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 другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи стиснення та резервного дублювання даних”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Огер В.М.
«____» _____ 2023 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Огеру Владиславу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи стиснення та резервного дублювання даних*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи стиснення та резервного дублювання даних*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Огер В.М. Дослідження та програмна реалізація системи стиснення та резервного дублювання даних. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиснення та резервного дублювання даних.

Метою розробки є дослідження та програмна реалізація системи стиснення та резервного дублювання даних.

Об'єктом дослідження є процес стиснення та резервного дублювання даних.

Предметом дослідження є методи стиснення та резервного дублювання даних.

Методи дослідження базуються на методах теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи стиснення та резервного дублювання даних.

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 Embarcadero Delphi.

Ключові слова: комп'ютерна інженерія, стиснення, дублювання

ABSTRACT

Oher V.M. Research and software implementation of the data compression and backup duplication system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the system of data compression and backup duplication.

The purpose of the development is the research and software implementation of the data compression and backup duplication system.

The object of research is the process of data compression and backup duplication.

The subject of research is the methods of data compression and backup duplication.

Research methods are based on reliability theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the data compression and backup duplication system.

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 Embarcadero Delphi environment.

Keywords: computer engineering, compression, duplication

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. Backup – це резервна копія всіх важливих даних, втрата яких загрожує певними неприємностями і які зберігаються в окремому місці. Для когось «важливі дані» – це тільки робочі файли, когось сильно засмутить втрата колекції улюблених фільмів, тому й розміри резервних копій у значній мірі залежать від ситуації. Зберігати копії завжди потрібно на окремому диску, на якому немає основних даних. Краще, звичайно, щоб цей диск був фізичним, але підійде навіть інший логічний розділ вінчестера.

Що ж стосується носія для резервних копій, то серед доступних для рядового користувача лідирують жорсткі диски (як звичайні, з інтерфейсом SATA, так і зовнішні, що підключаються по USB, а також мережні – NAS). На другому місці по надійності зберігання стоять оптичні диски, благо сучасні носії Blu-ray уміщають до 50 Гбайт даних. Однак працювати з ними дуже незручно через низьку швидкість запису, а також через неможливість часткового відновлення резервної копії. Замикають список флеш-накопичувачі. Вартість їх у перерахуванні на 1 Гбайт великувата, також не занадто висока надійність, але для повсякденного копіювання невеликої кількості робочих файлів вони цілком підходять. Особняком стоять хмарні сервіси. Робота з ними специфічна й може влаштувати далеко не кожного. По-перше, дані, передані в Інтернеті, можуть стати надбанням зловмисника, якщо він зламає пароль. По-друге, як би не був швидкий інтернет-канал, відновлення дійсно великих архівів (десятки гігабайт) з резервними копіями даних буде забирати значний час. Нарешті, сам факт залежності від з'єднання з Інтернетом не дуже підходить для резервного копіювання дійсно важливих даних.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи стиснення та резервного дублювання даних.

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

– Огляд існуючих систем стиснення та резервного дублювання даних.

– Дослідження системи стиснення та резервного дублювання даних.

– Програмна реалізація системи стиснення та резервного дублювання даних.

Об'єктом дослідження є процес стиснення та резервного дублювання даних.

Предметом дослідження є методи стиснення та резервного дублювання даних.

Методи дослідження базуються на методах теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод стиснення та резервного дублювання даних.

– Розроблено вітчизняний продукт стиснення та резервного дублювання даних, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі стиснення та резервного дублювання даних.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиснення та резервного дублювання даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Для чого потрібні програми резервного копіювання? Адже ми звикли, що збереження файлів – дуже проста операція, що відбувається буквально двома клацаннями миші. Але коли даних багато, коли вони розташовані в різних місцях жорсткого диска й зміні піддаються не все з них, а також хочеться робити резервні копії регулярно, те, виявляється, досить корисно доручити це завдання спеціалізованому ПЗ. Крім того, є один тип резервної копії, що і зовсім неможливо виконати без спеціального програмного забезпечення, а саме образ жорсткого диска.

Такий образ, по-перше, організовується набагато швидше, ніж здійснювалося б копіювання всіх файлів з жорсткого диска, оскільки відбувається посекторне читання й головкам жорсткого диска не потрібно постійно звертатися до таблиці розміщення файлів. По-друге, розгортання резервної копії з образу дозволяє відновити після краху повністю працездатну систему з усіма програмами, налаштуваннями й файлами буквально за кілька мінут, на відміну від ситуації, коли доводиться встановлювати заново операційну систему й всі програми. Правда, при створенні образа вже не вийде не включати в нього непотрібні файли – їх краще просто зберігати на іншому логічному або фізичному диску. Щоб образ диска займав менше місця, його можна стискати на льоту, як і дозволяє робити більшість програм. Ступінь стиску порівнянний з роботою будь-якого іншого архіватору. Образи розділів диска можна використовувати не тільки для резервного копіювання, але й для швидкого створення робітничого середовища на декількох комп'ютерах з однаковою конфігурацією.

1.2 Область застосування

Компанії, які вже провели оптимізацію своєї IT-інфраструктури, як і раніше зіштовхуються із проблемами при виборі рішення для резервного копіювання:

– Зменшення вікна резервного копіювання. Рівень консолідації росте – потрібно забезпечувати швидке резервне копіювання великої кількості інформації. У деяких компаніях більша частина сервісів повинна працювати в режимі 24x7. У результаті доводиться йти на збільшення показника RPO (Recovery Point Objective) – відсувати точку відновлення.

– Швидкий ріст користувальницьких даних. Системи електронного документообігу, обробка й зберігання мультимедіа-контенту, поштові сервери – ріст обсягу даних може становити 30-50% у рік і більше. Виникає необхідність в оптимізації резервного копіювання, наприклад, шляхом застосування дедуплікації.

– Втрата даних у більшості випадків стає неприпустимою. Залежність бізнес-процесів компанії від IT-інфраструктури ростуть, тому змушені простої через неможливість швидкого відновлення працездатності із ключовими сервісами ведуть до більших збитків. Втрата користувальницьких даних у більшості випадків стає зовсім неприпустимою: наприклад, відновлення даних з паперових носіїв у системах електронного документообігу найчастіше буває просто нездійсненно.

Перераховані вище проблеми утворять замкнуте коло: зростаючі обсяги даних вимагають збільшення вікна резервного копіювання, але ви не можете собі цього дозволити через ріст кількості сервісів, що працюють у режимі 24x7.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стиснення та резервного дублювання даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Убудовані засоби резервного копіювання Windows

Почати огляд програм для резервного копіювання є сенс з тих, що завжди під рукою, а саме, з убудованих в Windows засобів архівації й відновлення. Знайти їх нескладно, наприклад в Windows 7 програма «Архівація й відновлення» перебуває в папці «Обслуговування» меню «Пуск», а в Windows 8 цей пункт розміщений на Панелі керування, причому «класичний» варіант доступний за назвою «Відновлення файлів Windows 7». Альтернативний варіант резервного копіювання надає функція «історія файлів». Вона зручна, коли мова йде про «відкат» змін окремих файлів, і досить надійна (зміни у файлах зберігаються на зовнішньому накопичувачі або на мережному диску). Однак ця функція не так комфортна, коли справа стосується повного краху системи, наприклад, внаслідок виходу з ладу жорсткого диска. Тут придасться можливість створення повної копії диска за допомогою «Архівації й відновлення».

Завжди корисно мати диск аварійного відновлення системи, що легко створити у всіх актуальних версіях Windows. Його можна буде використовувати як для спроб відновити систему після програмного збою, так і для розгортання резервної копії після серйозних неполадок з «залізом».

Засіб «Архівація й відновлення» здатен автоматизувати процес створення резервних копій. Досить указати місце зберігання копії (програма досить розумно радить застосовувати зовнішній жорсткий диск) і вибрати, що саме копіювати. Причому Windows, знову ж, у стані позначити свої системні каталоги сама. Один раз створений розклад дозволить не забивати собі голову необхідністю

самостійно робити резервні копії. У цьому й складається основний зміст використання програм замість ручного копіювання.

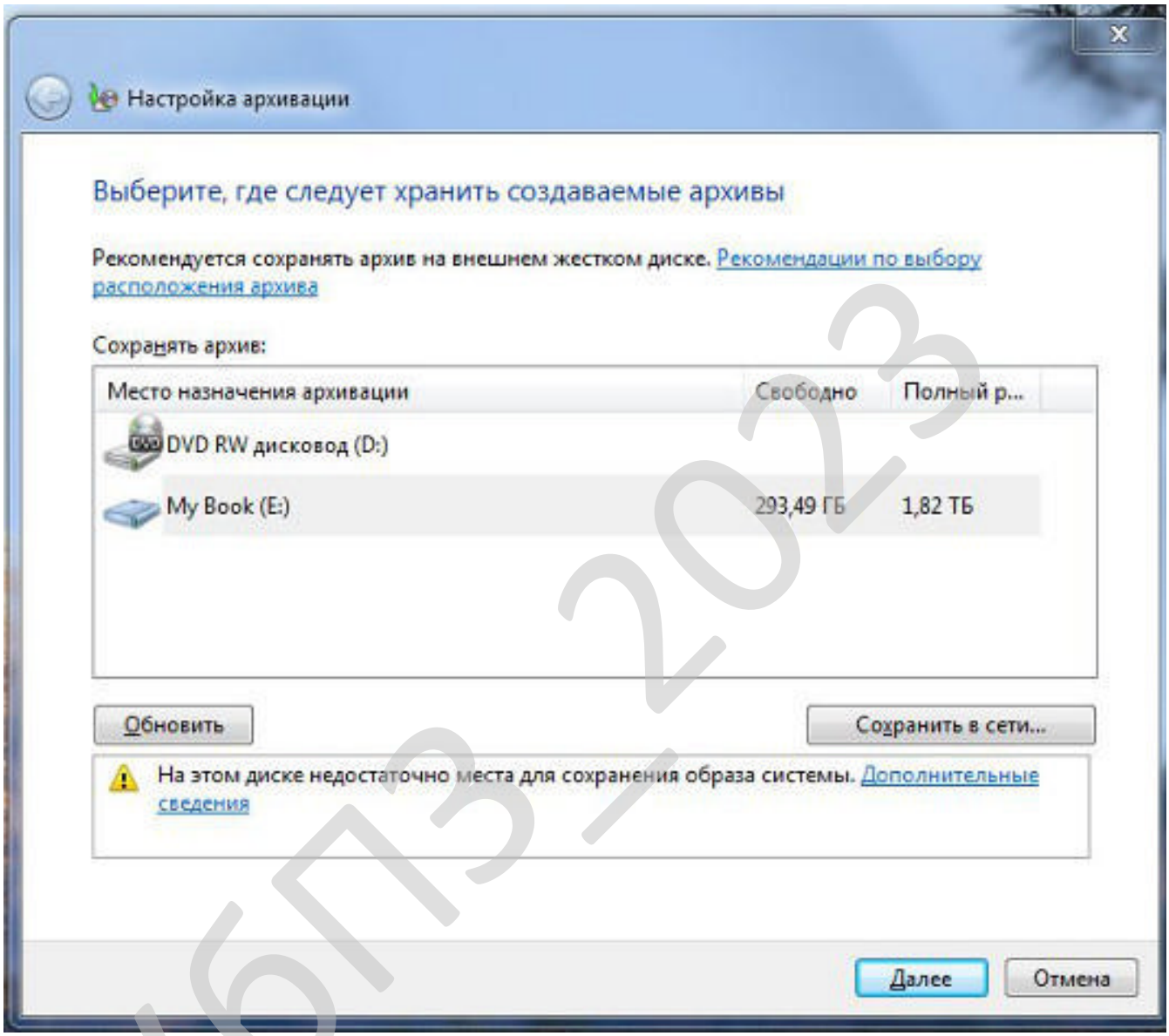

Рисунок 2.1 – Інтерфейс користувача убудованих засобів резервного копіювання Windows

Відновлення даних з резервної копії відбувається у зворотному порядку. Відтворити можна як каталоги з файлами, так і цілі диски, якщо було обране збереження логічних розділів. Windows не вміє стискати образ диска, а також створювати його посекторно, через що процес копіювання може серйозно

затягтися. Це, мабуть, головна претензія до даного засобу резервного копіювання. Також незадовільною є мінімальна кількість налаштувань. У результаті багато користувачів шукають альтернативні варіанти, удосталь пропоновані сторонніми розроблювачами ПЗ. Однак рядовому користувачеві досить засобів, убудованих в Windows, оскільки вони повністю виконують своє завдання – дозволяють створювати резервні копії обраних даних за розкладом.

Acronis True Image Home

Одним із самих популярних сторонніх пакетів для резервного копіювання й розгортання є Acronis True Image Home. Він платний. Також доступне доповнення Plus Pack, що дозволяє відновлювати систему на комп'ютері з конфігурацією, що відрізняється.

Як і треба з назви, програма Acronis True Image Home працює з образами (Image) логічних дисків, однак дає можливість досить гнучко набудовувати копіювання й окремі каталоги з файлами. Її найважливішим модулем є Acronis Media Builder – центр створення завантажувального носія. При наявності такий завантажувальної флешки можна дуже швидко (фактично, зі швидкістю прямого копіювання з диска на диск) створити повний образ логічного диска або відновити комп'ютер з образа. Вся процедура копіювання або відновлення на середньостатистичному вінчестері зі швидкістю 7200 rpm при місткості диска 30 Гбайт (містяться система й кілька програм) займає 5-7 мін.

Acronis True Image Home працює в трьох основних режимах: резервне копіювання документів і дисків, функціонування з використанням «майстра» (зручно для початківців) або в режимі вікна з повним набором функцій. Досвідченим користувачам варто відразу переходити в останній з них – режим вікна.

Максимальний набір налаштувань дозволяє вибирати, що, куди, як і коли копіювати. Причому налаштування планувальника досить докладні, можна навіть кожні кілька годин запускати копіювання особливо відповідальних даних. Їсти можливість настроїти й тип резервної копії.

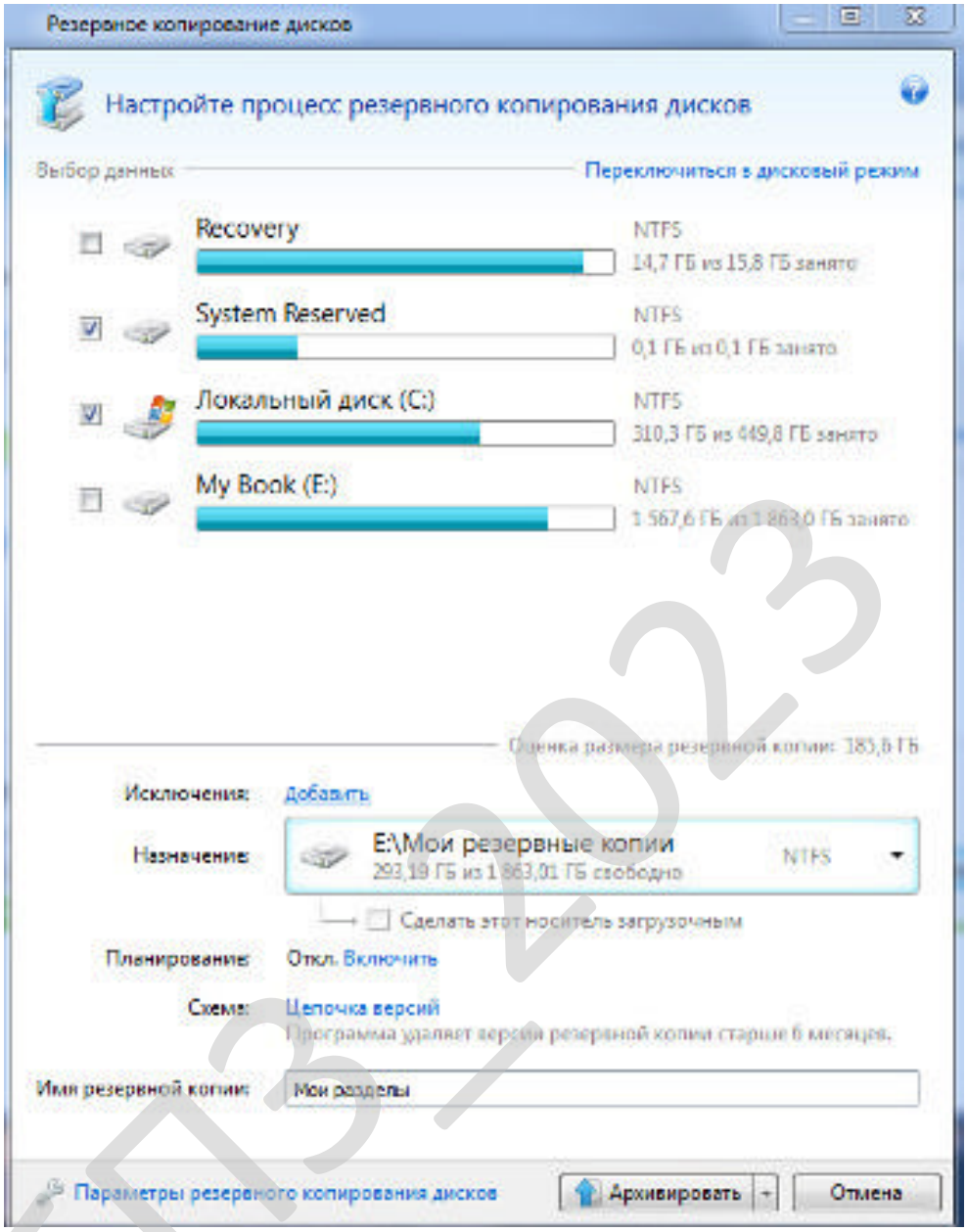

Рисунок 2.2 – Інтерфейс користувача Acronis True Image Home

Розглянемо, які типи «бекапів» існують.

Повну резервну копію ми вже розглядали. Це весь набір файлів, що зберігаються, або образ логічного диска «як є». Диференціальна (різницева) резервна копія містить тільки файли, змінені з моменту останнього повного резервного копіювання. Щоб відновити найбільш актуальну версію всіх даних, будуть потрібні одна повна й одна диференціальна резервні копії. Якщо ж диференціальних копій декілька, то можна вибирати різні версії змінених файлів.

Особливо корисно диференціальне резервування у випадку зараження диска вірусами, адже ніколи точно не знаєш, у який саме момент це відбулися і які файли постраждали. Нарешті, інкрементне резервування має на увазі додавання до архіву змінених файлів. Відновлення з інкрементного архіву також вимагає наявності хоча б однієї повної резервної копії.

Гнучкі налаштування Acronis True Image Home дозволяють скорочувати час створення резервних копій шляхом видалення «сміттєвих» (марних для архіву) файлів начебто файлу підкачування, дампів пам'яті або файлу з останнім сеансом гибернації. Архів може зберігатися в стислому виді, причому ступінь стиску припустимо встановлювати вручну. Природно, більший стиск споживає більше ресурсів процесора, але заощаджує час на створення й розгортання резервної копії. На архів може бути заданий пароль, що при бажанні буде продубльований відразу на декількох носіях. Загалом, складно уявити собі, які запити користувача не задовольнив би набір таких можливостей.

Особливою функцією Acronis True Image Home є «пісочниця» Try&Decide. У цьому режимі всі сторонні програми легко випробувати в спеціальному віртуальному середовищі. І якщо з'ясується, що яких-небудь ушкоджень не виникає, то ці програми можна буде застосовувати до реальної системи. Така функція вкрай корисна при тестуванні невідомого ПЗ, відкритті файлів із сумнівних джерел і інших потенційно небезпечних операціях.

Не обійшовся Acronis True Image Home і без власного сервісу онлайн-зберігання даних. Він платний, і до нього повною мірою ставляться всі міркування про надійність і безпеку, висловлені раніше відносно хмарних сервісів.

Norton Ghost

Даний пакет є одним з найстарших на ринку ПЗ, призначеного для резервного копіювання. Він напевно знаком досвідченим користувачам. Як і попередній продукт, Norton Ghost дозволяє створювати резервні копії всіх даних комп'ютера, вибраних папок і файлів, а також точки відновлення Windows і резервних копій, формованих іншими додатками.

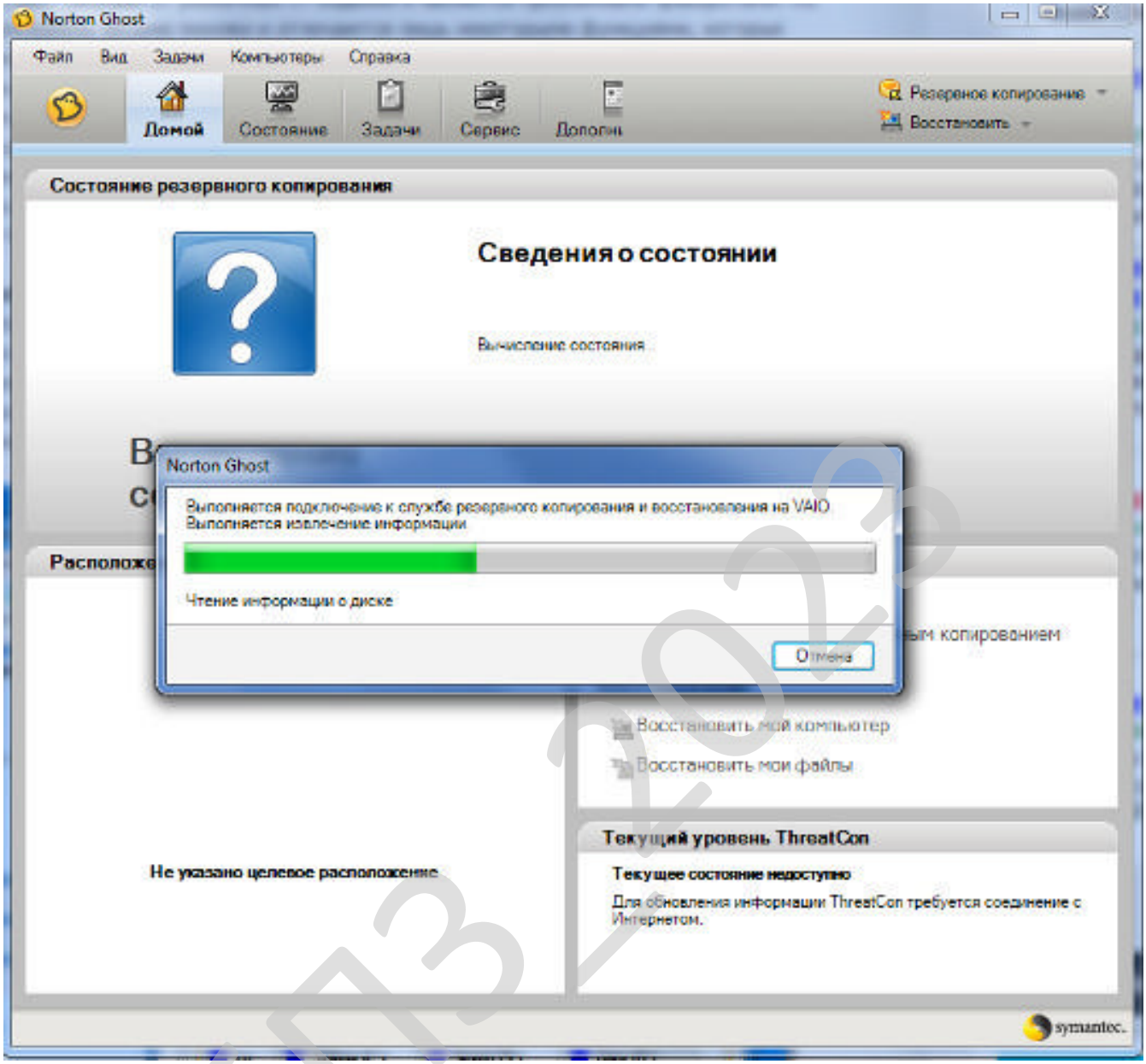

Рисунок 2.3 – Інтерфейс користувача Norton Ghost

При першому запуску програма звертається до архівів резервних копій, уже створеним на комп'ютері сторонніми засобами. Наприклад, у моєму випадку це були резервні копії ПЗ Sony VAIO.

Спосіб резервного копіювання, при якому створюється образ диска, мало чим відрізняється від пропонованого Acronis True Image Home. Також доступний стиск архіву зі ступенем ледве більше високої, чим у випадку з Acronis True Image Home. Це може виявитися корисним для стиснутих у дисковому просторі

користувачів. Втім, не варто забувати, що стискати можна тільки те, що ще не було стисле, наприклад мультимедійні файли практично не піддаються подальшому стиску.

Їсти можливість додавати до повного образа інкрементний або диференціальний архів, щоб скоротити час резервного копіювання. При копіюванні окремих файлів і папок дозволяється включити фільтр, що дозволяє виділити файли певного типу (наприклад, тільки текстові документи або графічні файли) і не допускає в архів системи «зайві» файли начебто вже зазначеного файлу підкачування.

Програма ця платна, поширюється з 30-денним пробним періодом, протягом якого накладається ряд обмежень. Так, неможливо робити повний образ жорсткого диска й створювати завантажувальний диск аварійного відновлення.

Існує також портативна (не потребуючої установки) версія програми, інтерфейс якої до болю знаком усім, хто працював з Norton Ghost у середовищі MS-DOS.

Paragon Drive Backup

Ця програма орієнтована в основному на досвідчених користувачів. Функціонал її аналогічний тому, що є в Acronis True Image Home, однак відсутні «дружні користувачеві» режими начебто «майстра». Як і у випадку з попередніми продуктами, дане рішення підтримує копіювання цілих дисків і окремих файлів і папок.

Робота з образами реалізована на дуже серйозному рівні – можна навіть окремо копіювати завантажувальну область диска. Природно, допускається стиск образа, однак при максимальному режимі пристойно «завантажуються» комп'ютерні ресурси й сама процедура відбувається досить повільно. На додаток до повних копій можна робити диференціальні й інкрементні архіви точно так само, як і за допомогою ПЗ Norton Ghost і Acronis True Image Home.

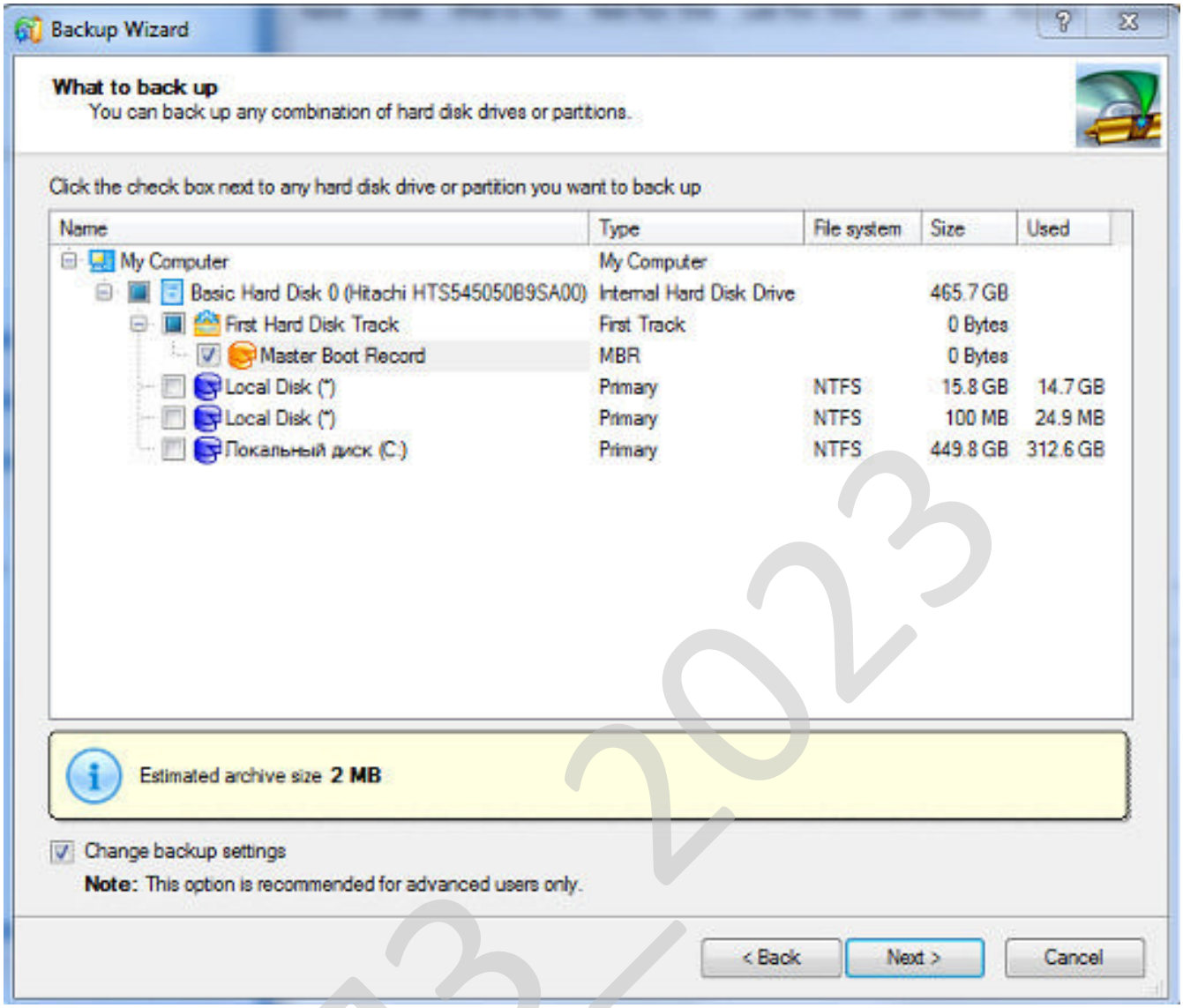

Рисунок 2.4 – Інтерфейс користувача Paragon Drive Backup

Резервне копіювання файлів здійснюється аналогічно тому, як це робить програма Norton Ghost, зокрема, підтримуються фільтри по типах файлів. Правда, відсутні групові фільтри, тому у випадку, коли потрібно, наприклад, зберегти тільки графічні файли, прийде вручну відзначати всі необхідні розширення. Підтримуються не тільки запис архіву на жорсткий диск, але і його відправлення на FTP-Сервер, у мережну локацію, а також «прожиг» на записуваний компакт-диск, DVD або Blu-ray. Уже створений архів можна перевірити на цілісність за допомогою убудованих засобів.

Цікавою функцією Paragon Drive Backup є створення «архівної капсули» (Backup Capsule) – особливого логічного розділу жорсткого диска. Його рекомендується розміщати після основних системних розділів. Він служить для збереження всього робочого програмного середовища комп'ютера. Після установки «капсули» з'являється можливість завантажувати комп'ютер з її, а не з основного розділу. Система, завантажена з «капсули», служить «полігоном» або «пісочницею», у якій припустимо робити потенційно небезпечні дії, наприклад, відкривати сумнівні файли або навіть «експериментувати» з вірусами. Зміни, які користувач зробила в «капсулі», можна або застосувати до «реального» системі, або відхилити.

Програма має 30-денний пробний період, протягом якого не можна створювати диск аварійного відновлення. Потім від її використання прийде відмовитися або оплатити повну версію.

HDClone

Це невелика утиліта з «суворим» інтерфейсом, що напевно прийдеться до смаку тим, хто вважає себе професіоналом у комп'ютерних технологіях. Ніяких графічних надмірностей, всі тільки по справі. Програма вміє лише створювати повні посекторні образи логічних дисків і відновлювати дані з них. При запуску можна вибрати режим роботи: або створення завантажувального диска, або копіювання безпосередньо в середовищі Windows.

В утиліту убудований тест швидкості, що дозволяє заздалегідь оцінити час резервного копіювання, а також що наочно демонструє переваги застосування багатодискових масивів. Програма вміє швидко стискати образи дисків, причому як на льоту, так і вже створені з її допомогою незжаті резервні копії, дефрагментувати розділи (це збільшить швидкість роботи системи після відновлення з архіву), «обрізати» образи розділів так, щоб виключати незайняте місце. При цьому образ диска можна шифрувати з використанням AES-алгоритму й установлювати пароль на доступ до нього.

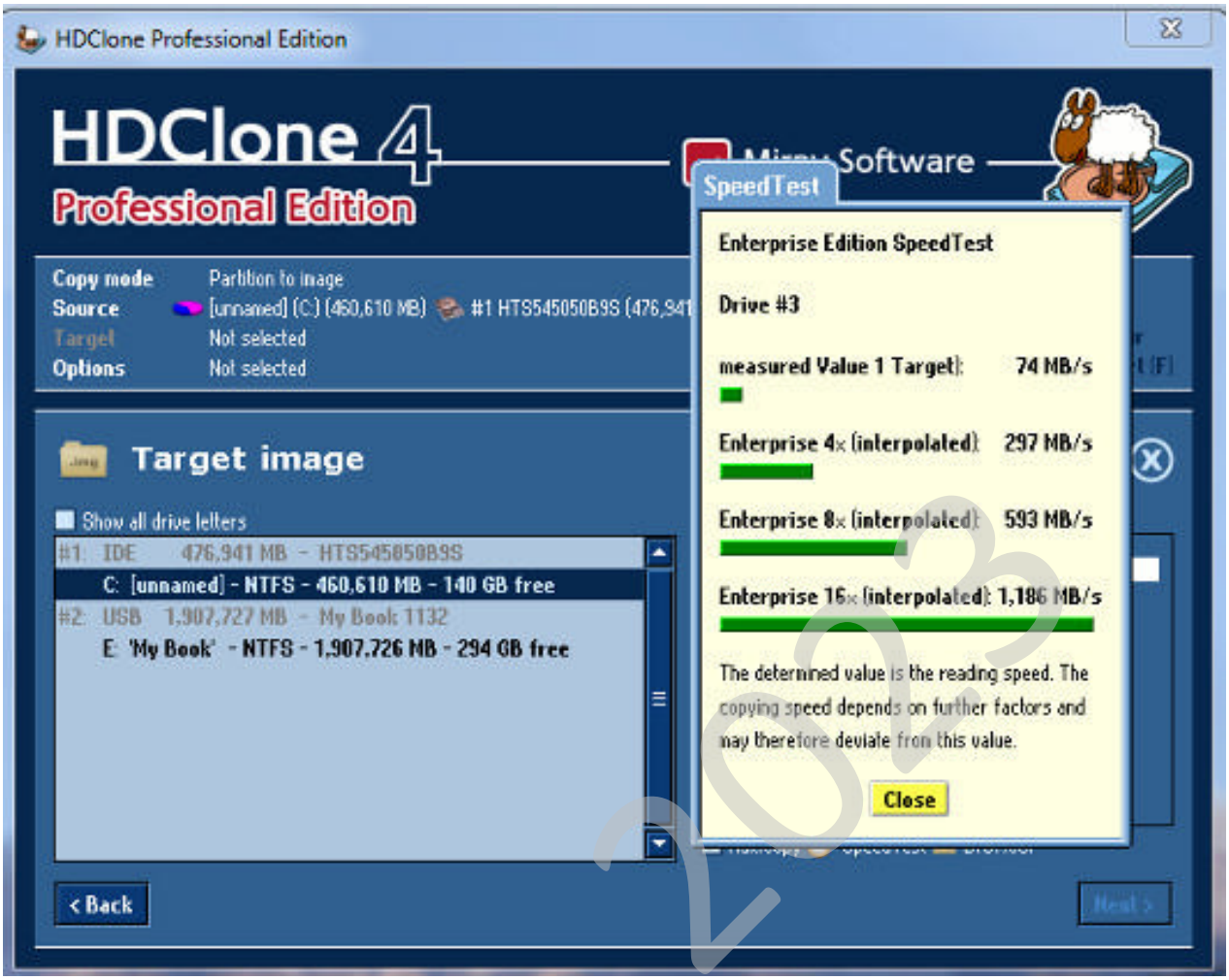

Рисунок 2.5 – Інтерфейс користувача HDClone

Як бачимо, функціональність програми орієнтована на тих користувачів, для яких резервне копіювання – частина щоденної рутинної роботи, наприклад на системних адміністраторів. Домашнім же користувачам програма може бути корисна тоді, коли потрібен простий інструмент для швидкого копіювання всіх даних комп'ютера.

Програма доступна в п'яти версіях – від безкоштовної до корпоративної. Відповідно змінюється й набір підтримуваних функцій, технологій і професійного встаткування (наприклад, з SCSI-дисками утиліта працює починаючи із професійної версії). У випадку безкоштовної версії дозволяється створювати тільки образи жорсткого диска цілком і не підтримуються компресія, шифрування й інші функції.

Засоби, убудовані в ОС Windows, цілком можуть задовольнити потреби більшості користувачів. Стандартна утиліта в стані зробити повну резервну копію системи, а також копіювати робочі каталоги, у тому числі за розкладом. Якщо потрібно більше гнучке налаштування архіву або особливі функції начебто «пісочниці», то вибір в основному буде між Paragon Drive Backup і Acronis True Image Home. Функціональність цих програм дуже схожа, і вона фактично дублюється, тому перевагу, як правило, буде залежати від поточної вартості програм. Ніколи лідируючий на ринку ПЗ для резервного копіювання, пакет Norton Ghost виглядає зараз декілька блідо на тлі своїх конкурентів, а рішення HDClone можна охарактеризувати як дуже вузькоспеціалізоване.

2.2 Обґрунтування вибору засобів для побудови системи 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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи стиснення та резервного дублювання даних.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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 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 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

Методи стиску даних мають досить довгу історію розвитку, що почалася задовго до появи першого комп'ютера. Стиск інформації – проблема, що має досить давню історію, набагато більше давню, ніж історія розвитку обчислювальної техніки, що (історія) звичайно йшла паралельно з історією розвитку проблеми кодування й шифровки інформації. Всі алгоритми стиску оперують вхідним потоком інформації, мінімальною одиницею якої є біт, а максимальної – декілька біт, байт або декілька байт. Метою процесу стиску, як правило, є одержання більше компактного вихідного потоку інформаційних одиниць із деякого споконвічно некомпактного вхідного потоку за допомогою деякого їхнього перетворення. Основними технічними характеристиками процесів стиску й результатів їхньої роботи є:

– ступінь стиску (compress rating) або відношення (ratio) обсягів вихідного й результуючого потоків;

– швидкість стиску – час, затрачуваний на стиск деякого обсягу інформації вхідного потоку, до одержання з нього еквівалентного вихідного потоку;

– якість стиску – величина, що показує на скільки сильно впакований вихідний потік, за допомогою застосування до нього повторного стиску по цьому ж або іншому алгоритмі.

Існує кілька різних підходів до проблеми стиску інформації. Одні мають досить складну теоретичну математичну базу, інші засновані на властивостях інформаційного потоку й алгоритмічно досить прості. Будь-який спосіб підхід і алгоритм, що реалізує стиск або компресію даних, призначений для зниження обсягу вихідного потоку інформації в бітах за допомогою її оборотного або

необоротного перетворення. Тому, насамперед, за критерієм, пов'язаному з характером або форматом даних, всі способи стиску можна розділити на дві категорії: оборотний і необоротний стиск.

Під необоротним стиском мають на увазі таке перетворення вхідного потоку даних, при якому вихідний потік, заснований на певному форматі інформації, представляє, з деякого погляду, досить схожий по зовнішніх характеристиках на вхідний потік об'єкт, однак відрізняється від нього обсягом. Ступінь подібності вхідного й вихідного потоків визначається ступенем відповідності деяких властивостей об'єкта (тобто стислої й незжатої інформації, відповідно до деякого певного формату даних), що представляється даним потоком інформації. Такі підходи й алгоритми використовуються для стиску, наприклад, даних растрових графічних файлів з низьким ступенем повторюваності байтів у потоці. При такому підході використовується властивість структури формату графічного файлу й можливість представити графічну картинку приблизно схожу по якості відображення (для сприйняття людським оком) декількома (а точніше n) способами. Тому, крім ступеня або величини стиску, у таких алгоритмах виникає поняття якості, тому що вихідне зображення в процесі стиску змінюється, то під якістю можна розуміти ступінь відповідності вихідного й результуючого зображення, оцінювана суб'єктивно, виходячи з формату інформації. Для графічних файлів така відповідність визначається візуально, хоча є й відповідні інтелектуальні алгоритми й програми. Необоротний стиск неможливо застосовувати в областях, у яких необхідно мати точну відповідність інформаційної структури вхідного й вихідного потоків. Даний підхід реалізований у популярних форматах подання відео й фото інформації, відомих як JPEG і JFIF алгоритми й JPG і JIF формати файлів.

Оборотний стиск завжди приводить до зниження обсягу вихідного потоку інформації без зміни його інформативності, тобто – без втрати інформаційної структури. Більше того, з вихідного потоку, за допомогою що відновлює або декомпресуючого алгоритму, можна одержати вхідний, а процес відновлення

називається декомпресією або розпакуванням, і тільки після процесу розпакування дані придатні для обробки відповідно до їхнього внутрішнього формату.

В оборотних алгоритмах кодування як процес можна розглядати зі статистичної точки зору, що ще більш корисно, не тільки для побудови алгоритмів стиску, але й для оцінки їхньої ефективності. Для всіх оборотних алгоритмів існує поняття вартості кодування. Під вартістю кодування розуміється середня довжина кодового слова в бітах. Надмірність кодування дорівнює різниці між вартістю й ентропією кодування, а гарний алгоритм стиску завжди повинен мінімізувати надмірність (нагадаємо, що під ентропією інформації розуміють міру її невпорядкованості.). Фундаментальна теорема Шеннона про кодування інформації говорить про те, що "вартість кодування завжди не менше ентропії джерела, хоча може бути як завгодно близька до неї". Тому, для будь-якого алгоритму, завжди є деяка межа ступеня стиску, обумовлена ентропією вхідного потоку.

Перейдемо тепер безпосередньо до алгоритмічних особливостей оборотних алгоритмів і розглянемо найважливіші теоретичні підходи до стиску даних, пов'язані з реалізацією систем, що кодують, і способи стиску інформації.

Стиск способом кодування серій

Найбільш відомий простий підхід і алгоритм стиску інформації оборотним шляхом – це кодування серій послідовностей (Run Length Encoding – RLE). Суть методів даного підходу складається в заміні ланцюжків або серій повторюваних байтів або їхніх послідовностей на один байт, що кодує, і лічильник числа їхніх повторень. Проблема всіх аналогічних методів полягає лише у визначенні способу, за допомогою якого алгоритм, що розпаковує, міг би відрізнити в результуючому потоці байтів кодовану серію від інших – некодованих послідовностей байтів. Рішення проблеми досягається звичайно проставлянням міток на початку кодованих ланцюжків. Такими мітками можуть бути, наприклад, характерні значення бітів у першому байті кодованої серії, значення

першого байта кодованої серії й т.п. Дані методи, як правило, досить ефективні для стиску растрових графічних зображень (BMP, PCX, TIF, GIF), тому що останні містять досить багато довгих серій повторюваних послідовностей байтів. Недоліком методу RLE є досить низький ступінь стиску або вартість кодування файлів з малим числом серій і, що ще гірше – з малим числом повторюваних байтів у серіях.

Стиск без застосування методу RLE

Процес стиску даних без застосування методу RLE можна розбити на два етапи: моделювання (modelling) і, властиво, кодування (encoding). Ці процеси і їхні алгоритми, що реалізують, досить незалежні й різні.

Процес кодування і його методи

Під кодуванням звичайно розуміють обробку потоку символів (у нашому випадку байтів або напівбайтів) у деякому алфавіті, причому частоти появи символів у потоці різні. Метою кодування є перетворення цього потоку в потік біт мінімальної довжини, що досягається зменшенням ентропії вхідного потоку шляхом обліку частот символів. Довжина коду, що представляє символи з алфавіту потоку повинна бути пропорційна обсягу інформації вхідного потоку, а довжина символів потоку в бітах може бути не кратна 8 і навіть змінної. Якщо розподіл імовірностей частот появи символів з алфавіту вхідного потоку відомо, то можна побудувати модель оптимального кодування. Однак, через існування величезного числа різних форматів файлів завдання значно ускладнюється тому що розподіл частот символів даних заздалегідь невідомо. У такому випадку, у загальному виді, використовуються два підходи.

Перший полягає в перегляді вхідного потоку й побудові кодування на підставі зібраної статистики (при цьому потрібно два проходи по файлі – один для перегляду й збору статистичної інформації, другий – для кодування, що декілька обмежує сферу застосування таких алгоритмів, тому що, таким чином, виключається можливість однопрохідного кодування "на льоту", застосовуваного в телекомунікаційних системах, де й обсяг даних, часом, не відомий, а їхня

повторна передача або розбір може зайняти невиправдано багато часу). У такому випадку, у вихідний потік записується статистична схема використаного кодування. Даний метод відомий як статичне кодування Хаффмена.

Другий метод – метод адаптивного кодування (adaptive coder method). Його загальний принцип полягає в тому, щоб міняти схему кодування залежно від характеру змін вхідного потоку. Такий підхід має однопрохідний алгоритм і не вимагає збереження інформації про використане кодування в явному виді. Адаптивне кодування може дати більший ступінь стиску, у порівнянні зі статичним, оскільки більш повно враховуються зміни частот вхідного потоку. Даний метод відомий як динамічне кодування Хаффмена.

У статичному кодуванні Хаффмена вхідним символам (ланцюжкам бітів різної довжини) ставляться у відповідність ланцюжка бітів, також, змінної довжини – їхньої коди. Довжина коду кожного символу береться пропорційної двійковому логарифму його частоти, узятому зі зворотним знаком. А загальний набір всіх різних символів, що зустрілися, становить алфавіт потоку. Це кодування є префіксним, що дозволяє легко його декодувати результативний потік, тому що, при префіксному кодуванні, код будь-якого символу не є префіксом коду ніякого іншого символу – алфавіт унікальний.

При використанні адаптивного кодування Хаффмена ускладнення алгоритму складається в необхідності постійного коректування дерева й кодів символів основного алфавіту відповідно до статистики вхідного потоку, яка змінюється.

Методи Хаффмена дають досить високу швидкість і помірковано гарну якість стиску. Ці алгоритми давно відомі й широко застосовується як у програмні (усілякі компресори, архіватори й програми резервного копіювання файлів і дисків), так і в апаратні (системи стиску "прошиті" у модеми й факси, сканери) реалізаціях.

Однак, кодування Хаффмена має мінімальну надмірність за умови, що кожний символ кодується в алфавіті коду символу окремим ланцюжком із двох

біт – {0, 1}. Основним же недоліком даного методу є залежність ступеня стиску від близькості ймовірностей символів до 2 у деякому негативному ступені, що пов'язане з тим, що кожний символ кодується цілим числом біт. Так при кодуванні потоку із двохсимвольним алфавітом стиск завжди відсутнє, тому що незважаючи на різні ймовірності появи символів у вхідному потоці алгоритм фактично зводить їх до 1/2.

Дана проблема, як правило, вирішується шляхом введення в алфавіт вхідного потоку нових символів виду "ab", "abc",. . . і т.п., де a, b, c – символи первинного вихідного алфавіту. Такий процес називається сегментацією або блокуванням вхідного потоку. Однак, сегментація не дозволяє повністю позбутися від втрат у стиску (вони лише зменшуються пропорційно розміру блоку), але приводить до різкого росту розмірів дерева кодування, і, відповідно, довжині коду символів вторинних алфавітів. Так, якщо, наприклад, символами вхідного алфавіту є байти зі значеннями від 0 до 255, то при блокуванні по двох символах ми одержуємо 65536 символів (різних комбінацій) і стільки ж листів дерева кодування, а при блокуванні по трьох – 16777216! Звичайно, при такому ускладненні, відповідно зростуть вимоги й до пам'яті й вчасно побудови дерева, а при адаптивному кодуванні – і вчасно відновлення дерева, що приведе до різкого збільшення часу стиску. Навпроти, у середньому, втрати складуть 1/2 біта на символ при відсутності сегментації, і 1/4 або 1/6 біта відповідно при її наявності, для блоків довжиною 2 і 3 біти.

Арифметичне кодування

Зовсім інше рішення пропонує т.зв. арифметичне кодування. Арифметичне кодування є методом, що дозволяє впаковувати символи вхідного алфавіту без втрат за умови, що відомо розподіл частот цих символів і є найбільш оптимальним, тому що досягається теоретична границя ступеня стиску.

Передбачувана необхідна послідовність символів, при стиску методом арифметичного кодування, розглядається як деякий двійковий дріб з інтервалу [0, 1). Результат стиску представляється, як послідовність двійкових цифр із

запису цього дробу. Ідея методу полягає в наступному: вихідний текст розглядається як запис цього дробу, де кожний вхідний символ є "цифрою" з вагою, пропорційною ймовірності його появи. Цим пояснюється інтервал, що відповідає мінімальній і максимальній ймовірностям появи символу в потоці.

При розробці цього методу виникають дві проблеми: по-перше, необхідна арифметика із плаваючою точкою, теоретично, необмеженої точності, і, по-друге, – результат кодування стає відомий лише при закінченні вхідного потоку. Однак, подальші дослідження показують, що можна практично без втрат обійтися цілочисельною арифметикою невеликої точності (16-32 розряду), а також домогтися інкрементальної роботи алгоритму: цифри коду можуть видаватися послідовно в міру читання вхідного потоку при обмеженні числа символів вхідного ланцюжка яким або розумним числом.

Моделі вхідного потоку

Кодування являє собою лише частина процесу впакування. Як було показано, арифметичне кодування має мінімальну надмірність при заданому розподілі символів вхідного потоку. Але який алфавіт вибрати і яким відповідним розподілом скористатися? Відповіді на ці питання дає побудова моделі вхідного потоку, що представляє собою деякий спосіб визначення можливого розподілу ймовірностей появи кожного чергового символу в потоці. Кожного, оскільки статичні моделі (у які розподіл приймається незмінним), у більшості випадків, не дають максимальної якості стиску. Набагато більший інтерес представляють так звані адаптивні моделі, що враховують поточний контекст потоку. Такі моделі дозволяють будувати швидкі однопрохідні алгоритми стиску, не потребуючих апріорних знань про вхідний потік даних і які будують розподіл "на льоту". В окрему групу виділяють також клас "локально адаптивних" алгоритмів, що віддають при побудові розподілу перевага деяким особливим, наприклад, останнім символам, що надійшли.

Можливі різні підходи до цієї проблеми: найпростіший з них – збір статистики появи кожного символу незалежно від інших (моделювання джерелом

Бернуллі, при якому ймовірність появи наступного символу не залежить від того, які символи зустрілися перед ним). Можливо, також і використання марковських моделей: збір статистики появи кожного символу в які виробляється з урахуванням деякої кількості попередніх символів, що з'являлися (у марковському джерелі першого порядку ймовірність появи символу залежить тільки від одного останнього символу, другого – від двох і т.д.). Марковські моделі можуть давати більше точну картину джерела, однак число станів у них більше, відповідно до більших буде обсяг збережених таблиць частот. Крім того, при використанні кодування Хаффмена вони можуть навіть погіршити якість стиску, оскільки породжувані ними ймовірності звичайно гірше наближаються ступенями 1/2.

Кодування сортуванням

Тут не можна не згадати простий і досить ефективний метод кодування джерела з невідомим розподілом частот, відомий як стиск за допомогою "стопки книг" або як стиск сортуванням або хешуванням. Ідея методу полягає в наступному: нехай алфавіт джерела складається з N символів з номерами 1, 2,..., N. Алгоритм, що кодує, зберігає послідовність символів, що представляє собою деяку перестановку символів у послідовності первинного вхідного алфавіту. При надходженні на вхід деякого символу c, що має в цій переставленій послідовності номер i, що кодує алгоритм записує код цього символу (наприклад, монотонний код). Потім символ, що надійшов, переставляється в початок послідовності й номера всіх символів, що знаходяться перед c, збільшуються на 1. Таким чином, що найбільше часто зустрічаються символи будуть переходити в початок списку й мати більше короткі коди, що у свою чергу знизить обсяг вихідного потоку при їхньому записі як символи вихідного потоку.

Двоступінчасте кодування. Алгоритм Лемпеля-Зіва

Всі розглянуті вище методи й моделі кодування припускали в якості вхідних дані ланцюжки символів (тексти) у деякому кінцевому алфавіті. При цьому залишався відкритим питання про зв'язок цього вхідного алфавіту

алгоритму, що кодує, з даному, підлягаючому впакуванню (звичайно також представленими у вигляді ланцюжків в алфавіті (при байтової організації звичайно складає з 256 різних символів – значень байт).

У найпростішому випадку для кодування як вхідний алфавіт можна використовувати саме ці символи (байти) вхідного потоку. Саме так працює метод squashing програми PKPAK (використане статичне кодування Хаффмена й двопрохідний алгоритм). Ступінь стиску при цьому відносно невелика – для текстових файлів порядку 50%. Набагато більшого ступеня стиску можна домогтися при виділенні із вхідного потоку повторюваних ланцюжків – блоків і кодування посилань на ці ланцюжки з побудовою хеш таблиць від першого до n-го рівня.

Метод, про який й піде мова, належить Лемпелю й Зиву, і звичайно називається LZ-compression. Суть його полягає в наступному: пакувальник постійно зберігає деяку кількість останніх оброблених символів у буфері. У міру обробки вхідного потоку знову, що надійшли символи, попадають у кінець буфера, зрушуючи попередні символи й витісняючи самі старі. Розміри цього буфера, називаного також ковзним словником (sliding dictionary), варіюються в різних реалізаціях систем, що кодують. Експериментальним шляхом установлено, що програма LHArc використовує 4-кілобайтний буфер, LHA і PkZip – 8-мі, а ARJ – 16-кілобайтний.

Потім, після побудови хеш таблиць алгоритм виділяє (шляхом пошуку в словнику) самий довгий початковий підрядок вхідного потоку, що збігається з одним з підрядків у словнику, і видає на вихід пари (length, distance), де length – довжина знайденого в словнику підрядку, а distance – відстань від її до вхідний підрядку (тобто фактично індекс підрядку в буфері, віднятий з його розміру). У випадку, якщо такий підрядок не знайдений, у вихідний потік просто копіюється черговий символ вхідного потоку.

У первісній версії алгоритму пропонувалося використовувати найпростіший пошук по всьому словнику. Час стиску при такій реалізації було

пропорційно добутку довжини вхідного потоку на розмір буфера, що зовсім непридатно для практичного використання. Однак, надалі, було запропоновано використовувати двійкове дерево й хешування для швидкого пошуку в словнику, що дозволило на порядок підняти швидкість роботи алгоритму.

Таким чином, алгоритм Лемпеля-Зіва перетворить один потік вихідних символів у два паралельних потоки довжин і індексів у таблиці (length + distance). Очевидно, що ці потоки є потоками символів із двома новими алфавітами, і до них можна застосувати один зі згадуваних вище методів (RLE, кодування Хаффмена або арифметичне кодування).

Так ми приходимо до схеми двоступінчастого кодування – найбільш ефективної із практично використовуваних у цей час. При реалізації цього методу необхідно домогтися погодженого виводу обох потоків в один файл. Ця проблема звичайно вирішується шляхом почергового запису кодів символів з обох потоків.

Алгоритм Лемпеля-Зіва-Велча (Lempel-Ziv-Welch – LZW)

Даний алгоритм відрізняють висока швидкість роботи як при впакуванні, так і при розпакуванні, досить скромні вимоги до пам'яті й проста апаратна реалізація. Недолік – низький ступінь стиску в порівнянні зі схемою двоступінчастого кодування. Припустимо, що в нас є словник, що зберігає рядки тексту й містить порядку від 2-х до 8-мі тисяч пронумерованих гнізд. Запишемо в перші 256 гнізд рядка, що складаються з одного символу, номер якого дорівнює номеру гнізда. Алгоритм переглядає вхідний потік, розбиваючи його на підрядки й додаючи нові гнізда в кінець словника. Прочитаємо кілька символів у рядок s і знайдемо в словнику рядок t – самий довгий префікс s. Нехай він знайдений у гнізді з номером n. Виведемо число n у вихідний потік, перемістимо покажчик вхідного потоку на length(t) символів уперед і додамо в словник нове гніздо, що містить рядок t+c, де c – черговий символ на вході (відразу після t). Алгоритм перетворить потік символів на вході в потік індексів осередків словника на виході. При розмірі словника в 4096 гнізд можна передавати 12 біт на кожний індекс. Кожний розпізнаний ланцюжок додає в словник одне гніздо. При

переповненні словника пакувальник може або припинити його заповнення, або очистити (повністю або частково).

При практичній реалізації цього алгоритму варто врахувати, що будь-яке гніздо словника, крім найперші, утримуючі односимвольні ланцюжки, зберігає копію деякого іншого гнізда, до якої в кінець приписаний один символ. Внаслідок цього можна обійтися простою обліковою структурою з одним зв'язком.

Звичайно, для системи стиску інформації гарний алгоритм – першочерговий, але не єдине болюче питання. Кінцевому користувачеві, як правило, немає справи до принципів організації функціонування й внутрішньої структури використовуваних їм програм, аби тільки працювали якісно. Під якістю систем стиску прийнято розуміти кілька критеріїв, які визначаються застосуванням при її реалізації й конкретному використанні. Так більшість застосувань необоротного стиску лежить в області технології зберігання графічної інформації – картинки й відео, що, у свою чергу локалізує алгоритм у рамках одного файлу для одного стандарту й характеру вхідного потоку. Однак, на практиці, через економію місця на накопичувачах, виникає необхідність стиску будь-якого файлу (у тому числі й здійсненному модулі). Цю проблему вирішують пакувальники. І, нарешті, проблему стиску декількох файлів і навіть всіх файлів каталогів і дисків вирішують програми – архіватори. Помітимо, що в список можливих завдань архіваторів входить не тільки стиск/добування інформації файлів різних форматів, але й збереження дерева файлової системи, атрибутів файлів, їхніх імен, деякої інформації, що коментує, створення архівів, що саморозпаковуються, архівація зі збереженням кодів циклічного контролю помилок, для гарантії абсолютної відповідності витягнутих файлів вихідним файлам, шифровку даних архіву й архівація з паролем, забезпечення користувача зручним інтерфейсом і ін. Тому, архіватори є однією із самих складних систем програм. У цей час, до перерахованим вище завдань можна побажати наявності деяких необов'язкових, але зручних властивостей і можливостей. Це

конфігуруємість пакета, наявність розвиненого віконного інтерфейсу, а не інтерфейсу командного рядка, налаштовуємість на певний тип інформації, збереження параметрів у файлі архіву, створення багатотомних і/або архівів, що самовитягають, і ін. Все це бажано мати при малій довжині файлу архіватора. Між програмами пакувальниками й архіваторами звичайно не є принципових розходжень, однак, пакувальники впаковують інформацію одного файлу в один файл, а архіватори утворять один файл вихідного потоку, що, втім, може бути автоматично нарізаний на файли рівної довжини для запису на гнучкі диски. Окрему групу становлять програми пакувальники, що займаються стиском здійсненних модулів і дисків. Також упакування даних за допомогою алгоритмів стиску використовують системи резервного копіювання, стиску пристроїв (логічних дисків MS-DOS), факс-модемні драйвери й утиліти й ін.

У даний момент на ринку програмних продуктів і серверах програмного забезпечення можна зустріти досить велике число архівуючих і стискаючих утиліт, більшість із яких доступні для некомерційного використання. Така доступність, насамперед, пов'язана з тим, що кожна, навіть комерційна, програма має потребу у великому ринку користувачів. А оскільки формати файлів архіваторів і пакувальників, що навіть використовують той самий алгоритм, не однакові, то такі програми мимоволі конкурують за ринок користувачів. Із цим також зв'язано й те, що підтримка більше популярних форматів файлових архівів починає включатися в інші утиліти й програми й використовувані формати стають стандартними форматами архівів (zip, arj, rar, ha, pak, cab і ін.). Стандартний формат має на увазі виняткову легкість при пошуку програми, необхідної для добування файлів зі стислого стану й підтримку їхніми іншими програмами.

3.2 Розробка структурної схеми

Структурно система стиснення та резервного дублювання даних складається з наступних блоків:

1. Блок збереження даних:

– Створюйте резервні копії зображень, відеозаписів, фільмів, музики, документів і інших файлів.

– Встановіть автоматичний запуск резервного копіювання тоді, коли вам потрібно, навіть якщо в цей час ви не використовуєте комп'ютер.

– Використовуйте будь-які пристрої зберігання інформації, включаючи носії USB, зовнішні жорсткі диски, пристрої NAS і хмарне сховище в Інтернет.

– Не відволікайтеся на створення резервних копій – все відбувається у фоновому режимі.

2. Блок відновлення файлів:

– Якщо ви ненавмисно видалили потрібний файл, можливо переглянути історію збережень і відновити його.

– Відкрийте історію створення резервних копій, скористайтеся попереднім переглядом і виберіть ту версію файлу, що хочете відновити.

– Можливо одержати доступ до ваших файлів у будь-якому місці й у будь-який час через веб-браузер або через мобільний додаток, або хмарне сховище в Інтернет.

– Відновлюйте окремі файли з образів у хмарне сховище в Інтернет.

3. Блок створення повного образа диска ПК:

– Створіть резервну копію диска цілком для наступного відновлення – збережете не тільки файли, але й точну конфігурацію ПК, включаючи операційну систему, додатки й настроювання.

– Збережіть повний образ диска в хмарне сховище в Інтернет.

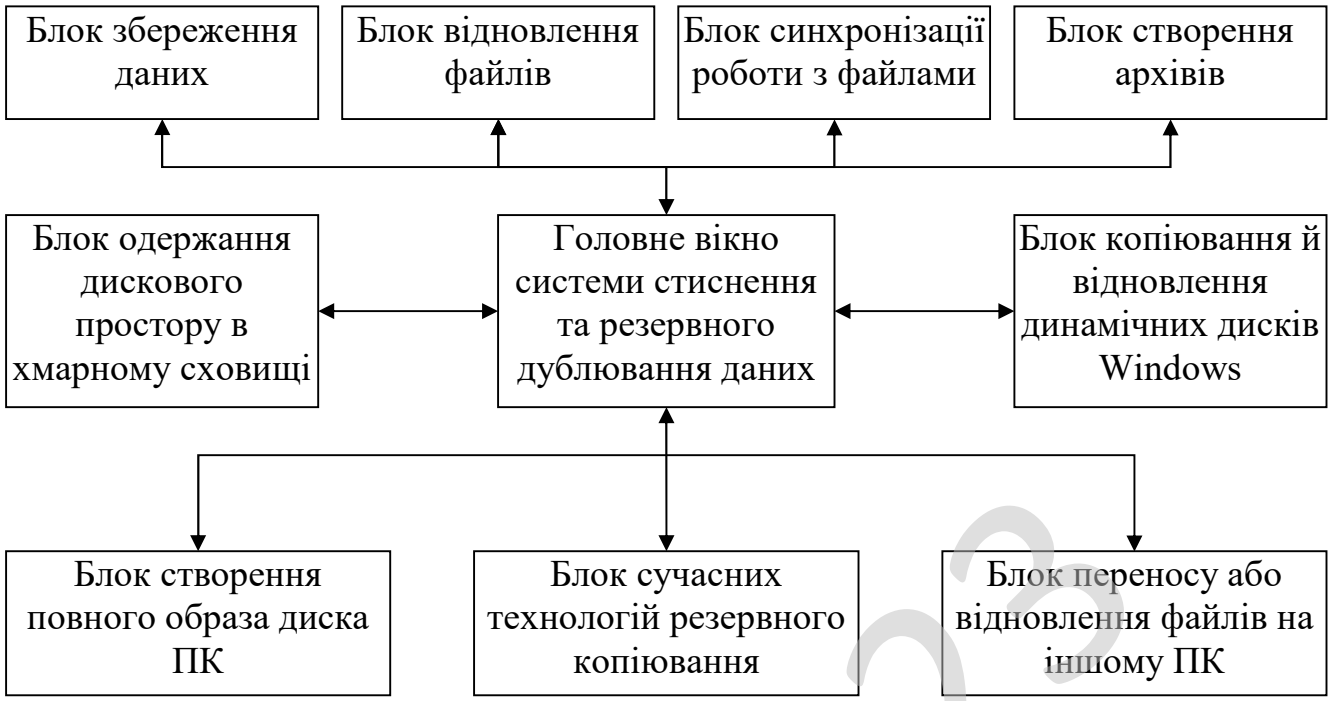

Рисунок 3.1 – Структурна схема системи

4. Блок одержання дискового простору в хмарному сховищі:

– Зберігайте копії найбільш важливих файлів у нашім надійному й безпечному центрі зберігання даних.

– Можливо одержати доступ до ваших файлів у будь-якому місці й у будь-який час.

– Використовуйте загальний доступ до файлів – відправляйте безпечно посилання на вміст, розміщений у хмарному сховищі.

– Можливо використовувати до п'яти пристроїв (настільні ПК, ноутбуки, планшети, смартфони й т.д.) для роботи із хмарним сховищем в Інтернет.

– Синхронізуйте роботу з файлами на всіх своїх пристроях.

5. Блок сучасних технологій резервного копіювання:

– Настройте створення інкрементних і диференціальних резервних копій так, щоб зберігати тільки зміни з моменту останнього резервного копіювання – так ви заощадите місце на диску.

– Перевіряйте нове встановлене ПЗ й відвідуйте будь-які веб-сайти без найменшого ризику – таку можливість дає технологія Try&Decide, що дозволяє вибирати, які зміни ви хочете зберегти на вашім комп'ютері.

– Потужний механізм шифрування AES-256, що відповідає міжнародним стандартам, забезпечує цілісність і недоторканність ваших даних.

6. Блок синхронізації роботи з файлами:

– Обрані файли автоматично додаються на всі ваші ПК і мобільні пристрої щораз при внесенні змін.

7. Блок переносу або відновлення файлів на іншому ПК:

– Відновіть повний образ диска, включаючи операційну систему, додатки, файл і настроювання, на ПК іншого типу або моделі.

– Переносите файли з однієї системи Windows в іншу, у тому числі на нові ПК.

– Відновіть повну конфігурацію системи в середовищі передвстановки Windows (WinPE).

8. Блок копіювання й відновлення динамічних дисків Windows:

– Копіюйте динамічні диски Windows.

– Відновлюйте динамічні томи на жорсткі диски "з нуля" або на попередньо настроєні динамічні диски.

9. Блок створення архівів.

3.3 Розробка функціональної схеми

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

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

Функціональна схема складається з наступних блоків:

– Блок стиску інформації.

– Блок інтерфейсу користувача програми.

– Блок резервування.

– Блок швидкого доступу до елементів програми.

– Блок операцій.

– Блок параметрів.

– Довідка.

Розглянемо ці функціональні елементи більш детально.

Блок стиску включає в себе наступні функціональні можливості:

– створення багатотомних архівів з можливістю завдання довільного розміру тому;

– створення архівів, що саморозпаковуються – SFX-архівів;

– створення багатотомних SFX-архівів;

– автоматичне видалення файлів після архівації;

– архівування каталогів і дисків повністю зі збереженням атрибутів файлів;

– приміщення в архів авторських коментарів;

– паролювання доступу до архіву;

– підтримка захищеного режиму (DPMI, VCPI), розширеної й розширюваної пам'яті;

– впровадження в архів циклічних кодів помилок, що дозволяють відновлювати ушкоджені архіви;

– видача докладної інформації із закінчення процесу архівації й на вимогу (коефіцієнт стиску, приблизний час стиску/розпакування, розміри файлів і т.п.).

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 точок, 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 точок 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 великий обсяг інформації – практично весь том, на якому встановлена операційна система. І процедура відновлення системи стає більше складною.

Рисунок 3.2 – Функціональна схема системи

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 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.

Типи архівації

Всі файли мають атрибути, серед яких є атрибут "архівний". При резервному копіюванні (архівації) програма враховує його стан.

Якщо атрибут установлений, файл (папка) архівується, якщо не встановлений, не архівується.

Після архівації файлу (папки) із установленим атрибутом "архівний", атрибут може бути скинутий або залишитися активним, залежно від обраного вами типу архівації.

Програма підтримує п'ять типів архівації:

– Звичайний архів – архівуються всі виділені файли й папки. Атрибут "архівний" скидається у всіх заархівіруваних файлів. Звичайний архів прискорює процес відновлення, тому що містить всі поточні версії файлів.

– Архів, що копіює – архівуються всі виділені файли й папки. Якщо якісь файли мають установлений атрибут "архівний", то він не скидається в процесі архівації. Такий тип архівації використовується, коли ви хочете швидко зберегти резервну копію тільки деяких файлів.

– Додатковий архів – архівуються тільки ті файли й папки, які були змінені з моменту останньої архівації, тобто із установленим атрибутом "архівний". Атрибут "архівний" скидається. Для повного відновлення системи будуть потрібні повний архів і всі додаткові архіви, створені після нього.

– Різницевий архів – архівуються тільки ті файли й папки, які були змінені з моменту останньої архівації, тобто із установленим атрибутом "архівний". Атрибут "архівний" не скидається. Для повного відновлення системи буде необхідні повний архів і останній різницевий архів.

– Щоденний архів – архівуються тільки файли й папки, створені або змінені в день архівації. Атрибут "архівний" залишається у файлів яким був до архівації.

Сполучення різних типів архівації дозволяє вам знайти "золоту середину" між часом копіювання й часом відновлення.

Звичайна + різницева:

– У понеділок виконується звичайна архівація, з вівторка по п'ятницю – різницева.

– Щораз архівуються всі зміни, що відбулися з понеділка.

– У випадку збою даних у п'ятницю вам доведеться відновити звичайний архів від понеділка й останній різницевий архів від четверга.

– Це вимагає більше часу на резервне копіювання, але менше – на відновлення.

Звичайна + додаткова:

– У понеділок виконується звичайна архівація, з вівторка по п'ятницю – додаткова.

– Щодня архівуються тільки змінені файли.

– У випадку збою даних у п'ятницю вам доведеться відновити звичайний архів від понеділка й послідовно кожний додатковий архів з вівторка по четвер.

– Цей підхід вимагає менше часу на резервне копіювання, але більше – на відновлення.

Звичайна + різницева + додаткова:

– Звичайно даний сценарій застосовується при двотижневому циклі резервного копіювання

– У понеділок виконується звичайна архівація, у вівторок, середовище, п'ятницю й суботу – різницева, а в четвер – додаткова.

– У випадку збою даних у суботу вам доведеться відновити звичайний архів від понеділка, додатковий архів від четверга й різницевий архів від п'ятниці.

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 магістерського проектування, наведена на рисунку 3.3.

Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ, далі можна перейти до перегляду журналу роботи ПЗ, роботи з архіву та в свою чергу до відновлення даних чи до переміщення файлів архіву.

З головного віка також можна перейти до резервування даних та далі з використанням бібліотеки резервування даних, початкових параметрів резервування, функцій резервування потрапити до обрання параметрів резервного копіювання.

Далі проходе перехід до складання шляхів даних для резервного копіювання, обрання каталогу для збереження та остаточне – резервне копіювання.

Рисунок 3.3 – Діаграма взаємодії процесів

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 ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ 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:

– Ініціалізація та виведення головного вікна ПЗ.

– Виведення початкового списку файлів та архівів.

– Чекання дій користувача.

– Запит створення архіву?

– Вибір алгоритму та параметрів архівування.

– Створення архіву та кодування файлу.

– Збереження архіву по встановленому шляху.

– Запит розархівування файлів?

– Декодування вибраних файлів.

– Збереження розархівованих файлів по встановленому шляху.

– Запит резервного копіювання?

– Вибір параметрів резервного копіювання.

– Вибір даних для резервного копіювання.

– Вибір каталогу для збереження даних.

– Резервне копіювання вибраних даних.

– Запит відновлення даних?

– Вибір та відновлення даних з резервних архівів.

– Сигнал WM_CLOSE (запит).

На рисунку 4.2 наведено блок-схему підпрограми архівації даних. Її робота складається з виконання наступних кроків:

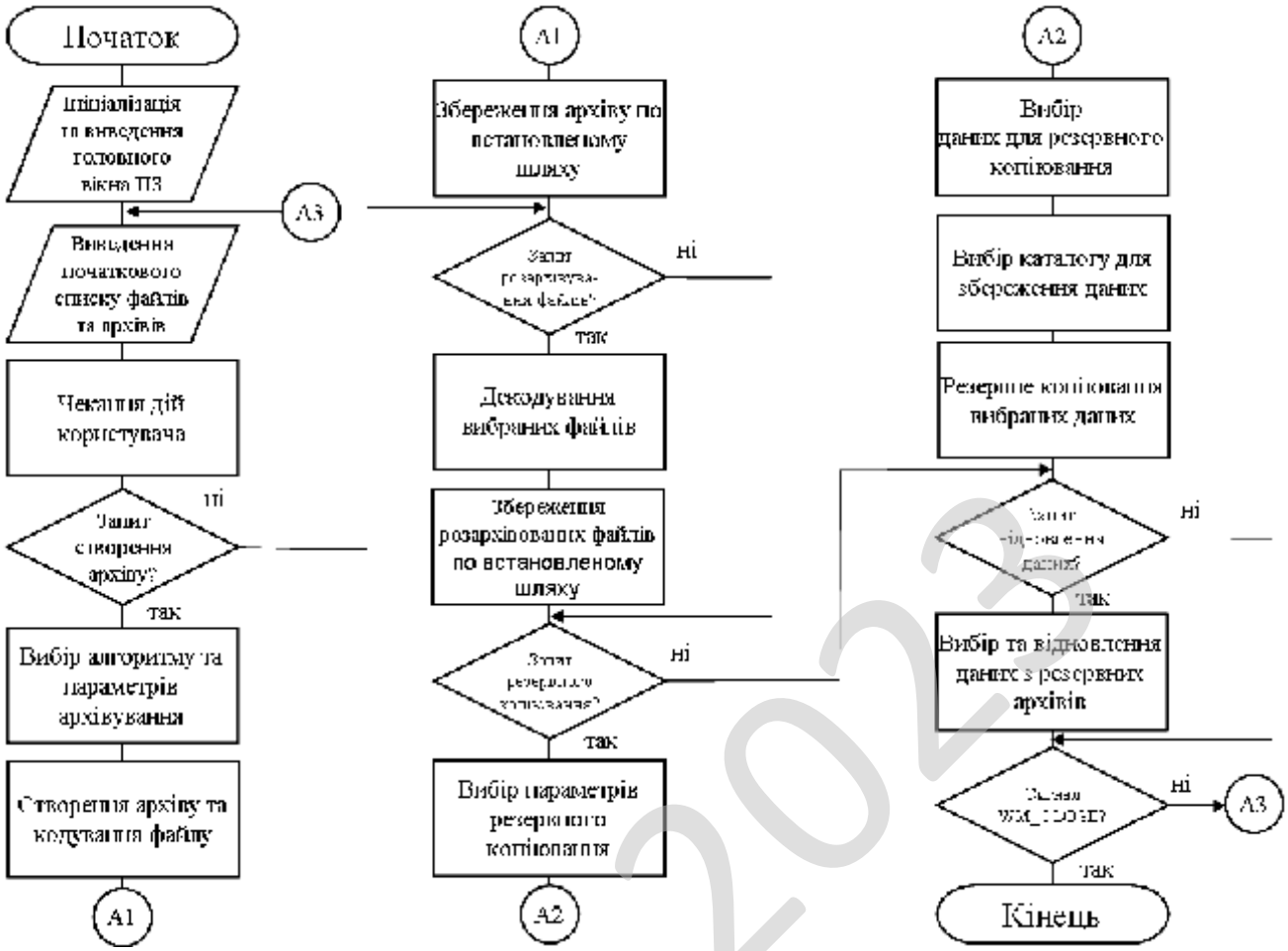

Рисунок 4.1 – Блок-схема основної програми

– Аналіз кількості входжень всіх символів у файл.

– Конвертування символів вхідного алфавіту за частотою появи по убуванню.

– Обрання двох символів з кінця списку.

– Формування нової змінної.

– Присвоєння створеній змінній суми частот появи двох вибраних символів.

– Вибрані символи видаляються зі списку.

– Змінна додається в список та відбувається його повторне сортування.

– В списку залишилася одна змінна (запит).

– Побудова кодового дерева на основі отриманих результатів.

– Створення словника префіксних кодів на основі кодового дерева.

– Збереження створеного словника.

– Читання символу з вхідного файлу.

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

– Кінець файлу EOF (запит).

– Збереження архіву, що містить словник та вихідний файл.

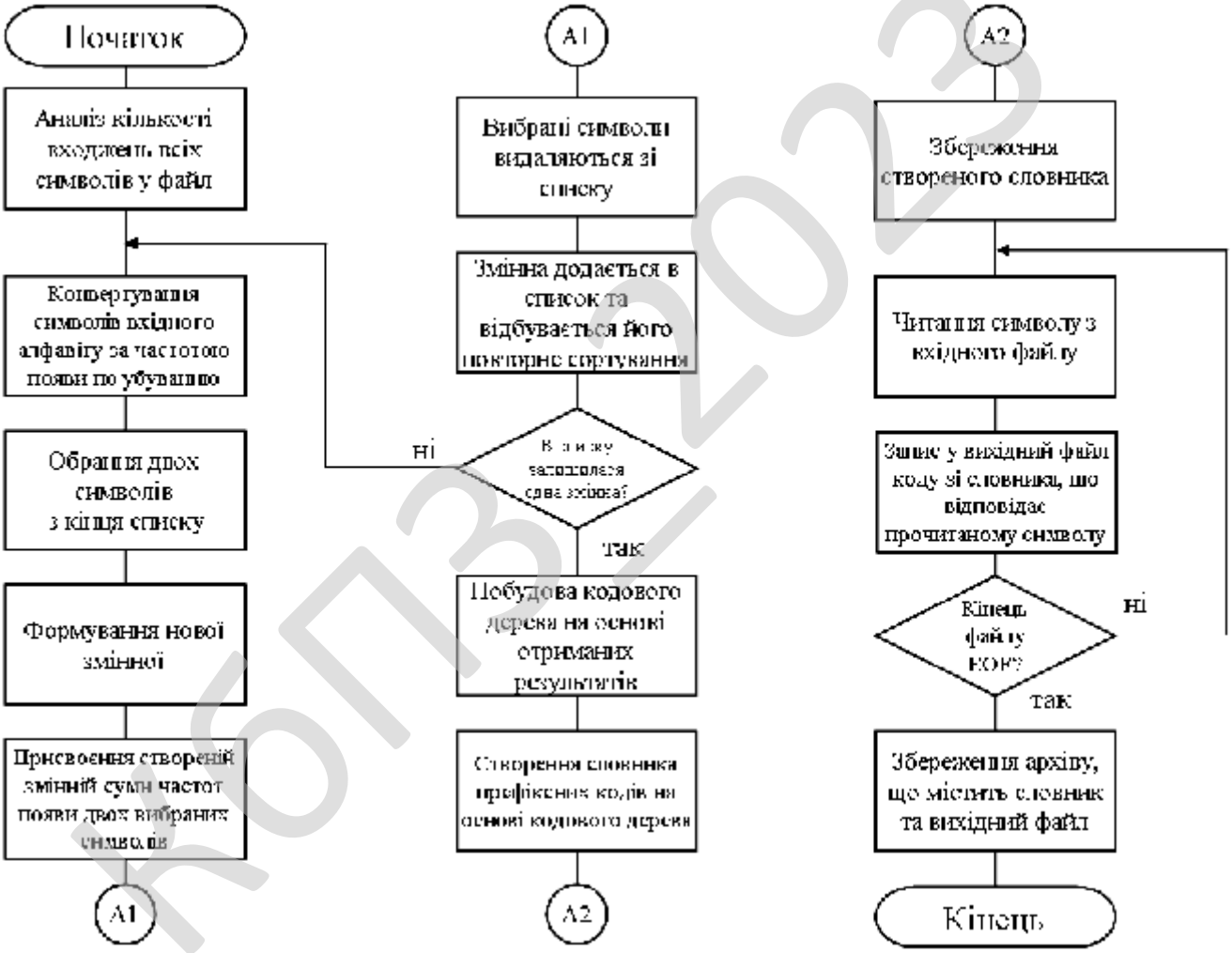

Рисунок 4.2 – Блок-схема підпрограми архівації даних

Опис алгоритмів функціонування системи

Архівації даних використовує код Хаффмана. Алгоритм Хаффмана – адаптивний жадібний алгоритм оптимального префіксного кодування алфавіту з мінімальною надмірністю.

Був розроблений в 1952 році аспірантом Массачусетського технологічного інституту Девідом Хаффманом при написанні ним курсової роботи. В даний час використовується в багатьох програмах стиснення даних.

На відміну від алгоритму Шеннона – Фано, алгоритм Хаффмана залишається завжди оптимальним і для вторинних алфавітів m_2 з більш ніж двома символами.

Цей метод кодування складається з двох основних етапів:

1. Побудова оптимального кодового дерева.

2. Побудова відображення коду-символів на основі побудованого дерева.

Один з перших алгоритмів ефективного кодування інформації був запропонований Д. А. Хаффманом в 1952 році.

Ідея алгоритму полягає в наступному: знаючи ймовірності символів у повідомленні, можна описати процедуру побудови кодів змінної довжини, що складаються з цілої кількості бітів.

Символам з більшою ймовірністю ставляться у відповідність більш короткі коди. Коди Хаффмана володіють властивістю префіксності (тобто жодне кодове слово не є префіксом іншого), що дозволяє однозначно їх декодувати.

Класичний алгоритм Хаффмана на вході отримує таблицю частот з якими зустрічаються символи у повідомленні.

Далі на підставі цієї таблиці будується дерево кодування Хаффмана (Н-дерево).

1. Символи вхідного алфавіту утворюють список вільних вузлів. Кожен лист має вагу, яка може бути рівною або ймовірності, або кількості входжень символу у стиснене повідомлення.

2. Вибираються два вільних вузла дерева з найменшими вагами.

3. Створюється їх батьковий вузол з вагою, рівною їх сумарній вазі.

4. Вузол-батько додається в список вільних вузлів, а два його нащадка видаляються з цього списку.

5. Одній дузі, котра виходить з вузла батька, ставиться у відповідність біт 1, інший – біт 0.

6. Кроки, починаючи з другого, повторюються до тих пір, поки в списку вільних вузлів не залишиться тільки один вільний вузол. Він і буде вважатися коренем дерева.

Наведемо частину коду, яка реалізує алгоритм Хаффмана.

```
unit AB;
//назва модулю
Interface
// інтерфейс
Uses
// бібліотеки
  SysUtils, Classes, Dialogs;
// константи
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
// реалізація
{ 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
//  з 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 SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i <<< 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і ⊕ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} <<< 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Налаштування; Довідка.;

– Функції програми: Сканування; Створення архіву; Відновлення; Виділення архіву; Додавання архіву;иНалаштування; Авторське право.;

– Блок пошуку файлів;

– Блок створених резервних копій;

– Блок журналу роботи ПЗ.

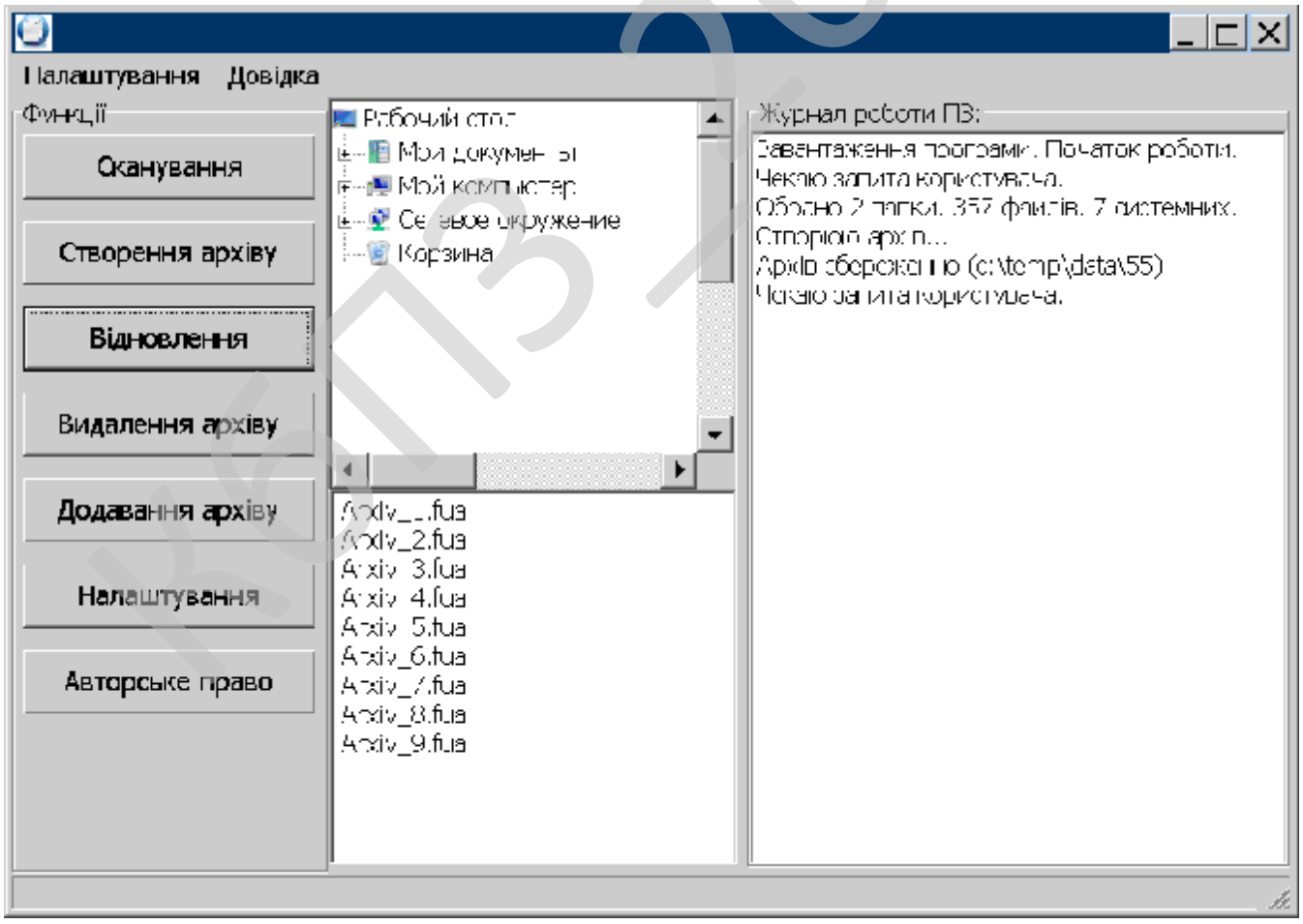

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права, де відображені дані розробника.

Було обрано Shareware умову розповсюдження. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

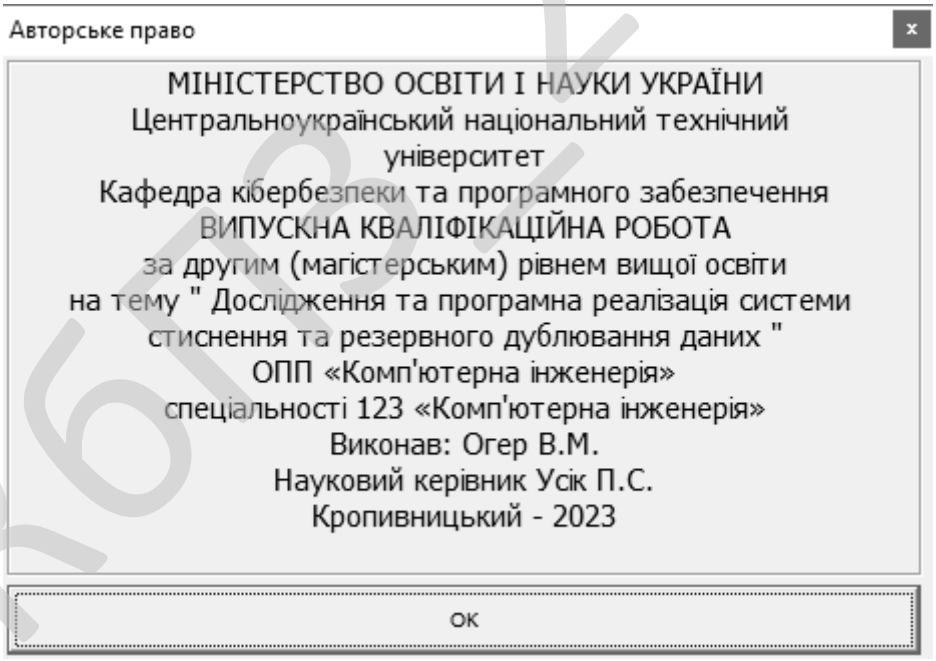

Рисунок 5.2 – Довідка автора

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стиснення та резервного дублювання даних.

Метою розробки є дослідження та програмна реалізація системи стиснення та резервного дублювання даних.

Об'єктом дослідження є процес стиснення та резервного дублювання даних.

Предметом дослідження є методи стиснення та резервного дублювання даних.

Методи дослідження базуються на методах теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод стиснення та резервного дублювання даних.

– Розроблено вітчизняний продукт стиснення та резервного дублювання даних, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи стиснення та резервного дублювання даних.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	17
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	7
8. Кількість форм вихідної інформації.	–	8
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	2
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	1
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	3
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	1
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	1
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	1
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	3
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}028} = 6{,}8 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}8 \cdot (0{,}88 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1 \cdot 1{,}1) = 8{,}38 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). *S* – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо *S* = 90 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 8{,}38^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 100 = 177 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	177	Ф 7.1-7.4
Впровадження	17	Д13
Всього	228	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{228 \cdot 1}{48\text{-}5} = 5,3 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	8	240	4
Кабельні господарства ЛОМ на 1 м.п.	2,5	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	45,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{45 \cdot 2}{1,2} = 75 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 75/(48 \cdot 8) = 0{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	12135	24270
Продакт-менеджер	0,25	12000	6000
Інженер-програміст	5,3	13000	137800
Інженер-електронщик	0,2	7500	3000
Інженер-системотехнік	0,25	7500	3750
Адміністратор мережі	0,5	7500	7500
Системний програміст	0,25	7500	3750
Дизайнер WEB	0,25	7500	3750
Інженер-верстальник	0,25	7700	3850
Бухгалтер-економіст	0,25	12500	6250
Всього за період розробки	$R_{сп}$ = 8,5	-	$Ф_{роб}$ = 199920

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \tag{7.8}$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{199920}{8,5 \cdot 48} = 490 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 9 \cdot 8 \cdot 20000 = 1440000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 144000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 9 \cdot 3500 = 31500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми supercomp.kiev.ua за 29.10.23 – джерело http://supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		12721
Системний блок		7721
Процесор	AMD A8-Series A8-9600 A8 9600 3,1 ГГц 6 Вт, він працює з відеокартою Radeon R (384 шейдерних ядер / 6 обчислювальни ядер) і двоканальним контролером пам'ят DDR4-18600	-
Системна плата	GIGABYTE B450M S2H V2, 1 x PCI-E 2. x1, 1 x PCI-E 3.0 x16, 4 x USB 3.1 Gen1, 1 VGA, 1 x RJ45, 2 x USB 2.0, 2 x PS/2, 1 DVI-D, 1 x HDMI, 1 x optical SPDIF out, 3 Audio, M.2 2280, 4 x Sata 6.0 Gb/s	-
Відеокарта	Інтегрована AMD Radeon R7	-
Жорсткий диск	480 GB SSD	-
Оперативна пам’ять	DDR4 16GB 2666 MHZ KINGSTO (KVR26N19S8/8) 2x8GB	-
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	-
Корпус	Logicpower 8702 – 550w 12cm	-
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	-
інше	Клавіатура, мишка	Подаруно
Монітор	Монітор BenQ GL2450HM Black	3600

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межа до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складем таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	9	12721	11448,9	125937,9
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	147569,4

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1440000	-	-
2. Передавальні пристрої	144000	-	-
Всього по групі	1584000	5	79200
Група 4			
3. Обчислювальна техніка	147570	-	-
Всього по групі	147570	40	59028
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1910976		$А_р$ = 174435,75

Примітка: вартість автомобіля Ford Transit Connect 2002 складає 121875 грн. взята по даним електронного ресурсу

https://auto.ria.com/uk/auto_ford_transit_connect_35495288.html

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 490 \cdot 228/17 = 6579 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6579 \cdot 10 \cdot 0{,}01 = 658 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(6579+658) = 1592 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картриджів, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 5 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 206 \cdot 5 \cdot 1 = 1030 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 17):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 35,6 грн./шт., DVD-R box – 55 грн./шт.

$$З_{М2} = 17 \cdot 55 = 935 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 500+508+1155 = 2163 \text{ грн.}$$

$$З_М = (1030 + 935 + 2163)/17 = 243 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 17 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 174436 \cdot 2/(17 \cdot 12) = 1710 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6579
2. Додаткова зарплата виконавців	$З_д$	658
3. Відрахування на соціальні потреби	$С_{оц}$	1592
4. Загальногосподарські витрати	$Г_{осп}$	987
5. Витрати на матеріали	$З_М$	243
6. Освоєння нових операційних систем, мов програмування	$О_п$	987
7. Амортизація основних фондів	$А_м$	1710
8. Повна собівартість програмного забезпечення	$С_п$	12756
9. Плановий прибуток	$П_р$	6378
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	19134
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3826,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	22960,8

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 6579+658+1592+987+243+987+1710 = 12756 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 12756 = 6378 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	22961
Всього капітальних витрат	–	22961

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	38650	19325
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	5740
Всього витрат за рік	$І$	38650	25065

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 120 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 38650 \text{ грн},$$

до:

$$З_{р\,нов} = 120 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 19325 \text{ грн.}$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Витрати на електроенергію не змінились , тому: $З_{ел\,баз} = З_{ел\,нов}$

.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	22961	–	5740,25
Всього відрахувань	-	–	22961	–	5740,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (19134 - 12756) \cdot 17 - (0,05 \cdot 1584000 + 0,4 \cdot 147570 + \\ +0,2 \cdot 121875 + 0,25 \cdot 40531 + 0,1 \cdot 17000) \cdot 2/12 = 79353 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1910976}{(19134 - 12756) \cdot 17 \cdot 12 / 2} = 2,9 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	17
2. Повна собівартість розробленої програми	Грн.	12756
3. Ціна розробленої програми	Грн.	19134
4. Плановий прибуток від реалізації розробленої програми	Грн.	6378
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1910976
7. Загальний прибуток від реалізації програмної продукції	Грн.	108426
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	79353
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,9
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	22961
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7845
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,7

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об’єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (38650\text{-}25065)\text{-}0{,}25 \cdot 22961 = 7845 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{22961}{38650 - 25065} = 1{,}7 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

8.2 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і

прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є

розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об'єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

8.3 Аналіз умов праці програміста

Умови праці в приміщенні, в якому знаходиться робоче місце програміста є сприятливими. Приміщення обладнане автономною системою газового опалення, основною перевагою якого є програмування режиму роботи в залежності від погодних умов, оскільки клімат є нестійким. Використовується система природної та штучної вентиляції, що забезпечує ефективну циркуляцію повітря. В кабінеті знаходиться кондиціонер *HYUNDAI ARN07HSSUAWF1/ARU07HSSUAWF1*.

Засоби копіювальної техніки знаходяться на достатньо далекій відстані від робочих місць, оскільки приміщення складає 15 м^2, а у відділі налічується два працівники, тобто концентрація озону та оксиду азоту в повітрі є невисокою. Таким чином на кожного програміста приходиться 7,5 м^2 що відповідає нормам Державним санітарним правилам і нормам ДСанПіН 3.3.2.007-98 [2]. Висота стелі приміщення складає 2,9 метри, що також не порушує нормативні вимоги.. Прибиральники підтримують порядок в службових приміщеннях, дотримуються санітарно-гігієнічних норм по прибиранню приміщень, витирають пил, підмітають підлогу наприкінці кожного робочого дня.

В цілому потрібно відмітати застарілість офісної техніки та відсутність клавіатур з ергономічною розкладкою та рідкокристалічних моніторів, які здійснюють менш негативний вплив на стан здоров'я працівників відділу.

Оформлення інтер'єру приміщення є відповідає вимогам з ергономіки та стимулює працівників до підвищення працездатності та зниження втоми. Стеля білого кольору створює оптичний ефект збільшення висоти приміщення, підлога пофарбована коричневим кольором, а стіни – у жовтий. Перевагами даного кольору є створення відчуття теплоти, здатність привертати увагу без додаткової втоми.

Висота столу складає 72 см., до того ж його можна регулювати відповідно до власних потреб. Стіл має достатній внутрішній об'єм, завдяки ширині у 70 см. та висоті простору під столом – у 60 см., є достатньо важким для забезпечення стійкості. Крісла забезпечують фізіологічно раціональну позу, мають підлокітники, здатні обертатися та регулятор висоти, кута нахилу спинки й відстані спинки від краю сидіння.

В кабінеті створено оптимальні умови праці відносно температури, вологості приміщення та вентиляції.

Наприкінці аналізу небезпечних факторів праці побудуємо підсумкову таблицю 8.1.

Щодо вимог електробезпеки, то приміщення за небезпекою ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, без пилу, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

Для запобігання поразки електричним струмом в приміщенні відділу використовується ряд організаційно-технічних заходів: розташування проводів живлення поза зоною пересування людей; допуск до роботи електроприладів тільки тих робітників, що знайомі із технікою безпеки; використання мережних продовжувачів з вбудованими запобіжниками на 0,1 А; при ремонті обладнання персонал попереджується.

Таблиця 8.1 – Підсумкова таблиця значень параметрів небезпечних факторів праці

Найменування параметра	Значення параметра		Нормативний документ
	Фактичне	Нормоване	
Освітленість штучна, лк	300	300	ДБН.В 2.5-28:2018 [1]
Значення КПО,%	1,0	1,1	ДБН.В 2.5-28:2018 [1]
Повітрообмін,м /год взимку влітку	 76 36	 80 80	 ДСН 3.3.6.042-99 ДСН 3.3.6.042-99
Температура повітря. °С взимку влітку	 22 24	 21-25 27-28	 ДСанПіН 3.3.2-007-98 ДСанПіН 3.3.2-007-98 [2]
Відносна вологість,% взимку влітку	 60 55	 <75 <60	 ДСанПіН 3.3.2-007-98 ДСанПіН 3.3.2-007-98 [2]
Швидкість переміщення повітря, м/с взимку	 0,16	 <0,2	 ДСанПіН 3.3.2-007-98 [2]
влітку	0,10	<0,2	ДСанПіН 3.3.2-007-98 [2]

Устаткування, що працює в приміщенні живиться від мережі 220 В та частотою 50 Гц. Споживачами цієї напруги є також джерела штучного освітлення. Вони розташовуються на висоті 2,9 м., що задовольняє нормі, відповідно до якого джерела освітлення повинні розташовуватися на висоті 2,5 м від підлоги.

Проводка схована. У якості розеток для підключення устаткування застосовуються розетки з заземленим кожухом, захищеного від випадкового доторку до струмоведучих частин. Електроустаткування, що знаходиться в приміщенні відділу відноситься до установок напругою до 1000 В.

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп’ютера, але тут використовуються системні блоки, що відповідають стандартові фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення.

Основні причини ураження людини електричним струмом на робочому місці:

– дотик до металевих неструмоведучих частин (корпусу, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції;

– нерегламентоване використання електричних приладів;

– відсутність інструктажу співробітників з правил електробезпеки.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ/м.

Отже за результатами проведеного аналізу можна зробити висновки, що всі показники знаходяться у межах запропонованих значень

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною множинного зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення будемо застосовувати вертикальні електроди з сталевого прокату круглого перерізу діаметром *45 мм.*, довжиною *L=2 м.,* та горизонтальний електрод – металева полоса з перетином *45·5* мм. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: $H=0,55$ *м*. Відстань між вертикальними заземлювачами (електродами) $A=3$ *м*. Глибина закладення горизонтального контура заземлення $t=0,75$ *м*. Опір заземлювача, який нормується: $R_{3H} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,75+2/2=1,75 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр ветикальньного електрода (задан):

$$D_в = 45 \text{ мм.} = 0,045 \text{ м.}$$

Відношення $A/L=3/2=1,5$.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O=0,366(\rho/L)[\lg(2L/D_в)+(1/2)\lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/2)[(\lg(2\cdot 2/0,045)+(1/2)\lg((4\cdot 1,75+2)/(4\cdot 1,75-2))]=$$

$$=20,6 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів $K_{ев}=0,53$ при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [11].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 20,6 / (0,53\cdot4)= 9,75 \approx 10\ шт.$$

Визначаємо довжину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot3\cdot10=30,7\approx 31\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П =0,366(\rho\cdot K_П/L_П)lg(2(L_П \cdot L_П)/(B\cdot t))=$$

$$=0,366(40\cdot5/40)\cdot lg((2\cdot40^2)/(0,045\cdot0,75))=11,57\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [11]:

$B = 45$ *мм.* $= 0,045$ *м.*- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [11]:

$$R =(R_O\cdot R_П)/(R_O\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(20,6\cdot 11,57)/(20,6\cdot 0,55+ 10\cdot 11,57\cdot 0,53)=3,3\ Ом.$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \le R_{ЗН}$ виконується ($3,3 \le 4$).

Так як R суттево быльше $R_{ЗН}$, зменшимо кількість вертикальних електродів до *8* і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює *8* при $R =3,9$ *Ом.*

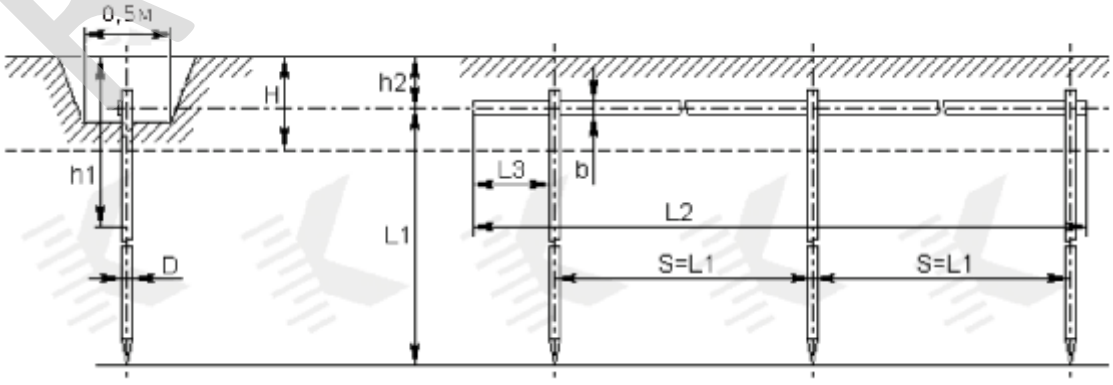

Рисунок 8.1 – Схема штучного заземлення .

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд питань пожежної безпеки, небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів о питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи стиснення та резервного дублювання даних.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів стиснення та резервного дублювання даних.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стиснення та резервного дублювання даних.

– Досліджена система стиснення та резервного дублювання даних.

– На основі отриманих результатів досліджень створена програмна реалізація системи стиснення та резервного дублювання даних.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стиснення та резервного дублювання даних.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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 Embarcadero Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 7845 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Огер В.М. Дослідження та програмна реалізація системи стиснення та резервного дублювання даних // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

3. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

4. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

5. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

6. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

7. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

8. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

9. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

11. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

12. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

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

14. Smirnov O., 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.

15. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

16. 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.

17. 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.

18. 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.

19. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

20. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

21. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

22. 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.

23. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

24. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

25. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

26. 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.

27. 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.

28. 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.

29. 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

30. 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.

31. 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.

32. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

33. 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.

34. 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.

35. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

36. 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).

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

38. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

39. Смірнова Т.В., Солових Є.К., Смірнов О.А., 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.

40. 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.

41. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

42. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

43. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

44. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

45. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

46. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. – Випуск 2 (46) – Х.: ХУПС – 2016. – С. 146-149.

47. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

48. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

50. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

51. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

52. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

53. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.0925 «Автоматизація й комп'ютерно-інтегровані технології». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 1.12.2011 року № 1/11-11258. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи стиснення та резервного дублювання даних.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи стиснення та резервного дублювання даних.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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 з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи стиснення та резервного дублювання даних;

– 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 Embarcadero Delphi.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 103 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Усік П.С.

Дослідження та програмна реалізація системи стиснення та резервного дублювання даних

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 54

Літера: РП

Кропивницький – 2023 року

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 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 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(' МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи стиснення та
резервного дублювання даних');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник:  Усік П.С.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив:  студент   Огер Владислав Миколайович');
Memo1.Lines.Add('                     гр. КІ-22М-1');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
end;

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