	Ім’я файлу (вибраного в правій панелі)
dateInfoRight	JLabel	Дата
sizeInfoRight	JLabel	Розмір
contentInfoRight	JLabel	Хеш-код за методом CRC 16
nameCompare	JLabel	Результат порівняння імен
dateCompare	JLabel	Результат порівняння дат
sizeCompare	JLabel	Результат порівняння розмірів
contentCompare	JLabel	Результат порівняння змісту
modeEqual	JLabel	Режим показу однакових файлів
modeDifferent	JLabel	Режим показу різних файлів
modeOriginal	JLabel	Режим показу унікальних файлів

Клас ComparatorListRenderer спеціальних полів не містить, оскільки є допоміжним.

Зведемо також до таблиці і опишемо функції, з яких складається кожний із перелічених класів.

Таблиця 2.6 – Функції класів, що входять до програми

Назва функції	Список параметрів	Опис
Клас ComparatorListRenderer		
getListCellRendererComponent	JList<?> list, Object value, int index, boolean isSelected, boolean cellHasFocus	Повертає значення створеного віджета відображення елемента списку
Клас UltimateCompare		
UltimateCompare		Конструктор головного вікна програми, який створює користувацький інтерфейс і забезпечує виконання алгоритма.
addFiles	boolean wing, boolean directory	Додає файли на панель
removeFiles	boolean wing	Видаляє файл з панелі
createMenuItem	String name, String command	Службова функція створення елемента меню
createChecksum	String filename	Підраховує контрольну суму

Закінчення таблиці 2.6

createCRC	String filename	Підраховує контрольний код
invertMarks		Службова функція, що міняє усі позначки на файлах на протилежні
performSimpleComparation		Виконує просте порівняння
performSync		Виконує синхронізацію
performFindDoubles		Шукає дублі виділеного файлу
actionPerformed	ActionEvent e	Обробник натискань кнопок та меню
setStatus	JLabel what, String constant	Встановлює статус одного із полів інтерфейсу
valueChanged	ListSelectionEvent e	Відстежує зміну значення в комбобоксі
forceRevalidate		Оновлює інтерфейс
Main	String[] args	Головна функція програми

Таким чином, створена програма повністю реалізує поставлену задачу побудови графічного інтерфейсу для програми порівняння файлів.

2.8 Висновки за розділом

У даному розділі бакалаврської роботи запропоновано метод блочного порівняння двох файлів, зокрема, особливу увагу приділено процесам порівняння файлів та обчислення та порівняння хеш-кодів. Розроблено програму виконання блочного порівняння файлів та каталогів за допомогою хеш-коду за методом CRC 16, який дозволяє відстежувати зміни змісту файлу без витрат на рядкове чи двійкове

порівняння. Саме такий метод дозволяє виконувати швидке порівняння достатньо великих обсягів даних.

Зокрема, розроблено метод блочного порівняння, процеси методу, алгоритм порівнння хеш-функцій та описано ресурси розробленої програми. Побудовано користувацький інтерфейс для програми блочного порівняння файлів, для чого були використані компонентів Swing. У процесі розробки програмного засобу були описані етапи створення, класи, функції та проведено тестування розробленої програми.

У загальному, по ключовій позиції «Застосування хеш-даних для операції порівняння файлів» маємо приріст під час роботи даної програми на 23%.

3 ЕКОНОМІЧНА ЧАСТИНА

3.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності [54].

Результатом магістерської кваліфікаційної роботи «Метод та програмна реалізація блочного порівняння файлів» є розробка програмного методу реалізації блочного порівняння файлів. Для проведення технологічного аудиту залучено трьох незалежних експертів. У нашому випадку такими експертами є: Колесник Ірина Сергіївна (к.т.н., доцент каф. обчислювальної техніки ВНТУ), Богомолов Сергій Віталійович (к.т.н., доцент каф. обчислювальної техніки ВНТУ) та Захарченко Сергій Михайлович (к.т.н., доцент каф. обчислювальної техніки ВНТУ).

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 3.1.

Таблиця 3.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 3.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Закінчення таблиці 3.1

11	Термін реалізації ідеї	Термін реалізації ідеї	Термін реалізації ідеї	Термін реалізації ідеї	Термін реалізації ідеї

	більший за 10 років	більший за 5 років. Термін окупності інвестицій більше 10-ти років	від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 3.2.

Таблиця 3.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Колесник	2 – Богомолов	3 – Захарченко
	Бали, виставлені експертами:		
1	4	3	4
Ринкові переваги (недоліки):			
2	3	2	2
3	4	4	4
4	4	3	4
5	3	4	3
Ринкові перспективи			
6	2	2	2
7	3	4	3
Практична здійсненність			
8	4	3	4
9	3	2	3
10	4	4	4
11	3	4	3
12	3	3	3
Сума балів	$СБ_1$=40	$СБ_2$=38	$СБ_3$=39
Середньоарифметична сума балів $\overline{СБ}$	39		

За даними таблиці 3.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 3.3.

Таблиця 3.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 39 балів, що відповідає рівню «вище середнього».

Предметом даної магістерської роботи є програмна реалізація порівняння файлів та каталогів за допомогою хеш-коду за методом CRC 16, який дозволяє відстежувати зміни змісту файлу без витрат на рядкове чи двійкове порівняння. Саме такий метод дозволяє виконувати швидке порівняння достатньо великих обсягів даних.

Процедура порівняння файлів широко використовується в галузі програмування, де ведеться спільна робота над великими проектами і є необхідним часто відслідковувати зміни, які вносять в робочі файли проектів різні користувачі і команди, тому швидкість відслідковування цих змін під час синхронізації файлів та (або) каталогів є актуальним завданням – щоб протягом порівняння не були внесені повторні зміни. Відомий алгоритм роботи методу CRC 16 дозволяє звести процедуру порівняння файлів до порівняння хеш-кодів, які вираховуються як строкове представлення контрольних кодів для блоків визначеного розміру, на які поділяється вміст файлу.

Задача блочного порівняння файлів має важливий прикладний характер, і для її розв’язання достатньо класичних методів, що і спричинює можливість застосування нових засобів сучасних інформаційних технологій з метою досягнення кращих показників аналізу та порівняння вмісту великих масивів файлів.

Нова розробка є актуальною, оскільки завжди існує потреба у організації порівняння великих масивів файлів, а використання спеціалізованого програмного забезпечення дозволить пришвидшити процес такого порівняння.

Нова розробка є кращою за існуючі аналоги практично за всіма обраними параметрами.

За можливістю порівняння внаслідок запуску системи всі програмні засоби є дієвими та мають відмінні характеристики. За можливістю порівняння внаслідок виявлення оновлень одна з утиліт, а саме, Allway Sync, не задовільняє вимогам, а два програмних засоби, а саме, Freefilesync та Sugarsync виконують цю функцію не недостатньому рівні, або такому, що не відповідає вимогам (чи не передбачено у функціоналі). За параметром «Можливість виключення системних даних» – утиліта Goodsync не має цього функціоналу, а сервіс Sugarsync – виконує це за додаткових умов. За параметром «Можливість виключення прихованих даних» - утиліти Goodsync та Viceversa не мають такого функціоналу, а інші програмні засоби на достатньому рівні виконують цю задачу. За показником «Можливість блочних замін у файлі» - усі програмні засоби не мають такого функціоналу крім нової розробки та частково – у Sugarsync, де такий підхід можна створити мануально, проте точково. За параметром «Застосування хеш-даних для операції порівняння файлів» - усі утиліти, крім частково Freefilesync, не мають цього функціоналу, а серед сервісів, що розглянуті тут – ця функціональність реалізована закритим методом.

Оцінимо числовими значеннями ці показники та занесемо дані у таблицю 3.4.

Таблиця 3.4 – Оцінка характеристик аналогів та нової розробки

	Можливість порівняння внаслідок підключенні носія	Можливість порівняння внаслідок запуску системи	Можливість порівняння виявлення внаслідок оновлень	Можливість виключення системних даних	Можливість виключення прихованих даних	Можливість блочних замін у файлі	Застосування хеш-даних порівняння дляоперації
Goodsync	10	9	9	2	3	1	1
Viceversa	10	9	10	8	3	1	1
Allway Sync	9	9	2	8	8	1	1
Freefilesync	8	10	5	9	8	1	4
Sugarsync	10	10	6	7	8	5	6
Dropbox	10	10	9	9	9	1	7
Нова розробка	10	10	9	10	9	8	9

Даний програмний продукт в кілька разів прискорює та полегшує процес резервного копіювання БД.

Продукт, який пропонується, є модифікацією продуктів, що вже існують на ринку. Запропонований метод та програмна реалізація блочного порівняння файлів будуть корисним для державних та приватних компаній, особливо за умов, що їх діяльність тісно пов’язана із галузями ІТ-індустрії.

Розробка технічно готова на 100%. Даний програмний продукт використовується підприємством, для оптимізації процедури резервного копіювання. Оформлено акт впровадження.

Розповсюдження програмного забезпечення є умовно-безкоштовним (Trial), тобто користувач може протестувати програму на протязі визначеного періоду часу, при чому для безкоштовного користування надається обмежена версія програмного забезпечення.

Комерціалізація розробки знаходиться на початковому етапі, необхідно провести пошук потенційних партнерів та інвесторів.

На ринку праці наявні фахівці відповідної кваліфікації для обслуговування та підтримки програмного продукту, регламентні обмеження відсутні і немає необхідності отримання дозвільних документів.

Серед методів розкрутки нової розробки, які дозволяють отримати віддачу від витрачених зусиль і закласти міцний фундамент для просування в майбутньому доцільно обрати поза магазинну форму продажу та принцип продажу товару за зразком (з використанням демо-версії програмного продукту).

3.2 Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської) та конструкторсько-технологічної роботи

Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної робіт складається з таких етапів:

1) розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2) розрахунок загальних витрат на виконання даної роботи;

3) прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат, які безпосередньо стосуються виконавця даного розділу роботи, приймаючи до уваги те, що для розробки програми було залучено одного розробника.

Основна заробітна плата розробника (дослідника) $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t \ [грн], \qquad (3.1)$$

де

М – місячний посадовий оклад розробника, 11000,00 грн.

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (22) дні;

t – число робочих днів роботи розробника (дослідника) – 66 днів.

$$З_о = \frac{11000}{22} \cdot 66 = 33000{,}00\,(\text{грн}).$$

Додаткова заробітна плата $З_д$ розробника, розраховується як 12 % від суми основної заробітної, тобто:

$$З_д = (0{,}1...0{,}12) \cdot З_о\,[грн]. \qquad (3.2)$$

$$З_д = 0{,}10 \cdot 33000{,}00 = 3300{,}00\ (\text{грн}).$$

Нарахування на заробітну плату $Н_{зп}$ розробника становлять 22 % і розраховуються за формулою:

$$Н_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}\,[грн], \qquad (3.3)$$

де

$З_о$ – основна заробітна плата розробників, грн;

$З_д$ – додаткова заробітна плата розробника, грн;

β – ставка єдиного соціального внеску, 22%.

$$Н_{зп} = (33000{,}00 + 3300{,}00) \cdot 0{,}22 = 7986{,}00\ (\text{грн}).$$

У спрощеному вигляді амортизаційні відрахування розраховуємо за формулою:

$$А = (Ц \cdot Т)/(12 \cdot Тв)\ [\text{грн}], \qquad (3.4)$$

де

Ц – загальна балансова вартість обладнання, приміщення тощо, грн;

Т – фактична тривалість використання, міс;

$Т_В$ – термін використання обладнання, приміщень тощо, роки.

Зроблені розрахунки зведено до таблиці 3.5.

Таблиця 3.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, роки	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Приміщення	200000	25	3	2000,00
Ноутбук	20000	5	3	1000,00
Всього				3000,00

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Крім того, за три місяці було сплачено 450 грн на послуги Інтернет (150 грн/міс).

Витрати на канцтовари склали 250 грн.

Інші витрати $І_в$, як 50% від суми основної заробітної плати розробника тобто:

$$В_{ін} = 50\% \cdot З_о \, [грн]. \quad (3.5)$$

$$В_{ін} = 0{,}5 \cdot 33000{,}00 = 16500{,}00 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи – В:

=33000,00+3300,00+7986,00+3000,00+450,00+250,00+16500,00 =64486,00(грн).

Загальна вартість всієї наукової роботи ($В_{заг}$) визначається за формулою:

$$В_{заг} = В/\alpha \, [грн], \quad (3.6)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях.

$$В_{заг} = 64486{,}00/1 = 64486{,}00 \text{ (грн)}.$$

Проведемо прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \text{ [грн]}, \quad (3.7)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, як розробка знаходиться на дослідного зразка, то $\beta \approx 0,9$.

$В_{заг}$ - загальна вартість всієї наукової роботи – 41489,40 грн.

$$ЗВ = \frac{64486,00}{0,9} = 71651,11 \text{ (грн)}.$$

Отже, прогноз загальних витрат на виконання та впровадження результатів становить 71651,11 грн.

3.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спробуємо кількісно спрогнозувати вигоду яку можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Зрозуміло, що всі зроблені тут розрахунки будуть приблизними і не передбачають деталізації.

В умовах ринку, узагальнюючим позитивним результатом, який планує отримати підприємець від впровадження результатів нової розробки, є збільшення чистого прибутку.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 3 місяці. Позитивні результати від впровадження розробки очікуються вже в перший рік впровадження.

Проведемо прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_i \; [грн], \qquad (3.8)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Функціональні переваги нового продукту дають можливість збільшити чистий прибуток підприємства на 2000 грн, а кількість одиниць реалізованої послуги збільшиться: протягом першого року – на 200 од., протягом другого року – ще на 300 од., протягом третього року – ще на 400 од.

Орієнтовно: реалізація до впровадження результатів наукової розробки складала 1 од., а прибуток, що його отримувало підприємство на одиницю послуги до впровадження результатів наукової розробки – 150 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року:

$$\Delta П_1 = 150 \cdot 1 + (150 + 2000) \cdot 100 = 215150{,}00 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2 = 150 \cdot 1 + (150 + 2000) \cdot (100 + 200) = 645150{,}00 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року:

$$\Delta П_3 = 150 \cdot 1 + (150 + 2000) \cdot (100 + 200 + 400) = 1505150{,}00 \text{ (грн)}.$$

Отже, комерційний ефект від впровадження розробки виражається у щорічному збільшенні чистого прибутку підприємства протягом трьох років.

3.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Щоб оцінити доцільність фінансування проекту, необхідно провести розрахунки ефективності вкладених інвестицій.

Основними показниками є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

На першому етапі розрахуємо теперішню вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР. Будемо вважати, що ЗВ = PV=71651,11 (грн).

На другому етапі розраховуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку було розраховане раніше. Результати вкладених у наукову розробку інвестицій виявляться за перший рік після провадження у тому, що у першому році підприємство отримає збільшення чистого прибутку на 215150,00 грн відносно базового року, у другому році – збільшення чистого прибутку на 645150,00 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 1505150,00 грн (відносно базового року).

На третьому етапі для спрощення подальших розрахунків будуємо вісь часу, на яку наносимо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуються у ті терміни, коли вони здійснюються.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 3.1.

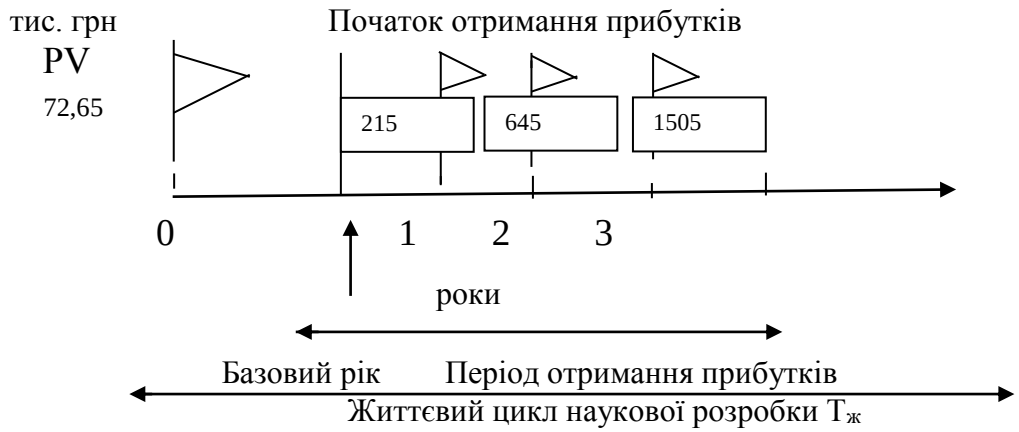


Рисунок 3.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

На четвертому етапі розраховуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою:

$$Е_{абс} = (ПП – PV) \text{ [грн]}, \quad (3.9)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t} \text{ [грн]}, \quad (3.10)$$

де

$\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні - 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”;

$$ПП = \frac{215150{,}00}{(1+0{,}1)^1} + \frac{645150{,}00}{(1+0{,}1)^2} + \frac{1505150{,}00}{(1+0{,}1)^3} = 1756810{,}40 \text{ (грн)}.$$

$$Е_{абс} = 1756810{,}40 – 71651{,}11 = 1685159{,}29 \text{(грн)}.$$

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки програмного продукту та їх впровадження принесе прибуток, тобто є доцільним, але це ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даної програми.

На п’ятому етапі розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \quad (3.11)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{1685159,29}{71651,11}} - 1 = \sqrt[3]{24,51} - 1 = 1,90 \text{ або } 190\%$$

Порівняємо $E_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. Спрогнозуємо величину $\tau_{мін.}$ У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \quad (3.12)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,2;

f – показник, що характеризує ризикованість вкладень; величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 190% > $\tau_{мін}$ = 30%, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

На шостому етапі розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [року]}. \quad (3.13)$$

$$T_{ОК} = \frac{1}{1,9} = 0,52 \text{ (року)}.$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок}$ < 3 років), то фінансування нової розробки є доцільним.

3.5 Висновок

В даному розділі було здійснено оцінювання комерційного потенціалу розробки. Проведено технологічний аудит з залученням трьох експертів.

Аналіз експертних даних показав, що рівень комерційного потенціалу розробки є високим. Дослідження комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що робить його конкурентоспроможним на ринку. Існуючі переваги нової розробки дозволять швидко її поширити на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальна вартість витрат на розробку і впровадження складає 71651,11 грн.

Абсолютна ефективність вкладених інвестицій в сумі 1685159,29 грн свідчить про отримання прибутку інвестором від впровадження програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 190%, що набагато вище за мінімальну бар'єрну ставку дисконтування, яка складає 30%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності складає 0,52 роки, що також свідчить про доцільність фінансування.

ВИСНОВКИ

У результаті виконання даної дипломної роботи розроблено програмний засіб та метод блочного порівняння файлів на персональному комп'ютері по CRC-кодах засобами Java. Високий рівень вирішення поставленої задачі досягнуто за рахунок використання сучасної мови програмування Java.

Можливості Java дозволяють інтегрувати в програму кілька варіантів порівняльних функцій: індивідуальну, групову, глибинне порівняння із контролем змісту, а також визначення унікальних файлів. Це зробить програму зручним інструментом програміста та керівника великими спільними проектами.

Обґрунтовано доцільність створення такої програми через поняття управління передаванням інформації та контроль достовірності. Доведено, що проблема синхронізації зводиться до задачі порівняння файлів.

В магістерській роботі виконано дослідження і аналіз сучасних методів та засобів порівняння та синхронізації файлів, розглянуто існуючі методи вирішення задачі порівняння файлів, технологічний ланцюжок роботи методу блочного порівняння файлів та розроблено новий метод блочного порівняння файлів, запропоновано ключові процеси роботи методу блочного порівняння файлів, алгоритм роботи процесів підрахунку та порівняння хеш-кодів за методом CRC 16, який дозволяє відстежувати зміни у вмісті файлу.

Розроблену програму виконання порівнянь файлів та каталогів за допомогою хеш-коду за методом CRC 16 можна використовувати для того, щоб відстежувати зміни змісту файлу без витрат на рядкове чи двійкове порівняння. Саме такий метод дозволяє виконувати швидке порівняння достатньо великих обсягів даних.

У загальному, по ключовій позиції «Застосування хеш-даних для операції порівняння файлів» маємо приріст швидкодії під час роботи даної програми на 23%.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Аникин Б. А. Аутсорсинг и аутстаффинг высокие технологии менеджмента: учеб. пособ. / Б. А. Аникин, И. Л. Рудая. – 2-е изд., перераб. и доп. – М.: ИНФРА-М, 2009. – 320 с. – (Высшее образование).
2. Гребешкова О. М. Аутсорсинг знань: потенціал партнерських від-носин підприємств у постіндустріальну епоху / О. М. Гребешкова, К.С. Денисенко // Стратегія економічного розвитку України: наук. зб. – К. : КНЕУ, 2012. – Вип. 29. – С. 191–199.
3. Євтошенко Н. О. Аутсорсинг в діяльності підприємств України: переваги та недоліки використання / Н. О. Євтошенко // Науковий вісник Ужгородського університету. – 2014. – No 1(42). – С. 44–47.
4. Заводовська І. І. Передумови та перспективи розвитку сучасного бізнесу на основі аутсорсингу / І. І. Заводська // Культура народов Причерноморья. – 2009. – No 80. – С. 43–45.
5. Загородній А. Г. Аутсорсинг та його вплив на витрати підприємства / А. Г. Загородній, Г. О. Партин // Фінанси України. – 2009. – No 9 (166). – С. 87–97.
6. Календжян С. О. Аутсорсинг и делегирование полномочий в деятельности компаний / С. О. Календжян. – М.: Дело, 2003. – 178 с.
7. Хейвуд Дж. Б. Аутсорсинг в поисках конкурентных преимуществ / Дж. Б. Хейвуд; [пер. с англ]. – М.: ИД “Вильямс”, 2004. – 176 с. 14. GlobalIndustryAnalysts [Електронний ресурс]. – Режим доступу: http://www.strategyr.com/methodology.asp.Eckel Bruce.
8. Офіційний сайт OwnCloud [Електронний ресурс] – Режим доступа: https://owncloud.com – Дата доступа: 25.05.2018.
9. Поднимаем сервис для хранения и синхронизации конфиденциальных данных [Электронный ресурс] – Режим доступу до ресурсу: https://xakep.ru/2014/09/24/cloud-crime/ – Дата доступа: 25.05.2018.

10. Офіційний сайт Nextcloud [Електронний ресурс] – Режим доступа: https://nextcloud.com – Дата доступа: 25.05.2018.

11. Офіційний сайт Pydio [Електронний ресурс] – Режим доступа: https://pydio.com – Дата доступа: 25.05.2018.

12. Офіційний сайт Seafile [Електронний ресурс] – Режим доступа: https://www.seafile.com/en/home/ – Дата доступа: 25.05.2018.

13. Peterson, W. Cyclic Codes for Error Detection [Text] / W. Peterson, D. Brown // Proceedings of the IRE. – 1961. – Vol. 49, Issue 1. – P. 228–235. doi: 10.1109/jrproc.1961.287814

14. Семеренко, В. П. Теория и практика CRC кодов: новые результаты на основе автоматных моделей [Текст] / В. П. Семеренко // Восточно-Европейский журнал передовых технологий. – 2015. – Т. 4, № 9 (76). – С. 38–48. doi: 10.15587/1729-4061.2015.47860

15. Walma, M. Pipelined cyclic redundancy check (CRC) calculation [Text] / M. Walma // 2007 16th International Conference on Computer Communications and Networks. – 2007. doi: 10.1109/icccn.2007.4317846

16. Krishna Reddy, K. V. An Optimization Technique for CRC Generation [Text] / K. V. Krishna Reddy // International Journal of Computer Trends and Technology (IJCTT). – 2013. – Vol. 4, Issue 9. – P. 3260–3265. – Available http://www.ijcttjournal.org

17. Sharma, H. FPGA implementation of 4-bit parallel Cyclic Redundancy Code [Text] / H. Sharma, S. Tomar, J. Kanungo // International Journal of Research in Engineering and Technology. – 2015. – Vol. 04, Issue 11. – P. 111–113. doi: 10.15623/ ijret.2015.0411021

18. Shreya Gawande, Dr. S. A. L. Design and Implementation of Parallel CRC for High Speed Application [Text] / S. Gawande, S. A. Ladhake // International Journal of Science and Research (IJSR). – 2015. – Vol. 4, Issue 2. – P. 90–92. – Available at: http:// www.ijsr.net

19. Koopman, P. Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks [Text] / P. Koopman, T. Chakravarty // International Conference on Dependable Systems and Networks, 2004. – 2004. doi: 10.1109/dsn.2004.1311885

20. Башмаков Александр Игоревич, Башмаков Игорь Александрович. Интеллектуальные информационные технологии: учеб. пособие для студ. вузов, обуч. по направлению подгот. дипломированных спец. "Информатика и вычислительная техника" - М. : МГТУ им.Н.Э.Баумана, 2005. - 302с.

21. Бигелоу Стивен Дж.. Сети: поиск неисправностей, поддержка и восстановление / Юрий Гороховский (пер.с англ.). - СПб. : БХВ-Петербург, 2005. - 1200с.

22. Вишневский, В. М.. Системы поллинга: теория и применение в широкополосных беспроводных сетях [Текст] / В. М. Вишневский, О. В. Семенова ; РАН, Институт проблем передачи информации им. А.А.Харкевича. - М. : Техносфера, 2007. - 309 с.

23. Галкин Валерий Александрович, Григорьев Юрий Александрович. Телекоммуникации и сети: Учеб. пособие для студ. вузов, обучающихся по спец. "Автоматизированные системы обработки информации и управления" направления подгот. дипломир. специалистов "Информатика и вычислительная техника" - М. : Издательство МГТУ им. Н.Э.Баумана, 2003. - 607с.

24. M. Fernandez, A. Malhotra, J. Marsh, M. Nagy, N. Walsh XQuery 1.0 and XPath 2.0 Data Model – W3C Working Draft, 2005.

25. M. Kay XSL Transformations 2.0 – W3C Working Draft, 2005.

26. P. C. Dibble Real – Time Java Platform Programming – NY, Prentice Hall, 2002

27. Altova Authentic Browser Edition Reference Manual – Altova Press, 2005.

28. B. Eckel Thinking in Java – NY, Prentice Hall, 2000

29. B. McLaughlin, J. Edelson Java and XML – O'Reilly, 2003

30. Code of Virginia, Title 18.2, Chap. 5 Crimes Against Property, Article 7.1. Computer Crimes.

31. Altova Authentic Desktop Edition Reference Manual – Altova Press, 2005.

32. Altova MapForce User's Guide and Reference Manual – Altova Press, 2005.

33. Altova Stylevision User's Guide and Reference Manual – Altova Press, 2005.

34. Altova XML Spy User's Guide and Reference Manual – Altova Press, 2005.

35. Applied XML Solutions – McMillan Computer Pub, 2002

36. D. Benyon, D. Stone, M. Woodroffe Experience with Developing Multimedia Courseware for the World Wide Web – //Human - Computer Studies, 47/1997.

37. E. R. Harold, W. Scott Means XML in a Nutshell – O'Reilly, 2005

38. E. van der Vlist. XML Schema – O'Reilly, 2002

39. G. Weber, M. Specht User Modeling and Adaptive Navigation Support in WWW - Based Tutoring Systems – University of Trier, Dept. of Psychology, NY, 1997.

40. J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, J. Philips, N. Jackiw, D. Suthers Developing Educational Software Components – // Educational Computing Research, 2001.

41. J. Shirazi Java Performance Tuning – Sebastopol, O'Reilly, 2000

42. S. Boag, D. Chamberlain, M. Fernandez, D. Florescu, J. Robie, J. Simeon XML Query Language – W3C Working Draft, 2005.

43. S. Mangano XSLT Cookbook – O'Reilly, 2000

44. S. Stelting, O. Maassen Applied Java Patterns – NY, Prentice Hall, 2001

45. T. Bray XML Specification 1.0 – W3C Recommendation, 1998.

46. United States Code, TITLE 17 - January 1, 1998. CHAPTER 1 - SUBJECT MATTER AND SCOPE OF COPYRIGHT.

47. URAN (Ukrainian Research and Academic Network), http: uran.net

48. Єжова. Л.Ф. «Алгоритмізація та програмування процедур обробки інформації». Київ: КНЕУ, 2000р.

49. Митчелл К. Керман. «Программирование и отладка. Учебный курс.» Москва – Киев: Вильямс, 2003.

50. Семотюк В. «Програмування в середовищі JAVA», Навч. посібник для студентів технічних спеціальностей Львів 2000.

51. Строуструп Б. «Справочное руководство по языку программирования JAVA с комментариями.» Издательство «Мир», 1992г.

52. Уинер Р. «Язык JAVA» М.:Мир, 1991.

53. Хосс Джексон, Тод Маркес «JAVA» справочник професионала. Москва: СП ЭКОМ, 2003.

54. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

ДОДАТКИ

Додаток А

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____

«____» _____ 20___ року

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської дипломної роботи

«МЕТОД ТА ПРОГРАМНА РЕАЛІЗАЦІЯ БЛОЧНОГО ПОРІВНЯННЯ ФАЙЛІВ»

08-023.ДР.005.00.000.ТЗ

Виконав: студент 2 курсу, групи 1КІ-18м

зі спеціальності:

123 «Комп'ютерна інженерія»

(шифр і назва напряму підготовки)

Гудименко О.О.

(прізвище та ініціали)

Керівник Савицька Л.А.

(прізвище та ініціали)

м. Вінниця – 2020 р.

1. Підстава для виконання дипломної роботи (ДР)

1.1 Предметом роботи даної програми є порівняння файлів на персональному комп'ютері по CRC-кодах засобами Java з можливістю інтегрувати в програму кілька варіантів порівняльних функцій: групову, індивідуальну, глибинне порівняння із контролем змісту, а також визначення унікальних файлів.

1.2 Наказ про затвердження теми дипломної роботи.

2. Мета і призначення ДР

Підсумком виконання даної дипломної роботи стала розробка методу блочного порівняння файлів та програми для виконання порівнянь файлів та каталогів за допомогою хеш-коду за методом CRC 16, яку можна використовувати для того, щоб відстежувати зміни змісту файлу без витрат на рядкове чи двійкове порівняння. Саме такий метод дозволяє виконувати швидке порівняння достатньо великих обсягів даних. З метою спростити роботу користувача був розроблений програмний продукт із зручним графічним інтерфейсом.

Високий рівень виконання поставленої задачі вирішено засобами мови програмування Java.

Основними перевагами є:

– Основні переваги – це безкоштовність і уміння розпаковувати різні типи форматів, серед яких є і rar. Архівувати Stuffit Expander взагалі не уміє, для цього є платний Stuffit Deluxe.

– Використання роздільної логіки доступу до даних від користування ними. Це було забезпечено шляхом створення двох проектів.

Основними недоліками є:

– Єдиним недоліком є те, що доступ може здійснюватися лише для читання. Метод entries() витягує всі записи з jar – файлу. Цей метод повертає список екземплярів Jarentry – по одній для кожного запису в jar – файлі.

3. Вихідні дані для виконання ДР

Список технічної літератури, аналіз, вивчення та дослідження процесів синхронізації та порівняння файлів чи папок, технічне завдання на магістерську роботу.

4. Матеріали, що подаються до захисту ДР

Пояснювальна записка ДР, графічні і ілюстративні матеріали, протокол попереднього захисту ДР на кафедрі, відгук наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до ДР українською та іноземною мовами.

5. Техніко-економічні показники

5.1 Витрати на програмні засоби, що використовуються в ході даної розробки, повинні бути мінімальними.

5.2 Лімітна ціна на програмне забезпечення не повинна перевищувати ціну аналога.

6. Порядок контролю виконання та захисту ДР

6.1.Робота виконується в три етапи, таблиця 6.1.

Таблиця 6.1 – Етапи виконання роботи.

Етап	Зміст	Початок	Кінець	Результат
1	Інформаційний пошук та огляд літературних джерел. Розробка методу та програмної реалізації блочного порівняння файлів.			Розділи 1 та 3.
2	Техніко-економічне обґрунтування теми дипломної роботи, розробка програмного засобу.			Чернетки матеріалів 1 розділу. Попередній захист.
3	Підготовка матеріалів пояснювальної записки.			Пояснювальна записка.

7. Загальний алгоритм роботи програми є таким:

1) Запам'ятовуємо перші 512 біт послідовності S.

2) Видаляємо перші 512 біт послідовності S (можна обійтися й без видалення, але тоді на першому кроці треба брати не перші 512, а наступні 512 біт).
3) Викликаємо функцію W. Параметри A,B,C,D - це поточні значення відповідних подвійних слів. Параметр T - це збережені 512 біт.
4) Додаємо до A A0.
5) B=B+B0.
6) C=C+C0.
7) D=D+D0.
8) Якщо довжина послідовності 0, виходимо.
9) Переходимо до кроку 1.

8. Порядок контролю та прийому

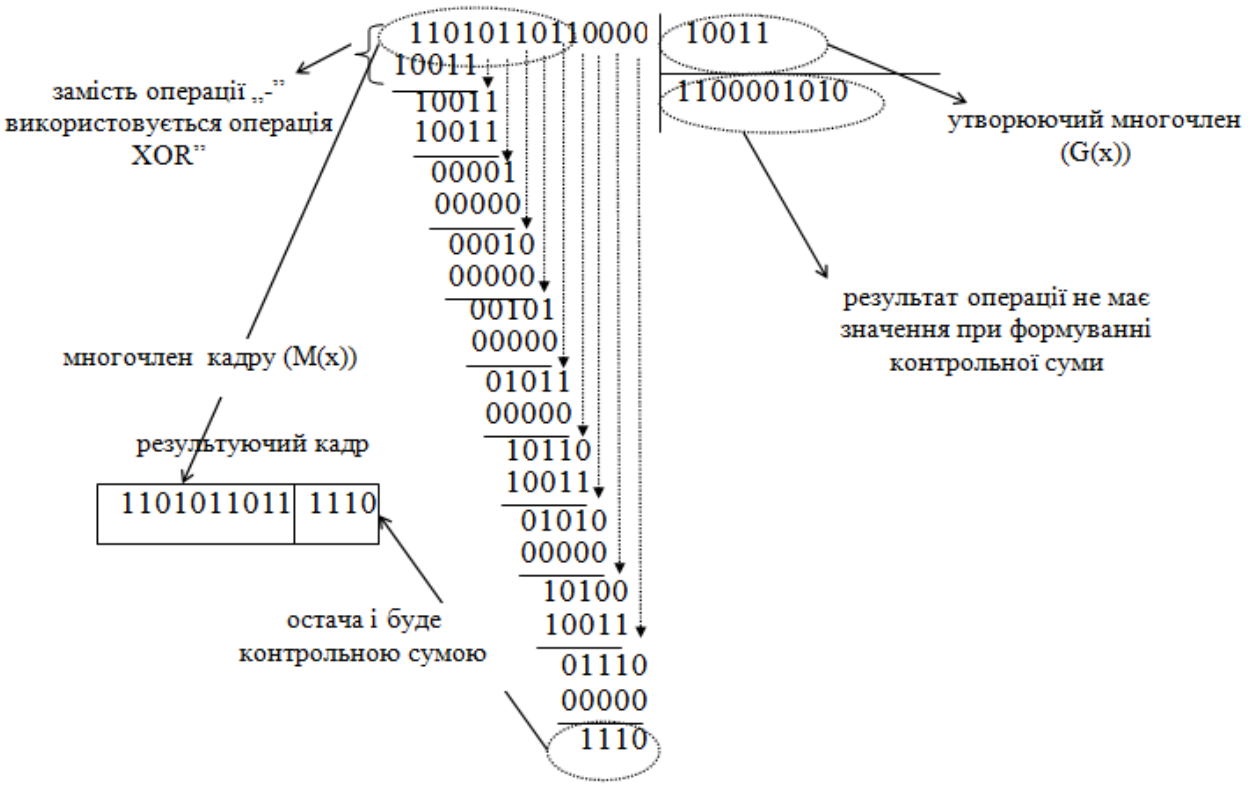
8.1 До приймання дипломної роботи надається:

- пояснювальна записка з відповідними узгодженнями;
- демонстрація програмного засобу;
- презентація;
- відгук керівника роботи та рецензента;

Технічне завдання до виконання прийняв _____ Гудименко О.О.

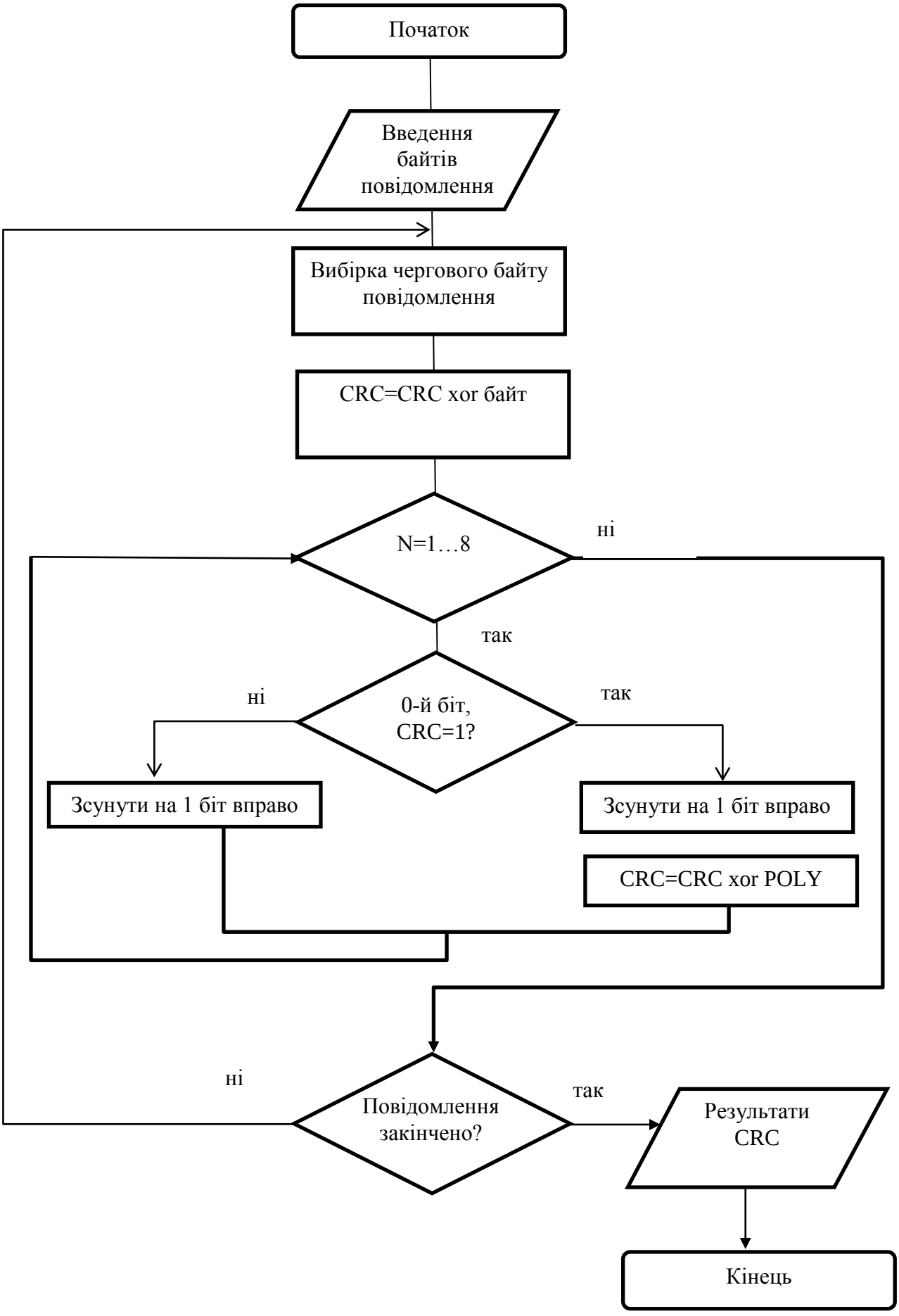
Додаток Б

Приклад розрахунку контрольної суми методом CRC16



Додаток В

Алгоритм розрахунку контрольної суми методом CRC16



Додаток Г

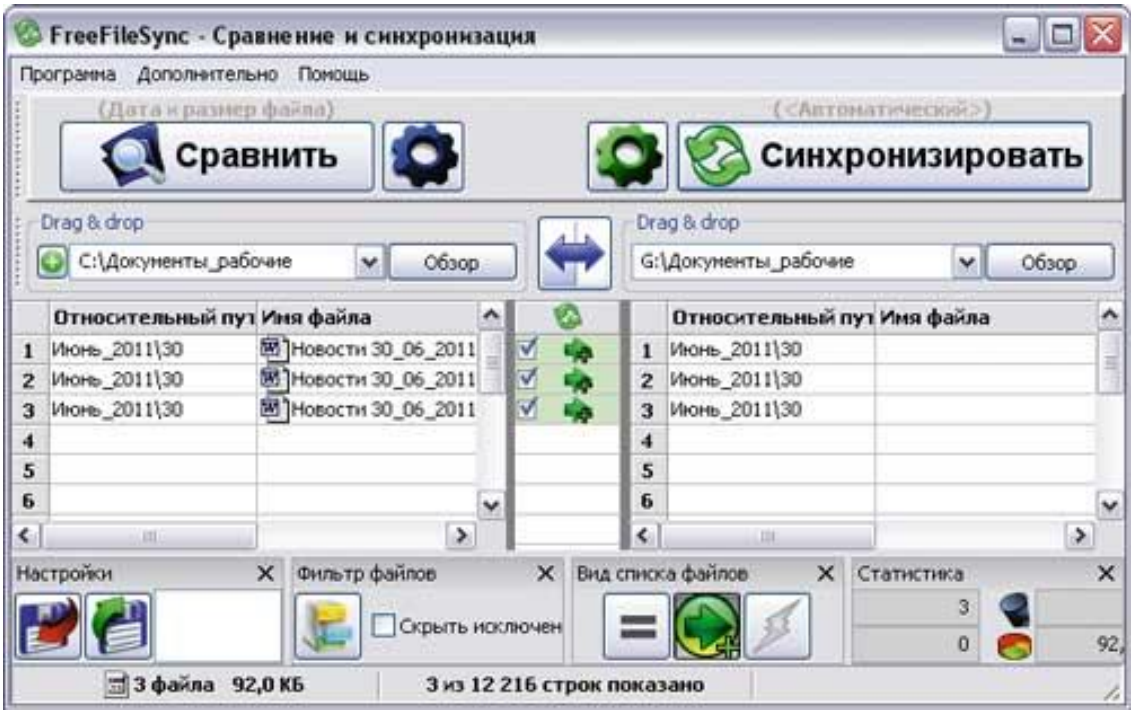
Результат аналізу в GoodSync

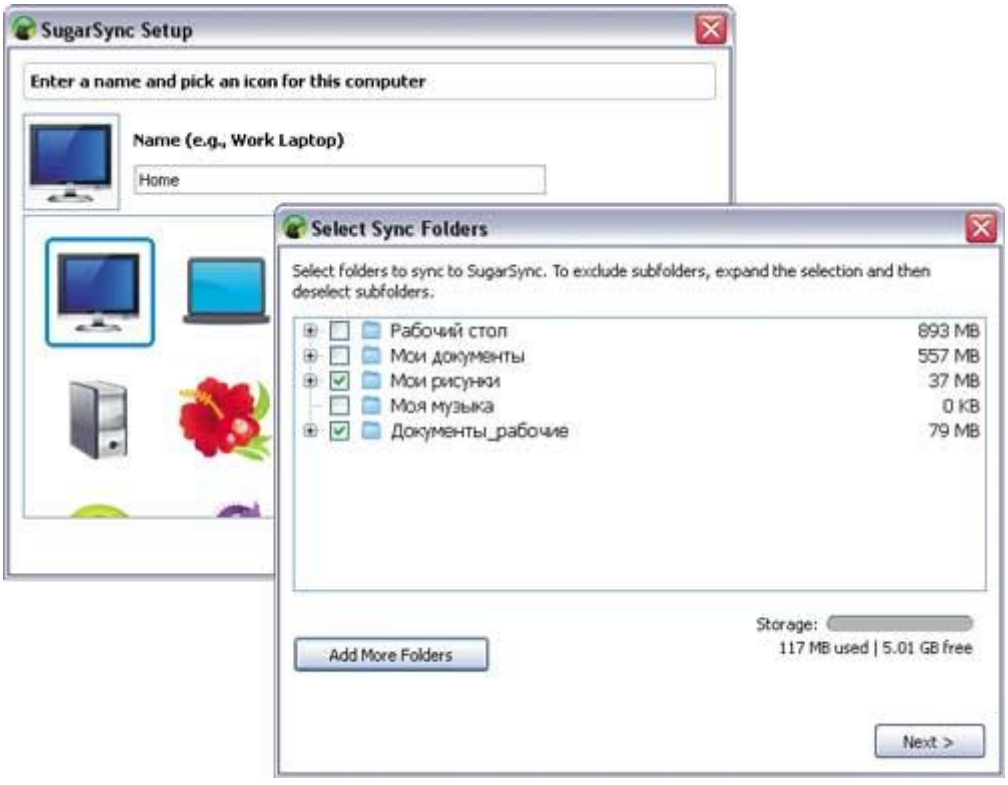
Додаток Д

Зовнішній вигляд Viceversa Pro та Allway Sync

Додаток Е

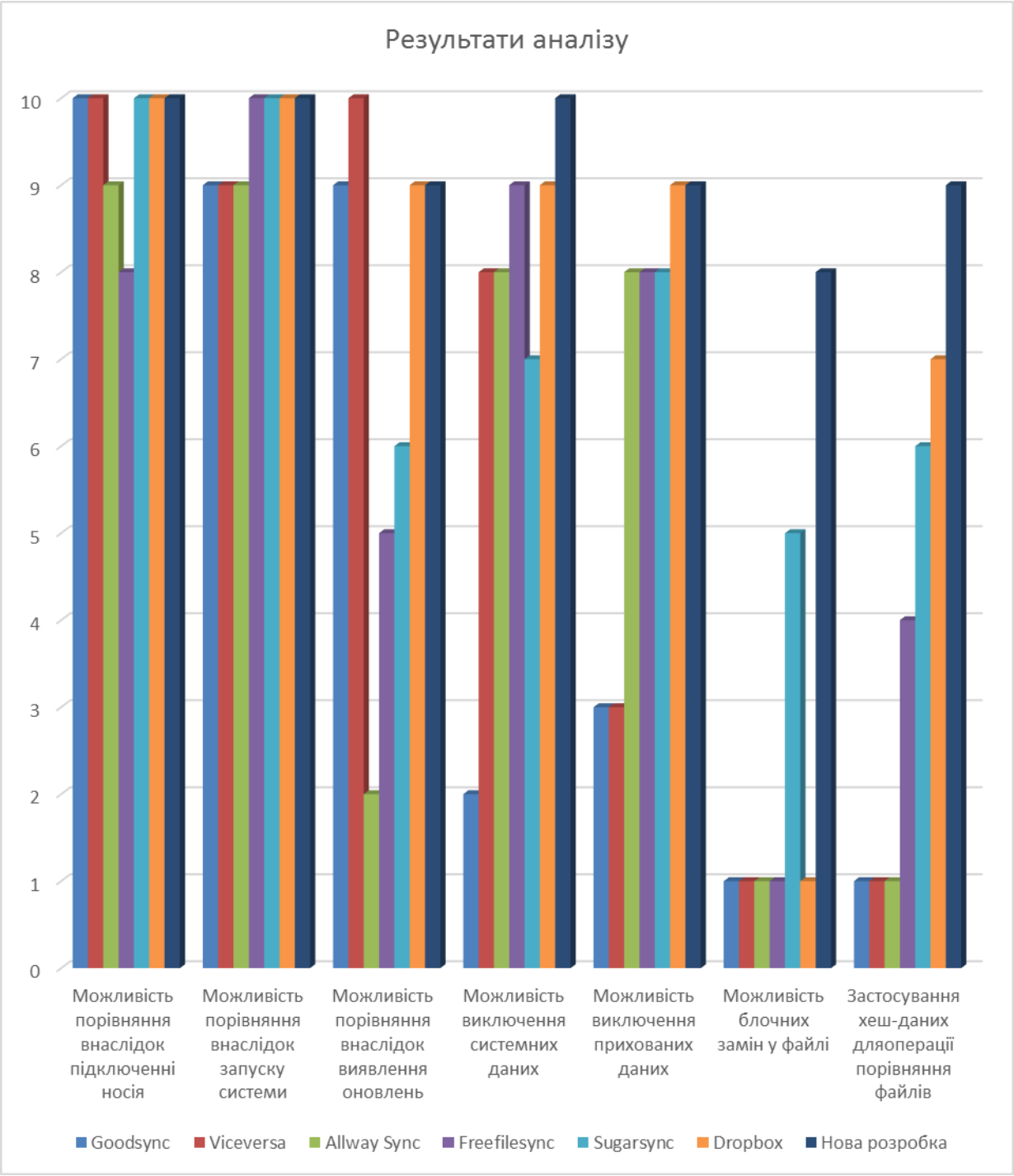
Інтерфейс Freefilesync та налаштування кліента в SugarSync





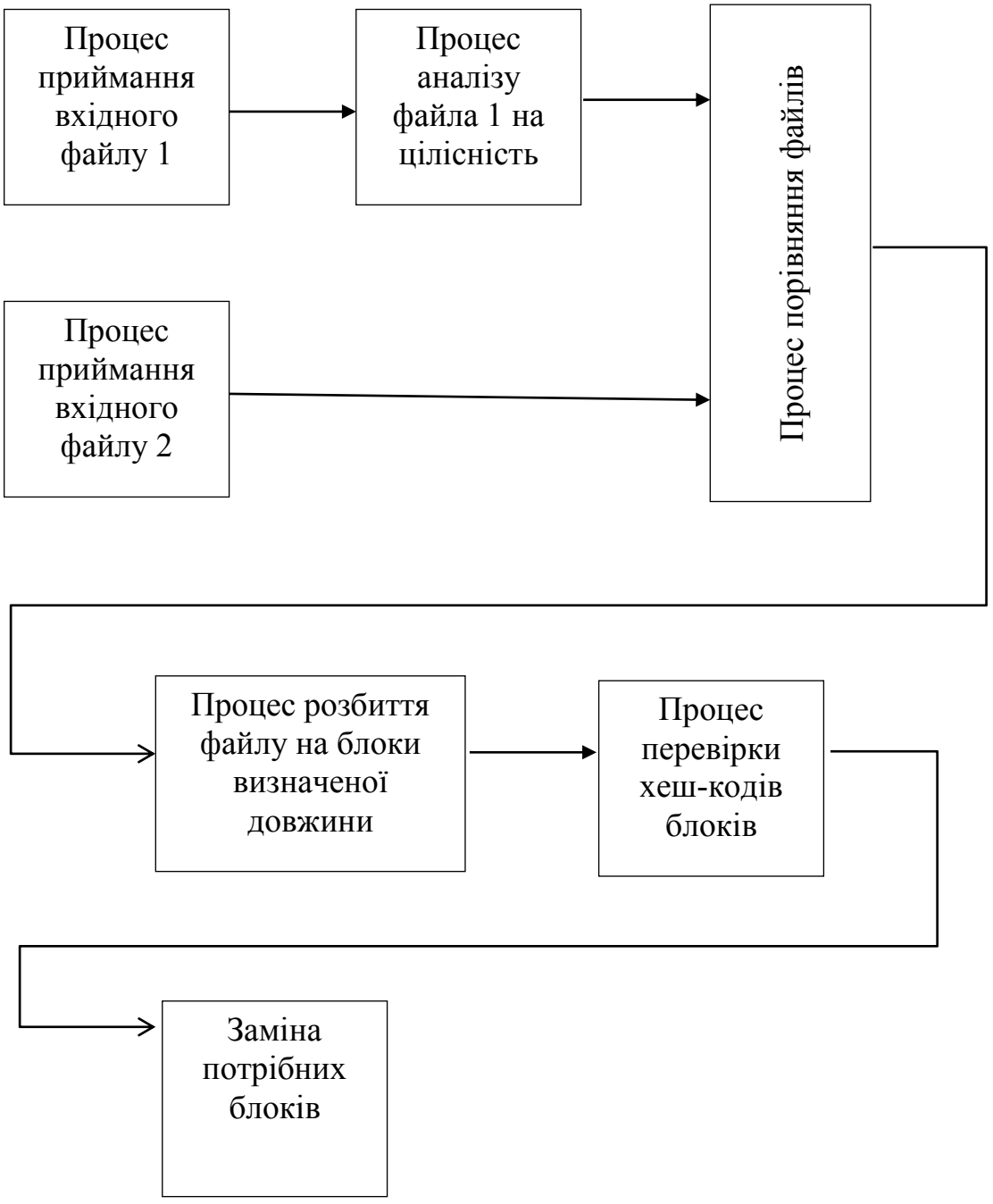
Додаток Ж

Оцінка конкурентів



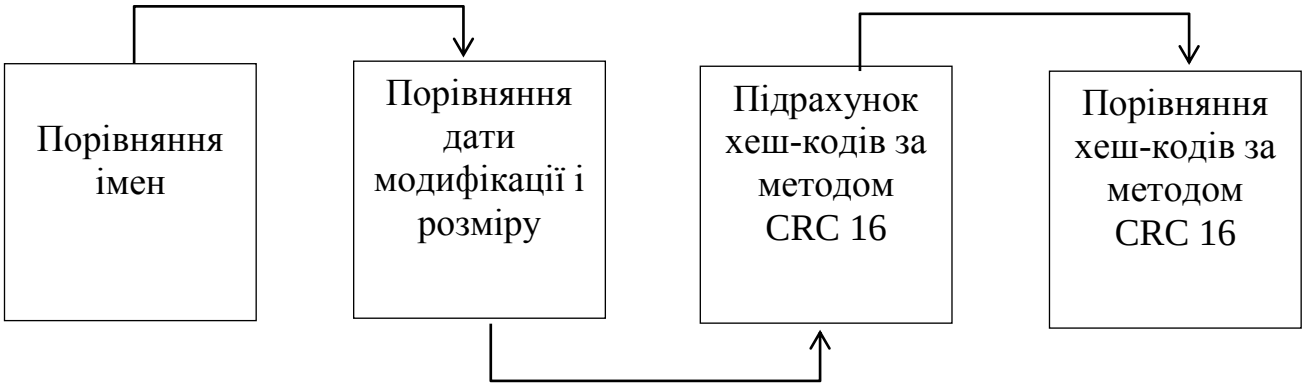
Додаток К

Схематичне представлення роботи методу блочного порівнння файлів



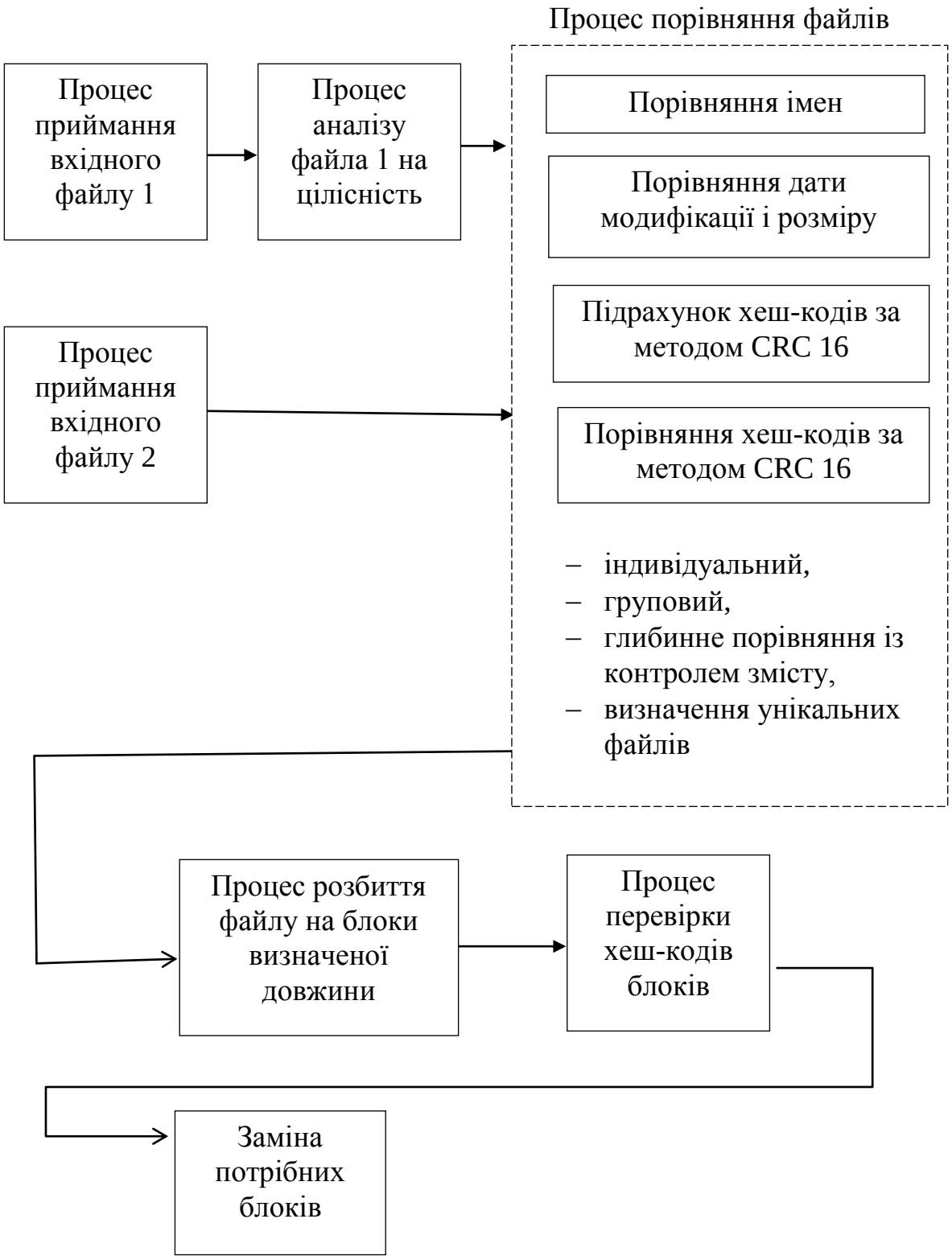
Додаток Л

Схематичне представлення роботи процесу порівнння файлів



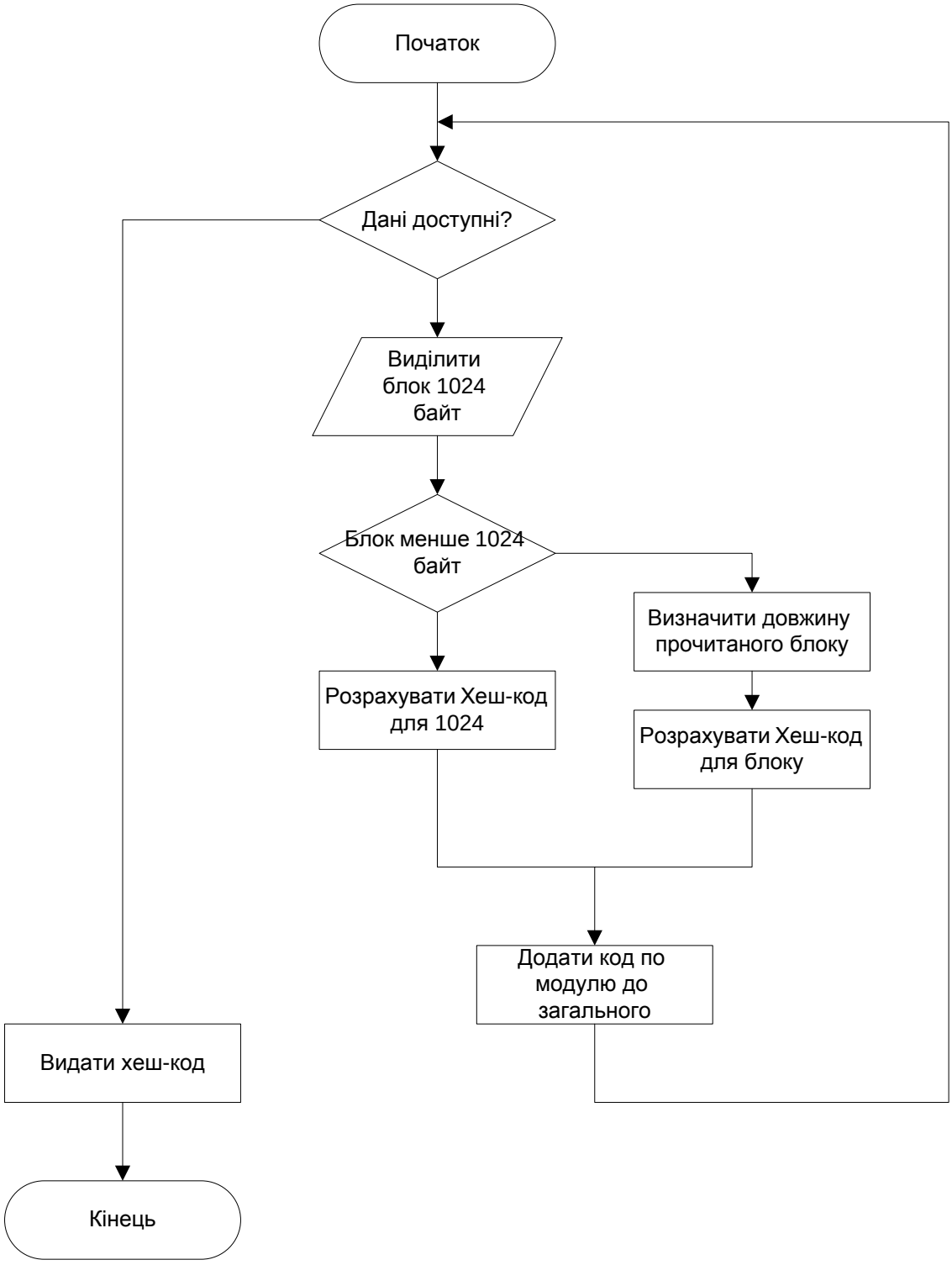
Додаток М

Схематичне розширене представлення роботи методу блочного порівнння файлів



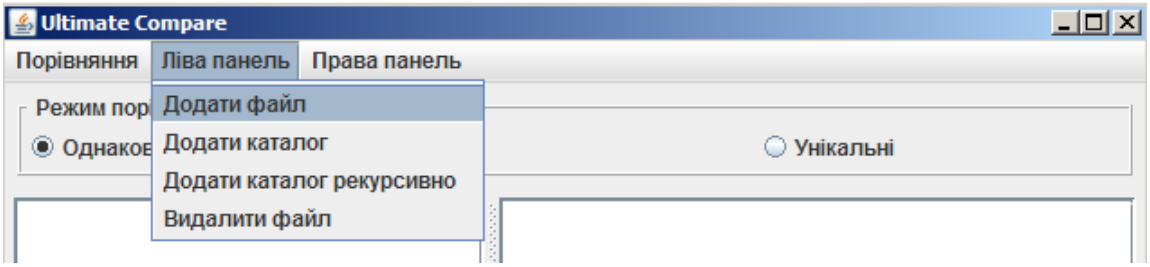
Додаток Н

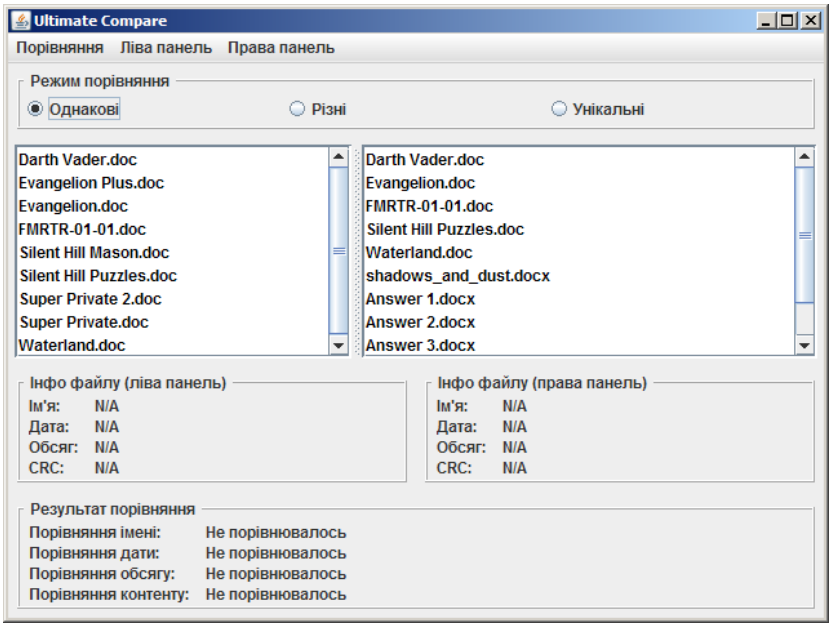
Блок схема роботи алгоритму процесів підрахунку та порівняння хеш-кодів за методом CRC 16

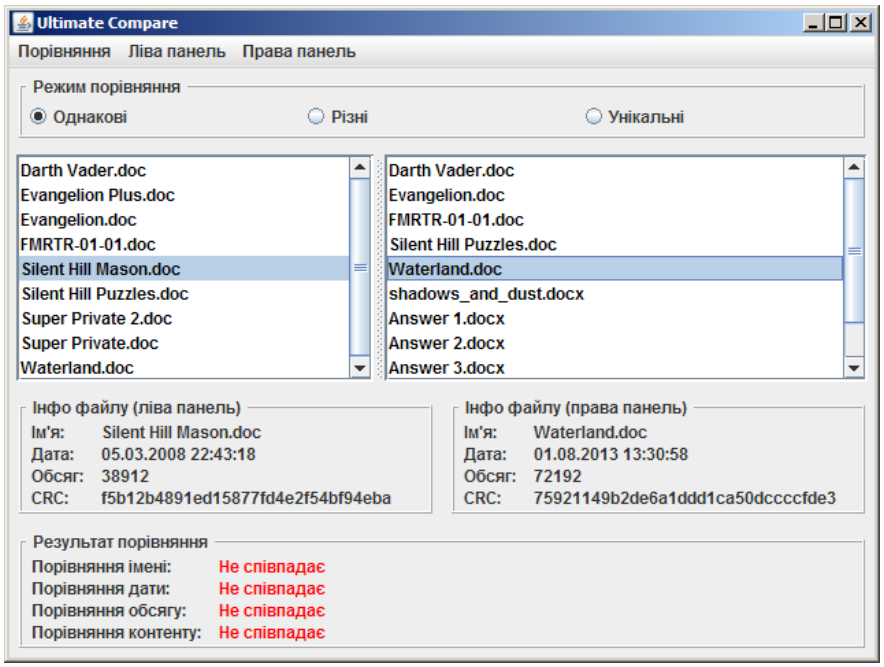


Додаток П

Тестування і перевірка правильності роботи програми







Додаток Р

Лістинг програми

```
package ua.richesa.tools.compare;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import java.security.MessageDigest;

import java.text.SimpleDateFormat;

import javax.swing.ButtonGroup;
import javax.swing.DefaultListModel;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JSeparator;
import javax.swing.JSplitPane;
import javax.swing.border.CompoundBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.EtchedBorder;
import javax.swing.border.TitledBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

public class UltimateCompare extends JFrame
 implements ActionListener, ListSelectionListener
{
 private static final String STATUS_EQUAL       = "Співпадає";
```

```
private static final String STATUS_NOTEQUAL    = "Не співпадає";
private static final String STATUS_NONCOMPARED = "Не порівнювалось";
private static final String STATUS_NA          = "N/A";

private JList leftWing = new JList(new DefaultListModel());
private JList rightWing = new JList(new DefaultListModel());

private JLabel nameInfoLeft = new JLabel(STATUS_NA);
private JLabel dateInfoLeft = new JLabel(STATUS_NA);
private JLabel sizeInfoLeft = new JLabel(STATUS_NA);
private JLabel contentInfoLeft = new JLabel(STATUS_NA);

private JLabel nameInfoRight = new JLabel(STATUS_NA);
private JLabel dateInfoRight = new JLabel(STATUS_NA);
private JLabel sizeInfoRight = new JLabel(STATUS_NA);
private JLabel contentInfoRight = new JLabel(STATUS_NA);

private JLabel nameCompare = new JLabel(STATUS_NONCOMPARED);
private JLabel dateCompare = new JLabel(STATUS_NONCOMPARED);
private JLabel sizeCompare = new JLabel(STATUS_NONCOMPARED);
private JLabel contentCompare = new JLabel(STATUS_NONCOMPARED);

private JRadioButton modeEqual     = new JRadioButton("Однакові");
private JRadioButton modeDifferent = new JRadioButton("Різні");
private JRadioButton modeOriginal  = new JRadioButton("Унікальні");

public class FileRecord
{
 public File currentFile;
 public boolean marked = false;

 public FileRecord(File curF) {currentFile = curF;}
 public String toString() {return currentFile.getName();}
}

public UltimateCompare()
{
 super();
 setTitle("Ultimate Compare");
 setSize(640, 480);

 leftWing.setName("LeftWing");
 leftWing.addListSelectionListener(this);
 leftWing.setCellRenderer(new ComparatorListRenderer());
 rightWing.setName("RightWing");
 rightWing.addListSelectionListener(this);
 rightWing.setCellRenderer(new ComparatorListRenderer());
```

```
  JPanel mainPane = new JPanel(new BorderLayout());

  JMenuBar menuBar = new JMenuBar();

  JMenu submenu = new JMenu("Порівняння");
  submenu.add(createMenuItem("Нове порівняння", "newCompare"));
  submenu.add(new JSeparator());
  submenu.add(createMenuItem("Просте порівняння", "simpleCompare"));
  submenu.add(createMenuItem("Синхронізація", "deepCompare"));
  submenu.add(createMenuItem("Знайти дублікати", "findDoubles"));
  submenu.add(createMenuItem("Очистити позначки", "clearCompare"));
  submenu.add(new JSeparator());
  submenu.add(createMenuItem("Вихід", "quit"));
  menuBar.add(submenu);

  submenu = new JMenu("Ліва панель");
  submenu.add(createMenuItem("Додати файл", "addLeftFile"));
  submenu.add(createMenuItem("Додати каталог", "addLeftDirectory"));
  submenu.add(createMenuItem("Додати каталог рекурсивно",
"addLeftDirectoryRecurse"));
  submenu.add(createMenuItem("Видалити файл", "removeLeftFile"));
  menuBar.add(submenu);

  submenu = new JMenu("Права панель");
  submenu.add(createMenuItem("Додати файл", "addRightFile"));
  submenu.add(createMenuItem("Додати каталог", "addRightDirectory"));
  submenu.add(createMenuItem("Додати каталог рекурсивно",
"addRightDirectoryRecurse"));
  submenu.add(createMenuItem("Видалити файл", "removeRightFile"));
  menuBar.add(submenu);

  setJMenuBar(menuBar);

  ButtonGroup modes = new ButtonGroup();
  modes.add(modeEqual);
  modes.add(modeDifferent);
  modes.add(modeOriginal);
  modeEqual.setSelected(true);

  JPanel buttonPane = new JPanel(new GridLayout(1, 3));
  buttonPane.add(modeEqual);
  buttonPane.add(modeDifferent);
  buttonPane.add(modeOriginal);
  buttonPane.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5),
                       new TitledBorder(new EtchedBorder(EtchedBorder.RAISED), "
Режим порівняння ")));
  mainPane.add(buttonPane, BorderLayout.NORTH);
```

```
  JPanel wrapPane = new JPanel(new BorderLayout());

  JSplitPane comparePane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                                new JScrollPane(leftWing), new JScrollPane(rightWing));
  wrapPane.add(comparePane, BorderLayout.CENTER);
  wrapPane.setBorder(new EmptyBorder(5, 5, 5, 5));
  mainPane.add(wrapPane, BorderLayout.CENTER);

  JPanel statusPane = new JPanel(new BorderLayout());
  JPanel innerPane = new JPanel(new GridLayout(1, 2));

  // Left information pane - begins
  JPanel infoPane = new JPanel(new BorderLayout());

  JPanel wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(new JLabel("Ім'я: "));
  wingPane.add(new JLabel("Дата: "));
  wingPane.add(new JLabel("Обсяг: "));
  wingPane.add(new JLabel("CRC: "));
  wingPane.setBorder(new EmptyBorder(0, 5, 0, 10));
  infoPane.add(wingPane, BorderLayout.WEST);

  wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(nameInfoLeft);
  wingPane.add(dateInfoLeft);
  wingPane.add(sizeInfoLeft);
  wingPane.add(contentInfoLeft);
  infoPane.add(wingPane, BorderLayout.CENTER);
  infoPane.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5),
                                new   TitledBorder(new   EtchedBorder(EtchedBorder.RAISED),   "
Інфо файлу (ліва панель) ")));
  innerPane.add(infoPane);
  // Left information pane - ends

  // Right information pane - begins
  infoPane = new JPanel(new BorderLayout());

  wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(new JLabel("Ім'я: "));
  wingPane.add(new JLabel("Дата: "));
  wingPane.add(new JLabel("Обсяг: "));
  wingPane.add(new JLabel("CRC: "));
  wingPane.setBorder(new EmptyBorder(0, 5, 0, 10));
  infoPane.add(wingPane, BorderLayout.WEST);

  wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(nameInfoRight);
  wingPane.add(dateInfoRight);
```

```
  wingPane.add(sizeInfoRight);
  wingPane.add(contentInfoRight);
  infoPane.add(wingPane, BorderLayout.CENTER);
  infoPane.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5),
                                new   TitledBorder(new   EtchedBorder(EtchedBorder.RAISED),   "
Інфо файлу (права панель) ")));
  innerPane.add(infoPane);
  // Right information pane - ends

  statusPane.add(innerPane, BorderLayout.CENTER);

  infoPane = new JPanel(new BorderLayout());

  wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(new JLabel("Порівняння імені: "));
  wingPane.add(new JLabel("Порівняння дати: "));
  wingPane.add(new JLabel("Порівняння обсягу: "));
  wingPane.add(new JLabel("Порівняння контенту: "));
  wingPane.setBorder(new EmptyBorder(0, 5, 0, 10));
  infoPane.add(wingPane, BorderLayout.WEST);

  wingPane = new JPanel(new GridLayout(4, 1));
  wingPane.add(nameCompare);
  wingPane.add(dateCompare);
  wingPane.add(sizeCompare);
  wingPane.add(contentCompare);
  infoPane.add(wingPane, BorderLayout.CENTER);
  infoPane.setBorder(new CompoundBorder(new EmptyBorder(5, 5, 5, 5),
                                new   TitledBorder(new   EtchedBorder(EtchedBorder.RAISED),   "
Результат порівняння ")));
  statusPane.add(infoPane, BorderLayout.SOUTH);

  mainPane.add(statusPane, BorderLayout.SOUTH);

  getContentPane().setLayout(new BorderLayout());
  getContentPane().add(mainPane, BorderLayout.CENTER);

  addWindowListener(new WindowAdapter()
  {
   public void windowClosing(WindowEvent e)
   {
    System.exit(0);
   }
  });
 }

 private void addRecurseFiles(String root, DefaultListModel model)
 throws IOException
```

```
{
 File current = new File(root);
 File[] innerFiles = current.listFiles();
 for(int i = 0; i < innerFiles.length; i++)
 {
  if(innerFiles[i].isDirectory())
   addRecurseFiles(innerFiles[i].getCanonicalPath(), model);
  else
   model.addElement(new FileRecord(innerFiles[i]));
 }
}

private void addFiles(boolean wing, boolean directory)
{
 JFileChooser chooser = new JFileChooser();
 if(directory)
  chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
 else
  chooser.setMultiSelectionEnabled(true);
 if(chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
 {
  DefaultListModel model = null;
  if(wing)
   model = (DefaultListModel) rightWing.getModel();
  else
   model = (DefaultListModel) leftWing.getModel();

  if(directory)
  {
   File[] selected = chooser.getSelectedFile().listFiles();
   for(int i = 0; i < selected.length; i++)
    model.addElement(new FileRecord(selected[i]));
  }
  else
  {
   File[] selected = chooser.getSelectedFiles();
   for(int i = 0; i < selected.length; i++)
    model.addElement(new FileRecord(selected[i]));
  }
 }
}

private void removeFiles(boolean wing)
{
 int[] selectedOnes = null;
 DefaultListModel model = null;
 if(wing)
 {
```

```
  model = (DefaultListModel)rightWing.getModel();
  selectedOnes = rightWing.getSelectedIndices();
 }
 else
 {
  model = (DefaultListModel)leftWing.getModel();
  selectedOnes = leftWing.getSelectedIndices();
 }
 while(selectedOnes.length > 0)
 {
  model.removeElementAt(selectedOnes[0]);
  if(wing)
   selectedOnes = rightWing.getSelectedIndices();
  else
   selectedOnes = leftWing.getSelectedIndices();
 }
}

private JMenuItem createMenuItem(String name, String command)
{
 JMenuItem newItem = new JMenuItem(name);
 newItem.setActionCommand(command);
 newItem.addActionListener(this);
 return newItem;
}

private byte[] createChecksum(String filename)
throws Exception
{
 InputStream fis =  new FileInputStream(filename);

 byte[] buffer = new byte[1024];
 MessageDigest complete = MessageDigest.getInstance("MD5");
 int numRead;
 do
 {
  numRead = fis.read(buffer);
  if (numRead > 0)
  {
   complete.update(buffer, 0, numRead);
  }
 } while (numRead != -1);
 fis.close();
 return complete.digest();
}

private String createCRC(String filename)
throws Exception
```

```
{
 byte[] b = createChecksum(filename);
 String result = "";
 for (int i=0; i < b.length; i++)
 {
  result +=
  Integer.toString( ( b[i] & 0xff ) + 0x100, 16).substring( 1 );
 }
 return result;
}

private void invertMarks()
{
 DefaultListModel leftModel = (DefaultListModel)leftWing.getModel();
 DefaultListModel rightModel = (DefaultListModel)rightWing.getModel();

 for(int i = 0; i < leftModel.getSize(); i++)
 {
  FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
  one.marked = !one.marked;
 }

 for(int i = 0; i < rightModel.getSize(); i++)
 {
  FileRecord one = (UltimateCompare.FileRecord) rightModel.getElementAt(i);
  one.marked = !one.marked;
 }
}

private void performSimpleComparation()
throws Exception
{
 DefaultListModel leftModel = (DefaultListModel)leftWing.getModel();
 DefaultListModel rightModel = (DefaultListModel)rightWing.getModel();

 if(modeEqual.isSelected())
 {
  for(int i = 0; i < leftModel.getSize(); i++)
  {
   for(int j = 0; j < rightModel.getSize(); j++)
   {
    FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
    FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(j);
    if(one.currentFile.getName().equals(two.currentFile.getName()))
    {
     one.marked = true;
     two.marked = true;
    }
```

```
  }
 }
}

if(modeDifferent.isSelected())
{
 for(int i = 0; i < leftModel.getSize(); i++)
 {
  for(int j = 0; j < rightModel.getSize(); j++)
  {
   FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
   FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(j);
   if(one.currentFile.getName().equals(two.currentFile.getName()))
   {
    one.marked = true;
    two.marked = true;
   }
  }
 }
 invertMarks();
}

if(modeOriginal.isSelected())
{
 invertMarks();

 for(int i = 0; i < leftModel.getSize(); i++)
 {
  FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
  String crcOne = createCRC(one.currentFile.getCanonicalPath());
  for(int j = 0; j < rightModel.getSize(); j++)
  {
   FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(j);
   if(crcOne.equals(createCRC(two.currentFile.getCanonicalPath())))
   {
    one.marked = false;
    two.marked = false;
   }
  }
 }

 for(int i = 0; i < rightModel.getSize(); i++)
 {
  FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(i);
  String crcTwo = createCRC(two.currentFile.getCanonicalPath());
  for(int j = 0; j < leftModel.getSize(); j++)
  {
   FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(j);
```

```
    if(crcTwo.equals(createCRC(one.currentFile.getCanonicalPath())))
    {
     one.marked = false;
     two.marked = false;
    }
   }
  }

 }

 forceRevalidate();
}

private void performSync()
throws Exception
{
 DefaultListModel leftModel = (DefaultListModel)leftWing.getModel();
 DefaultListModel rightModel = (DefaultListModel)rightWing.getModel();

 if(modeEqual.isSelected())
 {
  for(int i = 0; i < leftModel.getSize(); i++)
  {
   FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
   String crcOne = createCRC(one.currentFile.getCanonicalPath());
   for(int j = 0; j < rightModel.getSize(); j++)
   {
    FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(j);
    if(one.currentFile.getName().equals(two.currentFile.getName()) &&
       (one.currentFile.lastModified() == two.currentFile.lastModified()) &&
       (one.currentFile.length() == two.currentFile.length()) &&
       crcOne.equals(createCRC(two.currentFile.getCanonicalPath())))
    {
     one.marked = true;
     two.marked = true;
    }
   }
  }
 }

 if(modeDifferent.isSelected())
 {
  for(int i = 0; i < leftModel.getSize(); i++)
  {
   FileRecord one = (UltimateCompare.FileRecord) leftModel.getElementAt(i);
   String crcOne = createCRC(one.currentFile.getCanonicalPath());
   for(int j = 0; j < rightModel.getSize(); j++)
   {
```

```
    FileRecord two = (UltimateCompare.FileRecord) rightModel.getElementAt(j);
    if(one.currentFile.getName().equals(two.currentFile.getName()) &&
       (one.currentFile.lastModified() == two.currentFile.lastModified()) &&
       (one.currentFile.length() == two.currentFile.length()) &&
       crcOne.equals(createCRC(two.currentFile.getCanonicalPath())))
    {
     one.marked = true;
     two.marked = true;
    }
   }
  }
  invertMarks();
 }

 forceRevalidate();
}

private void performFindDoubles()
{
 try
 {
  DefaultListModel rightModel = (DefaultListModel)rightWing.getModel();

  FileRecord one = (UltimateCompare.FileRecord) leftWing.getSelectedValue();
  String crcOne = createCRC(one.currentFile.getCanonicalPath());

  for(int i = 0; i < rightModel.getSize(); i++)
  {
   FileRecord two = (UltimateCompare.FileRecord) rightModel.elementAt(i);
   if(crcOne.equals(createCRC(two.currentFile.getCanonicalPath())))
    two.marked = true;
  }
 } catch (Exception e)
 {
  System.err.println(e.getMessage());
  e.printStackTrace(System.err);
 }
 forceRevalidate();
}

public void actionPerformed(ActionEvent e)
{
 if(e.getActionCommand().equals("newCompare"))
 {
  ((DefaultListModel)leftWing.getModel()).clear();
  ((DefaultListModel)rightWing.getModel()).clear();

  nameInfoLeft.setText(STATUS_NA);
```

```
 dateInfoLeft.setText(STATUS_NA);
 sizeInfoLeft.setText(STATUS_NA);
 contentInfoLeft.setText(STATUS_NA);

 nameInfoRight.setText(STATUS_NA);
 dateInfoRight.setText(STATUS_NA);
 sizeInfoRight.setText(STATUS_NA);
 contentInfoRight.setText(STATUS_NA);

 setStatus(nameCompare, STATUS_NONCOMPARED);
 setStatus(dateCompare, STATUS_NONCOMPARED);
 setStatus(sizeCompare, STATUS_NONCOMPARED);
 setStatus(contentCompare, STATUS_NONCOMPARED);
}

if(e.getActionCommand().equals("clearCompare"))
{
 DefaultListModel model = (DefaultListModel)leftWing.getModel();
 for(int i = 0; i < model.getSize(); i++)
 {
  FileRecord current = (UltimateCompare.FileRecord) model.getElementAt(i);
  current.marked = false;
 }
 forceRevalidate();
}

if(e.getActionCommand().equals("addLeftFile"))
 addFiles(false, false);
if(e.getActionCommand().equals("addLeftDirectory"))
 addFiles(false, true);

if(e.getActionCommand().equals("addLeftDirectoryRecurse"))
{
 try
 {
  DefaultListModel model = (DefaultListModel) leftWing.getModel();
  JFileChooser chooser = new JFileChooser();
  chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
  if(chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
   addRecurseFiles(chooser.getSelectedFile().getCanonicalPath(), model);
 } catch (IOException exc)
 {
  System.err.println(exc.getMessage());
  exc.printStackTrace(System.err);
 }
}

if(e.getActionCommand().equals("removeLeftFile"))
```

```
 removeFiles(false);
if(e.getActionCommand().equals("addRightFile"))
 addFiles(true, false);

if(e.getActionCommand().equals("addRightDirectory"))
 addFiles(true, true);
if(e.getActionCommand().equals("removeRightFile"))
 removeFiles(true);

if(e.getActionCommand().equals("addRightDirectoryRecurse"))
{
 try
 {
  DefaultListModel model = (DefaultListModel) rightWing.getModel();
  JFileChooser chooser = new JFileChooser();
  chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
  if(chooser.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
   addRecurseFiles(chooser.getSelectedFile().getCanonicalPath(), model);
 } catch (IOException exc)
 {
  System.err.println(exc.getMessage());
  exc.printStackTrace(System.err);
 }
}

if(e.getActionCommand().equals("simpleCompare"))
{
 try
 {
  performSimpleComparation();
 } catch (Exception exc)
 {
  System.err.println(exc.getMessage());
  exc.printStackTrace(System.err);
 }
}

if(e.getActionCommand().equals("deepCompare"))
 {
  try
  {
   performSync();
  } catch (Exception exc)
  {
   System.err.println(exc.getMessage());
   exc.printStackTrace(System.err);
  }
 }
```

```
 if(e.getActionCommand().equals("findDoubles"))
  performFindDoubles();
}

private void setStatus(JLabel what, String constant)
{
 what.setForeground(Color.black);
 if(constant.equals(STATUS_EQUAL))
  what.setForeground(Color.green);
 if(constant.equals(STATUS_NOTEQUAL))
  what.setForeground(Color.red);
 what.setText(constant);
}

public void valueChanged(ListSelectionEvent e)
{
 try
 {
  JList current = (JList)e.getSource();
  if(current.getSelectedValue() == null)
   return;
  SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss");
  if(current.getName().equals("LeftWing"))
  {
   File currentFile = ((FileRecord)leftWing.getSelectedValue()).currentFile;
   nameInfoLeft.setText(currentFile.getName());
   dateInfoLeft.setText(format.format(currentFile.lastModified()));
   sizeInfoLeft.setText(String.valueOf(currentFile.length()));
   contentInfoLeft.setText(createCRC(currentFile.getCanonicalPath()));

   if(rightWing.getSelectedValue() != null)
   {
    File pretender = ((FileRecord)rightWing.getSelectedValue()).currentFile;

    if(currentFile.getName().equals(pretender.getName()))
     setStatus(nameCompare, STATUS_EQUAL);
    else
     setStatus(nameCompare, STATUS_NOTEQUAL);

    if(currentFile.lastModified() == pretender.lastModified())
     setStatus(dateCompare, STATUS_EQUAL);
    else
     setStatus(dateCompare, STATUS_NOTEQUAL);

    if(currentFile.length() == pretender.length())
     setStatus(sizeCompare, STATUS_EQUAL);
    else
```

```
      setStatus(sizeCompare, STATUS_NOTEQUAL);


if(createCRC(currentFile.getCanonicalPath()).equals(createCRC(pretender.getCanonicalPath
()))
      setStatus(contentCompare, STATUS_EQUAL);
     else
      setStatus(contentCompare, STATUS_NOTEQUAL);
    }
   }
   else
   {
    File currentFile = ((FileRecord)rightWing.getSelectedValue()).currentFile;
    nameInfoRight.setText(currentFile.getName());
    dateInfoRight.setText(format.format(currentFile.lastModified()));
    sizeInfoRight.setText(String.valueOf(currentFile.length()));
    contentInfoRight.setText(createCRC(currentFile.getCanonicalPath()));

    if(leftWing.getSelectedValue() != null)
    {
     File pretender = ((FileRecord)leftWing.getSelectedValue()).currentFile;

     if(currentFile.getName().equals(pretender.getName()))
      setStatus(nameCompare, STATUS_EQUAL);
     else
      setStatus(nameCompare, STATUS_NOTEQUAL);

     if(currentFile.lastModified() == pretender.lastModified())
      setStatus(dateCompare, STATUS_EQUAL);
     else
      setStatus(dateCompare, STATUS_NOTEQUAL);

     if(currentFile.length() == pretender.length())
      setStatus(sizeCompare, STATUS_EQUAL);
     else
      setStatus(sizeCompare, STATUS_NOTEQUAL);

if(createCRC(currentFile.getCanonicalPath()).equals(createCRC(pretender.getCanonicalPath
()))
      setStatus(contentCompare, STATUS_EQUAL);
     else
      setStatus(contentCompare, STATUS_NOTEQUAL);
    }
   }
  } catch (Exception exc)
  {
   System.err.println(exc.getMessage());
   exc.printStackTrace(System.err);
```

```
  }
 }
 public void forceRevalidate()
 {
  validate();
  repaint();
 }

 public static void main(String[] args)
 {
  UltimateCompare ultimateCompare = new UltimateCompare();
  ultimateCompare.setVisible(true);
 }
}
package ua.richesa.tools.compare;

import java.awt.Color;
import java.awt.Component;

import javax.swing.DefaultListCellRenderer;
import javax.swing.JList;

public class ComparatorListRenderer
 extends DefaultListCellRenderer
{
 public ComparatorListRenderer()
 {
  super();
 }

 public Component getListCellRendererComponent(JList<?> list, Object value, int index,
boolean isSelected,
                                               boolean cellHasFocus)
 {

  DefaultListCellRenderer renderer =
   (DefaultListCellRenderer) super.getListCellRendererComponent(list, value, index,
isSelected, cellHasFocus);

  if(value instanceof UltimateCompare.FileRecord)
  {
   UltimateCompare.FileRecord curRecord = (UltimateCompare.FileRecord)value;
   if(curRecord.marked)
    renderer.setForeground(Color.red);
   else
    renderer.setForeground(Color.black);
  }
```

```
  return renderer;
 }
}
```