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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Бабак А.Р.
«____» _____ 2023 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 р.
Рецензент _____
_____

Кафедра_КбПЗ_2023рік

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

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

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

Бабаку Андрію Романовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю*

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

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

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

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

Кафедра КОПЗ – 2023рік

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Бабак А.Р.
(прізвище та ініціали)

Кафедра_КбПЗ_2023рік

АНОТАЦІЯ

Бабак А.Р. Програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

Метою розробки є програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

Результат роботи – програмна реалізація системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

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 RAD Studio Delphi.

Ключові слова: кібербезпека, стиснене збереження інформації, підвищена надійністі

Кафедра_КБПЗ_2023рік

ABSTRACT

Babak A.R. Cybersecurity system software for compressed information storage with increased reliability. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system for compressed information storage with increased reliability.

The purpose of the development is the software of the cyber security system for the compressed storage of information with increased reliability.

The result of the work is the software implementation of the cyber security system for compressed storage of information with increased reliability.

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 RAD Studio Delphi environment.

Keywords: cybersecurity, compressed information storage, increased reliability

Кафедра_КБПЗ_2023рік

ЗМІСТ

Кафедра_КБПЗ_2023рік

					ВКРБ-125.23.0002.00.00.ПЗ			
Вим	Арк.	№ докум.	Підп.	Дата				
Розроб.		Бабак А.Р.			Програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю	Літ.	Аркуш	Аркушів
Перев.		Якименко Н.М.				Б	1	81
						ЦНТУ КБ-19		
Н.контр.		Гермак В.С.						
Затв.		Смірнов О.А.						

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

LZW – Алгоритм Зіва-Лемпела

Кафедра_КбПЗ_2023рік

ВСТУП

Актуальність теми. При експлуатації комп'ютера по самих різних причинах можливе псування або втрата інформації на жорстких дисках. Це може відбутися через фізичне псування жорсткого диска, неправильного коректування або випадкового знищення файлів, руйнування інформації комп'ютерним вірусом і т.д. Для того щоб зменшити втрати в таких ситуаціях, варто мати архівні копії використовуваних файлів і систематично обновляти копії змінюваних файлів.

Для збереження інформації можна звичайно її дублювати, однак при цьому копії займають стільки ж місця, скільки займають вихідні файли.

Більш зручно використовувати для створення копій спеціально розроблені програми архівації файлів. Ці програми дозволяють не тільки заощадити місце, але й поєднувати групи спільно використовуваних файлів в один архівний файл, що помітно полегшує ведення архівів.

Прийнято розрізняти архівацію й упакування (компресію, стиск) даних. У першому випадку мова йде про злиття декількох файлів і навіть каталогів у єдиний файл – архів (прикладом використання такої технології в чистому виді може служити формат TAR). У другому – про скорочення обсягу вихідних файлів шляхом усунення надмірності (у даній роботі розглядається впакування без втрат інформації, тобто з можливістю точного відновлення вихідних файлів). Як правило, сучасні архіватори забезпечують також стиск даних, будучи, таким чином, ще й пакувальниками, однак існують і чисто «пакувальні» утиліти типу Gzip, що стискають окремі файли, перетворюючи їх у формат Z або GZ.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

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

Кафедра_КБПЗ_2023рік

– Огляд існуючих систем для стисненого збереження інформації з підвищеною надійністю.

– Дослідження системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

– Програмна реалізація системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для стисненого збереження інформації з підвищеною надійністю.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КбПЗ_2023рік

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для формування програмного забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю. Для опису її дії введемо декілька основних понять.

Стиск інформації – це процес перетворення інформації, що зберігається у файлі, до виду, при якому зменшується надмірність у її поданні й відповідно потрібен менший обсяг пам'яті для зберігання.

Стиск інформації у файлах виробляється за рахунок усунення надмірності різними способами, наприклад за рахунок спрощення кодів, виключення з них постійних біт або подання повторюваної послідовності символів у вигляді коефіцієнта повторення й відповідних символів. Застосовуються різні алгоритми подібного стиску інформації. Стискуватися можуть як один, так і кілька файлів, які в стислому виді містяться в так званий архівний файл або архів.

Архівний файл – це спеціальним образом організований файл, що містить у собі один або кілька файлів у стислому або незжатому виді й службову інформацію про імена файлів, дату й час їхнього створення або модифікації, розміри і т.п. Метою упакування файлів звичайно є забезпечення більше компактного розміщення інформації на диску, скорочення часу й відповідно вартості передачі інформації з каналів зв'язку в комп'ютерних мережах. Крім того, упакування в один архівний файл групи файлів істотно спрощують їхній перенос із одного комп'ютера на інший, скорочує час копіювання файлів на диски, дозволяє захистити інформацію від несанкціонованого доступу, сприяє захисту від зараження комп'ютерними вірусами.

Кафедра_КБПЗ_2023рік

Ступінь стиску файлів характеризується коефіцієнтом Кс, обумовленим як відношення обсягу стислого файлу Vc до обсягу вихідного файлу Vo, виражене у відсотках. Ступінь стиску залежить від використовуваної програми, методу стиску й типу вихідного файлу. Найбільше добре стискуються файли графічних образів, текстові файли й файли даних, для яких ступінь стиску може досягати 5 – 40%, менше стискуються файли програм, що виконуються, і завантажувальних модулів – 60 – 90%. Майже не стискуються архівні файли.

Програми для архівації відрізняються використовуваними методами стиску, що відповідно впливає на ступінь стиску.

Архівація (упакування) – перенесення (завантаження) вихідних файлів в архівний файл у стислому або незжатому виді.

Розархівація (розпакування) – процес відновлення файлів з архіву точно в такому виді, який вони мали до завантаження в архів. При розпакуванні файли витягають із архіву й містяться на диск або в оперативну пам'ять.

Програми, що здійснюють упакування й розпакування файлів, називаються **програмами-архіваторами.**

1.2 Область застосування

При виборі інструмента для роботи з упакованими файлами й архівами варто враховувати два фактори: ефективність, тобто оптимальний баланс між економією дискової пам'яті й продуктивністю роботи, і сумісність, тобто можливість обміну даними з іншими користувачами.

Сумісність, мабуть, сьогодні більше важлива, тому що за ступенем стиску, що досягається, конкуруючі формати й інструменти розрізняються на відсотки (але не в рази), а обчислювальна потужність сучасних комп'ютерів робить час обробки архівів не настільки істотним показником, як, скажемо, десять років тому.

Кафедра_КбПЗ_2023рік

Тому при виборі інструмента для роботи з архівами найважливішим критерієм для більшості користувачів (у всякому разі тих, для кого обмін більшими масивами даних – насущна проблема), імовірно, є здатність програми «розуміти» найпоширеніші архівні формати, навіть якщо ці формати не найефективніші.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КбПЗ_2023рік

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Збільшення обчислювальної потужності комп'ютерів сприяє появі більше складних алгоритмів упакування даних, за допомогою яких можна одержувати файли меншого розміру. У той же час удосконалюються й уже існуючі алгоритми. Тому питання "який архіватор вибрати для стиску?" не втрачає актуальності. Саме тому ми вирішили провести невеликий тест і перевірити, як найпоширеніші архіватори справляються з компресією файлів різних типів. При цьому, ми звертали увагу й на швидкість стиску, адже в деяких випадках користувач готовий пожертвувати обсягом, аби тільки архів був створений швидше.

Перш ніж перейти безпосередньо до тестів, коротко зупинимося на основних факторах, які впливають на те, наскільки сильно стискуються дані:

– Тип файлів. Якщо файл, якому необхідно заархівувати, уже піддавався компресії, його повторний стиск навряд чи дасть гарні результати. Це відноситься, наприклад, до відеофайлів у форматі DIV, до графічних файлів у форматі JPEG, до файлів програм, що виконуються.

– Ступінь стиску. У налаштуваннях кожного архіватора можна вибрати налаштування стиску. Якщо метою є одержати файли як можна меншого розміру, вибирається максимальний ступінь стиску або навіть створення безперервного архіву (solid archive). Якщо ж має значення швидкість, вибирається мінімальний ступінь стиску.

Кафедра_КбПЗ_2023рік

– Формат стиску. Оскільки різні програми використовують різні алгоритми стиску, формат має досить велике значення. У цьому огляді ми зупинимося на таких розповсюджених форматах як Zip і RAR.

Дати об'єктивну оцінку кожному з архіваторів дуже важко, хоча б тому, що результати архівування дуже сильно відрізняються, залежно від обраних файлів. Тому, щоб більш точно оцінити можливості кожної із програм, ми вирішили провести серію пробних стисків.

Для проведення тесту ми створили кілька папок, у кожну з яких помістили файли різних типів:

– Графічні файли у форматі Tif. Загальна кількість файлів – 112 штук. Загальний розмір до архівації 200 520 282 байт;

– Аматорське відео з параметрами MJPG 320x240 30.00 fps 4535 Кб/с з налаштуваннями аудіо PCM 11024 Гц моно 88 Кб/с. Розмір файлу до архівації 203 396 754 байт;

– Аудіофайли у форматі MP3 з різними налаштуваннями стиску. Загальна кількість файлів – 33. Загальний розмір до архівації 200 685 630 байт;

– Файли у форматі DOC. Загальна кількість файлів – 150. Загальний розмір до архівації 100 009 472 байт.

Кожна із чотирьох папок була заархівірувана з мінімальними налаштуваннями стиску (висока швидкість – великий розмір файлу) і з максимальними налаштуваннями стиску (низька швидкість – маленький розмір файлу) у формати Zip, Rar і Ace. Оскільки деякі із програм, які ми відібрали для тестування, підтримують створення файлів у декількох форматах, такі архіватори як WinRar і Winace фігурували в тесті двічі.

Для архівації використовувалися програми:

– WinRar 3.51. Цей архіватор дає можливість створення архівів у форматах Zip і Rar, тому тести в програмі були проведені для кожного із цих форматів стиску. Незважаючи на те, що сьогодні практично всі архіватори

Кафедра_КПЗ_2023рік

підтримують розпакування файлів цього формату, архівувати в RAR можна тільки за допомогою WinRar;

– Winace. Програма дозволяє створювати архіви у форматі Zip і власному форматі Ace. Тести були проведені для обох форматів;

– 7-zip. Архіватор стискає файли у формат Zip і власний формат 7z. Однак через специфіку формату 7z він не був включений у тест. Використовувати цей формат для стиску даних великого розміру не має сенсу, оскільки він працює дуже повільно;

– Power Archiver. За допомогою програми можна стискати файли у формат Zip;

– Winzip. Програма робить стиск у формат Zip.

Результати, отримані для всіх типів файлів після стиску у формат Zip архіватором WinAce з мінімальними налаштуваннями, нас спантеличили, оскільки вони дуже сильно відрізнялися від результатів, які показали для формату Zip інші програми. Файли, стислі у формат Zip програмою WinAce, були значно більше, ніж ZIP-архіви, отримані за допомогою інших утиліт. Тому ми провели додатковий тест – заархівували всі чотири папки за допомогою WinAce, вибравши в налаштуваннях формату Zip опцію Store, тобто, "без компресії". Розмір отриманих архівів по байтах збігся з розмірами архівів, які були отримані при виборі мінімального ступеня стиску. Таким чином, імовірно, творці архіватора WinAce припустилися помилки в програмі, і замість компресії з мінімальним ступенем стиску у формат Zip програма файли не стискає.

У загальному ж, можна сказати, що стабільно гарні результати показує формат RAR. Приємно здивувала ступінь стиску MP3 і відеофайлів, які звичайно майже не стискуються. Остання версія WinZip також дає непогані результати. Якщо до виходу десятої версії програми альтернативні архіватори стискали файли у формат Zip краще WinZip, то тепер найвищий ступінь стиску досягається саме при використанні цієї програми. Кожний, хто пробував архівувати файли різних типів декількома архіваторами, знає, що вгадати заздалегідь, який

Кафедра_КОПЗ_2023рік

результат буде отриманий, часто буває неможливо. Навіть якщо ви досвідченим шляхом з'ясували, що графічні файли найкраще стискає архіватор 1, не виключено, що найдеться така картинка, що краще заархівує архіватор 2. Саме тому наші тести націлені, скоріше, на відображення загальних тенденцій, і їхні результати не можуть сприйматися як такі, які дійсні для всіх файлів, які ви будете стискати.

Результати тестів наведені на рисунках 2.1 – 2.16.

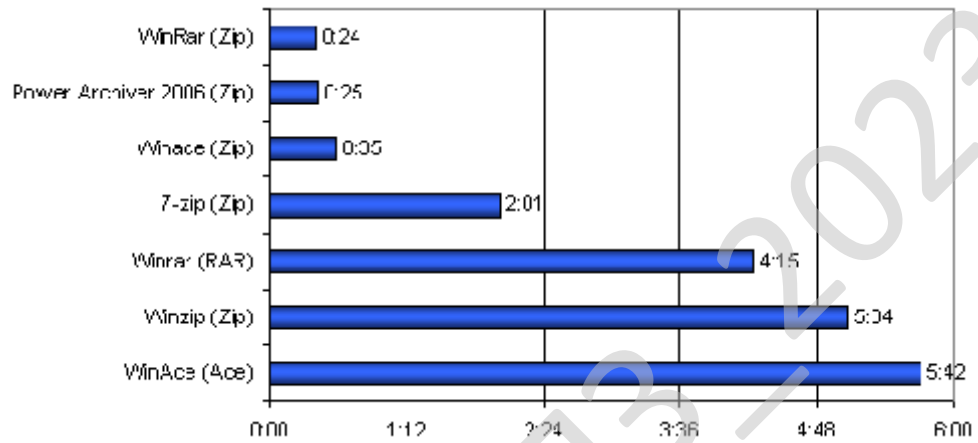

Рисунок 2.1 – Архівування відеофайлу з максимальними налаштуваннями стиску, хв

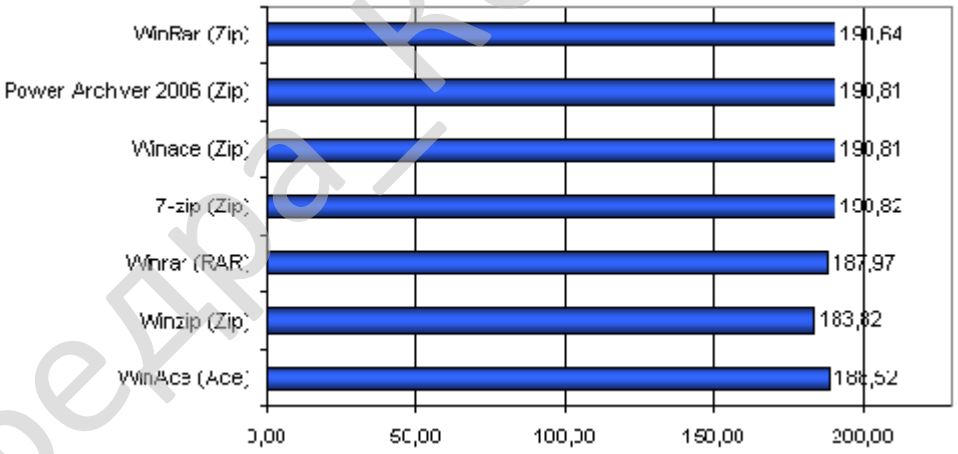

Рисунок 2.2 – Архівування відеофайлу з максимальними налаштуваннями стиску, Мбайт

Кафедра КбПЗ 2023рік

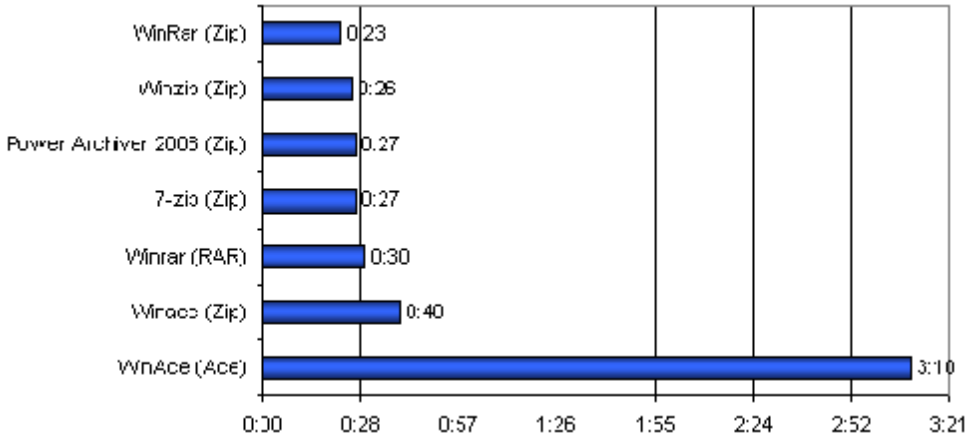

Рисунок 2.3 – Архівування відеофайлу з мінімальними налаштуваннями стиску, хв

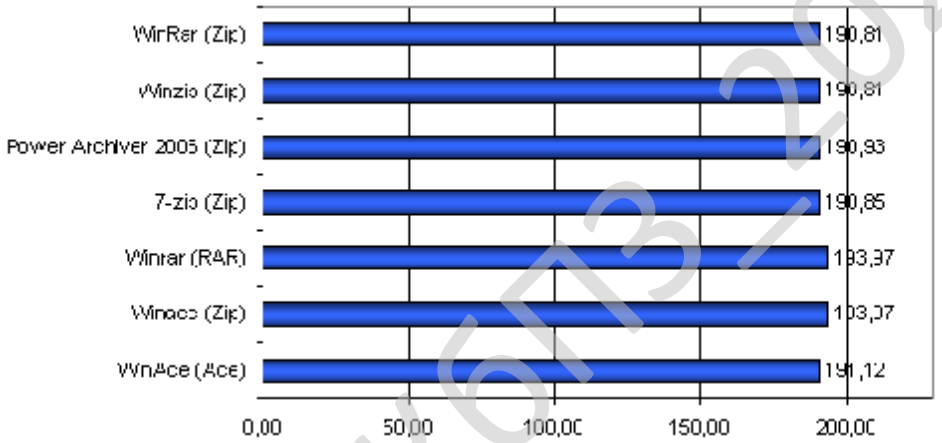

Рисунок 2.4 – Архівування відеофайлу з мінімальними налаштуваннями стиску, Мбайт

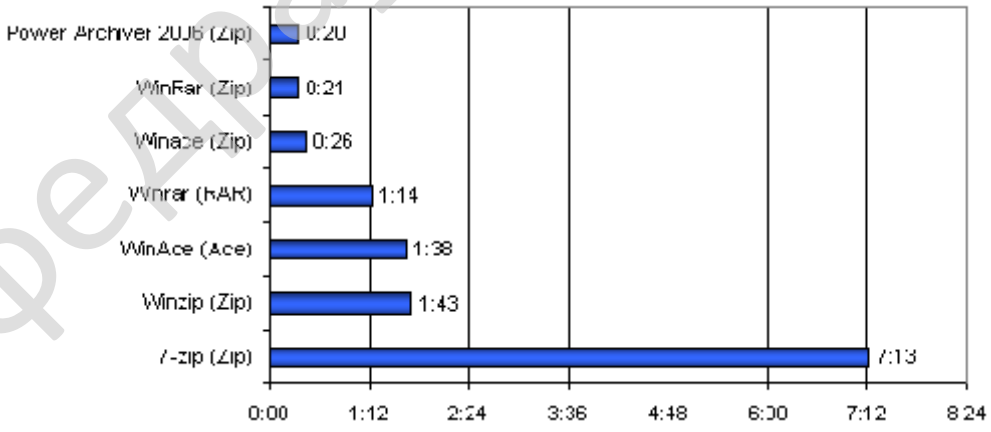

Рисунок 2.5 – Архівування графічних файлів з максимальними налаштуваннями стиску, хв

Кафедра_КОПЗ_2023рік

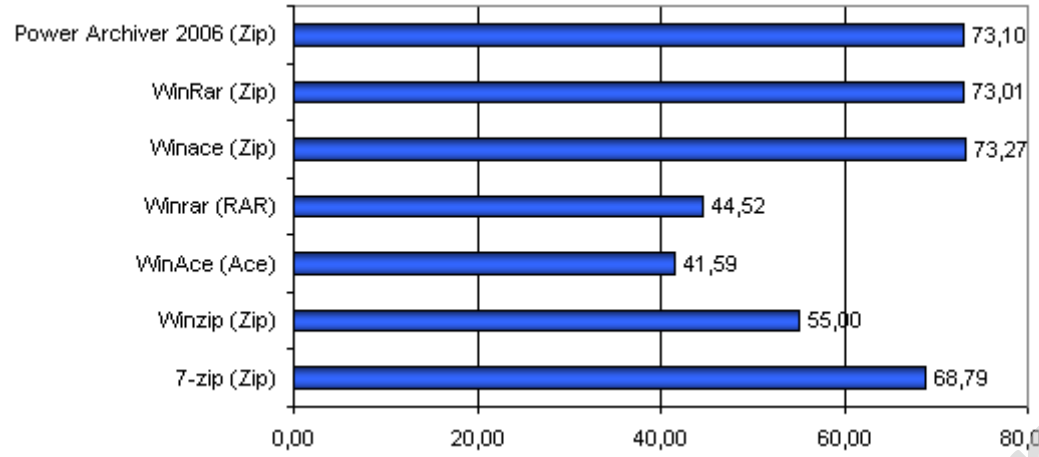

Рисунок 2.6 – Архівування графічних файлів з максимальними налаштуваннями стиску, Мбайт

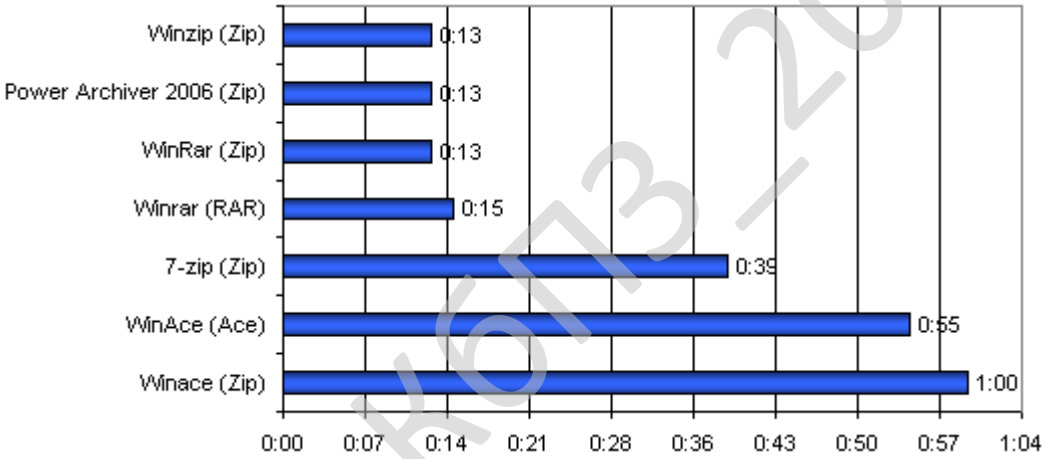

Рисунок 2.7 – Архівування графічних файлів з мінімальними налаштуваннями стиску, хв

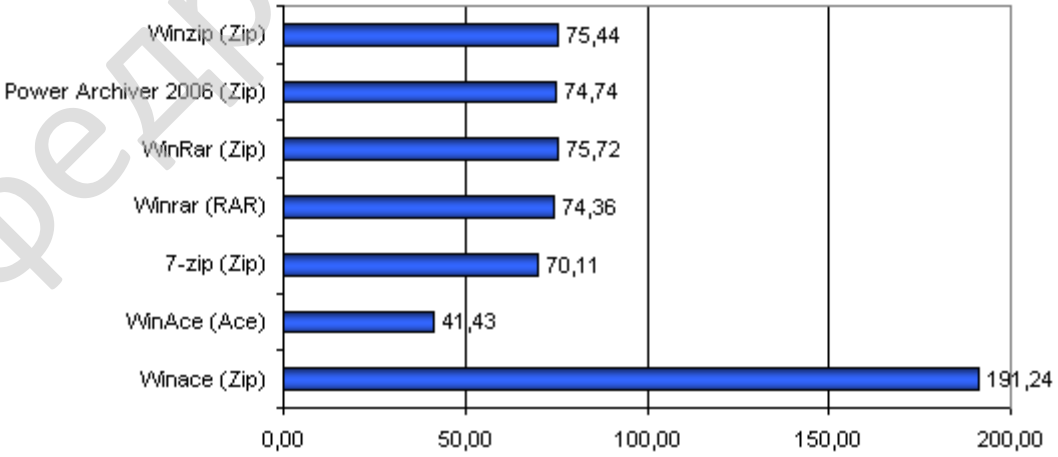

Рисунок 2.8 – Архівування графічних файлів з мінімальними налаштуваннями стиску, Мбайт

Кафедра_КОПЗ_2023рік

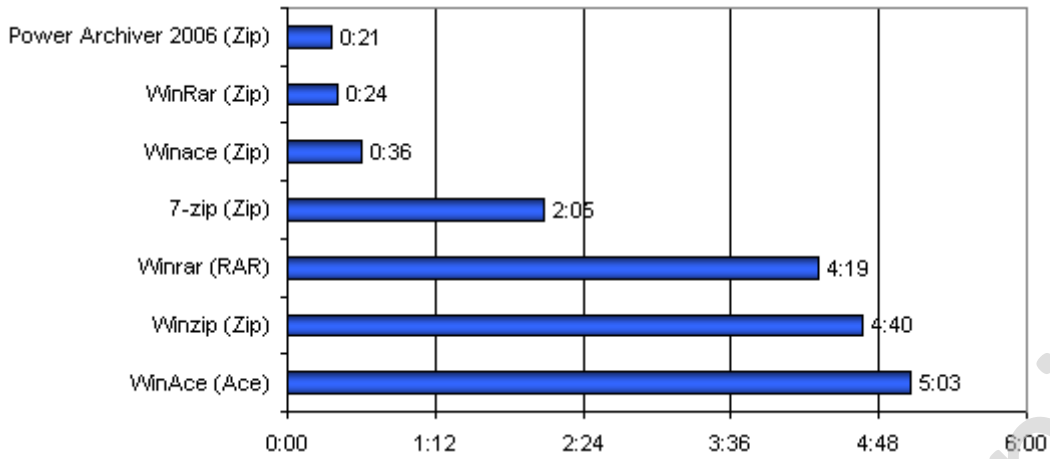

Рисунок 2.9 – Архівування звукових файлів з максимальними налаштуваннями стиску, хв

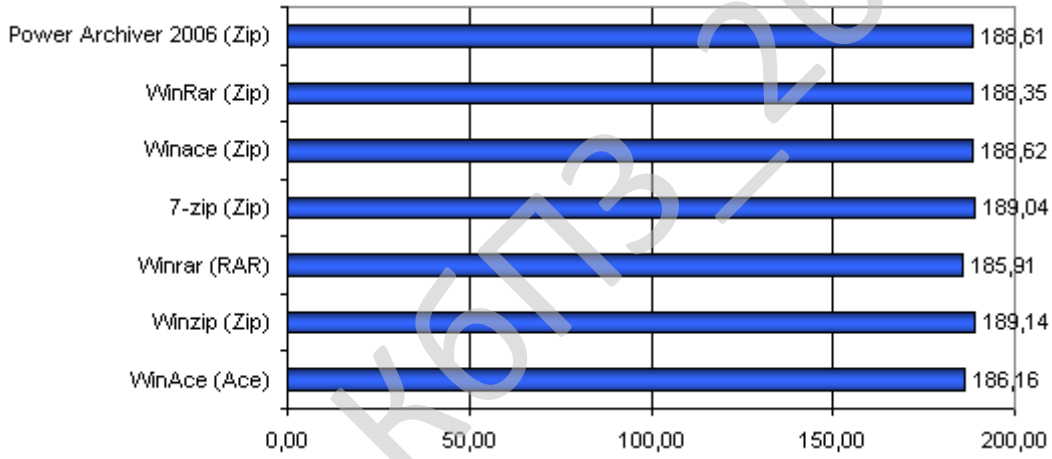

Рисунок 2.10 – Архівування звукових файлів з максимальними налаштуваннями стиску, Мбайт

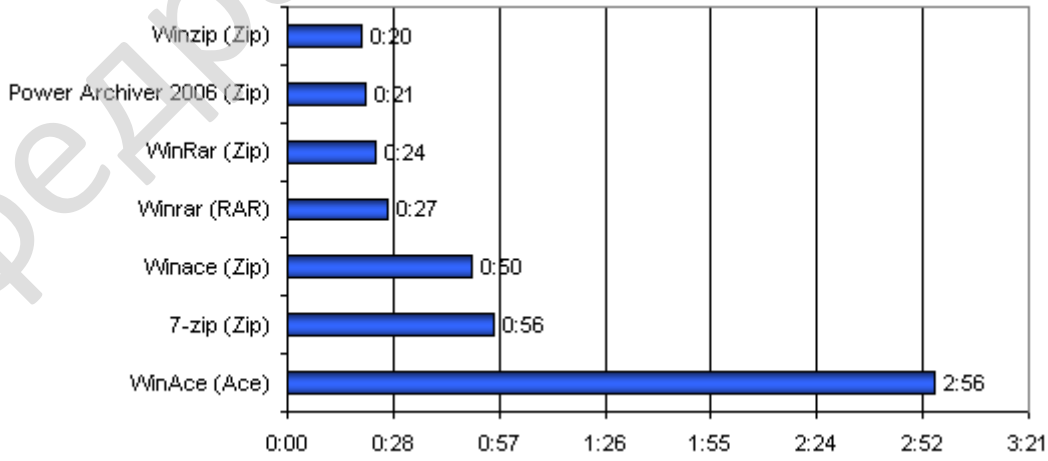

Рисунок 2.11 – Архівування звукових файлів з мінімальними налаштуваннями стиску, хв

Кафедра_КбПЗ_2023рік

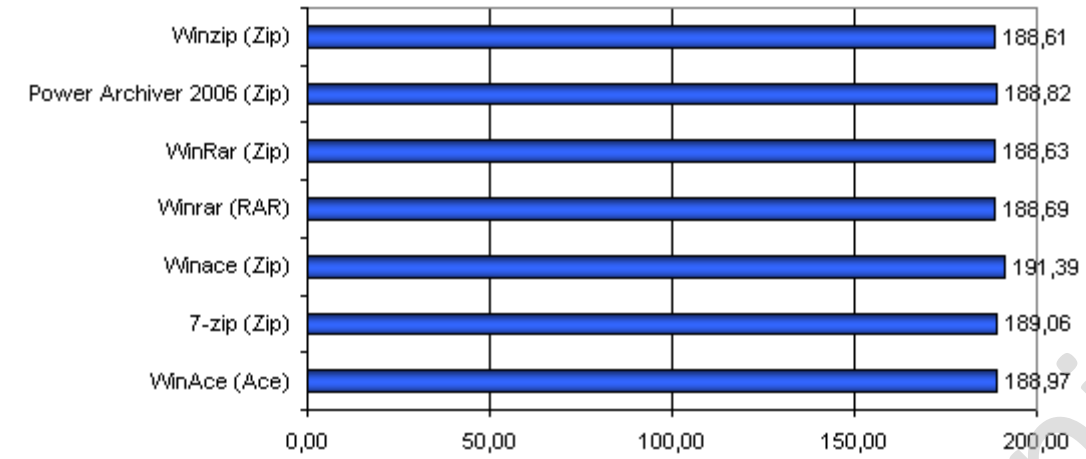

Рисунок 2.12 – Архівування звукових файлів з мінімальними налаштуваннями стиску, Мбайт

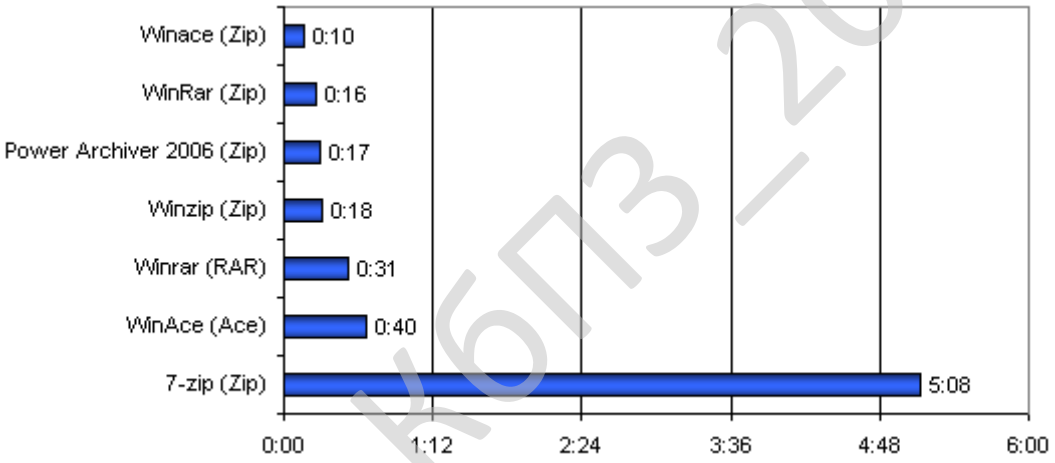

Рисунок 2.13 – Архівування текстових файлів з максимальними налаштуваннями стиску, хв

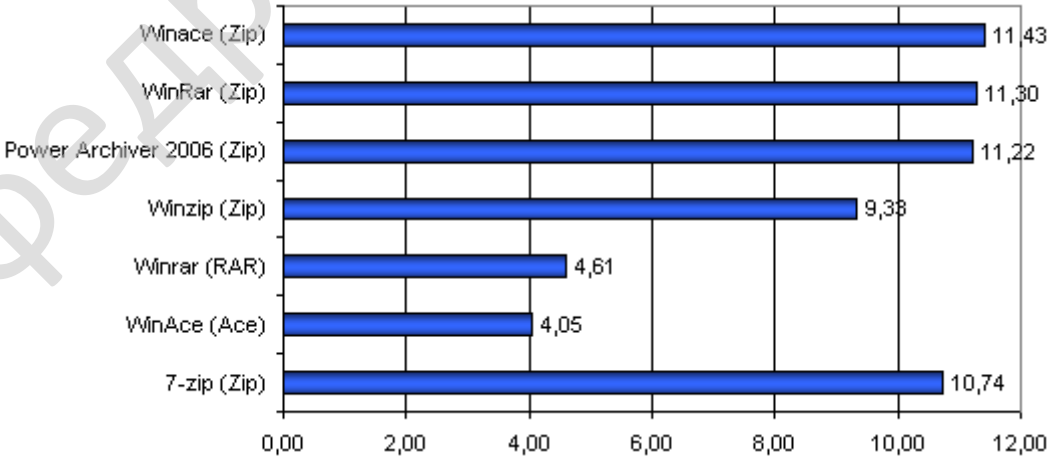

Рисунок 2.14 – Архівування текстових файлів з максимальними налаштуваннями стиску, Мбайт

Кафедра_КбПЗ_2023рік

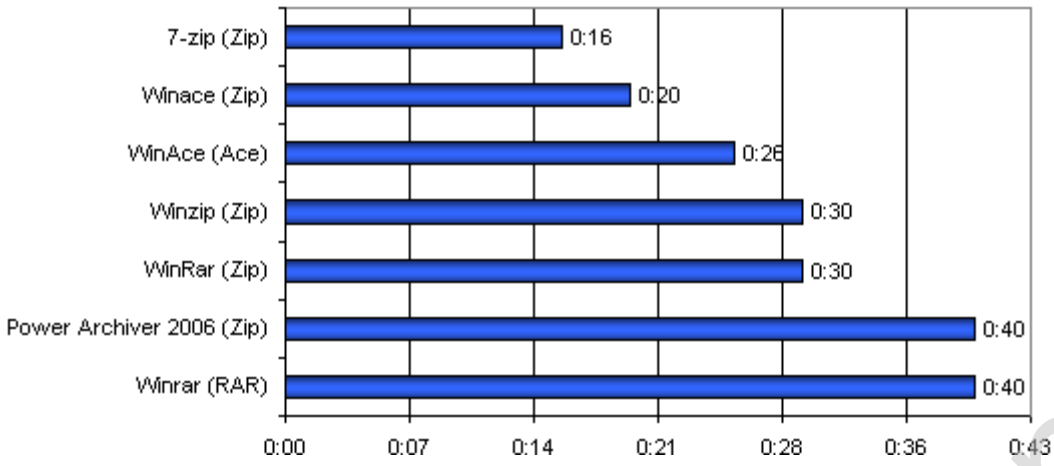

Рисунок 2.15 – Архівування текстових файлів з мінімальними налаштуваннями стиску, хв

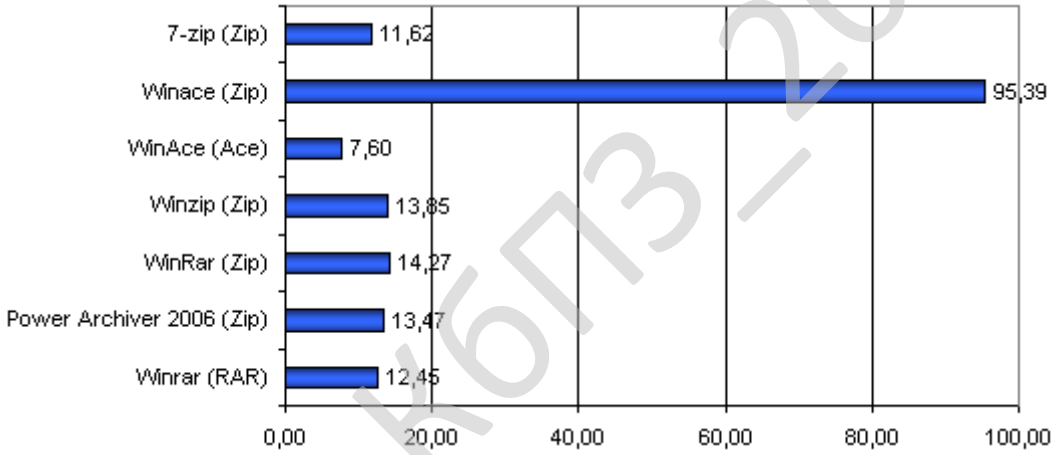

Рисунок 2.16 – Архівування текстових файлів з мінімальними налаштуваннями стиску, Мбайт

Проте, лідери тесту очевидні. Це Winrar і WinZip. Саме ці програми варто використовувати при стиску великих обсягів файлів (стосовно Winrar мова йде про стиск у формат RAR, а не Zip). Якщо Ви архівуєте великі обсяги даних для довгострокового збереження, має сенс витратити час і стиснути файли в обидва формати, а потім вибрати архів меншого розміру. Якщо ж такої можливості немає, краще використовувати формат RAR, тому що він у більшості випадків дає кращі результати. У цих програмних продуктах використовується алгоритм Зіва-Лемпела (LZW), тому подальші напрямки бакалаврського проектування будуть пов’язані саме з цим алгоритмом.

Кафедра_КБПЗ_2023рік

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

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

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

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

Кафедра_КБПЗ_2023рік

− 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.

Кафедра_КБПЗ_2023рік

– 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.

Кафедра_КОПЗ_2023рік

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.

Кафедра_КОПЗ_2023рік

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.

Кафедра_КОПЗ_2023рік

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне

Кафедра_КБПЗ_2023рік

забезпечення, яке призначено для системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

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

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

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

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

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

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

Кафедра_КБПЗ_2023рік

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис алгоритму стиснення інформації з підвищеною надійністю

Для реалізації програмного забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю пропонується використовувати алгоритм LZW-стиску та перешкодостійкий алгоритм Хеммінга.

Алгоритм LZW-стиску

Алгоритм LZW-стиску заміняє рядки символів деякими кодами. Це робиться без якого-небудь аналізу вхідного тексту. Замість цього при додаванні кожного нового рядка символів проглядається таблиця рядків. Стиск відбувається, коли код заміняє рядок символів. Коди, генеруємі LZW-алгоритмом, можуть бути будь-якої довжини, але вони повинні містити більше біт, ніж одиничний символ. Перші 256 кодів (коли використовуються 8-бітні символи) за замовчуванням відповідають стандартному набору символів. Інші коди відповідають оброблюваним алгоритмом рядкам.

Стиск

Алгоритм LZW-стиску в найпростішій формі наведений нижче. Щораз, коли генерується новий код, новий рядок додається в таблицю рядків. LZW постійно перевіряє, чи є рядок уже відомим, і , якщо так, виводить існуючий код без генерації нового.

Процедура LZW-стиску:

РЯДОК = черговий символ із вхідного потоку

WHILE вхідний потік не порожній DO

СИМВОЛ = черговий символ із вхідного потоку

IF РЯДОК+СИМВОЛ у таблиці рядків THEN

РЯДОК = РЯДОК+СИМВОЛ

Кафедра_КБПЗ_2023рік

ELSE

вивести у вихідний потік код для РЯДОК

додати в таблицю рядків РЯДОК+СИМВОЛ

РЯДОК = СИМВОЛ

END of IF

END of WHILE

вивести у вихідний потік код для РЯДОК

Простий рядок, використаний для демонстрації алгоритму, наведений в таблиці 3.1. Вхідний рядок є коротким списком англійських слів, розділених символом "/".

Вхідний рядок : /WED/WE/WEE/WEB/WET.

Таблиця 3.1 – Алгоритм стиску

Вхід(символи)	Вихід(коди)	Нові коди й відповідні рядки
/W	/	256 = /W
E	W	257 = WE
D	E	258 = ED
/	D	259 = D/
WE	256	260 = /WE
/	E	261 = E/
WEE	260	262 = /WEE
/W	261	263 = E/W
EB	257	264 = WEB
/	B	265 = B/
WET	260	266 = /WET
<EOF>	T	

Як ви можете помітити, аналізуючи алгоритм, його робота починається з того, що на першому кроці циклу він виконує перевірку на наявність рядка "/W" у

Кафедра_КбПЗ_2023рік

таблиці. Коли він не знаходить цей рядок, то генерує код для "/" і додає в таблицю рядок "/W". Т.к. 256 символів уже визначені для кодів 0-255, то першому певному рядку може бути поставлений у відповідність код 256. Після цього система читає наступну букву ("E"), додає другий підрядок ("WE") у таблицю й виводить код для букви "W".

Цей процес повторюється доти, поки другий підрядок, що складається із прочитаних символів "/" і "W", не зіставиться зі строковим номером 256. У цьому випадку система виводить код 256 і додає трьохсимвольний підрядок в таблицю. Цей процес триває доти, поки не вичерпається вхідний потік і все коди не будуть виведені.

Вихідний потік для заданого рядка показаний у таблиці 3.1, також як і отримана в результаті таблиця рядків. Як ви можете помітити, ця таблиця швидко заповнюється, тому що новий рядок додається в таблицю щораз, коли генерується код. У цьому явно виродженому прикладі було виведено п'ять закодованих підрядків і сім символів. Якщо використовувати 9-бітні коди для виводу, то 19-символьний вхідний рядок буде перетворений в 13.5-символьний вихідний рядок. Звичайно, цей приклад був обраний тільки для демонстрації. У дійсності стиск звичайно не починається доти, поки не буде побудована досить велика таблиця, звичайно після прочитання порядку 100 вхідних байт.

Розпакування

Алгоритму стиску відповідає свій алгоритм розпакування. Він одержує вихідний потік кодів від алгоритму стиску й використовує його для точного відновлення вхідного потоку. Однією із причин ефективності LZW-алгоритму є те, що він не має потреби в зберіганні таблиці рядків, отриманої при стиску. Таблиця може бути точно відновлена при розпакуванні на основі вихідного потоку алгоритму стиску. Це можливо тому, що алгоритм стиску виводить СТРОКОВІ й СИМВОЛЬНІ компоненти коду перш ніж він помістить цей код у вихідний потік. Це означає, що стислі дані не обтяжені необхідністю тягти за собою більшу таблицю перекладу.

Кафедра_КПЗ_2023рік

Алгоритм розпакування представлений на нижче. Відповідно до алгоритму стиску, він додає новий рядок у таблицю рядків щораз, коли читає із вхідного потоку новий код. Усе, що йому необхідно зробити в добавок – це перевести кожний вхідний код у рядок і переслати її у вихідний потік.

Процедура LZW-розпакування:

читати СТАРИЙ_КОД

вивести СТАРИЙ_КОД

WHILE вхідний потік не порожній DO

читати НОВИЙ_КОД

РЯДОК = перевести НОВИЙ_КОД

вивести РЯДОК

СИМВОЛ = перший символ РЯДКА

додати в таблицю перекладу СТАРИЙ_КОД+СИМВОЛ

СТАРИЙ_КОД = НОВИЙ_КОД

END of WHILE

Таблиця 3.2 – Алгоритм розпакування

Вхід НОВИЙ КОД	СТАРИЙ КОД	РЯДОК Вихід	СИМВОЛ	Новий вхід таблиці
/	/	/		
W	/	W	W	256 = /W
E	W	E	E	257 = WE
D	E	D	D	258 = ED
256	D	/W	/	259 = D/
E	256	E	E	260 = /WE
260	E	/WE	/	261 = E/
261	260	E/	E	262 = /WEE
257	261	WE	W	263 = E/W
B	257	B	B	264 = WEB
260	B	/WE	/	265 = B/
T	260	T	T	266 = /WET

Кафедра_КПЗ_2023рік

У таблиці 3.2 наведена схема роботи алгоритму на основі стислих даних, отриманих у вище наведеному прикладі. Важливо відзначити, що побудова таблиці рядків алгоритмом розпакування закінчується саме тоді, коли побудована таблиця рядків алгоритму стиску.

Вихідний потік ідентичний вхідному потоку алгоритму стиску. Відзначимо, що перші 256 кодів уже визначені для перекладу одиночних символів, також як і в алгоритмі стиску.

Проблеми

До нещастя алгоритм розпакування, наведений у таблиці 3.2, є все таким занадто простим. В алгоритмі стиску існують деякі виняткові ситуації, які створюють проблеми при розпакуванні. Якщо існує рядок, що представляє пари (РЯДОК СИМВОЛ) і вже певну в таблиці, а вхідний потік, що переглядається, містить послідовність РЯДОК СИМВОЛ РЯДОК СИМВОЛ РЯДОК, алгоритм стиску виведе код перш, ніж розпаковник одержить можливість визначити його.

Простий приклад ілюструє це. Припустимо, рядок "JOEYN" визначений у таблиці з кодом 300. Коли послідовність "JOEYNJOEYNJOEY" з'являється в таблиці, вихідний потік алгоритму стиску виглядає подібно тому, як показано в таблиці 3.3.

Вхідний рядок : ...JOEYNJOEYNJOEY...

Таблиця 3.3 – Кодування

Вхід(символи)	Вихід(коди)	Нові коди й відповідні рядки.
JOEYN	288 = JOEY	300 = JOEYN
A	N	301 = NA
.	.	.
.	.	.
.	.	.
JOEYNJ	300 = JOEYN	400 = JOEYNJ
JOEYNJO	400	401 = JOEYNJO

Кафедра_КПЗ_2023рік

Коли розпаковник переглядає вхідний потік, він спочатку декодує код 300, потім виводить рядок "JOEYN" і додає визначення для, скажемо, коду 399 у таблицю, хоча він уже міг там бути. Потім читає наступний вхідний код, 400, і виявляє, що його немає в таблиці. Це вже проблема. На щастя, це відбудеться тільки в тому випадку, якщо розпаковник зустріне невідомий код. Тому що це фактично єдина колізія, те можна без праці вдосконалити алгоритм.

Модифікований алгоритм передбачає спеціальні дії для ще невизначених кодів. У прикладі нижче розпаковник виявляє код 400, що ще не визначений. Тому що цей код не відомий, те декодується значення СТАРОГО_КОДУ, рівне 300. Потім розпаковник додає значення СИМВОЛУ, рівне "J", до рядка. Результатом є правильний переклад коду 400 у рядок "JOEYNJ".

Процедура LZW-розпакування:

```
читати СТАРИЙ_КОД
вивести СТАРИЙ_КОД
СИМВОЛ = СТАРИЙ_КОД
WHILE вхідний потік не порожній DO
        читати НОВИЙ_КОД
        IF NOT у таблиці перекладу НОВИЙ_КОД THEN
                РЯДОК = перевести СТАРИЙ_КОД
                РЯДОК = РЯДОК+СИМВОЛ
        ELSE
               РЯДОК = перевести НОВИЙ_КОД
        END of IF
        вивести РЯДОК
        СИМВОЛ = перший символ РЯДКА
        додати в таблицю перекладу СТАРИЙ_КОД+СИМВОЛ
        СТАРИЙ_КОД = НОВИЙ_КОД
END of WHILE
```

Кафедра_КБПЗ_2023рік

Реалізація

У програмі використовувалися коди довжиною 12, 13 і 14 біт. При довжині коду 12 біт потенційно можливо зберігати до 4096 рядків у таблиці. Щораз, коли читається новий символ, таблиця рядків повинна проглядатися для зіставлення. Якщо зіставлення не знайдене, новий рядок повинна бути додана в таблицю. Тут виникають дві проблеми. По-перше, таблиця рядків може досить швидко стати дуже великою. Навіть якщо довжина рядків у середньому обмежується 3 або 4 символами кожна, верхня межа довжин рядків може легко перевищити 7 або 8 байт на код. До того ж кількість пам'яті, необхідної для зберігання рядків, заздалегідь не відомо, тому що воно залежить від загальної довжини рядків.

Друга проблема полягає в організації пошуку рядків. Щораз, коли читається новий символ, необхідно організувати пошук для нового рядка виду РЯДОК+СИМВОЛ. Це означає підтримку відсортованого списку рядків. У цьому випадку пошук для кожного рядка включає число порівнянь порядку log2 від загального числа рядків. Використання 12-бітних слів потенційно дозволяє виконувати не більше 12 порівнянь для кожного коду.

Перша проблема може бути вирішена зберіганням рядків як комбінацій код/символ. Тому що кожний рядок у дійсності є поданням комбінації вже існуючого коду й додаткового символу, можна зберігати кожний рядок як окремий код плюс символ. Наприклад у розібраному вище прикладі рядок "/WEE" зберігається як код 260 і символ "E". Це дозволяє використовувати для зберігання тільки 3 байти замість 5 (включающих додатковий байт для кінця рядка). Ідучи назад, можна визначити, що код 260 зберігається як код 256 плюс додатковий символ "E". Нарешті, код 256 зберігається як "/" плюс "W".

Виконання порівняння рядків є небагато більше важким. Новий метод зберігання збільшує час, необхідне для порівняння рядків, але він не впливає на число порівнянь. Ця проблема вирішується використанням алгоритму хешування для зберігання рядків. Це означає, що код 256 не зберігається в якому-небудь

Кафедра_КОПЗ_2023рік

масиві за адресою 256, а зберігається в масиві за адресою, сформованому на основі самого рядка. При визначенні місця зберігання даного рядка можна використовувати тестовий рядок для генерації хеш-адреси й потім знайти цільовий рядок однократним порівнянням. Тому що код для будь-якого даного рядка не можна довідатися надалі інакше як по його позиції в масиві, необхідно зберігати код для даного рядка разом з даними рядка. У демонстраційній програмі для цього використовуються елементи трьох масивів : code_value[i], prefix_code[i] і append_character[i].

Коли необхідно додати новий код у таблицю, використовується хеш-функція в процедурі find_match для генерації коректного і. Процедура find_match генерує адресу й потім перевіряє, чи не використовувався він уже. Якщо це так, то find_match виконує другу пробу й так доти, поки не найдеться вільне місце.

Хеш-функція, використана в цій програмі – проста "хог"-типу хеш-функція. Префікс коду й додатковий символ комбінуються для формування адреси масиву. Якщо вміст префікса коду й символ у масиві зіставляються їм, то вертається коректна адреса. Якщо елемент масиву по цій адресі вже використаний, виконується фіксований зсув для пошуку нового місця. Це виконується доти, поки не буде знайдені вільне місце або не відбудеться зіставлення. Середнє число пошуків у такій таблиці – менше 3, якщо використовується таблиця на 25% більшого розміру, ніж необхідно. Воно може бути поліпшене шляхом збільшення розміру таблиці. Необхідно відзначити, що для того, щоб порядок вторинних проб працював, розмір таблиці повинен бути простим числом. Це пояснюється тим, що проба може бути будь-яким цілим між 1 і розміром таблиці. Якщо проба й розмір таблиці не є взаємно простими, пошук вільних місць може закінчитися невдачею, навіть якщо вони є.

Реалізація алгоритму розпакування має свій набір проблем. Одна із проблем алгоритму стиску тут зникає. Коли виконується стиск, необхідно організувати пошук у таблиці для даного рядка. При розпакуванні необхідно організувати перегляд для окремого коду. Це означає, що можна зберігати

Кафедра_КОПЗ_2023рік

префікси кодів і додаткові символи, індексуючись по їхньому строковому коді. Це усуває необхідність у хеш-функції й звільняє масив, що використовувався для зберігання значень кодів.

На жаль метод, використаний для зберігання строкових величин, приводить до того, що декодування рядків повинне виконуватися в інверсному порядку. Це значить, що всі символи для даного рядка при декодуванні повинні міститися в стековий буфер, а потім виводитися у зворотному порядку. У наведеній програмі це виконується функцією decode_string.

Проблема з'являється, коли читання вхідного потоку переривається при досягненні кінця потоку. Для цієї частки випадку в програмі зарезервований останній обумовлений код MAX_VALUE як ознака кінця даних. Це не є необхідним при читанні файлу, але може допомогти при читанні буфера стислих даних з пам'яті. Витрати на втрату одного обумовленого коду досить малі порівняно з усім процесом.

Результати

Досить важко охарактеризувати результативність якої-небудь техніки стиску даних. Ступінь стиску визначається різними факторами. LZW-стиск виділяється серед інших, коли зустрічається з потоком даних, що містять повторювані рядки будь-якої структури. Із цієї причини він працює досить ефективно, коли зустрічає англійський текст. Рівень стиску може досягати 50% і вище. Відповідно, стиск відеоформ і копій екранів показує ще кращі результати.

Труднощі при стиску файлів даних трохи більші. Залежно від даних, результат стиску може бути як гарним, так і не дуже задовільним. У деяких випадках "стислий" файл може перевершувати по своїх розмірах вихідний текст. Невеликий експеримент дасть Вам подання про те, добре або погано впаковуються Ваші дані.

Однією із проблем є те, що наведена програма не адаптується до різної довжини файлів. Використання 14– або 15-бітних кодів дає кращий ступінь стиску на великих файлах (це пояснюється тим, що для них будуються більші

Кафедра_КОПЗ_2023рік

таблиці рядків), але гірше працює з маленькими файлами. Такі програми, як "ARC", вирішують цю проблему використанням кодів змінної довжини. Наприклад, коли величина next_code перебуває між 256 і 511, "ARC" читає й виводить 9-бітні коди. Коли величина next_code стає настільки великий, що необхідно 10-бітні коди, процедури стиску й розпакування збільшують розмір коду. Це значить, що 12– і 15-бітні варіанти програми працюють добре й на маленьких файлах.

Іншою проблемою великих файлів є те, що зі збільшенням числа прочитаних байтів ступінь стиску може почати погіршуватися. Причина проста: тому що розмір таблиці рядків фіксований, після занесення певного числа рядків їх уже просто нікуди додати. Але побудована таблиця потрібна тільки для тої частини файлу, по якій вона була побудована. Частини, що залишилися, файлу можуть мати інші характеристики й у дійсності потрібна вже трохи відмінна таблиця.

Звичайним способом рішення цієї проблеми є контроль ступеня стиску. Після того, як таблиця рядків заповнена, пакувальник стежить за поводженням коефіцієнта стиску. Після певного ступеня його погіршення таблиця рядків очищається й починає будуватися заново.

Процедура розпакування визначає цей момент тим, що пакувальник записує у свій вихідний потік спеціальний код. Альтернативним способом є визначення найбільш часте рядків, що зустрічаються, і чищення інших. Адаптивна техніка, подібна цієї, може, однак, зустріти труднощі реалізації в програмах розумного розміру.

І, нарешті, можна брати вироблювані LZW-методом коди й пропускати їх через кодуючий фильтр Хаффмана, що адаптується,. Це дає трохи більший ступінь стиску, але вартість такої роботи більше високий, також як і час обробки.

Для підвищення надійності зберігання даних, при ушкодженні архівного файлу пропонується використовувати алгоритм Хеммінга.

Кафедра_КОПЗ_2023рік

Алгоритм Хеммінга

Коди Хеммінга є кодами, що самоконтролюються, тобто кодами, що дозволяють автоматично виявляти найбільш імовірні помилки при передачі даних. Для їхньої побудови досить приписати до кожного слова один додатковий (контрольний) двійковий розряд і вибрати цифру цього розряду так, щоб загальна кількість одиниць у зображенні будь-якого числа була, наприклад, парнею. Одиночна помилка в якому-небудь розряді переданого слова (у тому числі, може бути, і в контрольному розряді) змінить парність загальної кількості одиниць. Лічильники за модулем 2, що підраховують кількість одиниць, які втримуються серед двійкових цифр числа, можуть давати сигнал про наявність помилок.

При цьому, зрозуміло, ми не одержуємо ніяких вказівок про те, у якому саме розряді відбулася помилка, і, отже, не маємо можливості виправити неї. Залишаються непоміченими також помилки, що виникають одночасно у двох, у чотирьох або взагалі в парній кількості розрядів. Втім, подвійні, а тим більше чотириразові помилки покладаються малоймовірними.

Коди, що самокоректуються

Коди, у яких можливо автоматичне виправлення помилок, називаються що самокоректуються. Для побудови коду, що самокоректується, розрахованого на виправлення одиночних помилок, одного контрольного розряду недостатньо. Як видно з подальшого, кількість контрольних розрядів k повинне бути обране так, щоб задовольнялося нерівності:

$$2^k \geq k + m + 1,$$

або:

$$k \geq \log_2(k + m + 1),$$

де m – кількість основних двійкових розрядів кодового слова. Мінімальні значення k при заданих значеннях m, знайдені відповідно до цієї нерівності, наведені в таблиці 3.4.

Кафедра_КБПЗ_2023рік

Таблиця 3.4 – Мінімальні значення k при заданих значеннях m

Діапазон m	k_{min}
1	2
2-4	3
5-11	4
12-26	5
27-57	6

Маючи m+k розрядів, що самокоректується код можна побудувати наступним чином.

Привласнимо кожному з розрядів свій номер – від 1 до m+k; запишемо ці номери у двійковій системі числення. Оскільки $2^k > m + k$,кожний номер можна представити, мабуть, k-розрядним двійковим числом.

Припустимо далі, що всі m+k розрядів коду розбиті на контрольні групи, які частково перекриваються, причому так, що одиниці у двійковому поданні номера розряду вказують на його приналежність до певних контрольних груп. Наприклад: розряд № 5 належить до 1-й і 3-й контрольним групам, тому що у двійковому поданні його номера $5_{10} = \ldots 000101_2$ – 1-й і 3-й розряди містять одиниці.

Серед m+k розрядів коду при цьому є k розрядів, кожний з яких належить тільки до однієї контрольної групи:

Розряд № 1: $1_{10} = \ldots 000001_2$ належить тільки до 1-й контрольної групи.

Розряд № 2: $2_{10} = \ldots 000010_2$ належить тільки до 2-й контрольній групі.

Розряд № 4: $4_{10} = \ldots 000100_2$ належить тільки до 3-й контрольній групі.

…

Розряд № 2^{k-1} належить тільки до k-*й* контрольної групи.

Ці k розрядів ми й будемо вважати контрольними. Інші m розрядів, кожний з яких належить, щонайменше, до двох контрольних груп, будуть інформаційними розрядами.

Кафедра_КОТЗ_2023рік

У кожній з k контрольних груп будемо мати по одному контрольному розряді. У кожний з контрольних розрядів помістимо таку цифру (0 або 1), щоб загальна кількість одиниць у його контрольній групі було парним.

Наприклад, досить розповсюджений код Хеминга з m=7 і k=4.

Нехай вихідне слово (кодове слово без контрольних розрядів) – 0110101_2.

Позначимо P_i – контрольний розряд №i; а D_i – інформаційний розряд №i, де i = 1,2,3,4…

Таблиця 3.5 – Параметри й значення коду

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7
Інформаційне кодове слово :			0		1	1	0		1	0	1
p_1	1		0		1		0		1		1
p_2		0	0			1	0			0	1
p_3				0	1	1	0				
p_4								0	1	0	1
Кодове слово з контрольними розрядами:	1	0	0	0	1	1	0	0	1	0	1

Цікаво подивитися, як перекриваються контрольні групи в цьому випадку. Перша група контролює розряди № 3,5,7,9,11 вихідні коди, друга – 3,6,7,10,11 (№ групи = № контрольного розряду). Очевидно, що вони частково перекриваються.

Припустимо тепер, що даного кодового слова 10001100101 відбулася помилка в 11 -м розряді, так, що було прийнято нове кодове слово 10001100100. Зробивши в прийнятому коді перевірку парнесті усередині контрольних груп, ми виявили б, що кількість одиниць непарне в 1-й, 2-й і 4-й контрольних групах, і парне в 3-й контрольній групі. Це вказує, по-перше, на наявність помилки, по-друге, означає, що номер помилково прийнятого розряду у двійковому поданні

Кафедра_КБПЗ_2023рік

містить одиниці на першому, другому й четвертому місцях і нуль – на третім місці, т.до помилка тільки одна, і 3-я контрольна сума виявилася вірною.

Таблица 3.6 – Параметри коду при помилці

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011		
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	Контроль по парнесті в групі	Контрольний біт
Передане кодове слово:	1	0	0	0	1	1	0	0	1	0	1		
Прийняте кодове слово:	1	0	0	0	1	1	0	0	1	0	0		
p_1	1		0		1		0		1		0	Fail	1
p_2		0	0			1	0			0	0	Fail	1
p_3				0	1	1	0					Pass	0
p_4								0	1	0	0	Fail	1

Таблиця 3.7 – Виявлення помилки

	p_4	p_3	p_2	p_1	
У двійковому поданні	1	0	1	1	
У десятковому поданні	8		2	1	$\Sigma = 11$

З таблиці видно, що помилка відбулася в 11-м розряді і її можна виправити. Побудований код, зрозуміло, не розрахований на можливість одночасної помилки у двох розрядах.

Наприклад, коли помилки одночасно пройшли в 3-м і 7-м розрядах вихідного коду, перші й другий контрольні біти навіть не помітять підміни.

Кафедра_КБПЗ_2023рік

Коли в прийнятому коді виробляється перевірка парнесті усередині контрольних груп, випадок подвійної помилки нічим зовні не відрізняється від випадку одиночної помилки.

Код Хеммінга

Можна побудувати й такий код, що виявляв би подвійні помилки й виправляв одиночні. Для цього до коду, що самокоректується, розрахованому на виправлення одиночних помилок, потрібно приписати ще один контрольний розряд (розряд подвійного контролю). Повна кількість розрядів коду при цьому буде m+k+1. Цифра в розряді подвійного контролю встановлюється такий, щоб загальна кількість одиниць у всіх $m + k + 1$ розрядах коду було парним. Цей розряд не включається в загальну нумерацію й не входить у жодну контрольну групу.

Наприклад, код Хеминга з m=7 і k=4 Нехай інформаційне кодове слово – 0110101

Таблиця 3.8 – Параметри коду

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	Second Parity
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	
Інформаційне кодове слово:			0		1	1	0		1	0	1	
p_1	1		0		1		0		1		1	
p_2		0	0			1	0			0	1	
p_3				0	1	1	0					
p_4								0	1	0	1	
Кодове слово з контрольними розрядами:	1	0	0	0	1	1	0	0	1	0	1	1

При цьому можуть бути наступні випадки.

Кафедра_КБПЗ_2023рік

1. У прийнятому коді в цілому й по всіх контрольних групах кількість одиниць парне. Якщо потрійні помилки й помилки в більшій кількості розрядів виключаються, то перший випадок відповідає безпомилковому прийому коду.

Таблиця 3.9 – Параметри коду при помилці

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	Контроль по парності в групі	Контрольний біт	Контроль по парності в цілому	Контрольний біт у цілому
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7				
Передане кодове слово:	1	0	0	0	1	1	0	0	1	0	1				
Прийняте кодове слово:	1	0	0	0	1	1	0	0	1	0	1				
p_1	1		0		1		0		1		1	Pass	0		
p_2		0	0			1	0			0	1	Pass	0		
p_3				0	1	1	0					Pass	0		
p_4								0	1	0	1	Pass	0	1	Pass

Таблиця 3.10 – Виявлення помилки

	p_4	p_3	p_2	p_1	
У двійковому поданні	0	0	0	0	
У десятковому поданні					$\Sigma = 0$

Кафедра_КбПЗ_2023рік

2. У прийнятому коді в цілому кількість одиниць непарне, але у всіх контрольних групах кількість одиниць парне. Другий випадок – помилки тільки в розряді подвійного контролю.

Таблиця 3.11 – Параметри коду при помилці

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	Контроль по парнесті в групі	Контрольний біт	Контроль по парнесті в цілому	Контрольний біт у цілому
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7				
Передане кодове слово:	1	0	0	0	1	1	0	0	1	0	1				
Прийняте кодове слово:	1	0	0	0	1	1	0	0	1	0	1				
p_1	1		0		1		0		1		1	Pass	0		
p_2		0	0			1	0			0	1	Pass	0		
p_3				0	1	1	0					Pass	0		
p_4								0	1	0	1	Pass	0	0	Fail

Таблиця 3.12 – Виявлення помилки

	p_4	p_3	p_2	p_1	
У двійковому поданні	0	0	0	0	
У десятковому поданні					$\Sigma = 0$

Кафедра_КОПЗ_2023рік

3. У прийнятому коді в цілому й у деяких з контрольних груп кількість одиниць непарне. Третій випадок – одиночної помилки в якому-небудь із інших розрядів (можна виправити відповідно до наведеного вище правилами).

Таблиця 3.13 – Параметри коду при помилці

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	Контроль по парності в групі	Контрольний біт	Контроль по парності в цілому	Контрольний біт у цілому
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7				
Передане кодове слово :	1	0	0	0	1	1	0	0	1	0	1				
Прийняте кодове слово:	1	0	0	0	1	1	0	0	1	0	0				
p_1	1		0		1		0		1		0	Epic Fail	1		
p_2		0	0			1	0			0	0	Fail	1		
p_3				0	1	1	0					Pass	0		
p_4								0	1	0	0	Fail	1	1	Fail

Таблиця 3.14 – Виявлення помилки

	p_4	p_3	p_2	p_1	
У двійковому поданні	1	0	1	1	
У десятковому поданні	8		2	1	$\Sigma = 11$

З таблиці видно, що помилка відбулася в 11-м розряді й що її можна виправити.

Кафедра_КБПЗ_2023рік

4. У прийнятому коді в цілому кількість одиниць парне, але в деяких контрольних групах є непарна кількість одиниць – подвійна помилка.

Таблиця 3.15 – Параметри коду при помилці

№ розряду	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	Контроль по парності в групі	Контрольний біт	Контроль по парності в цілому	Контрольний біт у цілому
Розподіл контрольних і інформаційних розрядів	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7				
Передане кодове слово:	1	0	0	0	1	1	0	0	1	0	1				
Прийняте кодове слово:	1	0	1	0	1	0	0	0	1	0	1				
p_1	1		1		1		0		1		1	Fail	1		
p_2		0	1			0	0			0	1	Pass	0		
p_3				0	1	0	0					Fail	1		
p_4								0	1	0	1	Pass	0	1	Pass

Таблиця 3.16 – Виявлення помилки

	p_4	p_3	p_2	p_1	
У двійковому поданні	0	1	0	1	
У десятковому поданні		4		1	$\Sigma = 5$

Раз сума, що вийшла, не дорівнює нулю, а контрольний біт указує на помилку передачі, те виявляємо подвійну помилку. Виправлення подвійних помилок тут, звичайно, неможливо.

Кафедра КОПЗ 2023рік

Збільшуючи далі кількість контрольних розрядів, можна було б побудувати коди, розраховані на виправлення подвійних помилок і виявлення потрійних і т.д. Однак методи побудови цих кодів не цілком розроблені.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленого програмного забезпечення.

На ній відображено процес архівування та розархівування інформації за алгоритмом LZW, та підданя заархівованої інформації перешкодостійкому кодуванню за алгоритмом Хеммінга.

Ці алгоритми більш детально розглянуті у розділі 3.1.

Якщо коротко, то алгоритм LZW діє наступним чином.

Даний алгоритм при стиску (кодуванні) динамічно створює таблицю перетворення рядків: певним послідовностям символів (словам) ставляться у відповідність групи біт фіксованої довжини (звичайно 12-бітні). Таблиця ініціалізується всіма 1-символьними рядками (у випадку 8-бітних символів – це 256 записів). У міру кодування, алгоритм переглядає текст символ за символом, і зберігає кожну нову, унікальну 2-символьний рядок у таблицю у вигляді пари код/символ, де код посилається на відповідний перший символ. Після того як нова 2-символьний рядок збережений у таблиці, на вихід передається код першого символу. Коли на вході читається черговий символ, для нього по таблиці перебуває вже, що зустрічався рядок, максимальної довжини, після чого в таблиці зберігається код цього рядка з наступним символом на вході; на вихід видається код цього рядка, а наступний символ використовується в якості початку наступного рядка.

Кафедра_КОПЗ_2023рік

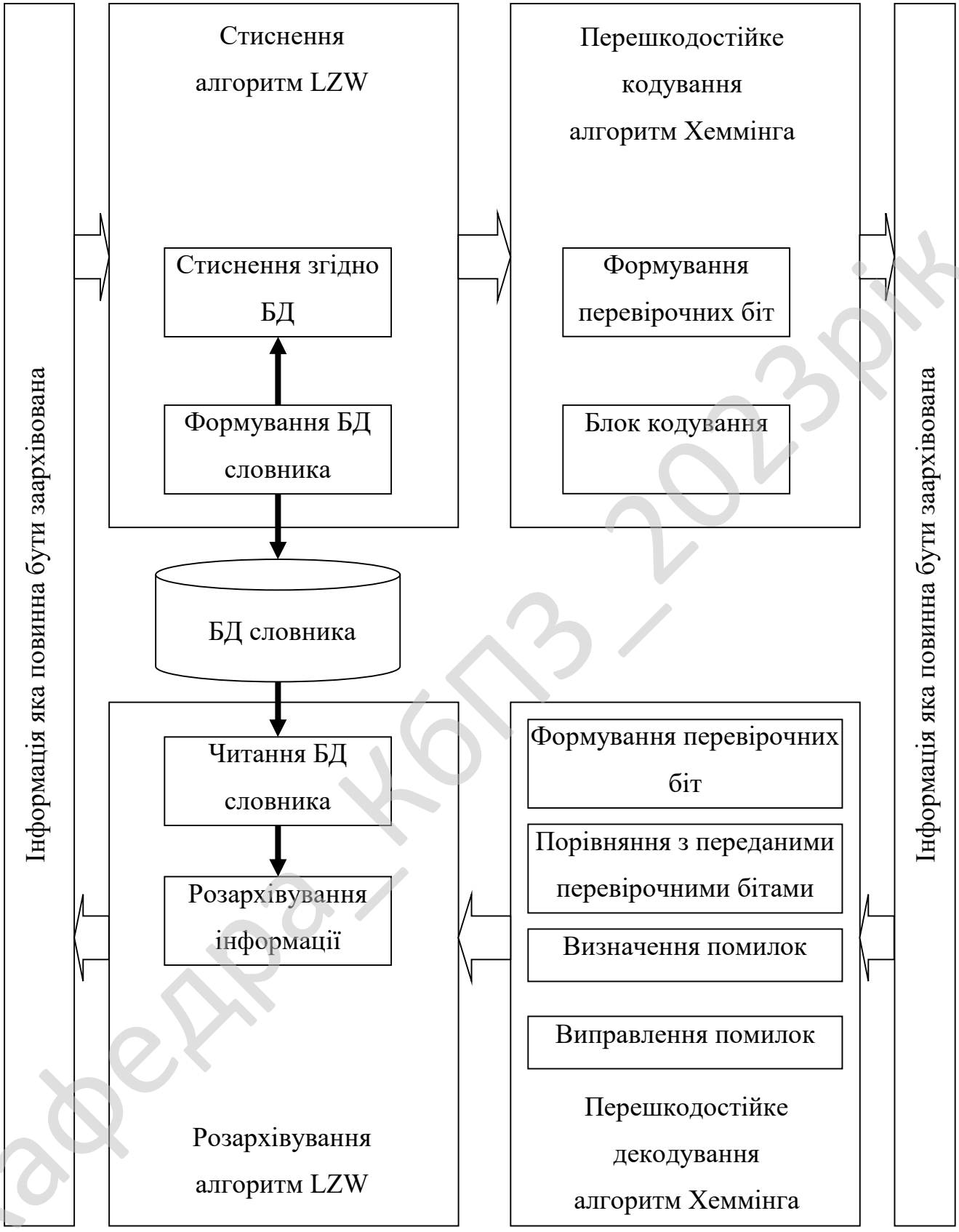

Рисунок 3.1 – Структурна схема системи

Кафедра_КБПЗ_2023рік

Алгоритму декодування на вході потрібно тільки закодований текст, оскільки він може відтворити відповідну таблицю перетворення безпосередньо по закодованому тексту.

Алгоритм LZW

– Ініціалізація словника всіма можливими односимвольними фразами. Ініціалізація вхідної фрази w першим символом повідомлення.

– Зчитати черговий символ K з кодуємого повідомлення.

– Якщо КІНЕЦЬ_ПОВІДОМЛЕННЯ, то видати код для w, інакше

– Якщо фраза w уже є в словнику, привласнити вхідній фразі значення w і перейти до Кроку 2, інакше видати код w, додати w у словник, привласнити вхідній фразі значення K і перейти до Кроку 2.

Кінець.

Алгоритм Хеммінга

Алгоритм Хеммінга діє наступними чином.

Код Хеммінга являє собою блоковий код, що дозволяє виявити й виправити помилково переданий біт у межах переданого блоку. Звичайно код Хеммінга характеризується двома цілими числами, наприклад, (11,7) використовуваний при передачі 7-бітних ASCII-кодів. Такий запис говорить, що при передачі 7-бітного коду використовується 4 контрольних біта (7+4=11). При цьому передбачається, що мала місце помилка в одному біті й що помилка у двох або більше бітах істотно менш імовірна. З обліком цього виправлення помилки здійснюється з певною ймовірністю. Наприклад, нехай можливі наступні правильні коди (всі вони, крім перш і останнього, відстоять друг від друга на відстань 4):

00000000

11110000

00001111

11111111

Кафедра_КБПВ_2023рік

При одержанні коду 00000111 не важко припустити, що правильне значення отриманого коду дорівнює 00001111. Інші коди відстоять від отриманого на більшу відстань Хеммінга.

Розглянемо приклад передачі коду букви s = 0x073 = 1110011 з використанням коду Хеммінга (11,7).

Таблиця 3.17 – Структура повідомлення по коду Хеммінга

Позиція біта:	11	10	9	8	7	6	5	4	3	2	1
Значення біта:	1	1	1	*	0	0	1	*	1	*	*

Символами * позначені чотири позиції, де повинні розміщатися контрольні біти. Ці позиції визначаються цілим ступенем 2 (1, 2, 4, 8 і т.д.). Контрольна сума формується шляхом виконання операції XOR (виключаєче АБО) над кодами позицій ненульових біт. У цьому випадку це 11, 10, 9, 5 і 3. Обчислимо контрольну суму.

Таблиця 3.18 – Контрольна сума

11 =	1011
10 =	1010
09 =	1001
05 =	0101
03 =	0011
S =	1110

Таким чином, приймач одержить наступний код.

Таблиця 3.19 – Отриманий код

Позиція біта:	11	10	9	8	7	6	5	4	3	2	1
Значення біта:	1	1	1	1	0	0	1	1	1	1	0

Кафедра_КБПЗ_2023рік

Просумуємо знову коди позицій ненульових біт і одержимо нуль.

Таблиця 3.20 – Сума кодів позицій ненульових біт

11 =	1011
10 =	1010
09 =	1001
08 =	1000
05 =	0101
04 =	0100
03 =	0011
02 =	0010
S =	0000

Ну а тепер розглянемо два випадки помилок в одному з біт посилки, наприклад, у біті 7 (1 замість 0) і в біті 5 (0 замість 1). Просумуємо коди позицій ненульових біт ще раз.

Таблиця 3.21 – Сума кодів позицій ненульових біт

11 =	1011
10 =	1010
09 =	1001
08 =	1000
07 =	0111
05 =	0101
04 =	0100
03 =	0011
02 =	0010
S =	0111

11 =	1011
10 =	1010
09 =	1001
08 =	1000
04 =	0100
03 =	0011
02 =	0010
S =	0001

В обох випадках контрольна сума дорівнює позиції біта, переданого з помилкою. Тепер для виправлення помилки досить інвертувати біт, номер якого

Кафедра_КБПЗ_2023рік

зазначений у контрольній сумі. Зрозуміло, що якщо помилка відбудеться при передачі більш ніж одного біта, код Хеммінга при даній надмірності виявиться марний.

У загальному випадку код має N=M+C біт і передбачається, що не більш ніж один біт у коді може мати помилку. Тоді можливо N+1 стан коду (правильний стан і n помилкових). Нехай M=4, а N=7, тоді слово-повідомлення буде мати вигляд: M4, M3, M2, C3, M1, C2, C1. Тепер спробуємо обчислити значення C1, C2, C3. Для цього використовуються рівняння, де всі операції являють собою додавання за модулем 2:

$$C1 = M1 + M2 + M4$$

$$C2 = M1 + M3 + M4$$

$$C3 = M2 + M3 + M4$$

Для визначення того, чи доставлене повідомлення без помилок, обчислюємо наступні вираження (додавання за модулем 2):

$$C11 = C1 + M4 + M2 + M1$$

$$C12 = C2 + M4 + M3 + M1$$

$$C13 = C3 + M4 + M3 + M2$$

Результат обчислення інтерпретується в такий спосіб.

Таблиця 3.21 – Результат обчислення

C11	C12	C13	Значення
1	2	4	Позиція біт
0	0	0	Помилок немає
0	0	1	Біт C3 не вірний
0	1	0	Біт C2 не вірний
0	1	1	Біт M3 не вірний
1	0	0	Біт C1 не вірний
1	0	1	Біт M2 не вірний
1	1	0	Біт M1 не вірний
1	1	1	Біт M4 не вірний

Кафедра_КБПЗ_2023рік

Описана схема легко переноситься на будь-яке число n і M.

Число можливих кодових комбінацій M завадостійкого коду ділиться на n класів, де N – число дозволених кодів. Поділ на класи здійснюється так, щоб у кожний клас увійшов один дозволений код і найближчі до нього (по відстані Хеммінга) заборонені коди. У процесі прийому даних визначається, до якого класу належить код, що прийшов. Якщо код прийнятий з помилкою, він заміняється найближчим дозволеним кодом. При цьому передбачається, що кратність помилки не більше q_m.

Можна довести, що для виправлення помилок із кратністю не більше q_mm (як правило, воно вибирається рівним $D = 2q_m + 1$). У теорії кодування існують наступні оцінки максимального числа N n-розрядних кодів з відстанню D.

Таблиця 3.22 – Визначення кількості помилок в залежності від кодової відстані

d=1	$n=2^n$
d=2	$n=2\ n^{-1}$
d=3	$n\ 2^n /(1+n)$
d=2q+1	$N \le 2^n (1 + \sum_{i=1}^{d} C_n^i)^{-1}$ (для коду Хеммінга ця нерівність перетворюється в рівність)

У випадку коду Хеммінга перші k розрядів використовуються в якості інформаційних, причому k= n – log(n+1), звідки випливає (логарифм по підставі 2), що k може приймати значення 0, 1, 4, 11, 26, 57 і т.д., це й визначає відповідні коди Хеммінга (3,1); (7,4); (15,11); (31,26); (63,57) і т.д.

3.3 Розробка функціональної схеми

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

Кафедра_КбПЗ_2023рік

Основними функціями розробленого програмного забезпечення є наступні:

– Вибір ступеню стиску файлів.
– Блок формування багатотомних архівів.
– Блок добування інформації з багатотомних архівів.
– Блок формування архівів, що саморозпаковуються.
– Вибір режиму solid.
– Блок виводу моніторингу роботи архіватора.
– Блок вибору способу керування програмою.
– Блок вибору мови інтерфейсу.
– Блок допомоги.

Розглянемо ці блоки більш детальніше.

Великі за обсягом архівні файли можуть бути розміщені на декількох дисках (томах). Такі архіви називаються багатотомними. Том – це складова частина багатотомного архіву.

Програми-архіватори дозволяють створювати й такі архіви, для добування з яких файлів, що втримуються в них, не потрібні які-небудь програми, тому що самі архівні файли можуть містити програму розпакування. Такі архівні файли називаються такими, що саморозпаковуються.

Архівний файл, що саморозпаковується, – це завантажувальний модуль, що виконується, який здатний до самостійного розархівації файлів, що перебувають у ньому, без використання програми-архіватора.

Архів, що саморозпаковується, одержав назву SFX-архів (Self-eXtracting). Архіви такого типу в MS DOS звичайно створюються у формі .exe-файлу.

Багато програм-архіваторів роблять розпакування файлів, вивантажуючи їх на диск, але є й такі, які призначені для створення впакованого модуля, що виконується, (програми). У результаті такого впакування створюється програмний файл із тими ж ім'ям і розширенням, що при завантаженні в оперативну пам'ять саморозпаковується й відразу запускається. Разом з тим

Кафедра_КОПЗ_2023рік

можливо й зворотне перетворення програмного файлу в розпакований формат.

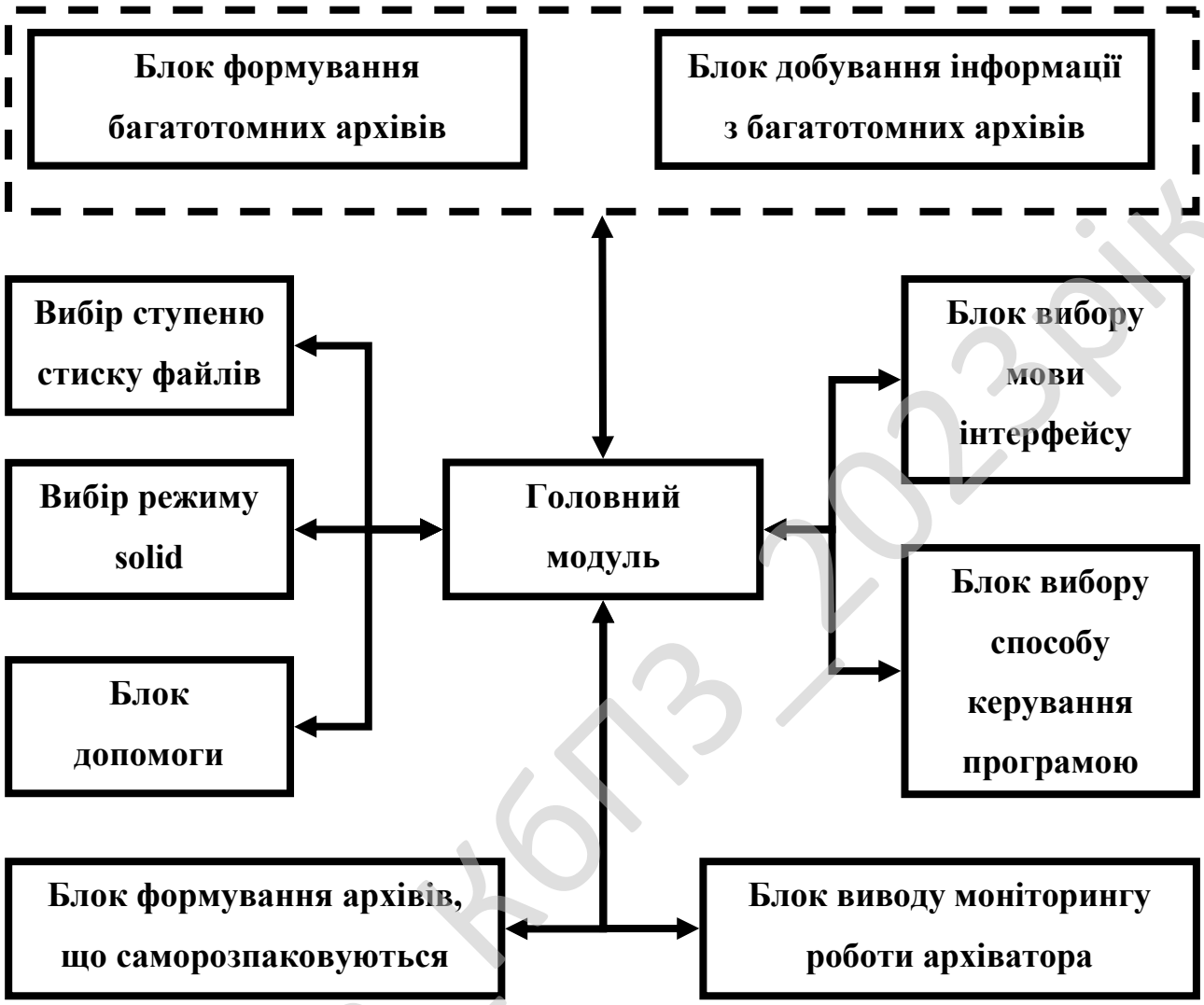

Рисунок 3.2 – Функціональна схема розробленої системи

Програми-архіватори крім звичайного режиму стиску, мають режим solid, у якому створюються архіви з підвищеним ступенем стиску й особою структурою організації. У таких архівах всі файли стискуються як один потік даних, тобто областю пошуку повторюваних послідовностей символів є вся сукупність файлів, завантажених в архів, і тому розпакування кожного файлу, якщо він не перший, пов'язана з обробкою інших. Архіви такого типу переважніше використовувати для архівування великої кількості однотипних файлів.

Кафедра_КбПЗ_2023рік

Способи керування програмою-архіватором

Керування програмою-архіватором здійснюється одним із двох способів:

– за допомогою командного рядка MS DOS, у якій формується команда запуску, що містить ім'я програми-архіватора, команду керування й ключі її налаштування, а також імена архівного й вихідного файлів;

– за допомогою убудованої оболонки й діалогових панелей, що з'являються після запуску програми й дозволяють вести керування з використанням меню й функціональних клавіш, що створює для користувача більше комфортні умови роботи.

Виконуючи запропоновані їй дії, програма-архіватор, як правило, виводить на екран протокол своєї роботи. Всі сучасні програми-архіватори оснащені екранами допомоги, які викликаються при уведенні в командному рядку тільки одного ім'я програми або ім'я із ключем /?. Допомога може бути короткої – на одному екрані або розгорнутої – на декількох. Багато хто архіватори мають екрани допомоги із прикладами складання команд для виконання різних операцій. Інформація допомоги звичайно виводиться на англійській або іншій міжнародній мові.

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

3.4 Розробка діаграми процесів

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

З нього ми бачимо, що процеси в системі взаємодіють наступним чином.

Спершу запускається процес початку/кінця рботи програми.

Він взаємодіє з процесом вибору файлів/архівів.

Процес вибору файлів/архівів, взаємодіє з наступними процесами:

– Процесом створення архіву.

Кафедра_КПЗ_2023рік

– Процесом вилучення файлів з архіву.

Розглянемо з якими процесами взаємодіє процес створення архіву.

Він взаємодіє з процесом стиснення алгоритмом LZW.

Процес стиснення алгоритмом LZW взаємодіє з наступними процесами:

– Процес формування бази даних словника.

– Процес стиснення згідно бази даних словника.

Ці два процеси взаємодіють між собою.

Крім того, процес стиснення алгоритмом LZW також взаємодіє з процесом кодування алгоритмом Лемінга, який у свою чергу, взаємодіє з процесом формування перевірочних біт.

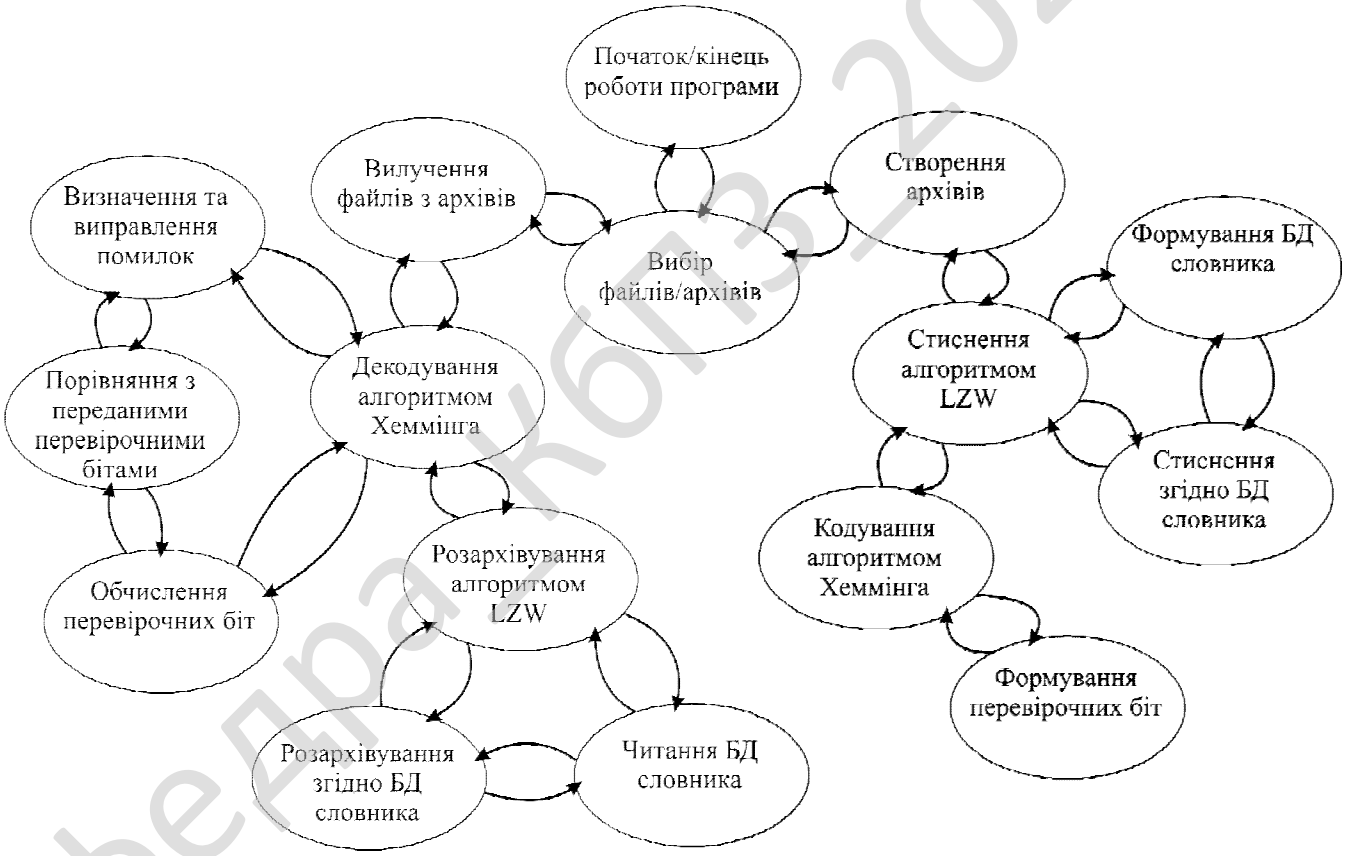

Рисунок 3.3 – Діаграма процесів розробленої системи

Перейдемо до розгляду того, з якими процесами взаємодіє процес вилучення файлів з архівів.

Кафедра КБПЗ – 2023рік

Процес вилучення файлів з архівів взаємодіє з процесом декодування алгоритму Хеммінга.

Цей процес взаємодіє з наступними процесами:

– Процес визначення та виправлення помилок.

– Процес обчислення перевірочних біт.

– Процес розархівування алгоритм LZW.

Процеси визначення та виправлення помилок та процес обчислення перевірочних біт, взаємодіють з процесом порівняння з переданими перевірочними бітами.

Процес розархівування алгоритм LZW взаємодіє з наступними процесами:

– Процес розархівування згідно бази даних словника.

– Процес читання бази даних словника.

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.

Кафедра_КбПЗ_2023рік

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми.

Після цього користувач обирає функцію створення архіву.

Функція створення архіву складається з виконання наступних кроків:

1. Вибір файлів для архівування.

2. Стиснення цих файлів за допомогою алгоритму архівування LZW.

3. Кодування отриманого архіву, за допомогою перешкодостійкого коду Хеммінга.

4. Запис отриманих даних у архів.

Якщо користувач вибирає функцію розархівування, то прогрмний продукт виконує наступні дії:

1. Вибір архіву.

2. Декодування отриманого архіву, за допомогою перешкодостійкого коду Хеммінга.

3. Розархівування цих файлів за допомогою алгоритму архівування LZW.

4. Запис розархівованих даних на диск.

Після цього користувач обирає працювати йому далі з програмою, або покинути її.

Розробка архіватора

Архіватор – це програма, що дозволяє стискати файли, щоб вони займали менший розмір. Це дуже корисна річ, особливо при переносі файлів на дискетах або розміщенні їх в Internet.

Кафедра_КОПЗ_2023рік

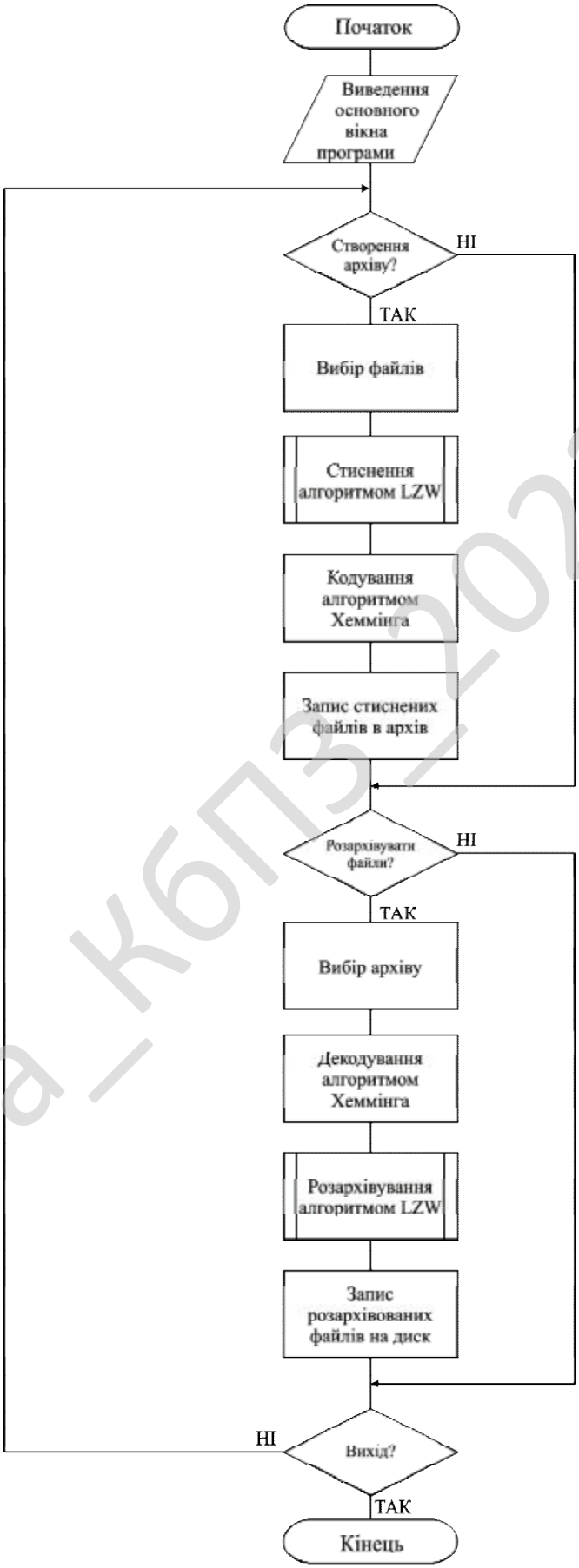

Рисунок 4.1 – Блок-схема роботи основної програми

Кафедра_КбПЗ_2023рік

Але, щоб стиснути файл необхідно його обробити по певному алгоритмі. Існує багато різних алгоритмів стиску, всі їх можна розділити на два типи: стиск із втратами й стиск без втрат. Ми розглянемо алгоритм стиску без втрат. Тут також існує безліч алгоритмів. Ми ж будемо писати програму на основі словникового алгоритму LZW (A. Lempel, J. Ziv, T.Weltch).

Ідея методу полягає в тому, що в процесі обробки вихідного файлу формується словник, слова в якому є шматочками коду вихідного файлу. При формуванні стислого файлу в нього записується не довга послідовність байт, а тільки ідентифікатор відповідного слова зі словника (його номер), або сам байт, якщо потрібного слова немає в словнику (далі замість слова “байт” я звичайно буду вживати “символ”). При цьому одержуємо вигоду за рахунок того, що даний ланцюжок (слово зі словника) може зустрічатися у файлі кілька разів, а довжина ідентифікатора слова значно коротше самого слова. При збереженні стислого файлу не потрібно зберігати словник, тому що даний алгоритм дозволяє створити словник у процесі розпакування стислого файлу.

Розглянемо словесні алгоритми

Передбачається, що ми маємо словник, у який можемо додавати слова, і здійснювати їхній пошук (у словнику). При додаванні нового слова воно одержує свій порядковий номер. Кожне слово унікально, тобто в словнику не може бути двох однакових слів.

Змінна NewByte є символом (байтом), а змінна SrcWord – словом (у нашому випадку це ANSIString).

Алгоритм стиску файлу

1. Очистимо словник; очистимо SrcWord
2. Якщо вихідний файл повністю лічений, то до п. 6
3. Завантажити в NewByte черговий символ з вихідного файлу
4. Якщо в словнику є слово SrcWord+NewByte

   тоді: привласнимо SrcWord=SrcWord+NewByte.

Інакше:

Кафедра_КПЗ_2023рік

запишемо у вихідний файл номер слова SrcWord;

додамо в словник слово SrcWord+NewByte;

привласнимо SrcWord=NewByte.

5. Перейти до п.2

6. Знайдемо в словнику слово SrcWord і збережемо його номер у вихідному файлі

7. Кінець

Блок схема описаного вище алгоритму стиску наведена на рисунку 4.2.

Алгоритм розпакування файлу

Примітка: NewCode – це не байт (у цьому випадку це тип Word)

1. Очистимо словник; очистимо SrcWord

2. Якщо вихідний файл повністю лічений, то до п. 16

3. Уважаємо в NewCode дані зі стислого файлу

4. Якщо NewCode – це просто символ

тоді:перейдемо до п. 5

інакше: перейдемо до п. 8

5. Запишемо NewCode у вихідний файл

6. Якщо в словнику є слово SrcWord+NewByte

тоді: привласнимо SrcWord=SrcWord+NewByte.

Інакше: додамо в словник слово SrcWord+NewByte;

привласнимо SrcWord=NewByte.

7. Перейдемо до п. 2

//NewCode – це номер слова в словнику.

Якщо NewCode більше, ніж слів у словнику //тобто слова з таким номером у словнику ще взагалі немає

тоді: привласнимо N=2;

привласнимо NewByte=SrcWord[1];

Якщо в словнику є слово SrcWord+NewByte

тоді: привласнимо SrcWord=SrcWord+NewByte.

Кафедра_КбПЗ_2023рік

Інакше: додамо в словник слово SrcWord+NewByte;

привласнимо SrcWord=NewByte.

Інакше: привласнимо N=1

9. Привласнимо SaveWord=слово_з_словника_з_номером(NewCode)

Збережемо SaveWord у вихідний файл

11. Запустимо цикл для і від N до довжини слова SaveWord

12. Привласнимо NewByte=SaveWord[i]

13. Якщо в словнику є слово SrcWord+NewByte

тоді: привласнимо SrcWord=SrcWord+NewByte.

Інакше: додамо в словник слово SrcWord+NewByte;

привласнимо SrcWord=NewByte.

14. Кінець циклу

15. Перейдемо до п. 2

16. Кінець

Блок-схема підпрограми розархівування наведена на рисунку 4.3.

Організація запису й читання файлу

Перш ніж приступитися до реалізації алгоритмів архівації й розпакування даних варто подумати про спосіб збереження даних на диску (і читання з диска). Стандартними засобами будь-якої мови програмування можно здійснювати читання-запис файлу тільки по байтах (я маю на увазі мінімально можливий тип даних). Для написання архіватора нам необхідно мати можливість читати й записувати дані по бітах. Для даного архіватора мною був написаний модуль **DFileStream**. Він дозволяє як зчитувати, так і записувати дані у файл ланцюжками біт, довжиною від 1 до 16 біт за раз. Т.к. завдання даного матеріалу – це створення тільки архіватора, те реалізація модуля **DFileStream** виходять за рамки викладу. Нижче будуть описані тільки необхідні функції з нього і їхнє призначення.

Кафедра_КОПЗ_2023рік

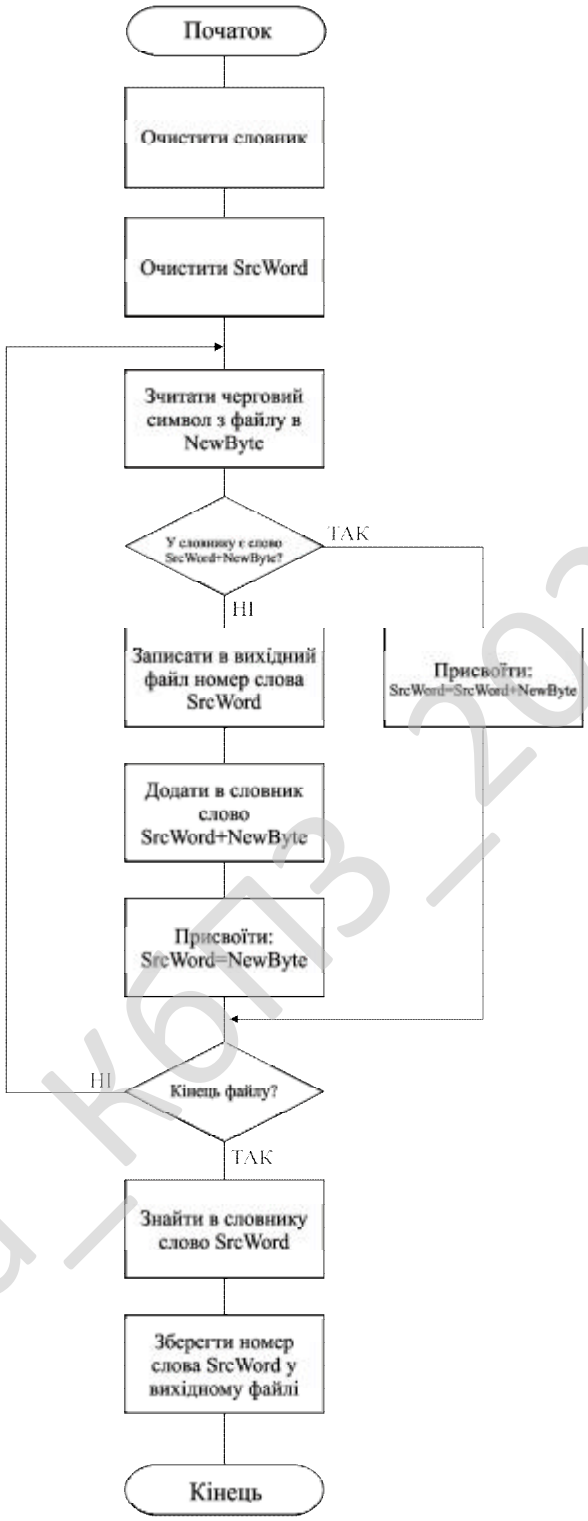

Рисунок 4.2 – Блок-схема роботи алгоритму стиску файлу

Кафедра_КОПЗ_2023рік

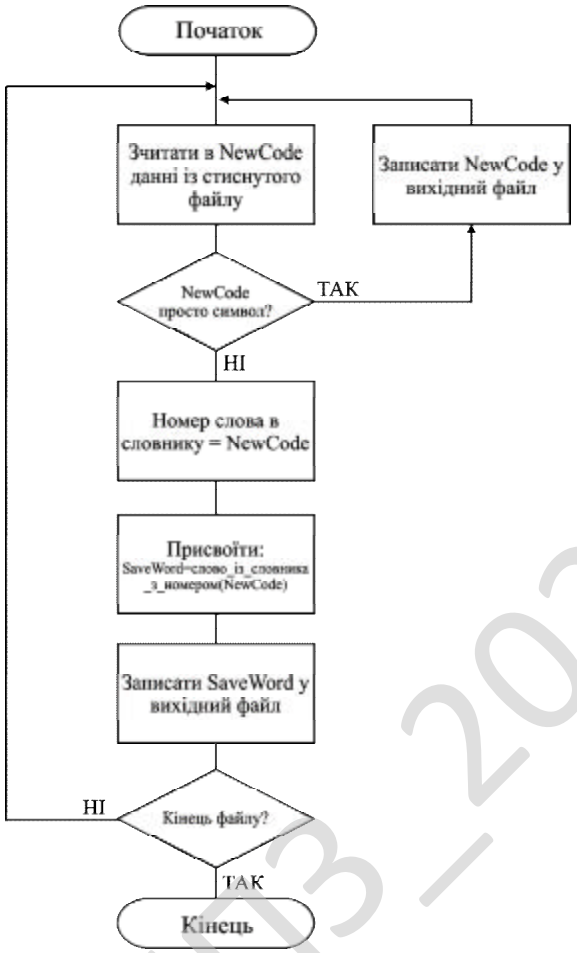

Рисунок 4.3 – Блок-схема роботи алгоритму розархівування файлу

Опис функції побітного запису.

```
TFileWriteStream                    //клас для організації побітного запису
          OpenStream(FileName:String)//відкрити файл FileName для запису
          CloseStream                        //закрити потік
          WriteBit(BitValue:Byte)            //записати у файл 1 (один) біт
          Write(BitMap:Word; BitsCount:Byte)//записати у файл декілька біт
                              (BitsCount), BitMap – бітова маска
          BytesWrote                         //кількість записаних байт
TFileReadStream                     //клас для організації побітного читання
          OpenStream(FileName:String)//відкрити файл FileName для читання
          CloseStream                        //закрити потік
          ReadBit                       //завантажити з файлу 1 (один) біт
          Read(BitsCount:Byte)          //завантажити з файлу декілька біт
                              (BitsCount)
          FileRead                      // чи перевіряє прочитаний файл повністю
```

Перераховані вище процедури й функції придадуться нам при реалізації алгоритмів стиску й розпакування файлів.

Кафедра_КОПЗ_2023рік

Словник

Вище вже були перераховані вимоги, пропоновані до словника. Для організації словника я написав модуль **Dictionary**, у якому описаний клас **TDictionary**. Нижче наведені необхідні функції даного класу.

```
TDictionary                    //клас, що описує словник
        Init                        //конструктор
        Done                        //деструктор
        Clear                  //очистити словник, і
                                проініціалізувати його
        Add(SingleWord:TSingleWord)//додати слово в словник
        Find(SingleWord:TSingleWord)    //знайти слово в словнику, якщо
                слова ні, те вернеться значення "-1"
        Words                       //Масив всіх слів у словнику
        Count                       //кількість слів у словнику
```

Як тип **TSingleWord** використовується стандартний тип **ANSIString**

```
Type    TSingleWord=ANSIString;
```

Після створення або очищення словника, перших 256 слів його заповнюються символами з таблиці ASCII.

Програмна реалізація алгоритмів

Самим серцем архіватора є функції стиску й розпакування файлів, тому постараємося написати їх (дотримуючись словесних алгоритмів). Весь інший програмний код можна буде наростити самостійно, якщо звичайно мати деякі пізнання в програмуванні.

Стиск файлу

Перед викликом функції стиску необхідно створити два бітових файлових потоки:

– для читання – вихідний файл;

– для запису – створюваний (вихідний) файл.

Також перед безпосереднім стиском необхідно у вихідному файлі зберегти ім'я й розмір вихідного файлу в прямому виді (не стислому). Це необхідно для наступного розпакування файлу.

Тепер необхідно встановити правила для формату даних у стислому файлі. Виділимо для номера словника 12 біт – одержимо 4096 можливих номерів (від 0

Кафедра_КбПЗ_2023рік

до 4095). Це значить, що словник може містити не більше 4096 слів. Для зберігання ж просто одного символу потрібно 8 біт.

Установимо наступні правила:

– якщо ми записуємо у вихідний файл просто один символ, то спочатку запишемо один біт, дорівнює нулю (“0”). Потім записуємо сам символ (тобто 8 його біт);

– якщо ми записуємо у вихідний файл номер слова зі словника, то спочатку запишемо один байт, дорівнює одиниці (“1”). Потім записуємо номер слова зі словника (тобто 12 його біт).

Все це нам необхідно, щоб при розпакуванні розрізнити, де в нас просто байт, а де номер слова.

А тепер напишемо код, дотримуючись словесного алгоритму.

Примітка: у процедурі використовуються глобальні змінні Dic:TDictionary; ReadStream:TFileReadStream; WriteStream:TfileWriteStream. Також викликається процедура запису номера слова у вихідний файл WriteByDic (вона буде розглянута нижче).

```
Procedure ProcessCompression;
Var
  SrcWord:TSingleWord;
  NewByte:Byte;
Begin
  Dic.Clear;       //очистимо словник
  SrcWord:='';
  While Not(ReadStream.FileRead) Do        //поки вихідний Файл не зчитаний
  Begin
    NewByte:=ReadStream.Read(8); //завантажуємо 1 байт
    If Dic.Find(SrcWord+Chr(NewByte)) Then //якщо слово є в словнику
      SrcWord:=SrcWord+Chr(NewByte)   //то змінимо слово SrcWord
    Else Begin     //інакше
      WriteByDic(Dic.Find(SrcWord));  //запишемо у вихідний файл
      Dic.Add(SrcWord+Chr(NewByte));  //додамо в словник
      SrcWord:=Chr(NewByte);
    End;
  End;
  WriteByDic(Dic.Find(SrcWord));
```

Кафедра_КбПЗ_2023рік

```
End;
```

Процедура WriteByDic зберігає у вихідному файлі або символ, або номер слова, залежно від переданого в неї номера.

```
Procedure WriteByDic(N:Integer);
Var
  Data:Word;
Begin
  If N<0 Then Exit;
  If N<256 Then    //якщо це просто символ
  Begin
    WriteStream.WriteBit(0);
    WriteStream.Write(N, 8);
  End
  Else Begin  //якщо це слово
    WriteStream.WriteBit(1);
    WriteStream.Write(N, 12);
  End;
End;
```

Розпакування файлу

Ініціалізуючі дії перед розпакуванням файлу аналогічні діям при запакуванні. Також необхідно вважати розмір вихідного файлу (помнете ми його зберігали) у змінну FSize.

Але от сам алгоритм буде більш складний (більше громіздкий).

```
Procedure ProcessDeCompression;
Var
  SrcWord, SaveWord:TSingleWord;
  i, N, NewCode:Integer;
  NewByte:Byte;
Begin
  Dic.Clear;
  SrcWord:='';
  While WriteStream.BytesWrote<Fsize Do    //поки не збережений весь файл
  Begin
    If ReadStream.ReadBit=0 Then
    Begin                              //якщо це просто символ
      NewByte:=ReadStream.Read(8);
      WriteStream.Write(NewByte, 8);  //перепишемо його на вихід
      If Dic.Find(SrcWord+Chr(NewByte)) Then   //якщо слово є в словнику
        SrcWord:=SrcWord+Chr(NewByte) //те змінимо слово SrcWord
```

КафедраКПЗ_2023рік

```
      Else Begin                    //інакше
        Dic.Add(SrcWord+Chr(NewByte));     //додамо в словник
        SrcWord:=Chr(NewByte);
      End;
    End
    Else Begin                      //якщо це номер слова
      NewCode:=ReadStream.Read(12);
      If NewCode>=Dic.Count Then    //якщо слова ще немає в словнику
      Begin
        N:=2;
        NewByte:=Ord(SrcWord[1]);
        If Dic.Find(SrcWord+Chr(NewByte)) Then //якщо слово є в словнику
          SrcWord:=SrcWord+Chr(NewByte)    //те змінимо слово SrcWord
        Else Begin                  //інакше
          Dic.Add(SrcWord+Chr(NewByte));   //додамо в словник
          SrcWord:=Chr(NewByte);
        End;
      End
      Else
        N:=1;
      End;
      SaveWord:=Dic.Words[NewCode];
      For i:=1 To Length(SaveWord) Do //збережемо слово
        WriteStream.Write(SaveWord[i], 8);
      For i:=N To Length(SaveWord) Do
      Begin
        NewByte:=Ord(SaveByte[i]);
        If Dic.Find(SrcWord+Chr(NewByte)) Then //якщо слово є в словнику
          SrcWord:=SrcWord+Chr(NewByte)    //те змінимо слово SrcWord
        Else Begin                  //інакше
          Dic.Add(SrcWord+Chr(NewByte));   //додамо в словник
          SrcWord:=Chr(NewByte);
        End;
      End;
    End;
  End;
End;
```

Кафедра_КБПЗ_2023рік

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

Вибір блокового шифру

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

Кафедра_КБПЗ_2023рік

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16(тому що AES працює з 16-байтними блоками).

KDF – функція створення ключа

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

PDF: функція створення псевдовипадкового числа

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

Генерація UMAC-тегів

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Кафедра_КоПЗ_2023рік

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

UMAC-32 UMAC-64 UMAC-96 UMAC-128

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 (K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 (K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 (K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

Універсальна функція гешування(UHASH)

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і, нарешті, до результату застосовується L3-HASH . Якщо при цьому довжина вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

Універсальна функція

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас

Кафедра_КБПЗ_2023рік

називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

L1-hash – перший етап

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

L2-hash – другий етап

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

L3-hash – третій етап

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

Кафедра_КбПЗ_2023рік

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображене головне вікно програми. З нього ми бачимо, що у програмі є 3 випадаючі меню:

– Файл.

– Параметри.

– Довідка.

Меню Довідка наведено на рисунку 5.5.

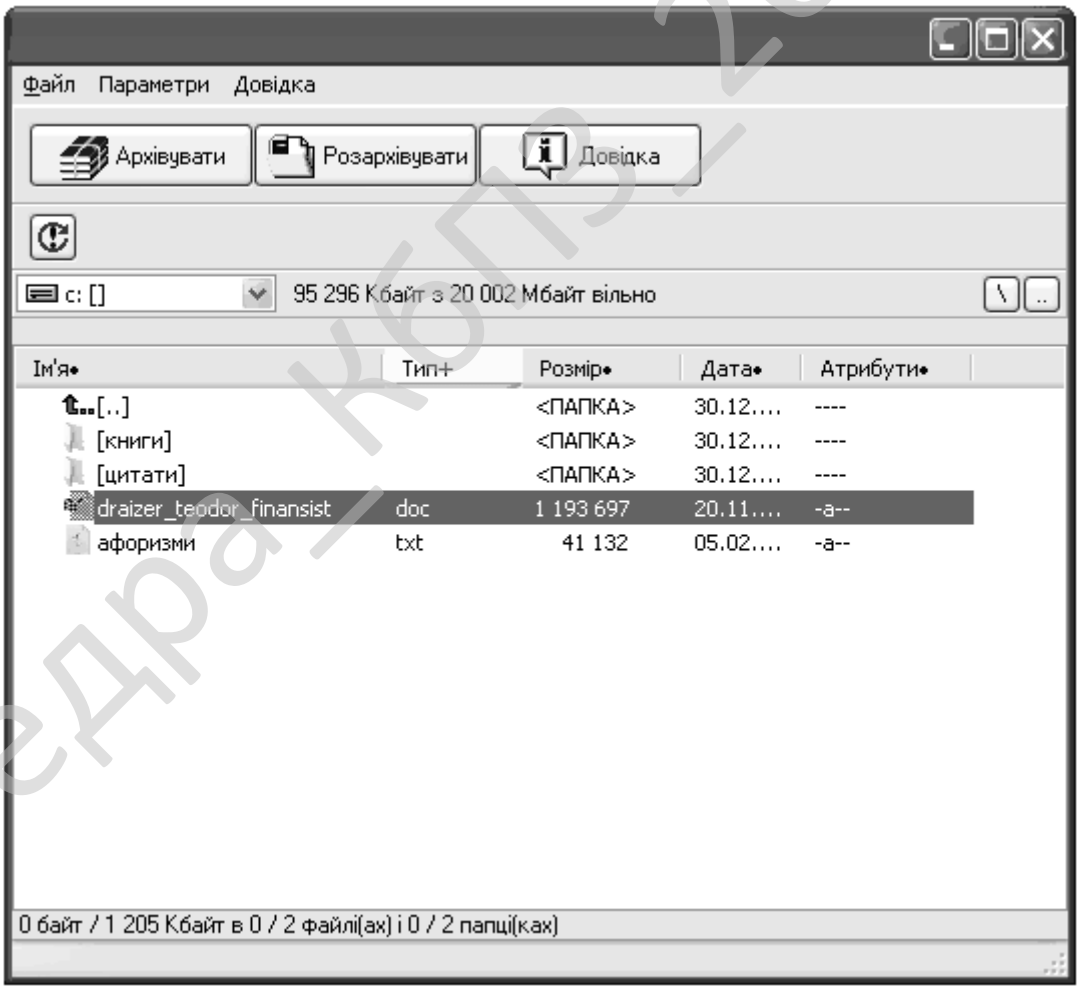

Рисунок 5.1 – Основне вікно програми

Кафедра_КБПЗ_2023рік

Також на головній панелі існують три кнопки, нажимаючі на які можливо проведення наступних дій:

– Архівувати файли.

– Розархівувати файл.

– Отримати довідку.

При виборі процедури архівування, на екран виводиться процес скріншот, якого зображено на рисунку 5.2.

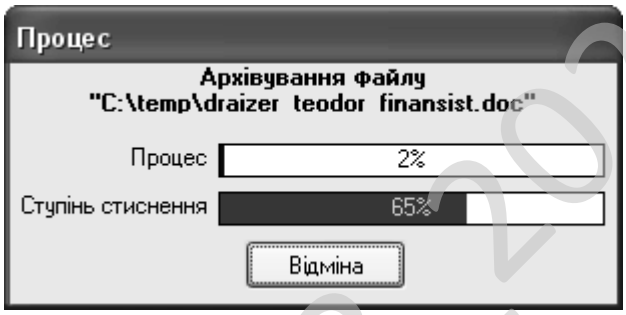

Рисунок 5.2 – Процес архівації

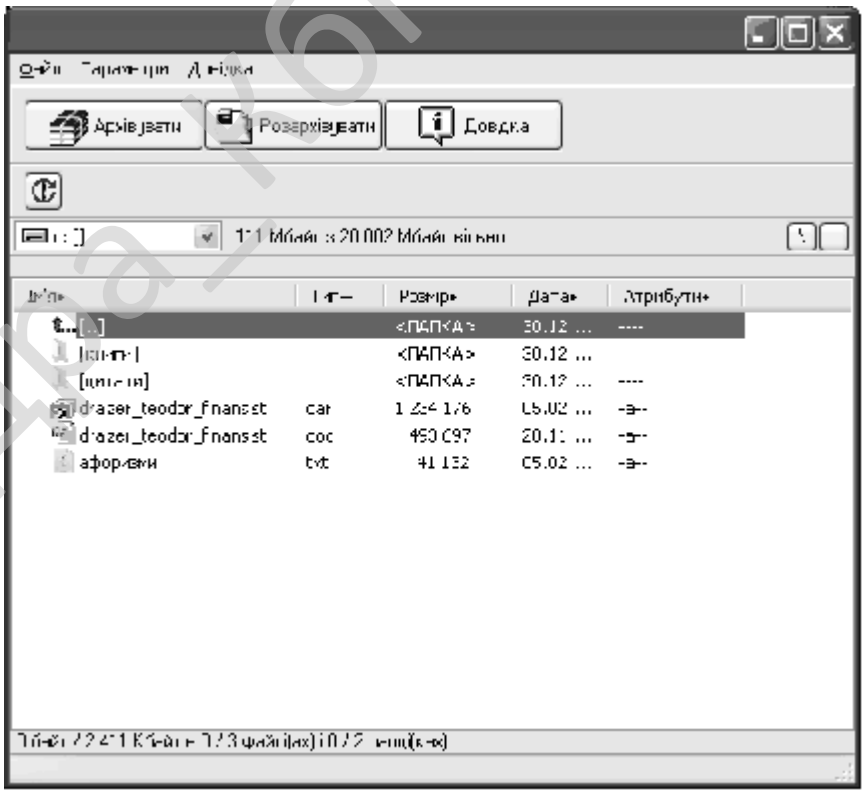

Рисунок 5.3 – Результат архівації

Кафедра_КБПЗ_2023рік

Результат процесу архівації зображено на рисунку 5.3.

При натисканні на кнопку розархівування, запускається процес зображений на рисунку 5.4.

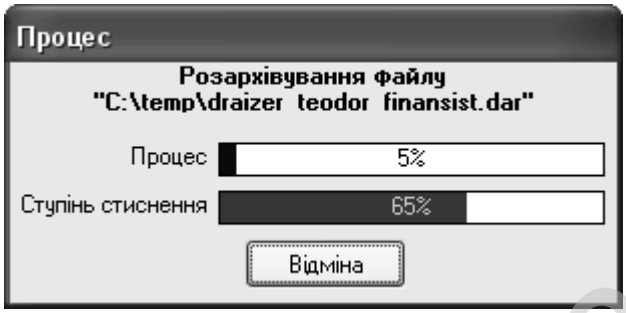

Рисунок 5.4 – Процес розархівації

Рисунок 5.5 – Довідка

Кафедра_КБПЗ – 2023рік

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для стисненого збереження інформації з підвищеною надійністю.

– Досліджена система для стисненого збереження інформації з підвищеною надійністю.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для стисненого збереження інформації з підвищеною надійністю.

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 RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для стисненого збереження інформації з підвищеною

Кафедра_КБПЗ_2023рік

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм UMAC.

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.

Кафедра_КБПЗ_2023рік

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

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

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

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

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

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

КафеДра_КОПЗ_2023рік

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

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

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

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

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

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

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

КафедраКБПЗ_2023рік

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

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

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

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

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

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

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

Кафедра_КБПЗ_2023рік

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

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

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

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

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

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

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

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

Кафедра_КОПЗ_2023рік

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

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

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

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

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

49. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

50. 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.

Кафедра_КОПЗ_2023рік

Додаток А
(обов'язковий)

Технічне завдання

Зміст

Кафедра_КБПЗ_2023рік

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

Кафедра_КБПЗ_2023рік

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю;

– 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.

Кафедра_КБПЗ_2023рік

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 RAD Studio Delphi.

Кафедра_КбПЗ_2023рік

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 81 аркуш.

8 Етапи розробки

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

Кафедра_КБПЗ_2023рік

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 13.06.2023 р.

Кафедра_КБПЗ_2023рік

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи кібербезпеки для стисненого збереження інформації з підвищеною надійністю

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 44

Літера: РП

Кропивницький – 2023 року

Кафедра_КбПЗ_2023рік

Файл Main.pas - Основна програма

```
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}
```

Кафедра КбПЗ_2023рік

```
Procedure TfmDAR.Compress;
Var
  NewFileName, OldFileName:String;
Begin
  NewFileName:=ChangeFileExt(lbItem.Caption, '.dar');
//  SaveDialog1.InitialDir:=lbPath.Caption;
  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 себе', '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);
```

Кафедра_КБПЗ_2023рік

```
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.
```

Кафедра_КбПЗ_2023рік

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;
```

Кафедра_КбПЗ_2023рік

```
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.
```

Кафедра_КбПЗ_2023рік

Файл Compressor.pas - Стиснення файлу

```
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;
```

Кафедра_КбПЗ_2023рік

```
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 довга}
  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  //те нічого не робимо
    Out:=-1024;
    InpWord:=New;  //Змінимо вхідне слово
  End
  Else Begin //Якщо немає в словнику, те...
    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
```

Кафедра_КоПЗ_2023рік

```
  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;
```

Кафедра_КбПЗ_2023рік

```
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, те 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);
```

Кафедра_КбПЗ_2023рік

```
  ProcessDeCompression(FSize);

  ReadStream.CloseStream;
  WriteStream.CloseStream;
End;


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

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

end.
```

Кафедра_КбПЗ_2023рік

Файл DeCompressor.pas - Основна програма

```
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);
```

Кафедра_КбПЗ_2023рік

```
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 довга}
  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  //те нічого не робимо
    Out:=-1024;
    InpWord:=New;  //Змінимо вхідне слово
  End
  Else Begin //Якщо немає в словнику, те...
    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;
```

КафедраКбПЗ_2023рік

```
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);
```

Кафедра_КбПЗ_2023рік

```
  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
```

Кафедра_КбПЗ_2023рік

```
        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, те 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;
```

Кафедра_КбПЗ_2023рік

```
end.
```

Кафедра_КбПЗ_2023рік

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;
```

Кафедра_КбПЗ_2023рік

```
  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.
```

Кафедра_КбПЗ_2023рік

Файл frFilePanel.pas - Інтерфейс

```
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;
```

Кафедра КОПЗ – 2023рік

```
    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;
{}
```

Кафедра_КбПЗ_2023рік

```
  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);
```

Кафедра КбПЗ 2023рік

```
  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-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;
```

Кафедра_КБІЗ_2023рік

```
  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);
```

Кафедра_КбПЗ_2023рік

```
  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;
```

Кафедра_КбПЗ_2023рік

```
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;
```

Кафедра_КбПЗ_2023рік

```
  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;
```

Кафедра_КбПЗ_2023рік

```
  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);
```

Кафедра_КбПЗ_2023рік

```
    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-I-}
    GetDir(Ord(Drive)-64, Dir);
    If Drive=Dir[1] Then
      ChDir(Dir)
    Else Begin
      Dir:=Drive+':\';
```

Кафедра_КбПЗ_2023рік

```
      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-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;
```

Кафедра_КбПЗ_2023рік

```
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.
```

Кафедра_КбПЗ_2023рік

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;
```

Кафедра_КбПЗ_2023рік

```
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;
```

Кафедра_КбПЗ_2023рік

```
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;
```

Кафедра_КбПЗ_2023рік

```
  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]);
```

Кафедра_КОПЗ_2023рік

```
    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;
```

Кафедра_КбПЗ_2023рік

```
  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;
```

Кафедра_КбПЗ_2023рік

```
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-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;
```

Кафедра_КбПЗ_2023рік

```
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.
```

КафеДра_КбПЗ_2023рік

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}
```

КафедраКОТЗ_2023рік

```
 {$ Q-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
```

Кафедра_КбПЗ_2023рік

```
      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-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
```

Кафедра_КбПЗ_2023рік

```
    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.
```

Кафедра_КбПЗ_2023рік

Файл about.pas - Довідка

```
unit about;

interface

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

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

var
  FmAbout: TFmAbout;

implementation

{$R *.dfm}

procedure TFmAbout.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Програмне забезпечення системи кібербезпеки для стисненого
збереження інформації з підвищеною надійністю ');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Якименко Н.М.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Бабак Андрій Романович');
Memo1.Lines.Add('                  гр. КБ-19');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
end;

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

Кафедра_КБПЗ_2023рік