Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу, комп'ютерного моніторингу
та інженерної графіки

ІНФОРМАЦІЙНА СИСТЕМА АРХІВУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ КОНТЕКСТНОГО МОДЕЛЮВАННЯ

Пояснювальна записка до магістерської кваліфікаційної роботи

Виконав: студент 2 курсу, групи ІСТ-18м
спеціальності 126 – «Інформаційні системи та технології»
Федюк О. П.

Керівник: к.т.н., доц. Крижановський Є.М.
Рецензент: _____

Вінниця ВНТУ – 2019 року

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу, комп'ютерного моніторингу
та інженерної графіки

Освітньо-кваліфікаційний рівень___магістр_____
Спеціальність _____126 - Інформаційні системи та технології___
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри САКМІГ
_____ д.т.н., проф. В. Б. Мокін
(підпис)
"__" _____ 201__ р.

ЗАВДАННЯ
на магістерську кваліфікаційну роботу студенту
Федюку Олександру Петровичу

1. Тема роботи : «Інформаційна система архівування даних з використанням контекстного моделювання»
керівник роботи Крижановський Є.М., к.т.н., доц., каф. САКМІГ

затверджені наказом вищого навчального закладу від "__"____20_року №__
2. Строк подання студентом роботи_____
3. Вихідні дані до роботи:
- мова програмування – С++
- наявні алгоритми ущільнення інформації
- дані про сучасні програмні додатки для ущільнення файлів
- вимоги користувачів до програмних додатків для ущільнення файлів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ; обґрунтування доцільності створення програмного додатку для ущільнення файлів; розробка структури та алгоритму роботи програмного додатку для ущільнення файлів за допомогою алгоритму контекстного моделювання; розробка плану тестування програмного додатку; висновки; додатки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема алгоритму ущільнення; структура програми; інтерфейс та його структура.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Крижановський Є.М., к.т.н., доц. каф. САКМІГ		
4	Глущенко Л.Д., к.е.н., доц. каф. ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	09.2019	
2	Розробка структури та алгоритмів програмного продукту	09.2019	
3	Розробка інформаційної системи	09.2019	
4	Економічна частина	10.2019	
5	Розробка інтерфейсу користувача додатку	10.2019	
6	Тестування роботи додатку	11.2019	
7	Розробка інструкції користувача	11.2019	
8	Оформлення матеріалів до захисту МКР	09.2019	

Студент _____ **Федюк О.П**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Крижановський Є.М**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська кваліфікаційна робота: 93 ст., 9 табл.,41 рис., 17 джерел.

Об’єкт досліджень – процес ущільнення інформації з використанням контекстного моделювання.

Дана робота присвячена розробці програмного продукту для ущільнення файлів за допомогою контекстного моделювання. Метою роботи є розробка інформаційної системи архівування даних, що базується на використанні адаптивних методів ущільнення, які дозволяють виконувати кодування за один прохід початкового файлу.

В ході виконання магістерської роботи були проаналізовані функціональні та не функціональні характеристики аналогів програмного продукту. Розроблено алгоритм роботи даного додатку, а також створено відповідний інтерфейс.

Дана інформаційна система для архівування даних за допомогою контекстного моделювання розроблена на мові С++ у середовищі програмування Microsoft Visual Studio 2015 Community, яке характеризується зручністю та зрозумілістю інтерфейсу.

Галузь застосування – підприємства чи організації, які займаються обробкою інфомації.

ІНФОРМАЦІЙНА СИСТЕМА АРХІВУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ КОНТЕКСТНОГО МОДЕЛЮВАННЯ, АРХІВАТОР, АЛГОРИТМИ УЩІЛЬНЕННЯ ДАНИХ, КОНТЕКСТНЕ МОДЕЛЮВАННЯ, СТЕПІНЬ УЩІЛЬНЕННЯ.

ABSTRACT

The master's qualification work: 93 p., 9 tables, 41 figures,17 sources.

The object of research is the process of automating the scheduling of training.

This paper is devoted to the development of software for file compression using context modeling techniques. The purpose of the work is to develop a software application based on the use of adaptive compression methods that allow to perform encoding in one pass of the original file.

In this work, the functional and non-functional characteristics of analogues for the software product were thoroughly analyzed. Therefore, the algorithm of work of the application has been developed, and the interface for the application is created.

The information system for compression data using context modeling is developed in the C ++ language in the Microsoft Visual Studio 2015 Community environment, which can be defined as convenient and easy to operate.

Field of application - enterprises or organizations engaged in information processing and data collection.

INFORMATION SYSTEM OF DATA ARCHIVING WITH USE OF CONTEXT MODELING, ARCHIVER, DATA COMPRESSION ALGORITHMS, CONTEXT MODELING, COMPRESSION DEGREE.

ЗМІСТ

ВСТУП

Актуальність. У наш час користувачі ПК використовують та оперують великими обсягами інформації. Для економії місця інформацію необхідно зменшувати за-допомогою спеціальних програм – архіваторів. Архіватор – це спеціальне програмне забезпечення для ущільнення даних. Такі програми не тільки стискають інформацію в окремому файлі, але можуть і об'єднати в один архів групу файлів, що є досить зручно.

Найбільш поширені програми-архіватори мають приблизно однакові можливості і жодна з них не перевершує іншу по всіх параметрах: одні програми працюють швидше, інші забезпечують кращу міру стиснення файлів. Навіть якщо порівнювати програми тільки по мірі стиснення, то серед них немає лідера.

Принцип роботи архіватора заснований на пошуку в файлі «надлишкової» інформації і подальшому її кодуванні з метою отримання мінімального об'єму. Сьогодні використовують два види ущільнення даних: з втратою та без втрати даних.

Заслуговують на увагу методи ущільнення без втрат. Серед них існує дві принципових схеми ущільнення – кодування Хаффмана і LZW-кодування, які формують основу для багатьох систем ущільнення. Ці схеми подають два різних підходи до ущільнення даних. Окремий клас утворюють методи, які відомі як арифметичне кодування. Даний алгоритм кодування використовує як основу технології ущільнення, імовірність появи символу у файлі, однак сам процес арифметичного кодування має принципові відмінності. У результаті арифметичного кодування символьна послідовність (рядок) замінюється дійсним числом, яке є більшим аніж нуль і меншим за одиницю [1].

З появою арифметичного кодування проблема генерації коду була фактично вирішена. З тих пір основна увага стала приділятися питанням, пов'язаним з моделюванням. Нові підходи опираються на парадигму ущільнення за допомогою універсального моделювання і кодування (universal modelling and coding), запропоновану Ріссаненом і Ленгдоном (Langdon) в 1981 р.

Найбільш широке застосування отримала техніка контекстного моделювання PPM (Prediction by Partial Matching), яка означає передбачення за частковим збігом та її модифікації. Вона забезпечує ущільнення в 3-4 рази для текстів і в 2-3 рази для об’єктних файлів при прийнятних обчислювальних затратах. На основі даної техніки було створено методи ущільнення сімейства PAQ, які дозволяють найкраще ущільнювати файли, та які будуть використовуватися в даній роботі.

Актуальність розробки полягає у можливості використання розробки системи на підприємствах або різних департаментах влади, які щоденно мають справу з великими обсягами інформації, з використанням як ліцензійного так і офіційно безкоштовного програмного забезпечення.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри системного аналізу, комп’ютерного моніторингу та інженерної графіки Вінницького національного технічного університету та плану наукової та навчально-методичної роботи.

Метою дослідження є підвищення рівня ущільнення інформації за рахунок реалізації оптимального алгоритму ущільнення інформації, а також розробки інтерфейсу, який відповідає сучасним стандартам та забезпечує високу швидкодію та стабільність роботи користувача та системи.

Для досягнення вищевказаної мети необхідне розв’язання такі **задачі:**

- проаналізувати відомі методи ущільнення файлів без втрат;
- реалізувати алгоритм для інформаційної системи архівування файлів на основі контекстного моделювання;
- розробити програмне забезпечення для роботи даної системи;
- виконати експериментальні дослідження ущільнення файлів за допомогою контекстного моделювання.

Використані у науковій роботі **методи дослідження**:

- порівняння – для визначення переваг та недоліків існуючих аналогів, які будуть враховані при створенні програмного продукту;

- узагальнення – для визначення загальних властивостей і ознак існуючих систем архівування з використанням контекстного моделювання;
- формалізація – для перетворення схем та діаграм у код мови програмування;
- аналіз – для поділу об'єкту дослідження на складові частини з метою їх незалежного дослідження;
- експеримент – для тестування та визначення ефективності розробки та аналізу доцільності його створення.

Об’єктом дослідження є процес ущільнення інформації.

Предметом дослідження є методи і інструменти для архівування даних.

Наукова новизна одержаних результатів. Результати, які були отримані в процесі вирішення задач та складають наукову новизну дослідження, полягають у вдосконалені та подальшому розвитку підходу до ущільнення файлів без втрат на основі алгоритму PPM, відмінністю якого є застосування контекстного моделювання, яке дає оцінку імовірності появи символу в залежності від попередніх, або контексту.

Практична цінність даної роботи – запропоновано рішення для оптимального архівування файлів, яке на відміну від існуючих використовує ефективний алгоритм ущільнення файлів з використанням контекстного моделювання.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується правильною постановкою завдань, коректним аналізом імнуючих аналогів та алгоритмів ущільнення даних під час доведення наукових положень, та порівнянням результатів з відомими існуючими аналогами.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані особисто автором програмної системи.

Апробація результатів роботи. Атрибуцію подальшого дослідження було здійснено на конференції «Інформаційні технології і автоматизація - 2019» (ГО «Клуб молодих вчених Одеської національної академії харчових технологій», 2019).

Публікації. За результатами магістерської кваліфікаційної роботи було опубліковано тези на науково-технічну конференцію: «Інформаційні технології і автоматизація - 2019» [2].

1 ЗАГАЛЬНІ ВІДОМОСТІ

1.1 Аналіз сучасного стану ущільнення інформації

Технології стиснення даних є цілком актуальними в умовах формування світового інформаційного простору. Головна причина використання зазначених технологій стиснення даних в інформаційно-комунікаційних системах полягає в бажанні передавати, обробляти й зберігати інформацію з найбільшою ефективністю. В деякому сенсі задача стиснення даних полягає у виділенні з інформаційного потоку найбільш значущої та інформативної його частини, яка дозволить відновити з втратами або без втрат всю початкову інформацію [3].

Архіватори усім відомі. Приклади таких програм – популярні в Інтернеті WinZip, WinRAR та 7Zip. Архіватор дозволяє запаковувати файли. Проте, в чому полягає ущільнення?

Спростивши справу можна представити це наступним чином. Використовуючи спеціальні алгоритмічні методи, архіватор знаходить у файлах послідовності, що часто повторюються, і замінює їх більш короткими кодами. Наприклад, в текстових файлах часто повторюються деякі літери, «е», «а» або знак пробілу. Архіватор обчислює число входжень символів у тексті, а потім будує оптимізовану таблицю символів, в якій найбільш символи, які найбільше мають найкоротші коди, як в азбуці Морзе. Весь текст перекодовується відповідно до нової таблиці, і процес повторюється спочатку. Адже часто зустрічаються не лише окремі літери, але і їх послідовності, наприклад сполучення «пр», «ст» або «,». Перекодування файлу завершується, коли його розмір після оптимізації перестає зменшуватись.

Зазвичай, архіватори дозволяють ущільнювати не лише текстові файли, але й графічні та звукові. Оскільки, текст представлений у вигляді набору кодів, архіватор з тим же успіхом ущільнює і нетекстові файли. Наприклад, у файлах програм, типу EXE або DLL, деякі коди команд зустрічаються частіше ніж у інших. Окрім цього, багато компонувальників здійснюють вирівнювання, при якому окремі сегменти

програми доповнюються нулями до тих пір, поки розмір сегмента не стане кратним, наприклад, 8 або 16 байтам. Відповідно, програмні файли також містять багато «води», яку можна «вижати» архіватором і ущільнити файл приблизно на 40-50%.

Ущільнення досить вигідне для поширення інформації, оскільки дозволяє без втрат розмістити її на носії меншого розміру і значно скоротити час завантаження по мережі. Але архіви, створені звичайними програмами ущільнення, мають один вагомий недолік: файли в них стають недоступними безпосередньо [4].

1.2 **Порівняльний аналіз аналогів**

Серед програм-архіваторів найбільш популярними є наступні:

7-Zip (рисунок 1.1) – один із найкращих безкоштовних архіваторів з відкритим вихідним кодом. Його новий формат архівів 7Z – дозволяє ущільнювати файли більш ефективно, ніж це робить RAR-формат. Також, що ще варто відмітити – це висока швидкість роботи при стисненні або розпакуванню файлів. Ну і на завершення: 7-Zip дуже легко та зручно вписується в провідник і є повністю безкоштовним [5].

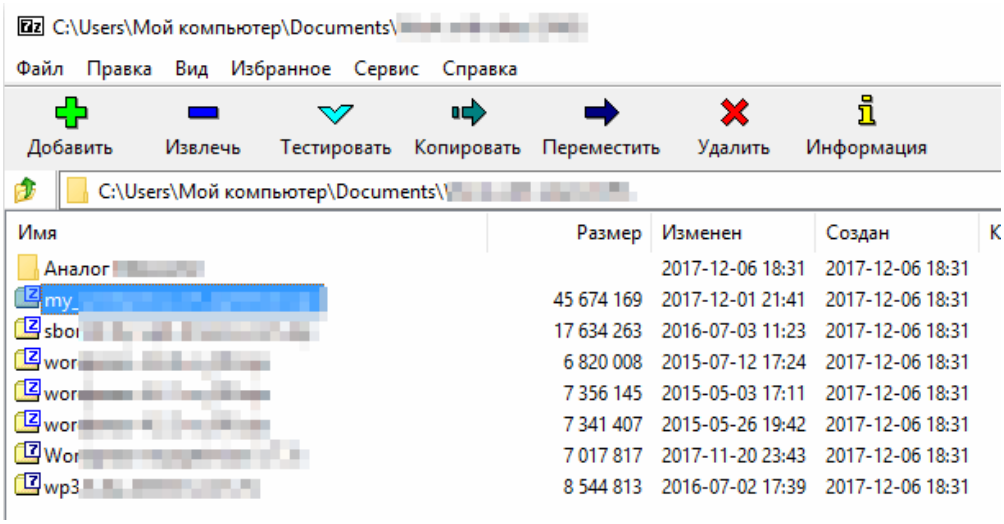


Рисунок 1.1 – Вікно програми 7-Zip

WinRAR (рисунок 1.2) – це потужна програма, яка включає в себе цілий спектр вбудованих додаткових функцій, які допоможуть оптимізувати роботу з архівами.

WinRAR підтримує усі популярні формати архівів (RAR, ZIP, CAB, ARJ, LZH, ACE, TAR, GZip, UUE, ISO, BZIP2, Z і 7-Zip).

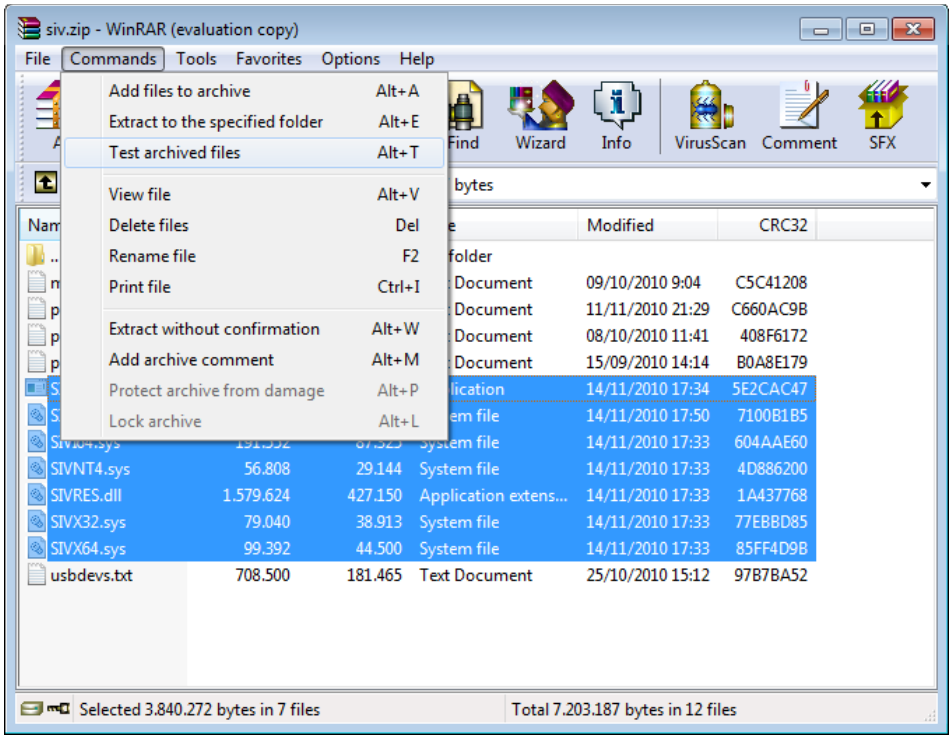


Рисунок 1.2 – Вікно програми WinRAR

WinRAR ідеально підходить для стиснення мультимедійних файлів. Програма автоматично розпізнає формат файлу та обирає оптимальний метод упакування. Особливий алгоритм архівації дозволяє чудово ущільнювати мультимедійні та виконавчі файли, а також бібліотеки об’єктних модулів.

WinRAR також ідеальний для передачі даних по Інтернету. 256-бітний криптографічний захист і технологія електронної підписки архівів подарують впевненість в захищеності файлів.

WinRAR – це shareware-продукт, що означає, що можна користуватися програмою протягом 40 днів абсолютно безкоштовно [6].

WinZip (рисунок 1.3) став чи не першою програмою компресії файлів зі зручним графічним інтерфейсом, завдяки чому завоював колосальну популярність (лише із сайта CNET скачано близько 100 млн. копій).

WinZip має вбудовані засоби для роботи з архівами інших форматів (tar, gzip, UUEncode, XXencode, BinHex, Mime). WinZip може працювати з архівами, створеними DOS-архіваторами ARJ, LZH або ARC. Починаючи з сьомої версії, WinZip дозволяє працювати також з CAB-архівами. Саме в такі архіви упаковані дистрибутиви Windows, Microsoft Office і багатьох інших програм [7].

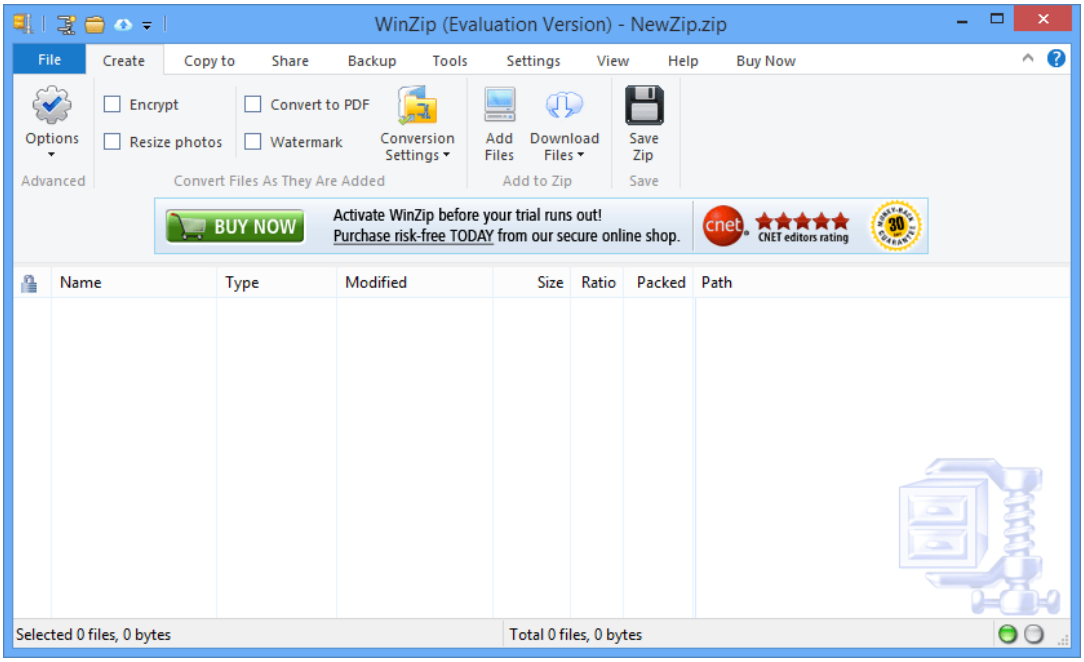


Рисунок 1.3 – Вікно програми WinZip

PeaZip (рисунок 1.4) – це безкоштовна утиліта архіватора файлів, заснована на технологіях Open Source 7-Zip, p7zip, FreeArc, PAQ та PEA-проектах.

Крос-платформенна, повнофункціональна та зручна альтернатива WinRAR, WinZip та аналогічним програмам, відкриває та витягує понад 180 форматів архівів: 001, 7Z, ACE (*), ARC, ARJ, BZ2, CAB, DMG , GZ, ISO, LHA, PAQ, PEA, RAR, TAR, UDF, WIM, XZ, ZIP ZIPX.

Випущений під відкритою ліцензією LGPLv3, безкоштовний для будь-якого використання (приватний та професійний), всі пакети PeaZip є безпечними для завантаженнями і не містять реклами та шкідливого програмного забезпечення.

Крім того, портативні пакети, для Linux і Windows, не потребують інсталяції: просто видобудьте та використовуйте програму [8].

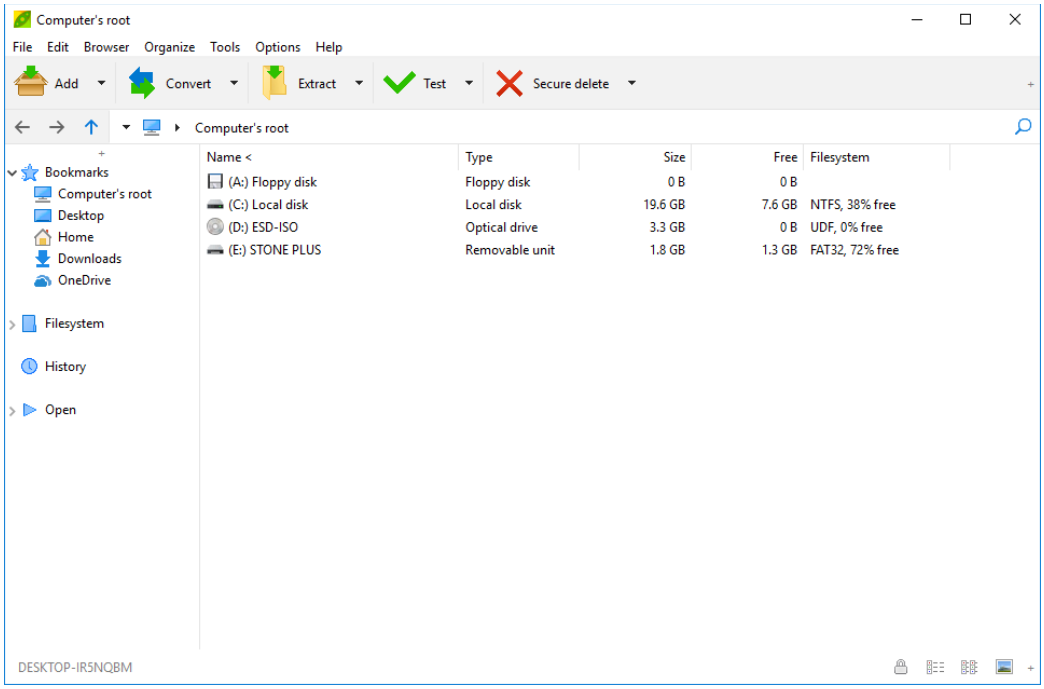


Рисунок 1.4 – Вікно програми PeaZip

Проаналізувавши усі аналоги, визначено їхні можливості та недоліки, які враховувались при створенні власного програмного забезпечення з назвою «Sezar» (табл. 1.1).

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	7-Zip	WinRAR	WinZip	PeaZip	Sezar
Не вимагає встановлення	-	-	-	+	+
Зрозумілий інтерфейс	+	+	-	-	+

Продовження таблиці 1.1

Критерій	7-Zip	WinRAR	WinZip	PeaZip	Sezar
Зручна навігація між папками	-	-	+	+	+
Шифрування даних	+	+	+	-	+
Прискорений метод Drag&Drop (перетягування в активне вікно)	-	-	-	-	+
Безкоштовно	+	-	-	+	+
Наявність перекладу інтерфейсу на українську мову	+	+	+	-	+
Загальна сума	4	3	3	3	7

Для оцінки якості роботи архіваторів необхідно дізнатися максимальні значення продуктивності роботи архіватора, степіні стиснення та час, який при цьому витрачається [9].

Продуктивність – це відношення стислих даних до швидкості стиснення і виражається формулою:

$$П = (ДС - ПС) / Шк, \quad (1.1)$$

де П - продуктивність,

ДС - до стиснення;

ПС - після стиснення;

Шк - швидкість стиснення.

Степінь стиснення – це відношення вихідних даних до стиснених даних і виражається формулою:

$$Ст = ДС / ПС, \quad (1.2)$$

де Ст - степінь стиснення.

В таблиці 1.2 подані типи файлів та їх початкові розміри.

Таблиця 1.2 – Файли до ущільнення

Тип	Опис	Розмір в Мб
Текст	Текстові документи в форматах xls(x), doc(x)	1000,436073
Зображення	Картинки в форматі jpeg	1000,863571
Медіа	Аудіо файли в форматі mp3	1006,598025

Розглянемо, що буде, якщо попередні файли максимально допустимо стиснути. При цьому призначимо кожному критерію вагу для кращого визначення оптимальних характеристик ущільнення розроблюваної інформаційної системи архівування з використанням контекстного моделювання. Наприклад, вага часу буде рівна 0,1, вага продуктивності – 0,3, а вага степеню стиснення – 0,6. Результати такого стиснення наведені у таблиці 1.3.

Таблиця 1.3 – Функціональні характеристики архіваторів

Текст				
Критерій	7-Zip	WinRAR	WinZip	PeaZip
Час (с)	97,66666667	42,33333333	112,00000000	149,66666667
Продуктивність	7,93995029	15,97546547	6,17543388	6,31442614
Степінь стиснення	4,447023017	3,086418988	3,239885491	18,065926208
Загальний коефіцієнт	14,81	10,87	14,99	27,72
Зображення				
Критерій	7-Zip	WinRAR	WinZip	PeaZip
Час (с)	3,33333333	28,33333333	137,33333333	102,333333

Продовження таблиці 1.3

Продуктивність	287,84590050	0,35517833	1,70074008	9,39526456
Степінь стиснення	24,430829643	1,010161211	1,304574419	25,660541412
Загальний коефіцієнт	101,34	3,537	14,9	33,33
Медіа				
Критерій	7-Zip	WinRAR	WinZip	PeaZip
Час (с)	98,00000000	73,66666667	43,33333333	190,00000000
Продуктивність	0,25402940	0,34935131	0,56270332	0,17333575
Степінь стиснення	1,025519046	1,025790770	1,025376167	1,033854792
Загальний коефіцієнт	10,475	8,075	5,1	19,651

Проаналізувавши дані із таблиці 1.3 можна зробити висновок, що найкращий архіватор для ущільнення текстових документів – це PeaZip, у якого степінь стиснення становить 18 разів, але йому на це знадобиться найбільше часу (2 хв і 20 секунд). А ось для ущільнення зображень краще використовувати 7-Zip, який за 3,3 секунди зменшив файл у 24,4 рази на відміну від Peazip, якому, щоб зменшити файл приблизно до такого розміру знадобилося 102 с. Що ж стосується медіа файлів, то усі архіватори показали майже однакові результати. Також можна помітити, що програма PeaZip найкраще стискує дані, але при цьому потребує найбільше часу з-поміж усіх. Почесне друге перше місце за якістю ущільнення файлів займає 7-Zip, який потребує набагато менше часу та має непогані значення продуктивності. WinRAR неприємно розчарував, бовтаючись майже у кінці.

Проаналізувавши дані максимально допустимого ущільнення файлів наявними аналогами було визначено основні характеристики для нашого архіватора. Оптимальна степінь стиснення тексту має становити в межах від 4 до 18, зображення – від 24 і вище, медіа – від 1,025.

Таблиця порівняльних характеристик аналогів показала, що розробка програмного продукту є доцільною. В результаті отримаємо продукт, що покриває недоліки існуючих рішень і забезпечує оптимальну степінь стиснення різного типу файлів.

1.3 Аналіз методів ущільнення без втрат

Серед алгоритмів ущільнення без втрат існують дві схеми ущільнення-кодування Хаффмана (Huffman) і LZW-кодування (за початковими літерами прізвищ Лемпел (Lempel) і Зів (Ziv) (його авторів) і Уелч (Welch), який його суттєво модифікував), які формують основу для багатьох систем ущільнення. Ці схеми подають два різних підходи до ущільнення даних. Окремий клас утворюють методи, які відомі як арифметичне кодування. Таким чином алгоритми ущільнення даних без втрат можна розділити на такі групи (рис. 1.5):

– статистичні методи ущільнення;

– словникові (евристичні) методи ущільнення;

– арифметичне кодування.

Рисунок 1.5 – Основні методи ущільнення даних

Статистичні алгоритми (Шеннона-Фано, Хаффмана, кодування за ступенем новизни, імовірнісне ущільнення та ін.) потребують знання ймовірностей появи

символів в повідомленні, оцінкою якої є частота появи символів у вхідних даних. Як правило, ці імовірності невідомі. З урахуванням цього статистичні алгоритми можна поділити на три класи.

1. Неадаптивні - використовують фіксовані, завчасно задані імовірності символів. Таблиця ймовірностей символів не передається разом з файлом, оскільки вона відома завчасно. Недолік: невеликий перелік файлів, для яких досягається прийнятний коефіцієнт ущільнення.

2. Напівадаптивні - для кожного файла будується таблиця частот символів і за її допомогою ущільнюють файл. Разом з ущільненим файлом передається таблиця символів. Такі алгоритми досить непогано ущільнюють більшість файлів, але необхідна додаткова передача таблиці частот символів, а також два проходи початкового файлу для виконання кодування.

3. Адаптивні - починають працювати з фіксованою початковою таблицею частот символів (зазвичай всі символи спочатку рівноймовірні) і в процесі роботи ця таблиця змінюється в залежності від символів, що зустрічаються у файлі. Переваги: однопрохідність алгоритму, не потребує передачі таблиці частот символів, достатньо ефективно стискає широкий клас файлів [1].

Евристичні (словникові) алгоритми ущільнення (типу LZ77, LZ78), як правило, шукають в файлі рядки, що повторюються, і будують словник фраз, що вже зустрічались. Зазвичай такі алгоритми мають цілий ряд специфічних параметрів (розмір буфера, максимальна довжина фрази і т.д.), підбір яких залежить від досвіду автора роботи, і ці параметри добираються таким чином, щоб досягти оптимального співвідношення часу роботи алгоритму, коефіцієнта ущільнення та переліку файлів, що добре стискаються.

Арифметичне кодування, подібно до статистичних методів, використовує імовірність появи символу у файлі. У результаті такого кодування символьна послідовність замінюється дійсним числом більше нуля і менше одиниці.

Як зазначалося у вступі, метод арифметичного кодування призвів до абсолютно нових підходів до ущільнення даних без втрат. Відповідно до однієї з таких концепцій процес ущільнення складається з двох самостійних частин:

– моделювання;

– кодування.

Під моделюванням розуміється побудова моделі інформаційного джерела, що породжує дані, які ущільнюються, а під кодуванням – відображення оброблюваних даних в стислу форму подання на підставі результатів моделювання (рис. 1.6).

«Кодувальник» створює вихідний потік, що є компактною формою подання оброблюваної послідовності, на підставі інформації, що поставляється йому «моделювальником».

Оцінювання ймовірності символів при моделюванні проводиться на підставі відомої статистики і можливо, апріорних припущень, тому часто говорять про завдання статистичного моделювання.

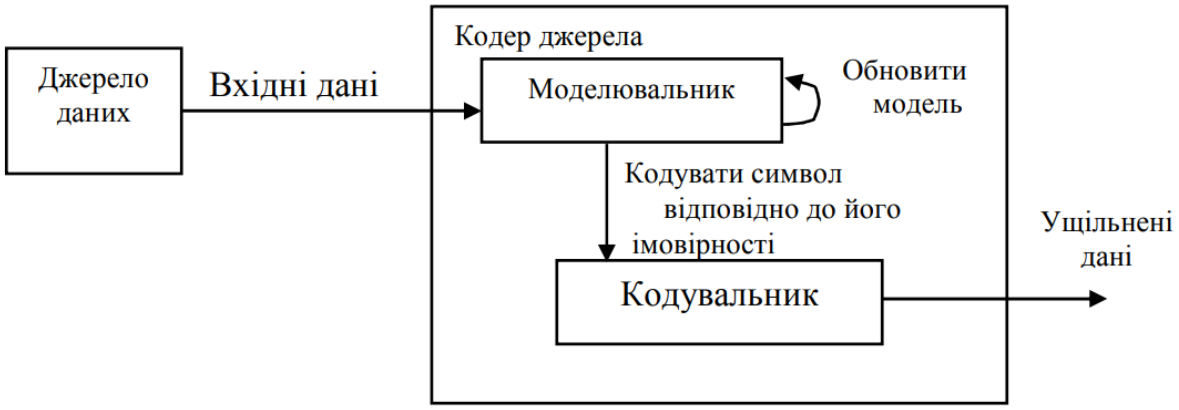


Рисунок 1.6 – Нова схема ущільнення даних

Можна сказати, що моделювальник передбачає імовірність появи кожного символу в кожній позиції вхідного рядка, звідси ще одне найменування цього компонента – «передбачувач» або «предиктор» (від predictor). Чим точніша оцінка імовірності появи символів, тим більше коди відповідають оптимальним, тим краще ущільнення. На етапі статистичного кодування виконується заміщення символу s_i, з оцінкою вірогідності появи $p(s_i)$ кодом завдовжки – $\log_2 p(s_i)$ біт.

Виділяють 4 варіанти моделювання:

1. Статичне.

2. Напівадаптивне.

3. Адаптивне (динамічне).

4. Блоково-адаптивне.

При статичному моделюванні для будь-яких даних, що обробляються, використовується одна і та ж модель. Опис моделі зберігається в структурах кодера і декодера. Недоліком такого підходу є погане ущільнення або навіть збільшення даних, що не відповідають заданій моделі.

Найбільш простий спосіб оцінювання імовірностей реалізується за допомогою напівадаптивного моделювання і полягає в попередньому підрахунку частот поява символів в блоці даних, що ущільнюються.

При адаптивному моделюванні у міру кодування модель змінюється за заданим алгоритмом після ущільнення кожного символу. Однозначність декодування досягається тим, що спочатку кодер і декодер мають ідентичну і звичайно дуже просту модель, модифікація якої при кодуванні і декодуванні виконується однаковим чином. Коефіцієнти ущільнення при адаптивному моделюванні можуть навіть досягати значень близьких до напівадаптивного.

Блоково-адаптивне моделювання є окремим випадком адаптивної стратегії. В цьому випадку модель обновлюється не після обробки кожного символу, а після обробки блока символів. Довжина блока може змінюватись в процесі кодування.

Аналіз розповсюджених типів даних виявляє сильну залежність ймовірності появи символів від попередніх символів у повідомленні, тобто більшість джерел повідомлень є джерелами з пам’яттю. Наприклад, для більшості європейських мов врахування безумовної імовірності символів дозволяє зменшити середню довжину коду до 4,5 біта на символ, а при використанні інформації про один попередній символ до 3,6 біта на символ. Врахування інформації про два попередні символи зменшує середню довжину коду до 3,2 біта на символ. Покращення ущільнення при врахуванні попередніх символів відзначається і для інших типів даних: об’єктних файлів, зображень, аудіоданих.

Моделювання, яке дає оцінку імовірності появи символу в залежності від попередніх, або контексту, називається контекстним моделюванням. Контекстне моделювання практично завжди застосовується як адаптивне.

Контекст в широкому смислі – це сукупність символів, що оточують поточний символ. Розрізняють також лівосторонні і правосторонні контексти, тобто послідовності символів, що примикають до поточного символу зліва і справа, відповідно. При контекстному моделюванні під контекстом розуміють саме лівосторонній контекст, оскільки контекстне моделювання практично завжди застосовується як адаптивне [1].

Якщо довжина контексту обмежена, то такий підхід називається контекстним моделюванням обмеженого порядку (finite context modeling), при цьому під порядком розуміється максимальна довжина N контекстів, що використовуються.

Найбільш відомим алгоритмом з цього сімейства є алгоритм PPM (англ. Prediction by Partial Matching – прогноз щодо часткового збігу). Алгоритм представляє собою адаптивний статистичний алгоритм стиснення даних без втрат, який заснований на контекстному моделюванні і прогнозі. Модель PPM використовує контекст, як безліч символів в стислому потоці, що передують даному, щоб пророкувати значення символу на основі статистичних даних. Сама модель PPM лише пророкує значення символу, безпосередній стиск здійснюється алгоритмами ентропійного кодування, як наприклад, алгоритмом Хаффмана або за допомогою арифметичного кодування [1].

Серед програмних додатків, які використовують даний алгоритм виділяється серія програмних додатків PAQ, яка розроблена Меттом Махоуні у 2002 році. Він використав поліпшену версію алгоритму PPM з використанням техніки під назвою «контекстне змішування», яка успішно дозволяє використовувати більше однієї статистичної моделі, щоб поліпшити прогноз по частоті появи символів. Також, можна сказати, що алгоритм, реалізований в PAQ набирає значної популярності завдяки дуже хорошому коефіцієнту стиснення (хоча і працює він дуже повільно). Але завдяки збільшенню швидкодії комп'ютерів швидкість процесу ущільнення стане менш критичною.

Отже, було вирішено обрати алгоритм ущільнення даних, що використовується серією ахіваторів PAQ, так як він показує найкращі показники стиснення даних.

1.4 Висновки

Аналіз стану питання показав, що архіватори є актуальними у наш час, так як здатні зменшити розмір файлу завдяки вирахування послідовностей, які повторюються. Але вони мають один вагомий недолік – файли в них стають недоступними безпосередньо.

Аналіз аналогів показав, що ідеального архіватора не існує. Кожен має якісь недоліки. На основі аналізу було складено список характеристик, якими повинен володіти програмний продукт:

– Не вимагає встановлення.

– Зрозумілий інтерфейс.

– Зручна навігація між папками.

– Шифрування даних

– Прискорений метод Drag&Drop (перетягування в активне вікно)

– Безкоштовний

– Наявність перекладу інтерфейсу на українську мову

Також було проаналізовано функціональні характеристики архіватора. Вони такі:

– Оптимальна степінь стиснення тексту – від 4 до 18.

– Оптимальна степінь стиснення зображення – від 24 і вище.

– Оптимальна степінь стиснення медіа – від 1,025.

Аналіз методів або алгоритмів ущільнення інформації без втрат показав, що доцільно використовувати алгоритм ущільнення PPM з використанням техніки під назвою «контекстне змішування», яка успішно використовується у архіваторах серії PAQ.

2 РОЗРОБКА СТУКТУРИ ТА АЛГОРИТМІВ РОБОТИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка структури інформаційної системи

Для реалізації функціоналу програмного продукту його доцільно розбити на модулі, які відповідають за ті чи інші властивості та функції. Це змешує складність розуміння коду та впорядковує для інших.

Також передбачається, і розробка форм, які необхідні для маніпулювання дій користувача з програмою. Форми або вікна є невід'ємною частиною графічного інтерфейсу користувача.

Отже, передбачено використання наступних модулів та форм:

– Головна форма – це форма, у якій безпосередньо відбуваються усі дії користувача. Вона викликається з запуском програми.

– Форма «Про програму» – це форма, яка слугує для ознайомлення користувачів із коротким описом програмного продукту, а саме: його призначенням, розробником та ліцензією. Викликається з головної форми.

– Форма для вибору степеня або режиму стиснення файлу або папки в залежності від вибору користувача. Існує 6 базових режимів: від найменш ефективного до найбільш. Ця форма, також як і попередня викликається з головного вікна і слугує для передачі аргументу (режиму стиснення), який ввів користувач.

– Заголовний файл «libzpaq.h» містить описи та методи класів, необхідних для архівації та розархівування файлів. Підключається до головної форми.

– Вихідний код «libzpaq.cpp» містить реалізацію методів заголовного файлу «libzpaq.h».

Загальний взаємозв'язок між модулями та формами проілюстровано на рисунку 2.1.

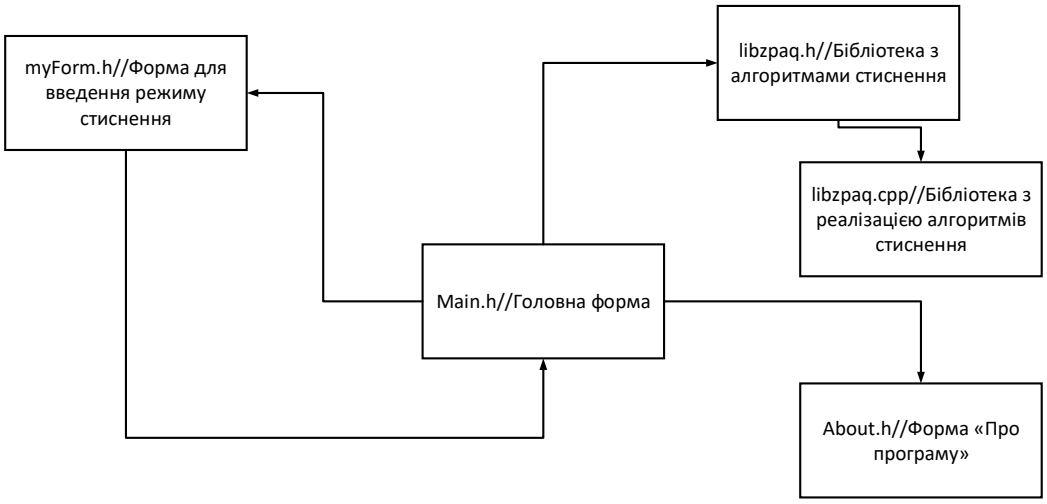


Рисунок 2.1 – Загальний взаємозв’язок між модулями та формами

З даного рисунку можна зауважити, що головна форма включає в себе усі ці модулі та форми. Вона ніби володіє ними визначає їх поведінку, крім модулів з описами алгоритмів. Також головна форма та форма для введення режиму стиснення мають двосторонній зв’язок, оскільки поведінка головної форми залежить в деякій мірі від форми для введення.

2.2 Опис алгоритму ущільнення даних з використанням контекстного моделювання

Серед усіх версій PAQ ZPAQ є найбільш сучасною, тому вона буде використовуватися у програмному додатку.

Відкритий стандарт ZPAQ визначає стиснене представлення для одного або більше послідовностей байтів (8 бітових значень). Потік ZPAQ складається з послідовності блоків, які можуть бути розпаковані незалежно. Блок складається з послідовності сегментів, які повинні бути розпаковані послідовно з початку блоку. Кожен сегмент може представляти масив байтів у пам'яті, файлі або сусідній частині файлу.

ZPAQ за бажанням використовує алгоритм стиснення даних контекстного змішування на основі серії PAQ (PAQ8, PAQ9, LPAQ). Розпакований потік

декодується один біт за раз, упаковується в байти, а потім перетворюється через необов'язковий постпроцесор, щоб скасувати перетворення, які мали на меті зробити дані більш стислими. Щось декодується за допомогою моделі, яка передбачає (визначає ймовірність), наступний біт на основі раніше розшифрованих бітів - арифметичного декодера, який приймає прогноз і стиснуті дані, і виводить біт. Біт повертається до моделі, щоб вона могла уточнити майбутні прогнози.

Декодовані дані для кожного блоку починаються з прапору, щоб вказати, чи слід його виводити безпосередньо або після обробки. В останньому випадку він складається з програми, а потім з вхідних даних. Виходом цієї програми є висновок декомпресора. У будь-якому випадку вихід може бути розділений на окремі масиви або файли, як описано в заголовках сегментів [10].

Модель являє собою набір компонентів, які роблять незалежні прогнози за контекстом та / або комбінуючи прогнози інших компонентів. Кожен компонент має контекст, який обчислюється з раніше розшифрованих бітів за допомогою програми, описаної в заголовку блоку. Наприклад, контекст може бути хешом останніх 20 біт. Для кожного блоку може бути довільно пов'язаним до 255 компонентів наступних типів:

– CM - Context Model - таблиця відображає контекст для прогнозування (спочатку $p_0 = p_1 = 1/2$) та лічильник. Після того, як трохи декодується, прогноз налаштовується пропорційно до помилки прогнозування та обернено пропорційний підрахунку, а підрахунок збільшується до заданого ліміту.

– ICM - Непряма контекстна модель. Хеш-таблиця вказує контексту на бітову історію або стан, що представляє обмежені значення раніше відомі біти 0 і 1 (спочатку обидва 0) і найновіший біт. Друга таблиця відображає історію до передбачення. Після того, як трохи декодується, історія оновлюється, а прогноз налаштовується, щоб зменшити помилку прогнозування.

– MATCH - індекс вказує контексту на останній випадок того самого контексту у вихідному буфері. Біти, що слідують за матчем, передбачаються з упевненістю, яка залежить від тривалості матчу. Індекс оновлюється кожні 8 біт.

– AVG - два прогнози поєднуються зваженим усередненням. Модель визначає

ваги.

– MIX2 – два прогнози поєднуються зваженим усередненням. Вага (спочатку 1/2) вибирається з таблиці за контекстом. Після розшифрування вага коригується пропорційно похибці прогнозування, коли різниця вхідних значень, що складаються з певною швидкістю навчання. Це сприяє найбільш точному компоненту в кожному можливому контексті.

– MIX – змішувач, подібний до MIX2, але з входом від суміжного блоку m-компонентів. Існує вага для кожного вводу. Ваги налаштовуються на користь найбільш точних компонентів, але не обмежуються доповненням до них. Маси спочатку 1/м.

– ISSE – оцінка непрямих вторинних символів. У таблиці відображається контекст бітової історії, як і ICM. Історія використовується як контекст для 2 вхідних MIX-ів із незалежними вагами і один вхід фіксований. Модель PAQ9A – це каскад цього ISSE з все більш високими замовленнями на контекст. Після розшифрування історія біт і вага оновлюються, як за допомогою ICM та MIX.

– SSE – оцінка середнього символу. SSE бере квантований прогноз введення та контекст і виводить нове (інтерполюється) прогноз із таблиці. Після декодування, найближчий запис до таблиці регулюється, щоб зменшити помилку прогнозування, як і в КМ.

Архітектура алгоритму наведена нижче, де Arith означає арифметичне кодування (рис. 2.2).

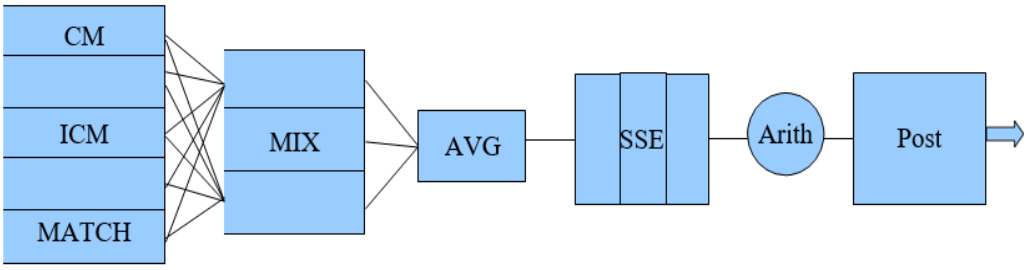


Рисунок 2.2 – Схема алгоритму ZPAQ

Компонентні вхідні та вихідні прогнози виражаються як лог-коефіцієнт. Якщо компонент передбачає, що 0 або 1 буде відбуватися з ймовірністю p0 та p1, то виходом буде (p= stretch (p_1) = ln (p_1/p_0). Прогнози обчислюються по фіксованій

послідовності компонентів із входом від попередніх компонентів. Вхід до арифметичного декодера - p1 з останнього компонента, де p_1 = squash (p) = 1 / (1 + e-p) - це зворотне розтягування (p_1) (рис. 2.3). Це впливає на зважування моделей з високими прогнозами надійності (великими величинами) більш сильно. Хеші контекстів обчислюються програмою, описаною в заголовку блоку, і виконуються один раз кожні 8 біт під час декомпресії, і об'єднуються з частково декодованим поточним байтом для формування повного контексту [10].

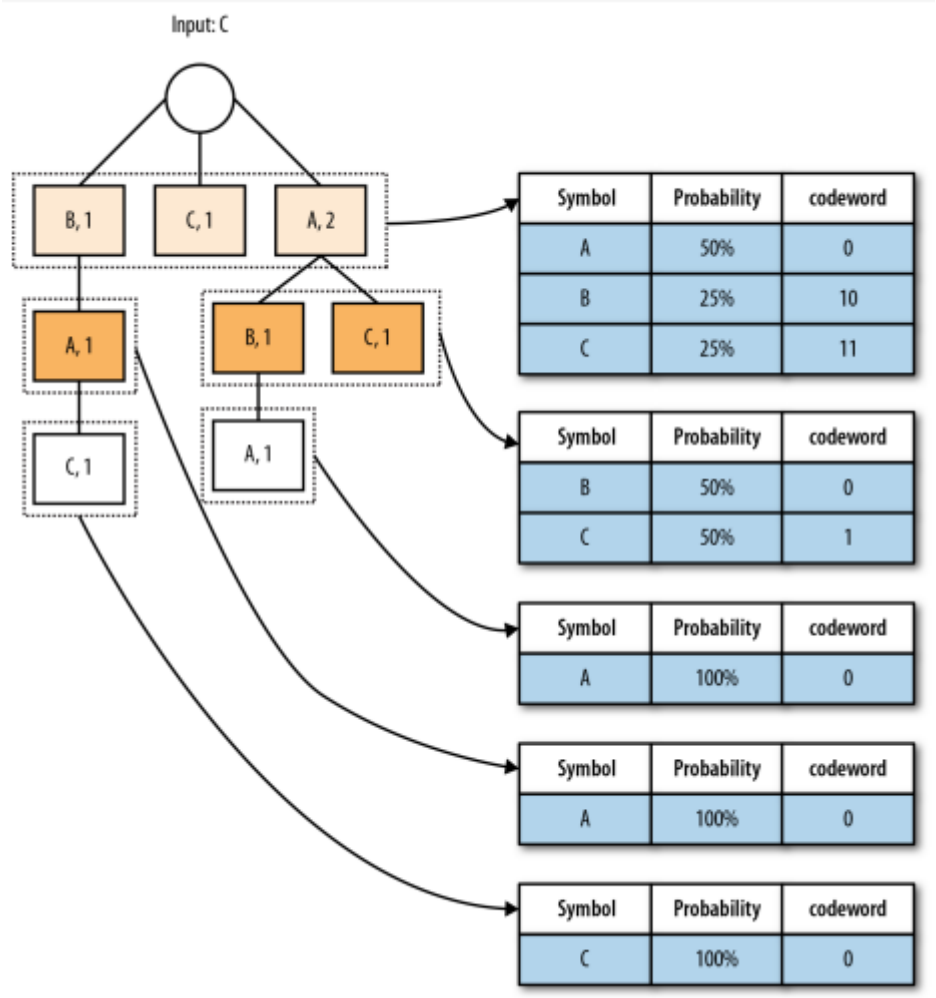


Рисунок 2.3 – Схема моделювання даних та їх декодування

Стиснені дані представляють двозначний номер високої точності в діапазоні (0, 1) з найважливішими бітами частки в першу чергу. Арифметичний декодер підтримує діапазон (низький, високий), спочатку (0, 1), який обмежує дані і зменшується, оскільки відбувається декодування. Декодер отримує прогноз і розподіляє діапазон на дві частини пропорційно p_0 і p_1. Яка частина зараз містить

стиснуті дані, визначається декодований біт і новий діапазон. Арифметичне кодування ефективно кодує кожен біт Y за ціною дуже близько до теоретичної межі бітів $\log_2(1/p_y)$. Для позначення кінця даних кожен декодований байт передує біт EOS (кінець сегмента), який становить 1 після останнього байта, кодований з p1 дуже близько 0. Декодер спроектований так, що кінець кодованого сегмента також може бути знайдений шляхом швидкого сканування без розпакування даних.

Модель і пост-процесор ініціалізуються на початку кожного блоку та підтримують інформацію про стан між сегментами. Арифметичний кодер ініціалізується на початку кожного сегмента. Границі сегмента невидимі для моделі; блок з'являється як безперервний потік байтів. Мета сегментів полягає в тому, щоб дозволити декомпресії до різних напрямків (наприклад, для різних файлів) і для сигналу пост-процесора в кінці кожного файлу.

Постпроцесор дозволяє виключити контекстну модель. У цьому випадку дані можуть зберігатися не стиснуті або розпаковуватися виключно постпроцесором за допомогою довільного алгоритму. Мета - дозволити швидке зберігання та вилучення злегка стиснутих або не стиснутих даних.

2.3 Висновки

Було побудовано базову структуру програмного додатку, яка складається з 3 форм: однієї головної та двох додаткових, та 2 модулів, які відповідають за виконання алгоритму стиснення файлів. Також, було проаналізовано алгоритм, який реалізований у ZPAQ, який є сучасною модифікацією алгоритму PAQ, та на основі цього аналізу складено схему, яка складається із таблиці, яка відображає контекст прогнозування, хеш-таблиці, змішувачів та арифметичного кодера.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА ЇЇ ТЕСТУВАННЯ

3.1 Аналіз та обґрунтування вибору мови програмування

Для реалізації програми доцільно використати об’єктно-орієнтовану мову програмування. До найпопулярніших із них належать C++, C# та Java.

C++ – високорівнева мова з підтримкою об'єктно-орієнтованої, узагальненої та процедурної парадигм програмування. Розроблена Б'ярном Страуструпом у 1979 році. Стандартна бібліотека C++ включає стандартну бібліотеку Сі з невеликими змінами, які роблять її відповіднішою для мови C++. Інша велика частина бібліотеки C++ заснована на Стандартній Бібліотеці Шаблонів (STL). Вона надає такі важливі інструменти, як контейнери (наприклад, вектори і списки) та ітератори (узагальнені вказівники), що надають доступ до цих контейнерів як до масивів. Крім того, STL дозволяє схожим чином працювати і з іншими типами контейнерів, наприклад, асоціативними списками, стеками, чергами. Використовуючи шаблони, можна писати узагальнені алгоритми, здатні працювати з будь-якими контейнерами або послідовностями, доступ до членів яких забезпечують ітератори [11].

C# – об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників, C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C# не підтримує множинне спадкування класів (на відміну від C++) або виведення типів (на відміну Haskell) [12].

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми

компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Sun Microsystems надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU General Public License. Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування [13].

Оцінюючи вищеописані особливості, виділимо деякі переваги використання C++ при розробці програми:

– швидкість роботи програм на C++, порівняно із C# та Java, найвища;

– наявність стандартної бібліотеки STL дозволяє оперувати складними структурами даних;

– наявність зручних інструментів для роботи із реєстром та файловою системою;

– наявність готових інтерфейсів для швидкої та зручної роботи із базою даних.

Порівняно із C++, мови Java та C# мають і певні недоліки:

– у Java ціною кросплатформеності є вимога наявності на комп'ютері віртуальної Java-машини, що приводить до уповільнення обчислень і практичної неможливості використання нових можливостей апаратної архітектури;

– збірка сміття призводить до втрати ефективності;

– використання вказівників у багатьох випадках є потужним та необхідним засобом, відсутнім у Java та C#.

Головною перевагою мови C++ є її швидкодія, а також можливість роботи на низькому рівні з пам'яттю, адресами, портами. Враховуючи

перераховані вище переваги, для розробки програми буде використано мову програмування C++.

3.2 Аналіз та обґрунтування вибору середовища розробки

Для вибору інтегрованого середовища розробки було розглянуто наступні середовища: Microsoft Visual Studio, Qt Creator, Eclipse CDT, CodeLite.

Microsoft Visual Studio – це серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight [14].

Qt Creator – інтегроване середовище розробки, призначене для створення кросплатформенних застосунків з використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в якій використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками. Qt Creator може використовувати GCC або Microsoft VC++ як компілятор і GDB як зневаджувач. Для Windows версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW [15].

Eclipse – це вільне модульне інтегроване середовище розробки програмного забезпечення на C/C++. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, засоби для управління сирцевими кодами, візуальні побудовники GUI тощо. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування [16].

CodeLite – це безкоштовне кросплатформенне середовище розробки програмного забезпечення для мов C, C++ з відкритим вихідним кодом, яке використовує інструментарій wxWidgets.

В таблиці 3.1 наведено порівняння вище описаних середовищ розробки.

Таблиця 3.1 – Порівняння інтегрованих середовищ розробки

Характеристика	Інтегровані середовища розробки			
	CodeLite	Eclipse CDT	Qt Creator	Microsoft Visual Studio
Розробка GUI	1	1	1	1
Покриття кода	0	0	0	1
Автодоповнення	1	1	1	1
Статичний аналіз коду	1	0	0	1
Налагоджувальник	1	1	1	1
Загальний коефіцієнт	4	3	3	5

Для проведення аналізу засобів розробки обрано такі середовища розробки, що використовують мову C++, як базову для програмування. Для проведення аналізу засобів розробки використано такі середовища розробки: CodeLite, Eclipse CDT, Qt Creator, Microsoft Visual Studio. Відповідно до характеристик аналізу обрано середовище Microsoft Visual Studio, оскільки дане середовище має всі оптимальні характеристики для вирішення поставлених задачі.

3.3 Розробка інтерфейсу користувача

Графічний інтерфейс користувача – це такий тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки.

Виконання дій в ГІК – це безпосередня маніпуляція з графічними елементами. Окрім комп'ютерів, GUI використовується в мобільних пристроях, таких, як мобільні телефони, планшети, електронні книги, портативні медіаплеєри тощо.

Є одновіконні (однодокументні), багато віконні (багато документні) та вкладочні графічні інтерфейси користувача.

В одновіконному інтерфейсі є одне головне вікно. Всі інші вікна – це діалогові вікна. Використання діалогових вікон є основним типом взаємодії користувача з комп’ютером. Діалог – це регламентований обмін інформацією між користувачем і пристроєм, що відбувається у реальному часі і направлений на вирішення певної задачі або формування визначеного переліку дій. Кожен діалог складається з окремих процесів вводу-виводу або передачі. Обмін інформації у діалогових системах відбувається за допомогою повідомлень. Повідомлення – числова інформація, що передається від користувача до приймача і організовує функціональну роботу пристроїв.

В багато віконному інтерфейсі є одне головне вікно. Всі інші вікна – дочірні вікна. Дочірнє вікно обмежено робочою зоною його батьківського вікна. Програма зазвичай використовує дочірні вікна, щоб поділити робочу зону батьківського вікна на функціональні області. Дочірнє вікно має тільки одне батьківське вікно, але батько може мати будь-яке число дочірніх вікон. Кожне дочірнє вікно, у свою чергу, може мати дочірні вікна. Дочірнє вікно може накладатися на інші дочірні вікна в тій же самій робочій області.

Закладковий інтерфейс – це різновид, в якому кожен документ відображається на окремій закладці загального вікна. Його інша назва – багато документний інтерфейс з закладками. Застосовується в браузерах, середовищах розробки (для

відображення кількох документів одного проекту), для групування елементів по групам.

В результаті, обрано одновіконний інтерфейс. Для реалізації програмного продукту використання багато віконного інтерфейсу не є оптимальним, тому що в створенні дочірніх вікон немає потреби.

Загалом інтерфейс програми складається із різних вікон, але всі дії розпочинаються у головному вікні, яке складається із головного меню, панелі інструментів (кнопок), робочої області(провідника) та панелі статусу (журналу виконання). Схематично його можна показати так (рис. 3.1).

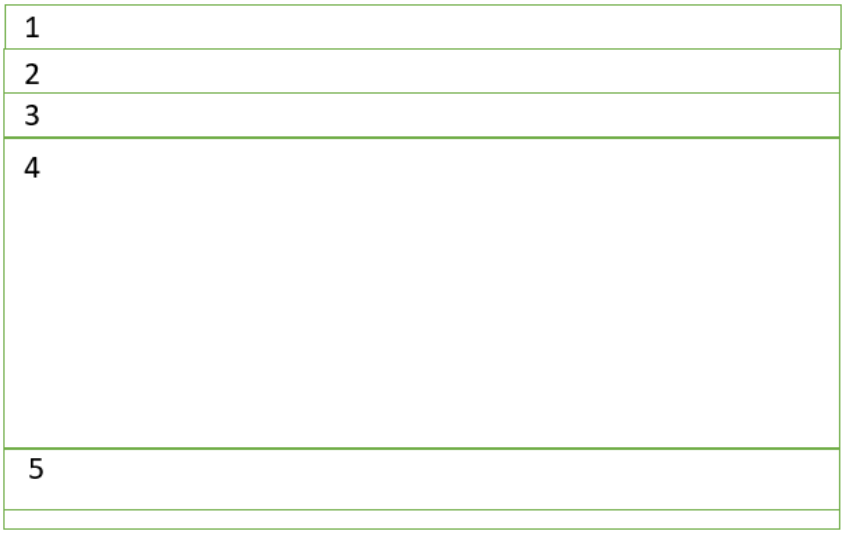


Рисунок 3.1 – Прототип інтерфейсу головного вікна продукту

Умовні позначення:

1. Панель керування вікном. Містить три кнопки для керування вікном: згорнути, максимізувати вікно та закрити його.

2. Панель меню. Панель меню містить наступні пункти меню:

а. Файл.

1. Відкрити.

2. Вихід.

б. Команди.

1. Додати. Створює або додає файли до архіву.

2. Видобути. Вилучає файли з архіву.

3. Переглянути. Переглядає файли або запускає їх.

4. Копіювати. Копіює файли в задану користувачем папку.

5. Видалити. Видаляє файли з комп’ютера, а не переміщує їх у корзину.

в. Консоль. Перехід користувача у режим консолі.

г. Мова.

1. Українська. Змінює мову усіх вікон на українську.

2. Англійська. Змінює мову усіх вікон на англійську.

д. Довідка.

1. Інструкція користувача.

2. Про програму.

3. Панель інструментів. Панель інструментів містить наступні інструменти(зліва направо:

1) Додати.
2) Видобути.
3) Переглянути.
4) Копіювати.
5) Видалити.

4. Робоча область (провідник). Робоча область представляє собою провідник, який слугує для зручної навігації користувача між папками.

5. Рядок стану (журнал виконання). Інформує користувача про помилки в програмному продукту та про хід виконання архівації та розархівування.

На рисунку 3.2 можна побачити як насправді виглядає інтерфейс головного вікна інфоррмаційної системи.

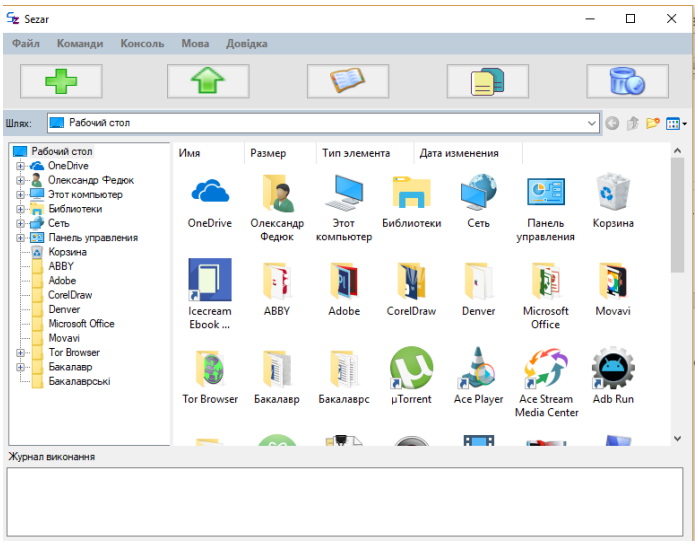


Рисунок 3.2 – Інтерфейс головного вікна програмного продукту

Також вікно здатне масштабуватися. Вікно у повному розмірі зображене на рисунку 3.3.

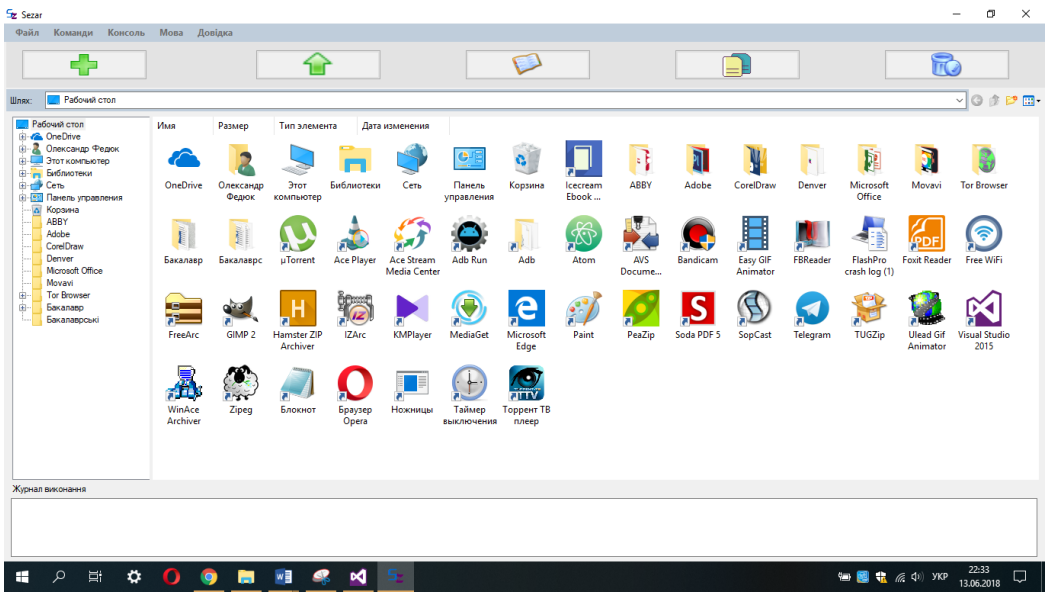


Рисунок 3.3 – Вікно програмного продукту у повному розмірі

Також було створено вікно «Про програму», яке надає користувачеві інформацію про програму, а саме: логотип, назва програми, опис програми, прізвище розробника та тип ліцензії програмного продукту. Це вікно зображене на рисунку 3.4.

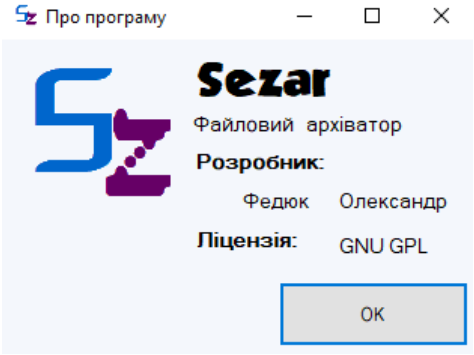


Рисунок 3.4 - Вікно «Про програму»

Вікно «Про програму» реалізоване за допомогою наступних елементів:

– Button – кнопка для закриття вікна.

– Label – текстовий надпис.

– pictureBox – вставка рисунка.

Також користувач викликає вікно провідника, яке викликається за допомогою класу OpenFileDialog/SaveFileDialog. Воно потрібні тоді, коли необхідно відкрити або зберегти файл. Вони викликають стандартні діалогові вікна для відкриття або збереження файлу.

Інтерфейс головного вікна містить так-звані «toolTip», основною метою яких є підказати користувачу назву кнопки або ще чогось іншого.

Інтерфейс є двомовним: англійським або українським. Якщо потрібно змінити мову, то натисніть на пункт меню «Мова» або «Language» і оберіть мову, яка рідніша.

Всі процеси у вікні виконуються асинхронно, тому вікно не зависає. Така асинхронність забезпечується за допомогою спеціальних об’єктів, в які загортаються найбільш складні операції з обчислювальноъ точки зору: архівація, розархівування, копіювання та видалення.

Підтримується технологія «Drag&Drop» завдяки наявності провідника на робочому просторі, який реалізований за допомогою динамічної бібліотеки «GongSolutions», яка пропонує рішення для Windows Forms.

Ще одне вікно було створено для задання режиму ущільнення. Воно зображене на риунку 3.5.

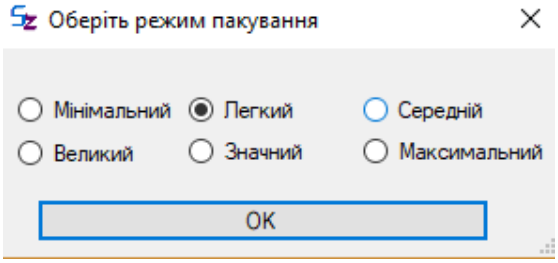


Рисунок 3.5 - Вікно для задання режиму пакування(стиснення)

Це вікно містить елементи «comboBox», які призначені для вибору одного рішення користувачем, та кнопку при натисканні на яку головне вікно отримує даний параметр(режим пакування).

Інтерфейс користувача створений за-допомогою технології Windows Forms. Вона спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код – це класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C#, C++, так і на VB.Net, J# та ін.

3.4 Тестування інформаційної системи

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. При тестуванні виявляються недоліки: відмови і дефекти як причини порушення роботи програми, збої як небажані ситуації, помилки як наслідки збоїв та ін. Тестування вважається успішним, якщо знайдено дефект або помилка, і вони відразу усуваються. Зазвичай для проведення тестування застосовуються методи структурного («білий ящик») та функціонального («чорний ящик») тестування [17].

При функціональному тестуванні вихідний код недоступний. Суть полягає в перевірці відповідності поведінки програми її зовнішній специфікації. Критерієм повноти тестування вважається перебір всіх можливих значень вхідних даних, що здійснити на практиці надзвичайно важко.

При структурному тестуванні текст програми відкритий для аналізу. Суть даного методу полягає в перевірці внутрішньої логіки програмного забезпечення. Повним тестуванням у цьому випадку буде таке, що приведе до перебору всіх можливих шляхів на графі передач керування програми. Число таких шляхів може досягати десятків тисяч. Крім того, виникає питання про створення тестів, що забезпечують дане покриття. Здійснити повне всеохоплююче тестування навіть простої програми вкрай важко, а часом і неможливо в силу обмеженості часу й ресурсів. Отже, необхідно мати певні критерії, за якими мають обиратися контрольні приклади та критерії зупинки процесу тестування.

Оскільки метод «білого ящика» є досить громіздким, а програма досить функціональна, доречно буде скористатися методом «чорного ящика», не втручаючись у вихідний текст програми.

– Протестуємо програмний продукт на ущільнення папки з текстовими файлами.

Отже, запускаємо нашу програму. Обираємо папку, тобто клацаємо лівою кнопкою миші на папку, яку хочемо стиснути, в нашому випадку це папка «wordpress-4.9.6-ru_RU» та натискуємо на значок «+» у лівому верхньому куті (рис. 3.6).

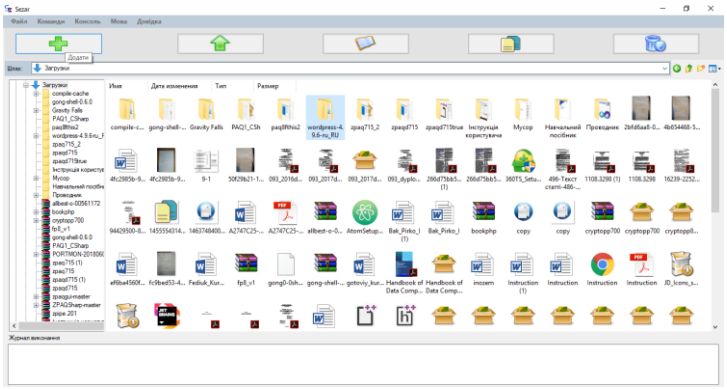


Рисунок 3.6 – Головне вікно

Відкривається вікно для вибору режиму стиснення. Обираємо максимальний режим стиснення даних, який найбільш ефективно ущільнює дані, та натискуємо на кнопку «ОК» (рис. 3.7).

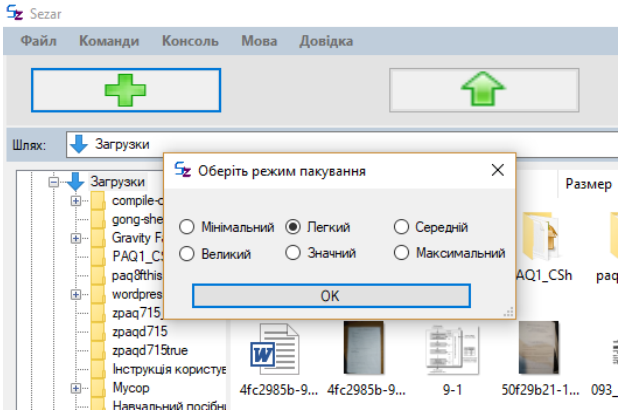


Рисунок 3.7 - Вікно для вибору режиму стиснення даних

Після чого викликається діалогове вікно для задання місця зберігання файлу архіву. Потрібно обрати шлях, який не містить кирилиці, інакше буде помилка в програмі. Вікно автоматично присвоїть назву архіву і встановить розширення «.zpaq». Але назву можна змінити, якщо бажаєте. Назву буде змінено на «WordPRESS». Після чого потрібно натиснути на кнопку «Сохранить» (рис. 3.8).

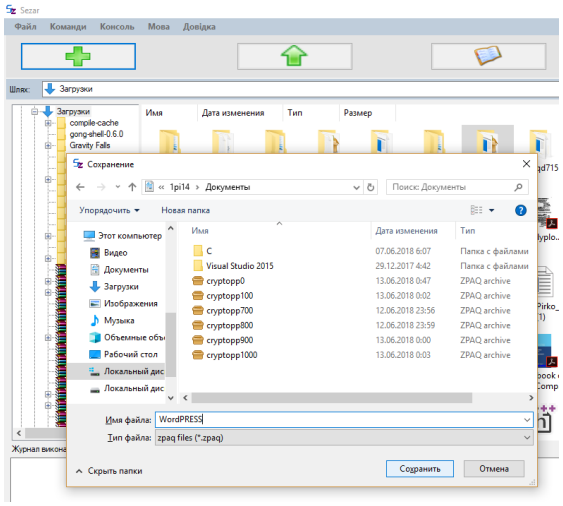


Рисунок 3.8 - Вікно для зберігання архіву

Після цього у журналі виконання з’являються записи (рис. 3.9), які свідчать про початок архівації даної папки. Ці записи дають інформацію про файли, які

заносяться до архіву, їх загальний прогрес архівації та час, який залишився для закінчення архівації.

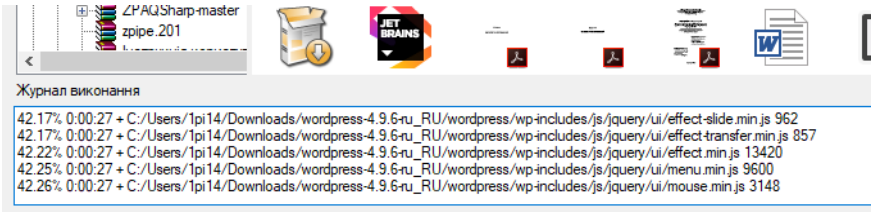


Рисунок 3.9 – Записи в журналі виконання

Після деякого часу процес архівації закінчиться. Про успішне виконання архівування буде наявність словосполучення в журналі виконання «(all OK)» (3.10).

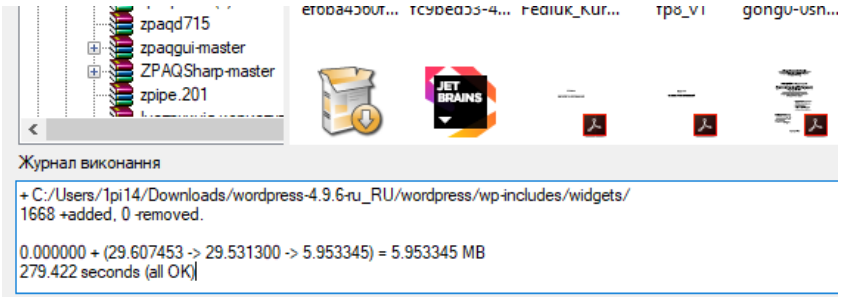


Рисунок 3.10 –Кінець архівації

Два останніх рядки в журналі виконання містять інформацію про початковий розмір файлу(29,607453 Мб), розмір після стиснення файлу(5,9533345 Мб) та час, за який за який відбулася архівація(279,422 с). Також по закінченню архівації користувач отримує повідомлення про успішне виконання архівації (рис. 3.11).

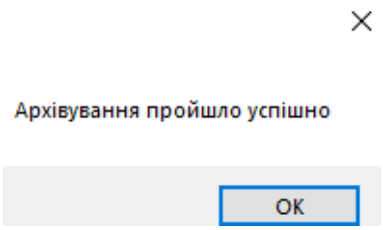


Рисунок 3.11 – Повідомлення про успішне закінчення архівування

Отже, папку з текстовими даними розміром 29 Мб було стиснуто до 5 Мб. В результаті такого стиснення розмір файлу зменшився в 6 разів.

Розархівування даних мало чим відрізняється від архівації. Потрібно обрати архів, далі натиснути на зелену стрілку, яка направлена вгору. Внаслідок чого викликається діалогове вікно для задання місця зберігання розпакованого архіву. У журналі виконання робляться відповідні записи про вилучення файлів з архіву та час, який залишився до закінчення процесу. Після успішного виконання буде отримано відповідне повідомлення.

– Протестуємо дію кнопок «Копіювати», «Переглянути» та «Видалити».

Кнопка «Копіювати» слугує для копіювання файлу або папки у вказану користувачем папку. Потрібно орати файл або папку для копіювання та натиснути на 4-ту кнопку зліва (рис. 3.12).

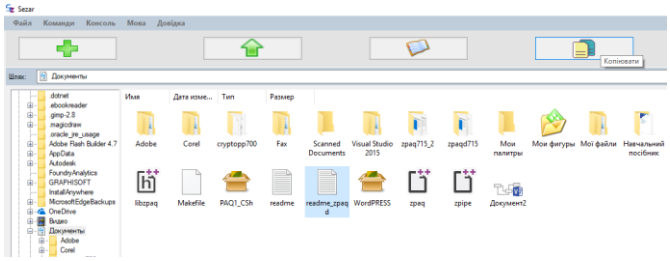


Рисунок 3.12 – Обираємо файл для копіювання

Потрібно обрати місце для зберігання копії та натиснути кнопку «Сохранить» (рис. 3.13).

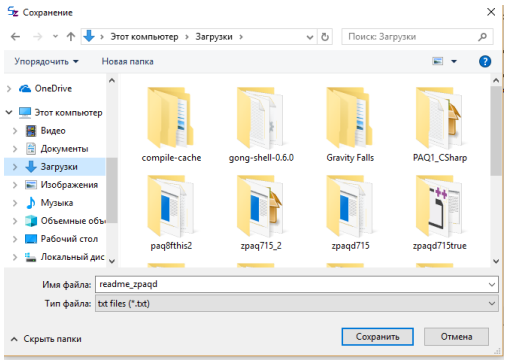


Рисунок 3.13 – Обираємо місце для копіювання

У разі, якщо копіювання файлу завершилося успішно, користувач отримує відповідне повідомлення (3.14).

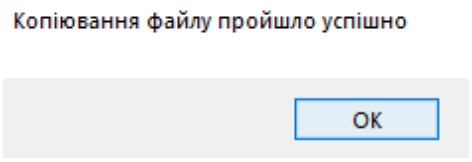


Рисунок 3.14 – Повідомлення про успішне виконання копіювання

Перевіряємо чи дійсно це наш файл (3.15).

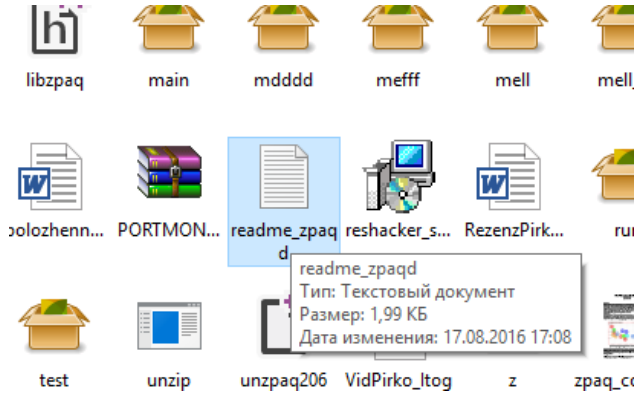


Рисунок 3.15 – Перевірка копіювання файлу

Так, це наш файл. Отже, копіювання працює дійсно належним чином.

Кнопка «Переглянути» слугує для відкриття файлу. Обираємо файл та натискуємо третю кнопку зліва або значок з відкритою книжкою (рис. 3.16).

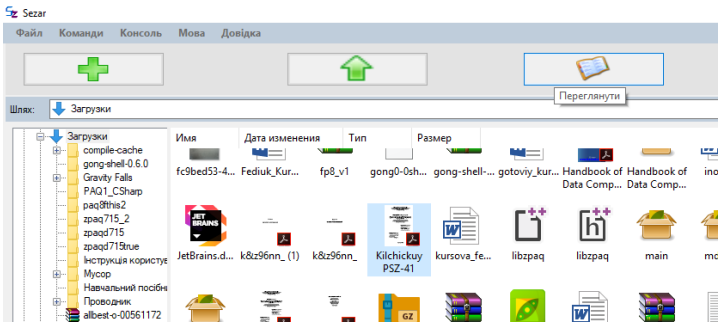


Рисунок 3.16 – Обираємо файл для перегляду та натискуємо на виділену кнопку

Файл відкривається відповідно програмою, встановленою за замовчуванням (рис. 3.17).

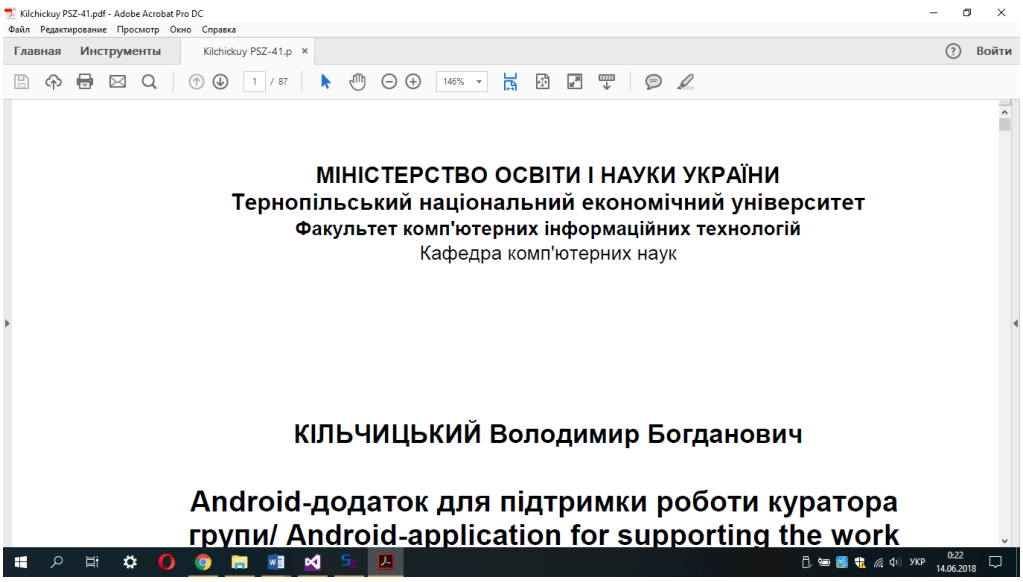


Рисунок 3.17 – Відкриття вибраного файлу програмою за замовчуванням

Також файл можна відкрити, натиснувши на ньому мишкою двічі, або клацнувши правою кнопкою миші та обравши пункт «Копировать» із випадаючого списку (рис. 3.18).

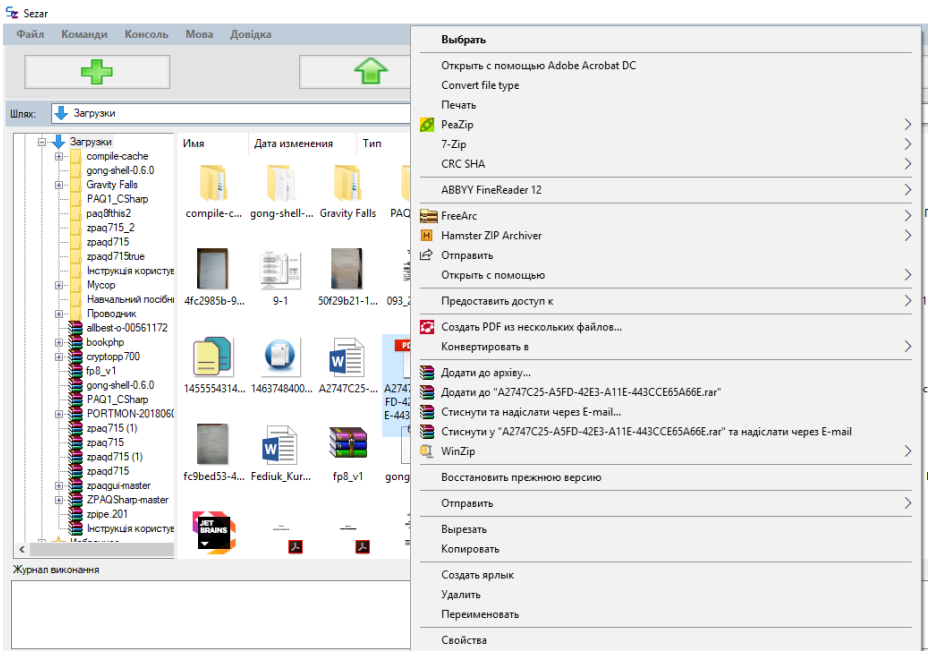


Рисунок 3.18 – Відкриття вибраного файлу іншим способом

Кнопка «Видалити» слугує для безповоротного видалення файлу або папки з комп’ютера.

Для видалення файлу необхідно обрати файл або папку для видалення та натиснути останню кнопку на панелі інструментів, яка має вигляд сміттєвого кошика (рис. 3.19).

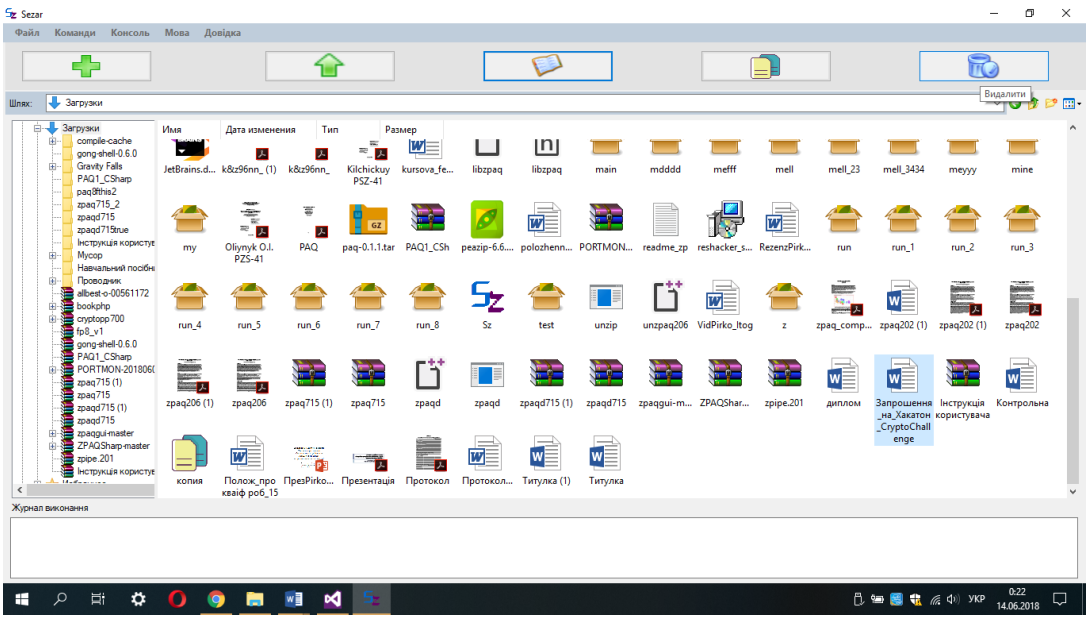


Рисунок 3.19 – Перевірка копіювання файлу

З’являється діалогове вікно, яке запитує користувача чи впевнений він у своєму рішенні (3.20).

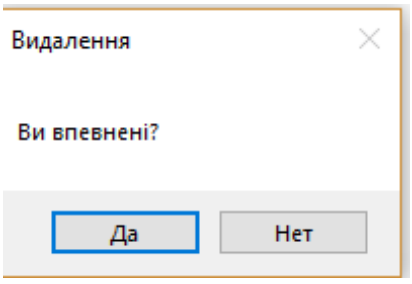


Рисунок 3.20 – Вікно для перевірки намірів користувача

У разі, якщо користувача відповів ствердно на попереднє запитання файл або папка видаляються. Також буде отримано повідомлення про успішне видалення (3.21).

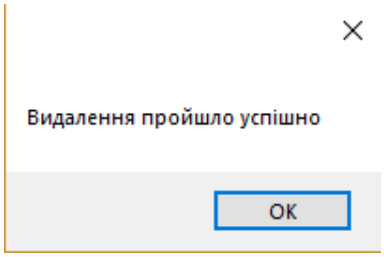


Рисунок 3.21 – Повідомлення про успішне видалення

– Протестуємо адаптивність головного вікна, тобто як змінюється вікно після зменшення або збільшення його ширини або висоти.

Зменшуємо ширину форми. Помічаємо, що розміри кнопок та інших елементів вікна також зменшуються у ширині (рис. 3.22).

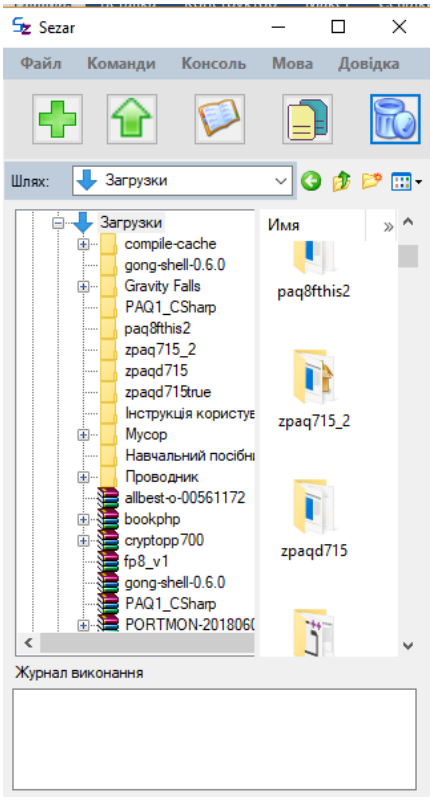


Рисунок 3.22 – Вигляд вікна після зменшення ширини вікна

Збільшуємо ширину головного вікна. Після збільшення ширини головного вікна, розміри кнопок, робочої області та журналу виконання також збільшилися у ширині (рис. 3.23)

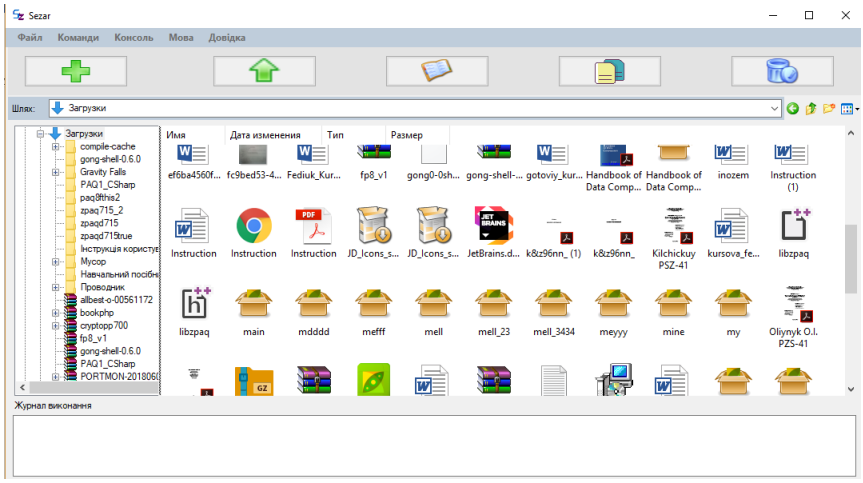


Рисунок 3.23 – Вигляд вікна після збільшення ширини вікна

Зменшуємо висоту головного вікна. Як бачимо, зменшується висота лише робочої області, яка містить провідник (рис. 3.24).

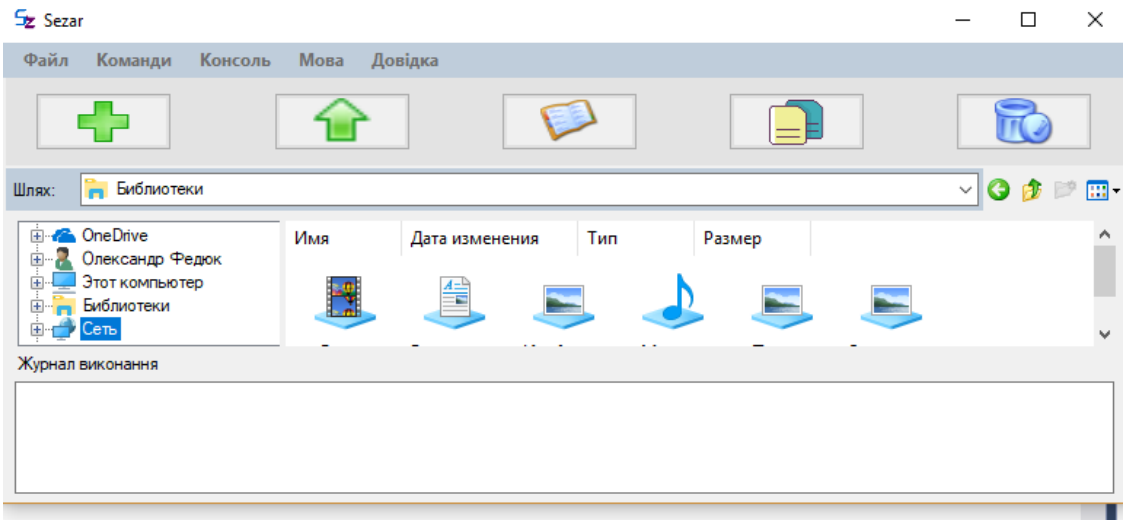


Рисунок 3.24 – Вигляд вікна після зменшення висоти вікна

Далі збільшимо висоту головного вікна, внаслідок чого збільшується висота робочої області (провідника) (рис. 3.25).

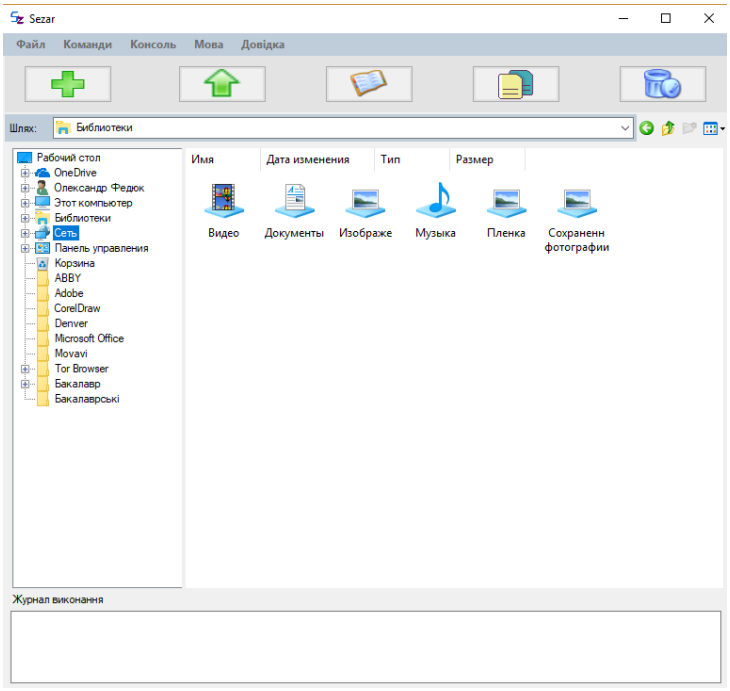


Рисунок 3.25 – Вигляд вікна після збільшення висоти вікна

– Протестуємо переключення мови інтерфейсу.

Для перевірки переключення мови інтерфейсу необхідно переключити мову на протилежну. Для цього необхідно навести мишкою на пункті меню «Мова» та обрати мову. У списку представлені дві мови: англійська та українська за замовчуванням. Обираємо англійську (рис. 3.26).

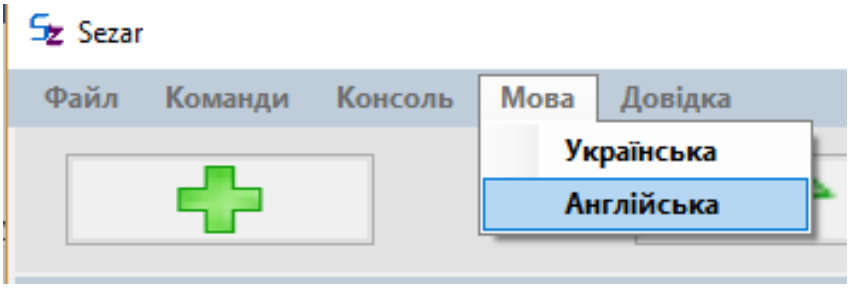


Рисунок 3.26 – Зміна мови інтерфейсу на англійську

Перевіряємо інтерфейс головного вікна та бачимо відповідно зміни. Пункти головного меню тепер на англійській мові. Назви деяких елементів, як наприклад над журналом виконання також на англійській мові (рис. 3.27).

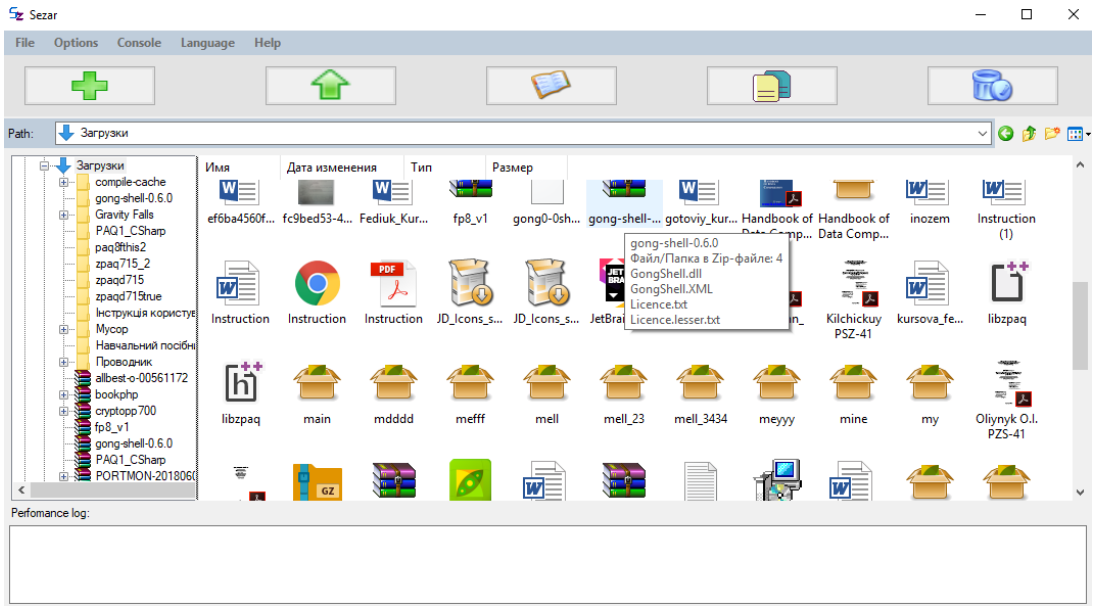


Рисунок 3.27 – Перевірка зміни мови

Перевіримо чи змінилася мова інших вікон: вікна «Про програму» та вікна для задання режиму пакуванню.

Для перевірки вікна «Про програму» обираємо пункт головного меню «Help» та натискуємо на його підпункті «About program» (рис. 3.28).

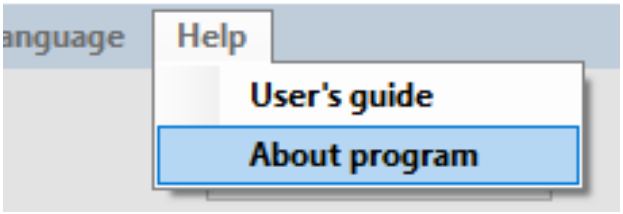


Рисунок 3.28 – Виклик вікна «Про програму»

Як бачимо інформація про програму тепер на англійській мові (рис 3.29).

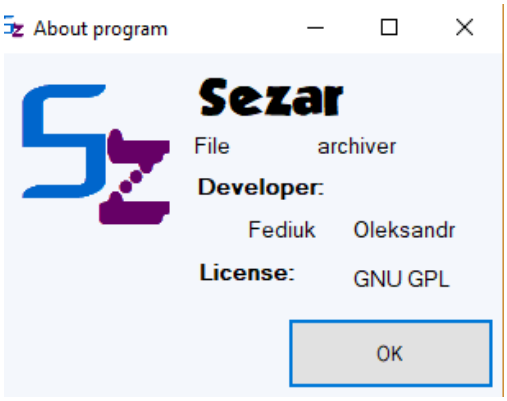


Рисунок 3.29 – Вікно «Про програму» на англійській мові

Для перевірки вікна задання пакування необхідно натиснути на кнопку додавання архіву, яка має вигляд зеленого знаку «+». Отримане вікно також англійською мовою (рис. 3.30).

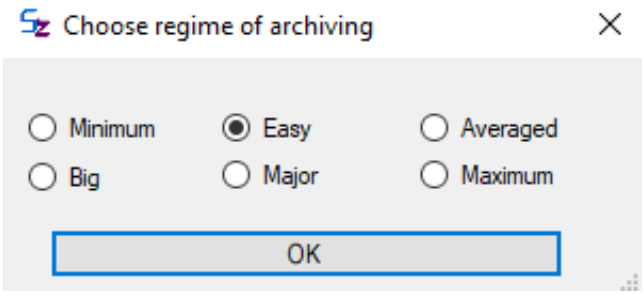


Рисунок 3.30 – Вікно для обрання режиму пакування на англійській мові

Також було проведено ущільнення текстових та графічних файлів розміром до 10 Гб. Максимальна степінь стиснення тексту становить 15, а зображення – 25. Щоправда часу для ущільнення потрібно досить таки багато. У випадку з текстом – 1600 с, а з зображеннями – 1200 с. Але в цілому програма задовольняє функціональні вимоги.

Отже, програма виконує свої безпосередні обов’язки, а саме ущільнює та витягує файли із архіву відповідно до визначених попередньо стандартів.

3.5 Висновки

У даному розділі обрана мова програмування C++, зокрема, через її швидкодію та наявність шаблонів проектування.

Також було розглянуто одновіконні та багато віконні інтерфейси. В результаті, для розробки програмного продукту обрано одновіконний інтерфейс, так як користувач буде взаємодіяти з головним вікном, яке в свою чергу буде викликати модальні вікна. Описаний інтерфейс головного вікна складається із панелі керування вікном, панелі меню, панелі інструментів, робочої області та рядку стану. В результаті отримано інтерфейс, що відповідає сучасним вимогам розробки графічних інтерфейсів і буде зручним у користуванні.

Було проведено тестуваннння додавання архіву до заданої користувачем папки. Були описана дії користувача та дія системи відповідно. В результаті стиснення 10 Гб папки з текстовими файлами було отримано показники ущільнення інформації. Тектстові файли здатні зменшуватися у 15 разів. Також було стиснуто папку з зображеннями. Її коефіцієнт стиснення – 25. Отже, вказаний програмний продукт задовольняє функціональні вимоги, які були задані у розділі 1.

Потім було перевірено дію кнопок «Копіювати», «Переглянути» та «Видалити». Кнопки працюють коректно, лише варто зауважити, що при видаленні об’єкт не переміщується в кошик, а видаляється взагалі.

Для користування програмою можна скористатись інструкцією користувача, яка коротко описує пункти меню і їх призначення, кнопки на панелі інструментів.

Таким чином можна зробити висновок, що поставлені задачі та цілі магістерської кваліфікаційної роботи виконані.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Роль та цінність економічного обґрунтування в останні роки істотно зросла. Під час цієї стадії інноваційного процесу виконується перехід від економічної моделі з нечіткими вимогами до більш розвиненої структури з визначеними економічними мотивами проекту. Вказаний етап є дуже важливим, оскільки його недооцінка може призвести до таких негативних наслідків як неефективність розроблюваної системи або нестача ресурсів при її розробці. Саме для передбачення вказаних проблем застосовується економічне обґрунтування прийнятих технічних рішень, яке перевіряє чи є розробка інформаційної системи є економічно доцільною.

Метою економічної частини магістерської кваліфікаційної роботи є оцінювання комерційного потенціалу розробки інформаційної системи автоматизованого формування розкладу, яка передбачає виконання наступних задач:

– визначити комерційний потенціал розроблюваної системи;

– передбачити витрати на виконання розробки та подальшу її реалізацію;

– визначити комерційний ефект від впровадження результатів наукової роботи;

– обчислити ефективність вкладених інвестицій та періоду їх окупності.

У даній роботі проводилась розробка інформаційної системи автоматизованого формування розкладу.

Для проведення технологічного аудиту було залучено двох незалежних експертів: керівник магістерської кваліфікаційної роботи – к.т.н., Крижановський Євген Миколайович та студент групи ІСТ-18м ВНТУ: Федюк О.П. Кожен з експертів був попередньо ознайомлений з розробкою інформаційної системи, та заповнив таблицю з відповідними критеріями для оцінки комерційного потенціалу наукової роботи за 5-ти бальною шкалою. Після цього було підраховано

середньоарифметичну суму балів та визначено рівень комерційного потенціалу вказаної розробки.

Оцінювання комерційного потенціалу розробки здійснено за 12-ма критеріями, наведеними в таблиці Д.1 Додатку Д.

Результати оцінювання комерційного потенціалу розробки занесені до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розроблюваної інформаційної системи

Критерії	Прізвище та ініціали експерта	
	Крижановський Є. М.	Федюк О.П.
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	4	3
4	3	3
5	3	3
6	4	3
7	3	3
8	4	3
9	3	3
10	3	4
11	3	3
12	4	4
Сума балів	$\overline{\text{СБ}} = 41$	$\overline{\text{СБ}} = 39$
Середньоарифметична сума балів $\overline{\text{СБ}}$	$\overline{\overline{\text{СБ}}} = \frac{\sum_1^i \overline{\text{СБ}}_i}{i} = \frac{41+39}{2} = 40$	

За даними, які представлені у таблиці 4.1, можна зробити висновок щодо рівня комерційного потенціалу розроблювальної інформаційної системи, використовуючи рекомендації, які наведені в таблиці 4.2.

Таблиця 4.2 – Рівні комерційного потенціалу розробки

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

Отже, за результатами, занесеними до таблиці 4.1, видно, що середньоарифметична сума балів дорівнює 40, тобто нова розробка має рівень комерційного потенціалу вище середнього.

Характеризуємо рівень комерційного потенціалу розробки:

- можливий шлях реалізації розробки: інформаційну систему впроваджено в департамент освіти;

- загальна якість розробки: головні результати, які були одержані в процесі вирішення встановлених задач становлять наукову новизну дослідження. Результати полягають у наступному: удосконалено інформаційну технологію розробки інформаційної системи створення розкладу з можливістю дуальної освіти, яка, на відміну від існуючих, передбачає можливість роботи з даними як за допомогою ліцензійного, так і за допомогою безкоштовного програмного забезпечення;

- перспективність розробки: на даний час в багатьох сферах управління використовуються інформаційні технології. Для управління даними, які містять також дані вчителів і керівників підприємств, на сучасному етапі використовують геоінформаційні системи. Управління навчальною і виробничою діяльністю в університетах є одним з напрямків освітньої політики;

- конкурентоспроможність розробки: на даний момент чимала кількість інформації знаходиться на паперових носіях або в електронних таблицях, що вимагає чималих затрат сил і часу на пошук та аналіз необхідної інформації. Для створення зручного навчального процесу необхідно використовувати сучасні технології.

Актуальність розробки полягає у можливості використання розробки системи в органах державного управління та громадських органах з використанням як ліцензійного так і офіційно безкоштовного програмного забезпечення, так і на підприємствах.

У таблиці 4.3 визначені основні технічні показники аналога і нової інформаційної системи автоматизованого формування розкладу.

Таблиця 4.3 - Основні технічні показники аналога і нової інформаційної системи автоматизованого формування розкладу

Функції	Аналог	Нова розробка
Інформаційна система	Відсутня	Наявна
База даних	Відсутня	Наявна
Архітектура системи	Відсутня	Наявна
Інтерфейс користувача	Відсутня	Наявна
Веб-ресурс	Відсутня	Відсутня
Реалізація електронного розкладу	Наявна	Наявна

Успішна реалізація вказаної інформаційної системи вимагає розробку регламентних документів та отримання значної кількості дозвільних документів на створення і впровадження нового продукту у систему освіти. Вказана інформаційна система може бути реалізована на сайті вищого навчального закладу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько-технологічної роботи

Щоб спрогнозувати витрати на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи потрібно виконати наступні задачі:

- *1-й етап*: розрахувати витрати на виконавців даного розділу роботи.
- *2-й етап*: визначити загальні витрати на виконання даної роботи.
- *3-й етап*: спрогнозувати загальних витрат на реалізацію та впровадження результатів даної роботи.

Розрахунок витрат на розробку методики дослідження здійснюється за наступними статтями калькуляції:

- основна заробітна плата виконавців;
- додаткова заробітна плата виконавців;
- нарахування на заробітну плату виконавців;
- амортизація персонального комп'ютера;
- витрати на матеріали, що були задіяні під час розробки інформаційної системи;
- витрати на електроенергію;
- інші витрати.

У розробці даного програмного продукту брав участь один спеціаліст та його науковий керівник.

Основна заробітна плата для спеціаліста визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t \ [грн], \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника, наукового керівника, грн.;

$Т_р$ – кількість робочих днів в місяці; $Т_р$ = 22 дні;

t – число днів роботи: розробника – 3 місяці по 22 дні (t = 66 днів); наукового керівника – 3 дні (t = 3 дні).

Розрахунки основної заробітної плати розробника та наукового керівника зводимо до таблиці 4.4.

Таблиця 4.4 – Розрахунки основної заробітної плати

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Розробник	7500	340,91	22	22500
Науковий керівник	9000	500	3	1500
Всього				24000

Додаткова заробітна плата $З_д$ розробника становить 10% від основної заробітної плати:

$$З_д = 0,10 \cdot 24000 = 2400 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{зп}$ розробника розраховується за наступною формулою:

$$H_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (4.2)$$

де $З_о$ – основна заробітна плата розробника;

$З_д$ – додаткова заробітна плата розробника;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування становить 22%.

$$Н_{зп} = (24000 + 2400) \cdot 0,22 = 5808 \text{ (грн)}.$$

Амортизація обладнання, комп’ютерів та приміщень, які були використані для виконання даного етапу роботи розраховуємо за формулою:

$$А=\frac{Ц \cdot Т}{12 \cdot Т_В} \text{ [грн]}, \quad (4.3)$$

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

Т – фактична тривалість використання, міс;

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

Розробка вказаної інформаційної системи проводилася протягом 3 місяців.

Відповідні розрахунки були зведені до таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Норма амортизації, %	Термін використання, міс	Величина амортизаційних відрахувань, грн.
Офісне приміщення	15000	10	3	375
Ноутбук	7000	10	3	175
Всього				550

Витрати на канцтовари - 200 грн.

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на енергію визначаємо за наступною формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (4.4)$$

де В – вартість 1кВт електроенергії;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи комп’ютера при створенні програмного продукту, годин;

$К_п$ – коефіцієнт використання потужності .

$$В_е = 2.00 \cdot 0,3 \cdot 660 \cdot 0,3 = 118,80 \text{ (грн)}.$$

Інші витрати $В_{ін}$ складаються з витрат на управління організацією, оплати службових відряджень, витрат на утримання, ремонт та експлуатацію основних засобів, витрат на опалення, освітлення, водопостачання, охорону праці тощо. Інші витрати будуть становити 100% від суми основної заробітної плати розробника:

$$В_{ін} = 24000 \cdot 1 = 24000 \text{ (грн)}.$$

Отже, сума усіх вказаних вище витрат (В) на реалізацію даного етапу роботи становить:

$$В = З_о + З_д + Н_{зп} + А + В_{мат} + В_е + В_{ін} \text{ [грн]},$$

$$В = 24000+2400+5808+550+200+118,80+24000=57076,8 \text{ (грн)}.$$

2-й етап. Розрахунок загальних витрат на виконання даної роботи. Загальну вартість всієї наукової роботи ($В_{заг}$) можна знайти за наступною формулою:

$$В_{заг} = \frac{В}{\alpha} \text{ [грн]}, \quad (4.5)$$

де α – частка витрат, які здійснюються самим виконавцем даного етапу роботи, у відносних одиницях.

Так, як робота здійснюється однією людиною, то частка витрат рівна 1.

$$В_{заг} = 57076,8 \text{ (грн)}.$$

3-й етап. Прогнозування загальних витрат на реалізацію та впровадження результатів виконаної роботи. Вказане прогнозування загальних витрат здійснюється за наступною формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \text{ [грн]}, \quad (4.6)$$

де β – коефіцієнт, який характеризує стадію реалізації даної роботи. Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;
- на стадії технічного проектування, то $\beta \approx 0{,}2$;
- на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;
- на стадії розробки технологій, то $\beta \approx 0{,}4$;
- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;
- на стадії розробки промислового зразка, $\beta \approx 0{,}7$;
- на стадії впровадження, то $\beta \approx 0{,}9$.

Отже, для отримання показника загальних витрат підставимо дані в формулу:

$$ЗВ = \frac{57076{,}8}{0{,}9} = 63418{,}67 \text{ (грн)}.$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

На виконання усіх необхідних робіт необхідно затратити 66 робочих днів. Дана розробка вважається економічно вигідною, тому термін її окупності становитиме один рік.

В умовах ринку єдиним позитивним результатом для підприємства вважається збільшення чистого прибутку після впровадження вказаної розробки. Зростання чистого прибутку означає для організації надходження додаткових коштів, які можуть бути використані для покращення фінансових результатів діяльності. Для даного засобу його можна оцінити у сьогоднішній вартості грошей.

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, від яких очікують отримання позитивних результатів від впровадження розробки, обчислюється за наступною формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}) \ [грн], \quad (4.7)$$

де $^{\Delta Ц_o}$ – покращення головного оціночного показника внаслідок впровадження результатів розробки у поточному році;

N – основний кількісний показник, що визначає діяльність підприємства у поточному році до впровадження результатів розробки;

ΔN – покращення головного кількісного показника діяльності підприємства внаслідок впровадження результатів розробки;

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

n – кількість років, протягом яких очікується отримання позитивних результатів внаслідок впровадження розробки;

λ – коефіцієнт, що враховує сплату податку на додану вартість. З 2014 року коефіцієнт λ = 0,8547;

ρ – коефіцієнт, що враховує рентабельність продукції. Знаходиться у межах від 0,2 до 0,3;

υ – ставка податку на прибуток. У 2019 року становить 18%.

Допустимо, що під час впровадження результатів наукової розробки стає кращою якість певного продукту, що дозволяє збільшити ціну його реалізації на 2000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 5 шт., протягом другого року – ще на 10 шт., протягом третього року – ще 15 шт.

Реалізація продукції до впровадження результатів наукової розробки становила 2 шт., а її ціна – 12000 грн.

Визначимо можливе підвищення чистого прибутку після впровадження результатів наукової розробки у кожному наступному році після базового.

Таким чином, підвищення чистого прибутку $\Delta П_i$ протягом першого року становитиме:

$$\Delta П_1=(2000 \cdot 2 + (12000 + 2000) \cdot 5) \cdot 0{,}8547 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 15558{,}96 \text{ (грн)}.$$

Протягом другого року:

$$\Delta П_2=(2000 \cdot 2 + (12000 + 2000) \cdot (5 + 10)) \cdot 0{,}8547 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 44994{,}83 \text{ (грн)}.$$

Протягом третього року:

$$\Delta П_3=(2000 \cdot 2 + (12000 + 2000) \cdot (5 + 10 + 15)) \cdot 0{,}8547 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 89148{,}63 \text{ (грн)}.$$

Таким чином, вказані вище обчислення показують, що комерційний ефект від впровадження розробки виражається у значному підвищенні чистого прибутку організації.

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Основними Комерційний ефект від можливого впровадження розробки ще не означає, що вона реально буде впроваджена. Якщо збільшення можливого прибутку від впровадження результатів наукової розробки є вигідним для організації, то це ще не означає, що інвестор погодиться фінансувати дану розробку, не виставляючи жодних своїх вимог.

Головними показниками для визначення доцільності фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та період їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає виконання наступних задач:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю можна вважати прогнозовану величину загальних витрат $ЗВ$ на реалізацію та впровадження результатів НДДКР, яка була розрахована раніше. Тобто, ЗВ = PV = 63418,67 грн.

2-й крок. Розрахунок очікуваного збільшення прибутку $\Delta П_i$, який отримає організація внаслідок впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку було визначене у попередньому розділі і становить: $\Delta П_1$=15558,96 грн; $\Delta П_2$=44994,83 грн; $\Delta П_3$=89148,63 грн

3-й крок. Для спрощення подальших розрахунків необхідно побудувати вісь часу, яка буде містити дані про всі платежі, а саме про інвестиції та прибутки, що будуть отримані під час виконання науково-дослідної роботи та впровадження її результатів.

Рисунок, який відображає рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рисунку 4.1.

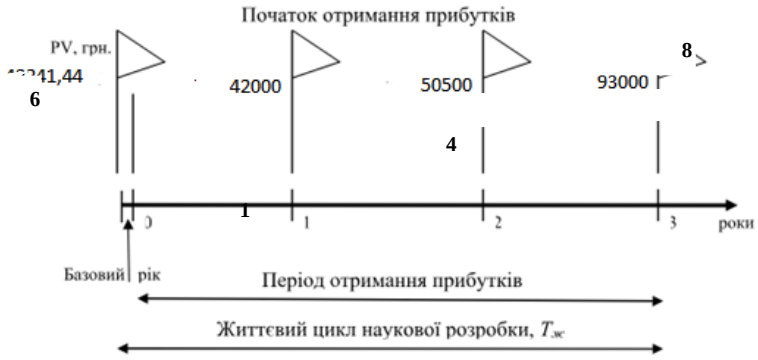


Рисунок 4.1 – Вісь часу з зазначенням платежів, що мають місце в процесі розробки та впровадження результатів НДДКР

4-й крок. Розрахунок абсолютної ефективності вкладених інвестицій $Е_{абс}$ за відповідною формулою:

$$Е_{абс} = (ПП – PV), \quad (4.8)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає організація від впровадження результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.9)$$

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

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

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

$$ПП = \frac{15558,96}{(1+0,1)^1} + \frac{44994,83}{(1+0,1)^2} + \frac{89148,63}{(1+0,1)^3} = 107553,63 \text{ (грн)}.$$

$$Е_{абс} = 107553,63 - 63418,67 = 44134,96 \text{ (грн)}.$$

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки інформаційної системи та її впровадження принесе прибуток, тобто є

доцільним, але це ще не означає те, що інвестор буде зацікавлений у вкладання власних коштів у реалізацію даної наукової розробки.

5-й крок. Розрахунок відносної (щорічної) ефективності вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.10)$$

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{44134,96}{63418,67}} - 1 = 0,193 \text{ або } 19.3\%$$

Порівняємо $E_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

Спрогнозуємо величину $\tau_{мін.}$ У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \quad (4.11)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,136;

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = 0,05.

$$\tau = 0,136 + 0,05 = 0,141$$

Оскільки $E_в$ = 19.3% > $\tau_{мін}$ = 14.1%, то у інвестора може бути потенційна зацікавленість у фінансуванні даної наукової розробки..

6-й крок. Обчислення періоду окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [року].} \quad (4.12)$$

$$T_{ОК} = \frac{1}{0{,}193} = 5{,}18 \text{ (року).}$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок}$ <= 5 років), то фінансування даної нової розробки в принципі є доцільним, але необхідно провести додаткові розрахунки.

4.5 Висновки

В даному розділі було виконано оцінювання комерційного потенціалу розробки .

Проведено технологічний аудит з залученням двох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки є високим.

Аналіз комерційного потенціалу розробки показав, що інформаційна система автоматизованого формування розкладу за своїми характеристиками не має аналогів і є перспективною розробкою. Вказана система має кращі функціональні показники, підтверджуючи конкурентоспроможність товару на ринку. Існуючі переваги нової розробки дозволять швидко її поширити та популяризувати.

Згідно із розрахунками всіх показників витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку становлять 63418,67 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 44134,96 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Щорічна ефективність вкладених в наукову розробку інвестицій складає 19.1 %, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 14.1%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить приблизно 5 років, що свідчить про доцільність фінансування нової розробки, але необхідність проведення додаткових розрахунків.

ВИСНОВКИ

В ході виконання магістерської роботи було досліджено предметну область ущільнення даних та реалізовано алгоритм контекстного моделювання для архівування даних. А також було визначено проблематику та актуальність даної теми. Аналіз стану питання показав, що архіватори є актуальними у наш час, так як ідеального архіватора не існує, так як кожен має якісь недоліки. На основі порівняльного аналізу стану сучасних архіваторів було складено список характеристик, якими повинен володіти розроблюваний програмний продукт.

Було розглянуто різні алгоритми ущільнення інформації для вирішення поставленої задач і вибрано остаточний алгоритм, який буде використовуватися для архівування файлів, а саме одна з модифікацій алгоритму PPM (Prediction by Partial Matching), яка застосовується у серії алгоритмів PAQ, найсучаснішим серед яких є ZPAQ. Даний алгоритм був описаний, а також була реалізована його схема, яка відображає його основні елементи.

Було вибрано середовище програмування MS VS 2015 та мова C++ для реалізації цього завдання. Після чого було написано готовий додаток з графічним інтерфейсом, який дозволяє користувачу проводити різноманітні дії над файлами та папками. Також було розглянуто одновіконні та багато віконні інтерфейси. В результаті, для розробки програмного додатку було обрано одновіконний інтерфейс, тому що користувач буде взаємодіяти з одним головним вікном, яке свою чергу буде викликати модальні вікна, що не будуть заважати його роботі з основним вікном. Описаний інтерфейс головного вікна складається із панелі керування вікном, панелі меню, панелі інструментів, робочої області та рядку стану. Всі елементи детально описані. В результаті отримано інтерфейс, що відповідає сучасним вимогам розробки графічних інтерфейсів і буде зручним у користуванні.

Було перевірено розроблену програму на реальних даних та отримані хороші результати. Даний програмний продукт зарекомендував себе, як такий який відповідає сучасним та функціональним вимогам, які були встановлені у першому розділі.

Судячи з результатів економічної оцінки дана реалізація має сильний потенціал до розширення, оскільки ущільнення інформації не є прив’язаним до конкретної області знань або департаменту, та може використовуватися як організацією, так і звичайним користувачем. Економічний аналіз також показав, що дана розробка цілком може зацікавити інвесторів, адже є досить таки приубутковою.

Тому виконання даного завдання можна вважати успішним, а отриманий розклад лише трохи, але не принципово, відрізнявся від того, що був сформований людиною за довгий час. Швидкість і зручність алгоритму та врахування вимог дуального підходу в освіті є його суттєвою перевагою.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Майданюк В. П. Кодування та захист інформації. Навчальний посібник / В.П. Майданюк - Вінниця: ВНТУ, 2009. - 164 с.

2. Використання алгоритму контекстного моделювання для розробки програми для ущільнення даних без втрат [Електронний ресурс] – Режим доступу до ресурсу: https://card-file.onaft.edu.ua/handle/123456789/10174.

3. Наукоємні технології [Електронний ресурс] – Режим доступу: http://jrnl.nau.edu.ua/index.php/SBT/article/view/5226/5737.

4. Каплун В. А. Захист операційних систем. Навчальний посібник / В. А.Каплун, В. П Майданюк - Вінниця: ВНТУ, 2007. – 185 с.

5. 7-Zip [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/7-Zip.

6. WinRAR – коротко о главном [Електронний ресурс] – Режим доступу: https://www.win-rar.com/features.html?&L=4.

7. Програми архівування файлів [Електронний ресурс] – Режим доступу: https://stboinf.wordpress.com/2010/02/16/програми-архівування-файлів/.

8. Free RAR TAR ZIP files utility [Електронний ресурс] – Режим доступу: http://www.peazip.org/.

9. Рейтинг популярных архиваторов WinZip, 7Zip, WinRar, Hamster, PeaZip [Електронний ресурс] – Режим доступу: https://it-critic.ru/rejting-populyarnyh-arhivatorov-winzip-7zip-winrar-hamster-peazip/#arhivator-winrar-x64-5-20.

10. The ZPAQ Open Standard Format for Highly Compressed Data - Level 1 Candidate [Електронний ресурс] – Режим доступу: http://mattmahoney.net/dc/zpaq.pdf.

11. C++ [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/C%2B%2B.

12. C Sharp [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp.

13. Java [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Java.

14. Microsoft Visual Studio [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

15. Qt Creator [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Qt_Creator.

16. Eclipse [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Eclipse.

17. Тестування програмного продукту [Електронний ресурс] – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm.

ДОДАТКИ

Додаток А

Технічне завдання

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики

ЗАТВЕРДЖУЮ
Завідувач кафедри САКМІГ
_____ д. т. н., проф. В.Б. Мокін
(підпис)
"__" _____ 2019

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу

ІНФОРМАЦІЙНА СИСТЕМА АРХІВУВАННЯ ДАНИХ З ВИКОРИСТАННЯМ КОНТЕКСТНОГО МОДЕЛЮВАННЯ

08-53.МКР.011.01.000 ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доц.
_____Є.М. Крижановський
(підпис)
"__" _____ 2019 р.
Розробив студент гр. ІСТ-18м
_____О.П. Федюк
(підпис)
"__" _____ 2019 р.

Вінниця 2019

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 201_ р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САКМІГ від «__» _____ 201_ р.

2. Джерела розробки

1) Майданюк В. П. Кодування та захист інформації. Навчальний посібник. - Вінниця: ВНТУ, 2009. - 164 с.

2) The ZPAQ Open Standard Format for Highly Compressed Data - Level 1 Candidate [Електронний ресурс] – Режим доступу: http://mattmahoney.net/dc/zpaq.pdf.

3. Мета і призначення роботи

Розробка інформаційної системи архівування даних з використанням контекстного моделювання.

4. Вихідні дані для проведення робіт

1) мова програмування – C++

2) наявні алгоритми ущільнення інформації

3) дані про сучасні програмні додатки для ущільнення файлів

4) вимоги користувачів до програмних додатків для ущільнення файлів.

5. Методи дослідження

1) Порівняльний аналіз.

2) Експеримент.

6. Етапи роботи і терміни їх виконання

1) Огляд існуючих аналогів..……..__. – __

2) Огляд існуючих алгоритмів ущільнення даних ……..…….__. – __

3) Розробка системи……..……..__. – __

4) Тестування системи ……..……..__. – __

7. Очікувані результати та порядок реалізації

Розробка інформаційної системи, яка відповідає функціональним та нефункціональним вимогам користувачів щодо архівування даних та відповідного програмного продукту.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист…….__.__.201_ р.

Початок розробки «___» _____ 201_ р.

Граничні терміни виконання МКР «___» _____201_ р.

Розробив студент групи ІСТ-18м _____ Федюк О. П.

Додаток Б

Інструкція користувача

Програма тестувалась та працює коректно у операційних системах Windows XP, Windows 7, Windows 10.

Так як програма не вимагає інсталяції, то для роботи з програмою достатньо запустити файл Sazer.exe.

Після цього користувач може запускати і працювати з програмним продуктом.

Структура інтерфейсу головного вікна (рис. 3.31):

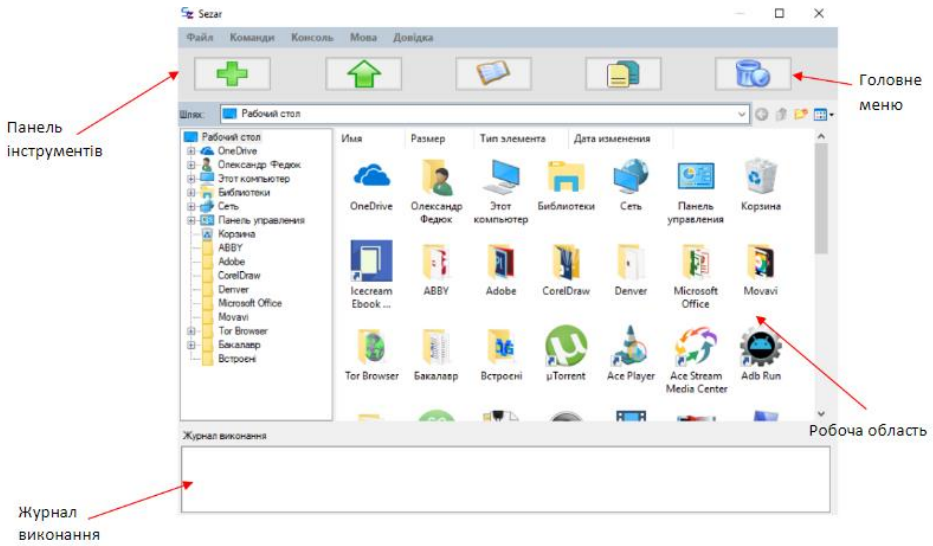


Рисунок 3.31 – Структура головного вікна

Головне меню знаходиться зверху та має наступну структуру:

1. Файл:
    а. Відкрити.
    б. Вихід.
2. Команди:
    а. Додати. Створити архів у вказаній папці.
    б. Видобути. Видобути файли з архіву у вказану папку.
    в. Переглянути. Переглянути файли.
    г. Копіювати. Копіювати файли.

д. Видалити. Видалити.

3. Мова:
    а. Українська. Змінити мову на українську.
    б. Англійська. Змінити мову на англійську.
4. Довідка:
    а. Інструкція користувача. Викликати довідку або інструкцію користувача.
    б. Про програму. Інформація про програму.

Панель інструментів має наступний вигляд:

 Створити архів або додати до нього файли.

 Видобути файли з архіву розпакувати, архів.

 Переглянути.

 Копіювати.

 Видалити.

Процес архівації та розархівування детально описується в журналі виконання, що знаходиться унизу вікна. У разі, якщо виникне помилка, у журналі вона з’явиться та попередить Вас.

Додаток В

Лістинг програми

```
// Initialize predictions
        for (int i = 0; i<256; ++i) h[i] = p[i] = 0;

        // Initialize components
        for (int i = 0; i<256; ++i)  // clear old model
            comp[i].init();
        int n = z.header[6]; // hsize[0..1] hh hm ph pm n (comp)[n] END 0[128] (hcomp) END
        const U8* cp = &z.header[7];  // start of component list
        for (int i = 0; i<n; ++i) {
            assert(cp<&z.header[z.cend]);
            assert(cp>&z.header[0] && cp<&z.header[z.header.isize() - 8]);
            Component& cr = comp[i];
            switch (cp[0]) {
            case CONS:  // c
                p[i] = (cp[1] - 128) * 4;
                break;
            case CM: // sizebits limit
                if (cp[1]>32) error("max size for CM is 32");
                cr.cm.resize(1, cp[1]);  // packed CM (22 bits) + CMCOUNT (10 bits)
                cr.limit = cp[2] * 4;
                for (size_t j = 0; j<cr.cm.size(); ++j)
                    cr.cm[j] = 0x80000000;
                break;
            case ICM: // sizebits
                if (cp[1]>26) error("max size for ICM is 26");
                cr.limit = 1023;
                cr.cm.resize(256);
                cr.ht.resize(64, cp[1]);
                for (size_t j = 0; j<cr.cm.size(); ++j)
                    cr.cm[j] = st.cminit(j);
                break;
            case MATCH:  // sizebits
                if (cp[1]>32 || cp[2]>32) error("max size for MATCH is 32 32");
                cr.cm.resize(1, cp[1]);  // index
                cr.ht.resize(1, cp[2]);  // buf
                cr.ht(0) = 1;
                break;
            case AVG: // j k wt
                if (cp[1] >= i) error("AVG j >= i");
                if (cp[2] >= i) error("AVG k >= i");
                break;
            case MIX2:  // sizebits j k rate mask
                if (cp[1]>32) error("max size for MIX2 is 32");
                if (cp[3] >= i) error("MIX2 k >= i");
                if (cp[2] >= i) error("MIX2 j >= i");
                cr.c = (size_t(1) << cp[1]); // size (number of contexts)
                cr.a16.resize(1, cp[1]);  // wt[size][m]
                for (size_t j = 0; j<cr.a16.size(); ++j)
                    cr.a16[j] = 32768;
                break;
            case MIX: {  // sizebits j m rate mask
                if (cp[1]>32) error("max size for MIX is 32");
                if (cp[2] >= i) error("MIX j >= i");
```

```
                        if (cp[3]<1 || cp[3]>i - cp[2]) error("MIX m not in 1..i-j");
                        int m = cp[3];  // number of inputs
                        assert(m >= 1);
                        cr.c = (size_t(1) << cp[1]); // size (number of contexts)
                        cr.cm.resize(m, cp[1]);  // wt[size][m]
                        for (size_t j = 0; j<cr.cm.size(); ++j)
                                cr.cm[j] = 65536 / m;
                        break;
                }
                case ISSE:  // sizebits j
                        if (cp[1]>32) error("max size for ISSE is 32");
                        if (cp[2] >= i) error("ISSE j >= i");
                        cr.ht.resize(64, cp[1]);
                        cr.cm.resize(512);
                        for (int j = 0; j<256; ++j) {
                                cr.cm[j * 2] = 1 << 15;
                                cr.cm[j * 2 + 1] = clamp512k(stretch(st.cminit(j) >> 8) * 1024);
                        }
                        break;
                case SSE: // sizebits j start limit
                        if (cp[1]>32) error("max size for SSE is 32");
                        if (cp[2] >= i) error("SSE j >= i");
                        if (cp[3]>cp[4] * 4) error("SSE start > limit*4");
                        cr.cm.resize(32, cp[1]);
                        cr.limit = cp[4] * 4;
                        for (size_t j = 0; j<cr.cm.size(); ++j)
                                cr.cm[j] = squash((j & 31) * 64 - 992) << 17 | cp[3];
                        break;
                default: error("unknown component type");
                }
                assert(compsize[*cp]>0);
                cp += compsize[*cp];
                assert(cp >= &z.header[7] && cp<&z.header[z.cend]);
        }
}

// Return next bit prediction using interpreted COMP code
int Predictor::predict0() {
        assert(initTables);
        assert(c8 >= 1 && c8 <= 255);

        // Predict next bit
        int n = z.header[6];
        assert(n>0 && n <= 255);
        const U8* cp = &z.header[7];
        assert(cp[-1] == n);
        for (int i = 0; i<n; ++i) {
                assert(cp>&z.header[0] && cp<&z.header[z.header.isize() - 8]);
                Component& cr = comp[i];
                switch (cp[0]) {
                case CONS:  // c
                        break;
                case CM:  // sizebits limit
                        cr.cxt = h[i] ^ hmap4;
                        p[i] = stretch(cr.cm(cr.cxt) >> 17);
                        break;
                case ICM: // sizebits
                        assert((hmap4 & 15)>0);
```

```
            if (c8 == 1 || (c8 & 0xf0) == 16) cr.c = find(cr.ht, cp[1] + 2, h[i] + 16 * c8);
            cr.cxt = cr.ht[cr.c + (hmap4 & 15)];
            p[i] = stretch(cr.cm(cr.cxt) >> 8);
            break;
case MATCH: // sizebits bufbits: a=len, b=offset, c=bit, cxt=bitpos,
                                    //              ht=buf, limit=pos
            assert(cr.cm.size() == (size_t(1) << cp[1]));
            assert(cr.ht.size() == (size_t(1) << cp[2]));
            assert(cr.a <= 255);
            assert(cr.c == 0 || cr.c == 1);
            assert(cr.cxt<8);
            assert(cr.limit<cr.ht.size());
            if (cr.a == 0) p[i] = 0;
            else {
                        cr.c = (cr.ht(cr.limit - cr.b) >> (7 - cr.cxt)) & 1; // predicted bit
                        p[i] = stretch(dt2k[cr.a] * (cr.c*-2 + 1) & 32767);
            }
            break;
case AVG: // j k wt
            p[i] = (p[cp[1]] * cp[3] + p[cp[2]] * (256 - cp[3])) >> 8;
            break;
case MIX2: { // sizebits j k rate mask
                                    // c=size cm=wt[size] cxt=input
            cr.cxt = ((h[i] + (c8&cp[5]))&(cr.c - 1));
            assert(cr.cxt<cr.a16.size());
            int w = cr.a16[cr.cxt];
            assert(w >= 0 && w<65536);
            p[i] = (w*p[cp[2]] + (65536 - w)*p[cp[3]]) >> 16;
            assert(p[i] >= -2048 && p[i]<2048);
}
                        break;
case MIX: {  // sizebits j m rate mask
                                    // c=size cm=wt[size][m] cxt=index of wt in cm
            int m = cp[3];
            assert(m >= 1 && m <= i);
            cr.cxt = h[i] + (c8&cp[5]);
            cr.cxt = (cr.cxt&(cr.c - 1))*m; // pointer to row of weights
            assert(cr.cxt <= cr.cm.size() - m);
            int* wt = (int*)&cr.cm[cr.cxt];
            p[i] = 0;
            for (int j = 0; j<m; ++j)
                        p[i] += (wt[j] >> 8)*p[cp[2] + j];
            p[i] = clamp2k(p[i] >> 8);
}
                        break;
case ISSE: { // sizebits j -- c=hi, cxt=bh
            assert((hmap4 & 15)>0);
            if (c8 == 1 || (c8 & 0xf0) == 16)
                        cr.c = find(cr.ht, cp[1] + 2, h[i] + 16 * c8);
            cr.cxt = cr.ht[cr.c + (hmap4 & 15)];  // bit history
            int *wt = (int*)&cr.cm[cr.cxt * 2];
            p[i] = clamp2k((wt[0] * p[cp[2]] + wt[1] * 64) >> 16);
}
                        break;
case SSE: { // sizebits j start limit
            cr.cxt = (h[i] + c8) * 32;
            int pq = p[cp[2]] + 992;
            if (pq<0) pq = 0;
```

```
                        if (pq>1983) pq = 1983;
                        int wt = pq & 63;
                        pq >>= 6;
                        assert(pq >= 0 && pq <= 30);
                        cr.cxt += pq;
                        p[i] = stretch(((cr.cm(cr.cxt) >> 10)*(64 - wt) + (cr.cm(cr.cxt + 1) >> 10)*wt) >> 13);
                        cr.cxt += wt >> 5;
                }
                                break;
                default:
                        error("component predict not implemented");
                }
                cp += compsize[cp[0]];
                assert(cp<&z.header[z.cend]);
                assert(p[i] >= -2048 && p[i]<2048);
        }
        assert(cp[0] == NONE);
        return squash(p[n - 1]);
}

// Update model with decoded bit y (0...1)
void Predictor::update0(int y) {
        assert(initTables);
        assert(y == 0 || y == 1);
        assert(c8 >= 1 && c8 <= 255);
        assert(hmap4 >= 1 && hmap4 <= 511);

        // Update components
        const U8* cp = &z.header[7];
        int n = z.header[6];
        assert(n >= 1 && n <= 255);
        assert(cp[-1] == n);
        for (int i = 0; i<n; ++i) {
                Component& cr = comp[i];
                switch (cp[0]) {
                case CONS:  // c
                        break;
                case CM:  // sizebits limit
                        train(cr, y);
                        break;
                case ICM: { // sizebits: cxt=ht[b]=bh, ht[c][0..15]=bh row, cxt=bh
                        cr.ht[cr.c + (hmap4 & 15)] = st.next(cr.ht[cr.c + (hmap4 & 15)], y);
                        U32& pn = cr.cm(cr.cxt);
                        pn += int(y * 32767 - (pn >> 8)) >> 2;
                }
                                break;
                case MATCH: // sizebits bufbits:
                                        //  a=len, b=offset, c=bit, cm=index, cxt=bitpos
                                        //  ht=buf, limit=pos
                {
                        assert(cr.a <= 255);
                        assert(cr.c == 0 || cr.c == 1);
                        assert(cr.cxt<8);
                        assert(cr.cm.size() == (size_t(1) << cp[1]));
                        assert(cr.ht.size() == (size_t(1) << cp[2]));
                        assert(cr.limit<cr.ht.size());
                        if (int(cr.c) != y) cr.a = 0;  // mismatch?
                        cr.ht(cr.limit) += cr.ht(cr.limit) + y;
```

```
            if (++cr.cxt == 8) {
                cr.cxt = 0;
                ++cr.limit;
                cr.limit &= (1 << cp[2]) - 1;
                if (cr.a == 0) {  // look for a match
                    cr.b = cr.limit - cr.cm(h[i]);
                    if (cr.b&(cr.ht.size() - 1))
                        while (cr.a<255
                            && cr.ht(cr.limit - cr.a - 1) == cr.ht(cr.limit - cr.a - cr.b -
1))
                            ++cr.a;
                }
                else cr.a += cr.a<255;
                cr.cm(h[i]) = cr.limit;
            }
        }
        break;
        case AVG:  // j k wt
            break;
        case MIX2: { // sizebits j k rate mask
                        // cm=wt[size], cxt=input
            assert(cr.a16.size() == cr.c);
            assert(cr.cxt<cr.a16.size());
            int err = (y * 32767 - squash(p[i]))*cp[4] >> 5;
            int w = cr.a16[cr.cxt];
            w += (err*(p[cp[2]] - p[cp[3]]) + (1 << 12)) >> 13;
            if (w<0) w = 0;
            if (w>65535) w = 65535;
            cr.a16[cr.cxt] = w;
        }
                    break;
        case MIX: {   // sizebits j m rate mask
                        // cm=wt[size][m], cxt=input
            int m = cp[3];
            assert(m>0 && m <= i);
            assert(cr.cm.size() == m*cr.c);
            assert(cr.cxt + m <= cr.cm.size());
            int err = (y * 32767 - squash(p[i]))*cp[4] >> 4;
            int* wt = (int*)&cr.cm[cr.cxt];
            for (int j = 0; j<m; ++j)
                wt[j] = clamp512k(wt[j] + ((err*p[cp[2] + j] + (1 << 12)) >> 13));
        }
                    break;
        case ISSE: { // sizebits j  -- c=hi, cxt=bh
            assert(cr.cxt == cr.ht[cr.c + (hmap4 & 15)]);
            int err = y * 32767 - squash(p[i]);
            int *wt = (int*)&cr.cm[cr.cxt * 2];
            wt[0] = clamp512k(wt[0] + ((err*p[cp[2]] + (1 << 12)) >> 13));
            wt[1] = clamp512k(wt[1] + ((err + 16) >> 5));
            cr.ht[cr.c + (hmap4 & 15)] = st.next(cr.cxt, y);
        }
                    break;
        case SSE:  // sizebits j start limit
            train(cr, y);
            break;
        default:
            assert(0);
        }
```

```
                cp += compsize[cp[0]];
                assert(cp >= &z.header[7] && cp<&z.header[z.cend]
                        && cp<&z.header[z.header.isize() - 8]);
        }
        assert(cp[0] == NONE);

        // Save bit y in c8, hmap4
        c8 += c8 + y;
        if (c8 >= 256) {
                z.run(c8 - 256);
                hmap4 = 1;
                c8 = 1;
                for (int i = 0; i<n; ++i) h[i] = z.H(i);
        }
        else if (c8 >= 16 && c8<32)
                hmap4 = (hmap4 & 0xf) << 5 | y << 4 | 1;
        else
                hmap4 = (hmap4 & 0x1f0) | (((hmap4 & 0xf) * 2 + y) & 0xf);
}}
```

Додаток Г

Графічна частина

Структура програми

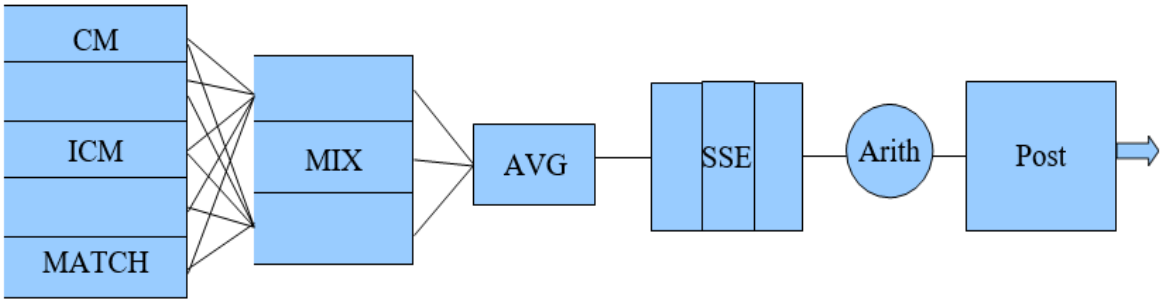
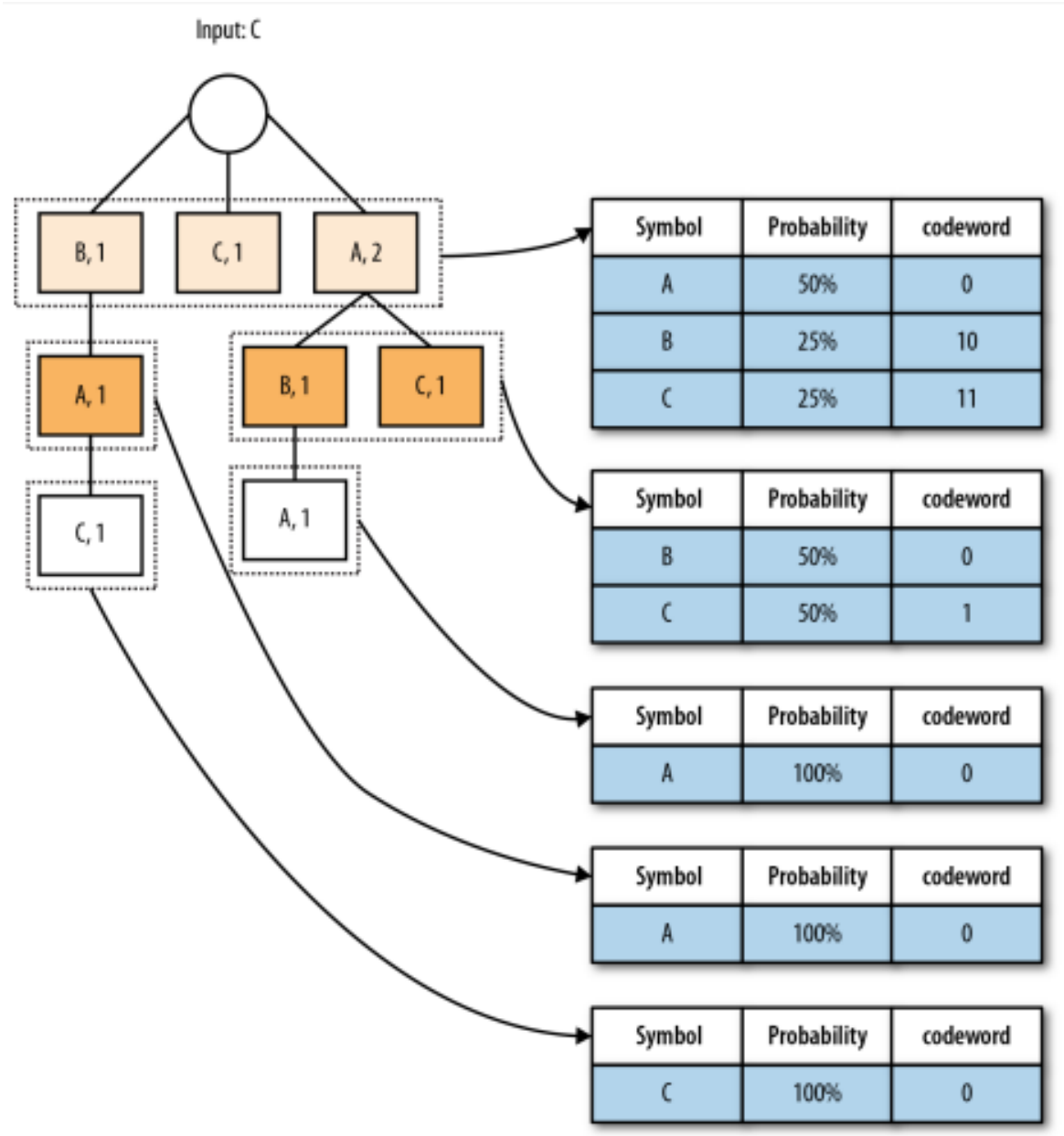
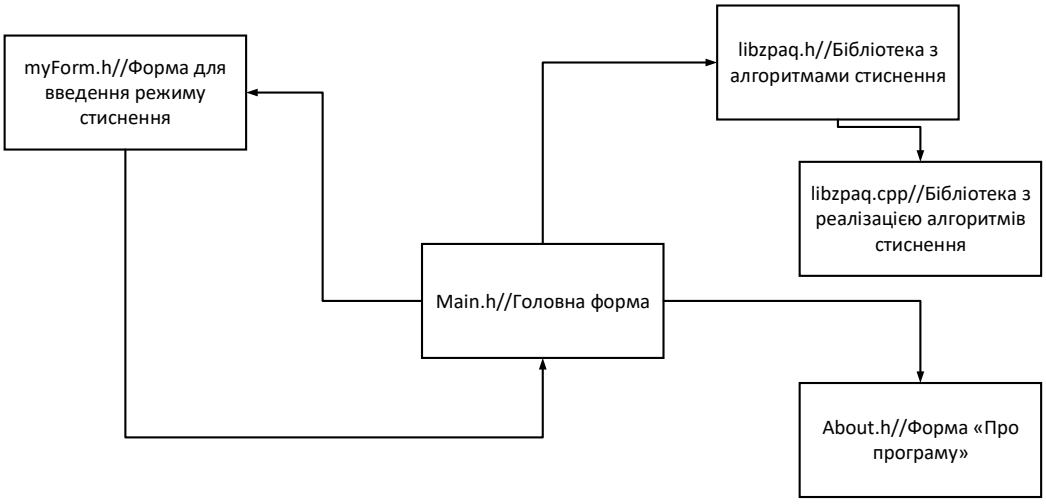


Схема алгоритму ущільнення



Загальний взаємозв’язок між модулями та формами



Порівняльні характеристики сучасних аналогів

Критерій	7-Zip	WinRAR	WinZip	PeaZip	Sezar
Не вимагає встановлення	-	-	-	+	+
Зрозумілий інтерфейс	+	+	-	-	+
Зручна навігація між папками	-	-	+	+	+
Шифрування даних	+	+	+	-	+
Прискорений метод Drag&Drop (перетягування в активне вікно)	-	-	-	-	+
Безкоштовно	+	-	-	+	+
Наявність перекладу інтерфейсу на українську мову	+	+	+	-	+
Загальна сума	4	3	3	3	7

Порівняльна характеристика функціональних можливостей аналогів

Файли до ущільнення

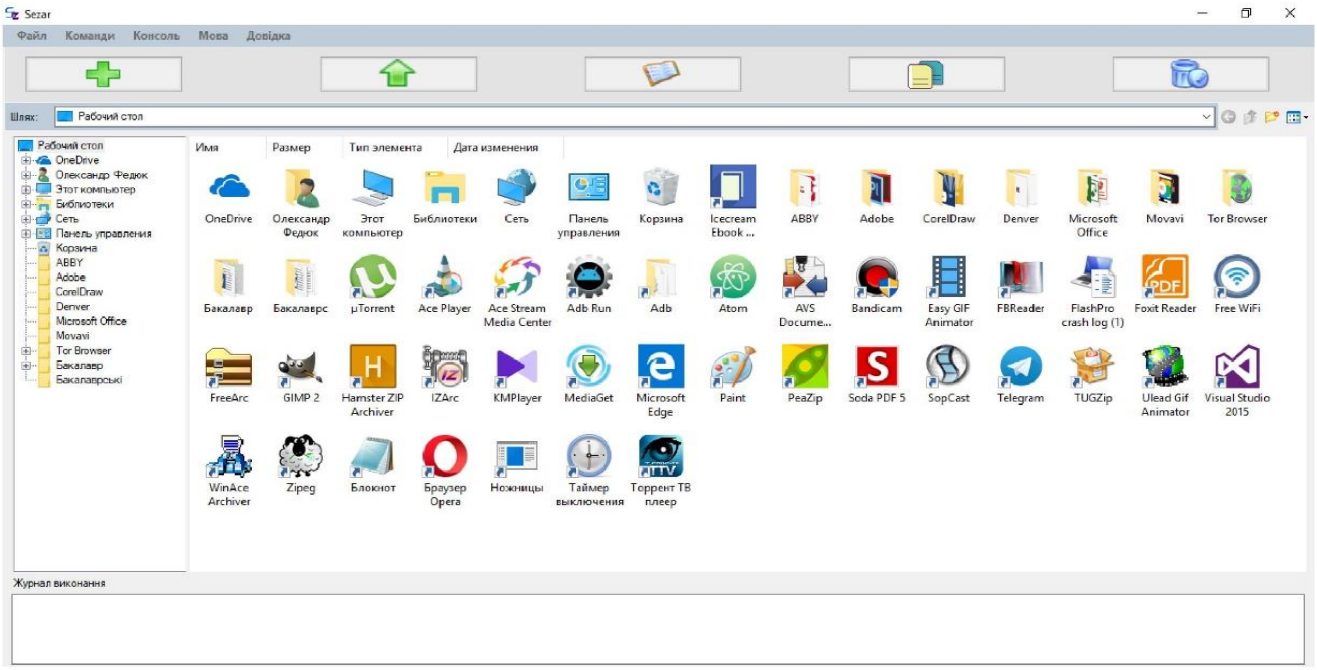
Тип	Опис	Розмір в Мб
Текст	Текстові документи в форматах xls(x), doc(x)	1000
Зображення	Картинки в форматі jpeg	1000
Медіа	Аудіо файли в форматі mp3	1006

Функціональні характеристики

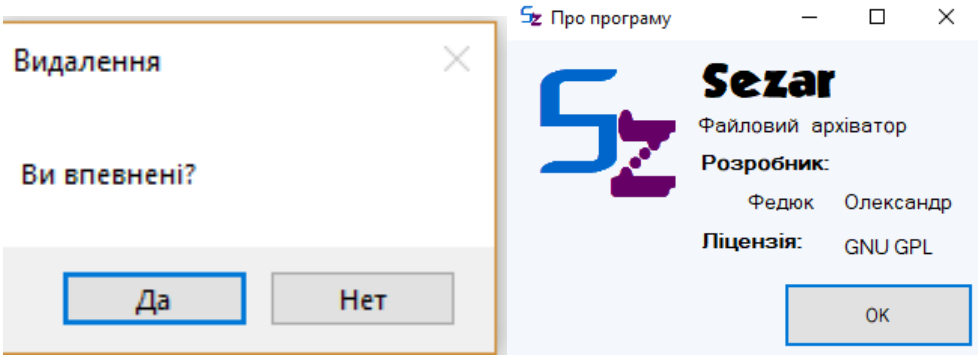
Текст				
Критерій	7-Zip	WinRAR	WinZip	PeaZip
Час (с)	97,66666667	42,33333333	112,00000000	149,66666667
Степінь стиснення	4,447023017	3,086418988	3,239885491	18,06592608

Зображення				
Критерій	7-Zip	WinRAR	WinZip	PeaZip
Час (с)	3,33333333	28,33333333	137,33333333	102,33333
Степінь стиснення	24,430829643	1,010161211	1,304574419	25,66054112

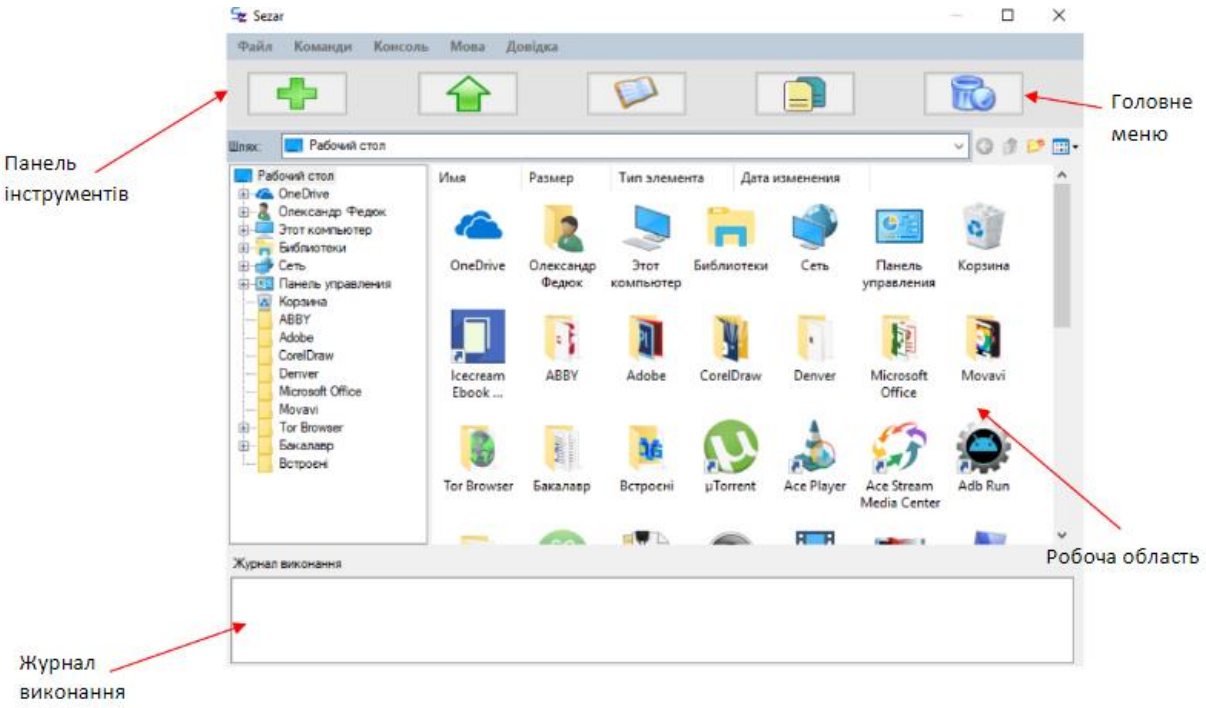
Інтерфейс головного вікна



Додаткові вікна інформаційної системи



Інтерфейс програми та його структурні елементи



Додаток Д

Таблиця критеріїв оцінювання розробки

Таблиця Д – Таблиця критеріїв оцінювання розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці Д

6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту