Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Горяний Д.В.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

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

Горяному Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 37-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Горяний Д.В. Дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Метою розробки є дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Об'єктом дослідження є процес електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Предметом дослідження є методи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

Результат роботи – програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

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

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерні науки, електронний документообіг

ABSTRACT

Horianyi D.V. Research and software implementation of the electronic document flow system of the department of cyber security and software. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the electronic document management system of the Department of Cyber Security and Software.

The purpose of the development is the research and software implementation of the electronic document management system of the department of cyber security and software.

The object of the study is the process of electronic document circulation of the department of cyber security and software.

The subject of the study is the methods of electronic document circulation of the department of cyber security and software.

The research methods are based on the methods of the theory of electronic document flow, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the electronic document flow system of the department of cyber security and software.

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

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

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

The program was developed in the Visual C++ environment.

Keywords: computer science, electronic document management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕА	–	електронний архів
ЕДО	–	електронний документообіг
САДД	–	система автоматизації електронного документообігу й діловодства
СЕА	–	системи електронних архівів
СЕД	–	система електронного документообіга
СДЕ	–	системи діловодства електронні
СУЕД	–	система управління електронним документообігом
ТЕ	–	типові елементи
ФФО	–	формат файлових об'єктів
EDMS		Electronic Document Management Systems

ВСТУП

Актуальність теми. Документообіг – рух документів в обліковому процесі з моменту їхнього складання до здачі в архів. Документи, складені в господарських підрозділах підприємства, передаються в бухгалтерію. В останній вони перевіряються за формою й змістом, групуються по однорідних ознаках і служать підставою для записів в облікових регістрах. Після записів документи перекладають у папки й використовують для різних довідок, перевірок і документальних ревізій. Документи, що мають науково-історичну цінність, після закінчення строків, здаються в місцевий архів, а інші знищуються.

Документообіг – кровоносна система будь-якого підприємства. Всі операції, зроблені кожним з підрозділів підприємства, ретельно документуються. Такому поданню піддаються не тільки операції, зроблені за участю зовнішніх підприємств і організацій, але й усе, що відбувається безпосередньо усередині самого підприємства. Не дивлячись на це, до 15% паперових документів губляться й на їхній пошук працівники витрачають до 30% робочого часу. При переході до електронних документів ріст продуктивності співробітників збільшується на 25 – 50 %, а час обробки одного документа скорочується на 80%. Кількість документів за рік виростає у два рази, а електронних документів на 7%.

Організація документообігу й керування їм є важливим завданням інформаційного менеджменту. Тут позначається необхідність забезпечення групової роботи виконавців над документами й надання в їхнє розпорядження інформаційних ресурсів підприємства. Отже, у широкому змісті електронну систему документообігу можна розглядати як електронну систему підтримки виконання.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

– Огляд існуючих систем електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– Дослідження системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– Програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Об'єктом дослідження є процес електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Предметом дослідження є методи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

– Удосконалено метод електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– Розроблено вітчизняний продукт електронного документообігу кафедри кібербезпеки та програмного забезпечення, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Електронний документообіг – модернізований програмний засіб, який допомагає навести порядок в документах. Він підтримує документи на всіх етапах їхнього життєвого циклу, від їх створення в стандартних шаблонах до процесу затвердження та електронного підпису, аж до етапу їх передачі клієнтам.

Завдяки інтеграції з OCR (оптичне розпізнавання символів) ви можете ефективно конвертувати відскановані документи в текст. Ви можете безпечно архівувати їх у сховищі цифрових документів, інтегрованому в систему. Завдяки модульній структурі ви можете легко налаштувати систему відповідно до вимог вашої компанії.

Додаток дозволяє контактувати з клієнтами через платформу для передачі електронних документів. Якщо вони містять кваліфікований підпис, стандартний компонент EDPS (Європейський інспектор із захисту даних) перевірить цей підпис і підтвердить достовірність листування.

EDPS підтримує повний спектр функцій управління кореспонденцією та відправленнями. Він забезпечує легкий доступ до документів із повною функціональністю пошуку (як із повнотекстовими запитами, так і з атрибутами). Інструмент також дозволяє конвертувати відскановані файли в редаговані формати (OCR).

Завдяки використанню чіткого графічного представлення у вигляді діаграм інструмент забезпечує постійне розуміння процесів. Ви постійно в курсі того, на якому етапі перебуває обробка доручення, і в курсі запитів на делегування або відпустку (ми забезпечуємо інтеграцію з HR-системою).

Один інструмент, багато переваг

– Більша ефективність управління завдяки постійному контролю документообігу.

– Вища ефективність роботи за рахунок ефективного контролю процесу

та стандартизації методів роботи.

– Більша безпека і зручність використання.

Система є комплексним засобом підвищення ефективності документообігу. Це скорочує час пошуку документів і забезпечує постійний доступ до архівів. СЕД зміцнює безпечне робоче середовище, зменшуючи ризик помилок і обмежуючи ризик втрати паперових документів. Інструмент також дозволить скоротити витрати на канцелярське приладдя.

1.2 Область застосування

Необхідність впровадження систем електронного документообігу в існуючих економічних умовах не можна поставити під сумнів. Цим обумовлене й кількість систем даного профілю представлених у цей час на ринку і їхня якість. Всі системи не тільки відповідають пропонованим вимогам, але я володіють рядом характеристик, які відрізняють певну систему від інших.

Таким чином, вибір системи електронного документообігу в кожному конкретному випадку може бути здійснений лише з урахуванням умов, у яких передбачається експлуатація даної системи.

При наявності грамотних фахівців в інформаційному відділі підприємства можливий вибір системи, що задовольняє всім запитам не тільки у функціональному варіанті, але й у ціновому.

Системи, що виконують дану функцію, у цей час, широко представлені на ринку інформаційних систем. Кожне підприємство пред'являє свій набір вимог до даного типу систем. Для вибору найбільш підходящого продукту необхідно провести порівняльний аналіз всіх представлених, керуючись вихідними вимогами.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Електронна система управління документами (СЕД) – це програмне забезпечення, яке централізовано зберігає та організовує документацію в одному цифровому сховищі. Типи СЕД включають «власноруч створені», запатентовані «локальні» та хмарні рішення. Мета СЕД полягає в тому, щоб надати структуровані та захищені цифрові можливості зберігання, видимість і контроль для всієї документації, яку створює ваш бізнес, щоб він міг ефективно працювати. СЕД діє як єдине джерело правди, яке полегшує співпрацю та економить вашій організації непотрібні витрати.

Кому потрібна СЕД?

«Документна анархія» все ще втрачає час, енергію та ресурси для багатьох організацій. Труднощі з пошуком та обміном документами все ще вважаються однією з головних подразнень і проблем у повсякденному житті багатьох працівників. Але для деяких компаній це буквально перешкоджає залученню нового бізнесу та ефективному розширенню. Погані процедури видачі документів і контролю можуть стати на заваді відповідності таким стандартам, як ISO 9001 і ISO 13485. У той же час, неадекватна інфраструктура безпеки, яка відображає конфіденційність даних, які ви контролюєте, може призвести до системних порушень із серйозними комерційними та репутаційними наслідками.

Більшість цифрових інструментів для файлів (наприклад, простий Google Drive) можуть принести негайну користь зростаючому бізнесу завдяки більш розумній структурі вашого сховища та доступу до спільних просторів для спільної роботи над документами. Але не всі вони призначені для вирішення

будь-яких бізнес-завдань або «вростають» у ці виклики разом з вами.

Як правильно вибрати СЕД

На що слід звернути увагу, обираючи систему електронного документообігу (СЕД)? Все залежить від рівня контролю, співпраці та гнучкості, яких вам потрібно досягти.

Однак є дві ключові функції СЕД, на які завжди слід звертати увагу:

1. Робочі процеси, які легко налаштувати. Ви можете полегшити співпрацю (навіть віддалену) і переконатися, що вся ваша команда, де б вони не були, працює над правильним документом і правильною версією. Хороша СЕД матиме процеси затвердження та контроль перегляду, щоб оптимізувати робочий процес і забезпечити продуктивність вашої команди .

2. Інтуїтивно зрозумілий контроль версій документа. Усі редакції та чернетки документа слід точно фіксувати та керувати ними. Ви повинні мати можливість використовувати автоматичний контроль версій документів і аудит на різних пристроях. Вся команда повинна мати доступ до актуальної точної інформації, оскільки це забезпечить безпеку вашої документації та цілісність даних.

Вибір між установкою стороннього спеціально розробленого рішення СЕД або підходом «зроби сам» (з використанням програм для обміну файлами, таких як Google Drive, Dropbox або Box) є одним із перших рішень, з якими ви можете зіткнутися.

Платформи для обміну файлами, такі як Google Drive або OneDrive, пропонують гідні, зручні мультиплатформенні можливості співпраці та керування документами. Їх легко налаштувати, і вони можуть бути «низькими» або «безкоштовними» (принаймні, для початку). Однак вони також можуть швидко стати недостатніми для масштабованого бізнесу, який зіткнувся зі складним контролем файлів і вимогами до зберігання.

Це особливо актуально для компаній, які створюють і співпрацюють над збільшенням обсягів технічної документації в різних форматах або накопичують

конфіденційні дані клієнтів.

SharePoint

Для тих, хто має складніші вимоги, SharePoint може здатися хорошим вибором, оскільки він дозволяє визначити та створити рішення, яке виконує саме те, що вам потрібно. Однак створення достатньо просунутого рішення SharePoint, щоб задовольнити формальні потреби керування документами, ймовірно, потребуватиме значного часу розробника та навіть підтримки спеціалістів для належного проектування, впровадження та підтримки.

Отже, коли мова заходить про розробку власної системи керування документами, ви повинні розглянути, чи зможете ви зробити її складною та достатньо безпечною для ваших цілей за реалістичну ціну. І навіть якщо ви можете, яке технічне обслуговування знадобиться, щоб підтримувати його в робочому стані в довгостроковій перспективі?

Для багатьох компаній вибір спеціально розробленої власної DMS вирішує ці проблеми, усуваючи головний біль проектування, розробки та підтримки системи. Ця опція дозволить вам зосередитися на найважливіших завданнях керування власними продуктами та обслуговування клієнтів.

Локальна чи хмарна

Стороннє рішення для керування документами може запропонувати вам високобезпечну, настроювану та складну функціональність DMS, знімаючи з вас головний біль щодо специфікацій і розробки. Але чи варто вибирати локальну установку чи хмарне рішення SaaS?

Локальні рішення вимагають від вас використання власних серверів і сховища, а це означає, що вам все одно доведеться виконувати власне обслуговування. Але ви можете бути більш впевнені, що всі ваші дані знаходяться під вашим контролем. Недоліки можуть включати великі початкові витрати та час, який знадобиться для встановлення та навчання користуванню.

Хмарні рішення, як правило, більш адаптивні та гнучкі. Але якщо ви використовуєте хмарне рішення, як його налаштувати? Ви отримуєте спеціальну

послугу, як-от віртуальний приватний сервер, чи ділитеся ресурсами? Чи файли зашифровані під час передавання? Як щодо безпеки центру обробки даних, резервного копіювання та резервування?

Яке б рішення ви не вибрали, переконавшись, що воно розроблено, щоб допомогти вам зберігати та обробляти інформацію відповідно до суворих стандартів ISO 27001, буде свідченням того, що компанія, з якою ви маєте справу, здатна забезпечити найвищий рівень безпеки даних.

Вам потрібне управління документами чи контроль документів?

Це важлива відмінність, яка стосується суті того, що вам може знадобитися для досягнення DMS. Хороший документообіг – це легкість і гнучкість зберігання, пошуку, оновлення, відстеження та спільного використання документів. Хороший контроль документів полягає у створенні ієрархій доступу, безпеки, можливостей аудиту, керування версіями, перегляду, затвердження та протоколу видачі. Не кожній організації потрібно буде досягти найвищого рівня контролю документів – тих рівнів, яких можуть вимагати аерокосмічні чи фармацевтичні компанії, – але багатьом знадобиться більше, ніж може підтримувати базове рішення Google Drive.

Чи потрібно, щоб моя СЕД була масштабованою?

Ці міркування повинні включати набагато більше, ніж можливість збільшити простір для зберігання або додати місця за розумну ціну. Йдеться про можливість відповідати складнішим вимогам обробки документів за допомогою потрібних інструментів у потрібний час для вашої компанії. На початку вашого бізнесу вам може знадобитися безпечне сховище з можливістю пошуку та простір для спільної роботи, щоб ваша команда могла почати колективну роботу над ідеями. Але коли ці ідеї перетворюються на реальні комерційні плани, можливості та терміни, вам може знадобитися ваша DMS, щоб працювати більше як система управління якістю . Для цього може знадобитися більш складний контроль документів і інструменти «фазового стробування» або здатність створювати «технічні файли», які можна перевірити, відповідно до суворих

специфікацій.

Вибір системи керування документами, яка може зростати разом зі складністю ваших потреб, може запобігти тому, щоб ви на ранній стадії загрузли у вимогах до документації, призначених для більших, відоміших корпорацій. Але це також може дати вам можливість швидко реалізувати більш контрольований підхід, коли ви починаєте думати про отримання ISO або підтвердження відповідності нормам.

Мене турбує вартість?

Очевидно. Але майте на увазі, що найдешевші можуть вимагати компромісів, коли мова заходить про контроль документів і масштабованість, тоді як найдорожчі «найкращі у своєму роді» вибори можуть заважати вашій маневреності в ключові моменти на шляху вашої компанії.

Так звані недорогі рішення, такі як Dropbox і OneDrive, можуть швидко стати дорогими, якщо ви оновите їх до версій Enterprise, оскільки вам потрібно додати місця та більше пам'яті. У той же час кількість зусиль, які ви витрачаєте на пошук обхідних шляхів для загальних рішень, щоб цифрові підписи та інші інтеграції працювали так, як вам потрібно, може збільшити ваші накладні витрати.

Але вартість «найкращого у своєму роді» програмного забезпечення для управління якістю може бути справді непомірно високою для бізнесу на початковій стадії: трирічні мінімальні контракти на суму понад 10 тисяч на рік не є рідкістю. Більше того, СЕД і процеси контролю документів, які вони можуть вимагати від вас, можуть не підходити для бізнесу в повному режимі ідей.

Висновок

Зазвичай рішення щодо DMS приймаються на основі того, що здається найпростішим для швидкого та дешевого виконання – лише для того, щоб організації згодом усвідомили обмеження свого вибору. Інші обирають найнадійніші та найсуворіші рішення, які вони можуть знайти (з цілком слушних причин), лише щоб відчути себе стриманими через свою негнучкість, коли

настають комерційні реалії.

Створення списку вимог MoSCoW (речей, які продукт *повинен мати*, *повинен мати*, *міг би мати* та *ще не матиме*) допоможе вам визначити пріоритети того, що для вас важливо зараз, і подумати про те, що стане важливим у майбутньому. Демонстрації та випробування потенційних рішень DMS також є хорошим місцем для початку. Вони можуть допомогти вам визначити, що вам дійсно потрібно і з чим ви можете працювати.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка додатків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка додатків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки додатків Windows дозволяють автоматизувати процес створення додатка. Для цього використовуються генератори додатків. Програміст відповідає на питання генератора додатків і визначає властивості додатка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор

додатків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої додатки. Подібні засоби автоматизованого створення додатків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики додатка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні додатки, а також додатки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину додатка програмістові прийдеться розробляти самостійно. Вихідний текст додатка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон додатка – це вже половина всієї роботи. Вихідні тексти додатків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти додатків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої додатки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-додатки на базі об’єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У

міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-додатків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система керування електронними документами (СЕД) – це тип програмного забезпечення, яке зберігає, організовує та керує документами у формі електронних файлів для організації. Ця система найкраще працює для документів, які не зазнають значних змін, таких як юридичні файли, фінансові звіти, маркетингова допомога та відсканові публікації.

Цифрова система електронного документообігу кафедри кібербезпеки та програмного забезпечення – це тип електронної системи керування документами, спеціально створеної для інженерних документів і креслень. Він централізує, організовує та керує цими документами, надаючи користувачам повні, дієві засоби та дані про активи, які вони можуть використовувати для підвищення ефективності та економії грошей. Зрештою, він забезпечує єдине джерело правди для всієї документації, допомагає організаціям підтримувати відповідність нормативним вимогам, надає настроювані робочі процеси для оптимізації співпраці з усіма внутрішніми відділами, забезпечує комплексний контроль версій документів і спрощує аудит.

Існує кілька ключових особливостей найкращої у своєму класі системи електронного документообігу кафедри кібербезпеки та програмного забезпечення. Перш за все, правильна система управління документами має бути простою у використанні, інтуїтивно зрозумілою та прийнятою вашими співробітниками. Він також має надавати єдине джерело правди для всіх критично важливих даних, щоб підтримувати якість, точність і повноту ваших даних і метаданих. Крім того, він пропонує прості в налаштуванні робочі процеси, які дозволяють спрощену співпрацю, простий контроль версій документів і аудит документів для підтримки нормативної відповідності. Це

також гарантує, що всі працюють над правильною версією, надає доступні дані для максимального підвищення продуктивності персоналу та забезпечує безпеку персоналу.

Правильна система електронного документообігу кафедри кібербезпеки та програмного забезпечення забезпечить багато переваг, зокрема допоможе користувачам обробляти, зберігати та отримувати записи, спрощувати контроль версій документів і оптимізувати співпрацю між відділами, установами та пристроями. Це також може покращити часові рамки проекту, готовність до аудиту та перевірки, посилити контроль і безпеку документів, знизити витрати, оптимізувати управління знаннями та сприяти відповідності.

Існують ключові відмінності між системою електронного документообігу кафедри кібербезпеки та програмного забезпечення (СЕД) і системою управління контентом (CMS). Хоча обидва інструменти працюють для керування документами та вмістом, CMS зазвичай використовується для зберігання веб-вмісту з різних веб-сайтів, тоді як СЕД більше зосереджується на інженерній документації для зберігання, довідок, управління робочим процесом і вдосконалення процесів.

У всьому світі існує багато конкуруючих стандартів для інструментів BIM та EDM, хоча стандарти ISO по суті замінюють їх, і згодом їх прийме весь світ. До них відноситься ISO 9001, який передбачає контроль документів. Іншими словами, ISO 9001 намагається гарантувати, що потрібні люди мають доступ до документів у разі потреби, що ці документи є актуальними та що жоден неавторизований користувач не може отримати доступ або змінити вміст документа. Зрештою, це допомагає гарантувати актуальність, вичерпність і достовірність основних інженерних документів, підвищуючи відповідність і підвищуючи ефективність роботи з часом.

СЕД може надати багато необхідних покращень для операцій і робочих процесів, особливо для компаній з великою кількістю документації або тих, хто хоче цифрово трансформувати свою діяльність. Ці переваги включають економію

коштів, спрощену відповідність стандартам ISO 9001, правилам Sarbanes Oxley, HIPAA та іншим регуляторним органам, покращене аварійне відновлення, продуктивність співробітників, керування аудитом, автоматизацію робочих процесів документів і безпеку.

Особливості вибору й впровадження СЕД

Основні особливості вибору СЕД

На думку галузевих аналітиків, СЕД стає необхідною, коли загальний обсяг документів, щорічно оброблюваних у підприємстві або організації, досягає 4000-5000. Варто сказати, що впровадження СЕД не є панацеєю від всіх проблем підприємства. Головною метою її впровадження є підвищення ефективності документообігу підприємства або організації, а, отже, у якімсь ступені, і ефективності їхньої роботи в цілому. Є й така думка, що головною метою впровадження СЕД є створення ефективного середовища керування й функціонування підприємства або організації. Перед вибором СЕД обов'язково необхідно сформулювати перелік завдань, які повинні допомогти вирішити її впровадження. Крім того, потрібно розробити докладний організаційний план її впровадження. Поставку й впровадження СЕД повинна здійснювати зовнішня фірма, що несе повну юридичну відповідальність перед підприємством за успіх проекту впровадження. До вибору фірми-постачальника СЕД потрібно підходити дуже скрупульозно, так як після підписання контракту, проплати яких-небудь засобів і початку впровадження СЕД із цією фірмою буде непросто припинити відносини у випадку, якщо підприємство буде не повністю задовольняти якість її роботи. Фактично, у випадку неправильного вибору фірми-підрядника, будуть дарма загублені гроші, час і нерви. Звичайно, при виборі СЕД необхідно уважно вивчити всі пропозиції, наявні на ринку, і обов'язково провести тендер. Дуже важливо також урахувати, є чи досвід успішних впроваджень у постачальника СЕД і наскільки він відповідає специфіці діяльності підприємства або організації. При цьому при виборі СЕД необхідно враховувати ряд наступних важливих моментів:

– Досить велика розмаїтість СЕД (як вітчизняної, так і закордонної розробки), представлених на українському ринку.

– Надійність компанії-постачальника СЕД і пропоновані нею умови поставки, впровадження й супроводи (за деякі роки існування вітчизняного IT-ринку вже були випадки, коли досить відомі й зовні благополучні компанії йшли з ринку.

– Можливість доробки СЕД у розумний термін (із прийнятними ціновими умовами) під специфіку підприємства.

У загальному випадку при виборі фірми-постачальника СЕД потрібно врахувати наступні вимоги:

– Серйозність фірми (наявність відомого ім'я на ринку, свого постійного офісу, бажання зберегти й зміцнити свій ринковий імідж за рахунок успішної реалізації чергового проекту впровадження СЕД і ін.).

– Розмір фірми, наявність у неї достатніх ресурсів для виконання проекту впровадження в договірний термін.

– Наявність у фірмі досить великого й стабільного колективу розроблювачів СЕД.

– Досвід фірми в розробці й впровадженні СЕД в аналогічних підприємствах і організаціях (найкраще, якщо їсти можливість подивитися впроваджені СЕД у роботі й поспілкуватися з їхніми користувачами).

– Чи існують у фірмі-розроблювачі технологія й стандарти по програмуванню, як вони оформлені (узаконені) і підтримуються.

– Чи існує у фірмі технологія впровадження, і як вона підтримується документально.

При цьому обирана для впровадження СЕД повинна задовольняти наступним загальним вимогам:

– В основі архітектури системи лежать бізнес-процеси (потоки робіт), у ході виконання яких створюються й переміщаються документи.

– У СЕД обов'язково повинна бути врахована специфіка українського документообігу й діловодства (вертикальна організація документообігу, відповідність українським ДСТ і ін.).

– У СЕД повинні бути реалізовані можливості спільної роботи (календарне групове планування, спільне використання інформації, "дошки оголошень", форуми й ін.).

– Повинні бути реалізовані функції оперативного контролю виконання документів і робіт (передача робіт між виконавцями відповідно до певної технології, контроль стану виконуваного процесу, виявлення відхилень процесу від його нормативного ходу, прогнозування впливу цих відхилень на ймовірний строк завершення всього процесу в цілому й ін.).

– Простота й гнучкість при установці, конфігуруванні й експлуатації.

– Використання розповсюджених платформ для організації групової роботи з документами.

– Наявність засобів для організації конфіденційного документообігу із захистом інформації від несанкціонованого доступу.

– Сумісність із сертифікованими засобами захисту інформації.

– Можливість одночасного використання електронних і паперових документів.

– Можливість інтеграції з розповсюдженими платформами.

– Гарна масштабованість.

– Наявність можливостей автоматизованого збору й аналізу статистичних даних про рух документів.

– В основі СЕД – відкрита клієнт-серверна архітектура.

– Можливість інтеграції з іншими додатками (CAD-системами, MRP/ERP системами, системами фінансового й правлінського обліку, системами електронної пошти й ін.).

– Доступність бази даних СЕД для інших додатків.

– Модульність СЕД і можливість нарощування її базових можливостей за допомогою убудованих інструментальних засобів.

– Можливість розподіленої обробки документів, "прозорої" для всіх користувачів СЕД.

– Наявність у СЕД модулів для сканування документів (або можливість інтеграції із професійними системами керування образами).

– Можливість роботи через Internet/intranet.

– Можливість роботи з мобільними (віддаленими) користувачами й групами користувачів.

– Підтримка української мови, включаючи екранні повідомлення й підказки, сортування даних і пошук інформації з різних слів і виражень.

– Прийнятність за ціною при поставці, впровадженні й супроводі.

Основні особливості впровадження СЕД

У свою чергу, при впровадженні СЕД доводиться зіштовхуватися з наступними основними проблемами, від рішення яких залежить успіх усього проекту впровадження:

– у більшості випадків, необхідність масштабної реорганізації підприємства;

– слабка формалізація бізнес-процесів і відсутність корпоративних стандартів;

– СЕД повинна бути впроваджена й використовуватися на всьому підприємстві (скрізь, де створюється, коректується й зберігається інформація), інакше успіх від її впровадження буде мінімальний (якщо взагалі буде);

– наявність певного опору впровадженню СЕД з боку співробітників підприємства ("опір змінам"), нерідко викликаного небажанням якої-небудь "прозорості" своєї діяльності;

– відсутність необхідного рівня підготовки в співробітників підприємства (у тому числі, і керівників нижнього, середнього й верхнього рівнів) до роботи із СЕД.

Впровадження СЕД варто здійснювати поступово, починаючи з найбільш важливої ланки документообігу (добре описаного і зрозумілого), автоматизація

якого дозволить швидко одержати позитивний ефект. У процесі впровадження СЕД необхідно забезпечити можливість роботи як по нової, так і за старою технологією, щоб не заважати повсякденній діяльності підприємства. Дуже важливу роль грає реальна підтримка керівництвом підприємства проекту впровадження (так званий "фактор першої особи"). При відсутності такої підтримки (наприклад, навіть в організації одержання всієї необхідної інформації при обстеженні підприємства), у найкращому разі, система буде впроваджена лише в окремих підрозділах підприємства (навряд чи від цього можна чекати якого-небудь помітного повернення інвестицій).

Проблему "опору змінам" можна вирішити шляхом поступового й планомірного впровадження елементів електронного документообігу, починаючи з найпростішого (наприклад, навчання співробітників роботі з електронною поштою й intranet-мережею, організації електронного архіву й ін.) і проведення необхідної роз'яснювальної роботи. У ході впровадження СЕД обов'язково варто організувати тренінги для співробітників підприємства, а також консультації для його посібника з організації переходу на електронну форму діловодства. Масштабне впровадження СЕД обов'язково повинне випереджатися пілотним проектом, у ході виконання якого з'ясовуються основні проблеми, які можуть зустрітися безпосередньо при впровадженні. Головне завдання пілотного проекту полягають у визначенні того, є чи ні (або передбачається) відчутний ефект від впровадження СЕД. У випадку успіху пілотного проекту (так званого "пілота") приймається остаточне рішення про впровадження СЕД, розробляється реальний проект впровадження разом з повним планом впровадження. Звичайно вартість пілотного проекту досягає 10% від вартості реального проекту.

Як правило, впровадження СЕД на підприємстві включає наступні основні етапи:

– Ретельний аналіз бізнес-процесів підприємства, стану використовуваного встаткування й технології.

– Розробка інформаційно-функціональної моделі підприємства, реінжиніринг його бізнес-процесів.

– Аналіз можливої конфігурації апаратно-програмних засобів, необхідної для впровадження СЕД.

– Виконання пілотного проекту.

– Затвердження результатів виконання пілотного проекту й розробка плану впровадження.

– Вибір і поставка необхідних для впровадження СЕД апаратно-програмних засобів.

– Поставка й інсталяція СЕД.

– Адаптація й налаштування СЕД.

– Перенос і конвертація даних з успадкованих систем.

– Навчання системних адміністраторів і користувачів роботі із СЕД.

– Підготовка контрольного приклада, програми й методики випробувань, проведення повного тестування СЕД.

– Розробка проектної, програмної, технічної й користувальницької документації.

– Завершення впровадження СЕД, здача її в промислову експлуатацію;

– Супровід СЕД.

3.2 Розробка структурної схеми

Структурна схема електронного документообігу кафедри кібербезпеки та програмного забезпечення зображена на рисунку 3.1. Вона побудована з урахуванням схеми розробленої відкритої системи обміну даними, яка розроблена на основі проведеного дослідження існуючих еталонних моделей, за результатами яких за основу для синтезу еталонної моделі середовища відкритої системи обміну даними була прийнята модель, описана в міжнародному стандарті ISO/IEC TR 14252:1996 (E), ANSI/IEEE Std 1003/ 0-1995 Information technology – Guide to the POSIX Open System Environment (OSE).

Рисунок 3.1 – Структурна схема системи

На основі даної моделі була розроблена еталонна модель середовища системи обміну даними, що складається із трьох технологічних рівнів:

– зовнішнього середовища;

– програмного забезпечення середнього рівня (посередника);

– користувальницьких додатків.

Розглянемо більш докладно елементи структурної схеми СЕД кафедри кібербезпеки та програмного забезпечення.

Система СЕД кафедри кібербезпеки та програмного забезпечення, побудована за допомогою предметно-орієнтованого інструмента, має багаторівневу архітектуру. Архітектура виступає гарантом доступності, надійності й безпеці системи, що дозволяє системі СЕД кафедри кібербезпеки та програмного забезпечення охопити всіх комп'ютеризованих співробітників (викладачів та учбово-допоміжний персонал) і підвищити ефективність роботи організації в цілому.

Основними структурними елементами архітектури є:

– Предметно-орієнтований інструмент розробки – середовище виконання коду, що реалізує інтерфейс служб і користувальницьких додатків (у тому числі сторонньої розробки) для доступу до системи. Зокрема, сервер веб-доступу СЕД кафедри кібербезпеки та програмного забезпечення, реалізований на платформі ASP.NET, використовує предметно-орієнтований інструмент розробки для реалізації всіх функцій системи, які стають доступні користувачам через веб-браузер.

– Служби перетворення документів. Перетворення документів в інші формати, добування з документів корисної інформації

– Служби введення документів. Масове введення документів у СЕД кафедри кібербезпеки та програмного забезпечення з різних джерел (сканери, БФП, файлова система, факси, електронна пошта й т.д.).

– Набір засобів інтеграції. Легка інтеграція з ERP-системами: двостороння синхронізація довідників, включення об'єктів системи в

workflow, робота з документами з ERP-системи. Workflow – це повна або часткова автоматизація бізнес-процесу, при якій документи, інформація або завдання передаються від одного учасника (бізнес-процесу) до іншого для виконання дій відповідно до набору керівних правил.

– Сервер веб-доступу до СЕД кафедри кібербезпеки та програмного забезпечення. Робота з електронними документами, завданнями й завданнями через веб-браузер.

– Розширення для SharePoint. Набір готових веб-частин і інтеграційних механізмів, що забезпечують доступ до даних СЕД кафедри кібербезпеки та програмного забезпечення з порталу на базі Microsoft SharePoint.

– Клієнти системи СЕД кафедри кібербезпеки та програмного забезпечення – додатки для кінцевих користувачів, інструментарій розробки, утиліти адміністрування системи. Клієнтом може бути як Windows-додаток, що використовує для доступу до системи предметно-орієнтований інструмент розробки, так і веб-браузер.

– Керуючі служби СЕД кафедри кібербезпеки та програмного забезпечення – служби, що забезпечують керування системою. Наприклад, служба workflow управляє роботою завдань СЕД кафедри кібербезпеки та програмного забезпечення, а сервіси зберігання СЕД кафедри кібербезпеки та програмного забезпечення відповідають за файлові сховища документів. Всі керуючі служби можуть бути встановлені як на один комп'ютер, так і на різні – з метою розподілу навантаження.

– Сервер реплікації. Створення ієрархічної системи розподілених систем, що обмінюються даними в режимі off-line; налаштовується состав даних, що, реплікуються.

– Служба файлових сховищ. Керування довгостроковим і архівним зберіганням великого обсягу документів, у тому числі медіаданих; настроювання політик переміщення даних по сховищах.

– СУБД – сховище даних і метаданих системи. Одним з важливих компонентів системи, що зберігаються в СУБД, є прикладна розробка СЕД кафедри кібербезпеки та програмного забезпечення, що визначає функціональність предметних модулів системи, замовлених, а також розроблених партнерами СЕД кафедри кібербезпеки та програмного забезпечення рішень.

– Файлові сховища – архіви більших або рідко використовуваних документів, які ефективніше тримати за межами СУБД; управляються власними службами.

Архітектура системи СЕД кафедри кібербезпеки та програмного забезпечення, будучи частиною інформаційної інфраструктури організації, демонструє характеристики, важливі для будь-якої корпоративної системи:

– Відкритість. Основа системи СЕД кафедри кібербезпеки та програмного забезпечення – платформа предметно-орієнтованого інструмента розробки – підтримує технології Microsoft COM і .NET. Вона містить готові інструменти інтеграції з корпоративними додатками, у тому числі набір функцій для обробки XML-документів. Корпоративні стандарти й відкрита структура даних дозволяють легко інтегрувати СЕД кафедри кібербезпеки та програмного забезпечення в інформаційну інфраструктуру організації.

– Розширюваність. Як правило, у кожній організації висувають унікальні вимоги до побудови електронного документообігу й рішенню завдань взаємодії. Об'єктна модель і предметно-орієнтований інструмент розробки дозволяють створювати власні й змінювати існуючі об'єкти для рішення специфічних завдань. Оскільки ядром системи є COM-сервер, що управляють функції системи можна використовувати в будь-яких сторонніх додатках.

– Масштабованість. Виділення декількох рівнів архітектури дозволяє підвищувати продуктивність системи не тільки за допомогою нарощування потужності апаратних засобів, але й завдяки розподілу служб по різних серверах. Механізм реплікації предметно-орієнтованого інструмента розробки дозволяє побудувати територіально розподілену систему, мінімізуючи як вимоги

до пропускної здатності каналів зв'язку за рахунок обсягу переданих даних між серверами, так і технічні вимоги до вторинних серверів. Виділення як SQL-серверних, так і файлових сховищ документів дозволяє гнучко управляти розподілом навантаження на сервера організації при доступі до документів.

– Надійність. Архітектура СЕД кафедри кібербезпеки та програмного забезпечення підтримує транзакційну модель, що гарантує цілісність дані системи протягом всіх стадій їхнього життєвого циклу. Керовані SQL– і файлові сховища документів дозволяють організувати надійне зберігання документів.

– Безпека. Для кожного об'єкта системи може бути задано, які користувачі або групи мають право виконувати з ним певні дії. Конфіденційні електронні документи й завдання можуть бути зашифровані безпосередньо в системі будь-яким CryptoAPI-сумісним криптопровайдером (у тому числі сертифікованим ДСТЗІ СБУ), що гарантує захист навіть від осіб, що мають необмежений доступ до даних. Протоколювання всіх дій користувача дозволить відновити історію роботи з об'єктами системи у випадку порушення режиму безпеки. Забезпечується високий захист від несанкціонованого доступу до сховищ документів всіх типів.

Таким чином, архітектура системи СЕД кафедри кібербезпеки та програмного забезпечення розроблена з урахуванням максимального використання всіх переваг сучасних технологій, платформ і предметно-орієнтованого підходу до побудови інформаційних систем керування.

3.3 Розробка функціональної схеми

На основі структурної схеми моделі середовища системи обміну даними була розроблена функціональна схема моделі середовища відкритої СЕД кафедри кібербезпеки та програмного забезпечення, при цьому визначені наступні об'єкти стандартизації:

– інтерфейс користувача;

– служби засобів управління даними;

– служби засобів управління системою;

– служби засобів управління сховищем;

– служба комунікацій;

– служби засобів проектування;

– служби засобів управління змістом;

– служби засобів моделювання процесів;

– служби засобів моделювання даних;

– формати файлових об'єктів;

– служби засобів забезпечення безпеки.

Отримані дані дозволили синтезувати функціональну схему СЕД кафедри кібербезпеки та програмного забезпечення (рисунок 3.2).

Дана модель використана при розробці функціонального стандарту середовища відкритої СЕД кафедри кібербезпеки та програмного забезпечення. Призначенням даного стандарту є:

– Забезпечення мобільності (переміщуваності) СЕД кафедри кібербезпеки та програмного забезпечення.

– Забезпечення інтероперабельності СЕД кафедри кібербезпеки та програмного забезпечення.

– Забезпечення масштабованості СЕД кафедри кібербезпеки та програмного забезпечення.

– Забезпечення адаптуємості системи.

Визначено, що область застосування функціонального стандарту СЕД кафедри кібербезпеки та програмного забезпечення установлює загальні положення по створенню й експлуатації відкритої системи управління документообігом на основі окремих модулів, що входять у єдину інформаційну систему організації.

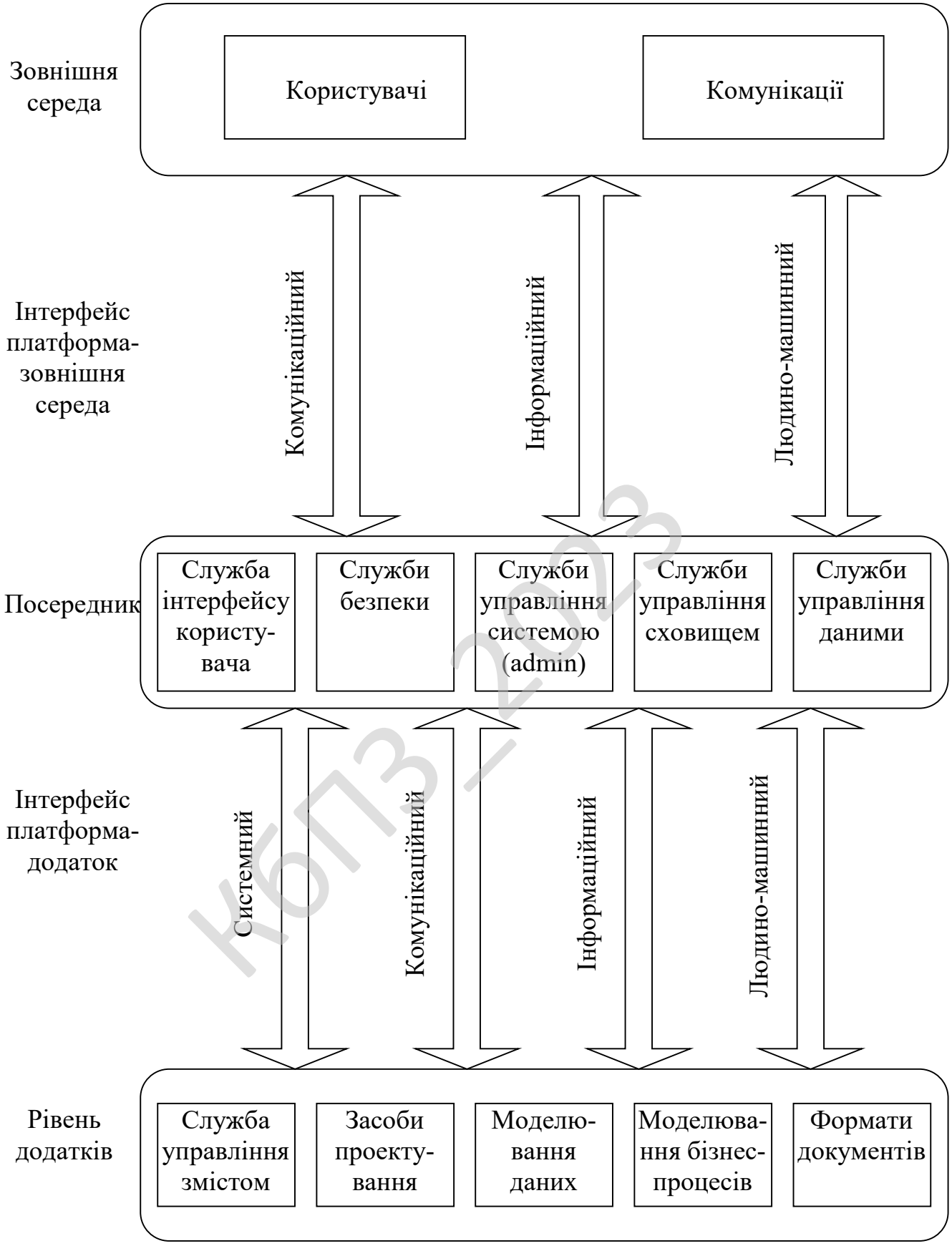

Рисунок 3.2 – Функціональна схема системи

Положення функціонального стандарту підлягають застосуванню при рішенні завдань:

– створення, модернізації СЕД кафедри кібербезпеки та програмного забезпечення;

– організації доступу користувачів до ресурсів СЕД кафедри кібербезпеки та програмного забезпечення;

– інтеграції обчислювальних і інформаційних ресурсів із СЕД кафедри кібербезпеки та програмного забезпечення.

Для заповнення профілю в роботі була розроблена методика вибору стандартів на основі експерименту.

Результатом розробки функціонального стандарту є набір вимог до елементів відкритої СЕД кафедри кібербезпеки та програмного забезпечення у вигляді функціонального стандарту – Профілю середовища відкритої СЕД кафедри кібербезпеки та програмного забезпечення.

Проведемо дослідження особливості функціонування елементів системи управління електронним документообігом і експериментальним дослідженням з визначення ефективного формату файлових об'єктів СЕД кафедри кібербезпеки та програмного забезпечення, як засобу підвищення техніко-економічних характеристик СЕД кафедри кібербезпеки та програмного забезпечення.

Виходячи з вимог до функціонала систем управління й аналізу існуючих СЕД кафедри кібербезпеки та програмного забезпечення, у системах управління виділені типові елементи (ТЕ):

– ТЕ «обробки інформації».

– ТЕ «передачі інформації».

– ТЕ «сполучна лінія».

– ТЕ «масив зберігання інформації».

– ТЕ «точка діалогу».

Дані типові елементи служать для опису інформаційної системи й інформаційних потоків у будь-якій системі, що неможливо зробити через описані

вище служби еталонної моделі середовища відкритої СЕД кафедри кібербезпеки та програмного забезпечення.

Таким чином показано, що через подібний базис типових елементів може бути представлена кожна, як завгодно складна інформаційна система.

Для проведення функціональної стандартизації ТЕ здійснимо перехід від ТЕ до служб СЕД кафедри кібербезпеки та програмного забезпечення.

Для перевірки експериментальним шляхом залежності ефективності роботи типових елементів СЕД кафедри кібербезпеки та програмного забезпечення від форматів використовуваних файлових об'єктів і вибору стандартів для відповідного розділу профілю розроблені:

– вимірювальна установка для виміру часу мережної затримки;

– програма статистичного аналізу файлових об'єктів СЕД кафедри кібербезпеки та програмного забезпечення;

– методики проведення вимірів часу мережної затримки й розмірів файлів у сховище СЕД кафедри кібербезпеки та програмного забезпечення.

Визначено співвідношення розмірів файлів залежно від типу інформації у файлах і їхніх розмірах. Визначено формати, у яких розмір файлу виходить мінімальним. Визначено, як саме впливає формат файлових об'єктів (ФФО) на роботу служб СЕД кафедри кібербезпеки та програмного забезпечення. На рисунку 3.2, на моделі СЕД кафедри кібербезпеки та програмного забезпечення показані служби, на роботу яких впливає формат файлових об'єктів. Ці служби виділені жовтим кольором.

Результати досліджень дозволили визначити залежність розмірів файлових об'єктів від типу їхнього вмісту й залежність часу мережної затримки, що підтвердило, що швидкодія системи залежить від розміру файлових об'єктів СЕД кафедри кібербезпеки та програмного забезпечення.

Також дослідження дозволило підтвердити залежність ефективності роботи СЕД кафедри кібербезпеки та програмного забезпечення від ФФО й визначити ефективний формат файлових об'єктів СЕД кафедри кібербезпеки та

програмного забезпечення, що дозволяє мінімізувати необхідні для роботи СЕД кафедри кібербезпеки та програмного забезпечення комунікаційні й обчислювальні ресурси, що дозволило гармонізувати стандарти файлових об'єктів у функціональному стандарті (профілі) середовища відкритої системи управління електронним документообігом і підвищити ефективність роботи ТЕ.

Таким чином показано, що залежно від структури документів є ефективні можливості по поліпшенню роботи СЕД кафедри кібербезпеки та програмного забезпечення залежно від сфери її застосування.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Головне вікно ПЗ.

– Введення документів.

– Перетворення документів.

– Моніторинг БД.

– Служба workflow.

– Сервер сеансів.

– Агент реплікації.

– Система управління електронним документообігом.

– Служба файлових сховищ.

– Файлові сховища.

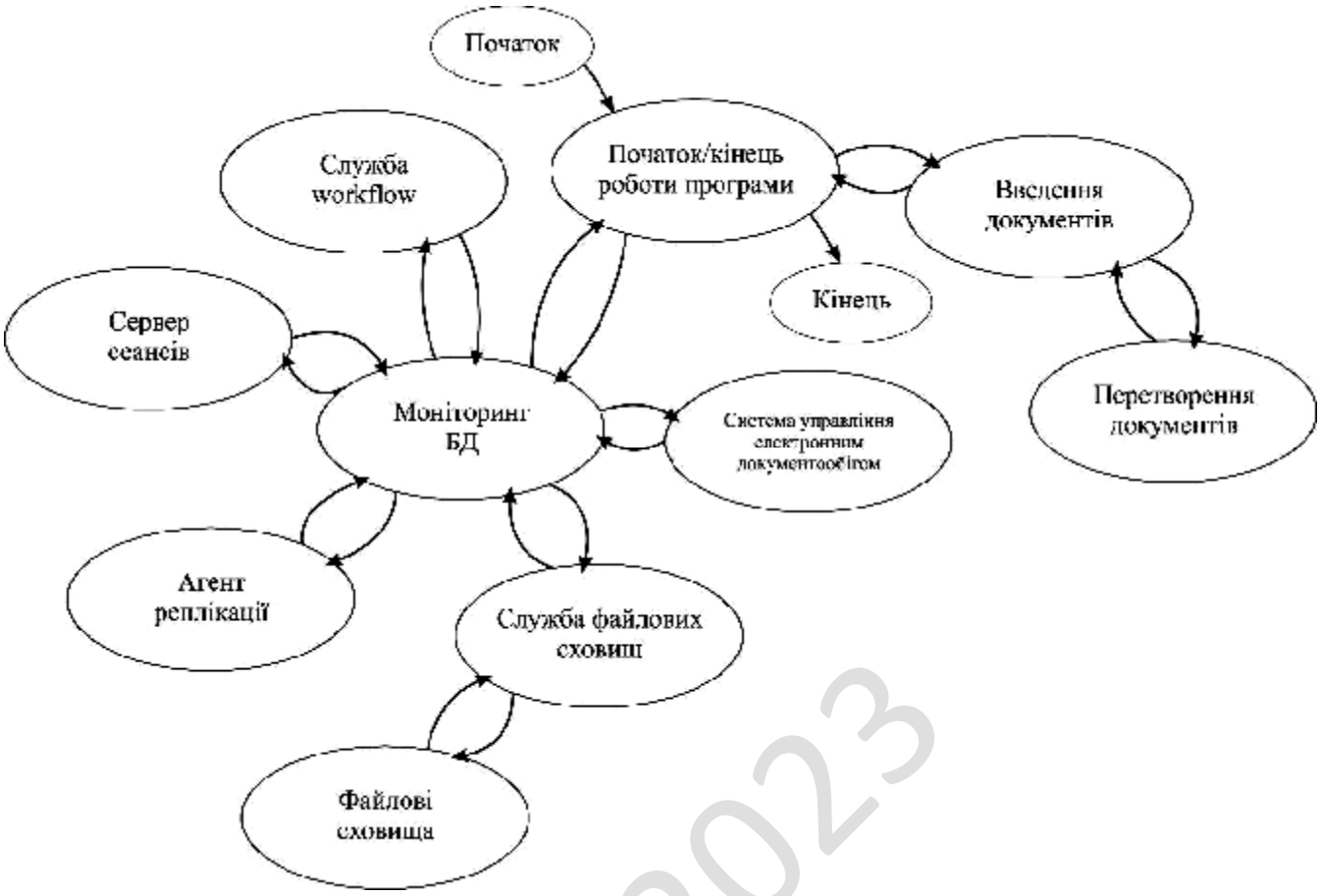

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

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

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було реалізовано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

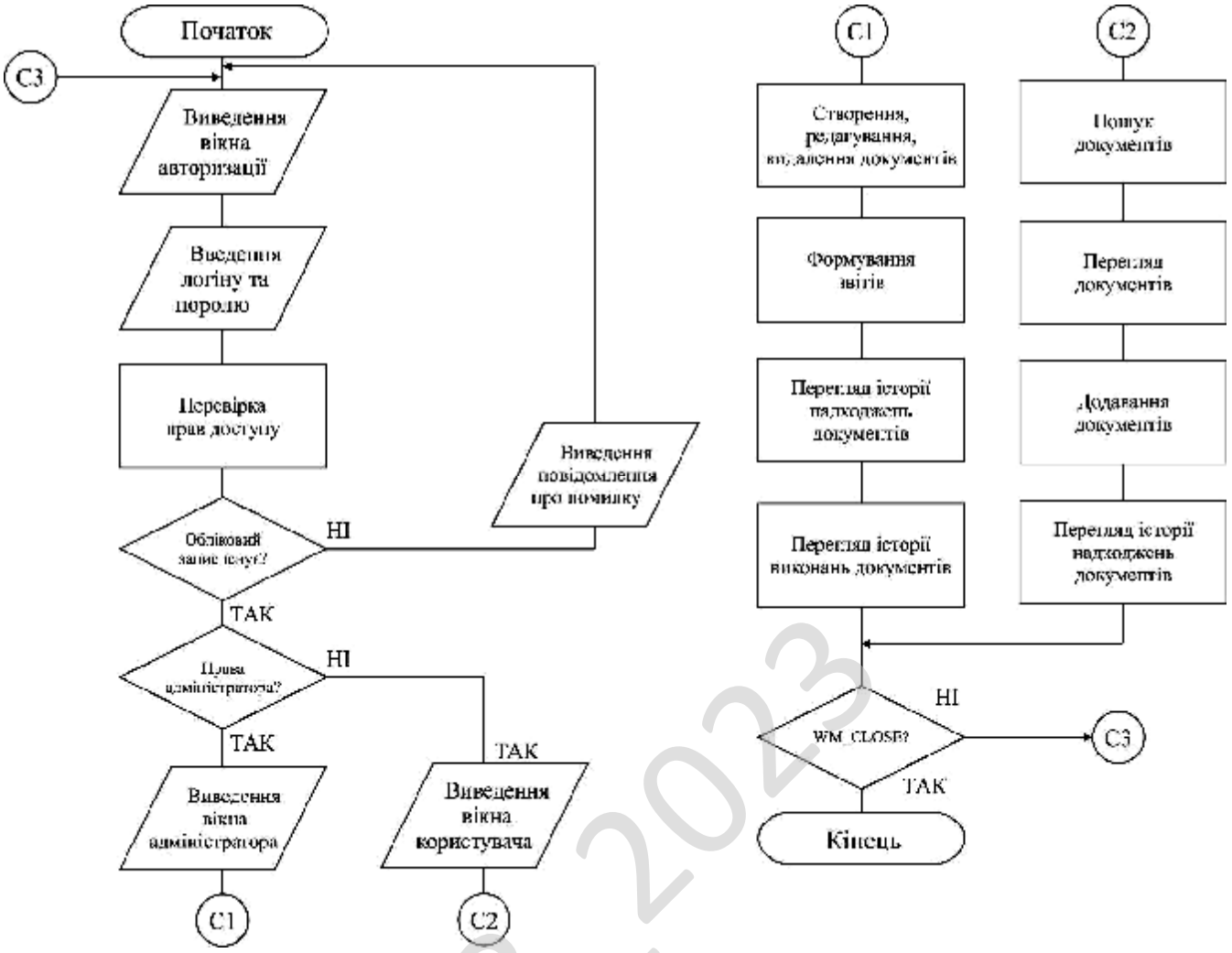

Рисунок 4.1 – Блок схема основної програми

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою. Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у

кілька разів і помітно поліпшити якість кінцевого продукту. UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

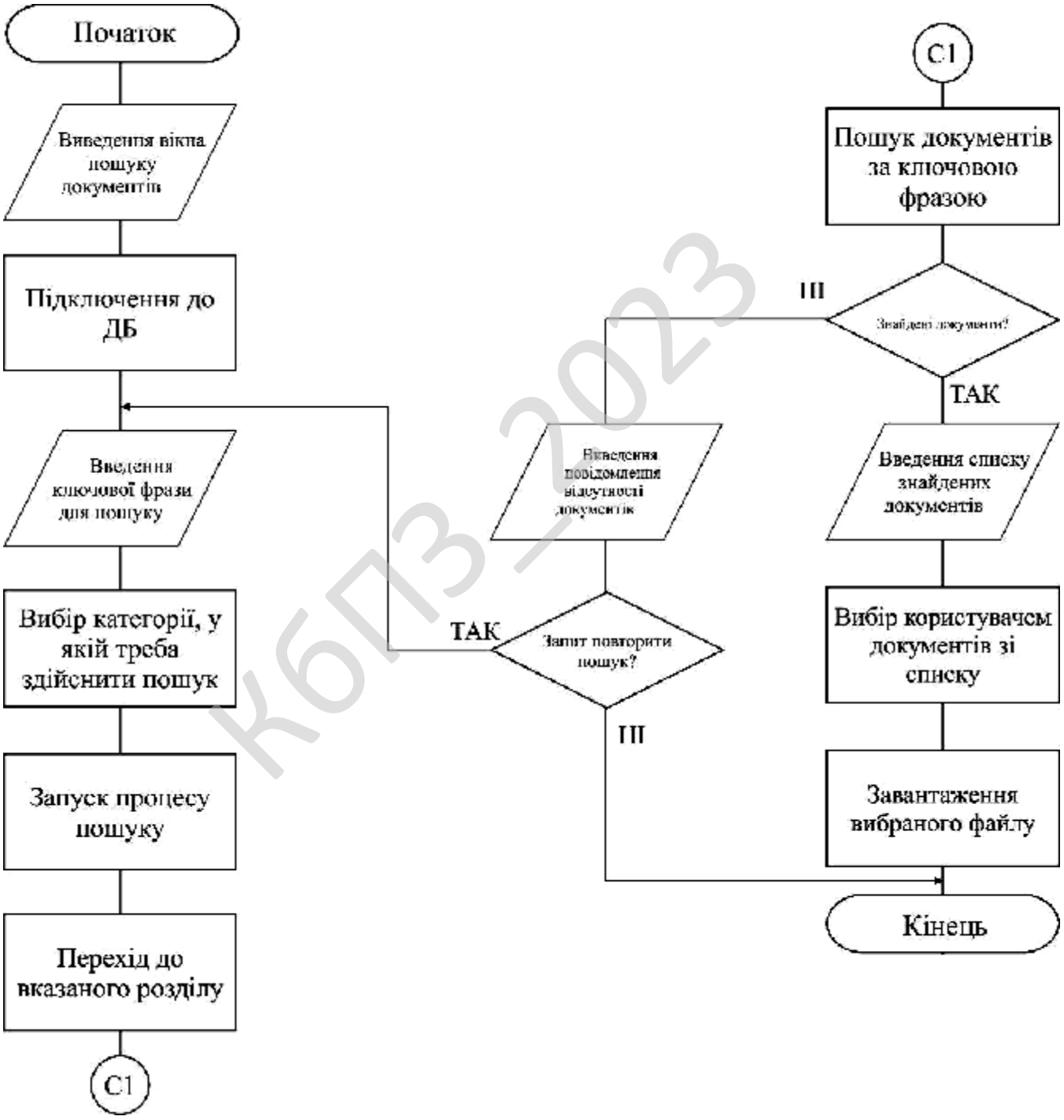

Рисунок 4.2 – Блок схема підпрограми

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом;

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів;

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії;

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо опис алгоритмів функціонування системи.

Проведемо оцінку якості роботи системи управління електронним документообігом. Виявлені переваги застосування теорії нечітких множин для рішення завдання оцінки ефективності роботи СУЕД у порівнянні із традиційними підходами теорії автоматичного управління. Розроблено критерії оцінки якості роботи системи управління електронним документообігом.

Показано архітектуру нечіткої моделі управління якістю роботи СУЕД. Основним модулем системи оцінки якості роботи СУЕД є блок прийняття рішень, хоча інші блоки не менш важливі для нормального функціонування моделі. Блок

оцінки станів, на основі вступник на його вхід інформації, будує формалізований опис ситуації, що виникла при роботі СУЕД.

У блоці видачі керуючих впливів здійснюється перехід від внутрішньої форми завдання керуючих рішень до зовнішньої форми.

Для оцінки якості роботи СУЕД скористаємося критерієм, вираженим безліччю M, елементи якого є інтервалами якісної (нечіткої) шкали:

M={«дуже добре», «добре», «задовільно» «незадовільно»}, для цього методом експертних оцінок були визначені конкретні значення процентного вмісту тексту, графіки й таблиць у файлах певного формату.

Виходячи з аналізу експериментальних даних, був отриманий масив $\{W_N(X)\}$, що містить інтервали значень фізичної величини, що характеризує ефективність роботи елемента, при яких забезпечується ефективна робота елементів пристрою (системи) і масив $\{WW_N(X)\}$, що містить інтервали значень фізичної величини, що характеризує функціонування елемента, і значення функції приналежності стану елемента системи до одного з можливих станів.

Розроблено вирішальні правила виходячи зі значень функції приналежності стану елемента одному з можливих станів для кожного з інтервалів базових значень.

Для перевірки запропонованих правил була зроблена побудова моделі оцінки якості роботи системи управління електронним документообігом у середовищі MatLab 7.0 для СУЕД фінансової організації.

Модель заснована на методі Мамдани, що має у своїй основі базу знань у вигляді сукупності нечітких предикатних правил виду:

П1: якщо x є A1, тоді z є B1,

П2: якщо x є A2, тоді z є B2,

. . . . . .

Пn: якщо x є An, тоді z є Bn,

З метою перевірки даних про те, що використовувані критерії можуть застосовуватися для діючих систем проведений ряд експериментів –

інструментальних досліджень на діючій системі, для чого якого були обмірювані й визначені величини:

– кількість переданих у СУЕД файлів;

– формати переданих у СУЕД файлів.

Отримано підтвердження того, що використовувані для оцінки ефективності роботи СУЕД критерії можуть застосовуватися для діючих систем.

Розроблено методику оцінки ефективності роботи елементів СУЕД.

Зроблено оцінку ефективності й повернення інвестицій від впровадження відкритої системи управління електронним документообігом.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом.

Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Віддалення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження.

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Приведемо частину програмного коду, яка реалізує функції головного вікна.

```
// NAME:       CWorker
// DESCRIPTION:    Конструктор класу
// INPUT:      N/D
CWorker::CWorker(void)
{
  listDoc = gcnew ArrayList();
  logInput = gcnew CLogger();
  logOutput = gcnew CLogger();
```

```
}
/*****************************************************/
// NAME:       GetIndexByISBNHash
// DESCRIPTION:    Функція одержання індексу документа в загальному списку по
вхідному номеру документа
// INPUT:      ddHashValue – значення хеша вхідного номера документа
/*****************************************************/
Int32 CWorker::GetIndexByISBNHash(Int32 HashValue)
{
// Цикл пошуку документа із заданим вхідним номером документа
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
// Хеш-значення вхідного номера документа заданого документа збігається з
// хеш – значенням вхідного номера документа знайденого документа
     if(HashValue == ((CDoc^)listDoc[i])->GetISBNHash())
     {
// Документ знайдений, повернути індекс
          return i;
     }
  }
// Документ не знайдений, повернути -1
  return -1;
}
/*****************************************************/
// NAME:       AddDoc
// DESCRIPTION:    Функція додавання нового документа
// INPUT:      SourceDoc – об'єкт "документ" для включення в список
// OUTPUT:         TRUE – документ успішно доданий
//        FALSE – такий документ уже існує в системі
/*****************************************************/
Boolean CWorker::AddDoc(CDoc^ SourceDoc)
{
// Документи з таким вхідним номером документа не існує?
  if(this->GetIndexByISBNHash(SourceDoc->GetISBNHash()) == -1)
  {
// Додати заданній документ у загальний список
     listDoc->Add(SourceDoc);
// Записати подія в лог
     logInput->WriteEvent(
          "Новий документ '" + ((CDoc^)SourceDoc)->GetName() +
          "', Автор '" + ((CDoc^)SourceDoc)->GetAuthor() +
          "', Вхідний номер документа " + ((CDoc^)SourceDoc)->GetISBN());
```

```
// Функція відробила успішно
     return true;
  }
  else
  {
// Документ із таким же вхідним номером документа існує, повернути помилку
     return false;
  }
}
/*****************************************************/
// NAME:       RemoveDoc
// DESCRIPTION:    Функція видалення документів із системи
// INPUT:      ddISBNHash – значення хеша вхідного номера документа
// OUTPUT:          0 – видалення зроблене
//        1 – видалення неможливо, не всі екземпляри перебувають у системі
//        2 – документ із заданим вхідним номером документа не знайдений
/*****************************************************/
Int32 CWorker::RemoveDoc(Int32 ddISBNHash)
{
// Знайти документ по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
// Документ із таким вхідним номером документа існує?
  if(ddIndex != -1)
  {
// Перевірити, що жодного документа немає в користувача
     if(((CDoc^)listDoc[ddIndex])->GetTotalNumber() ==
          ((CDoc^)listDoc[ddIndex])->GetFreeNumber())
     {
// Записати подія в лог
          logOutput->WriteEvent(
               "Виконаний документ '" + ((CDoc^)listDoc[ddIndex])->GetName() +
               "', Автор '" + ((CDoc^)listDoc[ddIndex])->GetAuthor() +
               "', Вхідний номер документа " + ((CDoc^)listDoc[ddIndex])-
>GetISBN());
// Видалити знайдений документ зі списку
          listDoc->RemoveAt(ddIndex);
// Функція відробила успішно
          return 0;
     }
     else
     {
// Повернути помилку, не всі документи перебувають у системі
```

```
          return 1;
      }
  }
  else
  {
// Повернути помилку, документ із таким вхідним номером документа не був знайдений
     return 2;
  }
}
/*****************************************************/
// NAME:      GiveDoc
// DESCRIPTION:    Функція видачі документів користувачеві
// INPUT:     listDoc – список документів користувача (для виконання додавання)
//        ddISBNHash – хеш вхідного номера документа, що потрібно одержати
// OUTPUT:         TRUE – операція пройшла успішно
//        FALSE – у наявності немає жодного екземпляра заданого документа
/*****************************************************/
Boolean CWorker::GiveDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
// Визначити індекс по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
// Зменшити кількість вільних документів
  if((ddIndex != -1) && ((CDoc^)listDoc[ddIndex])->DecFreeNumber())
  {
// Додати документ у список користувача
     listReader->Add(listDoc[ddIndex]);
// Функція відробила успішно
     return true;
  }
  else
  {
// Такого документа не є в наявності
     return false;
  }
}
/*****************************************************/
// NAME:      TakeDoc
// DESCRIPTION:    Функція прийняття документів від користувача назад у систему
// INPUT:// listDoc – список документів користувача (для виконання видалення)
//        ddISBNHash – хеш вхідного номера документа, що потрібно повернути
/*****************************************************/
void CWorker::TakeDoc(ArrayList^ listReader, Int32 ddISBNHash)
```

```
{
// Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
// Видалити документ зі списку користувача
// listReader->RemoveAt(ddIndex);
// Збільшити кількість вільних документів
  ((CDoc^)listDoc[ddIndex])->IncFreeNumber();
}
/*****************************************************/
// NAME:       LoadDocList
// DESCRIPTION:    Функція завантаження документів системи з файлу
// INPUT:      N/D
/*****************************************************/
void CWorker::LoadDocList()
{
// Перевірити існування файлу
  if(!File::Exists(DOCS_FILE))
  {
// Файл не знайдений, закінчити виконання функції
     return;
  }
// Відкрити файл
  StreamReader^ srDoc = gcnew StreamReader(DOCS_FILE);
// Продовжувати, поки не буде знайдений кінець файлу
  while(srDoc->EndOfStream == false)
  {
// Створити новий об'єкт "документ"
     CDoc^ NewDoc = gcnew CDoc();
// Завантажити з файлу й установити параметри документа
     NewDoc->SetName(srDoc->ReadLine());
     NewDoc->SetAuthor(srDoc->ReadLine());
     NewDoc->SetISBN(srDoc->ReadLine());
     NewDoc->SetTheme(srDoc->ReadLine());
     NewDoc->SetPages(Convert::ToInt32(srDoc->ReadLine()));
     NewDoc->SetTotalNumber(Convert::ToInt32(srDoc->ReadLine()));
     NewDoc->SetFreeNumber(Convert::ToInt32(srDoc->ReadLine()));
// Включити документ у загальний список документів
     listDoc->Add(NewDoc);
  }
// Закрити файл
  srDoc->Close();
```

```
}
/*****************************************************/
// NAME:       SaveDocList
// DESCRIPTION:    Функція збереження документів системи у файл
// INPUT:      N/D
/*****************************************************/
void CWorker::SaveDocList()
{
// Перевірити існування файлу
  if(File::Exists(DOCS_FILE))
  {
// Видалити файл
     File::Delete(DOCS_FILE);
  }
// Створити й відкрити файл
  StreamWriter^ swDoc = gcnew StreamWriter(DOCS_FILE);
// Цикл запису по одному документу
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
// Одержати параметри документа й записати їх у файл
     swDoc->WriteLine(((CDoc^)listDoc[i])->GetName());
     swDoc->WriteLine(((CDoc^)listDoc[i])->GetAuthor());
     swDoc->WriteLine(((CDoc^)listDoc[i])->GetISBN());
     swDoc->WriteLine(((CDoc^)listDoc[i])->GetTheme());
     swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetPages()));
     swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetTotalNumber()));
     swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetFreeNumber()));
  }
// Закрити файл
  swDoc->Close();
}
/*****************************************************/
// NAME:       FindDoc
// DESCRIPTION:    Функція пошуку документа по заданих параметрах
// INPUT:      strFindDoc – рядок для пошуку
/*****************************************************/
ArrayList^ CWorker::FindDoc(String^ strFindValue)
{
// Список для результату
  ArrayList^ listResult = gcnew ArrayList();
// Шукане значення без обліку регістра
  String^ strValue = strFindValue->ToLower();
```

```
// Задана порожній рядок?
  if(String::IsNullOrEmpty(strValue->Trim()))
  {
//Вивести весь список
     for(Int32 i = 0; i < listDoc->Count; i++)
     {
// Додати документ у результуючий список
          listResult->Add(listDoc[i]);
     }
  }
  else
  {
// Вибрати з умовою
     for(Int32 i = 0; i < listDoc->Count; i++)
     {
// Пошук рядка в кожному полі без обліку регістра
if(((((CDoc^)listDoc[i])->GetName())->ToLower())->IndexOf(strValue) != -1 ||
   ((((CDoc^)listDoc[i])->GetAuthor())->ToLower())->IndexOf(strValue) != -1 ||
   ((((CDoc^)listDoc[i])->GetISBN())->ToLower())->IndexOf(strValue) != -1 ||
   ((((CDoc^)listDoc[i])->GetTheme())->ToLower())->IndexOf(strValue) != -1 ||
   Convert::ToString((((CDoc^)listDoc[i])->GetPages()))->IndexOf(strValue) != -1)
          {
// Додати документ у результуючий список
               listResult->Add(listDoc[i]);
          }
     }
  }
// Повернути список збігів
  return listResult;
}
/*****************************************************/
// NAME: ViewDoc
// DESCRIPTION:    Функція перегляду інформації про заданий документ
// INPUT: ddISBNHash – хеш вхідного номера документа, інформацію про яку треба
переглянути
/*****************************************************/
CDoc^ CWorker::ViewDoc(Int32 ddISBNHash)
{
// Визначити індекс по вхідному номеру документа (документ існує,
// тому що його вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);
// Повернути елемент "документ" із загальне списку
```

```
  return (CDoc^)listDoc[ddIndex];
}
/*****************************************************/
// NAME:       ReadLogs
// DESCRIPTION:    Функція зчитування історії з файлу
// INPUT:      N/D
/*****************************************************/
void CWorker::ReadLogs()
{
// Файл із історією надходжень існує?
  if(File::Exists(LOG_INPUT))
  {
// Зчитування історії надходжень
     logInput->strLog = File::ReadAllText(LOG_INPUT);
  }
  else
  {
     logInput->strLog = "";
  }
// Файл із історією списань існує?
  if(File::Exists(LOG_INPUT))
  {
// Зчитування історії списань
     logOutput->strLog = File::ReadAllText(LOG_OUTPUT);
  }
  else
  {
     logOutput->strLog = "";
  }
}
/*****************************************************/
// NAME:       WriteLogs
// DESCRIPTION:    Функція збереження історії у файл
// INPUT:      N/D
/*****************************************************/
void CWorker::WriteLogs()
{
// Запис історії
  File::WriteAllText(LOG_INPUT, logInput->strLog);
  File::WriteAllText(LOG_OUTPUT, logOutput->strLog);
}
/*****************************************************/
```

```
// NAME: ClearLogs
// DESCRIPTION:    Функція очищення історії
// INPUT:     N/D
/*****************************************************/
void CWorker::ClearLogs()
{
  logInput->strLog = "";
  logOutput->strLog = "";
}
/*****************************************************/
// NAME:      ViewInputLog
// DESCRIPTION:    Функція перегляду історії надходжень
// INPUT:     N/D
/*****************************************************/
String^ CWorker::ViewInputLog()
{
  return logInput->strLog;
}
/*****************************************************/
// NAME:      ViewOutputLog
// DESCRIPTION:    Функція перегляду історії списань
// INPUT:     N/D
/*****************************************************/
String^ CWorker::ViewOutputLog()
{
  return logOutput->strLog;
}
/*****************************************************/
```


4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази.Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з’являться квантові комп’ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Меню.

– Функціональні вкладки: Документи. Звіти. Історія.

– Ключові рядки для пошуку.

– Інформація про документ.

У вищезазначених розділах реалізується наступний функціонал:

– Файл.

– Довідка.

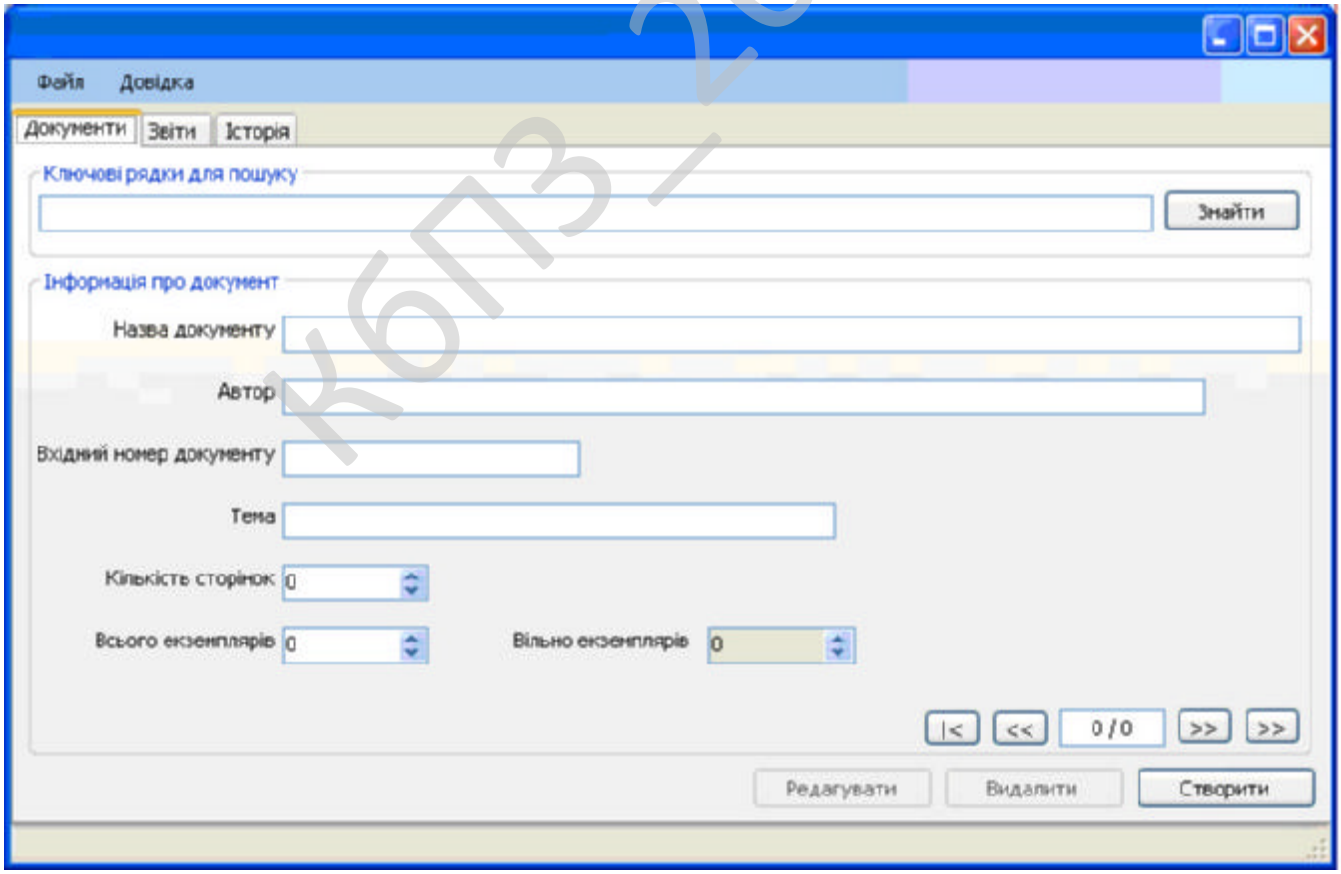

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди.

Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів.

Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм.

Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Метою розробки є дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Об'єктом дослідження є процес електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Предметом дослідження є методи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

– Удосконалено метод електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– Розроблено вітчизняний продукт електронного документообігу кафедри кібербезпеки та програмного забезпечення, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	40 (2 ост. цифри № зал*10^1)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000 (2 ост. цифри № зал*10^4)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

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

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

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

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

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

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

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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

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

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

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

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot83=168 \text{ люд/день}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3{,}8 \text{ ставки}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=567,5/(60·8)=1,2 ставки

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

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

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

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

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	10770	32310
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6700	24120
Інженер-системотехнік	0,25	6700	5025
Адміністратор мережі	0,5	6700	10050
Системний програміст	0,25	6700	5025
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6700	5025
Бухгалтер-економіст	0,5	9030	13545
Всього за період розробки	$R_{сп}$=8,25	-	$Ф_{роб}$=198300

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 400 \ грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

де $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sy – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

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

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

де Цм – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 30.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i7-4770 (4 (8) ядра по 3.4 – 3.9 GHz), 8 MB Smart Cache	-
Системна плата	Intel Q85 Express Chipset, 4x USB 3.0, 6x USB 2.0, 4x Audio, 1x VGA, 2x DisplayPort, 1x COM-порт, 1x LAN (RJ-45), 2x PS/2	-
Відеокарта	GIGABYTE GV-N730D3-1GL GeForc GT730 2 GB DDR3 64-bit D Sub+HDMI+DVI	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	16 GB DDR3	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 2.0, Mic+Audio, silver/black	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість ,шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

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

де Ne – Кількість екземплярів програм, шт.

$$З_о = 400 \cdot 209/40 = 2090 \text{ грн}$$

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

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

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=37% від суми основної та додаткової зарплати

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

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 851 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

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

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

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

Згідно виданих викладачем норм $n_{міс}$ приймаємо 0,33 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=105 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

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

$$З_{М1} = 105 \cdot 3 \cdot 0{,}33 = 105 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д., \quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн/шт.

$$З_{М2} = 41 \cdot 12 = 492 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 492 + 1702)/40 = 57 \text{ грн.}$$

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

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

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_п = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

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

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 140140 \cdot 3/(40 \cdot 12) = 876 \text{ грн}$$

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	851
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4711
9. Плановий прибуток	$П_р$	2591
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	7302
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1460,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9893

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$C_п = 2090+209+851+314+57+314+876 = 4711 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

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

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 4711 = 2591 \text{ грн.}$$

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

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.9 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	9893
Всього капітальних витрат	–	9893

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	17178	3221
2. Витрати на електроенергію	$З_{ел}$	3928	1963
3. Витрати на амортизацію	$З_{ам}$	0	2473
Всього витрат за рік	$І$	21106	7657

Витрати на оплату праці:

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

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 800 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 800 \cdot 16 \cdot 1{,}1 \cdot 1{,}22 = 17178\ грн.$$

до

$$З_{р\,нов} = 150 \cdot 16 \cdot 1{,}1 \cdot 1{,}22 = 3221\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

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

Зел баз=·0,5·2455·3,2 = 3928 грн

Зел нов=·0,5·1227·3,2 = 1963 грн

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	9893	–	2473,25
Всього відрахувань	-	–	9893	–	2473,25

7.8 Визначення економічної ефективності програмної продукції

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

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

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

$$Ев = (7302 - 4711) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 68605 \ грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(7302 - 4711) \cdot 40 \cdot 12 / 3} = 0,5 \text{ років}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4711
3. Ціна розробленої програми	Грн.	7302
4. Плановий прибуток від реалізації розробленої програми	Грн.	2591
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	103640
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	68605
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9893
11.Величина економічного ефекту у користувача програмної продукції	Грн.	10976
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,7

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (21106 - 7657) - 0{,}25 \cdot 9893 = 10975{,}75 \text{ грн.}$$

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{9893}{21106 - 7657} = 0{,}7 \text{ року}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Техніка безпеки – це система правил і заходів, які допомагають запобігти травмам, хворобам і аваріям на робочому місці або в повсякденному житті. Знання техніки безпеки дуже важливе, бо воно рятує життя і здоров'я людей. Наприклад, якщо людина працює на шахті, то вона повинненна знати правила безпеки у вугільних шахтах, щоб не потрапити під обвал або не підірватися на міні. Або якщо людина хоче навчитися програмувати, то вона повиненна дотримуватися гігієнічних вимог і не сидіти за комп'ютером занадто довго, щоб не зашкодити своїм очам і спині.

Охорона праці та здоров'я у сфері ІТ – це комплекс заходів, які спрямовані на забезпечення безпечних і здорових умов праці для працівників, які використовують інформаційні технології, а також на запобігання травматизму, професійним захворюванням і стресу.

Охорона праці та здоров'я у сфері ІТ включає такі напрями, як:

– Ергономіка – це наука про адаптацію робочого середовища до фізичних і психологічних особливостей людини[2]. Ергономіка вимагає врахування таких факторів, як розміри, форми, кольори, освітлення, шум, температура, вологість, вентиляція, постава, рухи, пози, втома, навантаження на очі та ін. Егономіка допомагає покращити комфорт, продуктивність і задоволення працівників.

– Комп'ютерна безпека – це захист комп'ютерних систем і даних від несанкціонованого доступу, зміни, знищення або блокування. Комп'ютерна безпека вимагає використання антивірусних програм, фаєрволів, паролей, шифрування, резервного копіювання та інших технологій. Комп'ютерна безпека допомагає запобігти крадіжці, шпигунству, шантажу, саботажу та іншим загрозам.

– Соціальна взаємодія – це процес спілкування між людьми у робочому колективі або через мережеві сервіси. Соціальна взаємодія вимагає дотримання правил етикету, поваги, толерантності, співробітництва та конструктивного діалогу. Соціальна взаємодія допомагає покращити настрій, мотивацію, комунікацію та творчий потенціал працівників.

Правила охорони праці і здоров'я для програмістів:

- Регулярно роби перерви в роботі. Вставай із-за столу і розминай м'язи.
- Налаштуй яскравість і контрастність монітору так, щоб не напружувати очі.
- Використовуй ергономічну мишку і клавіатуру, які зручно лягають у руку і не виклкають болю.
- Слідкуй за своєю поставою. Сиди прямо і не нахиляйся до екрану.
- Захищай свій комп'ютер від вірусів, шпигунських програм і хакерів. Оновлюй ативірусне програмне забезпечення і не відкривай підозрілі файли і посилання.
- Не забувай про соціальну взаємодію. Спілкуйся з колегами, друзями і родиною. – Відвідуй заходи, які тебе цікавлять. Не ізолюй себе від світу.
- Люби свою професію, але не забувай про інші сфери життя. Розвивай свої захопленя, хоббі і таланти. Знаходь рівновагу між роботою і відпочинком.

Закон України “Про охорону праці” визначає основні принципи, завдання, права і обов’язки суб’єктів відносин з охорони праці, а також організаційні та правові основи державного управління і контролю за дотриманням законодавства про охорну праці.

Згідно з цим законом, ІТ компанії повинні впроваджувати такі заходи з охорни праці:

- Створювати на підприємстві службу охорони праці або призначати відповідальних осіб, які забезпечують розроблення, реалізацію та контроль за дотриманням заходів з охорони праці.

- Забезпечувати безпечні і нешкідливі умови праці для працівників, використвючи сучасні засоби техніки безпеки, санітарно-гігієнічні умови, засоби клектинго та індивідуального захисту, оптимальні режими праці та відпочинку.

- Проводити атестацію робочих місць на відповідність нормативно-правовим атам з охорони праці та аудит з охорони праці.

- Проводити навчання та інструктаж з питань охорони праці, з надання першої медичної допомоги потерпілим від нещасних випадків і правил поведінки у разі виникнення аварії[5].

- Забезпечувати лікувально-профілактичне обслуговування працюючих, санітарно-побутове обслуговування, пільги і компенсації для працівників, які працюють у важких і шкідливих умовах.

- Нести відповідальність за порушення законодавства про охорону праці та зподіяння шкоди життю і здоров'ю працівників.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальні машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,38
Довжина	5,95
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	м2	не менше 6.0	8
Об'єм, V	м3	не менше 20.0	22,4

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація.

Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості

руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018, у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018, можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта

природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Працівники повинні дотримуватися статті 18 Закону України ”Про охорону праці” згідно цій статті працівники зобов'язані:

знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу;

дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства.

Для розробки заходів з умов поліпшення охорони праці ІТ спеціалістів потрібно виконати такі кроки:

Провести аналіз стану охорони праці на робочих місцях ІТ спеціалістів за допомогою спеціальних приладів або мобільних додатків. Визначити рівень освітленості, мікроклімату, електромагнітного випромінювання, нервово-

емоційного напруження, навантаження на органи зору, дрібні стереостатичні рухи кінцівок.

Порівняти отримані показники з нормативно-правовими актами з охорони праці, такими як ДСанПІН 3.3.2.007-98 [2], НПАОП 0.00-4.12-05, НПАОП 0.00-1.28-10 та іншими.

Виявити виробничі фактори, які не відповідають нормам і стандартам безпеки праці, і визначити їх вплив на здоров'я і продуктивність ІТ спеціалістів.

Розробити план комплексних заходів з охорони праці, який повинен містити такі елементи: цілі і завдання, терміни і виконавці, необхідні ресурси і кошторис, очікувані результати і критерії оцінки ефективності.

Запровадити заходи з охорони праці на робочих місях ІТ спеціалістiv, такі як: покращення освітлення, вентиляції і кондиціонування повітря, захист від електромагнітного випромінювання, зменшення нервово-емоціонного напруження, оптимізаціа режиму роботы і відпочинку, користувача засобами захисту органів зору та кінцівок.

Провести мониторинг і контроль за дотриманням заходiv з охорони праці на робочих місях ІТ спеціалістів за допомогою спеціальних приладів.

8.5 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 2,6 м, довжина – 2,6 м, висота – 3 м.

У зазначеному приміщенні працює 1 особа.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де: F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=2,6 х 2,6 = 6,76 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5 [6]);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін.}$)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: S – площа приміщення, S = 6,76 м^2;

h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 2,6 м;

B – довжина приміщення, B = 2,6 м.

Підставимо всі значення у формулу та визначемо індекса приміщення: i=0,43.

Знаючи індекс приміщення, за знаходимо n = 0,23 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначемо світловий потік: F=14548 Лм.

Будемо використовувати світлодіодні панелі Lebron L-LPU-Prismatic, світловий потік яких $F_л$ = 3000 Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: N= 14548/ 3000=4,8 шт.

Приймаємо необхідну кількість ламп 5 шт.

Висновки до розділу

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

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

4. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

5. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99**.** – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

6. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

7. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

8. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– Досліджена система електронного документообігу кафедри кібербезпеки та програмного забезпечення.

– На основі отриманих результатів досліджень створена програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання електронного документообігу кафедри кібербезпеки та програмного забезпечення.

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

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

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

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 9041:2020.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 10976 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горяний Д.В. Дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p
3. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.
4. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.
5. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.
6. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.
7. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
8. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
9. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.
10. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
11. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.
12. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

13. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

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

15. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

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

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

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

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

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

21. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-

quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

24. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

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

26. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

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

28. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and

Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

29. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

30. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

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

32. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

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

34. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та

комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

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

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

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

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

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

40. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

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

42. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

43. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

44. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

45. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces "Alexandru cel Bun". 2017. P7.

46. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA – 2017.

47. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

48. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп'ютерно-інтегровані

технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

49. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

50. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). – Харків: ХУПС. – 2016. – С.96-100.

51. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". – Випуск 8 (145). – Х.: ХУПС – 2016. – С. 77-80.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 37-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи електронного документообігу кафедри кібербезпеки та програмного забезпечення;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 97 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 15.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Смірнов О.А.

Дослідження та програмна реалізація системи електронного документообігу кафедри кібербезпеки та програмного забезпечення

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 71

Літера: РП

Кропивницький – 2023 року

Файл Worker.cpp - вікно адміністратора

```
  /*****************************************************************************/
#include "StdAfx.h"
#include "Worker.h"
#include "Doc.h"
#include "Logger.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Collections;
/*****************************************************************************/
#define DOCS_FILE "DATA.LMB"
#define LOG_INPUT "INPUT.LML"
#define LOG_OUTPUT      "OUTPUT.LML"
/*****************************************************************************/
// NAME:          CWorker
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
/*****************************************************************************/
CWorker::CWorker(void)
{
  listDoc = gcnew ArrayList();
  logInput = gcnew CLogger();
  logOutput = gcnew CLogger();
}
/*****************************************************************************/
// NAME:          GetIndexByISBNHash
// DESCRIPTION:   Функція одержання індексу документа в загальному списку по
вхідному номеру документа
// INPUT:         ddHashValue - значення хеша вхідного номера документа
/*****************************************************************************/
Int32 CWorker::GetIndexByISBNHash(Int32 HashValue)
{
  // Цикл пошуку документа із заданим вхідним номером документа
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
      // Хеш-значення вхідного номера документа заданого документа збігається з
      // хеш-     значенням вхідного номера документа знайденого документа?
      if(HashValue == ((CDoc^)listDoc[i])->GetISBNHash())
      {
            // Документ знайдений, повернути індекс
            return i;
      }
  }

  // Документ не знайдений, повернути -1
  return -1;
}
/*****************************************************************************/
// NAME:          AddDoc
// DESCRIPTION:   Функція додавання нового документа
// INPUT:         SourceDoc - об'єкт "документ" для включення в список
// OUTPUT:        TRUE - документ успішно доданий
//                FALSE - такий документ уже існує в системі
/*****************************************************************************/
Boolean CWorker::AddDoc(CDoc^ SourceDoc)
{
  // Документи з таким вхідним номером документа не існує?
  if(this->GetIndexByISBNHash(SourceDoc->GetISBNHash()) == -1)
  {
      // Додати заданній документ у загальний список
      listDoc->Add(SourceDoc);
      // Записати подія в лог
      logInput->WriteEvent(
            "Новий документ '" + ((CDoc^)SourceDoc)->GetName() +
            "', Автор '" + ((CDoc^)SourceDoc)->GetAuthor() +
```

```
            "', Вхідний номер документа " + ((CDoc^)SourceDoc)->GetISBN());
      // Функція відробила успішно
      return true;
  }
  else
  {
      // Документ із таким же вхідним номером документа існує, повернути помилку
      return false;
  }
}
/*************************************************************************/
// NAME:          RemoveDoc
// DESCRIPTION:   Функція видалення документів із системи
// INPUT:         ddISBNHash - значення хеша вхідного номера документа
// OUTPUT:              0 - видалення зроблене
//                      1 - видалення неможливо, не всі екземпляри перебувають у
системі
//                      2 - документ із заданим вхідним номером документа не
знайдений
/*************************************************************************/
Int32 CWorker::RemoveDoc(Int32 ddISBNHash)
{
  // Знайти документ по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Документ із таким вхідним номером документа існує?
  if(ddIndex != -1)
  {
      // Перевірити, що жодного документа немає в користувача
      if(((CDoc^)listDoc[ddIndex])->GetTotalNumber() ==
            ((CDoc^)listDoc[ddIndex])->GetFreeNumber())
      {

            // Записати подія в лог
            logOutput->WriteEvent(
                  "Виконаний документ '" + ((CDoc^)listDoc[ddIndex])->GetName()
+
                  "', Автор '" + ((CDoc^)listDoc[ddIndex])->GetAuthor() +
                  "', Вхідний номер документа " + ((CDoc^)listDoc[ddIndex])-
>GetISBN());
            // Видалити знайдений документ зі списку
            listDoc->RemoveAt(ddIndex);
            // Функція відробила успішно
            return 0;
      }
      else
      {
            // Повернути помилку, не всі документи перебувають у системі
            return 1;
      }
  }
  else
  {
      // Повернути помилку, документ із таким вхідним номером документа не був
знайдений
      return 2;
  }
}
/*************************************************************************/
// NAME:          GiveDoc
// DESCRIPTION:   Функція видачі документів користувачеві
// INPUT:         listDoc - список документів користувача (для виконання
додавання)
//                      ddISBNHash - хеш вхідного номера документа, що потрібно
одержати
// OUTPUT:        TRUE - операція пройшла успішно
//                      FALSE - у наявності немає жодного екземпляра заданого
документа
/*************************************************************************/
```

```
Boolean CWorker::GiveDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Зменшити кількість вільних документів
  if((ddIndex != -1) && ((CDoc^)listDoc[ddIndex])->DecFreeNumber())
  {
      // Додати документ у список користувача
      listReader->Add(listDoc[ddIndex]);
      // Функція відробила успішно
      return true;
  }
  else
  {
      // Такого документа не є в наявності
      return false;
  }
}
/*************************************************************************/
// NAME:          TakeDoc
// DESCRIPTION:   Функція прийняття документів від користувача назад у систему
// INPUT:         // listDoc - список документів користувача (для виконання
видалення)
//                      ddISBNHash - хеш вхідного номера документа, що потрібно
повернути
/*************************************************************************/
void CWorker::TakeDoc(ArrayList^ listReader, Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Видалити документ зі списку користувача
  // listReader->RemoveAt(ddIndex);

  // Збільшити кількість вільних документів
  ((CDoc^)listDoc[ddIndex])->IncFreeNumber();
}
/*************************************************************************/
// NAME:          LoadDocList
// DESCRIPTION:   Функція завантаження документів системи з файлу
// INPUT:         N/D
/*************************************************************************/
void CWorker::LoadDocList()
{
  // Перевірити існування файлу
  if(!File::Exists(DOCS_FILE))
  {
      // Файл не знайдений, закінчити виконання функції
      return;
  }

  // Відкрити файл
  StreamReader^ srDoc = gcnew StreamReader(DOCS_FILE);

  // Продовжувати, поки не буде знайдений кінець файлу
  while(srDoc->EndOfStream == false)
  {
      // Створити новий об'єкт "документ"
      CDoc^ NewDoc = gcnew CDoc();

      // Завантажити з файлу й установити параметри документа
      NewDoc->SetName(srDoc->ReadLine());
      NewDoc->SetAuthor(srDoc->ReadLine());
      NewDoc->SetISBN(srDoc->ReadLine());
      NewDoc->SetTheme(srDoc->ReadLine());
      NewDoc->SetPages(Convert::ToInt32(srDoc->ReadLine()));
      NewDoc->SetTotalNumber(Convert::ToInt32(srDoc->ReadLine()));
```

```
      NewDoc->SetFreeNumber(Convert::ToInt32(srDoc->ReadLine()));

      // Включити документ у загальний список документів
      listDoc->Add(NewDoc);
  }

  // Закрити файл
  srDoc->Close();
}
/*************************************************************************/
// NAME:          SaveDocList
// DESCRIPTION:   Функція збереження документів системи у файл
// INPUT:         N/D
/*************************************************************************/
void CWorker::SaveDocList()
{
  // Перевірити існування файлу
  if(File::Exists(DOCS_FILE))
  {
      // Видалити файл
      File::Delete(DOCS_FILE);
  }

  // Створити й відкрити файл
  StreamWriter^ swDoc = gcnew StreamWriter(DOCS_FILE);

  // Цикл запису по одному документу
  for(Int32 i = 0; i < listDoc->Count; i++)
  {
      // Одержати параметри документа й записати їх у файл
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetName());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetAuthor());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetISBN());
      swDoc->WriteLine(((CDoc^)listDoc[i])->GetTheme());
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetPages()));
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])-
>GetTotalNumber()));
      swDoc->WriteLine(Convert::ToString(((CDoc^)listDoc[i])->GetFreeNumber()));
  }

  // Закрити файл
  swDoc->Close();
}
/*************************************************************************/
// NAME:          FindDoc
// DESCRIPTION:   Функція пошуку документа по заданих параметрах
// INPUT:         strFindDoc - рядок для пошуку
/*************************************************************************/
ArrayList^ CWorker::FindDoc(String^ strFindValue)
{
  // Список для результату
  ArrayList^ listResult = gcnew ArrayList();

  // Шукане значення без обліку регістра
  String^ strValue = strFindValue->ToLower();

  // Задана порожній рядок?
  if(String::IsNullOrEmpty(strValue->Trim()))
  {
      //Вивести весь список
      for(Int32 i = 0; i < listDoc->Count; i++)
      {
            // Додати документ у результуючий список
            listResult->Add(listDoc[i]);
      }
  }
  else
  {
      // Вибрати з умовою
```

```
      for(Int32 i = 0; i < listDoc->Count; i++)
      {
            // Пошук рядка в кожному полі без обліку регістра
            if(((((CDoc^)listDoc[i])->GetName())->ToLower())->IndexOf(strValue)
!= -1 ||
               ((((CDoc^)listDoc[i])->GetAuthor())->ToLower())-
>IndexOf(strValue) != -1 ||
               ((((CDoc^)listDoc[i])->GetISBN())->ToLower())->IndexOf(strValue)
!= -1 ||
               ((((CDoc^)listDoc[i])->GetTheme())->ToLower())->IndexOf(strValue)
!= -1 ||
              Convert::ToString((((CDoc^)listDoc[i])->GetPages()))-
>IndexOf(strValue) != -1)
            {
                  // Додати документ у результуючий список
                  listResult->Add(listDoc[i]);
            }
      }
  }

  // Повернути список збігів
  return listResult;
}
/*************************************************************************/
// NAME:          ViewDoc
// DESCRIPTION:   Функція перегляду інформації про заданий документ
// INPUT:         ddISBNHash - хеш вхідного номера документа, інформацію про яку
треба переглянути
/*************************************************************************/
CDoc^ CWorker::ViewDoc(Int32 ddISBNHash)
{
  // Визначити індекс по вхідному номеру документа (документ існує, тому що його
вхідний номер документа був повідомлений користувачеві)
  Int32 ddIndex = this->GetIndexByISBNHash(ddISBNHash);

  // Повернути елемент "документ" із загальне списку
  return (CDoc^)listDoc[ddIndex];
}
/*************************************************************************/
// NAME:          ReadLogs
// DESCRIPTION:   Функція зчитування історії з файлу
// INPUT:         N/D
/*************************************************************************/
void CWorker::ReadLogs()
{
  // Файл із історією надходжень існує?
  if(File::Exists(LOG_INPUT))
  {
      // Зчитування історії надходжень
      logInput->strLog = File::ReadAllText(LOG_INPUT);
  }
  else
  {
      logInput->strLog = "";
  }

  // Файл із історією списань існує?
  if(File::Exists(LOG_INPUT))
  {
      // Зчитування історії списань
      logOutput->strLog = File::ReadAllText(LOG_OUTPUT);
  }
  else
  {
      logOutput->strLog = "";
  }
}
/*************************************************************************/
// NAME:          WriteLogs
```

```
// DESCRIPTION:   Функція збереження історії у файл
// INPUT:         N/D
/*****************************************************************************/
void CWorker::WriteLogs()
{
  // Запис історії
  File::WriteAllText(LOG_INPUT, logInput->strLog);
  File::WriteAllText(LOG_OUTPUT, logOutput->strLog);
}
/*****************************************************************************/
// NAME:          ClearLogs
// DESCRIPTION:   Функція очищення історії
// INPUT:         N/D
/*****************************************************************************/
void CWorker::ClearLogs()
{
  logInput->strLog = "";
  logOutput->strLog = "";
}
/*****************************************************************************/
// NAME:          ViewInputLog
// DESCRIPTION:   Функція перегляду історії надходжень
// INPUT:         N/D
/*****************************************************************************/
String^ CWorker::ViewInputLog()
{
  return logInput->strLog;
}
/*****************************************************************************/
// NAME:          ViewOutputLog
// DESCRIPTION:   Функція перегляду історії списань
// INPUT:         N/D
/*****************************************************************************/
String^ CWorker::ViewOutputLog()
{
  return logOutput->strLog;
}
/*****************************************************************************/
```

Файл Worker.resx - xml опис вікна адміністратора

```
<?xml version="1.0" encoding=" utf-8"?>
<root>
  <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
```

```
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <metadata name="menuStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>230, 17</value>
  </metadata>
  <metadata name="statusStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>332, 17</value>
  </metadata>
  <metadata name="groupInfo.Locked" type="System.Boolean, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <value>True</value>
  </metadata>
  <metadata name="groupInfo.Locked" type="System.Boolean, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
    <value>True</value>
  </metadata>
</root>
```

Файл Worker.h - бібліотека для файлу frmAbout.cpp

```
    #pragma once
/*************************************************************************/
#include "Worker.h"
#include "Doc.h"
#include "Counter.h"
#include "frmAbout.h"
/*************************************************************************/
#define PAGES_LESS                          "менше"
#define PAGES_GREATER                       "більше"
#define PAGES_EQUAL                         "дорівнює"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
namespace UsrsManager
{
    public ref class frmWorker : public System::Windows::Forms::Form
    {
/*************************************************************************/
    private:
      CWorker^ Worker;              // Об'єкт "Адміністратор"

      Boolean IsCreateClicked;      // Прапор натискання по кнопці "Створити"
      Boolean IsEditClicked;        // Прапор натискання по кнопці "Редагувати"

      ArrayList^ listView;          // Поточний список документів для
відображення

      Int32 ddCurrentIndex;         // Індекс поточного відображуваного елемента
/*************************************************************************/
private: System::Windows::Forms::Button^  btnClearLogs;
/*************************************************************************/
public:     frmWorker(void)
      {
            // Ініціалізація компонентів форми
            InitializeComponent();

            // Кнопки "Створити" і "Редагувати" не минулого натиснуті жодного
разу
            IsCreateClicked = false;
            IsEditClicked = false;

            // Створити список відображуваних документів
            listView = gcnew ArrayList();

            // Поточний індекс документа для відображення не визначений
            ddCurrentIndex = -1;
      }
/*************************************************************************/
protected: ~frmWorker()
      {
            if (components)
            {
                  delete components;
            }
      }
/*************************************************************************/
    private: System::Windows::Forms::ListBox^  lstPagesDocList;
    private: System::Windows::Forms::NumericUpDown^  nmrPagesNumber;
    private: System::Windows::Forms::ComboBox^  cmbPagesDirect;
    private: System::Windows::Forms::ListBox^  lstThemeDocList;
```

```
    private: System::Windows::Forms::ListBox^  lstAuthorDocList;
    private: System::Windows::Forms::CheckBox^  chkPagesFlag;
    private: System::Windows::Forms::MenuStrip^  menuStrip;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFile;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFileSave;
    private: System::Windows::Forms::ToolStripMenuItem^  menuFileExit;
    private: System::Windows::Forms::ToolStripMenuItem^  menuHelp;

    private: System::Windows::Forms::ToolStripMenuItem^  menuHelpAbout;
    private: System::Windows::Forms::GroupBox^  groupBox2;
    private: System::Windows::Forms::TextBox^  txtThemeFilter;
    private: System::Windows::Forms::TextBox^  txtAuthorFilter;
    private: System::Windows::Forms::Label^  lblFreeInstanceNumber;
    private: System::Windows::Forms::Label^  lblTotalInstanceNumber;
    private: System::Windows::Forms::CheckBox^  chkThemeDocFlag;
    private: System::Windows::Forms::CheckBox^  chkAuthorDocFlag;
    private: System::Windows::Forms::CheckBox^  chkFreeInstanceFlag;
    private: System::Windows::Forms::CheckBox^  chkTotalInstaceFlag;
    private: System::Windows::Forms::NumericUpDown^  nmrFreeNumber;
    private: System::Windows::Forms::Button^  btnRefresh;
    private: System::Windows::Forms::NumericUpDown^  nmrTotalNumber;
    private: System::Windows::Forms::TabPage^  tabHistory;
    private: System::Windows::Forms::GroupBox^  groupOutput;
    private: System::Windows::Forms::TextBox^  txtOutputHistory;
    private: System::Windows::Forms::GroupBox^  groupInput;
    private: System::Windows::Forms::TextBox^  txtInputHistory;
    private: System::Windows::Forms::NumericUpDown^  nmrPageNumber;
    private: System::Windows::Forms::TextBox^  txtTheme;
    private: System::Windows::Forms::TextBox^  txtISBN;
    private: System::Windows::Forms::TextBox^  txtAuthor;
    private: System::Windows::Forms::TextBox^  txtDocName;
    private: System::Windows::Forms::TabPage^  tabReports;
    private: System::Windows::Forms::Label^  lblFreeNumber;
    private: System::Windows::Forms::Label^  lblTotalNumber;
    private: System::Windows::Forms::Label^  lblPagesNumber;
    private: System::Windows::Forms::Label^  lblTheme;
    private: System::Windows::Forms::StatusStrip^  statusStrip;
    private: System::Windows::Forms::ToolStripStatusLabel^  lblStatus;
    private: System::Windows::Forms::TextBox^  txtFindString;
    private: System::Windows::Forms::TabControl^  tabControl;
    private: System::Windows::Forms::TabPage^  tabDocs;
    private: System::Windows::Forms::Button^  btnEdit;
    private: System::Windows::Forms::Button^  btnDelete;
    private: System::Windows::Forms::Button^  btnCreate;
    private: System::Windows::Forms::GroupBox^  groupInfo;
    private: System::Windows::Forms::Label^  lblISBN;
    private: System::Windows::Forms::Label^  lblAuthor;
    private: System::Windows::Forms::Label^  lblDocName;
    private: System::Windows::Forms::GroupBox^  groupFinding;
    private: System::Windows::Forms::Button^  btnFind;
    private: System::Windows::Forms::Button^  btnFirst;
    private: System::Windows::Forms::Button^  btnPrev;
    private: System::Windows::Forms::Button^  btnNext;
    private: System::Windows::Forms::Button^  btnLast;
    private: System::Windows::Forms::TextBox^  txtPosition;
    private: System::ComponentModel::Container ^components;
/*****************************************************************************/
#pragma region Windows Form Designer generated code
      /// <summary>
      /// Необхідний метод для підтримки Розроблювача - не модифікувати
      /// зміст цього методу з кодовим редактором.
      /// </summary>
      void InitializeComponent(void)
      {
            this->lstPagesDocList = (gcnew System::Windows::Forms::ListBox());
            this->nmrPagesNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->cmbPagesDirect = (gcnew System::Windows::Forms::ComboBox());
            this->lstThemeDocList = (gcnew System::Windows::Forms::ListBox());
```

```
            this->lstAuthorDocList = (gcnew System::Windows::Forms::ListBox());
            this->chkPagesFlag = (gcnew System::Windows::Forms::CheckBox());
            this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
            this->menuFile = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileSave = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileExit = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelp = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelpAbout = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
            this->txtThemeFilter = (gcnew System::Windows::Forms::TextBox());
            this->txtAuthorFilter = (gcnew System::Windows::Forms::TextBox());
            this->lblFreeInstanceNumber = (gcnew
System::Windows::Forms::Label());
            this->lblTotalInstanceNumber = (gcnew
System::Windows::Forms::Label());
            this->chkThemeDocFlag = (gcnew System::Windows::Forms::CheckBox());
            this->chkAuthorDocFlag = (gcnew System::Windows::Forms::CheckBox());
            this->chkFreeInstanceFlag = (gcnew
System::Windows::Forms::CheckBox());
            this->chkTotalInstaceFlag = (gcnew
System::Windows::Forms::CheckBox());
            this->nmrFreeNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->btnRefresh = (gcnew System::Windows::Forms::Button());
            this->nmrTotalNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->tabHistory = (gcnew System::Windows::Forms::TabPage());
            this->btnClearLogs = (gcnew System::Windows::Forms::Button());
            this->groupOutput = (gcnew System::Windows::Forms::GroupBox());
            this->txtOutputHistory = (gcnew System::Windows::Forms::TextBox());
            this->groupInput = (gcnew System::Windows::Forms::GroupBox());
            this->txtInputHistory = (gcnew System::Windows::Forms::TextBox());
            this->nmrPageNumber = (gcnew
System::Windows::Forms::NumericUpDown());
            this->txtTheme = (gcnew System::Windows::Forms::TextBox());
            this->txtISBN = (gcnew System::Windows::Forms::TextBox());
            this->txtAuthor = (gcnew System::Windows::Forms::TextBox());
            this->txtDocName = (gcnew System::Windows::Forms::TextBox());
            this->tabReports = (gcnew System::Windows::Forms::TabPage());
            this->lblFreeNumber = (gcnew System::Windows::Forms::Label());
            this->lblTotalNumber = (gcnew System::Windows::Forms::Label());
            this->lblPagesNumber = (gcnew System::Windows::Forms::Label());
            this->lblTheme = (gcnew System::Windows::Forms::Label());
            this->statusStrip = (gcnew System::Windows::Forms::StatusStrip());
            this->lblStatus = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->txtFindString = (gcnew System::Windows::Forms::TextBox());
            this->tabControl = (gcnew System::Windows::Forms::TabControl());
            this->tabDocs = (gcnew System::Windows::Forms::TabPage());
            this->btnEdit = (gcnew System::Windows::Forms::Button());
            this->btnDelete = (gcnew System::Windows::Forms::Button());
            this->btnCreate = (gcnew System::Windows::Forms::Button());
            this->groupInfo = (gcnew System::Windows::Forms::GroupBox());
            this->txtPosition = (gcnew System::Windows::Forms::TextBox());
            this->btnFirst = (gcnew System::Windows::Forms::Button());
            this->btnPrev = (gcnew System::Windows::Forms::Button());
            this->btnNext = (gcnew System::Windows::Forms::Button());
            this->btnLast = (gcnew System::Windows::Forms::Button());
            this->lblISBN = (gcnew System::Windows::Forms::Label());
            this->lblAuthor = (gcnew System::Windows::Forms::Label());
            this->lblDocName = (gcnew System::Windows::Forms::Label());
            this->groupFinding = (gcnew System::Windows::Forms::GroupBox());
            this->btnFind = (gcnew System::Windows::Forms::Button());
```

```
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPagesNumber))->BeginInit();
            this->menuStrip->SuspendLayout();
            this->groupBox 2-2->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrFreeNumber))->BeginInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrTotalNumber))->BeginInit();
            this->tabHistory->SuspendLayout();
            this->groupOutput->SuspendLayout();
            this->groupInput->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPageNumber))->BeginInit();
            this->tabReports->SuspendLayout();
            this->statusStrip->SuspendLayout();
            this->tabControl->SuspendLayout();
            this->tabDocs->SuspendLayout();
            this->groupInfo->SuspendLayout();
            this->groupFinding->SuspendLayout();
            this->SuspendLayout();
            //
            // lstPagesDocList
            //
            this->lstPagesDocList->FormattingEnabled = true;
            this->lstPagesDocList->HorizontalScrollbar = true;
            this->lstPagesDocList->Location = System::Drawing::Point(478, 117);
            this->lstPagesDocList->Name = L"lstPagesDocList";
            this->lstPagesDocList->Size = System::Drawing::Size(223, 199);
            this->lstPagesDocList->TabIndex = 13;
            //
            // nmrPagesNumber
            //
            this->nmrPagesNumber->Location = System::Drawing::Point(583, 89);
            this->nmrPagesNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrPagesNumber->Name = L"nmrPagesNumber";
            this->nmrPagesNumber->Size = System::Drawing::Size(64, 20);
            this->nmrPagesNumber->TabIndex = 6;
            //
            // cmbPagesDirect
            //
            this->cmbPagesDirect->FormattingEnabled = true;
            this->cmbPagesDirect->Location = System::Drawing::Point(479, 89);
            this->cmbPagesDirect->Name = L"cmbPagesDirect";
            this->cmbPagesDirect->Size = System::Drawing::Size(89, 21);
            this->cmbPagesDirect->TabIndex = 5;
            //
            // lstThemeDocList
            //
            this->lstThemeDocList->FormattingEnabled = true;
            this->lstThemeDocList->HorizontalScrollbar = true;
            this->lstThemeDocList->Location = System::Drawing::Point(242, 117);
            this->lstThemeDocList->Name = L"lstThemeDocList";
            this->lstThemeDocList->Size = System::Drawing::Size(223, 199);
            this->lstThemeDocList->TabIndex = 9;
            //
            // lstAuthorDocList
            //
            this->lstAuthorDocList->FormattingEnabled = true;
            this->lstAuthorDocList->HorizontalScrollbar = true;
            this->lstAuthorDocList->Location = System::Drawing::Point(6, 117);
            this->lstAuthorDocList->Name = L"lstAuthorDocList";
            this->lstAuthorDocList->Size = System::Drawing::Size(223, 199);
            this->lstAuthorDocList->TabIndex = 8;
            //
            // chkPagesFlag
            //
            this->chkPagesFlag->AutoSize = true;
            this->chkPagesFlag->Location = System::Drawing::Point(479, 68);
```

```
            this->chkPagesFlag->Name = L"chkPagesFlag";
            this->chkPagesFlag->Size = System::Drawing::Size(75, 17);
            this->chkPagesFlag->TabIndex = 4;
            this->chkPagesFlag->Text = L"Сторінок:";
            this->chkPagesFlag->UseVisualStyleBackColor = true;
            //
            // menuStrip
            //
            this->menuStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFile, this->menuHelp});
            this->menuStrip->Location = System::Drawing::Point(0, 0);
            this->menuStrip->Name = L"menuStrip";
            this->menuStrip->Size = System::Drawing::Size(722, 24);
            this->menuStrip->TabIndex = 9;
            this->menuStrip->Text = L"menuStrip1";
            //
            // menuFile
            //
            this->menuFile->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFileSave,
                this->menuFileExit});
            this->menuFile->Name = L"menuFile";
            this->menuFile->Size = System::Drawing::Size(45, 20);
            this->menuFile->Text = L"Файл";
            //
            // menuFileSave
            //
            this->menuFileSave->Name = L"menuFileSave";
            this->menuFileSave->Size = System::Drawing::Size(152, 22);
            this->menuFileSave->Text = L"Зберегти";
            this->menuFileSave->Click += gcnew System::EventHandler(this,
&frmWorker::menuFileSave_Click);
            //
            // menuFileExit
            //
            this->menuFileExit->Name = L"menuFileExit";
            this->menuFileExit->Size = System::Drawing::Size(152, 22);
            this->menuFileExit->Text = L"Вихід";
            this->menuFileExit->Click += gcnew System::EventHandler(this,
&frmWorker::menuFileExit_Click);
            //
            // menuHelp
            //
            this->menuHelp->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->menuHelpAbout});
            this->menuHelp->Name = L"menuHelp";
            this->menuHelp->Size = System::Drawing::Size(60, 20);
            this->menuHelp->Text = L"Довідка";
            //
            // menuHelpAbout
            //
            this->menuHelpAbout->Name = L"menuHelpAbout";
            this->menuHelpAbout->Size = System::Drawing::Size(166, 22);
            this->menuHelpAbout->Text = L"Про програму...";
            this->menuHelpAbout->Click += gcnew System::EventHandler(this,
&frmWorker::menuHelpAbout_Click);
            //
            // groupBox2
            //
            this->groupBox 2-2->Controls->Add(this->lstPagesDocList);
            this->groupBox 2-2->Controls->Add(this->nmrPagesNumber);
            this->groupBox 2-2->Controls->Add(this->cmbPagesDirect);
            this->groupBox 2-2->Controls->Add(this->chkPagesFlag);
            this->groupBox 2-2->Controls->Add(this->lstThemeDocList);
            this->groupBox 2-2->Controls->Add(this->lstAuthorDocList);
            this->groupBox 2-2->Controls->Add(this->txtThemeFilter);
            this->groupBox 2-2->Controls->Add(this->txtAuthorFilter);
            this->groupBox 2-2->Controls->Add(this->lblFreeInstanceNumber);
            this->groupBox 2-2->Controls->Add(this->lblTotalInstanceNumber);
```

```
            this->groupBox 2-2->Controls->Add(this->chkThemeDocFlag);
            this->groupBox 2-2->Controls->Add(this->chkAuthorDocFlag);
            this->groupBox 2-2->Controls->Add(this->chkFreeInstanceFlag);
            this->groupBox 2-2->Controls->Add(this->chkTotalInstaceFlag);
            this->groupBox 2-2->Location = System::Drawing::Point(8, 6);
            this->groupBox 2-2->Name = L"groupBox2";
            this->groupBox 2-2->Size = System::Drawing::Size(712, 325);
            this->groupBox 2-2->TabIndex = 0;
            this->groupBox 2-2->TabStop = false;
            //
            // txtThemeFilter
            //
            this->txtThemeFilter->Location = System::Drawing::Point(242, 90);
            this->txtThemeFilter->Name = L"txtThemeFilter";
            this->txtThemeFilter->Size = System::Drawing::Size(223, 20);
            this->txtThemeFilter->TabIndex = 7;
            //
            // txtAuthorFilter
            //
            this->txtAuthorFilter->Location = System::Drawing::Point(6, 90);
            this->txtAuthorFilter->Name = L"txtAuthorFilter";
            this->txtAuthorFilter->Size = System::Drawing::Size(223, 20);
            this->txtAuthorFilter->TabIndex = 6;
            //
            // lblFreeInstanceNumber
            //
            this->lblFreeInstanceNumber->AutoSize = true;
            this->lblFreeInstanceNumber->Location = System::Drawing::Point(476,
42);
            this->lblFreeInstanceNumber->Name = L"lblFreeInstanceNumber";
            this->lblFreeInstanceNumber->Size = System::Drawing::Size(26, 13);
            this->lblFreeInstanceNumber->TabIndex = 5;
            this->lblFreeInstanceNumber->Text = L"Н/Д";
            //
            // lblTotalInstanceNumber
            //
            this->lblTotalInstanceNumber->AutoSize = true;
            this->lblTotalInstanceNumber->Location = System::Drawing::Point(476,
19);
            this->lblTotalInstanceNumber->Name = L"lblTotalInstanceNumber";
            this->lblTotalInstanceNumber->Size = System::Drawing::Size(26, 13);
            this->lblTotalInstanceNumber->TabIndex = 4;
            this->lblTotalInstanceNumber->Text = L"Н/Д";
            //
            // chkThemeDocFlag
            //
            this->chkThemeDocFlag->AutoSize = true;
            this->chkThemeDocFlag->Location = System::Drawing::Point(240, 67);
            this->chkThemeDocFlag->Name = L"chkThemeDocFlag";
            this->chkThemeDocFlag->Size = System::Drawing::Size(176, 17);
            this->chkThemeDocFlag->TabIndex = 3;
            this->chkThemeDocFlag->Text = L"Список документів за темою:";
            this->chkThemeDocFlag->UseVisualStyleBackColor = true;
            //
            // chkAuthorDocFlag
            //
            this->chkAuthorDocFlag->AutoSize = true;
            this->chkAuthorDocFlag->Location = System::Drawing::Point(6, 67);
            this->chkAuthorDocFlag->Name = L"chkAuthorDocFlag";
            this->chkAuthorDocFlag->Size = System::Drawing::Size(165, 17);
            this->chkAuthorDocFlag->TabIndex = 2;
            this->chkAuthorDocFlag->Text = L"Список документів автора:";
            this->chkAuthorDocFlag->UseVisualStyleBackColor = true;
            //
            // chkFreeInstanceFlag
            //
            this->chkFreeInstanceFlag->AutoSize = true;
            this->chkFreeInstanceFlag->Location = System::Drawing::Point(6, 42);
            this->chkFreeInstanceFlag->Name = L"chkFreeInstanceFlag";
```

```
            this->chkFreeInstanceFlag->Size = System::Drawing::Size(208, 17);
            this->chkFreeInstanceFlag->TabIndex = 1;
            this->chkFreeInstanceFlag->Text = L"Кількість екземплярів на
виконання";
            this->chkFreeInstanceFlag->UseVisualStyleBackColor = true;
            //
            // chkTotalInstaceFlag
            //
            this->chkTotalInstaceFlag->AutoSize = true;
            this->chkTotalInstaceFlag->Location = System::Drawing::Point(6, 19);
            this->chkTotalInstaceFlag->Name = L"chkTotalInstaceFlag";
            this->chkTotalInstaceFlag->Size = System::Drawing::Size(293, 17);
            this->chkTotalInstaceFlag->TabIndex = 0;
            this->chkTotalInstaceFlag->Text = L"Загальна кількість екземплярів у
відібраному списку";
            this->chkTotalInstaceFlag->UseVisualStyleBackColor = true;
            //
            // nmrFreeNumber
            //
            this->nmrFreeNumber->Location = System::Drawing::Point(372, 196);
            this->nmrFreeNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrFreeNumber->Name = L"nmrFreeNumber";
            this->nmrFreeNumber->ReadOnly = true;
            this->nmrFreeNumber->Size = System::Drawing::Size(80, 20);
            this->nmrFreeNumber->TabIndex = 8;
            //
            // btnRefresh
            //
            this->btnRefresh->Location = System::Drawing::Point(611, 337);
            this->btnRefresh->Name = L"btnRefresh";
            this->btnRefresh->Size = System::Drawing::Size(99, 23);
            this->btnRefresh->TabIndex = 7;
            this->btnRefresh->Text = L"Зберегти";
            this->btnRefresh->UseVisualStyleBackColor = true;
            this->btnRefresh->Click += gcnew System::EventHandler(this,
&frmWorker::btnRefresh_Click);
            //
            // nmrTotalNumber
            //
            this->nmrTotalNumber->Location = System::Drawing::Point(139, 196);
            this->nmrTotalNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrTotalNumber->Name = L"nmrTotalNumber";
            this->nmrTotalNumber->Size = System::Drawing::Size(80, 20);
            this->nmrTotalNumber->TabIndex = 7;
            this->nmrTotalNumber->ValueChanged += gcnew
System::EventHandler(this, &frmWorker::nmrTotalNumber_ValueChanged);
            //
            // tabHistory
            //
            this->tabHistory->Controls->Add(this->btnClearLogs);
            this->tabHistory->Controls->Add(this->groupOutput);
            this->tabHistory->Controls->Add(this->groupInput);
            this->tabHistory->Location = System::Drawing::Point(4, 22);
            this->tabHistory->Name = L"tabHistory";
            this->tabHistory->Size = System::Drawing::Size(717, 366);
            this->tabHistory->TabIndex = 2;
            this->tabHistory->Text = L"Історія";
            this->tabHistory->UseVisualStyleBackColor = true;
            //
            // btnClearLogs
            //
            this->btnClearLogs->Location = System::Drawing::Point(631, 339);
            this->btnClearLogs->Name = L"btnClearLogs";
            this->btnClearLogs->Size = System::Drawing::Size(75, 23);
            this->btnClearLogs->TabIndex = 0;
            this->btnClearLogs->Text = L"Очистити";
            this->btnClearLogs->UseVisualStyleBackColor = true;
```

```
            this->btnClearLogs->Click += gcnew System::EventHandler(this,
&frmWorker::btnClearLogs_Click);
            //
            // groupOutput
            //
            this->groupOutput->Controls->Add(this->txtOutputHistory);
            this->groupOutput->Location = System::Drawing::Point(6, 166);
            this->groupOutput->Name = L"groupOutput";
            this->groupOutput->Size = System::Drawing::Size(700, 167);
            this->groupOutput->TabIndex = 1;
            this->groupOutput->TabStop = false;
            this->groupOutput->Text = L"Історія виконань";
            //
            // txtOutputHistory
            //
            this->txtOutputHistory->Location = System::Drawing::Point(6, 19);
            this->txtOutputHistory->Multiline = true;
            this->txtOutputHistory->Name = L"txtOutputHistory";
            this->txtOutputHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
            this->txtOutputHistory->Size = System::Drawing::Size(688, 139);
            this->txtOutputHistory->TabIndex = 2;
            //
            // groupInput
            //
            this->groupInput->Controls->Add(this->txtInputHistory);
            this->groupInput->Location = System::Drawing::Point(8, 3);
            this->groupInput->Name = L"groupInput";
            this->groupInput->Size = System::Drawing::Size(700, 157);
            this->groupInput->TabIndex = 0;
            this->groupInput->TabStop = false;
            this->groupInput->Text = L"Історія надходжень";
            //
            // txtInputHistory
            //
            this->txtInputHistory->Location = System::Drawing::Point(6, 19);
            this->txtInputHistory->Multiline = true;
            this->txtInputHistory->Name = L"txtInputHistory";
            this->txtInputHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
            this->txtInputHistory->Size = System::Drawing::Size(688, 130);
            this->txtInputHistory->TabIndex = 1;
            //
            // nmrPageNumber
            //
            this->nmrPageNumber->Location = System::Drawing::Point(139, 162);
            this->nmrPageNumber->Maximum = System::Decimal(gcnew cli::array<
System::Int32 >(4) {65535, 0, 0, 0});
            this->nmrPageNumber->Name = L"nmrPageNumber";
            this->nmrPageNumber->Size = System::Drawing::Size(80, 20);
            this->nmrPageNumber->TabIndex = 6;
            //
            // txtTheme
            //
            this->txtTheme->Location = System::Drawing::Point(139, 128);
            this->txtTheme->Name = L"txtTheme";
            this->txtTheme->Size = System::Drawing::Size(302, 20);
            this->txtTheme->TabIndex = 5;
            //
            // txtISBN
            //
            this->txtISBN->Location = System::Drawing::Point(139, 94);
            this->txtISBN->Name = L"txtISBN";
            this->txtISBN->Size = System::Drawing::Size(163, 20);
            this->txtISBN->TabIndex = 4;
            //
            // txtAuthor
            //
            this->txtAuthor->Location = System::Drawing::Point(139, 60);
```

```
            this->txtAuthor->Name = L"txtAuthor";
            this->txtAuthor->Size = System::Drawing::Size(505, 20);
            this->txtAuthor->TabIndex = 3;
            //
            // txtDocName
            //
            this->txtDocName->Location = System::Drawing::Point(139, 26);
            this->txtDocName->Name = L"txtDocName";
            this->txtDocName->Size = System::Drawing::Size(557, 20);
            this->txtDocName->TabIndex = 2;
            //
            // tabReports
            //
            this->tabReports->Controls->Add(this->btnRefresh);
            this->tabReports->Controls->Add(this->groupBox2);
            this->tabReports->Location = System::Drawing::Point(4, 22);
            this->tabReports->Name = L"tabReports";
            this->tabReports->Padding = System::Windows::Forms::Padding(3);
            this->tabReports->Size = System::Drawing::Size(717, 366);
            this->tabReports->TabIndex = 1;
            this->tabReports->Text = L"Звіти";
            this->tabReports->UseVisualStyleBackColor = true;
            //
            // lblFreeNumber
            //
            this->lblFreeNumber->AutoSize = true;
            this->lblFreeNumber->Location = System::Drawing::Point(263, 196);
            this->lblFreeNumber->Name = L"lblFreeNumber";
            this->lblFreeNumber->Size = System::Drawing::Size(103, 13);
            this->lblFreeNumber->TabIndex = 6;
            this->lblFreeNumber->Text = L"Вільно екземплярів";
            //
            // lblTotalNumber
            //
            this->lblTotalNumber->AutoSize = true;
            this->lblTotalNumber->Location = System::Drawing::Point(34, 196);
            this->lblTotalNumber->Name = L"lblTotalNumber";
            this->lblTotalNumber->Size = System::Drawing::Size(105, 13);
            this->lblTotalNumber->TabIndex = 5;
            this->lblTotalNumber->Text = L"Всього екземплярів";
            //
            // lblPagesNumber
            //
            this->lblPagesNumber->AutoSize = true;
            this->lblPagesNumber->Location = System::Drawing::Point(40, 162);
            this->lblPagesNumber->Name = L"lblPagesNumber";
            this->lblPagesNumber->Size = System::Drawing::Size(99, 13);
            this->lblPagesNumber->TabIndex = 4;
            this->lblPagesNumber->Text = L"Кількість сторінок";
            //
            // lblTheme
            //
            this->lblTheme->AutoSize = true;
            this->lblTheme->Location = System::Drawing::Point(108, 128);
            this->lblTheme->Name = L"lblTheme";
            this->lblTheme->Size = System::Drawing::Size(31, 13);
            this->lblTheme->TabIndex = 3;
            this->lblTheme->Text = L"Тема";
            //
            // statusStrip
            //
            this->statusStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->lblStatus});
            this->statusStrip->Location = System::Drawing::Point(0, 417);
            this->statusStrip->Name = L"statusStrip";
            this->statusStrip->Size = System::Drawing::Size(722, 22);
            this->statusStrip->TabIndex = 10;
            this->statusStrip->Text = L"statusStrip1";
            //
```

```
            // lblStatus
            //
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(109, 17);
            this->lblStatus->Text = L"toolStripStatusLabel1";
            //
            // txtFindString
            //
            this->txtFindString->Location = System::Drawing::Point(6, 19);
            this->txtFindString->Name = L"txtFindString";
            this->txtFindString->Size = System::Drawing::Size(609, 20);
            this->txtFindString->TabIndex = 0;
            //
            // tabControl
            //
            this->tabControl->Controls->Add(this->tabDocs);
            this->tabControl->Controls->Add(this->tabReports);
            this->tabControl->Controls->Add(this->tabHistory);
            this->tabControl->Location = System::Drawing::Point(0, 27);
            this->tabControl->Name = L"tabControl";
            this->tabControl->SelectedIndex = 0;
            this->tabControl->Size = System::Drawing::Size(725, 392);
            this->tabControl->TabIndex = 11;
            //
            // tabDocs
            //
            this->tabDocs->Controls->Add(this->btnEdit);
            this->tabDocs->Controls->Add(this->btnDelete);
            this->tabDocs->Controls->Add(this->btnCreate);
            this->tabDocs->Controls->Add(this->groupInfo);
            this->tabDocs->Controls->Add(this->groupFinding);
            this->tabDocs->Location = System::Drawing::Point(4, 22);
            this->tabDocs->Name = L"tabDocs";
            this->tabDocs->Padding = System::Windows::Forms::Padding(3);
            this->tabDocs->Size = System::Drawing::Size(717, 366);
            this->tabDocs->TabIndex = 0;
            this->tabDocs->Text = L"Документи";
            this->tabDocs->UseVisualStyleBackColor = true;
            //
            // btnEdit
            //
            this->btnEdit->Location = System::Drawing::Point(402, 337);
            this->btnEdit->Name = L"btnEdit";
            this->btnEdit->Size = System::Drawing::Size(99, 23);
            this->btnEdit->TabIndex = 14;
            this->btnEdit->Text = L"Редагувати";
            this->btnEdit->UseVisualStyleBackColor = true;
            this->btnEdit->Click += gcnew System::EventHandler(this,
&frmWorker::btnEdit_Click);
            //
            // btnDelete
            //
            this->btnDelete->Location = System::Drawing::Point(507, 337);
            this->btnDelete->Name = L"btnDelete";
            this->btnDelete->Size = System::Drawing::Size(99, 23);
            this->btnDelete->TabIndex = 15;
            this->btnDelete->Text = L"Видалити";
            this->btnDelete->UseVisualStyleBackColor = true;
            this->btnDelete->Click += gcnew System::EventHandler(this,
&frmWorker::btnDelete_Click);
            //
            // btnCreate
            //
            this->btnCreate->Location = System::Drawing::Point(612, 337);
            this->btnCreate->Name = L"btnCreate";
            this->btnCreate->Size = System::Drawing::Size(99, 23);
            this->btnCreate->TabIndex = 16;
            this->btnCreate->Text = L"Створити";
            this->btnCreate->UseVisualStyleBackColor = true;
```

```
            this->btnCreate->Click += gcnew System::EventHandler(this,
&frmWorker::btnCreate_Click);
            //
            // groupInfo
            //
            this->groupInfo->Controls->Add(this->txtPosition);
            this->groupInfo->Controls->Add(this->btnFirst);
            this->groupInfo->Controls->Add(this->btnPrev);
            this->groupInfo->Controls->Add(this->btnNext);
            this->groupInfo->Controls->Add(this->btnLast);
            this->groupInfo->Controls->Add(this->nmrFreeNumber);
            this->groupInfo->Controls->Add(this->nmrTotalNumber);
            this->groupInfo->Controls->Add(this->nmrPageNumber);
            this->groupInfo->Controls->Add(this->txtTheme);
            this->groupInfo->Controls->Add(this->txtISBN);
            this->groupInfo->Controls->Add(this->txtAuthor);
            this->groupInfo->Controls->Add(this->txtDocName);
            this->groupInfo->Controls->Add(this->lblFreeNumber);
            this->groupInfo->Controls->Add(this->lblTotalNumber);
            this->groupInfo->Controls->Add(this->lblPagesNumber);
            this->groupInfo->Controls->Add(this->lblTheme);
            this->groupInfo->Controls->Add(this->lblISBN);
            this->groupInfo->Controls->Add(this->lblAuthor);
            this->groupInfo->Controls->Add(this->lblDocName);
            this->groupInfo->Location = System::Drawing::Point(6, 65);
            this->groupInfo->Name = L"groupInfo";
            this->groupInfo->Size = System::Drawing::Size(702, 266);
            this->groupInfo->TabIndex = 9;
            this->groupInfo->TabStop = false;
            this->groupInfo->Text = L"Інформація про документ";
            //
            // txtPosition
            //
            this->txtPosition->Location = System::Drawing::Point(564, 241);
            this->txtPosition->Name = L"txtPosition";
            this->txtPosition->Size = System::Drawing::Size(58, 20);
            this->txtPosition->TabIndex = 11;
            this->txtPosition->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
            this->txtPosition->KeyDown += gcnew
System::Windows::Forms::KeyEventHandler(this, &frmWorker::txtPosition_KeyDown);
            //
            // btnFirst
            //
            this->btnFirst->Location = System::Drawing::Point(490, 241);
            this->btnFirst->Name = L"btnFirst";
            this->btnFirst->Size = System::Drawing::Size(31, 20);
            this->btnFirst->TabIndex = 9;
            this->btnFirst->Text = L"|<<";
            this->btnFirst->UseVisualStyleBackColor = true;
            this->btnFirst->Click += gcnew System::EventHandler(this,
&frmWorker::btnFirst_Click);
            //
            // btnPrev
            //
            this->btnPrev->Location = System::Drawing::Point(527, 241);
            this->btnPrev->Name = L"btnPrev";
            this->btnPrev->Size = System::Drawing::Size(31, 20);
            this->btnPrev->TabIndex = 10;
            this->btnPrev->Text = L"<<";
            this->btnPrev->UseVisualStyleBackColor = true;
            this->btnPrev->Click += gcnew System::EventHandler(this,
&frmWorker::btnPrev_Click);
            //
            // btnNext
            //
            this->btnNext->Location = System::Drawing::Point(628, 240);
            this->btnNext->Name = L"btnNext";
            this->btnNext->Size = System::Drawing::Size(31, 20);
```

```
            this->btnNext->TabIndex = 12;
            this->btnNext->Text = L">>";
            this->btnNext->UseVisualStyleBackColor = true;
            this->btnNext->Click += gcnew System::EventHandler(this,
&frmWorker::btnNext_Click);
            //
            // btnLast
            //
            this->btnLast->Location = System::Drawing::Point(665, 240);
            this->btnLast->Name = L"btnLast";
            this->btnLast->Size = System::Drawing::Size(31, 20);
            this->btnLast->TabIndex = 13;
            this->btnLast->Text = L">>|";
            this->btnLast->UseVisualStyleBackColor = true;
            this->btnLast->Click += gcnew System::EventHandler(this,
&frmWorker::btnLast_Click);
            //
            // lblISBN
            //
            this->lblISBN->AutoSize = true;
            this->lblISBN->Location = System::Drawing::Point(2, 94);
            this->lblISBN->Name = L"lblISBN";
            this->lblISBN->Size = System::Drawing::Size(137, 13);
            this->lblISBN->TabIndex = 2;
            this->lblISBN->Text = L"Вхідний номер документу";
            //
            // lblAuthor
            //
            this->lblAuthor->AutoSize = true;
            this->lblAuthor->Location = System::Drawing::Point(101, 60);
            this->lblAuthor->Name = L"lblAuthor";
            this->lblAuthor->Size = System::Drawing::Size(38, 13);
            this->lblAuthor->TabIndex = 1;
            this->lblAuthor->Text = L"Автор";
            //
            // lblDocName
            //
            this->lblDocName->AutoSize = true;
            this->lblDocName->Location = System::Drawing::Point(44, 26);
            this->lblDocName->Name = L"lblDocName";
            this->lblDocName->Size = System::Drawing::Size(95, 13);
            this->lblDocName->TabIndex = 0;
            this->lblDocName->Text = L"Назва документу";
            //
            // groupFinding
            //
            this->groupFinding->Controls->Add(this->btnFind);
            this->groupFinding->Controls->Add(this->txtFindString);
            this->groupFinding->Location = System::Drawing::Point(6, 6);
            this->groupFinding->Name = L"groupFinding";
            this->groupFinding->Size = System::Drawing::Size(702, 53);
            this->groupFinding->TabIndex = 8;
            this->groupFinding->TabStop = false;
            this->groupFinding->Text = L"Ключові рядки для пошуку";
            //
            // btnFind
            //
            this->btnFind->Location = System::Drawing::Point(621, 16);
            this->btnFind->Name = L"btnFind";
            this->btnFind->Size = System::Drawing::Size(75, 23);
            this->btnFind->TabIndex = 1;
            this->btnFind->Text = L"Знайти";
            this->btnFind->UseVisualStyleBackColor = true;
            this->btnFind->Click += gcnew System::EventHandler(this,
&frmWorker::btnFind_Click);
            //
            // frmWorker
            //
            this->AutoScaleDimensions = System::Drawing::Size(6, 13);
```

```
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(722, 439);
            this->Controls->Add(this->menuStrip);
            this->Controls->Add(this->statusStrip);
            this->Controls->Add(this->tabControl);
            this->Name = L"frmWorker";
            this->Text = L"Режим  адміністратора - Система Управління
Електронним Документообігом ";
            this->Load += gcnew System::EventHandler(this,
&frmWorker::frmWorker_Load);
            this->FormClosed += gcnew
System::Windows::Forms::FormClosedEventHandler(this,
&frmWorker::frmWorker_FormClosed);
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPagesNumber))->EndInit();
            this->menuStrip->ResumeLayout(false);
            this->menuStrip->PerformLayout();
            this->groupBox 2-2->ResumeLayout(false);
            this->groupBox 2-2->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrFreeNumber))->EndInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrTotalNumber))->EndInit();
            this->tabHistory->ResumeLayout(false);
            this->groupOutput->ResumeLayout(false);
            this->groupOutput->PerformLayout();
            this->groupInput->ResumeLayout(false);
            this->groupInput->PerformLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>nmrPageNumber))->EndInit();
            this->tabReports->ResumeLayout(false);
            this->statusStrip->ResumeLayout(false);
            this->statusStrip->PerformLayout();
            this->tabControl->ResumeLayout(false);
            this->tabDocs->ResumeLayout(false);
            this->groupInfo->ResumeLayout(false);
            this->groupInfo->PerformLayout();
            this->groupFinding->ResumeLayout(false);
            this->groupFinding->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();

      }
#pragma endregion
/*****************************************************************/
private: System::Void LockingForm()
       {
             // Блокування всіх елементів керування
             btnFind->Enabled = false;
             btnEdit->Enabled = false;
             btnDelete->Enabled = false;
             btnCreate->Enabled = false;
             btnFirst->Enabled = false;
             btnLast->Enabled = false;
             btnNext->Enabled = false;
             btnPrev->Enabled = false;
             txtPosition->Enabled = false;
             menuStrip->Enabled = false;
             tabControl->TabPages[1]->Enabled = false;
             tabControl->TabPages[2]->Enabled = false;
       }
/*****************************************************************/
private: System::Void UnlockingForm()
       {
             // Розблокування всіх елементів керування
             btnFind->Enabled = true;
             btnEdit->Enabled = true;
             btnDelete->Enabled = true;
             btnCreate->Enabled = true;
```

```
            btnFirst->Enabled = true;
            btnLast->Enabled = true;
            btnNext->Enabled = true;
            btnPrev->Enabled = true;
            txtPosition->Enabled = true;
            menuStrip->Enabled = true;
            tabControl->TabPages[1]->Enabled = true;
            tabControl->TabPages[2]->Enabled = true;
      }
/*************************************************************************/
private: System::Void UpdateView()
      {
            // Вивести історію
            txtInputHistory->Text = Worker->ViewInputLog();
            txtOutputHistory->Text = Worker->ViewOutputLog();

            // Список не порожній?
            if(listView->Count != 0)
            {
                  // Перевірити індекс поточного елемента
                  if(ddCurrentIndex >= listView->Count)
                  {
                        // Установити покажчик на останній елемент
                        ddCurrentIndex = listView->Count - 1;
                  }

                  // Значення індексу негативно?
                  if(ddCurrentIndex < 0)
                  {
                        // Установити індекс на перший елемент
                        ddCurrentIndex = 0;
                  }

                  // Одержати характеристики документа й заповнити поля форми
                  txtDocName->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetName();
                  txtAuthor->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetAuthor();
                  txtISBN->Text = ((CDoc^)listView[ddCurrentIndex])->GetISBN();
                  txtTheme->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetTheme();
                  nmrPageNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetPages();
                  nmrTotalNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetTotalNumber();
                  nmrFreeNumber->Value = ((CDoc^)listView[ddCurrentIndex])-
>GetFreeNumber();

                  // Установити поточну позицію
                  txtPosition->Text = Convert::ToString(ddCurrentIndex + 1) + "
/ " +
                        Convert::ToString(listView->Count);

                  // Розблокувати кнопки редагування й видалення
                  btnDelete->Enabled = true;
                  btnEdit->Enabled = true;
            }
            else
            {
                  // Очистити всі поля
                  txtDocName->Text = "";
                  txtAuthor->Text = "";
                  txtISBN->Text = "";
                  txtTheme->Text = "";
                  nmrPageNumber->Value = 0;
                  nmrTotalNumber->Value = 0;
                  nmrFreeNumber->Value = 0;
                  // Поточна позиція - 0
                  txtPosition->Text = "0/0";
```

```
                    // Установити індекс поточного елемента
                    ddCurrentIndex = 0;
                    // Заблокувати кнопки редагування й видалення
                    btnDelete->Enabled = false;
                    btnEdit->Enabled = false;
                    // Вивести стан
                    lblStatus->Text = "Немає записів для відображення";
              }
       }
/************************************************************************/
private: System::Void frmWorker_Load(System::Object^  sender, System::EventArgs^
e)
       {
              // Створити об'єкт "Адміністратор"
              Worker = gcnew CWorker();

              // Завантажити список документів
              Worker->LoadDocList();

              // Завантажити історію
              Worker->ReadLogs();

              // Обновити форму
             UpdateView();

             // Вивести стан
             lblStatus->Text = "Для відображення існуючого списку скористайтеся
пошуком";

             // Заповнення випадаючого списку вибору документів по кількості
сторінок
             cmbPagesDirect->Items->Add(PAGES_LESS);
             cmbPagesDirect->Items->Add(PAGES_GREATER);
             cmbPagesDirect->Items->Add(PAGES_EQUAL);
       }
 /************************************************************************/
private: System::Void btnCreate_Click(System::Object^  sender,
System::EventArgs^  e)
       {
              // Кнопка "Створити" була натиснута раніше?
              if(IsCreateClicked)
              {
                    // Забрати зайві пробіли у всіх полях
                    txtDocName->Text = txtDocName->Text->Trim();
                    txtAuthor->Text = txtAuthor->Text->Trim();
                    txtISBN->Text = txtISBN->Text->Trim();
                    txtTheme->Text = txtTheme->Text->Trim();

                    // Перевірити значимі поля на коректність
                    if(txtDocName->Text == "" || txtISBN->Text == "")
                    {
                          // Обов'язкові поля не заповнені
                          MessageBox::Show("Поля 'Назва документа' і 'вхідний
номер документа' обов'язкові для заповнення!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Перевірити кількість екземплярів
                    if(Convert::ToInt32(nmrTotalNumber->Value) <= 0)
                    {
                          // Вивести повідомлення про помилку
                          MessageBox::Show("Некоректне значення кількості
екземплярів",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }
```

```
                    // Створити новий об'єкт "Документ"
                    CDoc^ NewDoc = gcnew CDoc;
                    // Заповнити поля об'єкта
                    NewDoc->SetName(txtDocName->Text);
                    NewDoc->SetAuthor(txtAuthor->Text);
                    NewDoc->SetISBN(txtISBN->Text);
                    NewDoc->SetTheme(txtTheme->Text);
                    NewDoc->SetPages(Convert::ToInt32(nmrPageNumber->Value));
                    NewDoc->SetTotalNumber(Convert::ToInt32(nmrTotalNumber-
>Value));
                    NewDoc->SetFreeNumber(Convert::ToInt32(nmrTotalNumber-
>Value));
                    // Додати документ у загальний список (якщо документ має
унікальний вхідний номер документа)
                    if(!Worker->AddDoc(NewDoc))
                    {
                          MessageBox::Show("Документ із таким вхідним номером
документа вже існує!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Додати документ у список для відображення
                    listView->Add(NewDoc);

                    // Обновити поля на формі
                    ddCurrentIndex ++;
                    UpdateView();

                    // Розблокувати елементи керування
                    UnlockingForm();

                    // Перемінити напис на кнопці
                    btnCreate->Text = "Створити";

                    // Вивід стану
                    lblStatus->Text = "Запис доданий";

                   // Змінити значення прапора
                   IsCreateClicked = false;
             }
             else
             {
                    // Перемінити напис на кнопці
                    btnCreate->Text = "Зберегти";

                    // Заблокувати елементи керування
                    LockingForm();

                    // Розблокувати кнопку збереження
                    btnCreate->Enabled = true;

                    // Розблокувати поля для введення
                    txtDocName->Enabled = true;
                    txtAuthor->Enabled = true;
                    txtISBN->Enabled = true;
                    txtTheme->Enabled = true;
                    nmrPageNumber->Enabled = true;
                    nmrTotalNumber->Enabled = true;

                    // Установити початкові значення полів для введення
                    txtDocName->Text = "";
                    txtAuthor->Text = "";
                    txtISBN->Text = "";
                    txtTheme->Text = "";
                    nmrPageNumber->Value = 0;
                    nmrTotalNumber->Value = 1;
```

```
                  nmrFreeNumber->Value = 1;

                  // Вивести стан
                  lblStatus->Text = "Введіть значення полів";

                  // Змінити значення прапора натискання по кнопці "Створити"
                  IsCreateClicked = true;
            }
      }
/**************************************************************************/
private: System::Void btnDelete_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Видалити документ із загального списку
            Int32 ddResult = Worker->RemoveDoc(
                  ((CDoc^)listView[ddCurrentIndex])->GetISBNHash());

            // Перевірити код повернення
            switch(ddResult)
            {
            case 0:
                  {
                        // Документ вилучений успішно
                        // Видалити документ зі списку для виводу
                        listView->RemoveAt(ddCurrentIndex);
                        // Обновити форму
                        UpdateView();

                        break;
                  }
            case 1:
                  {
                        // Документ не був вилучений, не всі екземпляри в
системі
                        MessageBox::Show("Видалення неможливо, тому що не всі
документи перебувають у системі",
                              "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);

                        break;
                  }
            case 2:
                  {
                        // Документ не знайдений
                        MessageBox::Show("Видалення неможливо, документ у
системі відсутній",
                              "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                        // Видалити документ зі списку для виводу
                        listView->RemoveAt(ddCurrentIndex);
                        // Обновити форму
                        UpdateView();

                        break;
                  }
            }
      }
/**************************************************************************/
private: System::Void btnFind_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Забрати в рядку пошуку зайві пробіли й перетворити до нижнього
регістра
            String^ strValue = txtFindString->Text->Trim()->ToLower();

            // Перевірити рядок
            if(String::IsNullOrEmpty(strValue))
            {
                  // Рядок порожня, запропонувати вивести весь список
```

```
                if(MessageBox::Show("Рядок пошуку не заданий, вивести весь
список?",
                                                "Usrs Manager",
MessageBoxButtons::YesNo,
                                                MessageBoxIcon::Question) ==
::DialogResult::Yes)
                {
                      // Перейти на вивід списку
                      goto OUT_LIST;
                }
                // Завершити роботу функції
                return;
          }

OUT_LIST:     // Здійснити пошук документа по заданих параметрах
          listView = Worker->FindDoc(strValue);

          // Вивести стан
          lblStatus->Text = "Пошук завершений";
          // Обновити форму
          UpdateView();
    }
/*************************************************************************/
private: System::Void btnFirst_Click(System::Object^  sender, System::EventArgs^
e)
    {
          // Установити індекс на перший елемент
          ddCurrentIndex = 0;
          // Обновити форму
          UpdateView();
    }
/*************************************************************************/
private: System::Void btnPrev_Click(System::Object^  sender, System::EventArgs^
e)
    {
          // Установити індекс на попередній елемент
          ddCurrentIndex ---i;
          // Обновити форму
          UpdateView();
    }
/*************************************************************************/
private: System::Void btnNext_Click(System::Object^  sender, System::EventArgs^
e)
    {
          // Установити індекс на наступний елемент
          ddCurrentIndex ++;
          // Обновити форму
          UpdateView();
    }
/*************************************************************************/
private: System::Void btnLast_Click(System::Object^  sender, System::EventArgs^
e)
    {
          // Установити індекс на останній елемент
          ddCurrentIndex = listView->Count - 1;
          // Обновити форму
          UpdateView();
    }
/*************************************************************************/
private: System::Void btnEdit_Click(System::Object^  sender, System::EventArgs^
e)
    {
          // Кнопка "Редагувати" була натиснута раніше?
          if(IsEditClicked)
          {
                // Забрати зайві пробіли
                txtDocName->Text = txtDocName->Text->Trim();
                txtAuthor->Text = txtAuthor->Text->Trim();
                txtTheme->Text = txtTheme->Text->Trim();
```

```
                    // Перевірити значимі поля на коректність
                    if(txtDocName->Text == "")
                    {
                          // Обов'язкові поля не заповнені
                          MessageBox::Show("Поле 'Назва документа' обов'язково
для заповнення!",
                                "Usrs Manager", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                          return;
                    }

                    // Відредагувати об'єкт у списках
                    ((CDoc^)listView[ddCurrentIndex])->SetName(txtDocName->Text);
                    ((CDoc^)listView[ddCurrentIndex])->SetAuthor(txtAuthor-
>Text);
                    ((CDoc^)listView[ddCurrentIndex])->SetTheme(txtTheme->Text);
                    ((CDoc^)listView[ddCurrentIndex])-
>SetPages(Convert::ToInt32(nmrPageNumber->Value));

                    // (???) Видалити документ зі списку перегляду
                    //Worker->RemoveDoc(((CDoc^)listView[ddCurrentIndex])-
>GetISBNHash());
                    // (???) Додати документ у список перегляду
                    //Worker->AddDoc((CDoc^)listView[ddCurrentIndex]);

                    // Обновити форму
                    UpdateView();

                    // Розблокувати елементи керування
                    UnlockingForm();

                    // Розблокувати некоректируємі поля
                    txtISBN->Enabled = true;
                    nmrTotalNumber->Enabled = true;
                    nmrFreeNumber->Enabled = true;

                    // Перемінити напис на кнопці
                    btnEdit->Text = "Редагувати";

                   // Змінити значення прапора натискання кнопки "Редагувати"
                    IsEditClicked = false;
              }
              else
              {
                    // Перемінити напис на кнопці
                    btnEdit->Text = "Зберегти";

                    // Заблокувати елементи керування
                    LockingForm();

                    // Розблокувати кнопку збереження
                    btnEdit->Enabled = true;

                    // Заблокувати некоректируємі поля
                    txtISBN->Enabled = false;
                    nmrTotalNumber->Enabled = false;
                    nmrFreeNumber->Enabled = false;

                    // Змінити значення прапора натискання кнопки "Редагувати"
                    IsEditClicked = true;
              }
        }
/*********************************************************************/
private: System::Void menuFileSave_Click(System::Object^  sender,
System::EventArgs^  e)
        {
              // Зберегти список документів у файл
              Worker->SaveDocList();
```

```
            // Зберегти історію
            Worker->WriteLogs();

            // Вивести стан
            lblStatus->Text = "Збереження виконане";
      }
/*************************************************************************/
private: System::Void menuFileExit_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Вийти з додатка
            Application::Exit();
      }
/*************************************************************************/
private: System::Void nmrTotalNumber_ValueChanged(System::Object^  sender,
System::EventArgs^  e)
      {
            // Можливе натискання тільки при створенні документа
            // Кількість вільних екземплярів дорівнює загальній кількості
екземплярів
            nmrFreeNumber->Value = nmrTotalNumber->Value;
      }
/*************************************************************************/
   private: System::Void frmWorker_FormClosed(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e)
     {
            // Вийти з додатка
            Application::Exit();
     }
/*************************************************************************/
private: System::Void btnClearLogs_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Очистити історію
            Worker->ClearLogs();

            // Обновити форму
            UpdateView();

            // Вивести стан
            lblStatus->Text = "Очищення виконане";
      }
/*************************************************************************/
private: System::Void btnRefresh_Click(System::Object^  sender,
System::EventArgs^  e)
      {
            // Створити об'єкт "Лічильник"
            CCounter^ Counter = gcnew CCounter();

            // Установлений прапор підрахунку сумарної кількості екземплярів?
            if(chkTotalInstaceFlag->Checked)
            {
                  // Обчислити
                  lblTotalInstanceNumber->Text =
                        Convert::ToString(Counter-
>GetTotalInstanceNumber(listView));
            }

            // Установлений прапор підрахунку сумарної кількості вільних
екземплярів?
            if(chkFreeInstanceFlag->Checked)
            {
                  // Обчислити
                  lblFreeInstanceNumber->Text =
                        Convert::ToString(Counter-
>GetFreeInstanceNumber(listView));
            }
```

```
            // Встановлений прапор підрахунку документів заданого автора?
            if(chkAuthorDocFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержати список документів заданого автора
                  listTemp = Counter->GetDocListOfAuthor(listView,
txtAuthorFilter->Text);

                  // Очистити список на формі
                  lstAuthorDocList->Items->Clear();
                  // Вивести список знайдених документів
                  for(Int32 i = 0; i < listTemp->Count; i++)
                  {
                        // Зчитати параметри документа й додати в список на
формі
                        lstAuthorDocList->Items->Add(
                             Convert::ToString(i + 1) + ". '" +
                             ((CDoc^)listTemp[i])->GetName() + "', '" +
                             ((CDoc^)listTemp[i])->GetAuthor() + "', що входить
номер документа " +
                             ((CDoc^)listTemp[i])->GetISBN());
                  }

                  // Очистити тимчасовий список
                  listTemp->Clear();
            }

            // Установлений прапор виводу списку документів по заданій темі?
            if(chkThemeDocFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержить список документів по заданій темі
                  listTemp = Counter->GetDocListOnTheme(listView,
txtThemeFilter->Text);

                  // Очистити список для виводу на формі
                  lstThemeDocList->Items->Clear();
                  // Вивести список знайдених документів
                  for(Int32 i = 0; i < listTemp->Count; i++)
                  {
                        // Завантажити параметри документа й додати в список на
формі
                        lstThemeDocList->Items->Add(
                             Convert::ToString(i + 1) + ". '" +
                             ((CDoc^)listTemp[i])->GetName() + "', '" +
                             ((CDoc^)listTemp[i])->GetTheme() + "', що входить
номер документа " +
                             ((CDoc^)listTemp[i])->GetISBN());
                  }
                  // Очистити тимчасовий список
                  listTemp->Clear();
            }
            // Установлений прапор виводу списку документів із заданою
кількістю сторінок?
            if(chkPagesFlag->Checked)
            {
                  // Створити тимчасовий список
                  ArrayList^ listTemp = gcnew ArrayList();

                  // Одержати список документів, у яких кількість сторінок
задовольняє
                  //  заданій умові
                  listTemp = Counter->GetDocListByPages(listView,
                        cmbPagesDirect->Text, Convert::ToInt32(nmrPagesNumber-
>Value));
```

```
                    // Очистити список для виводу на формі
                    lstPagesDocList->Items->Clear();
                    // Вивести список знайдених документів
                    for(Int32 i = 0; i < listTemp->Count; i++)
                    {
                          // Завантажити параметри документа й додати в список на
формі
                          lstPagesDocList->Items->Add(
                               Convert::ToString(i + 1) + ". '" +
                               ((CDoc^)listTemp[i])->GetName() + "', " +
                               Convert::ToString(((CDoc^)listTemp[i])-
>GetPages())
                               + " стор., що входить номер документа " +
((CDoc^)listTemp[i])->GetISBN());
                    }
                    // Очистити тимчасовий список
                    listTemp->Clear();
              }
              // Вивести стан
              lblStatus->Text = "Відновлення необхідних звітів закінчене";
        }
/************************************************************************/
private: System::Void txtPosition_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e)
        {
              // Натиснута клавіша "Enter"?
              if( e-e->KeyCode == Keys::Enter)
              {
                    try
                    {
                          // Спробувати одержати введений номер документа для
перегляду
                          ddCurrentIndex = Convert::ToInt32(txtPosition->Text) -
1;
                    }
                    catch(...)
                    {
                          // Якщо в процесі перетворення номера з рядка в число
                          //  виникла помилка, те ніяк на це не реагувати
                    }
                    // Обновити форму
                    UpdateView();
              }
              else
              {
                    // Текстове поле містить символ "/"?
                    if(txtPosition->Text->IndexOf("/") != -1)
                    {
                          // Для початку введення нового номера позиції,
необхідно
                          //  очистити текстове поле
                          txtPosition->Text = "";
                    }
              }
        }
/************************************************************************/
private: System::Void menuHelpAbout_Click(System::Object^  sender,
System::EventArgs^  e)
        {
              // Відобразити форму з інформацією про програму
              UsrsManager::frmAbout^ frmNew = gcnew UsrsManager::frmAbout();
              frmNew->Show();
        }
/************************************************************************/
};
}
```

Файл Reader.cpp - вікно користувача

```
/*****************************************************************************/
#include "StdAfx.h"
#include "Reader.h"
#include "Worker.h"
#include "Logger.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Collections;
using namespace System::Windows::Forms;
/*****************************************************************************/
#define READER_FILE_EXTENSION ".LMR"
/*****************************************************************************/
// NAME:          CReader
// DESCRIPTION:   Конструктор класу
// INPUT:         strUserName - ім'я облікового запису користувача
/*****************************************************************************/
CReader::CReader(String^ strUserName)
{
      // Сформувати ім'я файлу облікового запису
      strFileName = strUserName + READER_FILE_EXTENSION;
      // Створити список для документів користувача
      listMyDoc = gcnew ArrayList();
      // Створити об'єкт для ведення історії
      logUsing = gcnew CLogger();
}
/*****************************************************************************/
// NAME:          Load
// DESCRIPTION:   Функція завантаження списку документів користувача з файлу
// INPUT:         Worker - об'єкт, через який здійснюється одержання інформації
про
//                         книзі по вхідному номеру документа
/*****************************************************************************/
Boolean CReader::Load(CWorker^ Worker)
{
      // Перевірка існування файлу користувача
      if(!File::Exists(strFileName))
      {
            // Видати запит на створення нового файлу
            if(MessageBox::Show("Файл користувача не знайдений, створити
новий?",
                  "Система УЕД", MessageBoxButtons::YesNo,
                  MessageBoxIcon::Warning) == DialogResult::Yes)
            {
                  // Створити файловий потік
                  FileStream^ fsReaderFile = gcnew FileStream(strFileName,
                        FileMode::CreateNew);
                  // Відкрити файл користувача
                  StreamWriter^ swReaderFile = gcnew StreamWriter(fsReaderFile);
                  // Кількість документів у користувача дорівнює 0
                  swReaderFile->WriteLine("0");
                  // Закрити файл
                  swReaderFile->Close();
                  // Закрити файловий потік
                  fsReaderFile->Close();
                  // Закінчити виконання функції, але виконати вхід
                  return true;
            }
            else
            {
                  // Закінчити виконання функції й не виконувати вхід
                  return false;
            }
      }
      // Відкрити файл користувача
```

```
	StreamReader^ srReaderFile = gcnew StreamReader(strFileName);
	// Завантажити кількість документів
	Int32 ddDocsNumber = Convert::ToInt32(srReaderFile->ReadLine());
	// Цикл зчитування інформації про документи
	for(Int32 i = 0; i < ddDocsNumber; i++)
	{
		// Зчитування вхідний номер документа з файлу
		Int32 ddISBNHash = Convert::ToInt32(srReaderFile->ReadLine());

		// Одержання інформації й додавання документи в список документів
користувача
		listMyDoc->Add(Worker->ViewDoc(ddISBNHash));
	}

	// Кінець файлу не досягнуть?
	if(!srReaderFile->EndOfStream)
	{
		// Завантажити історію одержань і повернень документів
		logUsing->strLog = srReaderFile->ReadToEnd();
	}
	else
	{
		// Обнулити рядок історії
		logUsing->strLog = "";
	}

	// Закрити файл користувача
	srReaderFile->Close();

	// Закінчити виконання функції й виконати вхід
	return true;
}
/*************************************************************************/
// NAME:          Save
// DESCRIPTION:   Функція збереження списку документів користувача у файл
// INPUT:         N/D
/*************************************************************************/
void CReader::Save()
{
	// Перевірка існування файлу користувача
	if(File::Exists(strFileName))
	{
		// Видалення старого файлу
		File::Delete(strFileName);
	}
	// Відкрити новий файл користувача
	StreamWriter^ swReaderFile = gcnew StreamWriter(strFileName);

	// Записати кількість документів
	Int32 ddDocsNumber = listMyDoc->Count;
	swReaderFile->WriteLine(Convert::ToString(ddDocsNumber));
	// Цикл запису інформації про документи
	for(Int32 i = 0; i < ddDocsNumber; i++)
	{
		// Запис вхідний номер документа у файл
		swReaderFile->WriteLine(((CDoc^)listMyDoc[i])->GetISBNHash());
	}
	// Записати лог
	swReaderFile->WriteLine(logUsing->strLog);
	// Закрити файл користувача
	swReaderFile->Close();
}
/*************************************************************************/
// NAME:          GetMyDocsList
// DESCRIPTION:   Функція одержання списку документів користувача
// INPUT:         N/D
/*************************************************************************/
ArrayList^ CReader::GetMyDocsList()
{
```

```
	return listMyDoc;
}
/*****************************************************************************/
// NAME:		RequireDoc
// DESCRIPTION:	Функція запиту заданого документа
// INPUT:		Worker - об'єкт "Адміністратор", через який здійснюється
одержання
//			ddHash - хеш-значення вхідного номера документа, що
потрібно одержати
/*****************************************************************************/
Boolean CReader::RequireDoc(CWorker^ Worker, Int32 ddHash)
{
	// Запит на одержання документи до "адміністратора"
	Boolean fResult = Worker->GiveDoc(listMyDoc, ddHash);
	// Результат одержання документів позитивний?
	if(fResult)
	{
		// Індекс останньої документи
		Int32 ddIndex = listMyDoc->Count - 1;
		// Записати операцію в лог
		logUsing->WriteEvent("Отриманий документ '" +
((CDoc^)listMyDoc[ddIndex])->GetName() +
			"', Автор '" + ((CDoc^)listMyDoc[ddIndex])->GetAuthor() +
			"', що входить номер документа " +
((CDoc^)listMyDoc[ddIndex])->GetISBN());

		// Операція пройшла успішно
		return true;
	}
	else
	{
		// Такого документа в наявності не є
		return false;
	}
}
/*****************************************************************************/
// NAME:		ReleaseDoc
// DESCRIPTION:	Функція повернення заданого документа
// INPUT:		Worker - об'єкт "Адміністратор", через який здійснюється
повернення
//			ddIndex - індекс документа в списку користувача
/*****************************************************************************/
void CReader::ReleaseDoc(CWorker^ Worker, Int32 ddIndex)
{
	// Повернути документ
	Worker->TakeDoc(listMyDoc, ((CDoc^)listMyDoc[ddIndex])->GetISBNHash());
	// Записати операцію в лог
	logUsing->WriteEvent(
		"Повернутий документ '" + ((CDoc^)listMyDoc[ddIndex])->GetName() +
		"', Автор '" + ((CDoc^)listMyDoc[ddIndex])->GetName() +
		"', що входить номер документа " + ((CDoc^)listMyDoc[ddIndex])-
>GetISBN());
	// Видалити документ зі списку користувача
	listMyDoc->RemoveAt(ddIndex);
}
/*****************************************************************************/
// NAME:		ViewLog
// DESCRIPTION:	Функція перегляду історії
// INPUT:		N/D
/*****************************************************************************/
String^ CReader::ViewLog()
{
	return logUsing->strLog;
}
/*****************************************************************************/
```

Файл Reader.resx - xml опис вікна користувача

```
 <?xml version="1.0" encoding=" utf-8"?>
<root>
  <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
```

```
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <metadata name="statusStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>436, 17</value>
  </metadata>
  <metadata name="menuStrip.TrayLocation" type="System.Drawing.Point,
System.Drawing, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a">
    <value>540, 17</value>
  </metadata>
</root>
```

Файл Reader.h - бібліотека для файлу frmAbout.cpp

```
#pragma once
/*************************************************************************/
#include "Worker.h"
#include "Doc.h"
#include "Reader.h"

#include "frmAbout.h"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
namespace UsrsManager
{
	public ref class frmReader : public System::Windows::Forms::Form
	{
/*************************************************************************/
	private:
	CWorker^ Worker;		// Об'єкт класу "Адміністратор"
	CReader^ Reader;		// Об'єкт класу "Користувач"

	ArrayList^ listView;	// Поточний відображуваний список документів

	Int32 ddCurrentIndex;	// Індекс поточного відображуваного елемента

	String^ strUserName;	// Поточне ім'я користувача
/*************************************************************************/
	public:
	frmReader(String^ strReaderName)
	{
		// Ініціалізація елементів керування на формі
		InitializeComponent();

		// Створити список відображуваних документів
		listView = gcnew ArrayList();

		// Поточний індекс не визначений
		ddCurrentIndex = -1;

		// Зберегти ім'я користувача
		strUserName = strReaderName;
	}
/*************************************************************************/
protected: ~frmReader()
	{
		if (components)
		{
			delete components;
		}
	}
/*************************************************************************/
	private: System::Windows::Forms::Button^  btnRequest;
	private: System::Windows::Forms::ToolStripStatusLabel^  lblStatus;
	private: System::Windows::Forms::StatusStrip^  statusStrip;
	private: System::Windows::Forms::TabPage^  tabFinding;
	private: System::Windows::Forms::GroupBox^  groupBox1;
	private: System::Windows::Forms::Label^  lblPagesNumber;
	private: System::Windows::Forms::Label^  lblTheme;
	private: System::Windows::Forms::Label^  lblISBN;
	private: System::Windows::Forms::Label^  lblAuthor;
	private: System::Windows::Forms::Label^  lblDocName;
	private: System::Windows::Forms::GroupBox^  groupFinding;
```

```
        private: System::Windows::Forms::Button^  btnFind;
        private: System::Windows::Forms::TextBox^  txtFindString;
        private: System::Windows::Forms::TabControl^  tabControl;
        private: System::Windows::Forms::TabPage^  tabMyDocs;
        private: System::Windows::Forms::Button^  btnReturn;
        private: System::Windows::Forms::GroupBox^  groupBox2;
        private: System::Windows::Forms::ListBox^  lstMyDocs;
        private: System::Windows::Forms::TabPage^  tabHistory;
        private: System::Windows::Forms::GroupBox^  groupBox3;
        private: System::Windows::Forms::TextBox^  txtHistory;
        private: System::Windows::Forms::ToolStripMenuItem^  menuHelp;

        private: System::Windows::Forms::ToolStripMenuItem^  menuHelpAbout;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFileExit;
        private: System::Windows::Forms::MenuStrip^  menuStrip;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFile;
        private: System::Windows::Forms::ToolStripMenuItem^  menuFileSave;
        private: System::Windows::Forms::TextBox^  txtPosition;
        private: System::Windows::Forms::Button^  btnFirst;
        private: System::Windows::Forms::Button^  btnPrev;
        private: System::Windows::Forms::Button^  btnNext;
        private: System::Windows::Forms::Button^  btnLast;
        private: System::Windows::Forms::TextBox^  txtPagesNumber;
        private: System::Windows::Forms::TextBox^  txtTheme;
        private: System::Windows::Forms::TextBox^  txtISBN;
        private: System::Windows::Forms::TextBox^  txtAuthor;
        private: System::Windows::Forms::TextBox^  txtDocName;

        private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Необхідний метод для підтримки Розроблювача - не модифікувати
        /// зміст цього методу з кодовим редактором.
        /// </summary>
        void InitializeComponent(void)
        {
            this->btnRequest = (gcnew System::Windows::Forms::Button());
            this->lblStatus = (gcnew
System::Windows::Forms::ToolStripStatusLabel());
            this->statusStrip = (gcnew System::Windows::Forms::StatusStrip());
            this->tabFinding = (gcnew System::Windows::Forms::TabPage());
            this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
            this->txtPosition = (gcnew System::Windows::Forms::TextBox());
            this->btnFirst = (gcnew System::Windows::Forms::Button());
            this->btnPrev = (gcnew System::Windows::Forms::Button());
            this->btnNext = (gcnew System::Windows::Forms::Button());
            this->btnLast = (gcnew System::Windows::Forms::Button());
            this->txtPagesNumber = (gcnew System::Windows::Forms::TextBox());
            this->txtTheme = (gcnew System::Windows::Forms::TextBox());
            this->txtISBN = (gcnew System::Windows::Forms::TextBox());
            this->txtAuthor = (gcnew System::Windows::Forms::TextBox());
            this->txtDocName = (gcnew System::Windows::Forms::TextBox());
            this->lblPagesNumber = (gcnew System::Windows::Forms::Label());
            this->lblTheme = (gcnew System::Windows::Forms::Label());
            this->lblISBN = (gcnew System::Windows::Forms::Label());
            this->lblAuthor = (gcnew System::Windows::Forms::Label());
            this->lblDocName = (gcnew System::Windows::Forms::Label());
            this->groupFinding = (gcnew System::Windows::Forms::GroupBox());
            this->btnFind = (gcnew System::Windows::Forms::Button());
            this->txtFindString = (gcnew System::Windows::Forms::TextBox());
            this->tabControl = (gcnew System::Windows::Forms::TabControl());
            this->tabMyDocs = (gcnew System::Windows::Forms::TabPage());
            this->btnReturn = (gcnew System::Windows::Forms::Button());
            this->groupBox2 = (gcnew System::Windows::Forms::GroupBox());
```

```
            this->lstMyDocs = (gcnew System::Windows::Forms::ListBox());
            this->tabHistory = (gcnew System::Windows::Forms::TabPage());
            this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
            this->txtHistory = (gcnew System::Windows::Forms::TextBox());
            this->menuHelp = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuHelpAbout = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileExit = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuStrip = (gcnew System::Windows::Forms::MenuStrip());
            this->menuFile = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->menuFileSave = (gcnew
System::Windows::Forms::ToolStripMenuItem());
            this->statusStrip->SuspendLayout();
            this->tabFinding->SuspendLayout();
            this->groupBox 1-1->SuspendLayout();
            this->groupFinding->SuspendLayout();
            this->tabControl->SuspendLayout();
            this->tabMyDocs->SuspendLayout();
            this->groupBox 2-2->SuspendLayout();
            this->tabHistory->SuspendLayout();
            this->groupBox 3-3->SuspendLayout();
            this->menuStrip->SuspendLayout();
            this->SuspendLayout();
            //
            // btnRequest
            //
            this->btnRequest->Location = System::Drawing::Point(567, 285);
            this->btnRequest->Name = L"btnRequest";
            this->btnRequest->Size = System::Drawing::Size(144, 23);
            this->btnRequest->TabIndex = 12;
            this->btnRequest->Text = L"Здійснити запит";
            this->btnRequest->UseVisualStyleBackColor = true;
            this->btnRequest->Click += gcnew System::EventHandler(this,
&frmReader::btnRequest_Click);
            //
            // lblStatus
            //
            this->lblStatus->Name = L"lblStatus";
            this->lblStatus->Size = System::Drawing::Size(109, 17);
            this->lblStatus->Text = L"toolStripStatusLabel1";
            this->lblStatus->Click += gcnew System::EventHandler(this,
&frmReader::lblStatus_Click);
            //
            // statusStrip
            //
            this->statusStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->lblStatus});
            this->statusStrip->Location = System::Drawing::Point(0, 365);
            this->statusStrip->Name = L"statusStrip";
            this->statusStrip->Size = System::Drawing::Size(724, 22);
            this->statusStrip->TabIndex = 13;
            this->statusStrip->Text = L"statusStrip1";
            //
            // tabFinding
            //
            this->tabFinding->Controls->Add(this->btnRequest);
            this->tabFinding->Controls->Add(this->groupBox1);
            this->tabFinding->Controls->Add(this->groupFinding);
            this->tabFinding->Location = System::Drawing::Point(4, 22);
            this->tabFinding->Name = L"tabFinding";
            this->tabFinding->Padding = System::Windows::Forms::Padding(3);
            this->tabFinding->Size = System::Drawing::Size(717, 314);
            this->tabFinding->TabIndex = 0;
            this->tabFinding->Text = L"Пошук";
            this->tabFinding->UseVisualStyleBackColor = true;
            //
```

```
            // groupBox1
            //
            this->groupBox 1-1->Controls->Add(this->txtPosition);
            this->groupBox 1-1->Controls->Add(this->btnFirst);
            this->groupBox 1-1->Controls->Add(this->btnPrev);
            this->groupBox 1-1->Controls->Add(this->btnNext);
            this->groupBox 1-1->Controls->Add(this->btnLast);
            this->groupBox 1-1->Controls->Add(this->txtPagesNumber);
            this->groupBox 1-1->Controls->Add(this->txtTheme);
            this->groupBox 1-1->Controls->Add(this->txtISBN);
            this->groupBox 1-1->Controls->Add(this->txtAuthor);
            this->groupBox 1-1->Controls->Add(this->txtDocName);
            this->groupBox 1-1->Controls->Add(this->lblPagesNumber);
            this->groupBox 1-1->Controls->Add(this->lblTheme);
            this->groupBox 1-1->Controls->Add(this->lblISBN);
            this->groupBox 1-1->Controls->Add(this->lblAuthor);
            this->groupBox 1-1->Controls->Add(this->lblDocName);
            this->groupBox 1-1->Location = System::Drawing::Point(6, 65);
            this->groupBox 1-1->Name = L"groupBox1";
            this->groupBox 1-1->Size = System::Drawing::Size(702, 214);
            this->groupBox 1-1->TabIndex = 9;
            this->groupBox 1-1->TabStop = false;
            this->groupBox 1-1->Text = L"Інформація про документ";
            //
            // txtPosition
            //
            this->txtPosition->Location = System::Drawing::Point(568, 187);
            this->txtPosition->Name = L"txtPosition";
            this->txtPosition->Size = System::Drawing::Size(58, 20);
            this->txtPosition->TabIndex = 9;
            this->txtPosition->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
            this->txtPosition->KeyDown += gcnew
System::Windows::Forms::KeyEventHandler(this, &frmReader::txtPosition_KeyDown);
            //
            // btnFirst
            //
            this->btnFirst->Location = System::Drawing::Point(494, 187);
            this->btnFirst->Name = L"btnFirst";
            this->btnFirst->Size = System::Drawing::Size(31, 20);
            this->btnFirst->TabIndex = 7;
            this->btnFirst->Text = L"|<<";
            this->btnFirst->UseVisualStyleBackColor = true;
            this->btnFirst->Click += gcnew System::EventHandler(this,
&frmReader::btnFirst_Click);
            //
            // btnPrev
            //
            this->btnPrev->Location = System::Drawing::Point(531, 187);
            this->btnPrev->Name = L"btnPrev";
            this->btnPrev->Size = System::Drawing::Size(31, 20);
            this->btnPrev->TabIndex = 8;
            this->btnPrev->Text = L"<<";
            this->btnPrev->UseVisualStyleBackColor = true;
            this->btnPrev->Click += gcnew System::EventHandler(this,
&frmReader::btnPrev_Click);
            //
            // btnNext
            //
            this->btnNext->Location = System::Drawing::Point(632, 186);
            this->btnNext->Name = L"btnNext";
            this->btnNext->Size = System::Drawing::Size(31, 20);
            this->btnNext->TabIndex = 10;
            this->btnNext->Text = L">>";
            this->btnNext->UseVisualStyleBackColor = true;
            this->btnNext->Click += gcnew System::EventHandler(this,
&frmReader::btnNext_Click);
            //
            // btnLast
```

```
            //
            this->btnLast->Location = System::Drawing::Point(669, 186);
            this->btnLast->Name = L"btnLast";
            this->btnLast->Size = System::Drawing::Size(31, 20);
            this->btnLast->TabIndex = 11;
            this->btnLast->Text = L">>|";
            this->btnLast->UseVisualStyleBackColor = true;
            this->btnLast->Click += gcnew System::EventHandler(this,
&frmReader::btnLast_Click);
            //
            // txtPagesNumber
            //
            this->txtPagesNumber->Location = System::Drawing::Point(139, 161);
            this->txtPagesNumber->Name = L"txtPagesNumber";
            this->txtPagesNumber->ReadOnly = true;
            this->txtPagesNumber->Size = System::Drawing::Size(48, 20);
            this->txtPagesNumber->TabIndex = 6;
            //
            // txtTheme
            //
            this->txtTheme->Location = System::Drawing::Point(139, 128);
            this->txtTheme->Name = L"txtTheme";
            this->txtTheme->ReadOnly = true;
            this->txtTheme->Size = System::Drawing::Size(302, 20);
            this->txtTheme->TabIndex = 5;
            //
            // txtISBN
            //
            this->txtISBN->Location = System::Drawing::Point(139, 94);
            this->txtISBN->Name = L"txtISBN";
            this->txtISBN->ReadOnly = true;
            this->txtISBN->Size = System::Drawing::Size(163, 20);
            this->txtISBN->TabIndex = 4;
            //
            // txtAuthor
            //
            this->txtAuthor->Location = System::Drawing::Point(139, 60);
            this->txtAuthor->Name = L"txtAuthor";
            this->txtAuthor->ReadOnly = true;
            this->txtAuthor->Size = System::Drawing::Size(505, 20);
            this->txtAuthor->TabIndex = 3;
            //
            // txtDocName
            //
            this->txtDocName->Location = System::Drawing::Point(139, 26);
            this->txtDocName->Name = L"txtDocName";
            this->txtDocName->ReadOnly = true;
            this->txtDocName->Size = System::Drawing::Size(557, 20);
            this->txtDocName->TabIndex = 2;
            //
            // lblPagesNumber
            //
            this->lblPagesNumber->AutoSize = true;
            this->lblPagesNumber->Location = System::Drawing::Point(40, 164);
            this->lblPagesNumber->Name = L"lblPagesNumber";
            this->lblPagesNumber->Size = System::Drawing::Size(99, 13);
            this->lblPagesNumber->TabIndex = 4;
            this->lblPagesNumber->Text = L"Кількість сторінок";
            //
            // lblTheme
            //
            this->lblTheme->AutoSize = true;
            this->lblTheme->Location = System::Drawing::Point(108, 128);
            this->lblTheme->Name = L"lblTheme";
            this->lblTheme->Size = System::Drawing::Size(31, 13);
            this->lblTheme->TabIndex = 3;
            this->lblTheme->Text = L"Тема";
            //
            // lblISBN
```

```
            //
            this->lblISBN->AutoSize = true;
            this->lblISBN->Location = System::Drawing::Point(2, 94);
            this->lblISBN->Name = L"lblISBN";
            this->lblISBN->Size = System::Drawing::Size(137, 13);
            this->lblISBN->TabIndex = 2;
            this->lblISBN->Text = L"Вхідний номер документу";
            //
            // lblAuthor
            //
            this->lblAuthor->AutoSize = true;
            this->lblAuthor->Location = System::Drawing::Point(101, 60);
            this->lblAuthor->Name = L"lblAuthor";
            this->lblAuthor->Size = System::Drawing::Size(38, 13);
            this->lblAuthor->TabIndex = 1;
            this->lblAuthor->Text = L"Автор";
            //
            // lblDocName
            //
            this->lblDocName->AutoSize = true;
            this->lblDocName->Location = System::Drawing::Point(44, 26);
            this->lblDocName->Name = L"lblDocName";
            this->lblDocName->Size = System::Drawing::Size(95, 13);
            this->lblDocName->TabIndex = 0;
            this->lblDocName->Text = L"Назва документу";
            //
            // groupFinding
            //
            this->groupFinding->Controls->Add(this->btnFind);
            this->groupFinding->Controls->Add(this->txtFindString);
            this->groupFinding->Location = System::Drawing::Point(6, 6);
            this->groupFinding->Name = L"groupFinding";
            this->groupFinding->Size = System::Drawing::Size(702, 53);
            this->groupFinding->TabIndex = 8;
            this->groupFinding->TabStop = false;
            this->groupFinding->Text = L"Ключові рядки для пошуку";
            //
            // btnFind
            //
            this->btnFind->Location = System::Drawing::Point(621, 19);
            this->btnFind->Name = L"btnFind";
            this->btnFind->Size = System::Drawing::Size(75, 23);
            this->btnFind->TabIndex = 1;
            this->btnFind->Text = L"Знайти";
            this->btnFind->UseVisualStyleBackColor = true;
            this->btnFind->Click += gcnew System::EventHandler(this,
&frmReader::btnFind_Click);
            //
            // txtFindString
            //
            this->txtFindString->Location = System::Drawing::Point(6, 19);
            this->txtFindString->Name = L"txtFindString";
            this->txtFindString->Size = System::Drawing::Size(609, 20);
            this->txtFindString->TabIndex = 0;
            //
            // tabControl
            //
            this->tabControl->Controls->Add(this->tabFinding);
            this->tabControl->Controls->Add(this->tabMyDocs);
            this->tabControl->Controls->Add(this->tabHistory);
            this->tabControl->Location = System::Drawing::Point(0, 27);
            this->tabControl->Name = L"tabControl";
            this->tabControl->SelectedIndex = 0;
            this->tabControl->Size = System::Drawing::Size(725, 340);
            this->tabControl->TabIndex = 14;
            //
            // tabMyDocs
            //
            this->tabMyDocs->Controls->Add(this->btnReturn);
```

```
            this->tabMyDocs->Controls->Add(this->groupBox2);
            this->tabMyDocs->Location = System::Drawing::Point(4, 22);
            this->tabMyDocs->Name = L"tabMyDocs";
            this->tabMyDocs->Padding = System::Windows::Forms::Padding(3);
            this->tabMyDocs->Size = System::Drawing::Size(717, 314);
            this->tabMyDocs->TabIndex = 1;
            this->tabMyDocs->Text = L"Документи";
            this->tabMyDocs->UseVisualStyleBackColor = true;
            //
            // btnReturn
            //
            this->btnReturn->Location = System::Drawing::Point(622, 285);
            this->btnReturn->Name = L"btnReturn";
            this->btnReturn->Size = System::Drawing::Size(85, 23);
            this->btnReturn->TabIndex = 1;
            this->btnReturn->Text = L"Повернути";
            this->btnReturn->UseVisualStyleBackColor = true;
            this->btnReturn->Click += gcnew System::EventHandler(this,
&frmReader::btnReturn_Click);
            //
            // groupBox2
            //
            this->groupBox 2-2->Controls->Add(this->lstMyDocs);
            this->groupBox 2-2->Location = System::Drawing::Point(8, 6);
            this->groupBox 2-2->Name = L"groupBox2";
            this->groupBox 2-2->Size = System::Drawing::Size(699, 273);
            this->groupBox 2-2->TabIndex = 0;
            this->groupBox 2-2->TabStop = false;
            this->groupBox 2-2->Text = L"Список документів користувача";
            this->groupBox 2-2->Enter += gcnew System::EventHandler(this,
&frmReader::groupBox2_Enter);
            //
            // lstMyDocs
            //
            this->lstMyDocs->FormattingEnabled = true;
            this->lstMyDocs->HorizontalScrollbar = true;
            this->lstMyDocs->Location = System::Drawing::Point(6, 19);
            this->lstMyDocs->Name = L"lstMyDocs";
            this->lstMyDocs->Size = System::Drawing::Size(687, 251);
            this->lstMyDocs->TabIndex = 0;
            //
            // tabHistory
            //
            this->tabHistory->Controls->Add(this->groupBox3);
            this->tabHistory->Location = System::Drawing::Point(4, 22);
            this->tabHistory->Name = L"tabHistory";
            this->tabHistory->Size = System::Drawing::Size(717, 314);
            this->tabHistory->TabIndex = 2;
            this->tabHistory->Text = L"Історія";
            this->tabHistory->UseVisualStyleBackColor = true;
            //
            // groupBox3
            //
            this->groupBox 3-3->Controls->Add(this->txtHistory);
            this->groupBox 3-3->Location = System::Drawing::Point(8, 8);
            this->groupBox 3-3->Name = L"groupBox3";
            this->groupBox 3-3->Size = System::Drawing::Size(700, 303);
            this->groupBox 3-3->TabIndex = 2;
            this->groupBox 3-3->TabStop = false;
            this->groupBox 3-3->Text = L"Історія перегляду документів";
            //
            // txtHistory
            //
            this->txtHistory->Location = System::Drawing::Point(6, 19);
            this->txtHistory->Multiline = true;
            this->txtHistory->Name = L"txtHistory";
            this->txtHistory->ReadOnly = true;
            this->txtHistory->ScrollBars =
System::Windows::Forms::ScrollBars::Both;
```

```
            this->txtHistory->Size = System::Drawing::Size(688, 278);
            this->txtHistory->TabIndex = 0;
            //
            // menuHelp
            //
            this->menuHelp->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) {this->menuHelpAbout});
            this->menuHelp->Name = L"menuHelp";
            this->menuHelp->Size = System::Drawing::Size(60, 20);
            this->menuHelp->Text = L"Довідка";
            //
            // menuHelpAbout
            //
            this->menuHelpAbout->Name = L"menuHelpAbout";
            this->menuHelpAbout->Size = System::Drawing::Size(166, 22);
            this->menuHelpAbout->Text = L"Про програму...";
            this->menuHelpAbout->Click += gcnew System::EventHandler(this,
&frmReader::menuHelpAbout_Click);
            //
            // menuFileExit
            //
            this->menuFileExit->Name = L"menuFileExit";
            this->menuFileExit->Size = System::Drawing::Size(152, 22);
            this->menuFileExit->Text = L"Вихід";
            this->menuFileExit->Click += gcnew System::EventHandler(this,
&frmReader::menuFileExit_Click);
            //
            // menuStrip
            //
            this->menuStrip->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFile, this->menuHelp});
            this->menuStrip->Location = System::Drawing::Point(0, 0);
            this->menuStrip->Name = L"menuStrip";
            this->menuStrip->Size = System::Drawing::Size(724, 24);
            this->menuStrip->TabIndex = 12;
            this->menuStrip->Text = L"menuStrip1";
            //
            // menuFile
            //
            this->menuFile->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(2) {this->menuFileSave,
                this->menuFileExit});
            this->menuFile->Name = L"menuFile";
            this->menuFile->Size = System::Drawing::Size(45, 20);
            this->menuFile->Text = L"Файл";
            //
            // menuFileSave
            //
            this->menuFileSave->Name = L"menuFileSave";
            this->menuFileSave->Size = System::Drawing::Size(152, 22);
            this->menuFileSave->Text = L"Зберегти";
            this->menuFileSave->Click += gcnew System::EventHandler(this,
&frmReader::menuFileSave_Click);
            //
            // frmReader
            //
            this->AutoScaleDimensions = System::Drawing::Size(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(724, 387);
            this->Controls->Add(this->statusStrip);
            this->Controls->Add(this->tabControl);
            this->Controls->Add(this->menuStrip);
            this->MaximizeBox = false;
            this->MinimizeBox = false;
            this->Name = L"frmReader";
            this->Text = L"Режим  користувача - Система Управління Електронним
Документообігом ";
            this->Load += gcnew System::EventHandler(this,
&frmReader::frmReader_Load);
```

```
            this->FormClosed += gcnew
System::Windows::Forms::FormClosedEventHandler(this,
&frmReader::frmReader_FormClosed);
            this->statusStrip->ResumeLayout(false);
            this->statusStrip->PerformLayout();
            this->tabFinding->ResumeLayout(false);
            this->groupBox 1-1->ResumeLayout(false);
            this->groupBox 1-1->PerformLayout();
            this->groupFinding->ResumeLayout(false);
            this->groupFinding->PerformLayout();
            this->tabControl->ResumeLayout(false);
            this->tabMyDocs->ResumeLayout(false);
            this->groupBox 2-2->ResumeLayout(false);
            this->tabHistory->ResumeLayout(false);
            this->groupBox 3-3->ResumeLayout(false);
            this->groupBox 3-3->PerformLayout();
            this->menuStrip->ResumeLayout(false);
            this->menuStrip->PerformLayout();
            this->ResumeLayout(false);
            this->PerformLayout();

      }
#pragma endregion
 /*************************************************************************/
 private: System::Void UpdateView()
       {
             // Список не порожній?
             if(listView->Count != 0)
             {
                   // Перевірити індекс поточного елемента
                   if(ddCurrentIndex >= listView->Count)
                   {
                         // Установити покажчик на останній елемент
                         ddCurrentIndex = listView->Count - 1;
                   }

                   // Поточний індекс менше нуля?
                   if(ddCurrentIndex < 0)
                   {
                         // Установити індекс на перший елемент
                         ddCurrentIndex = 0;
                   }

                   // Одержати характеристики документа й заповнити поля форми
                   txtDocName->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetName();
                   txtAuthor->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetAuthor();
                   txtISBN->Text = ((CDoc^)listView[ddCurrentIndex])->GetISBN();
                   txtTheme->Text = ((CDoc^)listView[ddCurrentIndex])-
>GetTheme();
                   txtPagesNumber->Text =
Convert::ToString(((CDoc^)listView[ddCurrentIndex])->GetPages());

                   // Документ є в наявності?
                   if(((CDoc^)listView[ddCurrentIndex])->GetFreeNumber() > 0)
                   {
                         // Розблокувати кнопку запросу
                         btnRequest->Enabled = true;
                   }
                   else
                   {
                         // Заблокувати кнопку запросу
                         btnRequest->Enabled = false;
                   }

                   // Установити й вивести поточну позицію
                   txtPosition->Text = Convert::ToString(ddCurrentIndex + 1) + "
/ " +
```

```
                        Convert::ToString(listView->Count);
            }
            else
            {
                  // Очистити всі поля
                  txtDocName->Text = "";
                  txtAuthor->Text = "";
                  txtISBN->Text = "";
                  txtTheme->Text = "";
                  txtPagesNumber->Text = "";
                  // Поточна позиція - 0
                  txtPosition->Text = "0/0";
                  // Установити індекс поточного елемента
                  ddCurrentIndex = 0;
                  // Вивести стан
                  lblStatus->Text = "Немає записів для відображення";
            }

            // Вивести історію
            txtHistory->Text = Reader->ViewLog();

            // Одержати список документів користувача
            ArrayList^ listTemp = gcnew ArrayList();
            listTemp = Reader->GetMyDocsList();

            // Очистити список на формі
            lstMyDocs->Items->Clear();
            // Кількість документів не дорівнює нулю?
            if(listTemp->Count != 0)
            {
                  // Заповнити список документів на формі
                  for(Int32 i = 0; i < listTemp->Count; i++)
                  {
                        // Одержати характеристики документа й вивести їх у
список
                        lstMyDocs->Items->Add(Convert::ToString(i + 1) + ". '"
+
                              ((CDoc^)listTemp[i])->GetName() + "', '" +
                              ((CDoc^)listTemp[i])->GetAuthor() + "', вхідний
номер документа " +
                              ((CDoc^)listTemp[i])->GetISBN());
                  }
            }
      }
/*********************************************************************/
private: System::Void frmReader_Load(System::Object^  sender, System::EventArgs^
e)
      {
            // Створити об'єкт "Адміністратор"
            Worker = gcnew CWorker();
            // Завантажити список документів
            Worker->LoadDocList();

            // Створити об'єкт класу "Користувач"
            Reader = gcnew CReader(strUserName);
            // Завантажити список документів, що перебувають у користувача
            Reader->Load(Worker);

           // Вивести стан
           lblStatus->Text = "Для відображення списку потрібних документів
скористайтеся пошуком";
            // Обновити форму
           UpdateView();
      }
 /*********************************************************************/
private: System::Void btnFind_Click(System::Object^  sender, System::EventArgs^
e)
      {
```

```
            // Забрати з рядку пошуку пробіли, перетворити до нижнього регістра
й зберегти
            String^ strValue = txtFindString->Text->Trim()->ToLower();

            // Перевірка рядка
            if(String::IsNullOrEmpty(strValue))
            {
                  // Рядок порожня, запропонувати вивести весь список
                  if(MessageBox::Show("Рядок пошуку не завдань, вивести весь
список?",
                                                "Система УЕД",
MessageBoxButtons::YesNo,
                                                MessageBoxIcon::Question) ==
::DialogResult::Yes)
                  {
                        // Перейти на вивід списку
                        goto OUT_LIST;
                  }
                  // Завершити роботу функції
                  return;
            }

OUT_LIST:   // Виконати функцію пошуку
            listView = Worker->FindDoc(strValue);

            // Вивести стан
            lblStatus->Text = "Пошук завершено";
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnFirst_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на перший елемент
            ddCurrentIndex = 0;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnPrev_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на попередній елемент
            ddCurrentIndex ---i;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnNext_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на наступний елемент
            ddCurrentIndex ++;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void btnLast_Click(System::Object^  sender, System::EventArgs^
e)
      {
            // Установити індекс на останній елемент
            ddCurrentIndex = listView->Count - 1;
            // Обновити форму
            UpdateView();
      }
/*************************************************************************/
private: System::Void txtPosition_KeyDown(System::Object^  sender,
System::Windows::Forms::KeyEventArgs^  e)
```

```
	{
		// Натиснута клавіша "Enter"?
		if( e->KeyCode == Keys::Enter)
		{
			try
			{
				// Спробувати одержати введений номер документа для
перегляду
				ddCurrentIndex = Convert::ToInt32(txtPosition->Text) -
1;
			}
			catch(...)
			{
			// Якщо в процесі перетворення номера з рядка в число
			//  виникла помилка, те ніяк на це не реагувати
			}

			// Обновити форму
			UpdateView();
		}
		else
		{
			// Текстове поле містить символ "/"?
			if(txtPosition->Text->IndexOf("/") != -1)
			{
				// Для початку введення нового номера позиції,
необхідно
				//  очистити текстове поле
				txtPosition->Text = "";
			}
		}
	}
/*************************************************************************/
private: System::Void btnRequest_Click(System::Object^  sender,
System::EventArgs^  e)
	{
		// Запросити документ, попередньо перетворивши значення вхідного
номера документа в текстовому
		//  поле до нижнього регістра й забравши бічні пробіли
		if(Reader->RequireDoc(Worker, txtISBN->Text->ToLower()->Trim()-
>GetHashCode()))
		{
			// Запит пройшов вдало, видалити документ із поточного списку
перегляду
			listView->RemoveAt(ddCurrentIndex);

			// Обновити форму
			UpdateView();
			// Вивести стан
			lblStatus->Text = "Документ успішно отримано";
		}
		else
		{
			// Неможливо одержати документ, вивести повідомлення про
цьому
			lblStatus->Text = "Вибачте, але на даний момент немає жодного
документу по даному запиту";
		}
	}
/*************************************************************************/
private: System::Void btnReturn_Click(System::Object^  sender,
System::EventArgs^  e)
	{
		// У списку документів є обраний елемент?
		if(lstMyDocs->SelectedIndex != -1)
		{
			// Повернути обрану документ
			Reader->ReleaseDoc(Worker, lstMyDocs->SelectedIndex);
```

```
                // Обновити форму
                UpdateView();
                // Вивести стан
                lblStatus->Text = "Документ виконано";
            }
            else
            {
                // Вивести повідомлення про помилку
                lblStatus->Text = "Спочатку виберіть документ для виконання";
            }
        }
/*************************************************************************/
private: System::Void menuFileSave_Click(System::Object^  sender,
System::EventArgs^  e)
        {
             // Збереження модифікованого списку документів у файл
            Worker->SaveDocList();

            // Збереження списку документів користувача
            Reader->Save();
        }
/*************************************************************************/
private: System::Void menuFileExit_Click(System::Object^  sender,
System::EventArgs^  e)
        {
            // Закрити додаток
            Application::Exit();
        }
/*************************************************************************/
private: System::Void frmReader_FormClosed(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e)
        {
            // Закрити додаток
            Application::Exit();
        }
/*************************************************************************/
private: System::Void menuHelpAbout_Click(System::Object^  sender,
System::EventArgs^  e)
        {
            // Відобразити форму з інформацією про програму
            UsrsManager::frmAbout^ frmNew = gcnew UsrsManager::frmAbout();
            frmNew->Show();
        }
/*************************************************************************/
private: System::Void groupBox2_Enter(System::Object^  sender,
System::EventArgs^  e) {
        }
private: System::Void lblStatus_Click(System::Object^  sender,
System::EventArgs^  e) {
        }
};
}
```

Файл frmLogin.resx - xml опис форми «Вхід у систему УЕД»

```
<?xml version="1.0" encoding=" utf-8"?>
<root>
    <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
    <xsd:import namespace=" " />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string"
msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/ microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
</root>
```

Файл frmLogin.h - бібліотека для файлу frmLogin.resx

```
/*************************************************************************/
#pragma once
/*************************************************************************/
#include "frmWorker.h"
#include "frmReader.h"
#include "Usrs.h"
/*************************************************************************/
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
/*************************************************************************/
#define WORKER_LOGIN    "worker"
/*************************************************************************/
namespace UsrsManager
{
      public ref class frmLogin : public System::Windows::Forms::Form
      {
/*************************************************************************/
      private:
            CUsrs^ Usrs;                  // Керуючий об'єкт класу "Система"
/*************************************************************************/
public:     frmLogin(void)
            {
                  InitializeComponent();
                  //
                  //Додавання цього коду в конструктор
                  //
            }
/*************************************************************************/
protected: ~frmLogin()
            {
                  if (components)
                  {
                        delete components;
                  }
            }
/*************************************************************************/
      private: System::Windows::Forms::GroupBox^  groupBox;
      private: System::Windows::Forms::TextBox^  txtPassword;
      private: System::Windows::Forms::TextBox^  txtLogin;
      private: System::Windows::Forms::Label^  lblPassword;
      private: System::Windows::Forms::Label^  lblLogin;
      private: System::Windows::Forms::Button^  btnEnter;
      private: System::Windows::Forms::Button^  btnExit;
      private: System::Windows::Forms::Button^  btnDelete;
      private: System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Необхідний метод для підтримки Розроблювача - не модифікувати
            /// зміст цього методу з кодовим редактором.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->groupBox = (gcnew System::Windows::Forms::GroupBox());
                  this->txtPassword = (gcnew System::Windows::Forms::TextBox());
                  this->txtLogin = (gcnew System::Windows::Forms::TextBox());
                  this->lblPassword = (gcnew System::Windows::Forms::Label());
                  this->lblLogin = (gcnew System::Windows::Forms::Label());
                  this->btnEnter = (gcnew System::Windows::Forms::Button());
                  this->btnExit = (gcnew System::Windows::Forms::Button());
                  this->btnDelete = (gcnew System::Windows::Forms::Button());
                  this->groupBox->SuspendLayout();
                  this->SuspendLayout();
```

```
                    //
                    // groupBox
                    //
                    this->groupBox->Controls->Add(this->txtPassword);
                    this->groupBox->Controls->Add(this->txtLogin);
                    this->groupBox->Controls->Add(this->lblPassword);
                    this->groupBox->Controls->Add(this->lblLogin);
                    this->groupBox->Location = System::Drawing::Point(12, 11);
                    this->groupBox->Name = L"groupBox";
                    this->groupBox->Size = System::Drawing::Size(237, 80);
                    this->groupBox->TabIndex = 8;
                    this->groupBox->TabStop = false;
                    this->groupBox->Text = L"Параметри облікового запису";
                    //
                    // txtPassword
                    //
                    this->txtPassword->Location = System::Drawing::Point(57, 45);
                    this->txtPassword->Name = L"txtPassword";
                    this->txtPassword->PasswordChar = '*';
                    this->txtPassword->Size = System::Drawing::Size(174, 20);
                    this->txtPassword->TabIndex = 1;
                    //
                    // txtLogin
                    //
                    this->txtLogin->Location = System::Drawing::Point(57, 19);
                    this->txtLogin->Name = L"txtLogin";
                    this->txtLogin->Size = System::Drawing::Size(174, 20);
                    this->txtLogin->TabIndex = 0;
                    //
                    // lblPassword
                    //
                    this->lblPassword->AutoSize = true;
                    this->lblPassword->Location = System::Drawing::Point(6, 48);
                    this->lblPassword->Name = L"lblPassword";
                    this->lblPassword->Size = System::Drawing::Size(44, 13);
                    this->lblPassword->TabIndex = 1;
                    this->lblPassword->Text = L"Пароль";
                    //
                    // lblLogin
                    //
                    this->lblLogin->AutoSize = true;
                    this->lblLogin->Location = System::Drawing::Point(13, 19);
                    this->lblLogin->Name = L"lblLogin";
                    this->lblLogin->Size = System::Drawing::Size(33, 13);
                    this->lblLogin->TabIndex = 0;
                    this->lblLogin->Text = L"Логін";
                    //
                    // btnEnter
                    //
                    this->btnEnter->Location = System::Drawing::Point(12, 98);
                    this->btnEnter->Name = L"btnEnter";
                    this->btnEnter->Size = System::Drawing::Size(75, 23);
                    this->btnEnter->TabIndex = 2;
                    this->btnEnter->Text = L"Вхід";
                    this->btnEnter->UseVisualStyleBackColor = true;
                    this->btnEnter->Click += gcnew System::EventHandler(this,
&frmLogin::btnEnter_Click);
                    //
                    // btnExit
                    //
                    this->btnExit->Location = System::Drawing::Point(174, 98);
                    this->btnExit->Name = L"btnExit";
                    this->btnExit->Size = System::Drawing::Size(75, 23);
                    this->btnExit->TabIndex = 3;
                    this->btnExit->Text = L"Вихід";
                    this->btnExit->UseVisualStyleBackColor = true;
                    this->btnExit->Click += gcnew System::EventHandler(this,
&frmLogin::btnExit_Click);
                    //
```

```
                    // btnDelete
                    //
                    this->btnDelete->Location = System::Drawing::Point(93, 98);
                    this->btnDelete->Name = L"btnDelete";
                    this->btnDelete->Size = System::Drawing::Size(75, 23);
                    this->btnDelete->TabIndex = 4;
                    this->btnDelete->Text = L"Видалення";
                    this->btnDelete->UseVisualStyleBackColor = true;
                    this->btnDelete->Click += gcnew System::EventHandler(this,
&frmLogin::btnDelete_Click);
                    //
                    // frmLogin
                    //
                    this->AutoScaleDimensions = System::Drawing::Size(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(260, 129);
                    this->Controls->Add(this->groupBox);
                    this->Controls->Add(this->btnEnter);
                    this->Controls->Add(this->btnExit);
                    this->Controls->Add(this->btnDelete);
                    this->MaximizeBox = false;
                    this->MinimizeBox = false;
                    this->Name = L"frmLogin";
                    this->Text = L"Вхід у систему УЕД";
                    this->groupBox->ResumeLayout(false);
                    this->groupBox->PerformLayout();
                    this->ResumeLayout(false);

            }
#pragma endregion
/*************************************************************************/
private: System::Boolean PrepareLoginPassword()
            {
                    // Перетворити ім'я до нижнього регістра
                    txtLogin->Text = txtLogin->Text->ToLower();
                    // Забрати в поле ім'я бічні пробіли
                    txtLogin->Text = txtLogin->Text->Trim();

                    // Поле для введення ім'я не містить значення?
                    if(String::IsNullOrEmpty(txtLogin->Text))
                    {
                          // Видати повідомлення про те, поле не містить значення
                          MessageBox::Show("Введіть ім'я користувача.", "Usrs
Manager",
                                MessageBoxButtons::OK, MessageBoxIcon::Warning);
                          // Вийти
                          return false;
                    }

                    // Перевірити введені символи
                    for(Int32 i = 0; i < txtLogin->Text->Length; i++)
                    {
                          // Перевіряється символ
                          Char chTest = txtLogin->Text[i];

                          // Перевірка на коректність
                          if(!(chTest >= 'A' && chTest <= 'Z') &&
                               !(chTest >= 'a' && chTest <= 'z') &&
                               !(chTest >= 'A' && chTest <= 'Я') &&
                               !(chTest >= 'a' && chTest <= 'я') &&
                               !(chTest == '_'))
                          {
                               // Видати повідомлення про те, поле містить
неприпустимий символ
                               MessageBox::Show("Логін містить неприпустимі
символи!",
                                     "Usrs Manager", MessageBoxButtons::OK,
                                     MessageBoxIcon::Warning);
```

```
                        // Вийти
                        return false;
                    }
                }

                return true;
            }
/*************************************************************************/
private: System::Void btnEnter_Click(System::Object^  sender, System::EventArgs^
e)
            {
                // Підготовка значень
                if(!PrepareLoginPassword())
                {
                    return;
                }

                // Уміст поля ім'я - логін адміністратора?
                if(txtLogin->Text == WORKER_LOGIN)
                {
                    // Увійти в обліковий запис адміністратора
                    if(Usrs->WorkerLogin(txtPassword->Text))
                    {
                        // Відобразити форму адміністратора
                        UsrsManager::frmWorker^ frmNew = gcnew
UsrsManager::frmWorker;
                        frmNew->Show();
                    }
                    else
                    {
                        // Вхід неможливий, закінчити роботу функції
                        return;
                    }
                }
                else
                {
                    // Увійти в обліковий запис користувача
                    if(Usrs->ReaderLogin(txtLogin->Text, txtPassword-
>Text))
                    {
                        // Відобразити форму для користувача
                        UsrsManager::frmReader^ frmNew =
                            gcnew UsrsManager::frmReader(txtLogin-
>Text);
                        frmNew->Show();
                    }
                    else
                    {
                        // Вхід неможливий, закінчити роботу функції
                        return;
                    }
                }

                // Сховати форму входу
                this->Hide();
            }
/*************************************************************************/
private: System::Void btnExit_Click(System::Object^  sender, System::EventArgs^
e)
            {
                // Закрити додаток
                Application::Exit();
            }
/*************************************************************************/
private: System::Void btnDelete_Click(System::Object^  sender,
System::EventArgs^  e)
            {
                // Одержати подтвержение на видалення
                if(MessageBox::Show("Ви дійсно хочете зробити видалення?",
```

```
                        "Usrs Manager", MessageBoxButtons::YesNo,
                        MessageBoxIcon::Information) == ::DialogResult::Yes)
                  {
                        // Підготовка значень логіна й пароля
                        if(!PrepareLoginPassword())
                        {
                              // Якщо виникли помилки, то завершити роботу
функції
                              return;
                        }

                        // Уміст поля ім'я - логін адміністратора?
                        if(txtLogin->Text == WORKER_LOGIN)
                        {
                                    // Обліковий запис адміністратора видалити
не можна
                                    MessageBox::Show("Неможливо видалити
обліковий запис адміністратора!",
                                          "Usrs Manager", MessageBoxButtons::OK,
                                          MessageBoxIcon::Error);
                        }
                        else
                        {
                              // Видалити обліковий запис користувача
                              if(Usrs->RemoveReader(txtLogin->Text,
txtPassword->Text))
                              {
                                    // Видалення пройшло успішно
                                    MessageBox::Show("Обліковий запис успішно
вилучений",
                                          "Usrs Manager",
MessageBoxButtons::OK,
                                          MessageBoxIcon::Information);

                              }
                              else
                              {
                                    // При видаленні відбулася помилка
                                    MessageBox::Show("Неможливо видалити
обліковий запис",
                                          "Usrs Manager",
MessageBoxButtons::OK,
                                          MessageBoxIcon::Error);
                              }
                        }
                  }
            }
/*************************************************************************/
};
}
```

Файл Doc.cpp - робота з документами

```
/*****************************************************************************/
#include "StdAfx.h"
#include "Doc.h"
/*****************************************************************************/
using namespace System;
/*****************************************************************************/
// NAME:          CDoc
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
// OUTPUT:        N/D
/*****************************************************************************/
CDoc::CDoc(void)
{
      ddTotalNumber = 0;      // Загальна кількість екземплярів дорівнює 0
      ddFreeNumber = 0; // Кількість вільних екземплярів дорівнює 0
}
/*****************************************************************************/
// NAME:          GetName
// DESCRIPTION:   Функція одержання назви документів
// INPUT:         N/D
// OUTPUT:        Назва документа
/*****************************************************************************/
String^ CDoc::GetName()
{
      return strName;
}
/*****************************************************************************/
// NAME:          SetName
// DESCRIPTION:   Функція установки назви документів
// INPUT:         strValue - назва документа
// OUTPUT:        TRUE - Назва документа установлена
//                      FALSE - Назва не встановлена, рядок-параметр порожня
/*****************************************************************************/
Boolean CDoc::SetName(String^ strValue)
{
      // Перевірити, чи не порожній рядок
      if(!String::IsNullOrEmpty(strValue))
      {
            // Установити значення
            strName = strValue;
            return true;
      }
      else
      {
            return false;
      }
}
/*****************************************************************************/
// NAME:          GetAuthor
// DESCRIPTION:   Функція одержання автора документа
// INPUT:         N/D
// OUTPUT:        Автор документи
/*****************************************************************************/
String^ CDoc::GetAuthor()
{
      // Повернути значення
      return strAuthor;
}
/*****************************************************************************/
// NAME:          SetAuthor
// DESCRIPTION:   Функція установки автора документа
// INPUT:         strValue - ім'я автора документа
// OUTPUT:        N/D
/*****************************************************************************/
void CDoc::SetAuthor(String^ strValue)
```

```
{
      // Установити значення
      strAuthor = strValue;
}
/*************************************************************************/
// NAME:          GetISBN
// DESCRIPTION:   Функція одержання вхідний номер документа
// INPUT:         N/D
// OUTPUT:        вхідний номер документа
/*************************************************************************/
String^ CDoc::GetISBN()
{
      // Повернути значення
      return strISBN;
}
/*************************************************************************/
// NAME:          SetISBN
// DESCRIPTION:   Функція установки вхідний номер документа
// INPUT:         strValue - вхідний номер документа
// OUTPUT:        TRUE - вхідний номер документа встановлений
//                      FALSE - вхідний номер документа не встановлений, рядок-
параметр порожня
/*************************************************************************/
Boolean CDoc::SetISBN(String^ strValue)
{
      // Перевірити, чи не порожній рядок
      if(!String::IsNullOrEmpty(strValue))
      {
            // Установити значення
            strISBN = strValue;
            // Обчислити й установити хеш-код вхідний номер документа
            ddISBNHash = strISBN->GetHashCode();
            return true;
      }
      else
      {
            return false;
      }
}
/*************************************************************************/
// NAME:          GetTheme
// DESCRIPTION:   Функція одержання теми
// INPUT:         N/D
// OUTPUT:        Тема документи
/*************************************************************************/
String^ CDoc::GetTheme()
{
      // Повернути значення
      return strTheme;
}
/*************************************************************************/
// NAME:          SetTheme
// DESCRIPTION:   Функція установки теми
// INPUT:         strValue - тема документи
// OUTPUT:        N/D
/*************************************************************************/
void CDoc::SetTheme(String^ strValue)
{
      // Установити значення
      strTheme = strValue;
}
/*************************************************************************/
// NAME:          GetPages
// DESCRIPTION:   Функція одержання кількості сторінок
// INPUT:         N/D
// OUTPUT:        Кількість сторінок у книзі
/*************************************************************************/
Int32 CDoc::GetPages()
{
```

```
	// Повернути значення
	return ddPages;
}
/*************************************************************************/
// NAME:		SetPages
// DESCRIPTION:	Функція установки кількості сторінок
// INPUT:		ddValue - кількість сторінок
// OUTPUT:		TRUE - кількість сторінок установлена
//			FALSE - кількість сторінок установлена, неприпустиме
значення аргументу
/*************************************************************************/
Boolean	CDoc::SetPages(Int32 ddValue)
{
	// Параметр має припустиме значення?
	if(ddValue >= 0)
	{
		// Установити значення
		ddPages = ddValue;
		return true;
	}
	else
	{
		return false;
	}
}
/*************************************************************************/
// NAME:		GetTotalNumber
// DESCRIPTION:	Функція одержання загальної кількості екземплярів
// INPUT:		N/D
// OUTPUT:		Загальна кількість екземплярів документи
/*************************************************************************/
Int32 CDoc::GetTotalNumber()
{
	// Повернути значення
	return ddTotalNumber;
}
/*************************************************************************/
// NAME:		SetTotalNumber
// DESCRIPTION:	Функція установки загальне кількості екземплярів
// INPUT:		ddValue - загальна кількість екземплярів
// OUTPUT:		TRUE - кількість екземплярів установлена
//			FALSE - кількість екземплярів установлена, негативне
значення
//			аргумента або аргумент перевищує кількість екземплярів,
що перебуває в
//			користувачів
/*************************************************************************/
Boolean	CDoc::SetTotalNumber(Int32 ddValue)
{
	// Перевірка параметра
	if(ddValue >= 0 && ddValue >= ddTotalNumber - ddFreeNumber)
	{
		// Установити значення
		ddTotalNumber = ddValue;
		return true;
	}
	else
	{
		return false;
	}
}
/*************************************************************************/
// NAME:		GetFreeNumber
// DESCRIPTION:	Функція одержання кількості вільних екземплярів
// INPUT:		N/D
// OUTPUT:		Кількість вільних екземплярів
/*************************************************************************/
Int32 CDoc::GetFreeNumber()
{
```

```
      // Повернути значення
      return ddFreeNumber;
}
/*************************************************************************/
// NAME:          SetFreeNumber
// DESCRIPTION:   Функція установки кількості вільних екземплярів
// INPUT:         ddValue - кількість вільних екземплярів
// OUTPUT:        TRUE - кількість вільних екземплярів установлене
//                      FALSE - значення параметра перевищує загальна кількість
екземплярів
/*************************************************************************/
Boolean     CDoc::SetFreeNumber(Int32 ddValue)
{
      // Порівняння параметра із загальною кількістю екземплярів
      if(ddTotalNumber >= ddValue)
      {
            // Установити значення
            ddFreeNumber = ddValue;
            return true;
      }
      else
      {
            // ddFreeNumber = ddTotalNumber;
            return false;
      }
}
/*************************************************************************/
// NAME:          GetISBNHash
// DESCRIPTION: Функція одержання хеш-коду від вхідного номера документа
// INPUT:         N/D
// OUTPUT: Хеш-         Код вхідний номер документа
/*************************************************************************/
Int32 CDoc::GetISBNHash()
{
      // Повернути хеш-значення вхідного номера документа
      return ddISBNHash;
}
/*************************************************************************/
// NAME:          DecFreeNumber
// DESCRIPTION:   Функція зменшення кількості вільних екземплярів
// INPUT:         N/D
// OUTPUT:        TRUE - кількість вільних екземплярів декрементовано
//                FALSE - зменшення неприпустимо, кількість вільних екземплярів
дорівнює 0
/*************************************************************************/
Boolean     CDoc::DecFreeNumber()
{
      // Кількість вільних екземплярів не дорівнює 0?
      if(ddFreeNumber != 0)
      {
            // Зменшити значення
            ddFreeNumber ---i;
            return true;
      }
      else
      {
            return false;
      }
}
/*************************************************************************/
// NAME:          IncFreeNumber
// DESCRIPTION:   Функція збільшення кількості вільних екземплярів
// INPUT:         N/D
// OUTPUT:        TRUE - кількість вільних екземплярів інкрементовано
//                FALSE - значення неприпустимо, кількість вільних екземплярів
дорівнює
//                      загальній кількості екземплярів
/*************************************************************************/
Boolean     CDoc::IncFreeNumber()
```

```
{
	// Кількість вільних екземплярів не дорівнює загальній кількості
екземплярів?
	if(ddFreeNumber != ddTotalNumber)
	{
		// Збільшити значення
		ddFreeNumber ++;
		return true;
	}
	else
	{
		return false;
	}
}
/*****************************************************************************/
```

Файл Doc.h - бібліотека для файлу Doc. cpp

```
     #pragma once
/*************************************************************************/
using namespace System;
/*************************************************************************/
ref class CDoc                                          // Клас "Документ"
{
private:
     String^ strName;                                   // Назва
     String^ strAuthor;                                 // Автор
     String^ strISBN;                                   // вхідний номер документа
     String^ strTheme;                                  // Тема
     Int32 ddPages;                                     // Кількість сторінок
     Int32 ddTotalNumber;                               // Загальна кількість
екземплярів
     Int32 ddFreeNumber;                                // Кількість вільних
(перебувають в
                                                        // системі) екземплярів
     Int32 ddISBNHash;                                  // Значення хеш-функції від
вхідний номер документа

public:
     CDoc(void);                                        // Конструктор класу

     String^ GetName();                                 // Функція одержання назви
документів
     String^ GetAuthor();                               // Функція одержання автора
документа
     String^ GetISBN();                                 // Функція одержання
вхідний номер документа
     String^ GetTheme();                                // Функція одержання теми
     Int32  GetPages();                                 // Функція одержання
кількості сторінок
     Int32  GetTotalNumber();                           // Функція одержання
загальне кількості екземплярів
     Int32  GetFreeNumber();                            // Функція одержання
кількості вільних екземплярів
     Int32  GetISBNHash();                              // Функція одержання хеша-
коду від вхідний номер документа

     Boolean SetName(String^ strValue);         // Функція установки назви
документів
     void SetAuthor(String^ strValue);          // Функція установки автора
документа
     Boolean SetISBN(String^ strValue);         // Функція установки вхідний
номер документа
     void SetTheme(String^ strValue);           // Функція установки теми
     Boolean      SetPages(Int32 ddValue);            // Функція установки
кількості сторінок
     Boolean      SetTotalNumber(Int32 ddValue);      // Функція установки
загальне кількості екземплярів
     Boolean      SetFreeNumber(Int32 ddValue); // Функція установки кількості
вільних екземплярів

     Boolean      DecFreeNumber();                    // Функція зменшення
кількості вільних екземплярів
     Boolean      IncFreeNumber();                    // Функція збільшення
кількості вільних екземплярів
};
/*************************************************************************/
```

Файл Usrs.cpp - робота з обліковими записами

```
    /*****************************************************************************/
#include "StdAfx.h"
#include "Usrs.h"
/*****************************************************************************/
using namespace System;
using namespace System::IO;
using namespace System::Windows::Forms;
using namespace System::Collections;
/*****************************************************************************/
#define USER_LIST_FILE              "USERLIST.TXT"
#define WORKER_NAME                       "ADMINISTRATOR"
#define WORKER_FILE_EXTENSION ".LMW"
#define READER_FILE_EXTENSION ".LMR"
/*****************************************************************************/
// NAME:          CUsrs
// DESCRIPTION:   Конструктор класу
// INPUT:         N/D
/*****************************************************************************/
CUsrs::CUsrs(void)
{
}
/*****************************************************************************/
// NAME:          WorkerLogin
// DESCRIPTION:   Функція входу й завантаження даних про адміністратора
// INPUT:         strPassword - пароль адміністратора
/*****************************************************************************/
Boolean CUsrs::WorkerLogin(System::String ^strPassword)
{
    // Перевірити існування файлу зі списком імен і паролів
    if(File::Exists(USER_LIST_FILE))
    {
     // Відкрити файл
     StreamReader^ srUserList = gcnew StreamReader(USER_LIST_FILE);

     // Завантажити перший рядок з ім'ям адміністратора
     String^ strLogin = srUserList->ReadLine();

     // Зрівняти введене ім'я з ім'ям у файлі
     if(String::Compare(strLogin, WORKER_NAME) != 0)
     {
      // Логіни не збігаються
      MessageBox::Show("Обліковий запис не знайдено.");
      // Повернути помилку
      return false;
     }

     // Завантажити другий рядок з паролем адміністратора
     String^ strPasswordInFile = srUserList->ReadLine();

     // Закрити файл
     srUserList->Close();

     // Зрівняти введений пароль із паролем у файлі
     if(String::Compare(strPassword, strPasswordInFile) != 0)
     {
      // Паролі не збігаються
      MessageBox::Show("Невірний пароль!", "Система УЕД",
            MessageBoxButtons::OK, MessageBoxIcon::Error);
      // Повернути помилку
      return false;
     }

     // Вхід виконаний
     return true;
    }
    else
```

```
    {
     // Створити файл зі списком користувачів
     StreamWriter^ swUserList = gcnew StreamWriter(USER_LIST_FILE);

     // Зберегти ім'я користувача й пароль
     swUserList->WriteLine(WORKER_NAME);
     swUserList->WriteLine(strPassword);

     // Закрити файл
     swUserList->Close();
    }

    return true;
}
/*************************************************************************/
// NAME:          ReaderLogin
// DESCRIPTION:   Функція входу й завантаження даних про користувача
// INPUT:         strName - ім'я облікового запису користувача
//          strPassword - пароль для входу
/*************************************************************************/
Boolean CUsrs::ReaderLogin(System::String^ strName, System::String^ strPassword)
{
    // Прапор удалого входу
    Boolean IsLoginValid = false,
      IsPasswordValid = false;

    // Перевірка існування файлу зі списком користувачів
    if(!File::Exists(USER_LIST_FILE))
    {
     // Видати повідомлення про помилку
     MessageBox::Show("Помилка! Для початку роботи необхідно створити обліковий
запис.",
      "Система УЕД", MessageBoxButtons::OK, MessageBoxIcon::Error);
     // Повернути помилку
     return false;
    }

    // Відкрити файл
    StreamReader^ srUserList = gcnew StreamReader(USER_LIST_FILE);

    do
    {
     // Завантажити ім'я й пароль
     String^ strNameInFile = srUserList->ReadLine();
     String^ strPasswordInFile = srUserList->ReadLine();

     if(String::Compare(strName, strNameInFile) == 0)
     {
      // Ім'я користувача знайдене
      IsLoginValid = true;

      // Зрівняти введений пароль із паролем у файлі
      if(String::Compare(strPassword, strPasswordInFile) == 0)
      {
            // Пароль збігається
            IsPasswordValid = true;
            // Закінчити цикл пошуку
            break;
      }
      else
      {
            // Ім'я знайдене, але пароль не збігається
            IsPasswordValid = false;
            // Закінчити цикл пошуку
            break;
      }

     }
    }
```

```
    while(srUserList->EndOfStream == false);

    // Закрити файл
    srUserList->Close();

    // Якщо логін збігається, а пароль не збігається, то перервати виконання
    if(IsLoginValid && !IsPasswordValid)
    {
     // Повідомити користувача
     MessageBox::Show("Невірний пароль!", "Система УЕД",
      MessageBoxButtons::OK, MessageBoxIcon::Error);
     // Повернути помилку
     return false;
    }

    // Якщо ні логін, ні пароль не збігаються, то створити новий обліковий запис
    if(!IsLoginValid && !IsPasswordValid)
    {
     // Запропонувати створити пользоватею новий обліковий запис
     if (MessageBox::Show("Облікового запису з таким ім'ям користувача не
знайдено, створити новий обліковий запис?",
                          "Система УЕД", MessageBoxButtons::YesNo,
                          MessageBoxIcon::Question) == DialogResult::Yes)
     {
      // Створити новий обліковий запис
      if(!AddReader(strName, strPassword))
      {
            MessageBox::Show("При створенні нового облікового запису виникла
помилка", "Система УЕД", MessageBoxButtons::OK,
                                    MessageBoxIcon::Error);
            // Не виконувати вхід
            return false;
      }
     }
     else
     {
      // Не виконувати вхід
      return false;
     }
    }

    return true;
}
/*********************************************************************/
// NAME:         AddReader
// DESCRIPTION:  Функція додавання нового облікового запису користувача
// INPUT:        strName - ім'я користувача
//          strPassword - пароль
/*********************************************************************/
Boolean CUsrs::AddReader(System::String ^strName, System::String ^strPassword)
{
    // Створити файловий потік
    FileStream^ fsUserList = gcnew FileStream(USER_LIST_FILE, FileMode::Append);
    // Відкрити файл зі списком користувачів
    StreamWriter^ swUserList = gcnew StreamWriter(fsUserList);

    // Записати ім'я й пароль
    swUserList->WriteLine(strName + Convert::ToChar(13) + Convert::ToChar(10) +
strPassword);

    // Закрити файл
    swUserList->Close();
    // Закрити файловий потік
    fsUserList->Close();

    // Створити файловий потік
    FileStream^ fsReaderFile = gcnew FileStream(strName->ToUpper() +
READER_FILE_EXTENSION,
                                                FileMode::CreateNew);
```

```
    // Відкрити файл користувача
    StreamWriter^ swReaderFile = gcnew StreamWriter(fsReaderFile);
    // Кількість документів у користувача дорівнює 0
    swReaderFile->WriteLine("0");
    // Закрити файл
    swReaderFile->Close();
    // Закрити файловий потік
    fsReaderFile->Close();

    return true;
}
/*************************************************************************/
// NAME:          RemoveReader
// DESCRIPTION:   Функція видалення облікового запису користувача
// INPUT:         strName - ім'я облікового запису
//                strPassword - пароль
/*************************************************************************/
Boolean CUsrs::RemoveReader(System::String ^strName, System::String
^strPassword)
{
    // Створити файловий потік
    FileStream^ fsUserList = gcnew FileStream(USER_LIST_FILE, FileMode::Open);

    // Пошук позиції рядка із заданим ім'ям користувача

    // Тимчасовий рядок
    String^ strTmp = "";
    // Лічильник символів
    Int64 dqCounter = 0;
    // Непарні рядки - рядка з іменами
    Boolean IsLoginString = true;
    // Цикл пошуку
    do
    {
     // Завантажити байт
     Char dbChar = fsUserList->ReadByte();
     // Якщо байт службовий або досягнуть кінець файлу
     if(dbChar == 0x0D || fsUserList->Position == fsUserList->Length)
     {
      // Якщо рядок ім'я
      if(IsLoginString)
      {
            // Зрівняти ім'я користувача з виділеним рядком
            if(String::Compare(strTmp, strName) == 0)
            {
                  // Скорегувати покажчик
                  dqCounter -= strTmp->Length;
                  // Перервати виконання циклу
                  break;
            }
      }

      // Обнулити рядок
      strTmp = "";
      // Інвертувати прапор рядка ім'я
      IsLoginString = !IsLoginString;
      // Якщо не кінець файлу
      if(fsUserList->Position + 1 != fsUserList->Length)
      {
            // Пропустити службовий символ
            fsUserList->ReadByte();
            dqCounter ++;
      }
     }
     else
     {
      // Додати лічений символ до тимчасового рядка
      strTmp += dbChar;
     }
```

```
        // Збільшить покажчик
        dqCounter ++;
      }
      // Продовжувати, поки не кінець файлу
      while(fsUserList->Position != fsUserList->Length);

      // Переміститися у файлі до знайденого рядка
      fsUserList->Position = dqCounter;
      // Заповнити ім'я пробілами
      for(Int32 i = 0; i < strTmp->Length; i++)
      {
        fsUserList->WriteByte(' ');
      }

      // Закрити файловий потік
      fsUserList->Close();
      // Видалити файл користувача
      File::Delete(strName + READER_FILE_EXTENSION);

      return true;
}
/*****************************************************************************/
```

Файл Usrs.h - бібліотека для файлу Usrs.cpp

```
     /*************************************************************************
/
#pragma once
/*************************************************************************/
using namespace System;
/*************************************************************************/
ref class CUsrs   // Клас реєстрації, видалення й авторизації користувачів
{
public:
     // Конструктор класу
     CUsrs(void);

     // Функція додавання нового облікового запису користувача
     Boolean AddReader(System::String^ strName, System::String^ strPassword);
     // Функція видалення облікового запису користувача
     Boolean RemoveReader(System::String^ strName, System::String^ strPassword);

     // Функція входу й завантаження даних про користувача
     Boolean ReaderLogin(System::String^ strName, System::String^ strPassword);
     // Функція входу й завантаження даних про адміністратора
     Boolean WorkerLogin(System::String^ strName);
};
/*************************************************************************/
```

Файл frmAbout.resx - xml опис форми «про програму...»

```
 <?xml version="1.0" encoding=" utf-8" ?>
- <root>
- <xsd:schema id="root" xmlns="" xmlns:xsd=" " xmlns:msdata="urn: schemas-
microsoft-com: xml-msdata">
  <xsd:import namespace=" " />
- <xsd:element name="root" msdata:IsDataSet="true">
- <xsd:complexType>
- <xsd:choice maxOccurs="unbounded">
- <xsd:element name="metadata">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="name" use="required" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
  <xsd:attribute name="mimetype" type="xsd:string" />
  <xsd:attribute ref="xml:space" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="assembly">
- <xsd:complexType>
  <xsd:attribute name="alias" type="xsd:string" />
  <xsd:attribute name="name" type="xsd:string" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="data">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1"
/>
  <xsd:element name="comment" type="xsd:string" minOccurs="0" msdata:Ordinal="2"
/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" msdata:Ordinal="1"
/>
  <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
  <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
  <xsd:attribute ref="xml:space" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="resheader">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1"
/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  </xsd:complexType>
  </xsd:element>
  </xsd:choice>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
- <resheader name="resmimetype">
  <value>text/ microsoft-resx</value>
  </resheader>
- <resheader name="version">
  <value>2.0</value>
  </resheader>
- <resheader name="reader">
  <value>System.Resources.ResXResourceReader, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
- <resheader name="writer">
```

```
  <value>System.Resources.ResXResourceWriter, System.Windows.Forms,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
- <data name="txtInfo.Text" xml:space="preserve">

  <value>МАГІСТЕРСЬКА РОБОТА
На тему: Дослідження та програмна реалізація системи електронного документообігу
кафедри кібербезпеки та програмного забезпечення
Керівник: Смірнов О.А.
Розробив: студент Горяний Дмитро Володимирович
гр. КН-22МЗ
ЦНТУ
м. Кропивницький 2023</value>
  </data>
  </root>
```

Файл frmAbout.h - бібліотека для файлу frmAbout.resx

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;


namespace UsrsManager
{

	/// <summary>
	/// Summary for frmAbout
	///

	/// </summary>
	public ref class frmAbout : public System::Windows::Forms::Form
	{
/*************************************************************************/
public:	frmAbout(void)
		{
			InitializeComponent();
			//
			//Додавання цього коду в конструктор
			//
		}
/*************************************************************************/
protected: ~frmAbout()
		{
			if (components)
			{
				delete components;
			}
		}
/*************************************************************************/
	private: System::Windows::Forms::Button^  btnOk;
	private: System::Windows::Forms::TextBox^  txtInfo;

	private: System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Необхідний метод для підтримки Розроблювача - не модифікувати
		/// зміст цього методу з кодовим редактором.
		/// </summary>
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^  resources =
(gcnew System::ComponentModel::ComponentResourceManager(frmAbout::typeid));
			this->btnOk = (gcnew System::Windows::Forms::Button());
			this->txtInfo = (gcnew System::Windows::Forms::TextBox());
			this->SuspendLayout();
			//
			// btnOk
			//
			this->btnOk->Location = System::Drawing::Point(205, 216);
			this->btnOk->Name = L"btnOk";
			this->btnOk->Size = System::Drawing::Size(75, 23);
			this->btnOk->TabIndex = 0;
			this->btnOk->Text = L"OK";
			this->btnOk->UseVisualStyleBackColor = true;
			this->btnOk->Click += gcnew System::EventHandler(this,
&frmAbout::btnOk_Click);
			//
```

```
                    // txtInfo
                    //
                    this->txtInfo->Location = System::Drawing::Point(7, 12);
                    this->txtInfo->Multiline = true;
                    this->txtInfo->Name = L"txtInfo";
                    this->txtInfo->ReadOnly = true;
                    this->txtInfo->Size = System::Drawing::Size(273, 198);
                    this->txtInfo->TabIndex = 3;
                    this->txtInfo->Text = resources->GetString(L"txtInfo.Text");
                    this->txtInfo->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
                    //
                    // frmAbout
                    //
                    this->AutoScaleDimensions = System::Drawing::Size(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(287, 243);
                    this->Controls->Add(this->btnOk);
                    this->Controls->Add(this->txtInfo);
                    this->MaximizeBox = false;
                    this->MinimizeBox = false;
                    this->Name = L"frmAbout";
                    this->Text = L"Про програму...";
                    this->ResumeLayout(false);
                    this->PerformLayout();

            }
#pragma endregion
/*************************************************************************/
private: System::Void btnOk_Click(System::Object^  sender, System::EventArgs^
e)
             {
                    // Закрити форму
                    frmAbout::Close();
             }
/*************************************************************************/
      };
}
```