Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Довгополов О.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Довгополову Олександру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 13-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Довгополов О.С.
(прізвище та ініціали)

АНОТАЦІЯ

Довгополов О.С. Програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

Метою розробки є програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

Результат роботи – програмна реалізація системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: кібербезпека, резервне копіювання

ABSTRACT

Dovhopolov O.S. Cyber security backup system software with archiving to ensure integrity. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system of backup copy with archiving to ensure integrity.

The goal of the development is to create a cybersecurity backup system software with archiving to ensure integrity.

The result of the work is a software implementation of a cyber security backup system with archiving to ensure integrity.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10 environment.

Keywords: cyber security, backup

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

7z	–	формат заархівованого файлу
BCJ	–	попередня обробка файлів, що виконуються
Blu-ray	–	носій інформації
CD	–	носій інформації
DVD	–	носій інформації
LZH	–	формат заархівованого файлу
LZMA	–	алгоритм архівування
LZP	–	знаходження повторів в тексті
Rar	–	формат заархівованого файлу
REP	–	знаходження повторів
zip	–	формат заархівованого файлу

ВСТУП

Актуальність теми. Люди, що активно працюють за комп'ютером, незалежно від їхнього віку, соціального статусу й сфери професійної діяльності, використовують безліч різноманітних електронних документів. Програміст створює вихідні тексти програм, секретар друкує накази й службові записки, фотограф обробляє свої знімки. Навіть у маленької нетямущої дитини на комп'ютері багато важливих документів – файлів збереження стану ігор.

Документи, на створення яких були витрачені місяці, роки кропіткої роботи, можна втратити в лічені секунди, причому, відбутися це може в будь-який момент часу, по незалежним від вас причинам. На відновлення важливих даних іде багато сил, часу, здоров'я. На жаль, найчастіше, користувач починає замислюватися про створення резервних копій тільки після втрати коштовної інформації.

Розумним і недорогим способом надійного збереження важливих даних від втрати є регулярне резервне копіювання. Для подібних цілей існують спеціальні програми, які всю турботу про схоронність інформації беруть на себе.

Регулярне архівування дозволяє знизити обсяг файлів для резервного копіювання. У випадку останнього доводиться враховувати розходження між файлами й базами даних, а також – в організаційному плані – між циклом резервного копіювання й наданням даних.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем резервного копіювання з архівацією для забезпечення цілісності.

– Дослідження системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

– Програмна реалізація системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі резервного копіювання з архівацією для забезпечення цілісності.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Завдання резервного копіювання прості й очевидні:

– Забезпечити схоронність можливо більше свіжих робочих даних при втраті або ушкодженні основних носіїв робочої інформації компанії.

– Забезпечити можливість перегляду старих версій файлів, у тому числі файлів, уже видалених з локальної мережі.

– Забезпечити надійне зберігання архівних даних протягом установленого періоду часу.

При цьому до системи резервного копіювання пред'являються певні вимоги:

– Запобігти несанкціонованому доступу до службової інформації.

– Забезпечити задану регулярність створення резервних копій.

– Забезпечити заданий час доступу до запитаної архівної або робочої інформації.

1.2 Область застосування

При розробці проектів по архівуванню варто заздалегідь визначити найближчі й віддалені цілі: підвищення продуктивності, дотримання правових приписань (Compliance) і т.д. Приміром, якщо комусь зі співробітників дається завдання на доробку якогось пристрою, то йому відразу ж знадобляться певні документи, зокрема, сама нова версія відповідного файлу й інформація про те, де й ким він оброблявся в минулому. Якщо зображення не переведене в цифрову форму, а існує лише на папері й перебуває в іншому місці, то продуктивна робота неможлива. Однак навіть якщо все необхідне є в цифровому виді, але не

класифіковано й не постачено однозначними характеристиками, то зібрати інформацію в повному обсязі дуже важко. При пошуку на файлових серверах, як правило, дуже важко вибрати саму останню редакцію креслення й зрозуміти, хто із цим файлом працював раніше.

Якщо рішення для архівування повинне відповідати вимогам аудита, то обов'язково враховуються строки зберігання документів.

Архівування здійснюється на два основних типи носіїв. По-перше, це рішення з однократним записом і багаторазовим читанням (Write Once, Read Many, WORM) у вигляді автозавантажників дисків DVD: цифрова інформація записується на DVD, і її вже не можна змінити. Друга можливість – системи адресації зберігання по вмісту (Content Addressed Storage, CAS) зі спеціальними жорсткими дисками, коли для однозначної ідентифікації оцифрована інформація забезпечується індивідуальним відбитком (Fingerprint), що привласнюється архівуємому файлу.

Архівування документів

Спочатку, крім визначення вимог IT, варто з'ясувати, які документи потрібно архівувати. Чи потрібна класифікація, і якщо так, те по яких ознаках? Спеціалізоване ПЗ для архівування документів підтримує функції індексації й пошуку інформації, а їхній облік спрощується завдяки концепціям класифікації. Звичайно програмне забезпечення надає інтерфейси до різних стандартних додатків. Якщо потрібне довгострокове зберігання, треба подумати й про те, що через якийсь час то або інший додаток уже не буде використовуватися, тому знадобляться спеціальні конвертери або програми для перегляду документів. Конвертація у формати TIFF і PDF вважається надійною з погляду її відтворення в майбутньому, але вона повинна здійснюватися зі збереженням всіх метаданих і можливостей пошуку.

Резервне копіювання

Регулярне архівування дозволяє знизити обсяг файлів для резервного копіювання. У випадку останнього доводиться враховувати розходження між

файлами й базами даних, а також – в організаційному плані – між циклом резервного копіювання й наданням даних. Стандартно передбачається повне резервне копіювання (Full Backup) один раз у тиждень і додаткове щоденне інкрементальне копіювання. Останнє веде до того, що при повнім відновленні даних їхній обсяг збільшується, іноді дворазово. Тому зазначений спосіб найбільш придатний для невеликої кількості даних. При значних обсягах інтервал часу для відновлення обмежений, тому виконується лише щоденне повне резервне копіювання.

Застосування «синтетичного резервного копіювання» дозволяє уникнути необхідності щоденного повного перенесення даних із клієнта на резервний ландшафт. Замість цього спочатку здійснюється повне резервне копіювання, а потім – інкрементальне, коли застарілі дані повної копії заміняються на нові. Далі система дублює цю «синтетичну» резервну копію з кешу на стрічку у вигляді повної фізичної резервної копії.

У результаті в користувача повинна зберігатися остання повна резервна копія на дисковому кеші або ж найбільш близька за часом остання повна резервна копія на стрічці, включаючи всю інформацію й дані, необхідні для повного відновлення. Для підвищення продуктивності процесу резервного копіювання користувальницькі дані направляються в стрічкові бібліотеки через дисковий кеш. Крім стрічкових систем, все частіше зустрічаються дискові системи зберігання, здатні емулювати відповідні стрічкові системи (віртуальні стрічкові бібліотеки).

Резервне копіювання баз даних

При резервному копіюванні баз даних необхідно звертати увагу не тільки на створення віддаленої копії, але й на її погодженість. Як правило, бази даних містять великі обсяги інформації, тому процес резервного копіювання затягається надовго. Тим часом у вихідну базу даних можуть бути внесені зміни. Існує велика ймовірність того, що області з файлами, копії яких уже створені, зміняться, а в ще

не збережених сегментах з'являться нові метадані. Така резервна копія абсолютно марна – її не можна відновити до функціональної бази даних.

Для протидії цьому явищу постачальники відповідних рішень розробили інтерфейси оперативного резервного копіювання, такі як Oracle Recovery Manager (RMAN). Тепер разом з інтерфейсами до клієнтських програм резервного копіювання дані можна зберігати прямо – з бази даних на носій резервної копії. Одночасно механізми RMAN дозволяють зберегти все, що необхідно для забезпечення погодженості резервних копій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки резервного копіювання з архіваціею для забезпечення цілісності, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

У даному розділі будуть розглянуті програми, що дозволяють оперувати папками й конкретними файлами. Загальний принцип роботи всіх учасників огляду ідентичний – ви створюєте завдання, у яку включаєте найбільш важливі документи, після чого плануєте графіка її запусків.

ABC Backup Pro

ABC Backup Pro – дуже простій, доступний продукт для створення резервних копій. Відповідно до твердження розроблювачів, освоєння їхньої програми не тяжчий завчання перших трьох букв латинського алфавіту. Настроювання всіх завдань здійснюється шляхом проходження від чотирьох до семи кроків, залежно від типу завдання. Вся процедура структурована за допомогою Майстра.

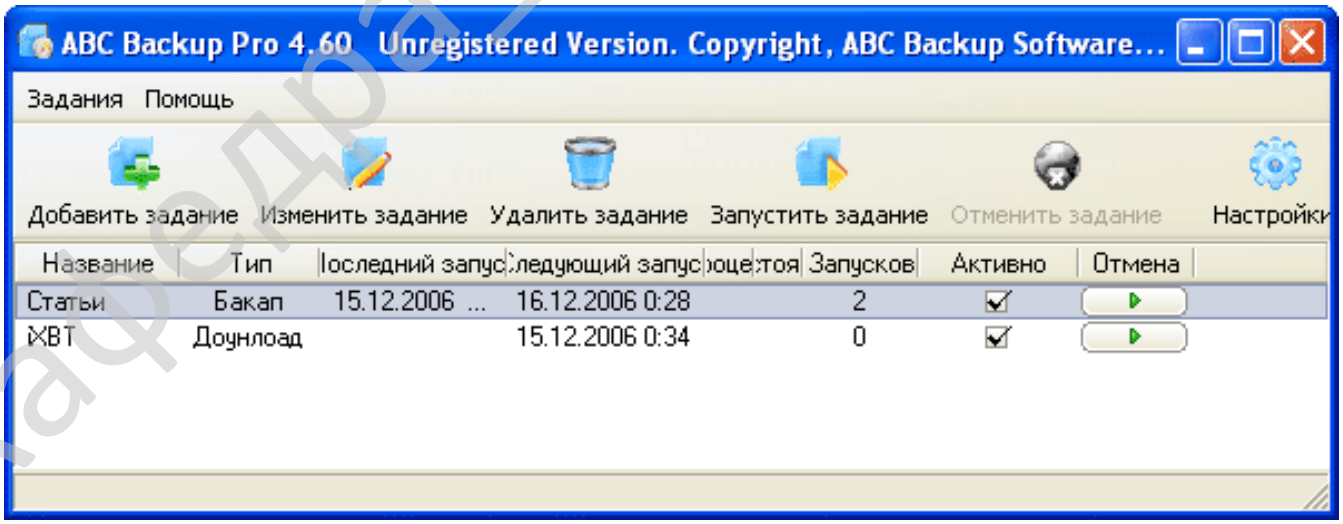

Рисунок 2.1 – Робоче вікно ABC Backup Pro

Робоче вікно додатка складається із простої панелі інструментів і лінійного списку користувальницьких завдань. Ви можете бачити в списку тип кожного завдання, інформацію про час останнього й найближчого виконання, індикатор стану завдання, кількість успішних запусків.

Уведення нового завдання здійснюється за допомогою кнопки «Додати завдання». При цьому відкривається робоче вікно із сімома вкладками, перемикання між якими зручно робити за допомогою кнопки Next. Як тільки ви коректно введете всі настроювання поточної групи опцій, дана кнопка стає активної, і ви можете перейти до наступного етапу.

На першому етапі від вас потрібно ввести тип завдання і його пріоритет. Програма підтримує створення резервних копій на жорсткому диску й у межах локальної мережі, відновлення даних, створення дзеркала FTP-ресурсу, прийом і віддачу файлів за протоколом FTP, копіювання інформації на оптичні носії. Найбільш затребуваний тип операцій – створення резервної копії, і на ньому варто зупинитися більш детально.

Спочатку вам необхідно вибрати джерело копіювання файлів – локальна файлова система або FTP-сервер. Ви можете вказувати папки й файли, що беруть участь у резервному копіюванні. Після завершення вибору вихідних даних вам пропонується вказати метод компресії. Вибір невеликий, ви можете використовувати лише архіватор ZIP, указуючи ступінь стиску даних. Архів може бути захищений паролем. При генерації ім'я файлу резервної копії можна використовувати різноманітні змінні, що вказують на час створення архіву. Збереження даних може супроводжуватися PGP-шифруванням.

Наступний етап настроювань полягає у вказівці кінцевого розташування резервної копії. Ви можете розташовувати її на жорсткому диску, у локальній мережі, на FTP-сервері, а також на оптичних носіях. До складу ABC Backup Pro включений модуль прожигу CD і DVD дисків. Настроювання параметрів запису містять у собі вказівку швидкості запису, імпорту сесій, вибору файлової системи, закриття диска. Говорячи інакше, прожиг болванок – не формальність,

розроблювачі підійшли до даного питання досить серйозно. Зрозуміло, жодне серйозне резервне копіювання не обходиться без настроювання планувальника. ABC Backup Pro дозволяє вказувати діапазон часу (дата/час), протягом якого повинна виконуватися завдання. Періодичність може бути щоденною, щотижневою, щомісячною, а також одиночною. У випадку вибору тижня або місяця як тимчасовий інтервал, ви можете вказувати конкретні одиниці усередині списку (дні тижня, назви місяців).

Як додаткові опції завдання можна призначити вивантаження з пам'яті певної програми, а також запуск додатків до й після добутку резервного копіювання.

Розроблювачі затверджують, що їхній продукт володіє інкрементальним копіюванням. Це неправда. Ви можете лише використовувати унікальні імена файлів, прив'язані до поточної дати, але вміст архівів завжди являє собою повну копію заздалегідь певних даних.

Значок додатка постійно перебуває в системному лотку, звідки можна викликати контекстне меню зі списком призначених завдань і здійснювати основні операції з ними.

ABC Backup Pro – простій, зручний механізм резервного копіювання з неякісною локалізацією.

Active Backup Expert Pro

Active Backup Expert Pro призначений винятково для створення резервних копій. Планування подій можна здійснювати за допомогою іншого продукту того ж розроблювача – Active Task Manager. Після першого старту програми вас привітає Майстер, що пропонує відразу організувати проект резервного копіювання. Під час першого кроку ви вводите ім'я проекту, після чого переходите до процедури вказівки файлів і папок, що беруть участь у резервному копіюванні. Ви можете задавати окремі правила фільтрації списків файлів, щоб не вказувати кожний елемент окремо. Як наступний крок допускається вказівка правил виключення. Подібна система фільтрації списків дозволяє гнучко

вказувати, які конкретно файли необхідно архівувати. Наприклад, досить логічним рішенням можна вважати виключення з резервного копіювання файлів з розширенням *.bak.

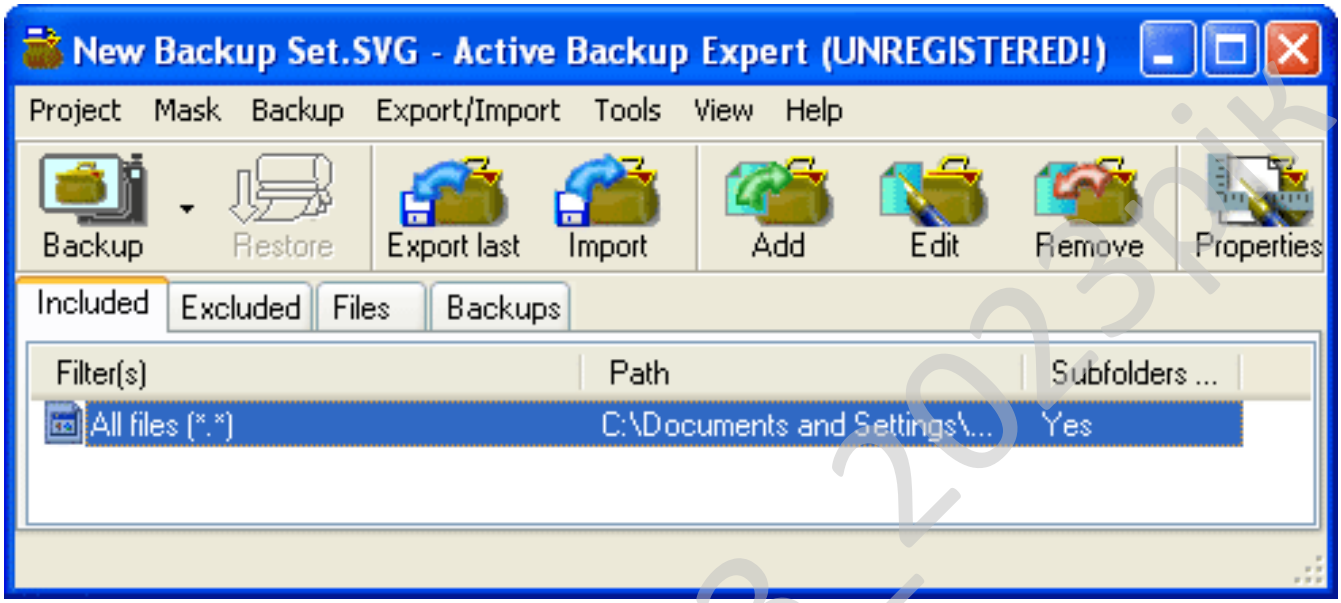

Рисунок 2.2 – Робоче вікно Active Backup Expert Pro

Наступним кроком іде вказівка місця розташування архівних файлів. Ви можете вказувати не тільки локальну файлову систему, але й FTP-ресурси, локальну мережу, а також розділ оптичного привода. Останнє рішення не зовсім інтуїтивно зрозуміло. Так, програма дозволяє створювати резервні копії на CD і DVD носіях. У подібних випадках це підкреслюється окремою групою опцій, що забезпечує коректне настроювання роботи привода.

Після вибору мети резервного копіювання, вам пропонується зайнятися настроюванням додаткових опцій програми. І, нарешті, під час останнього кроку роботи Майстра ви можете ознайомитися з усіма пунктами поточного проекту, після чого програма порекомендує створити окремий ярлик, відповідальний за виклик поточного документа. Як уже було сказано раніше, програма не має власного планувальника, і резервне копіювання може бути запущено тільки вручну, якщо не прибігати до сторонніх продуктів.

Ви можете в будь-який момент додати нові файли в проект, поміняти поточні правила, фільтри. Крім того, існує окрема група опцій, керування якою застосовне до всіх проектів, що відкриваються.

По-перше, допускається вибір алгоритму стиску архівних файлів. Ви можете вибрати архіватор ZIP, але при цьому встановлюється обмеження на розмір архіву, 2ГБ. Використання формату CAB дозволяє зняти дані обмеження. Архів може бути що саморозпаковується, доступ до нього допускається закривати паролем, а також використовувати шифрування.

Active Backup Expert Pro має убудовані інструменти створення віддаленого з'єднання із провайдером. Файли резервної копії можуть автономно передаватися по модему.

Програма містить досить потужні механізми резервного копіювання даних, однак сама її концепція не зовсім звична, вона йде врозріз із традиційною структурою продуктів даного класу.

ApBackUP

ApBackUP – досить потужний програмний продукт, що володіє серією незвичайних, цікавих можливостей, відсутніх у конкурентів. За допомогою даної програми ви можете призначати відразу кілька завдань резервного копіювання, кожна з яких буде виконуватися відповідно до власного графіка, зазначеним у планувальнику.

Створення завдання здійснюється за допомогою Майстра. Незареєстрована версія не дозволить вам описати більше двох завдань. На першому етапі вам необхідно вказати тип збереження даних – архівування, звичайне копіювання або перенос інформації на FTP-сервер. Далі вам просять указати папки на жорсткому диску, що беруть участь у резервному копіюванні. Після вибору кожного елемента ви можете ввести довільну маску, тим самим, відсіваючи зайві дані. Усередині одного завдання може перебувати кілька папок і кілька різних масок. Крім того, допускається окрема вказівка виключень за допомогою аналогічного вікна вибору папок і масок.

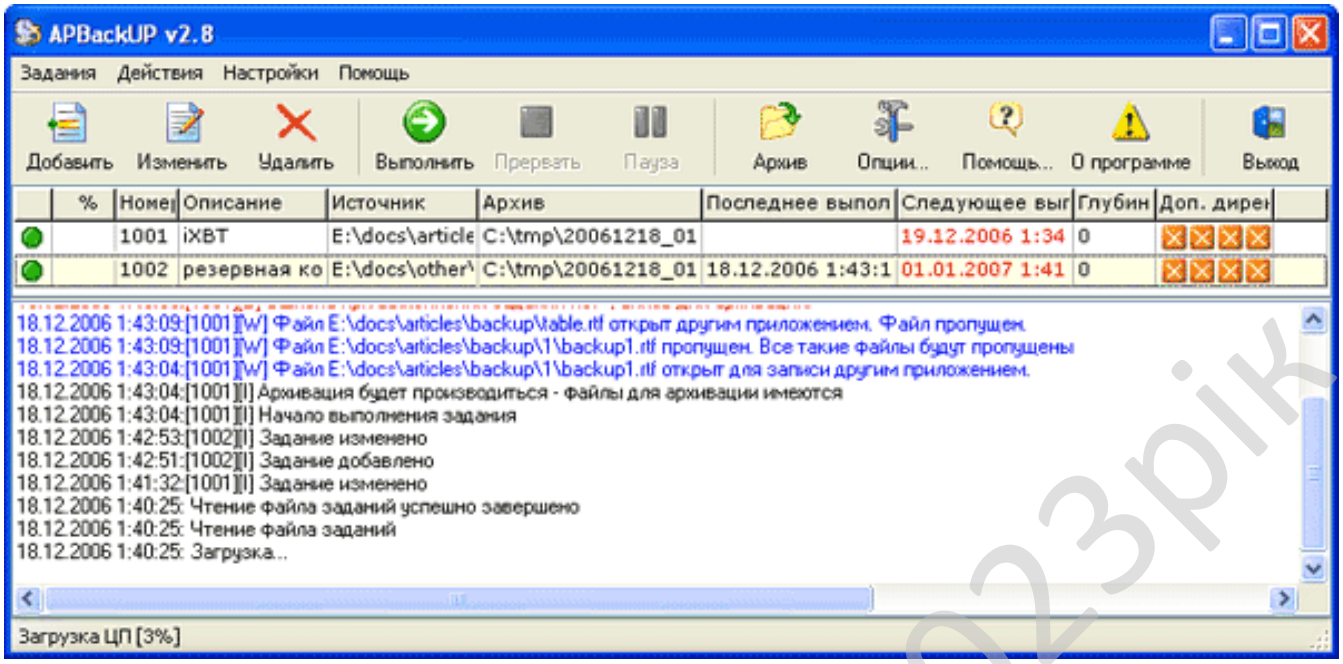

Рисунок 2.3 – Робоче вікно ApBackUP

На наступному етапі ви вказуєте шлях до папки з резервними копіями, а також уводите правила створення імен архівних файлів. Після завершення всіх готувань керування переходить до убудованого планувальника.

ApBackUP пропонує безліч варіантів планування. Ви можете вибирати різні інтервали часу, визначати умови, при дотриманні яких можна запускати процедуру резервного копіювання. По закінченні планування програма пропонує ознайомитися із властивостями поточного завдання, у рамках яких ви можете скорегувати безліч додаткових параметрів.

Перше, на що варто звернути увагу – це можливість використання будь-яких установлених у системі архіваторів. Від вас потрібно лише заздалегідь прописати параметри конфігурації зовнішнього модуля в спеціальному вікні «Настроювання зовнішніх архіваторів». До складу дистрибутива включена велика база популярних продуктів даного класу, наприклад, WinRAR, 7-Zip, Ace, Arj і інші.

Програма дозволяє відсилати по e-mail не тільки повідомлення, але й сам архів з резервною копією. У планувальнику можна описувати умови, дотримання

яких дозволяє почати резервне копіювання. Наприклад, у випадку підключення до Мережі через з'єднання, що комутується, ApBackUP може відслідковувати його статус.

Ще одна досить цікава особливість програми полягає в можливості копіювання архіву з резервною копією в три додаткові папки на жорсткому диску. Кожна з копій може мати унікальні настроювання.

Завдання в ApBackUP можуть виконуватися паралельно. Ви можете задавати різні пріоритети завданням, щоб при виникненні перетинань швидкість архівування більше важливого завдання не сильно знижувалася. Крім того, можна вказати загальний пріоритет всім завданням, з метою відрегулювати максимальне завантаження процесора.

ApBackUP – дуже вдала програма, що не володіє якими-небудь серйозними, явними недоліками, недоробками. ApBackUP залишає враження добре збалансованого закінченого продукту.

Auto Backup

Auto Backup нескладна програма для автоматичного резервного копіювання важливої інформації на жорсткому диску з можливістю збереження даних усередині локальної мережі, а також на віддалених FTP-серверах.

Ви можете створювати кілька незалежних завдань, однак, без можливості якого-небудь угруповання. Створення нового завдання здійснюється за допомогою єдиного діалогового вікна із вкладками. Як джерело можна вказати кілька папок на жорсткому диску з можливістю додаткового застосування фільтрів входження й виключення. Приймачем резервної копії може стати папка на жорсткому диску, загальний ресурс на робочій станції в локальній мережі, а також FTP-сервер. Планувальник дозволяє вибирати конкретні дні тижня, дні місяця, протягом яких допускається резервне копіювання. Ви можете також призначити стартовий час початку завдання й інтервал повторення, вимірюваний у хвилинах.

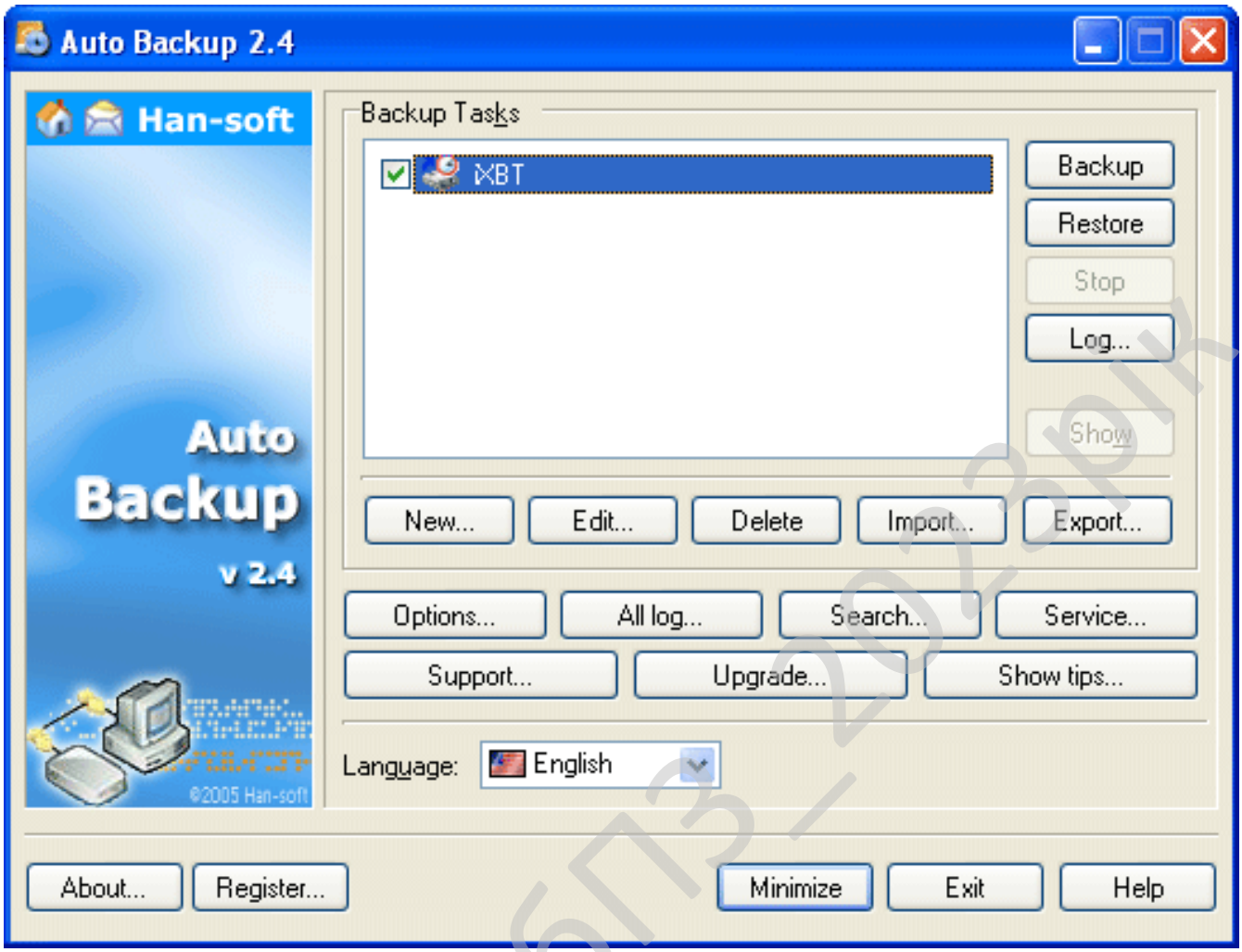

Рисунок 2.4 – Робоче вікно Auto Backup

Auto Backup підтримує інкрементальне й диференціальне копіювання, а також дозволяє створювати архіви, що саморозпаковуються. Дані можуть захищатися паролем і шифруванням. Програма також підтримує розбивку архівів на частини, відповідно до ємності популярних типів носіїв, а також шляхом уведення довільного значення.

Перед початком виконання завдання, а також по її завершенню можна запускати зовнішні додатки. Програма має одну цікаву особливість, що полягає в можливості запуску програм у випадку невдалої спроби резервного копіювання – своєрідна розрада, бальзам на душу. Крім того, по завершенню виконання завдання можна відправляти повідомлення по електронній пошті.

Програма може працювати як служба Windows. Всі дії Auto Backup записуються в докладний журнал. У контекстне меню провідника додається новий пункт Restore with Auto Backup. Для відновлення даних не обов'язково навіть запускати основний модуль програми. Auto Backup може запускати кілька потоків одночасно, що дозволяє, наприклад, сполучити резервне копіювання й відновлення. Інформація про виконання всіх фонових завдань відображається усередині спливаючого вікна (baloon) біля системного лотка.

Auto Backup – надійна програма без яких-небудь надмірностей і прикрас. Незручне, погано структуроване розташування елементів керування компенсується вдалим, раціональним підбором функціональних можливостей.

Back2zip

Back2zip – дуже проста безкоштовна програма для створення резервних копій. Розроблювачі реалізували в продукті лише мінімальний набір інструментів, якого, як не дивно, вистачає в більшості життєвих ситуацій.

Усередині головного вікна програми зосереджені всі основні інструменти керування резервним копіюванням. Спочатку ви вказуєте папки на жорсткому диску, які необхідно архівувати. Потім необхідно задати розклад виконання завдання. Правила елементарні – ви вказуєте інтервал створення резервних копій, а також уводите точний час доби, після якого програма може почати роботу. Небагато дивним виглядає список доступних інтервалів часу. Логічно було б припустити навність зміни доби, як відправна крапка для створення резервної копії. Наприклад, «щодня після 17.00». Але ні, максимальний інтервал часу, доступний програмі, рівняється всього лише 6 годинникам.

Програма дозволяє вказувати період часу, протягом якого можуть зберігатися старі архівні копії. Максимальний строк – 2 тижні. Знову ж – це не зовсім логічно. Чому не можна було дозволити користувачеві самостійно визначати строк зберігання резервних копій?

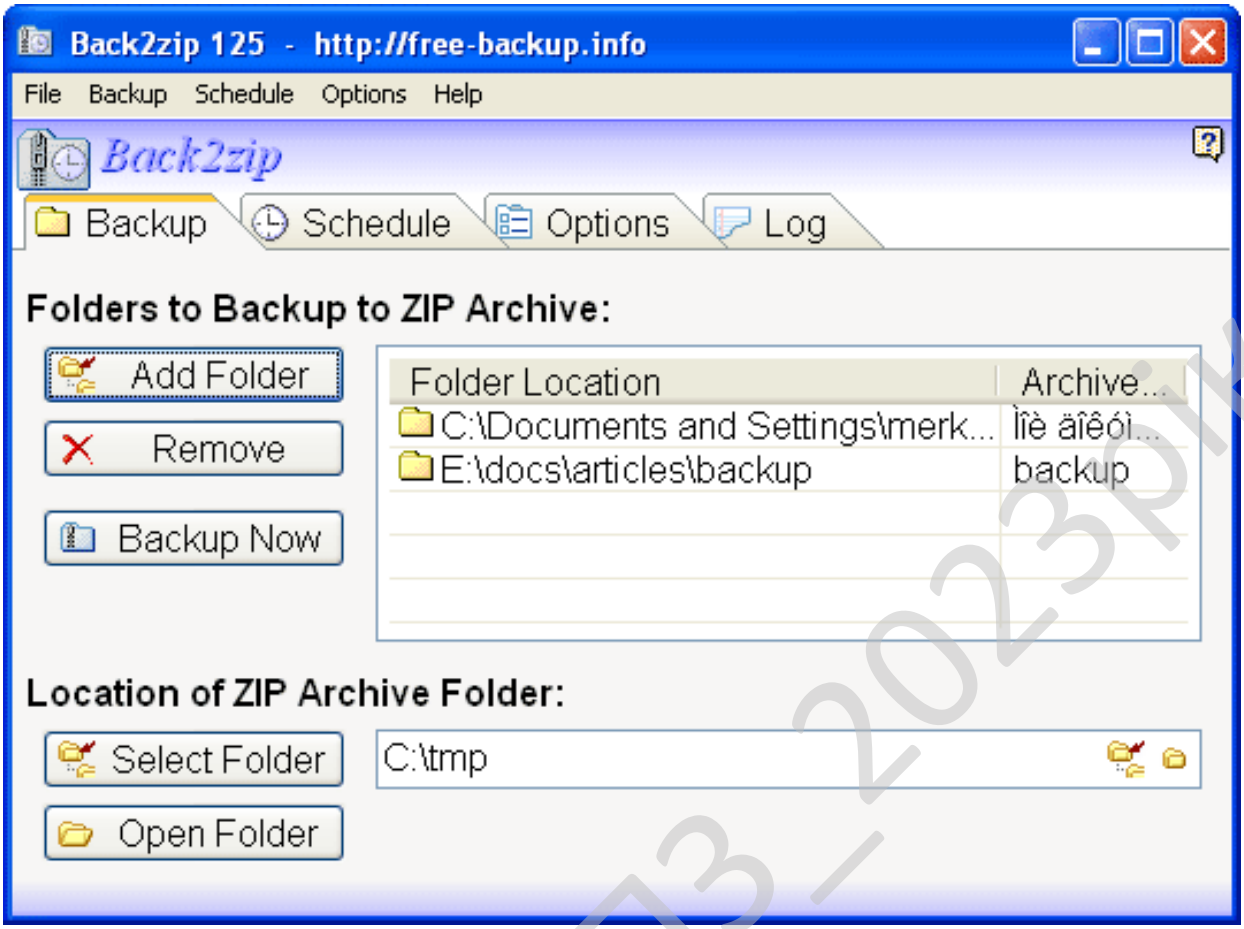

Рисунок 2.5 – Робоче вікно Back2zip

На цьому, як не дивно, опис можливостей програми закінчується. Як уже було сказано, Back2zip – дуже простий продукт, не обтяжений численними опціями, функціями.

Backup4all

Backup4all вдало сполучає у собі широкі функціональні можливості й простоту освоєння. Створення нових проектів резервного копіювання здійснюється за допомогою Майстра, що розкладає всі операції на кілька кроків. Усередині робітника вікна додатка ви можете переглядати деревоподібну структуру поточного проекту, виділяючи різні стани файлів – змінені, нові, виключені й інші. У якість заздалегідь певних прикладів вам пропонується резервне копіювання папок «Мої документи» і «Мої рисунки», а також «Вибране» Internet Explorer.

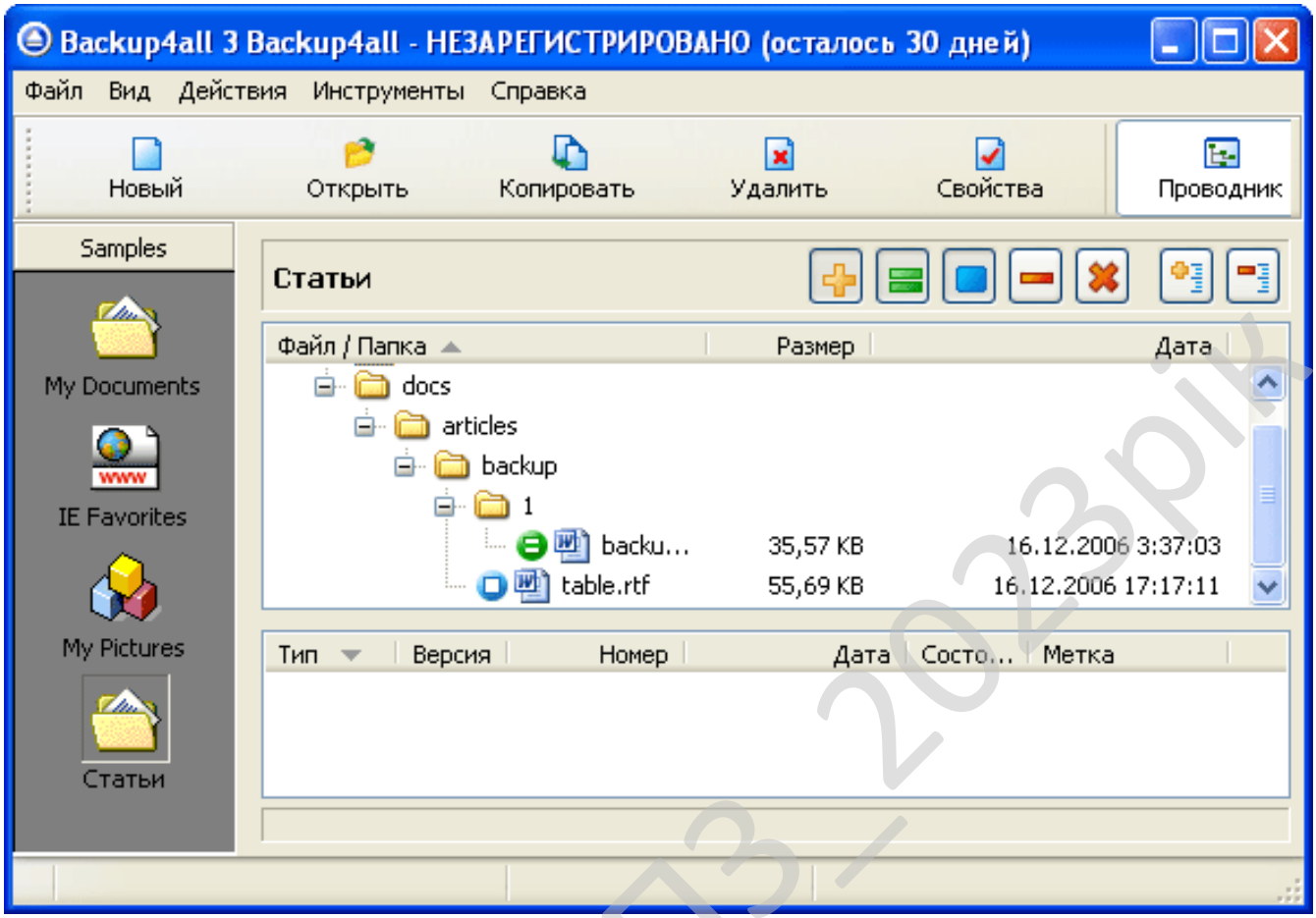

Рисунок 2.6 – Робоче вікно Backup4all

Всі завдання можуть бути представлені у вигляді деревоподібної структури, заснованої на групах. За замовчуванням ви маєте один лише елемент Samples. Якщо ви створите додаткові групи завдань, то їхній список буде відображатися зверху/знизу бічної панелі. Досить лише клацнути мишею по заголовку елемента, щоб перемкнутися на поточну групу.

Створення нового завдання резервного копіювання починається із вказівки її ім'я, вибору групи, унікального значка. Вам надається також можливість відразу вибрати набір визначених настроювань. Кожне завдання може супроводжуватися текстовим описом.

Далі вам має бути вибрати папки й конкретні файли, що беруть участь у резервному копіюванні. Крім локальних ресурсів, ви можете витягати дані з мережного оточення.

Як одержувач резервної копії можна вказувати локальну файлову систему, мережу або FTP-сервер. Якщо ви збираєтеся зберігати копії на жорстких дисках, то необхідно вказати додатково папку призначення. Втім, ви можете відразу вибрати логічний диск оптичного привода, після чого програма запропонує вказати кілька додаткових опцій, властивих винятково прожигу дисків. Backup4all використовує власний модуль, що працює з пишучими приводами. Допускається вказівка довільної швидкості запису, можливість примусового очищення перезаписуваних дисків перед записом. Ви можете також використовувати файлову систему UDF у сполученні з DirectCD/InCD, що дає можливість прозорого додавання файлів на носії, без метушні зі створенням нових сесій, імпортом старих даних. Використання FTP як одержувач резервних копій дозволяє вам підключатися до серверів прямо або через проксі, з використанням авторизації, SSL-шифрування. Ви можете жорстко обмежувати швидкість прийому й віддачі даних.

Резервне копіювання може бути чотирьох типів – повне, інкрементальне, диференціальне, а також створення дзеркала. Залежно від процентного співвідношення відмінностей поточної резервної копії від вихідного архіву, програма може автоматично заміняти інтелектуальні методи резервування створенням повної копії. Backup4all підтримує кілька методів шифрування архівів, а також дозволяє закривати доступ до резервної копії паролем.

Якщо під час вказівки вихідних файлів, призначених для резервного копіювання, ви вибрали кілька папок, те, швидше за все, не всі дані мають потребу в збереженні. Ви можете працювати із двома типами фільтрів – що включаються й що виключаються. Як наслідок, залежно від конкретної ситуації можна вказати унікальні правила включення й виключення файлів.

Наступний крок дозволяє описати деякі додаткові моменти процедури резервного копіювання. Наприклад, від якого критерію відштовхуватися, з'ясовуючи, змінився чи файл ні? Програма дозволяє вибрати безліч параметрів оцінки стану даних. Різноманітні атрибути файлової системи, контрольна сума,

навність індексації й безліч інших параметрів можуть виступати як факторів, що визначають змінився чи файл ні. Резервне копіювання може бути пов'язане з якою-небудь сторонньою дією, наприклад, із запуском додатка. Backup4all пропонує призначити подібні дії, чинені до резервного копіювання й після. Виконання завдання може супроводжуватися звуковими сигналами, а також відправленням повідомлення про виконання на довільну адресу електронної пошти.

І, нарешті, останній етап роботи Майстра – це опис роботи планувальника. Backup4all дозволяє використовувати убудований інструмент Windows, але має також власного планувальника. Програма має велику кількість варіантів правил розкладу виконання резервного копіювання. Наприклад, робоча станція може очікувати періоду низького завантаження процесора, щоб виконати завдання, мінімально заважаючи роботі системи. Інший приклад – якщо програма виявить, що комп'ютер працює не від електричної мережі, а від батарей, то створення резервної копії необхідно відкласти.

Будь-який проект може бути збережений на жорсткий диск у вигляді окремого ярлика. Наприклад, ви вирішили використовувати убудований планувальник. Під час настроювання завдання вкажіть запуск ярлика проекту Backup4all. При настанні події, планувальник автоматично викличе сторонній інструмент резервного копіювання.

Програма дозволяє наочно відслідковувати статус резервного копіювання. За допомогою групи фільтрів, розташованих на панелі інструментів, ви можете вказувати критерії відображуваних даних, що дозволяє наочно відслідковувати всі зміни файлів. Backup4all може відображати докладну статистику по кожному проекті, а також експортувати дані в CSV-файли.

Перед відновленням даних ви можете вказувати конкретні версії файлів, а також виконувати розпакування в довільний каталог (не поверх вихідних даних).

Backup4all володіє однією цікавою можливістю, названої One Touch Backup. Її суть полягає в здатності програми виявляти що підключаються USB-

пристрою й у випадку успіху відразу пропонувати запуск резервного копіювання. У настроюваннях програми досить указати вихідну папку або вибрати заздалегідь певне завдання. Резервне копіювання запускається одним клацанням миші, звідси й назва – «резервне копіювання по першому дотику».

Backup4all має якісну локалізацію, що дозволяє використовувати програму навіть тим, хто не знайомий з англійською мовою. Дану програму не можна назвати самої потужною, функціональною у всьому секторі ринку. Backup4all націлений на тих, кому важливі не тільки широкі функціональні можливості, але й зручність використання. Ви можете гнучко міняти зовнішній вигляд Backup4all (набудовувати панелі, міняти їхні розкладки), ґрунтуючись на власних перевагах. Це, може бути, не настільки важливо для програми, що виконує резервне копіювання, але, навіть виконуючи сугубо технічні операції, хочеться не втрачати краси, зручності, наочності. З Backup4all можна виконувати резервне копіювання з комфортом.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення

виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Резервне копіювання (англ. backup) – процес створення копії даних на носії (жорсткому диску, дискеті й т.д.), призначеному для відновлення даних в оригінальному місці їхнього розташування у випадку їхнього ушкодження або руйнування, що відповідають програмами – резервними дублікаторами даних.

Найменування операцій:

– Резервне копіювання даних (Резервне дублювання даних) – процес створення копії даних.

– Відновлення даних – процес відновлення в оригінальному місці.

Резервне копіювання необхідно для можливості швидкого й недорогого відновлення інформації (документів, програм, настроювань і т.д.) у випадку втрати робочої копії інформації з якої-небудь причини.

Крім цього вирішуються суміжні проблеми:

– Дублювання даних.

– Передача даних і робота із загальними документами.

Вимоги до системи резервного копіювання:

– Надійність зберігання інформації – забезпечується застосуванням відказостійкого встаткування систем зберігання, дублюванням інформації й заміною загубленої копії іншої у випадку знищення однієї з копій (у тому числі як частина відказостійкості).

– Простота в експлуатації – автоматизація (по можливості мінімізувати участь людини: як користувача, так і адміністратора).

– Швидке впровадження – проста установка й настроювання програм, швидке навчання користувачів.

Для резервного копіювання дуже важливим питанням є вибір підходящої схеми ротації носіїв (наприклад, магнітних стрічок). Найбільше часто використовують наступні схеми:

– Одноразове копіювання (custom) – найпростіша схема, що не передбачає ротації носіїв. Всі операції проводяться вручну. Перед копіюванням адміністратор задає час початку резервування, перераховує файлові системи або каталоги, які потрібно копіювати. Цю інформацію можна зберегти в базі, щоб її можна було використовувати знову. При одноразовому копіюванні найчастіше застосовується повне копіювання.

– Проста ротація.

– «Дід, батько, син».

– «Ханойська вежа».

– «10 наборів».

Схеми «Ханойська вежа» і «10 наборів» використовуються нечасто, так як багато систем резервування їх не підтримують.

Зберігання резервної копії:

– Стрічка стримера – запис резервних даних на магнітну стрічку стримера.

– «Хмарний» бекап» – запис резервних даних по «хмарній» технології через онлайн-служби спеціальних провайдерів.

– DVD або CD – запис резервних даних на компактні диски.

– HDD – запис резервних даних на жорсткий диск комп'ютера.

– LAN – запис резервних даних на будь-яку машину усередині локальної мережі.

– FTP – запис резервних даних на FTP-Сервери.

– USB – запис резервних даних на будь-яке USB-Сумісний пристрій (таке, як флеш-карта або зовнішній жорсткий диск).

– ZIP, JAZ, MO – резервне копіювання на дискети ZIP, JAZ, MO.

Методи боротьби із втратою інформації

Втрата інформації буває по різних причинах:

1. Експлуатаційні поломки носіїв інформації.

Опис: випадкові поломки в межах статистики відмов, пов'язані з необережністю або виробітком ресурсу. Звичайно ж, якщо якась важлива інформація вже загублена, то можна звернутися в спеціалізовану службу – але надійність цього не стовідсоткова.

Боротьба: зберігати всю інформацію (кожний файл) мінімум у двох екземплярах (причому кожний екземпляр на своєму носії даних). Для цього застосовуються:

– RAID 1, що забезпечує відновлення самої свіжої інформації. Файли, розташовані на сервері з RAID, більше захищені від поломок, чим ті, які зберігаються на локальній машині;

– Ручне або автоматичне копіювання на інший носій. Для цього може використовуватися система контролю версій, спеціалізована програма резервного копіювання або підручні засоби на зразок cmd-файлу, що запускається періодично.

2. Стихійні й техногенні нещастя.

Опис: шторм, землетрус, крадіжка, пожежа, прорив водопроводу – все це приводить до втрати всіх носіїв даних, розташованих на певній території.

Боротьба: єдиний спосіб захисту від стихійних лих – тримати частина резервних копій в іншому приміщенні.

3. Шкідливі програми. Опис: у цю категорію входить випадково занесене ПЗ, що навмисно псує інформацію – віруси, хробаки, «троянські коні». Іноді факт зараження виявляється, коли чимала частина інформації перекручена або знищена.

Боротьба:

– Встановка антивірусних програм на робочі станції. Найпростіші антивірусні міри – відключення автозавантаження, ізоляція локальної мережі від Інтернету, і т.д.

– Забезпечення централізованого відновлення: перша копія антивірусу одержує відновлення прямо з Інтернету, а інші копії настроєні на папку, куди перша завантажує відновлення; також можна настроїти проксі-сервер таким чином, щоб відновлення кешувалися (це все міри для зменшення трафіка).

– Мати копії в такому місці, до якого вірус не добереться – виділений сервер або знімні носії.

– Якщо копіювання йде на сервер: забезпечити захист сервера від вірусів (або встановити антивірус, або використовувати ОС, для якої ймовірність зараження мала). Зберігати версії достатньої давнини, щоб існувала копія, що не контактувала із зараженим комп'ютером.

– Якщо копіювання йде на знімні носії: частина носіїв зберігати (без дописування на них) досить довго, щоб існувала копія, що не контактувала із зараженим комп'ютером.

4. Людський фактор. Опис: навмисне або ненавмисне знищення важливої інформації – людиною, спеціально написаною шкідливою програмою або збійним ПЗ.

Боротьба:

– Ретельно розставляються права на всі ресурси, щоб інші користувачі не могли модифікувати чужі файли. Виключення робиться для системного адміністратора, що повинен мати всі права на всі, щоб бути здатним виправити помилки користувачів, програм і т.д.

– Забезпечити працюючу систему резервного копіювання – тобто, систему, якою люди реально користуються і яка досить стійка до помилок оператора. Якщо користувач не користується системою резервного копіювання, вся відповідальність за схоронність лягає на нього.

– Зберігати версії достатньої давнини, щоб при виявленні зіпсованих даних файл можна було відновити.

– Перед переустановкою ОС варто обов'язково копіювати весь уміст роздягнула, на якій буде встановлена ОС, на сервер, на інший розділ або на CD / DVD.

– Оперативно обновляти ПЗ, що запідозрено у втраті даних.

Далі ми запропонуємо простий алгоритм резервного копіювання, придатний для будь-якої пишучої апаратури й носіїв даних. Алгоритм використовувався для CDR/CDRW і згодом для DVDR/DVDRW, але може бути розповсюджений на будь-яке встаткування.

Оперативно копіювати робітники дані на зовнішні носії складно й малоефективно. Для створення копії робочих даних найкраще використовувати додатковий файл-сервер. У випадку з Windows у якості файл-сервера для копіювання можна використовувати команду xcopy, запускаючи її за розкладом, наприклад, раз у півгодини. Її недоліком є "невміння" стирати видалені на основному сервері файли, а також неможливість у певних випадках копіювати відкриті користувачами на запис і не збережені файли. Втім, наявність на додатковому сервері файлів, які видалені на основному – не завжди недолік. Іноді це величезний плюс – наприклад, у випадку випадкового стирання користувачем файлу на мережному диску. Сподіваємося, факт необхідності зберігання всіх файлів на мережних ресурсах роз'ясняти не потрібно.

Отже, із затримкою не більше на півгодини ми маємо всі дані на додатковому сервері. Відкриті й недоступні для копіювання файли зуміє скопіювати в іншому сеансі, або ж по закінченні робочого дня. Така система дозволяє вирішити завдання копіювання робочих даних, а також відшукання копій віддалених файлів. Для відшукання копій будь-яких файлів давниною в кілька днів, дані з додаткового сервера варто щодня записувати на DVD. Щоб уникнути зайвих витрат, для цього можна використовувати DVD-RW (або DVD+RW, що непринципово), по одному диску на кожний робочий день того періоду, за який Ви хочете мати резервні копії. Це може бути тиждень, місяць або

навіть квартал. В останньому випадку вже простіше купити одноразові DVD-R і складувати їх нескінченно.

Для перегляду архівних даних давниною в кілька тижнів або місяців, резервні копії треба з відповідним інтервалом записувати на одноразові носії й зберігати в надійному місці. Можливо, записувати копії двічі й зберігати в різних місцях – в одному (віддаленому, але надійному), і в іншому (легко й швидко доступному).

Залежно від ступеня конфіденційності інформації, файли, що підлягають резервному копіюванню, можна заархівувати як є, заархівувати із паролем або ж зашифрувати. Для архівування найкраще використовувати архіватор, зі складним паролем не коротше 6 символів, що включають символи @#$%^&, букви в різних регістрах і цифри. Для шифрування рекомендуємо створювати контейнери, відформатовані в FAT32, за допомогою TrueCrypt для Linux або Windows, а потім класти в них готові архіви.

Устаткування й носії резервного копіювання

У цей час найбільш дешевим засобом резервного копіювання є пишучий DVD-дисковід. Пише він досить швидко, і є присутнім практично в будь-якому сучасному комп'ютері. Носії досить дешеві, що дозволяє мати будь-яку необхідну кількість архівних копій. Обмеженням є обсяг однобічного одношарового DVD-диска – 4,7 млрд. байт. Двосторонні диски використовувати нема рації. Двослойні диски й більше нові носії використовувати теж нема рації через дорожнечу пишучого й читаючого встаткування.

Нема рації не заощаджувати на DVD-дисководах, не купувати дорогі. NEC або Teac відмінно вирішать проблему. Рекомендуємо, проте, попередньо почитати відкликання про наявних на даний момент у продажі моделях. За швидкістю гнатися не має глузду, вистачить восьмишвидкістного дисководу (для DVD-RW).

Якщо у Вас у мережі багато великих файлів (фотографії, поліграфічні макети), сумарний обсяг яких в архіві (або навіть обсяг окремих файлів)

перевищує 4 гігабайти, то варто задуматися над покупкою декількох зовнішніх жорстких дисків, щоб використовувати їх замість DVD, щодня міняти й тримати копії в надійному віддаленому сховищі.

Особливістю програми є те, що:

– вона створює не один архів, а серію архівів, протягом тижня, даючи їм імена по днях тижня;

– архівація виконується тільки раз у день (принаймні в один каталог). Для запобігання повторної архівації, до протоколу архіву заноситься дата;

– у щодня створювані архіви попадають тільки змінені файли, заощаджуючи місце на зайвих копіях;

– в обраний користувачем день, раз у тиждень, виконується тотальна архівація, де збираються всі архівуємі файли;

При створенні нового архіву, старий перейменовується, дозволяючи зберегти, про всякий випадок, копії за минулий тиждень...

У такий спосіб система зберігає двотижневу історію стану даних, практично не витрачаючи для цього дискового простору – нічого подібного не має навіть багатогигабайтні засоби відновлення системи. Різниця лише в тому, що відновлення доводиться виконувати вручну – але, у цьому випадку, ця велика перевага – краща версія визначається не сліпим алгоритмом, а живою людиною.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленої системи. Структурна схема показує як блоки програми зв’язані між собою.

З нього ми бачимо, що система складається з наступних основних структурних блоків:

– Блок інтерфейсу користувача програми.

– Блок швидкого доступу до елементів програми.

– Блок резервного копіювання.

– Блок виконання операцій архівування.

– Блок алгоритмів за допомогою яких відбувається архівування/розархівування файлів.

– Блок форматів архівів, з якими працює програма.

Блок резервного копіювання виконує наступні операції:

– Повне резервування (Full backup). Повне резервування звичайно торкається всієї вашої системи й всіх файлів. Щотижневе, щомісячне й щоквартальне резервування має на увазі повне резервування. Перше щотижневе резервування повинне бути повним резервуванням, звичайно виконуваним по п'ятницях або протягом вихідних, протягом якого копіюються всі бажані файли. Наступні резервування, виконувані з понеділка по четвер до наступного повного резервування, можуть бути додатковими або диференціальними, головним чином для того, щоб зберегти час і місце на носії. Повне резервування варто проводити, принаймні, щотижня.

– Диференціальне резервування (Differential backup). При диференціальному резервуванні кожний файл, що був змінений з моменту останнього повного резервування, копіюється щораз заново. Диференціальне резервування прискорює процес відновлення. Усе, що вам необхідно, це остання повна й остання диференціальна резервна копія. Популярність диференціального резервування росте, так як всі копії файлів робляться в певні моменти часу, що, наприклад, дуже важливо при зараженні вірусами.

– Додаткове резервування (Incremental backup). При додатковому ("інкрементальному") резервуванні відбувається копіювання тільки тих файлів, які були змінені з тих пор, як востаннє виконувалося повне або додаткове резервне копіювання. Наступне додаткове резервування додає тільки файли, які були змінені з моменту попереднього додаткового резервування. У середньому, додаткове резервування займає менше часу, так як копіюється менша кількість файлів. Однак, процес відновлення даних займає більше часу, так як повинні бути відновлені дані останнього повного резервування, плюс дані всіх наступних

додаткових резервувань. При цьому, на відміну від диференціального резервування, що змінилися або нові файли не заміщають старі, а додаються на носій незалежно.

– Пофайловий метод. Система пофайлового резервування запитує кожний індивідуальний файл і записує його на носій. Завжди варто використовувати пропоновану опцію верифікації. При верифікації, всі копіюємі з диска дані перечитуються із джерела й перевіряються або побайтно рівняються з даними на носії. Так як фрагментировані файли на диску через більшу кількість виконуваних операцій пошуку сповільнюють процес резервування, то продуктивність можна звичайно збільшити роблячи регулярну дефрагментацію диска. При дефрагментації блоки даних розташовуються один по одному, один за одним так, щоб вони були доступні в кеші читання, що попереджає.

– Блокове інкрементальне копіювання (Block level incremental).

Блок виконання операцій архівування, у свою чергу, включає у себе наступні блоки:

– Архівування.

– Розархівування.

– Відновлення пошкоджених архівів.

– Створення/розпакування багатотомних архівів.

– Створення архівів, що саморозпаковуються.

Блок алгоритмів за допомогою яких відбувається архівування/розархівування файлів включає в себе реалізацію наступних алгоритмів:

– Алгоритми стиску без втрат.

– Алгоритми стиску з використанням словника.

– Алгоритми стиску із втратами.

При цьому в архівуванні використовуються тільки алгоритми стиску без втрат.

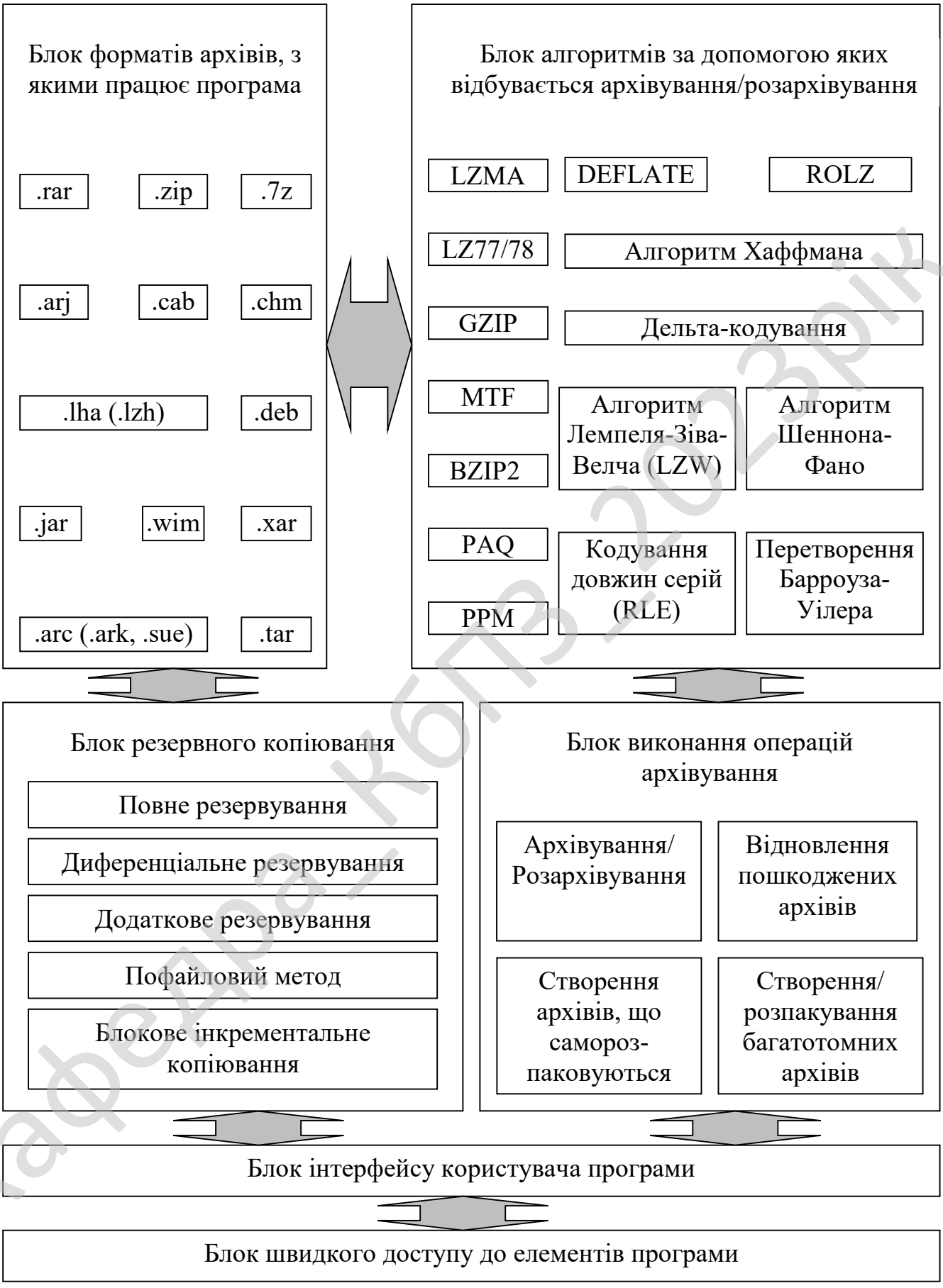

Рисунок 3.1 – Структурна схема системи

Блок форматів архівів, з якими працює програма, включає в себе наступні формати:

– RAR – розповсюджений пропрієтарний формат стиску даних і програма-архіватор. Формат розроблений російським програмістом Євгенієм Рошалом (звідси й назва RAR: Roshal Archiver). Він написав програму-архіватор для впакування/розпакування RAR, під DOS, потім і для інших операційних систем. Версія для Microsoft Windows поширюється в складі багатоформатного архіватора із графічним інтерфейсом WinRAR. Програма поширюється як умовно-безкоштовне програмне забезпечення (shareware). Для файлів формату RAR звичайно використовується розширення .rar і MIME-Тип application/ x-rar-compressed.

– ZIP – популярний формат стиску даних і архівування файлів. Файл у цьому форматі звичайно має розширення .zip і зберігає в стислому або незжатому виді один або кілька файлів, які можна з нього витягти шляхом розпакування за допомогою спеціальної програми. ZIP був розроблений Філом Кацем для використання в програмі PKZIP. Згодом з'явилася безліч інших утиліт, що працюють із цим форматом.

– 7z – формат алгоритму 7-Zip – вільний файловий архіватор з високим ступенем стиску даних. Підтримує кілька алгоритмів стиску й безліч форматів даних, включаючи власний формат 7z с високоефективним алгоритмом стиску LZMA. Програма розробляється з 1999 року і є безкоштовною, а також має відкритий вихідний код, більша частина якого вільно поширюється на умовах ліцензії GNU LGPL, за винятком коду декомпресора unRAR, що має обмеження. Основною платформою є Windows, де доступні дві версії програми: із графічним інтерфейсом і версія для командного рядка. Консольна версія була портирована співтовариством розроблювачів для систем стандарту POSIX під загальною назвою p7zip. Портировані версії для інших систем, так само як і оригінальна програма 7-zip, доступні на сайті системи SourceForge. 7-Zip є переможцем

SourceForge.net Community Choice Awards 2007 року в категоріях: кращий проект і кращий технічний дизайн.

– ARJ – файловий архіватор. Розроблений Робертом К. Джангом (Robert K. Jung). (Походження найменування ARJ: Archiver Robert Jung). ARJ компресія подібна PKZIP 1.02. Існує також версія ARJ з відкритим вихідним кодом, доступна під більш, ніж десятьома операційними системами, включаючи DOS, 16– і 32-х розрядні версії Windows і OS/2, різні варіанти UNIX і Linux. Існує також версія Russian NVL, що дозволяє захищати архіви за допомогою шифрування алгоритмом GOST.

– ARC – формат стиску даних без втрат і архівування файлів від System Enhancement Associates. Файл у цьому форматі звичайно має розширення .arc, .ark або .sue і зберігає в стислому або незжатому виді один або кілька файлів, які можна з нього витягти шляхом розпакування за допомогою спеціальної програми.

– Cabinet (.cab) – формат файлів для архівів зі стиском, що застосовується в операційних системах сімейства Microsoft Windows. Формат підтримує стиск і цифрові підписи, використовується в різних технологіях установників від Microsoft: Setup API, Device Installer, AdvPack (для установки компонентів Active через Internet Explorer) і Windows Installer.

– HTMLHelp (Microsoft Compressed HTML Help, Microsoft Compiled HTML Help, .CHM) – пропріетарний формат файлів контекстної довідки, розроблений корпорацією Microsoft і випущений в 1997 році як заміна формату WinHelp. Містить у собі набір HTML-сторінок, може також містити в собі зміст із посиланнями на сторінки, предметний покажчик, а також базу для повнотекстового пошуку по вмісту сторінок. Всі вхідні в.CHM файли стислі алгоритмом LZX. Для перегляду .CHM-Файлів використовується стандартний засіб перегляду, убудоване в усі версії Microsoft Windows, починаючи з Windows 98, і Windows NT. Крім того, існує ряд сторонніх програм-проглядачів, FBReader, etc. Для створення .CHM-Файлів «Майкрософт» надає безкоштовний засіб HTML Help Workshop.

– deb – розширення імен файлів «бінарних» пакетів для поширення й установки програмного забезпечення в ОС проекту Debian, і інших, що використовують систему керування пакетами dpkg.

– JAR файл – це Java-архів. Являє собою звичайний ZIP-архів, у якому втримується частина програми мовою Java. Щоб JAR файл був що виконується, він повинен містити файл MANIFEST.MF у каталозі META-INF, у якому повинен бути зазначений головний клас програми (такий клас повинен містити метод main). Номер версії JAR задається параметром Manifest-Version і є обов'язковим. В SDK 1.2 значення цього параметра повинне бути дорівнює 1.0.

– LHA – безкоштовний архіватор і відповідний формат архівування файлів (які мають розширення ім'я .LZH). Як і прабатько, розроблявся для архівування текстових файлів.

– Windows Imaging Format (WIM) – це файл-орієнтований формат образа диска. Формат був розроблений компанією Microsoft для розгортання останніх релізів операційних систем сімейства Windows – Windows Vista/7 і Windows Server 2008, які використовують його як частину стандартної процедури установки. Втім, його можна використовувати й з іншими релізами Windows; крім того він застосовується в Windows Fundamentals for Legacy PCs – компактної ОС від Microsoft для застарілих PC, створеної на базі Microsoft Windows XP Embedded Service Pack 2 і вийшла 8 липня 2006 року.

– xar (eXtensible ARchive format) – це формат стиску даних і архівування файлів з відкритим вихідним кодом. Файл у цьому форматі звичайно має розширення .xar і зберігає в стислому або незжатому виді один або кілька файлів, які можна з нього витягти шляхом розпакування за допомогою спеціальної програми. XAR був створений у рамках проекту OpenDarwin і використовується в операційній системі Mac OS X 10.5 для процедури установки програмного забезпечення.

– tar – формат бітового потоку або файлу архіву, а також назва традиційної для Unix програми для роботи з такими архівами. Програма tar була

стандартизована в POSIX.1-1998, а також пізніше в POSIX.1-2001. Спочатку програма tar використовувалася для створення архівів на магнітній стрічці, а в наш час tar використовується для зберігання декількох файлів усередині одного файлу, для поширення програмного забезпечення, а також по прямому призначенню – для створення архіву файлової системи. Одним з переваг формату tar при створенні архівів є те, що в архів записується інформація про структуру каталогів, про власника й групу окремих файлів, а також тимчасові мітки файлів.

Алгоритми стиску без втрат підрозділяються на наступні:

– gzip (скорочення від GNU Zip) – утиліта стиску й відновлення (декомпресії) файлів, що використовує алгоритм DEFLATE. Використовується в основному в UNIX-системах, у ряді яких є стандартом де-факто для стиску даних.

– LZ77 і LZ78 – алгоритми стиску без втрат, опубліковані в статтях Абрахама Лемпеля і Якоба Зіва в 1977 і 1978 роках. Ці алгоритми найбільш відомі варіанти в сімействі LZ*, що містить у собі також LZW, LZSS, LZMA і інші алгоритми. Обидва алгоритми відносяться до словникових методів, на відміну від інших методів зменшення надмірності, таких як RLE і арифметичний стиск. LZ77 є алгоритмом з «ковзним вікном», що еквівалентно неявному використанню словникового підходу, уперше запропонованого в LZ78.

– LZMA (Lempel-Ziv-Markov chain-Algorithm) – алгоритм стиску даних, розроблювальний з 2001 року, використовується в архіваторі 7-Zip для створення стислих архівів у форматі 7z. Алгоритм заснований на схемі стиску даних по словнику, подібній з використаною в LZ77, і забезпечує високий коефіцієнт стиску (звичайно перевищуючий коефіцієнт, одержуваний при стиску з використанням bzip2), а також дозволяє використовувати словники різного розміру (до 4 Гб).

– MTF – Рух до початку (move-to-front, MTF) – перетворення для кодування даних (звичайно потоку байтів) розроблене для поліпшення продуктивності ентропійного кодування. При гарній реалізації, воно достатньо

швидко для включення як додатковий крок в алгоритмах стиску даних. Також може використовуватися разом з BWT, перетворенням Барроуза-Уилера.

– bzip2 – безкоштовна вільна утиліта командного рядка (а також алгоритм) з відкритим вихідним кодом для стиску даних. Розроблено й уперше опублікована Джуліаном Сьюардом у липні 1996 (версія 0.15). Стабільність і популярність компресора росли протягом декількох років, і версія 1.0 була опублікована наприкінці 2000 року.

– Deflate – це алгоритм стиску без втрат, що використовує комбінацію алгоритму LZ77 і алгоритму Хаффмана. Він був описаний Філом Кацом для 2-ї версії своєї утиліти для створення архівів PKZIP, що згодом був визначений в RFC 1951. Deflate вважається вільним від всіх існуючих патентів, і поки патент на LZW (який використовується у форматі GIF) залишався в силі, це привело до використання deflate у файлах, стисливих gzip, і зображеннях у форматі PNG вдобавок до формату ZIP, для якого Кац його спроектував.

– PAQ – серія вільних архіваторів з текстовим інтерфейсом, які спільними зусиллями розроблювачів піднялися в перші місця рейтингів багатьох тестів стиску даних (хоча й ціною процесорного часу й обсягу пам'яті). Кращий результат у цій серії на більшості тестів був отриманий архіватором PAQ8JD, створеним спільними зусиллями Метта Махони, Олександра Ратушняка, Сергія Оснача, Пшемислава Скибиньского й Білла Петтиса, і випущеним 30 грудня 2006 року. Однак, у деяких тестах він відстає від WinRK (створеного Малькомом Тейлором у січні 2005 року) у режимі PWCM. PWCM (PAQ weighted context mixing, «PAQ зважене змішування контекстів») – стороння пропрієтарна реалізація алгоритму PAQ. Спеціально настроєні версії алгоритму PAQ виграли призи в Приз Хаттера й Калгари Корпус Челлендж.

– ROLZ (Reduced Offset LZ – алгоритм Лемпела-Зива зі скороченими зсувами) – словниковий алгоритм стиску даних, близький до LZ77, але використовуючий деякі контекстні прийоми для зменшення числа активних зсувів. Саме поняття ROLZ уперше ввів Malcolm Taylor у своєму архіваторі RK в

1999 році й даний алгоритм є одним з найбільш сучасних підходів до побудови швидких ефективних алгоритмів стиску.

– Алгори́тм Ле́мпеля-Зіва-Ве́лча (Lempel-Ziv-Welch, LZW) – це універсальний алгоритм стиску даних без втрат, створений Абрахамом Лемпелем (Abraham Lempel), Якобом Зівом (Jacob Ziv) і Терри Велчем (Terry Welch). Він був опублікований Велчем в 1984 році, як поліпшена реалізація алгоритму LZ78, опублікованого Лемпелем і Зівом в 1978 році. Алгоритм розроблений так, щоб його можна було швидко реалізувати, але він не обов'язково оптимальний, оскільки він не проводить ніякого аналізу вхідних даних.

– Алгоритм Шеннона-Фано – один з перших алгоритмів стиску, що вперше сформулювали американські вчені Шеннон і Фано. Даний метод стиску має велику подібність із алгоритмом Хаффмана, що з'явився на кілька років пізніше. Алгоритм використовує коди змінної довжини: символ, що часто зустрічається, кодується кодом меншої довжини, що рідко зустрічається – кодом більшої довжини. Коди Шеннона-Фано префіксні, тобто, ніяке кодове слово не є префіксом будь-якого іншого. Ця властивість дозволяє однозначно декодувати будь-яку послідовність кодових слів.

– PPM (Prediction by Partial Matching – пророкування по частковому збігу) – адаптивний статистичний алгоритм стиску даних без втрат, заснований на контекстному моделюванні й пророкуванні. Модель PPM використовує контекст – безліч символів у незжатому потоці, що передують даному, щоб пророкувати значення символу на основі статистичних даних. Сама модель PPM лише пророкує значення символу, безпосередній стиск здійснюється алгоритмами ентропійного кодування, як наприклад, алгоритм Хаффмана, арифметичне кодування. Довжина контексту, що використовується при пророкуванні звичайно сильно обмежена. Ця довжина позначається n і визначає порядок моделі PPM, що позначається як PPM(n). Необмежені моделі так само існують і позначаються просто PPM*. Якщо пророкування символу по контексту з n символів не може бути зроблено, то відбувається спроба пророчити його за

допомогою n-1 символів. Рекурсивний перехід до моделей з меншим порядком відбувається поки пророкування не відбудеться в одній з моделей, або коли контекст стане нульової довжини (n=0). Моделі ступеня 0 і -1 варто описати особливо. Модель нульового порядку еквівалента случаю контекстно-вільного моделювання, коли імовірність символу визначається винятково із частоти його появи в стисливому потоці даних. Подібна модель звичайно застосовується разом з кодуванням по Хаффману. Модель порядку -1 представляє собою статичну модель, яка присвоює імовірності символу визначене фіксоване значення значення; звичайно всі символи, які можуть зустрітися в стисливому потоці даних, при цьому вважаються рівноімовірнми. Для одержання доброї оцінки імовірностей символу необхідно враховувати контексти різних довжин. PPM представляє собою варіант стратегії перемішування, коли оцінки імовірностей, зроблені на підставі контекстів різних довжин, поєднуються в одну загальну імовірність. Отримана оцінка кодується будь-яким ентропійним кодером (ЕК), звичайно це якийсь різновид аріфметичного кодера. На етапі ентропійного кодування й відбувається властиво стиск. Велике значення для алгоритму PPM має проблема обробки нових символів, що ще не зустрічалися у вхідному потоці. Це проблема зветься проблема нульової частоти. Деякі варіанти реалізацій PPM думають лічильник нового символу рівним фіксованій величині, наприклад, одиниці. Інші реалізації, як наприклад, PPM-D, збільшують псевдолічильник нового символу щораз, коли, дійсно, у потоці з'являється новий символ. (Інакше кажучи, PPM-D оцінює ймовірність появи нового символу як відношення числа унікальних символів до загального числа використовуваних символів). Опубліковані дослідження алгоритмів сімейства PPM з'явилися в середині 1980-х років. Програмні реалізації не були популярні до 1990-х років, тому як моделі PPM вимагають значну кількість оперативної пам'яті. Сучасні реалізації PPM є кращими серед алгоритмів стиску без втрат для текстів природною мовою.

– Дельта-кодування (Delta encoding) – спосіб збереження або передачі даних у формі різниці (дельти) між послідовними даними замість самих даних.

Це часто називається дельта-компресія, так як деякі зразки кодування можуть одержувати кодовані дані в більше короткому виді, чим вихідні дані.

– Алгоритм Хаффмана – адаптивний жадібний алгоритм оптимального префіксного кодування алфавіту з мінімальною надмірністю. Був розроблений в 1952 році аспірантом Массачусетского технологічного інституту Девідом Хаффманом при написанні їм курсової роботи. У цей час використовується в багатьох програмах стиску даних. На відміну від алгоритму Шеннона-Фано, алгоритм Хаффмана залишається завжди оптимальним і для вторинних алфавітів m_2 з більш ніж двома символами.

– Кодування довжин серій (Run-length encoding, RLE) або Кодування повторів – простий алгоритм стиску даних, що оперує серіями даних, тобто послідовностями, у яких той самий символ зустрічається кілька разів підряд. При кодуванні рядок однакових символів, що становлять серію, заміняється рядком, що містить сам повторюваний символ і кількість його повторів.

– Перетворення Барроуза-Уілера (Burrows-Wheeler transform, BWT, також історично називається стиском, що блочно-сортує, хоча стиском і не є) – це алгоритм, використовуваний у техніках стиску даних для перетворення вихідних даних. BWT використовується в архіваторі bzip2. Алгоритм був винайдений Майклом Барроузом і Девідом Уілером. Терміном BWT також називають і повні алгоритми стиску, що використовують BWT як один із кроків.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Блок інтерфейсу користувача програми.

– Блок резервування.

– Блок швидкого доступу до елементів програми.

– Блок роботи з файлами.

– Блок виконання команд.

– Блок операцій.

– Блок параметрів.

– Довідка.

Блок резервування складається з наступних елементів:

1. Резервне копіювання. Регулярне архівування дозволяє знизити обсяг файлів для резервного копіювання. У випадку останнього доводиться враховувати розходження між файлами й базами даних, а також – в організаційному плані – між циклом резервного копіювання й наданням даних. Стандартно передбачається повне резервне копіювання (Full Backup) один раз у тиждень і додаткове щоденне інкрементальне копіювання. Останнє веде до того, що при повнім відновленні даних їхній обсяг збільшується, іноді дворазово. Тому зазначений спосіб найбільш придатний для невеликої кількості даних. При значних обсягах інтервал часу для відновлення обмежений, тому виконується лише щоденне повне резервне копіювання. Застосування «синтетичного резервного копіювання» дозволяє уникнути необхідності щоденного повного перенесення даних із клієнта на резервний ландшафт. Замість цього спочатку здійснюється повне резервне копіювання, а потім – інкрементальне, коли застарілі дані повної копії заміняються на нові. Далі система дублює цю «синтетичну» резервну копію з кешу на стрічку у вигляді повної фізичної резервної копії. У результаті в користувача повинна зберігатися остання повна резервна копія на дисковому кеші або ж найбільш близька за часом остання повна резервна копія на стрічці, включаючи всю інформацію й дані, необхідні для повного відновлення. Для підвищення продуктивності процесу резервного копіювання користувальницькі дані направляються в стрічкові бібліотеки через дисковий кеш. Крім стрічкових систем, все частіше зустрічаються дискові системи зберігання, здатні емулювати відповідні стрічкові системи (віртуальні стрічкові бібліотеки).

2. Резервне копіювання баз даних. При резервному копіюванні баз даних необхідно звертати увагу не тільки на створення віддаленої копії, але й на її погодженість. Як правило, бази даних містять великі обсяги інформації, тому процес резервного копіювання затягається надовго. Тим часом у вихідну базу даних можуть бути внесені зміни. Існує велика ймовірність того, що області з файлами, копії яких уже створені, зміняться, а в ще не збережених сегментах з'являться нові метадані. Така резервна копія абсолютно марна – її не можна відновити до функціональної бази даних. Для протидії цьому явищу постачальники відповідних рішень розробили інтерфейси оперативного резервного копіювання, такі як Oracle Recovery Manager (RMAN). Тепер разом з інтерфейсами до клієнтських програм резервного копіювання дані можна зберігати прямо – з бази даних на носій резервної копії. Одночасно механізми RMAN дозволяють зберегти все, що необхідно для забезпечення погодженості резервних копій.

3. Архівування й відновлення стану системи. При цьому будуть архівуватися наступні дані:

– системний реєстр;

– база даних зареєстрованих класів об'єктів (Class Registration);

– системні завантажувальні файли;

– база даних служб сертифікатів (тільки на серверах, на яких установлена служба сертифікатів);

– база даних Active Directory і папка SYSVOL (на контролерах доменів).

Для архівування стану системи, а також для наступного відновлення, обов'язково потрібні права адміністратора даного комп'ютера. Відновлення Active Directory необхідно виконувати тільки при завантаженні системи в режимі відновлення служб каталогів (запуск меню вибору режими завантаження операційної системи вибираються в початковий момент завантаження натисканням клавіші F8). База даних Active Directory – це інформація, що ставиться до Каталогу, включаючи об'єкти й атрибути домена, схему,

конфігурацію й інформацію Глобального Каталогу. Базу даних Active Directory розглядають як транзакційну, що означає, що кожна зміна в ній виконується як окрема транзакція (транзакція – неподільна операція, тобто поки обидві сторони, що приймають участь у транзакції, не завершать своєї частини її обробки, транзакція не вважається завершеною). Ця природа бази даних допомагає підтримувати її цілісність у випадку збоїв. База даних Active Directory складається з декількох файлів:

– Ntds.dit – це файл бази даних, у якому зберігаються всі Об'єкти; за замовчуванням цей файл (і інші, зазначені тут) розташований у папці "%systemroot%\NTDS".

– Edb*.log – цей файл є журналом транзакцій; перш ніж будь-яка зміна буде записано в базу даних, інформація про нього спочатку заноситься в журнал транзакцій; кожний файл edb*.log має розмір 10 МБ, за замовчуванням використовується "кругове" ведення журналу, тобто якщо журнал заповнений, то файл починає перезаписувати дані про самі старі зміни; якщо "кругове" ведення журналу відключене, те після заповнення файлу він перейменовується у файл edbxxxxx.log, із цифрами xxxxx, що представляють його порядковий номер, починаючи з 00001.

– Edb.chk – це файл контрольних крапок, використовуваний AD для відстеження змін, записуваних у файл ntds.dit ; він використовується для цілей відновлення (наприклад, якщо контролер домену ушкоджений і інформація про зміни не заноситься в базу даних, файл контрольних крапок служить як маркер, що відзначає, які із записів у журналі повинні бути записані в базу даних надалі).

– Res1.log і Res2.log – є резервними файлами журналів, по 10 МБ кожного. Їхнє призначення – дозволити Active Directory продовжувати ведення журналу змін у випадку, якщо на жорсткому диску не залишається вільного місця, тому в резерві завжди залишається 20 МБ вільного дискового простору, що використовується тільки якщо буде потреба.

4. Автоматичне аварійне відновлення системи. На відміну від резервного копіювання стану системи, при якому зберігається тільки частина файлів операційної системи, резервне копіювання для автоматичного аварійного відновлення системи (ASR, Automated System Recover) архівує більший обсяг інформації – практично весь тім, на якому встановлена операційна система. І процедура відновлення системи стає більше складною.

Блок швидкого доступу до елементів програми включає до себе наступні елементи:

– Додати в архів.

– Витягти з архіву.

– Тест.

– Перегляд архіву.

– Видалення.

– Знайти.

– Майстер роботи з архівом.

– Інформація.

– Виправити помилки в архіві.

Блок роботи з файлами включає в себе:

– Відкрити архів.

– Вибрати диск.

– Встановити пароль.

– Скопіювати файли у буфер обміну.

– Вставити файли з буферу обміну.

– Виділити усе.

– Зняти виділення.

– Вихід.

Блок виконання команд включає в себе наступні команди:

– Додати файли у архів.

– Витягти у вказану папку.

– Протестувати файли у архіві.

– Переглянути файл.

– Видалити файл.

– Перейменувати файл.

– Файл на друк.

– Додати архівний коментар.

– Додати інформацію для відновлення.

– Заблокувати архів.

Блок операцій включає у себе наступні операції:

– Майстер архіву.

– Перевірити архіви на віруси.

– Перетворити архіви.

– Відновити архіви.

– Перетворити архіви в SFX.

– Знайти файли.

– Показати інформацію.

– Створити звіт.

Блок параметрів включає в себе настроювання наступних параметрів:

– Настроювання.

– Імпорт/експорт файлу.

– Список файлів.

– Дерево файлів.

– Теми.

Довідка включає в себе наступні пункти:

– Зміст.

– Web-сторінка програми.

– Про програму.

Рисунок 3.2 – Функціональна схема системи

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. Процеси взаємодіють наступним чином.

Спершу запускається процес виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес перегляду списку архівів та файлів на диску.

– Процес вибору файлів/архівів.

– Процес резервування даних.

Процес резервування даних взаємодіє з наступними процесами:

– Процес вибору даних для відновлення, який, у свою чергу, взаємодіє з процесом відновлення даних.

– Процес вибору параметрів резервного копіювання.

Процес вибору параметрів резервного копіювання взаємодіє з процесом вибору даних для резервного копіювання.

Процес вибору даних для резервного копіювання взаємодіє з процесом вибору каталогу для збереження.

Процес вибору каталогу для збереження взаємодіє з процесом резервного копіювання.

Процес вибору файлів/архівів взаємодіє з наступними процесами:

– Процес видалення файлів/архівів.

– Процес переміщення файлів/архівів.

– Процес вилучення файлів з вибраних архівів, який взаємодіє з процесом збереження розархівованих файлів на диску.

– Процес архівування вибраних файлів.

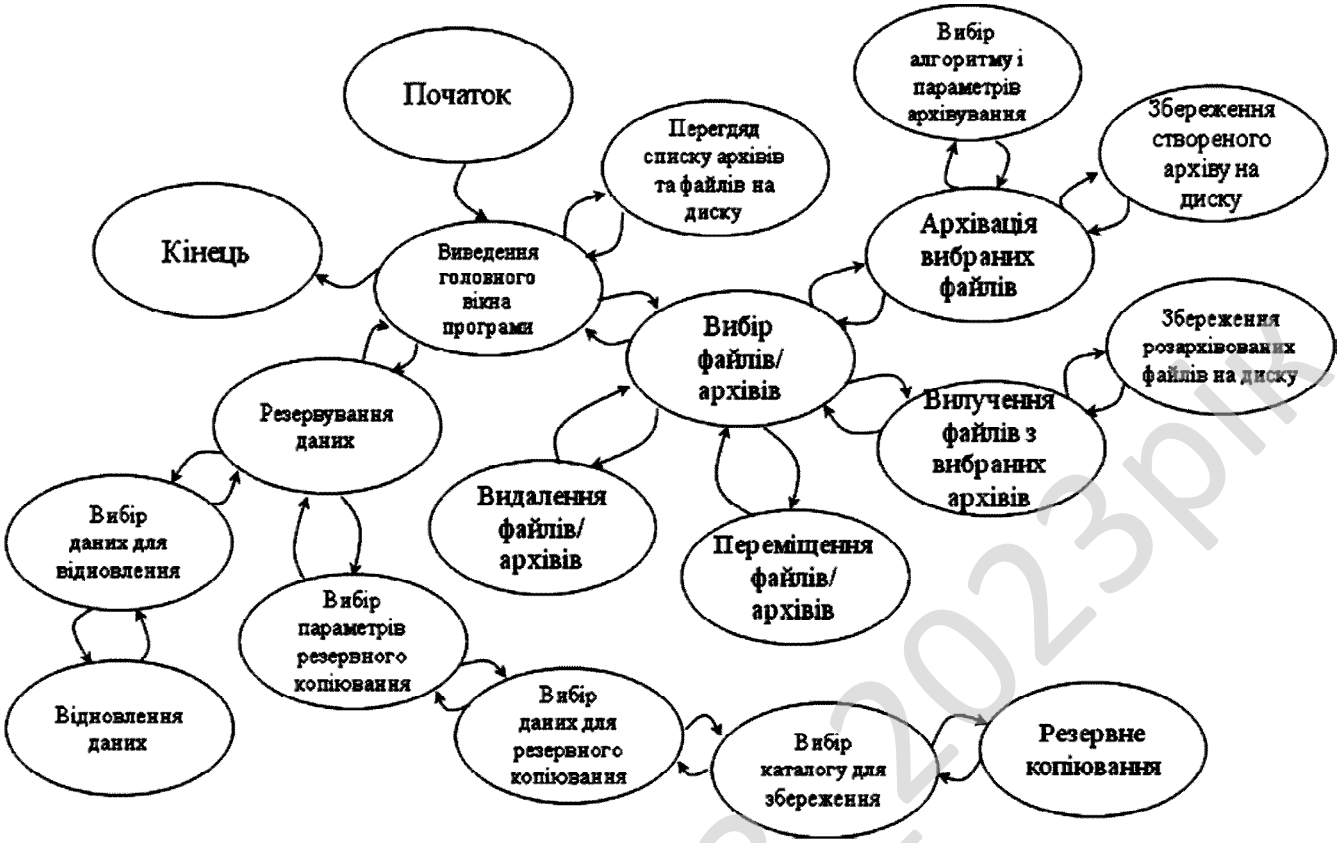

Рисунок 3.3 – Діаграма взаємодії процесів системи

Процес архівування вибраних файлів взаємодіє з наступними процесами:

– Процес вибору алгоритму і параметрів архівування.

– Процес збереження створеного архіву на диску.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається перегляд списку файлів та архівів.

Наступним кроком є вибір файлів/архівів.

Якщо користувач обирає створення архіву, тоді програма виконує наступні дії:

– Вибір алгоритму та параметрів архівування.

– Кодування файлу.

– Збереження архіву.

Якщо користувач обирає розархівування файлів, тоді програма виконує наступні дії:

– Декодування обраних файлів.

– Збереження розархівованих файлів.

Якщо користувач обирає резервне копіювання, тоді програма виконує наступні дії:

– Вибір параметрів резервного копіювання.

– Вибір даних для резервного копіювання.

– Вибір каталогу для збереження.

– Резервне копіювання вибраних даних.

Якщо користувач обирає відновлення даних, тоді програма вибирає та відновлює дані з резервних архівів.

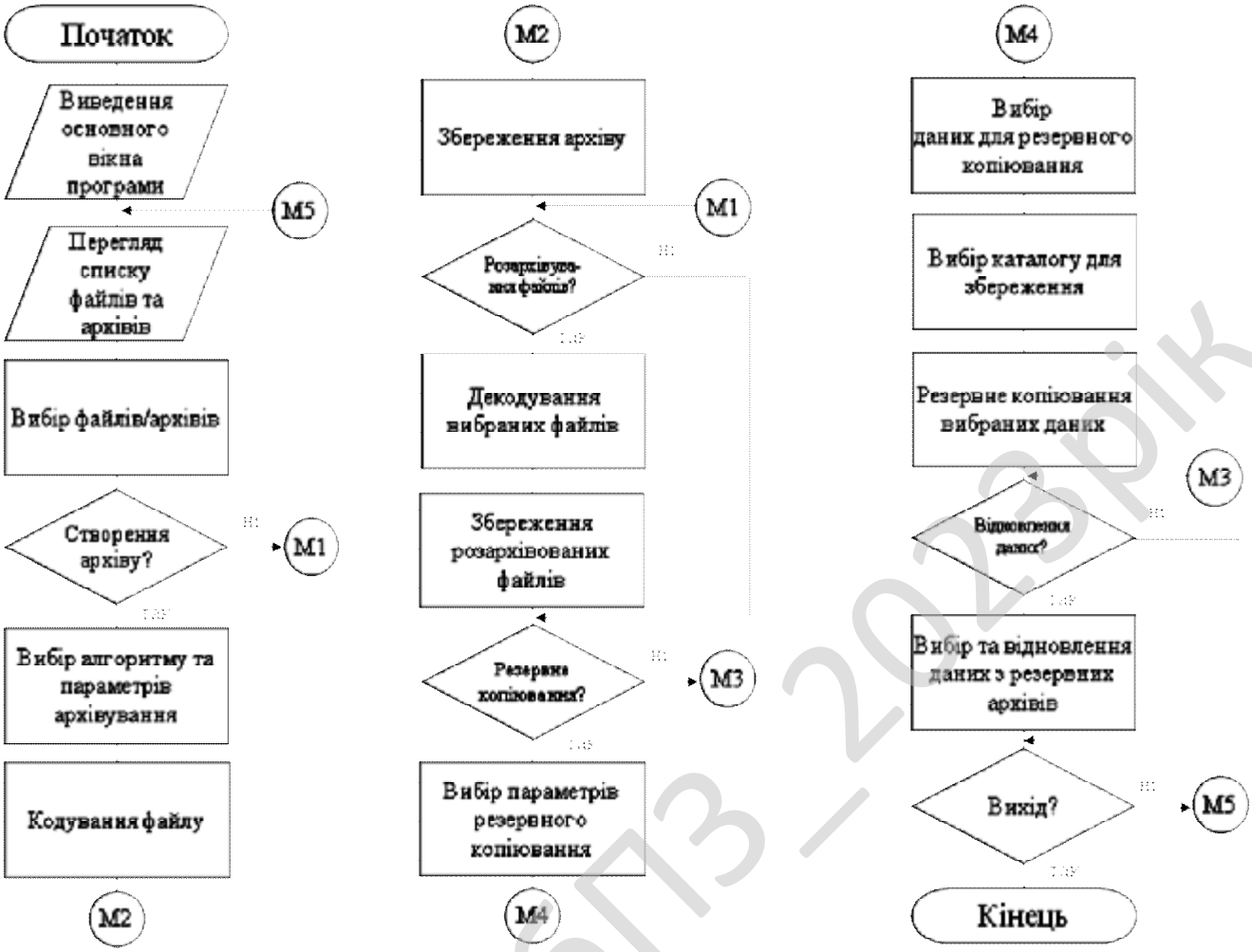

Рисунок 4.1 – Блок-схема роботи основної програми

Після усього цього користувач обирає: працювати йому далі з програмою, або ні.

На рисунку 4.2 зображено блок-схему підпрограми архівування алгоритмом Хаффмана. Вона працює наступним чином:

– Підраховується кількість входжень всіх символів у файл.

– Сортуються символи вхідного алфавіту за частотою появи по убуванню.

– Обираються два послідовні символи з кінця списку.

– Створюється нова змінна.

– Новій змінній присвоюється сума частот появи двох обраних символів.

– Обрані символи видаляються зі списку.

– Змінна додається в список та відбувається його повторне сортування.

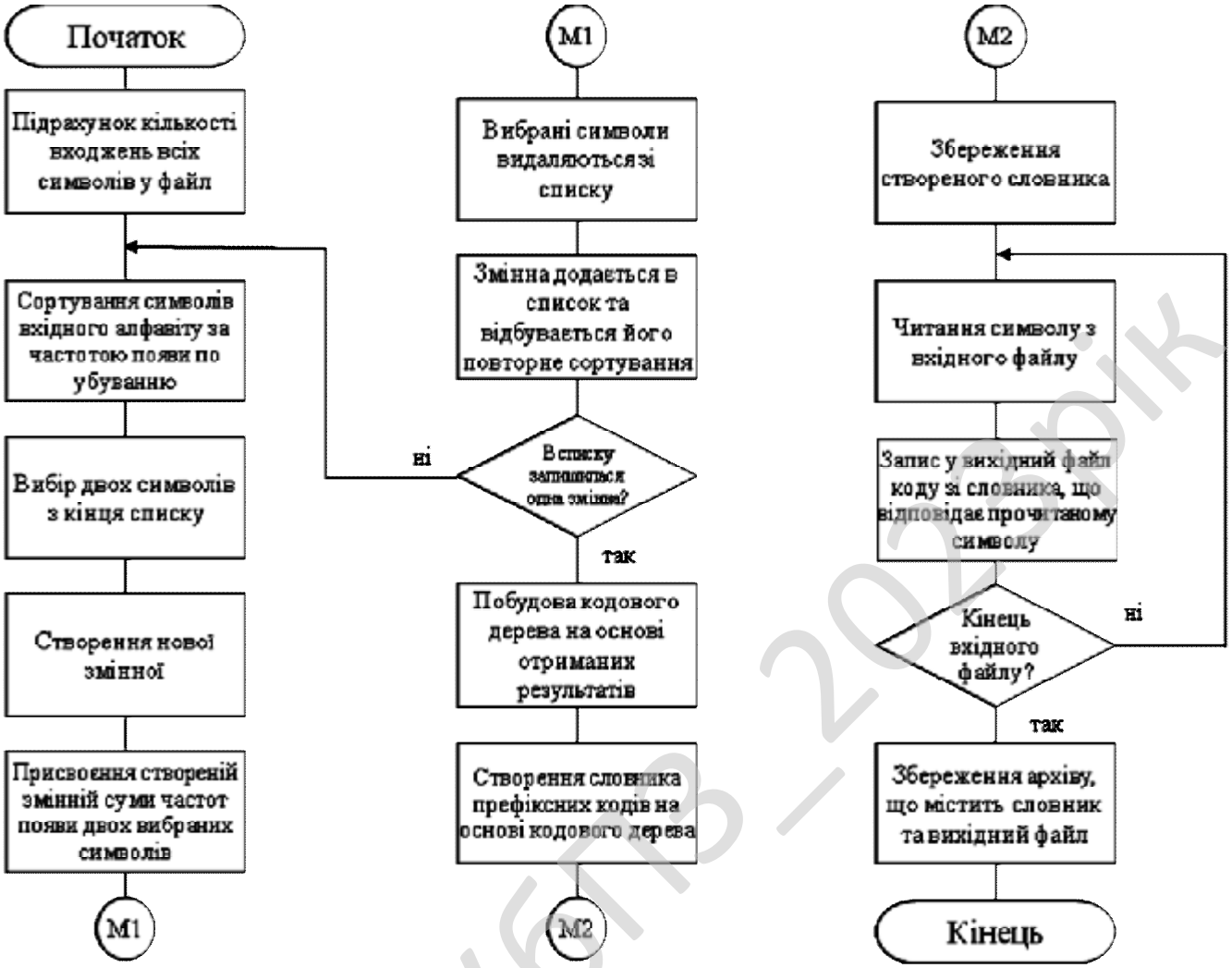

Рисунок 4.2 – Блок-схема підпрограми архівування алгоритмом Хаффмана

Якщо в списку залишилася тільки одна змінна, тоді виконуються наступні дії:

– Будується кодове дерево на основі отриманих результатів.

– Створюється словник префікс них кодів на основі кодового дерева.

– Зберігається створений словник.

– Читається символ з вхідного файлу.

– Відбувається запис у вихідний файл коду зі словника, що відповідає прочитаному символу.

Якщо досягається кінець файлу, тоді відбувається збереження архіву, що містить словник та вихідний файл.

Наведемо частину коду, яка реалізує алгоритм Хаффмана.

```
unit CHuffman;
interface
uses
  SysUtils, Classes, Dialogs;
const
  ALPHABETSIZE = 256;
type
  TTree = class;
  THuffman = class;
  TTree = class(TObject)
  private
    fchild0 : TTree;      // нащадки "0" і "1"
    fchild1 : TTree;
    fleaf : Boolean;      // ознака листового дерева
    fcharacter : Integer; // вхідний символ
    fweight : Integer;    // вага цього символу
  public
    procedure Tree(character, weight : integer; leaf : boolean);
    procedure traverse(code : string; h : THuffman);
    property child0 : TTree read fchild0 write fchild0;
    property child1 : TTree read fchild1 write fchild1;
    property leaf : Boolean read fleaf write fleaf;
    property character : Integer read fcharacter write fcharacter;
    property weight : Integer read fweight write fweight;
  end;
  THuffman = class(TObject)
  private
    // поля :)
    fweights : array [0.. ALPHABETSIZE-1] of Integer;// ваги символів
    fcode : array [0.. ALPHABETSIZE-1] of String;// коди Хаффмана
    ftree : array [0.. ALPHABETSIZE-1] of TTree;// робочий масив дерев
    // методи доступу до масиву :)
    function GetCodeValue(Index: Integer): String;
    function GetTreeValue(Index: Integer): TTree;
    function GetWeightValue(Index: Integer): Integer;
    procedure SetCodeValue(Index: Integer; const Value: String);
    procedure SetTreeValue(Index: Integer; const Value: TTree);
    procedure SetWeightValue(Index: Integer; const Value: Integer);
  public
    // методи
```

```
    procedure makeCode;
    procedure growTree(var data : array of Integer);
    function getLowestTree(used : integer) : integer;
    function coder(var data : array of integer) : string;
    function decoder(data : string) : string;
    // властивості
    property weights[Index: Integer] : Integer read GetWeightValue write
SetWeightValue;
    property code[Index: Integer] : String read GetCodeValue write SetCodeValue;
    property tree[Index: Integer] : TTree read GetTreeValue write SetTreeValue;
  end;
var
  Huffman : THuffman;
implementation
{ TTree }
procedure TTree.Tree(character, weight: integer; leaf: boolean);
begin
  fleaf := leaf;
  fcharacter := character;
  fweight := weight;
end;
(*  Обхід дерева з генерацією кодів
    1. "Роздрукувати" листове дерево й записати код Хаффмана в масив
    2. Рекурсивно обійти ліве піддерево (з генеруванням коду).
    3. Рекурсивно обійти праве піддерево. *)
procedure TTree.traverse(code: String; h: THuffman);
begin
  if leaf then
  begin
    h.code[Ord(character)] := code;
    ShowMessage('Символ: ' + Chr(character) +'  '+'Вага: '+IntToStr(weight) +'
'+'Двійковий код: '+ code);
  end;
  if child0 <> nil then child0.traverse(code + '0', h);
  if child1 <> nil then child1.traverse(code + '1', h);
end;
{ THuffman }
(* шукаємо саме "легке" дерево *)
function THuffman.getLowestTree(used: integer): integer;
var
  min, i : Integer;
begin
```

```
  min := 0;
  for i:=1 to used-1 do
   if tree[i].weight < tree[min].weight
    then min := i;   Result := min;
 end;
(* кодує дані рядком з 1 і 0 *)
function THuffman.coder(var data: array of Integer): String;
var   str : String;
        i : Integer;
begin
str := '';
   for i:=0 to High(data) do
     str := str + code[data[i]];
     Result := str;
end;
function THuffman.decoder(data : string): string;
 var   str : String;
         c : Integer;
 begin
   str := '';
 while(length(data) > 0) do
  begin
    for c:=0 to ALPHABETSIZE-1 do
    if (weights[c] > 0) AND (code[c] = copy(data, 1, length(code[c]))) then
    begin
      data := copy(data, length(code[c])+1, length(data));
      str := str + chr(c) + ' - ' + code[c] + #13;
    end;
  end;
  Result := str;
end;
(* ростимо дерево *)
procedure THuffman.growTree(var data: array of Integer);
var
  i, c, w, min, used, weight0 : Integer;
  temp : TTree;
begin
  for i:=0 to ALPHABETSIZE-1 do weights[i] := 0;
  for i:=0 to High(data) do weights[data[i]] := weights[data[i]] + 1; // уважаємо
ваги символів
  //  заповнюємо масив з "листових" дерев
  //  с використаними символами
```

```
  used := 0;
  for c:=0 to ALPHABETSIZE-1 do
  begin
    w := weights[c];
    if w <> 0 then
    begin
      inc(used);
      tree[ used-1] := TTree.Create;
      tree[ used-1].Tree(c, w, true);
    end;
  end;
  while used > 1 do                          // парами зливаємо легені вітки
  begin
    min := getLowestTree(used);              // шукаємо 1 вітку
    weight0 := tree[min].weight;
    temp := TTree.Create;                    // створюємо нове дерево
    temp.child0 := tree[min];                // і прищеплюємо 1 вітку
    dec(used);
    tree[min] := tree[used];                 // на місце 1 вітки кладемо
                                             // останнє дерево в списку
    min := getLowestTree(used);              // шукаємо 2 вітку й
    temp.child1 := tree[min];                // прищеплюємо її до нов.дер.
    temp.weight := weight0 + tree[min].weight; // уважаємо вагу нов.дер.
    tree[min] := temp;                       // нов.дер. кладемо на місце 2
вітки
  end;                                       // все! залишилося 1 дерево
Хаффмана
end;
(* запускаємо обчислення кодів Хаффмана *)
procedure THuffman.makeCode;
begin
  tree[0].traverse('', self);
end;
function THuffman.GetCodeValue(Index: Integer): String;
begin
  Result := fcode[Index];
end;
function THuffman.GetTreeValue(Index: Integer): TTree;
begin
  Result := ftree[Index];
end;
function THuffman.GetWeightValue(Index: Integer): Integer;
```

```
begin
  Result := fweights[Index];
end;
procedure THuffman.SetCodeValue(Index: Integer; const Value: String);
begin
  fcode[Index] := Value;
end;
procedure THuffman.SetTreeValue(Index: Integer; const Value: TTree);
begin
  ftree[Index] := Value;
end;
procedure THuffman.SetWeightValue(Index: Integer; const Value: Integer);
begin
  fweights[Index] := Value;
end;
end.
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтовых ключів.

3. Для шифрування 10-байтового блоку:

а. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують

операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено основне вікно програми. Воно складається з наступних блоків:

– Блок меню.

– Блок кнопок швидкого доступу до елементів програми.

– Вікно обирання файлів, які потрібно за архівувати/розахівувати.

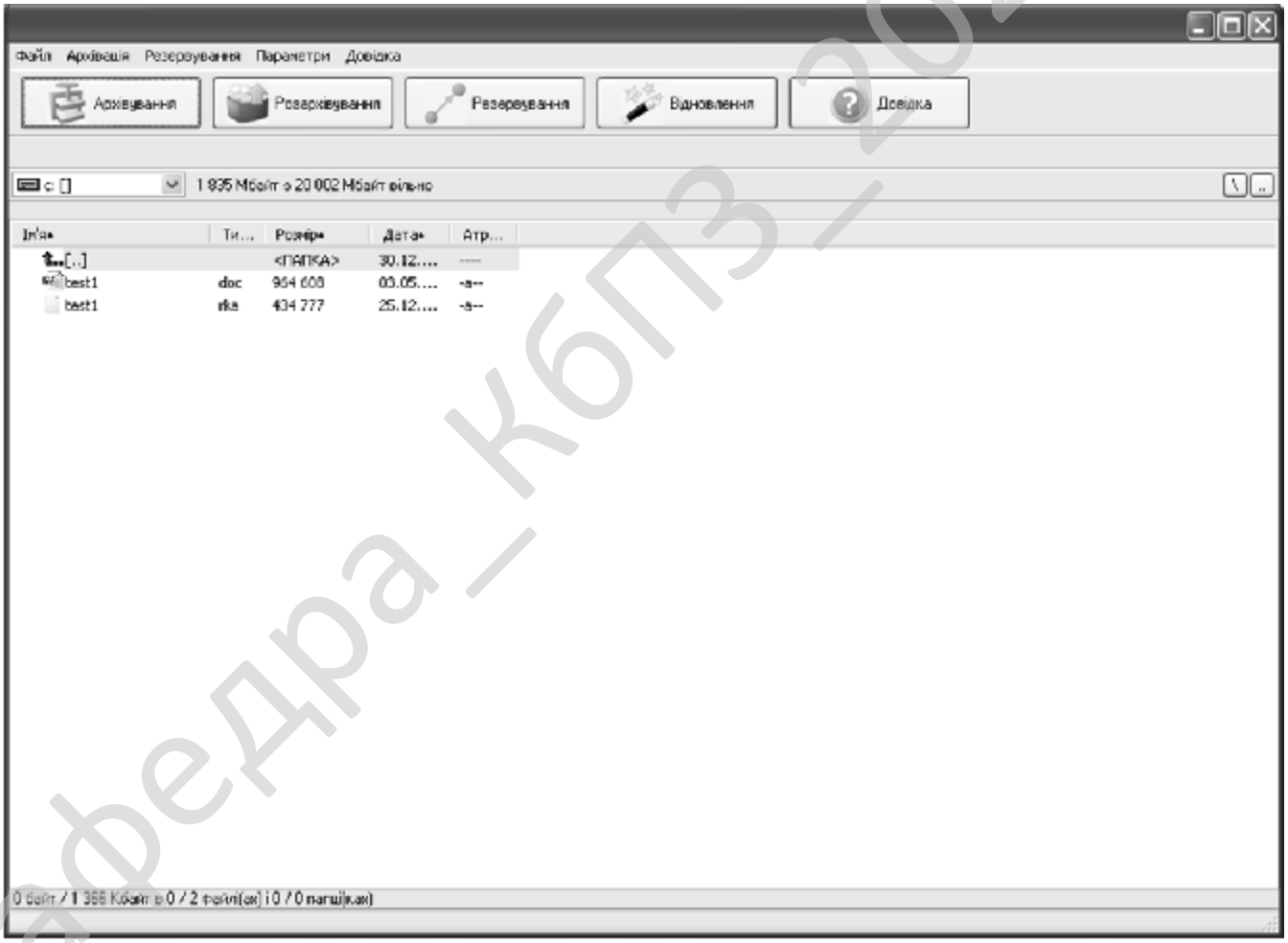

Рисунок 5.1 – Основне вікно програми

Блок меню складається з наступних елементів:

– Файл.

– Архівація.

– Резервування.

– Параметри.

– Довідка.

Блок кнопок швидкого доступу до елементів програми складається з наступних елементів:

– Архівування.

– Розархівування.

– Резервування.

– Відновлення.

– Довідка.

На рисунку 5.2 зображено процес архівації.

На рисунку 5.3 зображено вікно довідки, з якої надається інформація про керівника проекту, виконавця проекту, тему проекту та місце виконання проекту.

Рисунок 5.2 – Процес архівації

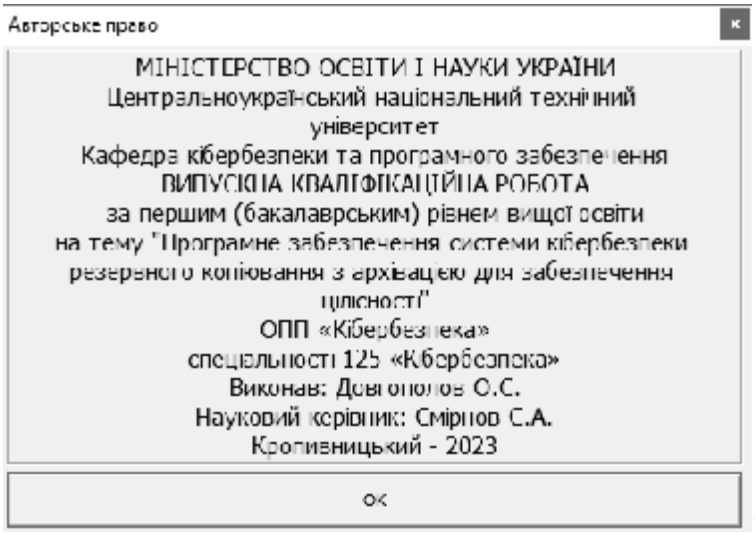

Рисунок 5.3 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем резервного копіювання з архівацією для забезпечення цілісності.

– Досліджена система резервного копіювання з архівацією для забезпечення цілісності.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання резервного копіювання з архівацією для забезпечення цілісності.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки резервного копіювання з архівацією для забезпечення

цілісності. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи кібербезпеки Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

2. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

3. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

4. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

5. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

6. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

7. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

8. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

9. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

10. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings.* Volume 2740, 2020, Pages 102-114. **(Scopus)**.

11. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

12. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

14. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

15. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

16. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus)**.

17. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

18. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

19. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

20. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

21. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. (**Scopus)**.

22. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

23. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019. **(Scopus)**.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629. **(Scopus)**.

25. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

27. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

28. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

30. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

31. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

32. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

35. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного

процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

39. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

40. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

41. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

42. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

43. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

44. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь - 2020. - № 3. - С. 50-61.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

46. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

47. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 13-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 76 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки резервного копіювання з архівацією для забезпечення цілісності

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

Файл frFilePanel.pas - інтерфейс користувача

```
unit frFilePanel;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, StdCtrls, ExtCtrls, FileCtrl,
  Files, Buttons;

Type
  TColumnsSize=Array [0..4] Of Integer;
  TDeactivateProcedure=Procedure Of Object;
  TfrFilePanel = class(TFrame)
    pnDrives: TPanel;
    pnDriveInfo: TPanel;
    lbCurrentPath: TLabel;
    lvFiles: TListView;
    pnFilesInfo: TPanel;
    dcbxDrive: TDriveComboBox;
    btDirRoot: TButton;
    btDirUp: TButton;
    lbDriveInfo: TLabel;
    bbRefresh: TBitBtn;

    procedure lvFilesColumnClick(Sender: TObject; Column: TListColumn);
    procedure dcbxDriveChange(Sender: TObject);
    procedure lvFilesKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure lvFilesDblClick(Sender: TObject);
    procedure btDirUpClick(Sender: TObject);
    procedure btDirRootClick(Sender: TObject);
    procedure lvFilesColumnRightClick(Sender: TObject; Column: TListColumn;
      Point: TPoint);
    procedure lvFilesEditing(Sender: TObject; Item: TListItem;
      var AllowEdit: Boolean);
    procedure lvFilesChange(Sender: TObject; Item: TListItem;
      Change: TItemChange);
    procedure lvFilesEnter(Sender: TObject);
    procedure lbCurrentPathClick(Sender: TObject);
    procedure bbRefreshClick(Sender: TObject);
  private
    { Private declarations }
    AllFiles:TFiles;

    SecondEdit:Boolean;
    Inited:Boolean;

    LastCaption, LastExt:String;

  public
    { Public declarations }
    flbxFiles:TFileListBox;
    CurrentFullPath:String;
    CurrentDrive:Char;
    CurrentPath:String;
    NowRoot:Boolean;

    SortColumn:Byte;
    SortAscending:Boolean;
    NowActive:Boolean;
    UseCopyToDir:String;
    OtherPanelDeactivate:TDeactivateProcedure;
    lbPathEx, lbItemEx:TLabel;
```

```
    Procedure Init(FilesListBox:TFileListBox; ImageList:TImageList;
Deactivation:TDeactivateProcedure; lbPath, lbItem:TLabel);
    Procedure Done;

    Procedure MakeOutLabels;
    Procedure Activate;
    Procedure Deactivate;
    Procedure CheckActive;

    Procedure Refresh;
    Procedure Sort;

    Procedure SetPath(Path:String);

    Procedure ShowItem(Item:TFileRecord);
    Procedure ShowFiles;

    Procedure GetItemByList(ListItem:TListItem; Var Item:TFileRecord);

    Procedure ShowInfo;
    Function ChangeCheck(ListItem:TListItem):Boolean;

    Procedure SetColumnsSize(ColumnsSize:TColumnsSize);
    Procedure GetColumnsSize(Var ColumnsSize:TColumnsSize);

    Procedure SelectLastItem;

    Function TryRename(ListItem:TListItem; NewName:String):Boolean;
    Function TryOneDelete(Item:TFileRecord):Integer;
    Function TryDelete:Boolean;
    Procedure TryCopyFile;
    Procedure TryMoveFile;

    Procedure EditFile;
    Procedure CreateFolder;

    Procedure SetDrive(Drive:Char);
    Procedure CheckCurrentPath;
  end;


implementation
Uses
  StrConsts, FilesEx, fmErrorDrive, fmNameQuery, fmAnyMessage,
  DeCompressor, Main;

{$R *.DFM}


Procedure TfrFilePanel.Init(FilesListBox:TFileListBox; ImageList:TImageList;
Deactivation:TDeactivateProcedure; lbPath, lbItem:TLabel);
Begin
  AllFiles.Init;

  flbxFiles:=FilesListBox;
  lvFiles.LargeImages:=ImageList;
  lvFiles.SmallImages:=ImageList;
  lvFiles.StateImages:=ImageList;
  SortColumn:=1;
  SortAscending:=True;

  SecondEdit:=False;
  Inited:=True;
  LastCaption:='';
  LastExt:='';
  OtherPanelDeactivate:=Deactivation;
  lbPathEx:=lbPath;
  lbItemEx:=lbItem;
{}
```

```
  Activate;
  SetDrive('C');
End;

Procedure TfrFilePanel.Done;
Begin
  Deactivate;
  AllFiles.Done;
End;

Procedure TfrFilePanel.MakeOutLabels;
Var
  Item:TFileRecord;
Begin
  If lbPathEx<>nil Then
  Begin
    lbPathEx.Caption:=CurrentFullPath;
    lbPathEx.Hint:=CurrentFullPath;
  End;

  If lbItemEx<>nil Then
  Begin
    GetItemByList(lvFiles.ItemFocused, Item);
    lbItemEx.Caption:=GetFullName(Item);
  End;
End;

Procedure TfrFilePanel.Activate;
Begin
  If @OtherPanelDeactivate<>nil Then OtherPanelDeactivate;
  NowActive:=True;
  UseCopyToDir:=ConstCopyToDir;
  lbCurrentPath.Color:=ConstLabelActiveColor;
  lvFiles.SetFocus;
  MakeOutLabels;
End;

Procedure TfrFilePanel.Deactivate;
Begin
  NowActive:=False;
  ConstCopyToDir:=CurrentFullPath;
  lbCurrentPath.Color:=ConstLabelNonActiveColor;
End;

Procedure TfrFilePanel.CheckActive;
Begin

End;

Procedure TfrFilePanel.Refresh;
Var
  i:Integer;
  Item:TFileRecord;
Begin
  CheckCurrentPath;

  AllFiles.Clear;

  flbxFiles.Directory:=CurrentFullPath;
  flbxFiles.Update;
  flbxFiles.FileType:=[ftReadOnly, ftHidden, ftSystem, ftArchive, ftDirectory,
ftNormal];
  If flbxFiles.Items.Count<=0 Then Exit;

  CurrentFullPath:=IncludeTrailingBackslash(flbxFiles.Directory);
  CurrentDrive:=ExtractFileDrive(CurrentFullPath)[1];

  NowRoot:=IsRoot(CurrentFullPath);
```

```
  For i:=0 To flbxFiles.Items.Count-1 Do
  Begin
    GetItemByFileName(flbxFiles.Items[i], Item);
    If IsDirectory(Item) Then
    Begin
      If ((Item.Name<>'.') And (Item.Name<>'..') And (Item.Name<>'')) Then
        AllFiles.Add(Item);
    End
    Else Begin
      AllFiles.Add(Item);
    End;
  End;
  Sort;
  ShowFiles;

  SelectLastItem;
  ShowInfo;

 {$ I-}
  If IOResult<>0 Then MessageBeep(48);
 {$I+}
End;

Procedure TfrFilePanel.Sort;
Var
  i:Integer;
  ColCapt:String;
Begin
  For i:=0 To lvFiles.Columns.Count-1 Do
  Begin
    ColCapt:=lvFiles.Column[i].Caption;
    Delete(ColCapt, Length(ColCapt), 1);
    lvFiles.Column[i].Caption:=ColCapt+ConstNoSort;
  End;
  ColCapt:=lvFiles.Column[SortColumn].Caption;
  Delete(ColCapt, Length(ColCapt), 1);
  If SortAscending Then
    lvFiles.Column[SortColumn].Caption:=ColCapt+ConstSortAscending
  Else
    lvFiles.Column[SortColumn].Caption:=ColCapt+ConstSortDescending;

  Case SortColumn Of
    0: AllFiles.SortByName(SortAscending);
    1: AllFiles.SortByExt(SortAscending);
    2: AllFiles.SortBySize(SortAscending);
    3: AllFiles.SortByDateTime(SortAscending);
    4: AllFiles.SortByAttr(SortAscending);
  Else
    AllFiles.SortByName(SortAscending);
  End;
End;

procedure TfrFilePanel.lvFilesColumnClick(Sender: TObject;
  Column: TListColumn);
begin
  If lvFiles.ItemFocused<>nil Then
  Begin
    LastCaption:=lvFiles.ItemFocused.Caption;
    LastExt:=lvFiles.ItemFocused.SubItems[0];
  End
  Else Begin
    LastCaption:='';
    LastExt:='';
  End;

  If Column.Index=SortColumn Then
    SortAscending:=Not(SortAscending)
  Else
    SortAscending:=True;
```

```
  SortColumn:=Column.Index;
  Sort;
  ShowFiles;

  SelectLastItem;
end;

Procedure TfrFilePanel.SetPath(Path:String);
Var
  TmpStr:String;
Begin
  TmpStr:=ExcludeTrailingBackslash(Path);
  TmpStr:=ExtractFileName(TmpStr);

  If TmpStr='..' Then
  Begin
    LastCaption:=ExcludeTrailingBackslash(CurrentFullPath);

LastCaption:=ConstDirLeftBracket+ExtractFileName(LastCaption)+ConstDirRightBrack
et;
    LastExt:='';
  End;

  CurrentFullPath:=Path;

  Refresh;
End;

Procedure TfrFilePanel.ShowItem(Item:TFileRecord);
Var
  FormattedItem:TFileFormattedRecord;
  ListItem:TListItem;
Begin

    GetFormattedItem(Item, FormattedItem);
    ListItem:=lvFiles.Items.Add;
    With ListItem Do
    Begin
      ImageIndex:=GetItemImageIndex(Item);
      If Item.Checked Then
        StateIndex:=0
      Else
        StateIndex:=-1;
      If IsDirectory(Item) Then
        Caption:=ConstDirLeftBracket+FormattedItem.Name+ConstDirRightBracket
      Else
        Caption:=FormattedItem.Name;
      SubItems.Add(FormattedItem.Ext);
      SubItems.Add(FormattedItem.Size);
      SubItems.Add(FormattedItem.DateTime);
      SubItems.Add(FormattedItem.Attr);
    End;

End;

Procedure TfrFilePanel.ShowFiles;
Var
  i:Integer;
  Item:TFileRecord;
Begin
  lvFiles.Items.Clear;

  If NowRoot Then
    lvFiles.AllocBy:=AllFiles.ItemsCount
  Else
    lvFiles.AllocBy:=AllFiles.ItemsCount+1;

  If Not(NowRoot) Then ShowItem(DirUpItem);
```

```
  For i:=0 To AllFiles.ItemsCount-1 Do
  Begin
    AllFiles.GetItem(i, Item);

    ShowItem(Item);

  End;

End;

Procedure TfrFilePanel.GetItemByList(ListItem:TListItem; Var Item:TFileRecord);
Var
  i:Integer;
Begin
  i:=lvFiles.Items.IndexOf(ListItem);
  If Not(NowRoot) Then Dec(i);
  If i<0 Then
    Item:=DirUpItem
  Else
    AllFiles.GetItem(i, Item);
End;

Procedure TfrFilePanel.ShowInfo;
Var
  TotalBytes, TotalFree:Int64;
  TotalDirs, TotalFiles, CheckedDirs, CheckedFiles:Integer;
  TotalSize, CheckedSize:Int64;
  TmpStr:String;
Begin
  GetDiskSize(CurrentDrive, TotalBytes, TotalFree);
  lbCurrentPath.Caption:=CurrentFullPath;
  lbCurrentPath.Hint:=CurrentFullPath;
  TmpStr:=GetCompactSize(TotalFree)+' з '+GetCompactSize(TotalBytes)+' вільно';
  lbDriveInfo.Caption:=TmpStr;
  lbDriveInfo.Hint:=TmpStr;

  AllFiles.GetInfo(TotalDirs, TotalFiles, CheckedDirs, CheckedFiles, TotalSize,
CheckedSize);
  TmpStr:=' '+GetCompactSize(CheckedSize)+' / '+GetCompactSize(TotalSize)+' в '+
    GetFormattedSize(CheckedFiles, 1, True)+' / '+GetFormattedSize(TotalFiles,
1, True)+' файлі(ах) і '+
    GetFormattedSize(CheckedDirs, 1, True)+' / '+GetFormattedSize(TotalDirs, 1,
True)+' папці(ках)';
  pnFilesInfo.Caption:=TmpStr;
  pnFilesInfo.Hint:=TmpStr;
End;

Procedure TfrFilePanel.SetColumnsSize(ColumnsSize:TColumnsSize);
Var
  i:Integer;
Begin
  For i:=0 To lvFiles.Columns.Count-1 Do
    lvFiles.Columns.Items[i].Width:=ColumnsSize[i];
End;

Procedure TfrFilePanel.GetColumnsSize(Var ColumnsSize:TColumnsSize);
Var
  i:Integer;
Begin
  For i:=0 To lvFiles.Columns.Count-1 Do
    ColumnsSize[i]:=lvFiles.Columns.Items[i].Width;
End;

procedure TfrFilePanel.dcbxDriveChange(Sender: TObject);
begin
  If Not(Inited) Then Exit;
  SetDrive(dcbxDrive.Drive);
  Refresh;
  Activate;
```

```
end;

procedure TfrFilePanel.lvFilesKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
Var
  TmpStr:String;
  TmpBool:Boolean;
begin
  Case Key Of
    VK_Return:Begin
        lvFilesDblClick(Self);
      End;
    VK_Left:Begin
        lvFiles.ItemFocused:=lvFiles.Items.Item[0];
        lvFiles.Selected:=lvFiles.Items.Item[0];
      End;
    VK_Right:Begin
        lvFiles.ItemFocused:=lvFiles.Items.Item[lvFiles.Items.Count-1];
        lvFiles.Selected:=lvFiles.Items.Item[lvFiles.Items.Count-1];
      End;
    VK_Back:Begin
        If ssCtrl In Shift Then
          btDirRootClick(Self)
        Else
          btDirUpClick(Self);
      End;
    VK_Delete:Begin
        TryDelete;
      End;
    VK_F8:Begin
        TryDelete;
      End;
    VK_F2:Begin
        lvFilesEditing(Self, lvFiles.ItemFocused, TmpBool);
      End;
    VK_F3:Begin
        EditFile;
      End;
    VK_F4:Begin
        If ssShift In Shift Then
        Begin
          TmpStr:=ConstLastRequest;
          If Not(GetNameQuery('Редагувати файл:', TmpStr)) Then Exit;
          ExecuteOneFile(CurrentFullPath, ConstNotepadFile, TmpStr);
          Exit;
        End;
        EditFile;
      End;
    VK_F5:Begin
        TryCopyFile;
      End;
    VK_F6:Begin
        TryMoveFile;
      End;
    VK_F7:Begin
        CreateFolder;
      End;
  End;
end;

Function TfrFilePanel.ChangeCheck(ListItem:TListItem):Boolean;
Var
  Item:TFileRecord;
  ID:Integer;
Begin
  Result:=False;
  GetItemByList(ListItem, Item);
  If Item.Name=ConstDirUp Then Exit;
  ID:=Item.ID;
```

```
  AllFiles.Items[ID].Checked:=Not(AllFiles.Items[ID].Checked);
  If AllFiles.Items[ID].Checked Then ListItem.StateIndex:=ConstCheckedImageIndex
Else ListItem.StateIndex:=ConstUnCheckedImageIndex;
  Result:=AllFiles.Items[ID].Checked;
  ShowInfo;
End;

Procedure TfrFilePanel.SelectLastItem;
Var
  ListItem:TListItem;
  StartIndex:Integer;
Begin
  StartIndex:=0;
  Repeat
    ListItem:=lvFiles.FindCaption(StartIndex, LastCaption, False, False, False);
    If ListItem=nil Then
    Begin
      lvFiles.ItemFocused:=lvFiles.Items.Item[0];
      lvFiles.Selected:=lvFiles.Items.Item[0];
      LastCaption:='';
      LastExt:='';
      Exit;
    End;
    If ListItem.SubItems[0]=LastExt Then
    Begin
      lvFiles.ItemFocused:=ListItem;
      lvFiles.Selected:=ListItem;
      LastCaption:='';
      LastExt:='';
      Exit;
    End;
    StartIndex:=ListItem.Index;
  Until False;
End;

Function TfrFilePanel.TryRename(ListItem:TListItem; NewName:String):Boolean;
Var
  Item:TFileRecord;
  OldName, Name, Ext:String;
  Tmp:Integer;
  TmpStr:String;
Begin
  Result:=False;
  Name:=ExtractFileName(NewName);
  Ext:=ExtractFileExt(NewName);
//  NewName:=CurrentFullPath+Name;
  If Ext<>'' Then
  Begin
    Delete(Name, Length(Name)-Length(Ext)+1, Length(Ext));
    Delete(Ext, 1, 1);
  End;
  If Name='' Then Exit;

  GetItemByList(ListItem, Item);
  OldName:=CurrentFullPath+GetFullName(Item);
  Tmp:=RenameOneFile(OldName, NewName);
  If Tmp<0 Then
  Begin
    TmpStr:=GetFileError(Tmp)+#0;
    Application.MessageBox(@TmpStr[1], 'Помилка!', MB_ICONERROR Or MB_OK);
    Exit;
  End;

  If IsDirectory(Item) Then
    LastCaption:=ConstDirLeftBracket+Name+ConstDirRightBracket
  Else
    LastCaption:=Name;
  LastExt:=Ext;
```

```
  Result:=True;
End;

Function TfrFilePanel.TryOneDelete(Item:TFileRecord):Integer;
Begin
  If Item.Name=ConstDirUp Then
  Begin
    Result:=F_ER_ERROR;
    Exit;
  End;
  If IsDirectory(Item) Then
    Result:=DeleteOneDir(CurrentFullPath+Item.Name)
  Else
    Result:=DeleteOneFile(CurrentFullPath+GetFullName(Item));
End;

Function TfrFilePanel.TryDelete:Boolean;
Var
  ListItem:TListItem;
  Item:TFileRecord;
  Tmp:Integer;
  TmpStr:String;
Begin
  Result:=False;
  ListItem:=lvFiles.ItemFocused;
  GetItemByList(ListItem, Item);
  If Item.Name=ConstDirUp Then Exit;

  TmpStr:='Ви дійсно хочете видалити "'+GetFullName(Item)+'"?'+#0;
  If Application.MessageBox(@TmpStr[1], 'Попередження', MB_ICONQUESTION Or
MB_YESNO)=IDNO Then Exit;

  Tmp:=TryOneDelete(Item);
  If Tmp=F_ER_SUCCESS Then
  Begin
    Result:=True;
    Refresh;
    Exit;
  End;
  TmpStr:=GetFileError(Tmp)+#0;
  Application.MessageBox(@TmpStr[1], 'Помилка!', MB_ICONERROR Or MB_OK);
  Result:=False;
End;

Procedure TfrFilePanel.TryCopyFile;
Var
  Item:TFileRecord;
  FileName, TmpStr:String;
  Tmp:Integer;
Begin
  GetItemByList(lvFiles.ItemFocused, Item);
  If IsDirectory(Item) Then
  Begin
    Application.MessageBox('Неможливо копіювати папку', 'Помилка!', MB_ICONERROR
Or MB_OK);
    Exit;
  End;
  FileName:=UseCopyToDir+GetFullName(Item);
  If Not(GetNameQuery('Скопіювати файл "'+GetFullName(Item)+'" в:', FileName))
Then Exit;
  If ExtractFileName(FileName)=FileName Then
    FileName:=CurrentFullPath+FileName;
  ShowAnyMessage('Копіювання...', 'Копіюється файл
"'+CurrentFullPath+GetFullName(Item)+'" в "'+FileName+'"');
  Tmp:=CopyOneFile(CurrentFullPath+GetFullName(Item), FileName, True);
  HideAnyMessage;
  If Tmp=F_ER_SUCCESS Then
  Begin
    MessageBeep(0);
```

```
    LastCaption:=Item.Name;
    LastExt:=Item.Ext;
    Refresh;
    Exit;
  End;
  TmpStr:=GetFileError(Tmp)+#0;
  Application.MessageBox(@TmpStr[1], 'Помилка!', MB_ICONERROR Or MB_OK);
End;

Procedure TfrFilePanel.TryMoveFile;
Var
  Item:TFileRecord;
  NewName:String;
Begin
  GetItemByList(lvFiles.ItemFocused, Item);
  If Item.Name=DirUpItem.Name Then Exit;
  NewName:=UseCopyToDir+GetFullName(Item);

  If GetNameQuery('Перемістити "'+GetFullName(Item)+'" в:', NewName) Then
  Begin
    If TryRename(lvFiles.ItemFocused, NewName) Then Refresh;
  End;
End;

Procedure TfrFilePanel.EditFile;
Var
  ListItem:TListItem;
  Item:TFileRecord;
Begin
  ListItem:=lvFiles.ItemFocused;
  GetItemByList(ListItem, Item);
  If IsDirectory(Item) Then Exit;

  ExecuteOneFile(CurrentFullPath, ConstNotepadFile, GetFullName(Item));
End;

Procedure TfrFilePanel.CreateFolder;
Var
  FolderName:String;
  Tmp:Integer;
  TmpStr:String;
Begin
  FolderName:=ConstLastRequest;
  If Not(GetNameQuery('Створити папку:', FolderName)) Then Exit;
  If FolderName='' Then Exit;
  Tmp:=CreateOneFolder(CurrentFullPath+FolderName);
  If Tmp=F_ER_SUCCESS  Then
  Begin
    LastCaption:=ConstDirLeftBracket+FolderName+ConstDirRightBracket;
    LastExt:='';
    Refresh;
    Exit;
  End;
  TmpStr:=GetFileError(Tmp)+#0;
  Application.MessageBox(@TmpStr[1], 'Помилка!', MB_ICONERROR Or MB_OK);
End;

Procedure TfrFilePanel.SetDrive(Drive:Char);
Var
  Dir:String;
Begin
  Repeat
    Drive:=UpCase(Drive);
   {$ I-}
    GetDir(Ord(Drive)-64, Dir);
    If Drive=Dir[1] Then
      ChDir(Dir)
    Else Begin
      Dir:=Drive+':\';
```

```
      ChDir(Dir);
    End;
    If IOResult<>0 Then
    Begin
      Dir:=Drive+':\';
      ChDir(Dir);
    End;
   {$I+}
    If IOResult=0 Then
    Begin
      dcbxDrive.Drive:=Drive;
      CurrentDrive:=Drive;
      CurrentFullPath:=Dir;
      Exit;
    End;
    ChooseNewDrive(Drive);
  Until False;
End;

Procedure TfrFilePanel.CheckCurrentPath;
Var
  Dir:String;
Begin
 {$ I-}
  ChDir(CurrentFullPath);
 {$I+}
  If IOResult<>0 Then
  Begin
    SetDrive(ExtractFileDrive(CurrentFullPath)[1]);
  End;
  Dir:=CurrentFullPath+#0;
  SetCurrentDirectory(@Dir[1]);
End;


procedure TfrFilePanel.lvFilesDblClick(Sender: TObject);
Var
  ListItem:TListItem;
  Item:TFileRecord;
  ErrorCode:Integer;
  ErrorString:String;
begin
  ListItem:=lvFiles.ItemFocused;
  If ListItem=nil Then Exit;

  GetItemByList(ListItem, Item);

  If IsDirectory(Item) Then
  Begin
    SetPath(CurrentFullPath+Item.Name);
    Exit;
  End;

  If UpperCase(Item.Ext)=ConstArchiveExt Then
  Begin
    fmDAR.DeCompress;
    Refresh;
    Exit;
  End;

  fmDAR.Compress;
  Refresh;
end;

procedure TfrFilePanel.btDirUpClick(Sender: TObject);
begin
  Activate;
  SetPath(CurrentFullPath+'..');
end;
```

```
procedure TfrFilePanel.btDirRootClick(Sender: TObject);
begin
  Activate;
  SetPath(IncludeTrailingBackslash(ExtractFileDrive(CurrentFullPath)));
end;

procedure TfrFilePanel.lvFilesColumnRightClick(Sender: TObject;
  Column: TListColumn; Point: TPoint);
begin
  Activate;
  Column.Width:=-1;
end;

procedure TfrFilePanel.lvFilesEditing(Sender: TObject; Item: TListItem;
  var AllowEdit: Boolean);
Var
  FileItem:TFileRecord;
  NewName:String;
begin
  AllowEdit:=False;
  GetItemByList(Item, FileItem);
  If FileItem.Name=DirUpItem.Name Then Exit;

  NewName:=GetFullName(FileItem);
  If GetNameQuery('Перейменувати "'+NewName+'" в:', NewName) Then
  Begin
    If TryRename(Item, NewName) Then Refresh;
  End;

  AllowEdit:=False;
end;

procedure TfrFilePanel.lvFilesChange(Sender: TObject; Item: TListItem;
  Change: TItemChange);
Begin
  MakeOutLabels;
end;

procedure TfrFilePanel.lvFilesEnter(Sender: TObject);
begin
  Activate;
end;

procedure TfrFilePanel.lbCurrentPathClick(Sender: TObject);
begin
  Activate;
end;

procedure TfrFilePanel.bbRefreshClick(Sender: TObject);
begin
  Refresh;
  Activate;
end;

end.
```

Файл Pr_backup_and_arch.dpr – головний файл проекту

```
program Pr_backup_and_arch;

uses
  Forms,
  DeCompressorU in 'DeCompressorU.pas',//алгоритми архівування
  DFileStreamU in 'DFileStreamU.pas', //алгоритми розархівування
  DictionaryU in 'DictionaryU.pas',//словник
  Backup_and_archU.pas in 'Backup_and_arch.pas',//форма архівування
  fmExtractDirU in 'fmExtractDirU.pas' {fmExtractDir},//форма добування з архіву
  Main in 'Main.pas' {fmPR_BACKUP_AND_ARCH}, //головне вікно
  About in 'About.pas' {fmAbout},//файл інформації про розробника
  frFilePanelU in 'frFilePanelU.pas' {frFilePanel: TFrame},//панелі
  FilesExU in 'FilesExU.pas',
  FilesU in 'FilesU.pas',
  fmAnyMessageU in 'fmAnyMessageU.pas' {fmAnyMessage},//форма повідомлень
  fmErrorDriveU in 'fmErrorDriveU.pas' {fmErrorDrive},//форма помилок
  fmNameQueryU in 'fmNameQueryU.pas' {fmNameQuery},//форма запитів
  StrConsts in 'StrConsts.pas',// константи
  fmProcessU in 'fmProcessU.pas' {fmProcess};//форма процесу роботи

{$R *.RES}
//основна программа
begin
  Application.Initialize;
  Application.Title := 'Резервне копіювання та архіватор';
  Application.CreateForm(TfmNameQuery, fmNameQuery); //створення вікна запитів
  Application.CreateForm(TfmExtractDir, fmExtractDir); //створення вікна папки
куди розахівовувати
  Application.CreateForm(TfmProcess, fmProcess); //створення вікна процесу роботи
  Application.Run; //створення вікна початку роботи
  Application.CreateForm(TfmPR_BACKUP_AND_ARCH, fmPR_BACKUP_AND_ARCH);//створення
вікна архівування
  Application.CreateForm(TfmAbout, fmAbout); //створення вікна інформації про
розробника
  Application.CreateForm(TfmAnyMessage, fmAnyMessage); //створення вікна
виведення повідомлень
  Application.CreateForm(TfmErrorDrive, fmErrorDrive); //створення вікна
виведення помилок
end.
```

Файл Dictionary.pas - створення словника

```
unit Dictionary;

interface

Const
  ConstBitsForDic=12;
  ConstWordsCount=1 Sh ConstBitsForDic;

Type
  TSingleWord=ANSIString;

  TDictionary=Object
    Words:Array[0.. ConstWordsCount-1] Of TSingleWord;
    Count:Integer;

    Constructor Init;
    Destructor Done;

    Procedure Clear;
    Function Add(SingleWord:TSingleWord):Integer;
    Function Find(SingleWord:TSingleWord):Integer;

    Function Compare(Word1, Word2:TSingleWord):Boolean;
  End;

Function GetWordLength(SingleWord:TSingleWord):Integer;
Function AddCharToWord(Var SingleWord:TSingleWord; Ch:Byte):Integer;
Procedure ClearWord(Var SingleWord:TSingleWord);

implementation

Constructor TDictionary.Init;
Begin
  Clear;
End;

Destructor TDictionary.Done;
Begin
  Clear;
End;

Procedure TDictionary.Clear;
Var
  i:Integer;
Begin
  For i:=0 To ConstWordsCount-1 Do
  Begin
    Words[i]:='';
  End;
  Count:=0;
End;

Function TDictionary.Add(SingleWord:TSingleWord):Integer;
Begin
  Result:=-1;
  If Count>=ConstWordsCount Then Exit;
  If SingleWord='' Then Exit;
  Result:=Find(SingleWord);
  If Result>=0 Then Exit;
  Words[Count]:=SingleWord;
  Result:=Count;
  Inc(Count);
End;

Function TDictionary.Find(SingleWord:TSingleWord):Integer;
```

```
Var
  i:Integer;
Begin
  Result:=-1;
  If Count<=0 Then Exit;
  If SingleWord='' Then Exit;
  For i:=0 To Count-1 Do
  Begin
    If SingleWord=Words[i] Then
    Begin
      Result:=i;
      Exit;
    End;
  End;
End;

Function TDictionary.Compare(Word1, Word2:TSingleWord):Boolean;
Begin
  Result:=(Word1=Word2);
End;

Function GetWordLength(SingleWord:TSingleWord):Integer;
Begin
  Result:=Length(SingleWord);
End;

Function AddCharToWord(Var SingleWord:TSingleWord; Ch:Byte):Integer;
Begin
  Result:=0;
  SingleWord:=SingleWord+Char(Ch);
End;

Procedure ClearWord(Var SingleWord:TSingleWord);
Begin
  SingleWord:='';
End;

end.
```

Файл Compressor.pas - кодування

```
unit Compressor;

interface

Function CompressFile(SrcFile, DestFile:String):Integer;
Function DeCompressFile(SrcFile, DestDir:String):Integer;

implementation

Uses
  SysUtils, FileStreams, Dictionaries;

Var
  ReadStream:TFileReadStream;
  WriteStream:TFileWriteStream;
  Dic:TDictionary;
  NowBitsForDic:Byte;


Procedure WriteFileName(FileName:String; FSize:Integer);
Var
  i:Word;
Begin
  FileName:=ExtractFileName(FileName);
  WriteStream.Write(Length(FileName), 16);
  For i:=1 To Length(FileName) Do
    WriteStream.Write(Ord(FileName[i]), 8);
  WriteStream.Write(0, 8);

  i:=FSize And $FFFF;
  WriteStream.Write(i, 16);
  i:=FSize Sh 16;
  WriteStream.Write(i, 16);

  WriteStream.Write(0, 16);
End;

Procedure WriteSingleByte(Ch:Byte);
Begin
  WriteStream.WriteBit(0);
  WriteStream.Write(Ch, 8);
End;

Procedure WriteDoubleByte(Ch:Word; Cnt:Byte);
Begin
  WriteStream.WriteBit(1);
  WriteStream.Write(Ch, Cnt);
End;

Procedure WriteByDic(N:Integer);
Var
  Data:Word;
Begin
  If N<-256 Then Exit;
  Data:=Abs(N);
  If N<0 Then
  Begin
    Dec(Data);
    WriteSingleByte(Data);
  End
  Else Begin
    WriteDoubleByte(Data, NowBitsForDic);
  End;
End;
```

```
Function FindInDic(SingleWord:TSingleWord):Integer;
Begin
  If GetWordLength(SingleWord)=1 Then
  Begin
    Result:=-(Integer(SingleWord[1])+1);
    Exit;
  End;
  Result:=Dic.Find(SingleWord);
  If Result<0 Then Result:=-1024;
End;

Procedure ProcessNewByte(Var InpWord:TSingleWord; NewChar:Byte; Var
Out:Integer);
Var
  New:TSingleWord;
Begin
  New:=InpWord;
 {Якщо рядок уже занадто довгий}
  If AddCharToWord(New, NewChar)<0 Then
  Begin
    Out:=FindInDic(New);  //запишемо цей рядок
    ClearWord(InpWord);
    AddCharToWord(InpWord, NewChar);  //Змінимо вхідне слово
    Exit;
  End;

  Out:=FindInDic(New);
  If Out>=-256 Then  //якщо є в словнику
  Begin  //то нічого не робимо
    Out:=-1024;
    InpWord:=New;  //Змінимо вхідне слово
  End
  Else Begin //Якщо немає в словнику, то...
    Dic.Add(New);  //додамо новий рядок у словник
    Out:=FindInDic(InpWord);  //а на вихід пошлемо попередній рядок
    ClearWord(InpWord);
    AddCharToWord(InpWord, NewChar);  //Змінимо вхідне слово
  End;
End;

Procedure ProcessCompression;
Var
  SrcWord:TSingleWord;
  NewByte:Byte;
  Out:Integer;
Begin
  NowBitsForDic:=ConstBitsForDic;

  If ReadStream.FileRead Then Exit;
  ClearWord(SrcWord);
  NewByte:=ReadStream.Read(8);
  AddCharToWord(SrcWord, NewByte);

  While Not(ReadStream.FileRead) Do
  Begin
    NewByte:=ReadStream.Read(8);
    ProcessNewByte(SrcWord, NewByte, Out);
    WriteByDic(Out);
  End;
  Out:=FindInDic(SrcWord);
  WriteByDic(Out);

  WriteStream.Write(0, 8-WriteStream.AdditionalBits+8);
End;

Function CompressFile(SrcFile, DestFile:String):Integer;
Var
  FSize:Integer;
```

```
Begin
  Result:=0;

  Dic.Clear;
  ReadStream.OpenStream(SrcFile);
  WriteStream.OpenStream(DestFile);

  FSize:=ReadStream.Size;

  WriteFileName(SrcFile, FSize);

  ProcessCompression;

  ReadStream.CloseStream;
  WriteStream.CloseStream;
End;

Function ReadFileName(Var FSize:Integer):String;
Var
  Len, i:Word;
  Tmp:Integer;
Begin
  Result:='';
  FSize:=0;

  Len:=ReadStream.Read(16);
  For i:=1 To Len Do
    Result:=Result+Char(ReadStream.Read(8));
  ReadStream.Read(8);

  FSize:=ReadStream.Read(16);

  Tmp:=ReadStream.Read(16);
  Tmp:=Tmp Sh 16;
  FSize:=Tmp+FSize;

  ReadStream.Read(16);
End;

Procedure SaveSingleWord(SingleWord:TSingleWord);
Var
  i:Word;
  Tmp:Byte;
Begin
  If GetWordLength(SingleWord)=0 Then Exit;
  For i:=1 To GetWordLength(SingleWord) Do
  Begin
    Tmp:=Byte(SingleWord[i]);
    WriteStream.Write(Tmp, 8);
  End;
End;

Procedure WriteByDicOrig(N:Integer);
Var
  Data:Word;
  SrcWord:TSingleWord;
Begin
  If N<-256 Then Exit;
  Data:=Abs(N);
  If N<0 Then
  Begin
    Dec(Data);
    WriteStream.Write(Data, 8);
  End
  Else Begin
    SrcWord:=Dic.Words[Data];
    SaveSingleWord(SrcWord);
  End;
End;
```

```
Procedure ProcessDeCompression(FSize:Integer);
Var
  SrcWord, SaveWord:TSingleWord;
  NewByte, BitFlag:Byte;
  Out, Temp:Integer;
  OrdinaryWord:Boolean;
  i:Integer;
Begin
  NowBitsForDic:=ConstBitsForDic;
  OrdinaryWord:=True;

  If FSize=0 Then Exit;

  ClearWord(SrcWord);

  While ((WriteStream.BytesWrote<FSize) And Not(ReadStream.FileRead)) Do
  Begin
    BitFlag:=ReadStream.ReadBit;
    If BitFlag=0 Then  //якщо це просто байт
    Begin
      NewByte:=ReadStream.Read(8);
      ProcessNewByte(SrcWord, NewByte, Out);
      WriteStream.Write(NewByte, 8);
    End
    Else Begin  //якщо це словникове слово
      Out:=ReadStream.Read(NowBitsForDic);
      If Out>=Dic.Count Then  //якщо потрібного слова немає в словнику
      Begin
        ProcessNewByte(SrcWord, Byte(SrcWord[1]), Temp);  //створюємо його з
випередженням
        OrdinaryWord:=False;
      End
      Else OrdinaryWord:=True;

      SaveWord:=Dic.Words[Out];
      SaveSingleWord(SaveWord);

      If OrdinaryWord Then  //якщо звичайне слово, то стандартний перебір
      Begin
        For i:=1 To GetWordLength(SaveWord) Do
        Begin
          NewByte:=Byte(SaveWord[i]);
          ProcessNewByte(SrcWord, NewByte, Out);
        End;
      End
      Else Begin  //інакше скорочений перебір
        For i:=2 To GetWordLength(SaveWord) Do
        Begin
          NewByte:=Byte(SaveWord[i]);
          ProcessNewByte(SrcWord, NewByte, Out);
        End;
      End;
    End;
  End;
End;

Function DeCompressFile(SrcFile, DestDir:String):Integer;
Var
  FSize:Integer;
  DestFile:String;
Begin
  Result:=0;

  Dic.Clear;
  ReadStream.OpenStream(SrcFile);

  DestFile:=DestDir+ReadFileName(FSize);
  WriteStream.OpenStream(DestFile);
```

```
  ProcessDeCompression(FSize);

  ReadStream.CloseStream;
  WriteStream.CloseStream;
End;


initialization
  Dic.Init;
  ReadStream:=TFileReadStream.Init;
  WriteStream:=TFileWriteStream.Init;

finalization
  Dic.Done;
  ReadStream.Done;
  WriteStream.Done;

end.
```

Файл DeCompressor.pas - декодування

```
unit DeCompressor;

interface

Const
  ConstFirstSignature:String[64]=
    'DAr compver0.9.0.0 do NOT change this data! blah-blah-blah!';

Type
  TCompressProcessProc=Function(OrigSize, OrigPos, Comp:Integer):Boolean;

Var
  CompressProcessProc:TCompressProcessProc;

Function CompressFile(SrcFile, DestFile:String):Integer;
Function DeCompressFile(SrcFile, DestDir:String):Integer;

implementation

Uses
  SysUtils, DFileStream, Dictionary, Main;

Var
  ReadStream:TFileReadStream;
  WriteStream:TFileWriteStream;
  Dic:TDictionary;
  NowBitsForDic:Byte;


Function CPP(OrigSize, OrigPos, Comp:Integer):Boolean;
Begin
//емуляція індикації процесу
  Result:=True;
End;

Procedure WriteSignature;
Var
  i:Byte;
Begin
  For i:=1 To Length(ConstFirstSignature) Do
    WriteStream.Write(Ord(ConstFirstSignature[i]), 8);
End;

Procedure WriteFileName(FileName:String; FSize:Integer);
Var
  i:Word;
Begin
  FileName:=ExtractFileName(FileName);
  WriteStream.Write(Length(FileName), 16);
  For i:=1 To Length(FileName) Do
    WriteStream.Write(Ord(FileName[i]), 8);
  WriteStream.Write(0, 8);

  i:=FSize And $FFFF;
  WriteStream.Write(i, 16);
  i:=FSize Sh 16;
  WriteStream.Write(i, 16);

  WriteStream.Write(0, 16);
End;

Procedure WriteSingleByte(Ch:Byte);
Begin
  WriteStream.WriteBit(0);
```

```
  WriteStream.Write(Ch, 8);
End;

Procedure WriteDoubleByte(Ch:Word; Cnt:Byte);
Begin
  WriteStream.WriteBit(1);
  WriteStream.Write(Ch, Cnt);
End;

Procedure WriteByDic(N:Integer);
Var
  Data:Word;
Begin
  If N<-256 Then Exit;
  Data:=Abs(N);
  If N<0 Then
  Begin
    Dec(Data);
    WriteSingleByte(Data);
  End
  Else Begin
    WriteDoubleByte(Data, NowBitsForDic);
  End;
End;

Function FindInDic(SingleWord:TSingleWord):Integer;
Begin
  If GetWordLength(SingleWord)=1 Then
  Begin
    Result:=-(Integer(SingleWord[1])+1);
    Exit;
  End;
  Result:=Dic.Find(SingleWord);
  If Result<0 Then Result:=-1024;
End;

Procedure ProcessNewByte(Var InpWord:TSingleWord; NewChar:Byte; Var
Out:Integer);
Var
  New:TSingleWord;
Begin
  New:=InpWord;
 {Якщо рядок уже занадто довгий}
  If AddCharToWord(New, NewChar)<0 Then
  Begin
    Out:=FindInDic(New);  //запишемо цей рядок
    ClearWord(InpWord);
    AddCharToWord(InpWord, NewChar);  //Змінимо вхідне слово
    Exit;
  End;

  Out:=FindInDic(New);
  If Out>=-256 Then  //якщо є в словнику
  Begin  //то нічого не робимо
    Out:=-1024;
    InpWord:=New;  //Змінимо вхідне слово
  End
  Else Begin //Якщо немає в словнику, то...
    Dic.Add(New);  //додамо новий рядок у словник
    Out:=FindInDic(InpWord);  //а на вихід пошлемо попередній рядок
    ClearWord(InpWord);
    AddCharToWord(InpWord, NewChar);  //Змінимо вхідне слово
  End;
End;

Procedure ProcessCompression;
Var
  SrcWord:TSingleWord;
  NewByte:Byte;
```

```
  Out:Integer;
Begin
  NowBitsForDic:=ConstBitsForDic;

  If ReadStream.FileRead Then Exit;
  ClearWord(SrcWord);
  NewByte:=ReadStream.Read(8);
  AddCharToWord(SrcWord, NewByte);

  While Not(ReadStream.FileRead) Do
  Begin
    NewByte:=ReadStream.Read(8);
    ProcessNewByte(SrcWord, NewByte, Out);
    WriteByDic(Out);
   //Індикація
    If Not(CompressProcessProc(ReadStream.Size, ReadStream.BytesRead,
WriteStream.BytesWrote)) Then
    Begin
      Exit;
    End;
  End;
  Out:=FindInDic(SrcWord);
  WriteByDic(Out);

  WriteStream.Write(0, 8-WriteStream.AdditionalBits+8);
End;

Function CompressFile(SrcFile, DestFile:String):Integer;
Var
  FSize:Integer;
Begin
  Result:=0;

  Dic.Clear;
  ReadStream.OpenStream(SrcFile);
  WriteStream.OpenStream(DestFile);

  FSize:=ReadStream.Size;

  WriteSignature;
  WriteFileName(SrcFile, FSize);

  ProcessCompression;

  ReadStream.CloseStream;
  WriteStream.CloseStream;
End;

Function ReadSignature:Boolean;
Var
  i:Byte;
Begin
  Result:=False;
  For i:=1 To 64 Do
    If Ord(ConstFirstSignature[i])<>ReadStream.Read(8) Then Exit;
  Result:=True;
End;

Function ReadFileName(Var FSize:Integer):String;
Var
  Len, i:Word;
  Tmp:Integer;
Begin
  Result:='';
  FSize:=0;

  Len:=ReadStream.Read(16);
  For i:=1 To Len Do
    Result:=Result+Char(ReadStream.Read(8));
```

```
  ReadStream.Read(8);

  FSize:=ReadStream.Read(16);

  Tmp:=ReadStream.Read(16);
  Tmp:=Tmp Sh 16;
  FSize:=Tmp+FSize;

  ReadStream.Read(16);
End;

Procedure SaveSingleWord(SingleWord:TSingleWord);
Var
  i:Word;
  Tmp:Byte;
Begin
  If GetWordLength(SingleWord)=0 Then Exit;
  For i:=1 To GetWordLength(SingleWord) Do
  Begin
    Tmp:=Byte(SingleWord[i]);
    WriteStream.Write(Tmp, 8);
  End;
End;

Procedure WriteByDicOrig(N:Integer);
Var
  Data:Word;
  SrcWord:TSingleWord;
Begin
  If N<-256 Then Exit;
  Data:=Abs(N);
  If N<0 Then
  Begin
    Dec(Data);
    WriteStream.Write(Data, 8);
  End
  Else Begin
    SrcWord:=Dic.Words[Data];
    SaveSingleWord(SrcWord);
  End;
End;

Procedure ProcessDeCompression(FSize:Integer);
Var
  SrcWord, SaveWord:TSingleWord;
  NewByte, BitFlag:Byte;
  Out, Temp:Integer;
  OrdinaryWord:Boolean;
  i:Integer;
Begin
  NowBitsForDic:=ConstBitsForDic;
  OrdinaryWord:=True;

  If FSize=0 Then Exit;

  ClearWord(SrcWord);

  While ((WriteStream.BytesWrote<FSize) And Not(ReadStream.FileRead)) Do
  Begin
    BitFlag:=ReadStream.ReadBit;
    If BitFlag=0 Then  //якщо це просто байт
    Begin
      NewByte:=ReadStream.Read(8);
      ProcessNewByte(SrcWord, NewByte, Out);
      WriteStream.Write(NewByte, 8);
    End
    Else Begin  //якщо це словникове слово
      Out:=ReadStream.Read(NowBitsForDic);
      If Out>=Dic.Count Then  //якщо потрібного слова немає в словнику
```

```
      Begin
        ProcessNewByte(SrcWord, Byte(SrcWord[1]), Temp);  //створюємо його з
випередженням
        OrdinaryWord:=False;
      End
      Else OrdinaryWord:=True;

      SaveWord:=Dic.Words[Out];
      SaveSingleWord(SaveWord);

      If OrdinaryWord Then  //якщо звичайне слово, то стандартний перебір
      Begin
        For i:=1 To GetWordLength(SaveWord) Do
        Begin
          NewByte:=Byte(SaveWord[i]);
          ProcessNewByte(SrcWord, NewByte, Out);
        End;
      End
      Else Begin  //інакше скорочений перебір
        For i:=2 To GetWordLength(SaveWord) Do
        Begin
          NewByte:=Byte(SaveWord[i]);
          ProcessNewByte(SrcWord, NewByte, Out);
        End;
      End;
    End;
   //Індикація
    If Not(CompressProcessProc(FSize, WriteStream.BytesWrote,
ReadStream.BytesRead)) Then
    Begin
      Exit;
    End;
  End;
End;

Function DeCompressFile(SrcFile, DestDir:String):Integer;
Var
  FSize:Integer;
  DestFile:String;
Begin
  Result:=0;

  Dic.Clear;
  ReadStream.OpenStream(SrcFile);

  If Not(ReadSignature) Then
  Begin
    Result:=-1;
    Exit;
  End;
  DestFile:=DestDir+ReadFileName(FSize);
  WriteStream.OpenStream(DestFile);

  ProcessDeCompression(FSize);

  ReadStream.CloseStream;
  WriteStream.CloseStream;
End;

initialization
  Dic.Init;
  ReadStream:=TFileReadStream.Init;
  WriteStream:=TFileWriteStream.Init;
  CompressProcessProc:=CPP;
finalization
  Dic.Done;
  ReadStream.Done;
  WriteStream.Done;
end.
```

Файл fmProcess.pas - Візуалізація процесу архівування/розархівування

```
unit fmProcess;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, Gauges, ExtCtrls;

type
  TfmProcess = class(TForm)
    ggProcess: TGauge;
    ggRatio: TGauge;
    bbCabcel: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    lbProcessInfo: TLabel;
    procedure bbCabcelClick(Sender: TObject);
  private
    { Private declarations }
    Continue:Boolean;
  public
    { Public declarations }
    Function ShowProcess(OrigSize, OrigPos, Comp:Integer):Boolean;
    Procedure Compress(OldFileName, NewFileName:String);
    Procedure DeCompress(OldFileName, NewFileName:String);
  end;

var
  fmProcess: TfmProcess;

implementation

uses DeCompressor;

{$R *.DFM}

Procedure TfmProcess.Compress(OldFileName, NewFileName:String);
Begin
  lbProcessInfo.Caption:='Архівування файлу "'+OldFileName+'"';
  Show;

  Continue:=True;
  CompressFile(OldFileName, NewFileName);
  Sleep(1000);
  Hide;
End;

Procedure TfmProcess.DeCompress(OldFileName, NewFileName:String);
Begin
  lbProcessInfo.Caption:='Розархівування файлу "'+OldFileName+'"';
  Show;

  Continue:=True;
  DeCompressFile(OldFileName, NewFileName);
  Sleep(1000);
  Hide;
End;

Function TfmProcess.ShowProcess(OrigSize, OrigPos, Comp:Integer):Boolean;
Begin
  ggProcess.MaxValue:=OrigSize;
  ggProcess.Progress:=OrigPos;
```

```
//  ggRatio.MaxValue:=OrigSize;
  ggRatio.MaxValue:=OrigPos;
  ggRatio.Progress:=Comp;

  Result:=Continue;
  Application.ProcessMessages;
End;

procedure TfmProcess.bbCabcelClick(Sender: TObject);
begin
  Continue:=False;
  Application.ProcessMessages;

  Application.MessageBox('Процес перервано користувачем', 'Відміна',
MB_ICONINFORMATION Or MB_OK);
  Hide;
end;

end.
```

Файл Backup_and_arch.pas - реалізація роботи архіватора

```
unit Backup_and_arch;

{$H+}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids, XPMan, Math;

type
  TForm1 = class(TForm)
    Button1: TButton;
    OpenDialog1: TOpenDialog;
    SG1: TStringGrid;
    Button2: TButton;
    StringGrid1: TStringGrid;
    Button3: TButton;
    SaveDialog1: TSaveDialog;
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

TStr = string[20];
TKey = string[1];
TStack = array[0..65000] of TKey;

TPElSp = ^TElSp;
TElSp = record
  key: byte;
  cnt: cardinal;
  next: TPElSp;
  end;

TPArNode = ^TArNode;
TArNode = record
  value: real;
  kod: TKey;
  left: TPArNode;
  right: TPArNode;
  parent: TPArNode;
  symb: byte;
  end;

TTree = record
  key: byte;
  cnt: cardinal;
  huff: TStr;
  n0: extended;
  deep: boolean;
  st_tree: TPArNode;
  end;

TArTree = array of TTree;

TFileRec = record
  key: byte;
```

```
  kod: byte;
  size_kod: byte;
  end;

TGetFile = record
  key: byte;
  kod: string[16];
  size_kod: byte;
  end;

TArFRec = array of TGetFile;

var
  stack: TStack;
  PBS: integer;
  Form1: TForm1;
  head_lsp: TPElSp;
  f, fo: file;
  size, tmpi, cnti: byte;
  a, atmp: TArTree;
  n: int64;
  head_tree: TPArNode;
  okk: boolean;
implementation

{$R *.dfm}

procedure Push(key: TKey);
begin
 stack[PBS]:= key;
 inc(PBS);
end;

procedure Pop(var key: TKey);
begin
 dec(PBS);
 key:= stack[PBS];
end;

procedure AddEl(var start: TPElSp; PNew: TPElSp);
var
 WP: TPElSp;
begin
 PNew^.next:= nil;
 if start = nil then start:= PNew
 else
  begin
  WP:= start;
  while WP^.next <> nil do
   WP:= WP^.next;
  WP^.next:= PNew;
end;
end;

function FindEl(start: TPElSp; key: byte; var FindPoint: TPElSp): boolean;
begin
 if start = nil then
 begin
  result:= false;
  exit;
 end;
 result:= false;
 FindPoint:= start;
 while (FindPoint <> nil) and (FindPoint^.key <> key) do
  begin
  findpoint:= findpoint^.next;
  end;
  if (findpoint <> nil) then result:= true;
end;
```

```
function FindElMax(start: TPElSp; var FindPoint: TPElSp): boolean;
var
 PrevPoint: TPElSp;
begin
 if start = nil then
 begin
  result:= false;
  exit;
 end;
 FindPoint:= start;
 PrevPoint:= start^.next;
 while (PrevPoint <> nil) do
  begin
  if PrevPoint^.cnt > FindPoint^.cnt then
   FindPoint:= PrevPoint;
  Prevpoint:= PrevPoint^.next;
  end;
  result:= true;
end;

procedure DelEl(var start: TPElSp; key: byte);
var
PrevPoint, WPoint: TPElSp;
begin
if start = nil then exit;
 prevpoint:= nil;
 wpoint:= start;
 while (wpoint <> nil) and (wpoint^.key <> key) do
 begin
  prevpoint:= wpoint;
  wpoint:= wpoint^.next;
 end;
 if (wpoint = nil) or (wpoint^.key > ord(key)) then
 exit;
 if prevpoint = nil then
  start:= start^.next
 else
  prevpoint^.next:= wpoint^.next;
  Dispose(Wpoint);
end;

procedure DelSp(var head: TPElSp);
var
 w: TPElSp;
begin
  if head = nil then exit;
  while (head <> nil) do
  begin
  w:= head;
  head:= head^.next;
  Dispose(w);
  end;
end;

function FindMin(a: TArTree; size: integer): integer;
var
 z, i: integer; min: extended; ok: boolean;
begin
 ok:= false;
 for z:= size-1 downto 0 do
 begin
  if a[z].deep <> true then
  begin
  min:= a[z].n0;
  result:= z;
  ok:= true;
  i:= z-1;
  end;
```

```
  if ok = true then
   break;
 end;
  for z:= i downto 0 do
   if (a[z].n0 < min) and (a[z].deep <> true) then
   begin
    min:= a[z].n0;
    result:= z;
   end;
end;

function StrBToInt(bin: string): byte;
var
i: integer; v: byte;
begin
  v:= 0;
  result:= 0;
  for i:=length(bin) downto 1 do
  begin
    result:= result + StrToInt(bin[i]) * StrToInt(FloatToStr(exp(ln(2)*v)));
    inc(v);
  end;
end;

function IntToStr(in_int: byte): string;
var
 k: TKey;
 t1, t2: byte;
begin
  PBS:= 0;
  t1:= in_int;
  t2:= in_int;
  result:= '';
  while (t2 <> 1) and (t2 <> 0) do
  begin
   t1:= t2 mod 2;
   k:= IntToStr(t1);
   push(k);
   t2:= t2 div 2;
  end;
   k:= IntToStr(t2);
   push(k);
  while (PBS <> 0) do
  begin
   pop(k);
   result:= result + k;
  end;
end;

function FindElAr(ar: TArTree; size: byte; key: byte): byte;
var
 i: integer;
begin
 for i:= 0 to size-1 do
  if ar[i].key = key then
  begin
   result:= i;
   exit;
  end;
end;

function FindElAr(ar: TArFRec; size: byte; key: byte): byte;
var
 i: integer;
begin
 for i:= 0 to size-1 do
  if ar[i].key = key then
  begin
   result:= i;
```

```
   exit;
  end;
end;


procedure reverse(var pesn: TStr);
var
 i: integer; tmp: char;
begin
for i:=1 to (length(pesn) div 2) do
begin
    tmp:=pesn[i];
    pesn[i]:=pesn[length(pesn)-i+1];
    pesn[length(pesn)-i+1]:= tmp;
end;
end;

procedure CreateKod(head: TPArNode; var ar: TArTree; size: byte);
var
 k: TKey;
begin
 if (head = nil) then exit;
   push(head^.kod);
   CreateKod(head^.left, ar, size);
   tmpi:= FindElAr(ar, size, head^.symb);
   cnti:=0;
   while (PBS<>0) do
   begin
    inc(cnti);
    pop(k);
    ar[tmpi].huff:= ar[tmpi].huff + k;
   end;
   reverse(ar[tmpi].huff);
   PBS:= PBS + cnti;
   CreateKod(head^.right, ar, size);
   dec(PBS);
end;

procedure GetNode(a: TArTree; size: integer);
var
 tmp: TPArNode;
begin
 tmp:= a[ size-1].st_tree;
 while (tmp^.parent <> nil) do
  tmp:= tmp^.parent;
 head_tree:= tmp;
end;

function EndHaffman(a: TArTree; size: integer): extended;
var
 tmp: TPArNode;
begin
 tmp:= a[ size-1].st_tree;
 while (tmp^.parent <> nil) do
  tmp:= tmp^.parent;
 result:= tmp^.value;
end;

procedure OptimizeKod(var ar: TArTree; size: byte);
var
 i, j: integer; ok: boolean; str: string; news, tmp: string[1];
begin
 for i:= 0 to size-1 do
 begin
  ok:= true;
  SetLength(str, length(ar[i].huff));
  str:= ar[i].huff;
  tmp:= str[1];
  if tmp = '0' then
```

```
  begin
   for j:=1 to length(str) do
    if (str[j] <> '0') then
     ok:= false;
   end else
     ok:= false;
   if ok then
   begin
   SetLength(news, length(ar[i].huff));
   for j:=1 to length(ar[i].huff) do
    news[j]:= '1';
    ar[i].huff:= news;
   end;
 end;
end;

function GetText(ar: TArTree; size: byte): widestring;
var
 i: byte;
begin
result:= '';
 for i:= 0 to size-1 do
  result:= result + a[i].huff;
end;

procedure Countall(a: TArTree; size: byte);
var
 i: integer;
 H, Hmax, LiPi, Kss, Koe: real;
begin
  Hmax:= log2(size);
  H:= 0;
  LiPi:= 0;
  for i:= 0 to size - 1 do
  begin
   H:= H + a[i].n0 * log2(a[i].n0);
   LiPi:= LiPi + length(a[i].huff) * a[i].n0;
  end;
  H:= (-1)*H;
  liPi:= log2(LiPi);
  Kss:= Hmax / LiPi;
  Koe:= H / LiPi;
  Form1.StringGrid1.Cells[1,0]:= FloatToStr(Hmax);
  Form1.StringGrid1.Cells[1,1]:= FloatToStr(H);
  Form1.StringGrid1.Cells[1,2]:= FloatToStr(LiPi);
  Form1.StringGrid1.Cells[1,3]:= FloatToStr(Kss);
  Form1.StringGrid1.Cells[1,4]:= FloatToStr(Koe);
end;

procedure TForm1.Button1Click(Sender: TObject);
var
 s, tmps: string;
 tmp: char;
 i, i1, i2, j: cardinal;
 tpos: integer;
 El, tel: TPElSp;
 ElTree, tmp1, tmp2: TPArNode;
 lvl, outkod, inkod: word;
 f1, H, LiPi: extended;
 rec: TFileRec;
 res: widestring;
 str: string;
 chk: byte;
begin
  Opendialog1.FilterIndex:= 1;
  head_lsp:= nil;
  head_tree:= nil;
  size:= 0;
  n:= 0;
```

```
PBS:= 0;
SetLength(a, 0);
OpenDialog1.Execute;
if OpenDialog1.FileName = '' then
begin
  ShowMessage('Не обрано файл!');
  exit;
end;
AssignFile(f, OpenDialog1.FileName);
Reset(f, 1);
while not eof(f) do
begin
 BlockRead(f, tmp, 1);
 if FindEl(head_lsp, ord(tmp), tel) = false then
 begin
 New(El);
 El^.key:= ord(tmp);
 El^.cnt:= 1;
 AddEl(head_lsp, El);
 inc(size);
 inc(n);
 end else
 begin
 inc(tel^.cnt);
 inc(n);
 end;
end;
SetLength(a, size);
i:= 0;
while (head_lsp <> nil) do
begin
  FindElMax(head_lsp, tel);
  a[i].key:= tel^.key;
  a[i].cnt:= tel^.cnt;
  a[i].n0:= a[i].cnt / n;
  a[i].deep:= false;
  a[i].huff:='';
  new(ElTree);
  ElTree^.value:= 0;
  ElTree^.left:= nil;
  ElTree^.right:= nil;
  ElTree^.parent:= nil;
  ElTree^.kod:= '';
  ElTree^.symb:= a[i].key;
  a[i].st_tree:= ElTree;
  inc(i);
  DelEl(head_lsp, tel^.key);
end;
while (EndHaffman(a, size) <> 1) do
 begin
  i1:= FindMin(a, size);
  a[i1].deep:= true;
  i2:= FindMin(a, size);
  a[i2].n0:= a[i2].n0 + a[i1].n0;
  new(ElTree);
  ElTree^.value:= a[i2].n0;
  ElTree^.parent:= nil;
  ElTree^.kod:='';
  tmp1:= a[i1].st_tree;
  while (tmp1^.parent <> nil) do
   tmp1:= tmp1^.parent;
  tmp2:= a[i2].st_tree;
  while (tmp2^.parent <> nil) do
   tmp2:= tmp2^.parent;
  ElTree^.left:= tmp1;
  ElTree^.right:= tmp2;
  tmp1^.parent:= ElTree;
  tmp1^.kod:= '0';
  tmp2^.kod:= '1';
```

```
    tmp2^.parent:= ElTree;
  end;
  GetNode(a, size);
  CreateKod(head_tree, a, size);
  seek(f, 0);
  assignfile(fo, 'test.asd');
  rewrite(fo, 1);
  BlockWrite(fo, size, 1);
  BlockWrite(fo, size, 1);
  for i:=0 to size-1 do
  begin
   rec.key:= a[i].key;
   rec.kod:= StrBToInt(a[i].huff);
   rec.size_kod:= length(a[i].huff);
   BlockWrite(fo, rec, sizeof(TFileRec));
  end;
  j:= 0;
   res:= '';
   while (not eof(f)) do
   begin
     tpos:= filepos(f);
     BlockRead(f, inkod, 1);
     i:= FindElAr(a, size, inkod);
     res:= res + a[i].huff;
     inc(j);
   end;
   while length(res)<>0 do
   begin
    i:= 0;
    str:= '';
    while (i<>8) and (i<>length(res)) do
    begin
     inc(i);
     str:= str + res[i];
    end;
    delete(res, 1, i);
    outkod:= StrBToInt(str);
    BlockWrite(fo, outkod, 1);
   end;
    seek(fo, 1);
    chk:= length(str);
    BlockWrite(fo, chk, 1);
  Countall(a, size);
  for i:=0 to size-1 do
  begin
    SG1.Cells[0,i+1]:= IntToStr(a[i].key);
    SG1.Cells[1,i+1]:= chr(a[i].key);
    SG1.Cells[2,i+1]:= IntToStr(a[i].cnt);
    SG1.Cells[3,i+1]:= IntToStr(length(a[i].huff));
    SG1.Cells[4,i+1]:= a[i].huff;
    SG1.RowCount:= SG1.RowCount + 1;
  end;
    SG1.RowCount:= SG1.RowCount - 1;
    closefile(f);
    closefile(fo);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 DelSp(head_lsp);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  PBS:= 0;
  SG1.Cells[0,0]:= 'Код';
  SG1.Cells[1,0]:= 'Символ';
  SG1.Cells[2,0]:= 'Частота';
  SG1.Cells[3,0]:= 'Довжина';
```

```
  SG1.Cells[4,0]:= 'Код Хаффмана';
  StringGrid1.Cells[0,0]:= 'H';
  StringGrid1.Cells[0,1]:= 'Hmax';
  StringGrid1.Cells[0,2]:= 'Lcp';
  StringGrid1.Cells[0,3]:= 'Кс.с.';
  StringGrid1.Cells[0,4]:= 'К.е.';
end;

function GetFromKod(a: TArFRec; size: byte; key: string; var pos: byte):
boolean;
var
 i: byte;
begin
 result:= false;
 pos:= 0;
 for i:=0 to size-1 do
  if a[i].kod = key then
  begin
   result:= true;
   pos:= i;
   exit;
  end;
end;

function CorrectKod(str: string; itSize: byte): string;
var
 i, j, k: integer; tmp: string[16];
begin
  if length(str) = itSize then
  begin
   result:= str;
   exit;
  end;
  tmp:= '0000000000000000';
  i:= itSize - length(str);
  delete(tmp, itSize+1, 16);
  k:= 1;
  for j:=i+1  to itSize do
  begin
    tmp[j]:= str[k];
    inc(k);
  end;
  result:= tmp;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
fin, fout: file; arr: TArFRec; i, j, ind:integer;
 size1, pos, tmp, chk: byte;
rec: TFileRec; data: byte; res, str: widestring;
tmps: string;
begin
  Opendialog1.FilterIndex:= 2;
  size1:= 0;
  OpenDialog1.Execute;
  if OpenDialog1.FileName = '' then
  begin
    ShowMessage('Не обраний вихідний файл!');
    exit;
  end;
  AssignFile(fin, OpenDialog1.FileName);
  Reset(fin, 1);
  SaveDialog1.Execute;
  if SaveDialog1.FileName = '' then
  begin
    ShowMessage('Не обраний файл для архівації!');
    exit;
  end;
  AssignFile(fout, SaveDialog1.FileName);
```

```
  rewrite(fout, 1);
  BlockRead(fin, size1, 1);
  BlockRead(fin, chk, 1);
  SetLength(arr, size1);
  for i:= 0 to size 1-1 do
  begin
   BlockRead(fin, rec, sizeof(TFileRec));
   arr[i].key:= rec.key;
   arr[i].kod:= CorrectKod(IntToStr(rec.kod), rec.size_kod);
   arr[i].size_kod:= rec.size_kod;
  end;
   res:= '';
   while (not eof(fin)) do
   begin
     if ((filepos(fin) + 1) <> filesize(fin)) then
     begin
      BlockRead(fin, data, 1);
      tmps:= CorrectKod(IntToStr(data), 8);
      res:= res + tmps;
      inc(j);
     end else
     begin
      BlockRead(fin, data, 1);
      tmps:= CorrectKod(IntToStr(data), chk);
      res:= res + tmps;
     end;
   end;
   while length(res)<>0 do
   begin
    i:= 0;
    str:= '';
    while (GetFromKod(arr, size1, str, pos)=false) and (i<>length(res)) do
    begin
     inc(i);
     str:= str + res[i];
    end;
    delete(res, 1, i);
    BlockWrite(fout, arr[pos], 1);
  end;
end;

end.
```

Файл Main.pas - головне вікно програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ComCtrls, ExtCtrls, frFilePanel, StdCtrls, FileCtrl, ImgList,
  AppEvnts, Buttons, XPMan;

type
  TfmDAR = class(TForm)
    mmMenu: TMainMenu;
    miFile: TMenuItem;
    miExit: TMenuItem;
    miHelp: TMenuItem;
    miAbout: TMenuItem;
    frFilePanel: TfrFilePanel;
    pnTop: TPanel;
    sbStatus: TStatusBar;
    FileListBox1: TFileListBox;
    ImageList1: TImageList;
    miSplit1: TMenuItem;
    miAddToArchive: TMenuItem;
    miExtract: TMenuItem;
    miExtractTo: TMenuItem;
    miFileInformation: TMenuItem;
    bbAddToArchive: TBitBtn;
    bbExtractTo: TBitBtn;
    bbFileInformation: TBitBtn;
    lbPath: TLabel;
    lbItem: TLabel;
    SaveDialog1: TSaveDialog;
    ApplicationEvents1: TApplicationEvents;
    XPManifest1: TXPManifest;
    N1: TMenuItem;
    procedure miExitClick(Sender: TObject);
    procedure miAboutClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure miExtractToClick(Sender: TObject);
    procedure miAddToArchiveClick(Sender: TObject);
    procedure ApplicationEvents1Hint(Sender: TObject);
    procedure bbFileInformationClick(Sender: TObject);
    procedure frFilePanelbbRefreshClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    Procedure Compress;
    Procedure DeCompress;
  end;

var
  fmDAR: TfmDAR;

implementation

uses About, DeCompressor, fmExtractDir, fmProcess;//, frFilePanel;

Var
  FirstRun:Boolean;

{$R *.DFM}

Procedure TfmDAR.Compress;
Var
```

```
  NewFileName, OldFileName:String;
Begin
  NewFileName:=ChangeFileExt(lbItem.Caption, '.dar');
  SaveDialog1.FileName:=NewFileName;

  If Not(SaveDialog1.Execute) Then Exit;
  NewFileName:=SaveDialog1.FileName;
  OldFileName:=lbPath.Caption+lbItem.Caption;
  If OldFileName=NewFileName Then
  Begin
    Application.MessageBox('Неможливо архівувати файл у собе', 'Помилка!',
MB_ICONERROR Or MB_OK);
    Exit;
  End;

  Enabled:=False;
  fmProcess.Compress(OldFileName, NewFileName);
  Enabled:=True;
End;

Procedure TfmDAR.DeCompress;
Var
  NewFileName, OldFileName:String;
Begin
  fmExtractDir.dlbxDirs.Directory:=lbPath.Caption;
  If fmExtractDir.ShowModal=mrCancel Then Exit;

  OldFileName:=lbPath.Caption+lbItem.Caption;
  NewFileName:=IncludeTrailingBackslash(fmExtractDir.dlbxDirs.Directory);

  Enabled:=False;
  fmProcess.DeCompress(OldFileName, NewFileName);
  Enabled:=True;
End;

Function _ShowProcess(OrigSize, OrigPos, Comp:Integer):Boolean;
Begin
  Result:=fmProcess.ShowProcess(OrigSize, OrigPos, Comp);
End;

procedure TfmDAR.miExitClick(Sender: TObject);
begin
  Close;
end;

procedure TfmDAR.miAboutClick(Sender: TObject);
begin
  fmAbout.ShowModal;
end;

procedure TfmDAR.FormActivate(Sender: TObject);
begin
  If FirstRun Then
  Begin
    frFilePanel.Init(FileListBox1, ImageList1, nil, lbPath, lbItem);
    FirstRun:=False;
  End;
end;

procedure TfmDAR.FormCreate(Sender: TObject);
begin
  FirstRun:=True;
  CompressProcessProc:=_ShowProcess;
end;

procedure TfmDAR.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  frFilePanel.Done;
end;
```

```
//розархівувати
procedure TfmDAR.miExtractToClick(Sender: TObject);
begin
  If UpperCase(ExtractFileExt(lbItem.Caption))<>'.DAR' Then Exit;
  DeCompress;
  frFilePanel.Refresh;
end;

//архівувати
procedure TfmDAR.miAddToArchiveClick(Sender: TObject);
begin
  Compress;
  frFilePanel.Refresh;
end;

procedure TfmDAR.ApplicationEvents1Hint(Sender: TObject);
begin
  sbStatus.Panels[0].Text:=Application.Hint;
end;
//довідка
procedure TfmDAR.bbFileInformationClick(Sender: TObject);
begin
 fmAbout.ShowModal;
end;
//обновити
procedure TfmDAR.frFilePanelbbRefreshClick(Sender: TObject);
begin
  frFilePanel.bbRefreshClick(Sender);

end;

end.
```

Файл Files.pas - робота з файлами

```
  unit Files;

interface

Uses
  classes, SysUtils, FileCtrl,
  StrConsts;

Const
  MaxFilesCount=10240;
  MaxFileTypes=36;

Type
  TImagesIndex=Array [0.. MaxFileTypes-1, 0..1] Of String;

  TRecordID=Object
    ID:Integer;
  End;
  TFileRecord=Record
    ID:Integer;
    Name:String;
    Ext:String;
    Size:Integer;
    DateTime:TDateTime;
    Attr:Integer;
    Checked:Boolean;
    Tag:Integer;
  End;
  TFileFormattedRecord=Record
    Name,
    Ext,
    Size,
    DateTime,
    Attr:String;
    Checked:Boolean;
    Tag:Integer;
  End;

  TFiles=Object
  Protected
    slDirs, slFiles:TStringList;
    DirsID, FilesID:Array[0.. MaxFilesCount-1] Of TRecordID;

    Procedure FinishSort(Ascending, DirAscending:Boolean);
  Public
    ItemsID:Array[0.. MaxFilesCount-1] Of Integer;
    Items:Array[0.. MaxFilesCount-1] Of TFileRecord;
    ItemsCount:Integer;
    Tag:Integer;

    Constructor Init;
    Destructor Done;

    Function Add(Item:TFileRecord):Boolean;
    Procedure Clear;
    Procedure GetItem(ItemNo:Integer; Var Item:TFileRecord);

    Procedure SortByName(Ascending:Boolean);
    Procedure SortByExt(Ascending:Boolean);
    Procedure SortBySize(Ascending:Boolean);
    Procedure SortByDateTime(Ascending:Boolean);
    Procedure SortByAttr(Ascending:Boolean);
    Procedure GetInfo(Var TotalDirs, TotalFiles, CheckedDirs,
CheckedFiles:Integer; Var TotalSize, CheckedSize:Int64);
  End;
```

```
Const
  DirUpItem:TFileRecord=(
    ID:0;
    Name:ConstDirUp;
    Ext:'';
    Size:0;
    DateTime:0;
    Attr:faDirectory;
    Checked:False;
    Tag:0);

  Images:TImagesIndex=(
    ('','19'),
    ('EXE','14'),
    ('BAT','14'),
    ('COM','14'),
    ('LNK','19'),
    ('PIF','17'),
    ('TXT','20'),
    ('HTM','21'),
    ('HTML','21'),
    ('DOC','22'),
    ('DOT','22'),
    ('XLS','23'),
    ('RAR','26'),
    ('ZIP','27'),
    ('ARJ','27'),
    ('PAS','28'),
    ('DPR','28'),
    ('DFM','28'),
    ('DCU','28'),
    ('MP3','30'),
    ('M3U','31'),
    ('WAV','30'),
    ('MID','30'),
    ('OGG','30'),
    ('AVI','32'),
    ('MPE','32'),
    ('MPG','32'),
    ('MPEG','32'),
    ('BMP','34'),
    ('GIF','33'),
    ('JPG','33'),
    ('PCX','33'),
    ('ICO','33'),
    ('INI','20'),
    ('REG','35'),
    ('DAR','36'));

Function IsDirectory(Item:TFileRecord):Boolean;
Function IsRoot(Path:String):Boolean;

Function GetFormattedName(Name:String):String;
Function GetFormattedExt(Ext:String):String;
Function GetFormattedSize(Size:Integer; SizeWidth:Byte;
UseSeparators:Boolean):String;
Function GetFormattedDateTime(DateTime:TDateTime):String;
Function GetFormattedAttr(Attr:Integer):String;
Procedure GetFormattedItem(Item:TFileRecord; Var
FormattedItem:TFileFormattedRecord);

Procedure GetItemByFileName(FileName:String; Var Item:TFileRecord);

Function GetItemImageIndex(Item:TFileRecord):Integer;

Function GetCompactSize(Size:Int64):String;

Function GetFullName(Item:TFileRecord):String;
```

```
implementation


Constructor TFiles.Init;
Begin
  slDirs:=TStringList.Create;
  slFiles:=TStringList.Create;
  ItemsCount:=0;
  Tag:=0;
End;

Destructor TFiles.Done;
Begin
  slDirs.Free;
  slFiles.Free;
End;

Function TFiles.Add(Item:TFileRecord):Boolean;
Begin
  Result:=False;
  If ItemsCount>=MaxFilesCount Then Exit;
  Result:=True;
  Item.ID:=ItemsCount;
  Items[ItemsCount]:=Item;
  ItemsID[ItemsCount]:=ItemsCount;
  Inc(ItemsCount);
End;

Procedure TFiles.Clear;
Begin
  slDirs.Clear;
  slFiles.Clear;
  ItemsCount:=0;
End;

Procedure TFiles.GetItem(ItemNo:Integer; Var Item:TFileRecord);
Begin
  If ((ItemNo<0) Or (ItemNo>=ItemsCount)) Then Exit;
  Item:=Items[ItemsID[ItemNo]];
End;

Procedure TFiles.FinishSort(Ascending, DirAscending:Boolean);
Var
  i:Integer;
Begin
  slDirs.Sort;
  slFiles.Sort;

  If slDirs.Count>0 Then
    For i:=0 To slDirs.Count-1 Do
      If DirAscending Then
        ItemsID[i]:=TRecordID(slDirs.Objects[i]).ID
      Else
        ItemsID[i]:=TRecordID(slDirs.Objects[slDirs.Count-i-1]).ID;
  If slFiles.Count>0 Then
    For i:=slDirs.Count To slFiles.Count-1+slDirs.Count Do
      If Ascending Then
        ItemsID[i]:=TRecordID(slFiles.Objects[ i-slDirs.Count]).ID
      Else
        ItemsID[i]:=TRecordID(slFiles.Objects[slFiles.Count+slDirs.Count-i-
1]).ID;
End;

Procedure TFiles.SortByName(Ascending:Boolean);
Var
  Item:TFileRecord;
  i:Integer;
Begin
```

```
  If ItemsCount<=0 Then Exit;
  slDirs.Clear;
  slFiles.Clear;
  For i:=0 To ItemsCount-1 Do
  Begin
    Item:=Items[i];
    If IsDirectory(Item) Then
    Begin
      slDirs.Add(GetFormattedName(Item.Name));
      DirsID[slDirs.Count-1].ID:=i;
      slDirs.Objects[slDirs.Count-1]:=TObject(DirsID[slDirs.Count-1]);
    End
    Else Begin
      slFiles.Add(GetFormattedName(Item.Name));
      FilesID[slFiles.Count-1].ID:=i;
      slFiles.Objects[slFiles.Count-1]:=TObject(FilesID[slFiles.Count-1]);
    End;
  End;
  FinishSort(Ascending, Ascending);
End;

Procedure TFiles.SortByExt(Ascending:Boolean);
Var
  Item:TFileRecord;
  i:Integer;
Begin
  If ItemsCount<=0 Then Exit;
  slDirs.Clear;
  slFiles.Clear;
  For i:=0 To ItemsCount-1 Do
  Begin
    Item:=Items[i];
    If IsDirectory(Item) Then
    Begin
      slDirs.Add(GetFormattedName(Item.Name));
      DirsID[slDirs.Count-1].ID:=i;
      slDirs.Objects[slDirs.Count-1]:=TObject(DirsID[slDirs.Count-1]);
    End
    Else Begin
      slFiles.Add(GetFormattedExt(Item.Ext));
      FilesID[slFiles.Count-1].ID:=i;
      slFiles.Objects[slFiles.Count-1]:=TObject(FilesID[slFiles.Count-1]);
    End;
  End;
  FinishSort(Ascending, True);
End;

Procedure TFiles.SortBySize(Ascending:Boolean);
Var
  Item:TFileRecord;
  i:Integer;
  TmpStr:String;
Begin
  If ItemsCount<=0 Then Exit;
  slDirs.Clear;
  slFiles.Clear;
  For i:=0 To ItemsCount-1 Do
  Begin
    Item:=Items[i];
    If IsDirectory(Item) Then
    Begin
      slDirs.Add(GetFormattedName(Item.Name));
      DirsID[slDirs.Count-1].ID:=i;
      slDirs.Objects[slDirs.Count-1]:=TObject(DirsID[slDirs.Count-1]);
    End
    Else Begin
      TmpStr:=GetFormattedSize(Item.Size, 10, False);
      slFiles.Add(TmpStr);
      FilesID[slFiles.Count-1].ID:=i;
```

```
      slFiles.Objects[slFiles.Count-1]:=TObject(FilesID[slFiles.Count-1]);
    End;
  End;
  FinishSort(Ascending, True);
End;

Procedure TFiles.SortByDateTime(Ascending:Boolean);
Var
  Item:TFileRecord;
  i:Integer;
  TmpStr:String;
Begin
  If ItemsCount<=0 Then Exit;
  slDirs.Clear;
  slFiles.Clear;
  For i:=0 To ItemsCount-1 Do
  Begin
    Item:=Items[i];
    If IsDirectory(Item) Then
    Begin
      slDirs.Add(GetFormattedName(Item.Name));
      DirsID[slDirs.Count-1].ID:=i;
      slDirs.Objects[slDirs.Count-1]:=TObject(DirsID[slDirs.Count-1]);
    End
    Else Begin
      Str(Item.DateTime:16:10, TmpStr);
      slFiles.Add(TmpStr);
      FilesID[slFiles.Count-1].ID:=i;
      slFiles.Objects[slFiles.Count-1]:=TObject(FilesID[slFiles.Count-1]);
    End;
  End;
  FinishSort(Ascending, True);
End;

Procedure TFiles.SortByAttr(Ascending:Boolean);
Var
  Item:TFileRecord;
  i:Integer;
Begin
  If ItemsCount<=0 Then Exit;
  slDirs.Clear;
  slFiles.Clear;
  For i:=0 To ItemsCount-1 Do
  Begin
    Item:=Items[i];
    If IsDirectory(Item) Then
    Begin
      slDirs.Add(GetFormattedName(Item.Name));
      DirsID[slDirs.Count-1].ID:=i;
      slDirs.Objects[slDirs.Count-1]:=TObject(DirsID[slDirs.Count-1]);
    End
    Else Begin
      slFiles.Add(GetFormattedAttr(Item.Attr));
      FilesID[slFiles.Count-1].ID:=i;
      slFiles.Objects[slFiles.Count-1]:=TObject(FilesID[slFiles.Count-1]);
    End;
  End;
  FinishSort(Ascending, True);
End;

Procedure TFiles.GetInfo(Var TotalDirs, TotalFiles, CheckedDirs,
CheckedFiles:Integer; Var TotalSize, CheckedSize:Int64);
Var
  i:Integer;
Begin
  TotalDirs:=0;
  TotalFiles:=0;
  CheckedDirs:=0;
  CheckedFiles:=0;
```

```
  TotalSize:=0;
  CheckedSize:=0;

  If ItemsCount<=0 Then Exit;
  For i:=0 To ItemsCount-1 Do
  Begin
    If IsDirectory(Items[i]) Then
    Begin
      Inc(TotalDirs);
      If Items[i].Checked Then Inc(CheckedDirs);
    End
    Else Begin
      Inc(TotalFiles);
      If Items[i].Checked Then Inc(CheckedFiles);
    End;
    If Items[i].Checked Then CheckedSize:=CheckedSize+Items[i].Size;
    TotalSize:=TotalSize+Items[i].Size;
  End;
End;


Function IsDirectory(Item:TFileRecord):Boolean;
Begin
  If ((Item.Attr And faDirectory)>0) Then Result:=True Else Result:=False;
End;

Function IsRoot(Path:String):Boolean;
Begin
  Result:=(Path=IncludeTrailingBackslash(ExtractFileDrive(Path)));
End;

Function GetFormattedName(Name:String):String;
Begin
  Result:=Name;
End;

Function GetFormattedExt(Ext:String):String;
Begin
  Result:=Ext;
End;

Function GetFormattedSize(Size:Integer; SizeWidth:Byte;
UseSeparators:Boolean):String;
Var
  i, i3:Integer;
  Res:String;
Begin
  Str(Size:SizeWidth, Result);
  If Not(UseSeparators) Then Exit;
  If Length(Result)<=3 Then Exit;
  i3:=0;
  Res:=Result;
  Result:='';
  For i:=Length(Res) DownTo 1 Do
  Begin
    If i3=3 Then
    Begin
      Result:=ConstSizeSeparator+Result;
      i3:=0;
    End;
    Result:=Res[i]+Result;
    Inc(i3);
  End;
End;

Function GetFormattedDateTime(DateTime:TDateTime):String;
Begin
  Result:=DateTimeToStr(DateTime);
End;
```

```
Function GetFormattedAttr(Attr:Integer):String;
Begin
  Result:=' ---';
  If ((Attr And faReadOnly)>0) Then Result[1]:=ConstReadOnly;
  If ((Attr And faArchive)>0) Then Result[2]:=ConstArchive;
  If ((Attr And faHidden)>0) Then Result[3]:=ConstHidden;
  If ((Attr And faSysFile)>0) Then Result[4]:=ConstSystem;
End;

Procedure GetFormattedItem(Item:TFileRecord; Var
FormattedItem:TFileFormattedRecord);
Begin
  FormattedItem.Name:=GetFormattedName(Item.Name);
  FormattedItem.Ext:=GetFormattedExt(Item.Ext);
  If IsDirectory(Item) Then
  Begin
    FormattedItem.Size:=ConstDirectory;
  End
  Else Begin
    FormattedItem.Size:=GetFormattedSize(Item.Size, 1, True);
  End;
  FormattedItem.DateTime:=GetFormattedDateTime(Item.DateTime);
  FormattedItem.Attr:=GetFormattedAttr(Item.Attr);
End;

Procedure GetItemByFileName(FileName:String; Var Item:TFileRecord);
Var
  F:File;
Begin
  FillChar(Item, SizeOf(Item), 0);
  If FileExists(FileName) Then
  Begin
    Item.Name:=ExtractFileName(FileName);
    Item.Ext:=ExtractFileExt(FileName);
    If Item.Ext<>'' Then
    Begin
      Delete(Item.Name, Length(Item.Name)-Length(Item.Ext)+1, Length(Item.Ext));
      Delete(Item.Ext, 1, 1);
    End;
    Item.DateTime:=FileDateToDateTime(FileAge(FileName));
    Item.Attr:=FileGetAttr(FileName);

   {$ I-}
    AssignFile(F, FileName);
    Reset(F, 1);
    If IOResult=0 Then
      Item.Size:=FileSize(F)
    Else
      Item.Size:=0;
    Close(F);
   {$I+}
    Exit;
  End;

  Item.Name:=ExtractFileName(FileName);
  Delete(Item.Name, 1, 1);
  Delete(Item.Name, Length(Item.Name), 1);
  If DirectoryExists(ExtractFilePath(FileName)+Item.Name) Then
  Begin
    Item.Ext:='';
    Item.Size:=0;
    Item.DateTime:=0;
    Item.Attr:=faDirectory;
  End
  Else Begin
    Item.Name:='';
  End;
End;
```

```
Function GetItemImageIndex(Item:TFileRecord):Integer;
Var
  i:Integer;
Begin
  If Item.Name=ConstDirUp Then
  Begin
    Result:=10;
    Exit;
  End;
  If IsDirectory(Item) Then
  Begin
    Result:=11;
    Exit;
  End;
  Item.Ext:=UpperCase(Item.Ext);
  For i:=0 To MaxFileTypes-1 Do
  Begin
    If Item.Ext=Images[i, 0] Then
    Begin
      Result:=StrToInt(Images[i, 1]);
      Exit;
    End;
  End;
  Result:=19;
End;

Function GetCompactSize(Size:Int64):String;
Begin
  If Size<=ConstBytesLimit Then
  Begin
    Result:=GetFormattedSize(Size,1, True)+' '+ConstBytes;
    Exit;
  End;
  If Size<=ConstKBytesLimit Then
  Begin
    Result:=GetFormattedSize((Size Div 1024), 1, True)+' '+ConstKBytes;
    Exit;
  End;
  Result:=GetFormattedSize((Size Div (1024*1024)), 1, True)+' '+ConstMBytes;
End;

Function GetFullName(Item:TFileRecord):String;
Begin
  If Item.Ext='' Then
    Result:=Item.Name
  Else
   Result:=Item.Name+'.'+Item.Ext;
End;

end.
```

Файл FilesEx.pas - обробка помилок

```
unit FilesEx;

interface
Uses
  SysUtils, FileCtrl, ShellApi, Windows;

Const
  F_ER_SUCCESS=0;
  F_ER_HIMSELF=-1;
  F_ER_EXISTS =-2;
  F_ER_NOT_EXISTS=-3;
  F_ER_DIREXISTS=-4;
  F_ER_NOTCOPY=-128;
  F_ER_ERROR=-255;

  ConstCopyToDir:String='';

Procedure GetDiskSize(CurrentDrive:Char; Var TotalBytes, TotalFree:Int64);
Procedure GetRealDiskSize(Drive:Char; Var TotalBytes, TotalFree:Double);

Function ExecuteOneFile(WorkDir, FileName, Params:String):Integer;
Function GetExecuteError(ErrorCode:Integer):String;

Function CopyOneFile(FromFile, ToFile:String; PrevCheck:Boolean):Integer;
Function GetFileError(ErrorCode:Integer):String;

Function RenameOneFile(OldName, NewName:String):Integer;
Function DeleteOneFile(FileName:String):Integer;
Function DeleteOneDir(FileName:String):Integer;

Function CreateOneFolder(FolderName:String):Integer;

implementation


function GetDiskFreeSpaceEx(lpDirectoryName: PAnsiChar;
              var lpFreeBytesAvailableToCaller : Integer;
              var lpTotalNumberOfBytes: Integer;
              var lpTotalNumberOfFreeBytes: Integer) : boolean;
              stdcall;
              external 'kernel32'
              name 'GetDiskFreeSpaceEx';

Procedure GetDiskSize(CurrentDrive:Char; Var TotalBytes, TotalFree:Int64);
Var
  Drive:Byte;
Begin
  CurrentDrive:=UpCase(CurrentDrive);
  Drive:=Ord(CurrentDrive)-64;
  TotalBytes:=DiskSize(Drive);
  TotalFree:=DiskFree(Drive);
End;

Procedure GetRealDiskSize(Drive:Char; Var TotalBytes, TotalFree:Double);
Var
  AvailToCall : integer;
  TheSize : integer;
  FreeAvail : integer;
  TheDrive:String;
Begin
  TheDrive:=Drive+':\'+#0;

  GetDiskFreeSpaceEx(@TheDrive[1], AvailToCall, TheSize, FreeAvail);
{$IFOPT Q+}
```

```
{$DEFINE TURNOVERFLOWON}
 {$ Q-}
{$ENDIF}
  If TheSize >= 0 then
    TotalBytes := TheSize
  Else
    if TheSize = -1 then
    begin
      TotalBytes := $7FFFFFFF;
      TotalBytes := TotalBytes * 2;
      TotalBytes := TotalBytes + 1;
    end
    else begin
      TotalBytes := $7FFFFFFF;
      TotalBytes := TotalBytes + abs($7FFFFFFF - TheSize);
    end;

  If AvailToCall >= 0 then
    TotalFree := AvailToCall
  else
    if AvailToCall = -1 then
    begin
      TotalFree := $7FFFFFFF;
      TotalFree := TotalFree * 2;
      TotalFree := TotalFree + 1;
    end
    else begin
      TotalFree := $7FFFFFFF;
      TotalFree := TotalFree + abs($7FFFFFFF - AvailToCall);
    end;
End;

Function ExecuteOneFile(WorkDir, FileName, Params:String):Integer;
Begin
  FileName:=FileName+#0;
  WorkDir:=WorkDir+#0;
  Params:=Params+#0;

  Result:=ShellExecute(0, 'open', @FileName[1], @Params[1], @WorkDir[1],
SW_SHOWNORMAL);
End;

Function GetExecuteError(ErrorCode:Integer):String;
Begin
  Result:='';
  If ErrorCode>32 Then Exit;
  Case ErrorCode Of
    0                     : Result:='Системі не вистачає пам'яті або ресурсів';
    ERROR_FILE_NOT_FOUND  : Result:='Файл не знайдений';
    ERROR_PATH_NOT_FOUND  : Result:='Шлях не знайдений';
    ERROR_BAD_FORMAT      : Result:='Помилка у форматі файлу';
    SE_ERR_ACCESSDENIED   : Result:='Доступ до файлу закритий';
    SE_ERR_ASSOCINCOMPLETE: Result:='Файлова асоціація невірна';
    SE_ERR_DLLNOTFOUND    : Result:='Динамічна бібліотека не знайдена';
    SE_ERR_NOASSOC        : Result:='Відсутній додаток, пов'язаний з даним типом
файлу';
    SE_ERR_OOM            : Result:='Недостатньо пам'яті для завершення
операції';
    SE_ERR_SHARE          : Result:='Помилка спільного доступу';
  Else
    Result:='Помилка при запуску програми';
  End;
End;

Function CopyOneFile(FromFile, ToFile:String; PrevCheck:Boolean):Integer;
Begin
  If PrevCheck Then
  Begin
    If FromFile=ToFile Then
```

```
    Begin
      Result:=F_ER_HIMSELF;
      Exit;
    End;
    If FileExists(ToFile) Then
    Begin
      Result:=F_ER_EXISTS;
      Exit;
    End;


  End;

  FromFile:=FromFile+#0;
  ToFile:=ToFile+#0;

  If Not(CopyFile(@FromFile[1], @ToFile[1], False)) Then
    Result:=F_ER_NOTCOPY
  Else
    Result:=F_ER_SUCCESS;
End;

Function GetFileError(ErrorCode:Integer):String;
Begin
  Result:='';
  Case ErrorCode Of
    F_ER_SUCCESS: Result:='';
    F_ER_HIMSELF: Result:='Не можна копіювати файл у себе';
    F_ER_EXISTS : Result:='Такий файл уже існує';
    F_ER_DIREXISTS : Result:='Така папка вже існує';
    F_ER_NOT_EXISTS : Result:='Такий файл відсутній';
    F_ER_NOTCOPY: Result:='Помилка при копіюванні';
    F_ER_ERROR: Result:='Помилка при роботі з файлом';
  End;
End;

Function RenameOneFile(OldName, NewName:String):Integer;
Begin
  If (FileExists(NewName) Or DirectoryExists(NewName)) Then
  Begin
    Result:=F_ER_EXISTS;
    Exit;
  End;
  If MoveFile(PChar(OldName+#0), PChar(NewName+#0)) Then
    Result:=F_ER_SUCCESS
  Else
    Result:=F_ER_NOTCOPY;
End;

Function DeleteOneFile(FileName:String):Integer;
Begin
  If Not(FileExists(FileName)) Then
  Begin
    Result:=F_ER_NOT_EXISTS;
    Exit;
  End;
{$ I-}
  FileSetAttr(FileName, faArchive);
  IOResult;
{$I+}
  If SysUtils.DeleteFile(FileName) Then
    Result:=F_ER_SUCCESS
  Else
    Result:=F_ER_ERROR;
End;

Function DeleteOneDir(FileName:String):Integer;
Begin
  If Not(DirectoryExists(FileName)) Then
```

```
  Begin
    Result:=F_ER_NOT_EXISTS;
    Exit;
  End;
  If RemoveDir(FileName) Then
    Result:=F_ER_SUCCESS
  Else
    Result:=F_ER_ERROR;
End;

Function CreateOneFolder(FolderName:String):Integer;
Begin
  If DirectoryExists(FolderName) Then
  Begin
    Result:=F_ER_DIREXISTS;
    Exit;
  End;
  If CreateDir(FolderName) Then
    Result:=F_ER_SUCCESS
  Else
    Result:=F_ER_ERROR;
End;

end.
```

Файл about.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TFmAbout = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FmAbout: TFmAbout;

implementation

{$R *.dfm}

procedure TFmAbout.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Програмне забезпечення системи кібербезпеки резервного
копіювання з архівацією для забезпечення цілісності');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник:  Смірнов С.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив:  студент   Довгополов Олександр Сергійович');
Memo1.Lines.Add('                     гр. КБ-20-3СК');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
end;

procedure TFmAbout.Button1Click(Sender: TObject);
begin
FmAbout.Close;
end;
end.
```