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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____Генкуленко О. Ю.
«____» _____ 20 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є. В.
«____» _____ 20 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Генкуленку Олексію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів*

керівник роботи *Мелешко Єлизавета Владиславівна, д-р техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №12-02 від 05.01.2023 року

2. Строк подання студентом роботи до захисту *20.05.2023 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів*

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

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

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

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

6. Дата видачі завдання « 21 » грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	22.05.2023 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Генкуленко О.Ю. Програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначено для реалізації системи кібербезпеки асиметричного шифрування електронних документів.

Метою роботи є створення системи кібербезпеки для асиметричного шифрування електронних документів.

Результат роботи – програмна реалізація системи кібербезпеки для асиметричного шифрування електронних документів.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

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 на мові програмування C#.

Ключові слова: кібербезпека, захист інформації, шифрування, електронний документообіг

ABSTRACT

Henkulenko O.Yu. Cybersecurity system software for asymmetric encryption of electronic documents. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi 2023

In this bachelor's qualification work, software that is designed for system for asymmetric encryption of electronic documents was developed.

The purpose of the development is the software of the system for asymmetric encryption of electronic documents.

The result of the work is the software implementation of the system for asymmetric encryption of electronic documents.

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

A user-friendly interface is developed. The instructions for working with the software are provided.

The program can be used on an IBM PC running Windows 10/11.

The program was developed in the C# programming language.

Keywords: cybersecurity, information protection, encryption, electronic document management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АР – (application programming interface) інтерфейс програмування застосунків, набір чітко визначених методів для взаємодії різних компонентів.

DES – Data Encryption Standard, алгоритм симетричного шифрування.

RSA – алгоритм асиметричного шифрування, абревіатура від прізвищ його авторів Rivest, Shamir, Adleman.

БД – база даних.

ОС – операційна система.

РКІ – інфраструктура відкритих ключів.

СУБД – система управління базами даних.

ВСТУП

Сьогоднішній світ залежить від електронної інформації, що зберігається та обробляється в комп'ютерних системах та передається через мережі зв'язку. Що створює потребу в захисті цієї інформації від зловмисників, які можуть використовувати її для кримінальних або шкідливих цілей. Саме тому виникає необхідність у розробці систем кібербезпеки, які забезпечують захист електронної інформації від таких загроз.

Одним із методів забезпечення кібербезпеки електронної інформації є асиметричне шифрування даних. Цей тип методів шифрування використовує пару ключів: публічний та приватний. Публічний ключ відкритий для доступу всіх користувачів, які можуть зашифрувати повідомлення та відправити його власнику приватного ключа. Приватний ключ використовується для розшифрування отриманого повідомлення, зашифрованого відповідним публічним ключем. Таким чином, забезпечується безпека обміну даними між двома сторонами, не передбачаючи необхідності передавати приватний ключ по мережі зв'язку.

Для реалізації асиметричного шифрування даних необхідне відповідне програмне забезпечення. Це програмне забезпечення повинне забезпечувати зручний та безпечний доступ до публічних та приватних ключів, а також забезпечувати можливість шифрування та розшифрування електронних документів з використанням цих ключів.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів.

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

– Дослідження існуючих методів асиметричного шифрування та забезпечення інформаційної безпеки електронних документів.

– Розробка алгоритмів для системи кібербезпеки для асиметричного шифрування електронних документів.

– Програмна реалізація системи кібербезпеки для асиметричного шифрування електронних документів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі асиметричного шифрування та забезпечення інформаційної безпеки електронних документів.

Тому, розробка програмного забезпечення системи кібербезпеки для асиметричного шифрування електронних документів є важливою задачею у сфері забезпечення безпеки і конфіденційності інформації в електронному вигляді та вирішувалася у даній кваліфікаційній бакалаврській роботі. Таке програмне забезпечення може застосовуватись у різних сферах, де потрібне захищене зберігання і передача даних, таких як банківські установи, медичні заклади, юридичні компанії, державні органи та інші.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система асиметричного шифрування електронних документів може застосовуватися з метою забезпечення конфіденційності, цілісності та доступності цифрової інформації, що зберігається, передається та обробляється у комп'ютерних мережах. Завдяки системі асиметричного шифрування, документи можуть бути передані через комп'ютерні мережі у безпечному форматі, таким чином зменшуючи ризик їх втрати або розголошення.

Одним з основних призначень системи асиметричного шифрування електронних документів є захист від несанкціонованого доступу до інформації. За допомогою публічного ключа можна шифрувати дані таким чином, що їх можна розшифрувати тільки з використанням відповідного приватного ключа. Це дозволяє зберегти конфіденційність даних під час їх передачі через мережу Інтернет та локальні комп'ютерні мережі.

Другим важливим призначенням системи асиметричного шифрування є забезпечення цілісності даних, тобто захист від несанкціонованої модифікації даних під час їх передачі, а також забезпечення автентифікації користувачів. В такому разі система асиметричного шифрування може використовуватися для створення та перевірки цифрових підписів, що будуть підтверджувати цілісність, незмінність та авторство документів. В такому разі при використанні системи асиметричного шифрування підписується не весь документ, а лише його хеш-сума, що дозволяє перевірити, чи було змінено документ під час його передачі. Якщо хеш-сума не співпадає з оригінальною, то це свідчить про те, що документ було змінено при передачі.

1.2 Область застосування

Асиметричне шифрування є одним з найбільш надійних та безпечних способів захисту конфіденційної інформації. З його допомогою можна зашифрувати електронні документи таким чином, щоб тільки отримувач, який має відповідний приватний ключ, міг розшифрувати їх. Також на його основі можна створювати електронні цифрові підписи для перевірки авторства та цілісності документів.

Цифровий підпис електронних документів на основі асиметричного шифрування стає все більш актуальним у сучасному світі, оскільки він забезпечує достовірність, цілісність та незмінність інформації, що передається.

Область застосування програмного забезпечення для шифрування та створення цифрового підпису електронних документів досить широка і включає такі основні напрямки:

– Електронна комерція та електронний банкінг. Для захисту та підтвердження автентичності електронних документів, таких як контракти, рахунки та інші документи, пов’язані з онлайн-торгівлею.

– Юридичні послуги та електронне врядування. Шифрування та цифрові підписи використовуються у сфері електронного врядування для надання послуг громадянам та організаціям, автентифікації користувачів та забезпечення конфіденційності даних.

– Медицина та страхування. Використання шифрування та цифрового підпису дозволяє лікарям, аптекам та страховим компаніям обмінюватися конфіденційною медичною інформацією, забезпечуючи її приватність, цілісність та автентичність.

– тощо.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Для захисту електронних документів застосовують шифрування та цифровий підпис. Шифрування забезпечує конфіденційність передаваємої по мережі інформації. Цифровий підпис забезпечує цілісність, захист від модифікацій документа та підтверджує авторство даних.

Для шифрування даних можуть використовуватися як симетричні, так і асиметричні алгоритми шифрування. При цьому слід зазначити, що симетричні алгоритми більш швидкі, тож краще підходять для шифрування великих файлів, але, на жаль, при їх застосуванні виникає проблема передачі секретного ключа шифрування одержувачу зашифрованих даних. При використанні симетричних алгоритмів шифрування електронних документів найчастіше секретний ключ для передачі по незахищеному каналу зв’язку шифрують асиметричними алгоритмами шифрування. Асиметричне шифрування дозволяє вирішити проблему з передачею ключа шифрування, оскільки використовує два типи ключів: закритий та відкритий. Дані, які зашифровані відкритим ключем одержувача, можна розшифрувати тільки відповідним його закритим ключем.

Для цифрового підпису використовуються тільки асиметричні алгоритми шифрування в поєднанні з хеш-функціями. Дані, які підписані закритим ключем відправника, перевіряються його відкритим ключем, і тільки тоді перевірка дає позитивний результат, якщо файл був підписаний відповідним відправнику закритим ключем та не був модифікований після. Перед підписанням електронного документу його попередньо можна зашифрувати, таким чином забезпечивши і конфіденційність, і достовірність даних.

Одним з важливих елементів цифрового підпису електронних документів є створення відкритих та закритих ключів. Цим як правило займається так званий Центр сертифікатів. Кожен користувач одержує сертифікат, що містить відкритий ключ та загальнодоступну інформацію про користувача і відповідний закритий ключ. Сертифікати користувачів публікуються у відкритому доступі. Дані з них (відкриті ключі) можна використовувати для шифрування файлів відповідним одержувачам, або для перевірки документів, підписаних їх цифровими підписами. Закриті ключі, зберігаються у повному секреті у їх власників.

Для реалізації правильної роботи центрів сертифікатів використовується так звана інфраструктура відкритих ключів (PKI). Розглянемо принципи її функціонування.

У випадку, коли необхідність впровадження інфраструктури публічного ключа (PKI) в організації стає необхідною реальністю, виникає складне завдання вибору конкретного реалізаційного рішення для PKI. Вибір реалізації впливає на можливість інтеграції з наявною структурою і впливає на майбутню інформаційну інфраструктуру організації, яка повинна базуватися на впровадженому PKI.

З метою уникнення негативних наслідків від вибору рішення, що здійснюється під впливом рекламних матеріалів або ціни продукту, проведено порівняння провідних рішень для створення PKI. Також була здійснена спроба позиціонувати ці рішення на певні сегменти ринку.

Для проведення порівняння були обрані рішення, що задовольняють основним вимогам, які ставляться до корпоративних продуктів:

– Поділ ключів шифрування й електронного цифрового підпису.

– Підтримка відкритих стандартів.

– Підтримка або намір про підтримку Українських стандартів шифрування й електронного цифрового підпису.

– Висока продуктивність і надійність рішення.

– Поділ ролей адміністратора системи, адміністратора безпеки, аудиторів і кінцевих користувачів (відповідно до профілю захисту).

– Підтримка великої кількості користувачів.

– Масштабована, гетерогенна й легко розширювана архітектура.

– Гнучка й стандартизована система керування, аудита й генерації звітів.

– Наявність персонального менеджера сертифікатів (клієнтської частини).

– Підтримка системи однократної автентифікації.

Серед рішень, доступних на міжнародному ринку PKI, найбільш повністю відповідають вищезазначеним параметрам такі продукти: Entrust Authority 6.0, iPlanet Certificate Management System 4.7, Microsoft Certificate Server 2.0, RSA Keon 5.7, UniCERT PKI 3.5.3.

Порівняння цих продуктів вироблялося за наступними параметрами:

1. Архітектура рішення:

– Модульність.

– Масштабованість.

– Модель довіри.

2. Функціональність продукту:

– Керування сертифікатами.

– Керування списками відкликання сертифікатів.

– Адміністрування

3. Підтримувані криптографічні стандарти:

– Міжнародні криптографічні стандарти.

– Українські стандарти шифрування й ЕЦП.

4. Функції, виконувані клієнтською частиною.

5. Документація і якість технічної підтримки.

6. Засоби розробки.

При порівнянні за параметром модульність був розглянутий склад обраних рішень, здатний забезпечити повнофункціональну роботу

інфраструктури відкритих ключів, описану робочою групою PKIX (Public Key Infrastructure Working Group).

Entrust PKI

Entrust/Authority - це сертифікатний центр, який входить до інфраструктури Entrust/PKI і використовується для випуску і керування сертифікатами, створення пар ключів (відкритий і закритий) для шифрування даних, управління базою даних інфраструктури відкритих ключів і встановлення політики безпеки для організації в цілому.

Entrust/RA - це центр реєстрації, який надає графічний інтерфейс для єдиного керування всіма компонентами інфраструктури відкритих ключів.

Entrust/Authority Database - це база даних, в якій зберігається вся інформація про користувачів і інфраструктуру Entrust/Authority.

Entrust/Profile Server - це сервіс, який забезпечує безпечну роботу користувачів у мережі і централізоване зберігання профілів віддалених користувачів.

Entrust/Timestamp - це сервіс, який забезпечує безпечну мітку часу і використовується для підтримки додатків, які потребують точного часу, а також для забезпечення незаперечності транзакцій.

Entrust/AutoRA - це сервіс, який призначений для корпорацій з великою кількістю співробітників і забезпечує безпечний спосіб додавання користувачів до системи Entrust/PKI без участі адміністраторів.

Entrust/WebConnector - це сервіс, який надає можливість отримати сертифікат для ідентифікації веб-клієнтів, веб-серверів, підпису об’єктів та VPN (IPSec) на базі Microsoft Windows.

Entrust/VPNConnector – це сервіс, що забезпечує цифровими сертифікатами маршрутизатори, міжмережні екрани та інші пристрої, що забезпечують віддалений доступ, і гарантує доступ до перерахованих вище пристроїв тільки за наявності дійсного цифрового сертифікату користувача.

iPlanet CMS

Certificate Manager – це сертифікаційний центр, який об'єднує функції випуску, відкликання та підтримки бази виданих сертифікатів, а також періодичне створення списків відкликання сертифікатів. У структурі з одним центром, Certificate Manager обробляє клієнтські запити, а в розподіленій структурі PKI вони надходять від Registration Manager.

Registration Manager – це компонент структури PKI, на який Certificate Manager у розподіленій структурі PKI передає функції автентифікації та складання запиту на сертифікацію.

Data Recovery Manager – це сервіс, який забезпечує збереження та відновлення секретних ключів користувачів.

Online Certificate Status Manager – це сервіс, який забезпечує перевірку статусу сертифіката в режимі реального часу.

Microsoft CS

Інфраструктура відкритих ключів Microsoft може бути розглянута на двох рівнях, залежно від ступеня інтеграції з Active Directory:

1. Рівень підприємства: Цей рівень тісно інтегрований з Active Directory і призначений для підтримки системи безпеки дерева/лісу домена. Включає такі компоненти як шифрована файлова система, ідентифікація користувачів домена та система безпечних комунікацій. На рівні підприємства можуть бути наступні модулі:

- Enterprise Root CA: Це засвідчуючий центр доменної ієрархії Windows 7.

- Enterprise Subordinate CA: Цей модуль видає сертифікати комп'ютерам і користувачам домена Windows 7.

- Key Management Server: Цей сервер відповідає за зберігання та відновлення ключів.

2. Рівень автономної організації: Цей рівень призначений для підтримки інфраструктури відкритих ключів поза доменною системою Windows 7. Включає наступні компоненти:

- Stand-alone Root CA: Це засвідчуючий центр для автономної організації.

- Stand-alone Subordinate CA: Цей модуль видає сертифікати поза структурою домена Windows 7.

- Кожен з цих рівнів відповідає за різні аспекти управління сертифікатами і ключами в залежності від інтеграції з Active Directory та потреб організації.

RSA Keon

RSA Keon CA є засвідчуючим центром, який відповідає за випуск та відкликання сертифікатів, підтримку бази виданих сертифікатів і періодичне створення списків відкликання. Крім цього, RSA Keon CA може безпосередньо обробляти клієнтські запити або приймати їх від RSA Keon RA.

RSA Keon RA є компонентом структури RSA Keon і виконує функції по автентифікації та складанню запиту на сертифікацію. У розподіленій структурі PKI, RSA Keon CA переносить ці функції на RSA Keon RA.

RSA Keon Security Server є сервером, який відповідає за автентифікацію та авторизацію користувачів, перевірку дійсності сертифіката, зберігання мандатів користувачів і агентів, профілів користувачів, а також за автентифікацію та аудит.

RSA Keon Agents є модулями, які забезпечують єдинократну автентифікацію в інфраструктурі RSA Keon, а також шифрування сесій.

Кожен з цих компонентів відіграє важливу роль у функціонуванні інфраструктури RSA Keon, забезпечуючи безпеку та надійність управління сертифікатами та автентифікацією.

UniCERT PKI

Unicert PKI складається з базового модуля "Основний", який є необхідним для ядра системи. Поліпшені та розширені модулі представляють собою надбудову над ядром і надають додаткові сервіси Інфраструктури Відкритих Ключів.

Модуль "Основний" містить базові технології для розгортання інфраструктури відкритих ключів.

Certificate Authority (CA) є засвідчуючим центром, який забезпечує керування сертифікатами відкритих ключів протягом усього їхнього життєвого циклу.

Certificate Authority Operator (CAO) є адміністраторським інтерфейсом для управління Центром Сертифікації, що контролює всі адміністративні функції як кореневого, так і підпорядкованого CA.

Registration Authority (RA) є Центром реєстрації, який забезпечує підтвердження дійсності ідентифікаторів користувачів.

Registration Authority Operator (RAO) є адміністраторським інтерфейсом для Центра Реєстрації, який забезпечує перевірку та схвалення заявок на отримання сертифіката.

Gateway є сервісом, який приймає віддалені запити на отримання сертифіката (через електронну пошту, для систем VPN або з веб-браузерів) і розподіляє отримані сертифікати.

Token Manager є сервісом, який підтримує криптографічні апаратні пристрої.

Поліпшений модуль розширює функціональність системи і забезпечує безпечну роботу й віддалене адміністрування.

WebRAO є сервісом, який забезпечує просте й безпечне підтвердження сертифікатів з використанням стандартного веб-браузера.

WebRAO Server є сервером, який дозволяє операторам підключатися до віддаленої служби реєстрації через Інтернет.

Advanced Registration Module (ARM) є модулем, який автоматизує випуск великої кількості сертифікатів і забезпечує підтримку централізованих політик безпеки.

Key Archive Server (KAS) є сервером, який здійснює архівування, зберігання й відновлення особистих ключів шифрування, зменшуючи ризики втрати даних і затрати на відновлення.

Advanced Publishing Module (APM) є модулем, який дозволяє використовувати Active Directory для зберігання сертифікатів і списків відкликання сертифікатів (CLR).

Розширений модуль вносить компоненти зв'язку з іншими програмними продуктами й рішеннями до структури PKI.

Attribute Certificate Server (ACS) є сервером, який забезпечує контроль доступу до ресурсів на основі ролей, підвищуючи ефективність і гнучкість PKI-рішень.

Timestamp Server (TSS) є сервером, який надає послуги перевірки існування документа в певний момент часу.

Roaming Server є сервером, який дозволяє користувачам підписувати транзакції з використанням секретного ключа без апаратних засобів.

При порівнянні за критерієм "модульність", особливо привабливими виглядають рішення Entrust і UniCERT, які надають модулі для всіх областей використання інфраструктури відкритих ключів. Вони вирізняються наявністю сервісів часової мітки, підтримкою "бродячих" користувачів і протоколу бездротових додатків, спеціально призначених для забезпечення безпечних бездротових транзакцій. Ці рішення є унікальними у цьому аспекті порівняння.

Таким чином, можна зробити наступні висновки щодо різних додатків інфраструктури відкритих ключів (PKI).

Entrust PKI

Даний продукт рекомендується для використання в секторах корпорації й середніх компаній, у силу своєї масштабованості, підтримки гетерогенних середовищ, підтримки гібридної моделі довіри, поділу функцій шифрування й цифрового підпису, підтримки SSO, інтеграції в більшість комерційних додатків, підтримки відкритих стандартів і присутності персонального менеджера ключів і сертифікатів, що сполучить у собі функції захисту від НСД.

З виявлених недоліків слід зазначити відсутність коробкового рішення по відкликанню й призупиненню дії сертифікатів, наявність клієнта тільки під Windows і відсутність підтримки Українських стандартів.

iPlanet CMS

Продукт iPlanet CMS рекомендується для використання в середніх, дрібних компаніях, ISP і WEB проектах завдяки своїй масштабованості, підтримці гетерогенних середовищ, можливості поділу функцій шифрування й цифрового підпису, а також використанню унікальної технології Cloned CA. Також варто відзначити наявність власного OCSP сервера, підтримку відкритих стандартів і наявність персонального менеджера ключів і сертифікатів.

Проте серед виявлених недоліків можна відзначити слабку підтримку SSO, відсутність підтримки PKIX-CMP, відсутність підтримки українських стандартів, а також відсутність коробкових продуктів для підтримки ERP і баз даних. Ці недоліки можуть ускладнити позиціонування цього продукту для корпоративних клієнтів, які мають високі вимоги до цих аспектів.

Microsoft CS

Продукт Microsoft CS рекомендується для використання в корпоративних, середніх та дрібних компаніях, а також у веб-проектах, які розгорнуті на сервері IIS. Однак, перед використанням необхідно враховувати, що інформаційна політика організації повинна бути орієнтована виключно на корпорацію Microsoft і обмежувати використання систем і додатків, що не мають логотипу "Microsoft compatible" (як правило, це продукти Microsoft). Також необхідно мати розгорнуту Active Directory і операційні системи Windows NT/2000/XP/7.

Серед недоліків варто відзначити обмежену підтримку ієрархічної моделі довіри, використання власних стандартів та відсутність можливості змінювати політики (за прес-релізами Microsoft, версія Windows.NET буде мати цю можливість).

RSA Keon

Продукт RSA Keon рекомендується для використання в корпоративних, середніх і малих компаніях через його прийнятну масштабованість, підтримку гетерогенних середовищ, можливість поділу функцій шифрування й цифрового підпису, підтримку гібридної моделі довіри, підтримку SSO, інтеграцію з багатьма комерційними додатками, підтримку відкритих і Українських стандартів, а також наявність персонального менеджера ключів і сертифікатів, що забезпечує функції захисту від НСД.

Серед недоліків можна відзначити відсутність можливості відкликання й призупинення дії сертифікатів у клієнтському додатку, а також наявність клієнта тільки для платформи Windows.

UniCERT PKI

Продукт UniCERT PKI рекомендується для використання в середніх і малих компаніях, ISP і WEB проектах завдяки його масштабованості, модульності, підтримці гетерогенних середовищ і використанню гібридної моделі довіри, а також підтримці відкритих стандартів.

Серед виявлених недоліків можна відзначити обмежену підтримку SSO, відсутність підтримки Українських стандартів та відсутність коробкових продуктів для підтримки ERP і баз даних. Це може обмежувати можливості цього продукту для корпоративних клієнтів, які мають високі вимоги до цих аспектів.

2.2 Обгрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного забезпечення було обрано мову програмування Visual C# та реляційну базу даних Microsoft SQL Server. Мова C# включає в себе найкращі характеристики численних попередників. Розглянемо переваги та особливості цієї мови програмування.

Незважаючи на суттєві відмінності між компонентною об’єктною моделлю COM (основним стандартом Microsoft для компонентного проектування та реалізації програмного забезпечення) і моделлю Java Beans, мова програмування C# має багато спільного з мовою Java. Крім того, вона успадкувала численні особливості від свого попередника - мови Visual Basic, створеної компанією Microsoft.

Мова програмування C# базується на строгій компонентній архітектурі і реалізує передові механізми забезпечення безпеки коду.

Найбільш характерні риси, які спільні для мов програмування C# і Java, можна виокремити наступним чином. По-перше, обидві мови відносяться до об’єктно-орієнтованих мов програмування і мають однакові концепції успадкування, реалізації інтерфейсів, обробки виняткових ситуацій і збору сміття. Простори імен реалізовані у цих мовах подібним чином. Обидві мови відрізняються сильною типізацією і динамічним завантаженням коду під час виконання програми.

Мова програмування C# успадкувала певні механізми від свого безпосереднього попередника - мови програмування C++. Це включає "перевантаження" операторів, небезпечні арифметичні операції з плаваючою крапкою та інші особливості синтаксису.

Основні можливості мови програмування C#:

- компонентно-орієнтоване програмування (властивості, події);
- уніфікована система типізації (UTS);
- атрибути;
- обробка подій та виняткових ситуацій;
- властивості як засіб інкапсуляції даних;
- оператор циклу для колекцій foreach;
- механізми boxing і unboxing;
- індексатори (Indexers);
- перевантажені оператори;

– прямокутні масиви;
– делегати (Delegates).

Незважаючи на те, що мова програмування C# успадкувала ряд конструктивних синтаксичних механізмів і особливостей від мов C++, Visual Basic і Java, вона має додаткові можливості, що не обмежуються цими попередніми мовами.

У мові програмування C# реалізовано два основних підходи до створення типів: посилальні типи і типи-значення. Посилальні типи використовують посилання (покажчики) на значення, тоді як типи-значення містять власне значення. Змінні типів-значень зберігаються в стеці, тоді як посилальні типи використовують "кучу" для динамічного виділення і звільнення пам’яті для зберігання змінних.

Мова C# має різні засоби для опису об’єктів, таких як простори імен, класи, інтерфейси, структури, перерахування та делегати.

Також, мова програмування C# включає контекстні угоди, які забороняють дворазовий опис змінних у межах одного області опису та встановлюють довільну послідовність описів, включаючи описи локальних змінних до їх першого використання.

Мова програмування C# має наступні переваги:

– об’єктно-орієнтоване програмування;
– компонентно-орієнтоване програмування;
– безпечний код;
– уніфікована система типізації;
– підтримка подієво-орієнтованого програмування;
– власна мова для створення застосувань в середі .NET;
– об’єднання кращих ідей сучасних мов;
– підтримка програмування: Java, C++, Visual Basic і ін.

Після аналізу основних особливостей мови програмування C# та вивчення структури та принципів побудови програм на цій мові, можна виділити низку помітних переваг даної мови програмування.

Найбільш важливою характеристикою мови C# є її справжня об'єктно-орієнтованість. Крім того, мова C# реалізує компонентно-орієнтований підхід, що сприяє зменшенню залежностей від архітектури машини, забезпечує більшу гнучкість, переносимість та повторне використання програмних фрагментів.

Важливою особливістю, що відрізняє мову C# від попередників, є початковий фокус на безпеку коду, що особливо помітно порівняно з мовами C і C++.

Уніфікована система типізації мови C#, яка максимально наближена до Common Type System, що використовується в Microsoft .NET, є важливою перевагою мови.

Мова C# також відрізняється розширеною підтримкою подійно-орієнтованого програмування та має власне середовище для розробки додатків в рамках Microsoft .NET, що забезпечує найглибшу і ефективну інтеграцію з ним.

Об'єднання найкращих ідей сучасних мов програмування робить мову C# мовою нового покоління, а не просто сумою їх переваг.

Незважаючи на значну кількість принципових переваг порівняно з існуючими аналогами, мова програмування C# також має деякі недоліки, які можуть бути суб'єктивними, локальними або тимчасовими. Слід зазначити, що мова програмування C# має досить складний синтаксис. Вона вимагає від розробника досить глибокого розуміння мови і її правил, що може бути викликом для початківців. Також можна відзначити, що обсяг концептуальних ідей у мові C# відносно невеликий порівняно з іншими мовами програмування. Деякі високорівневі концепції можуть бути менш розвинуті або менш прості у використанні порівняно з іншими мовами. Проте ці недоліки не є суттєвими.

Мова програмування C# набула великої популярності і отримує все більшу підтримку серед розробників інструментальних засобів. Це свідчить про визнання

її переваг і придатності для багатьох проектів. Тому саме мову програмування C# було обрано для реалізації даної роботи.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки асиметричного шифрування електронних документів.

В процесі розробки бакалаврської роботи необхідно виконати наступні за:

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, розробити функціональну та структурну схеми системи;

в) розробити алгоритми для реалізації програмного забезпечення стосовно теми роботи, побудувати блок-схеми алгоритмів програми та підпрограм;

г) розробити програмне забезпечення системи, що дозволить реалізувати задачу, поставлену технічним завданням задачу;

д) організувати інтерфейс користувача та обробку виключних ситуацій;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Архітектура додатка на базі сокетів

Розроблений додаток ґрунтується на архітектурі клієнт-сервер з використанням сокетного з'єднання.

Концепція клієнт-сервер є одним з видів розподілених систем, де сервер обробляє запити клієнтів. Клієнт та сервер взаємодіють між собою за допомогою відповідного протоколу, у даному випадку – сокетів. Клієнт – це програма, що використовує ресурси, а сервер - програма, яка надає ці ресурси клієнтам.

Сокети – це програмний інтерфейс для забезпечення обміну інформацією між процесами. Цей обмін може відбуватися як на одному комп'ютері, так і між комп'ютерами, пов'язаними мережею. Сокет є абстрактним об'єктом, який представляє кінцеву точку з'єднання.

Сокет є одним з кінців двостороннього каналу зв'язку між двома програмами, що працюють в мережі. Реалізація сокетів забезпечує інкапсуляцію протоколів мережевого і транспортного рівнів.

Існують два види сокетів: потокові та дейтаграмні.

Потокові сокети – це сокет з встановленим з'єднанням, який складається з потоку байтів, який може бути двонаправленим, тобто через цю кінцеву точку додаток може і передавати і отримувати дані. Потоковий сокет гарантує виправлення помилок, обробляє доставку і зберігає послідовність даних. Потокові сокети досягають такого рівня якості за рахунок використання протоколу Transmission Control Protocol (TCP). TCP забезпечує поступлення даних на іншу сторону в потрібній послідовності і без помилок.

Для цього типу сокетів шлях формується до початку передачі електронних повідомлень. Цим гарантується, що дві сторони, які беруть участь у взаємодії,

приймають і відповідають. Якщо дані повинні гарантовано доставлятися іншій стороні або вони мають великий розмір і, якщо надійність зв'язку між двома додатками має першочергове значення, то використання саме потокових сокетів є більш прийнятним, ніж дейтаграмних.

Слід розрізняти клієнтські та серверні сокети. Клієнтські сокети грубо можна порівняти з кінцевими апаратами телефонної мережі, а серверні - з комутаторами. Клієнтський додаток (наприклад, браузер) використовує лише клієнтські сокети, а серверний (наприклад, веб-сервер, якому браузер посилає запити) – як клієнтські, так і серверні сокети.

Кожен процес може створити слухаючий сокет (серверний сокет) і прив'язати його до будь-якого порту комп'ютера. Той хто слухає процес зазвичай знаходиться в циклі очікування, тобто прокидається при появі нового з'єднання. При цьому зберігається можливість просто перевірити наявність з'єднань на даний момент, встановити тайм-аут для операції і так далі.

При створенні сокету, необхідно визначити три параметри: стиль взаємодії, простір імен, і протокол. Стиль взаємодії контролює, як сокет обробляє дані, що передаються, і визначає кількість партнерів взаємодії. Через сокети дані передаються блоками (пакетами). Стиль взаємодії визначає, як ці пакети будуть оброблені і як вони передаються від відправника до одержувача.

Стилі з'єднання гарантують доставку всіх пакетів у тому порядку, в якому вони були відправлені. Якщо під час передачі пакети були втрачені або доставлені в неправильному порядку, одержувач автоматично відправляє запит на їх повторну передачу.

Цикл життя сервера складається зі створення сокету, прив'язки сокету до адреси, виклику listen, що дозволяє з'єднання з сокетом, виклику accept, що приймає вхідні з'єднання, і потім закриття сокета. Дані не читаються і не записуються безпосередньо через сокет сервера, замість цього, кожен раз коли програма приймає нове з'єднання, ОС створює окремий сокет, використовується

при передачі даних по цьому з’єднанню. Архітектура з’єднання показана на рисунку 3.1.

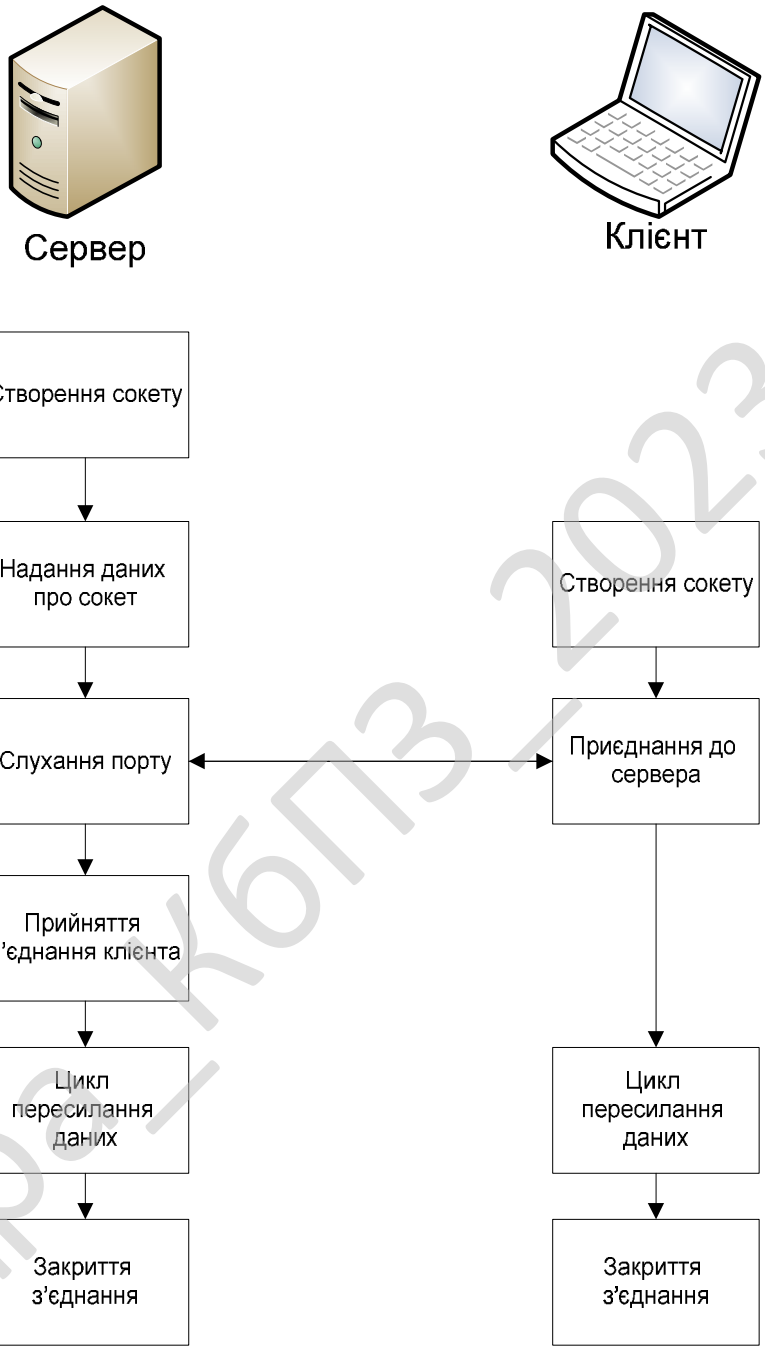

Рисунок 3.1 - Зображення архітектури «клієнт – сервер» на основі сокетів

Опис бази даних

Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає

спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків.

Характерна риса файлового підходу – вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв’язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних.

База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об’єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, як у випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв’язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

Призначення та класифікація систем управління базами даних

СУБД – це складна програмна система накопичення та з наступним маніпулюванням даними, що представляють інтерес для користувача. Кожній прикладній програмі СУБД надає інтерфейс з базою даних та має засоби безпосереднього доступу до неї. Таким чином, СУБД відіграє центральну роль в функціонуванні автоматизованого банку даних.

Архітектурно СУБД складається з двох великих компонент (рис.2.1). За допомогою мови опису даних (МОД) створюються описи елементів, груп та записів даних, а також взаємозв'язки між ними, які, як правило, задаються у вигляді таблиць. В залежності від конкретної реалізації СУБД мову опису даних підрозділяють на мову опису схеми бази даних (МОС) та мову опису підсхем бази даних (МОП). Слід особливо зазначити, що МОД дозволяє створити не саму базу даних, а лише її опис.

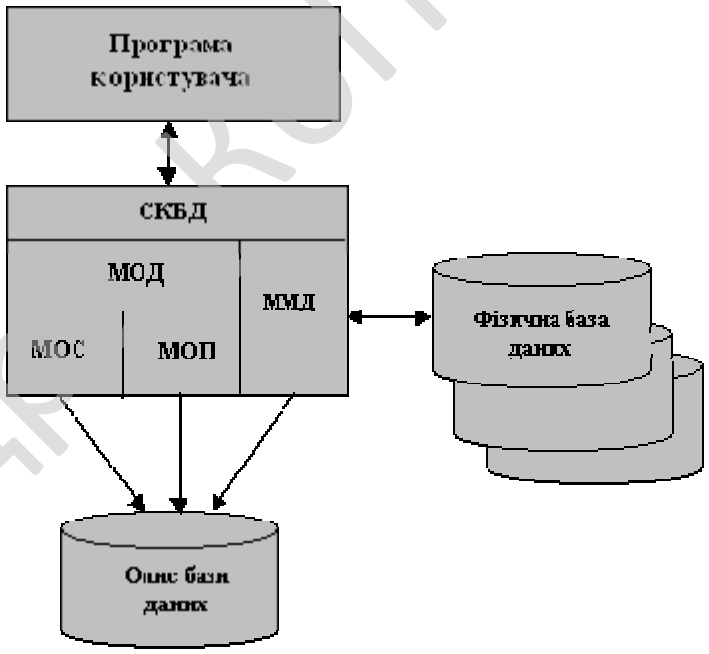

Рисунок 3.2 - Архітектура СУБД

Для виконання операцій з базою даних в прикладних програмах використовується мова маніпулювання даними (ММД). Фактична структура фізичного зберігання даних відома тільки СУБД.

З метою забезпечення зв'язків між програмами користувачів і СУБД (що особливо важливо при мультипрограмному режимі роботи операційної системи) в СУБД виділяють особливу складову - резидентний модуль системи керування базами даних. Цей модуль значно менший від всієї СУБД , тому на час функціонування автоматизованого банку інформації він може постійно знаходитись в основній пам'яті ЕОМ та забезпечувати взаємодію всіх складових СУБД і програм, які до неї звертаються.

Приведена структура притаманна усім СУБД, котрі розрізняються обмеженнями та можливостями по виконанню відповідних функцій. Отже, процес порівняння і оцінки таких систем для одного конкретного застосування зводиться до співставлення можливостей наявних СУБД з вимогами користувачів.

До недавнього часу при організації обробки інформації на ЕОМ застосовувався підхід, при якому на основі інформації одного і того ж об'єкту управління (наприклад, матеріальних ресурсів) в залежності від її вигляду (нормативна. розцінкова тощо) і ступеню постійності формувались масиви лінійної структури двох типів: умовно-постійні (з інформацією, яка використовувалась багато разів протягом довгого часу) і умовно-перемінні (з фактичною або поточною інформацією). Створення і багаторазове використання масивів з умовно-постійною інформацією має ті переваги, які дозволяють значно спростити первинну документацію шляхом виведення з її складу ряд постійних реквізитів. знизити трудомісткість робіт на стадії заповнення первинних документів, підготовки і вроду фактичної або поточної інформації до ЕОМ. Недоліком таких масивів, які мають лінійну структуру, є то що інформація одного і того я об'єкту управління розосереджується поміж багатьох різних масивів (нормативних, планових та ін.), що неминуче веде до дублювання деяких реквізитів, ускладненню при спільній їх обробці тощо, а головне - не дає змоги

реалізувати принцип незалежності від прикладних програм користувача. Лінійні масиви, сформовані традиційним способом, ефективні, як правило, а позиції одного застосування.

3 розвитком інформаційного забезпечення систем автоматизованої обробки інформації, прагненням забезпечити виконання нових режимів обробки даних у реальному часі і з мультидоступом до схованих даних позначилась нова тенденція до складення інформаційного забезпечення розподілених баз даних. В умовах використання таких баз створюються комплексні масиви нелінійної структури, які мають усі дані про ту чи іншу предметну область або про керований об'єкт як постійного, так і перемінного характеру.

Взагалі база даних є сукупність даних на машинних носіях, які використовуються при функціонуванні системи обробки інформації, організовані по визначеним правилам, які передбачають загальні принципи описування збереження і маніпулювання ними, а також які незалежні від прикладних програм. В основі організації бази даних є модель даних, яка визначає правила, у відповідності з якими структуруються дані. За допомогою моделі представляється велика кількість даних і описуються взаємно зв'язки між ними. Найбільш поширені такі моделі даних: ієрархічна, мережева, реляційна.

В ієрархічній моделі зв'язок даних "один до одного" (1:1) означає, що кожному значенню (екземпляру) елемента даних А відповідає одне і тільки одне значення, пов'язаного з ним елемента В. Наприклад, поміж такими елементами пар даних, як код готової продукції і її найменуванням є вищезазначений зв'язок, так як тільки кожному коду продукції відповідає одне її найменування.

Зазначимо, що ієрархічна модель даних будується на основі принципу підпорядкованості поміж елементами даних і представляє собою деревоподібну структуру, яка складається із вузлів (сегментів) і дуг (гілок). Дерево у ієрархічній структурі упорядковане за існуючими правилами розташування його сегментів і гілок: на верхньому рівні знаходиться один, кореневий (вихідний) сегмент, сегмент другого рівня, породжений, залежить від першого, вихідного; доступ до

кожного породженого (крім кореневого) здійснюється через його вихідний сегмент; кожний сегмент може мати по декілька екземплярів конкретних значень елементів даних, а кожний елемент породженого сегменту пов’язаний з екземпляром вихідного і створює один логічний запис; екземпляр породженого сегменту не може існувати самостійно, тобто без кореневого сегменту; при вилученні екземпляру кореневого сегмента також вилучаються усі підпорядковані і взаємопов’язані з ним екземпляри породжених сегментів.

В мережевій моделі зв’язок "один до багатьох" (1:В) означає, що значенню елемента А відповідають багато (більше одного) значень, пов’язанню з ним елементів В. Наприклад, поміж елементами даних "код виробу" (елемент А) і "кодом матеріалів" (елементи В) існує такий взаємозв’язок бо на виготовлення одного виробу використовується багато різних матеріалів.

Мережева модель даних представляє собою орієнтований граф з пойменованими вершинами і дугами. Вершини графа - записи, які представляють собою по іменовану сукупність логічних взаємозв’язаних елементів даних або агрегатів даних. Під агрегатом даних розуміють пошановану сукупність елементів даних, які є усередині запису. Для кожного типу записів може бути кілька екземплярів конкретних значень його інформаційних елементів Два записи, взаємозв’язані дугою, створюють набір даних. Запис, з якого виходить дуга, називається власником набору, а запис, до якого вона направлена, - членом набору.

В реляційній моделі зв’язок "багатьох до багатьох" (В:В) указує на те, що декільком значенням елементів даних А відповідає декілька значені елементів даних В. Наприклад, поміж елементами даних "код операції технологічного процесу" і "табельний номер працівника" існує зазначені взаємозв’язок, так як багато операцій технологічного процесу можуть виконувати різні працівники (табельні номери) і навпаки.

Реляційна модель даних являє собою набір двомірних плоских таблиць, що складаються з рядків і стовпців. Первинний документ або лінійний масив

являє собою плоску двомірну таблицю. Така таблиця називається відношенням, кожний стовбець-атрибутом, сукупність значень одного типу (стовпця) –доменом, а рядка – кортежем. Таким чином, стовпці таблиці являються традиційними елементами даних, а рядки – записами. Таблиці (відношення) мають імена. Імена також присвоюються і стовпцям таблиці. Кожний кортеж (запис) відношення має ключ. Ключі є прості і складні. Простий ключ-це ключ, який складається з одного атомарного атрибуту, значення якого унікальне (яке не повторюються).Складний ключ складається з двох і більше атрибутів. Для зв'язків відношень друг з другом в базі даних є зовнішні ключі. Атрибут або комбінація атрибута відношення є зовнішнім ключем, якщо він не є основним (первинним) ключем цього відношення, але являється первинним ключем для другого відношення.

Різновидністю баз даних, з точки зору їх зберігання і використання, є розподіленні бази даних. Ці бази даних широко використовуються при організації комплексів взаємопов'язаних АРМ фахівців, на яких застосовуються ПЕОМ .

Розподілена база даних - це сукупність логічно зв'язаних баз даних або частин однієї бази, які розпаралелені поміж декількома територіально – розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

Проектування бази даних

Основні принципи створення БД: цілісність, вірогідність, контроль, захист від несанкціонованого доступу , єдність і гнучкість, стандартизація та уніфікація, адаптивність, мінімізація введення і виведення інформації (однократність введення інформації, принцип введення - виведення тільки змін).

Цілісність здатність даних задовольняти принцип повного узгодження, точність, доступність і достовірне відображення реального стану об'єкта.

Вимоги до інформаційного забезпечення (ГОСТ 24.104-85 "Автоматизовані системи управління. Загальні вимоги") такі:

1. Інформаційне забезпечення має бути достатнім для виконання всіх функцій ІС, які автоматизуються.

2. Для кодування інформації, яка використовується тільки в цій ІС, має бути застосовані класифікатори , які є у замовника ІС.

3. Для кодування в ІС вихідної інформації, яка використовується на вищому рівні, мають бути використані класифікатори цього рівня, крім спеціально обумовлених випадків.

4. Інформаційне забезпечення ІС має бути суміщене з інформаційним забезпеченням систем, які взаємодіють з нею, за змістом, системою кодування, методами адресації, форматами даних і формами подання інформації, яка отримується і видається інформаційною системою.

5. Форми документів, які створюються інформаційною системою, мають відповідати вимогам стандартів УСД чи нормативно - технічним документам замовника ІС.

6. Форми документів і відеокадрів, які вводяться чи коригуються через термінали ІС, мають бути погодженні з відповідними технічними характеристиками терміналів.

7. Сукупність інформаційних масивів ІС має бути організована у вигляді бази даних на машинних носіях.

8. Форми подання вихідної інформації ІС мають бути узгоджені із замовником (користувачем) системи.

9. Терміни і скорочення, які застосовуються у вихідних повідомленнях, мають бути загальноприйнятими в цій предметній області й погоджені із замовником системи.

10. У ІС мають бути передбачені необхідні заходи щодо контролю і оновлення даних в інформаційних масивах ІС, оновлення масивів після відмови будь-яких технічних засобів ІС, а також контролю ідентичності однойменної інформації в базах даних.

Можуть створюватись також самостійні інформаційні засоби і вироби для конкретного користувача.

Ефективне функціонування інформаційної системи об’єкта можливе лише при відповідній організації інформаційної бази - сукупності впорядкованої інформації, яка використовується при функціонуванні ІС і поділяється на зовнішньо – і внутрішньомашинну (машинну) бази (ГОСТ 34.003-90).

Зовнішньомашинна інформаційна база – частина інформаційної бази, яка являє собою сукупність повідомлень, сигналів і документів, призначених для безпосереднього сприйняття людиною без застосування засобів обчислювальної техніки.

Внутрішньомашинна інформаційна база – частина інформаційної бази, що використовується в ІС на носіях даних.

Така зовнішньомашинна ІБ має багато модифікацій від подання у вигляді повідомлень на паперовому носії, запитів на екрані дисплея, мовного спілкування з ЕОМ та ін.

Внутрішньомашинна ІБ пройшла три етапи еволюції.

Перший етап характеризується роз’єднаним фондом даних:

1) програми розв’язання кожної окремої задачі становили одне ціле з масивами, які оброблялися;

2) використання будь-якого масиву для іншої задачі забезпечувалось індивідуально пристосуванням до форм подання даних, структур елементів масивів і. т. ін.;

3) опис даних не потрібний, оскільки структура раніше була відома;

4) коригування масивів виконувалось індивідуальними засобами;

5) задача розв’язувалася в пакетному режимі, користувач отримував результати винятково у вигляді машинограм і виробничих документів через групу підготовки і оформлення даних.

Дані розглядаємо на трьох рівнях, і є пряма залежність логічного рівня програм, фізичного та логічного рівня збереження.

Другий етап централізований фонд даних.

1) Дані відокремлені від процедур їх обробки і організовані в бібліотеці масивів загального користування. Подання інформації, формати елементів даних і структура масивів уніфіковані і не залежать від конфігурації пам'яті та організації.

2) Опис даних відокремлено як від програм, так і від самих даних, тому дані і програми їх обробки стають значною мірою незалежними. Це полегшує зміну структур даних і програм. Але реорганізація бібліотеки і її окремих груп компонентів потребує зміни програми обробки.

Залишаються залежні логічні рівні програми і збереження.

Третій етап – організація баз даних характеризується:

1) об'єднанням не лише інформації, а й апаратно програмних засобів її поповнення, коригування і видачі користувачеві;

2) повним відокремленням функцій нагромадження, ведення і реорганізації даних від функцій їх обробки. Дані коригуються поза рівнем програм користувача за допомогою власного апарату бази даних;

3) появою логічного буферу, системи управління базою даних, розв'язки між програмами користувача і базою даних;

4) можливістю оперативної реалізації довільних запитів у режимі безпосереднього зв'язку з ЕОМ;

5) високим ступенем централізації загальносистемних масивів, яка передбачає спільне використання загальних даних;

6) різноманітністю даних і поєднанням в довільні логічні структури;

7) наявністю потужного програмного забезпечення і мовних засобів.

8) Усі рівні незалежні.

Нині існують другий і третій етапи. Основною задачею є визначення потрібної кількості баз даних і оптимального розподілу інформації між ними з урахуванням того, що економічний об'єкт - це динамічна система, яка перебуває в постійному розвитку. Використовуючи пріоритет виробничих функцій, необхідно

побудувати таку базу даних. Так , навколо поняття «Модель виробу» формуються дві оболонки: внутрішня являє конструкторську документацію, зовнішня - технологічну і управлінську інформацію (рис. 3.3).

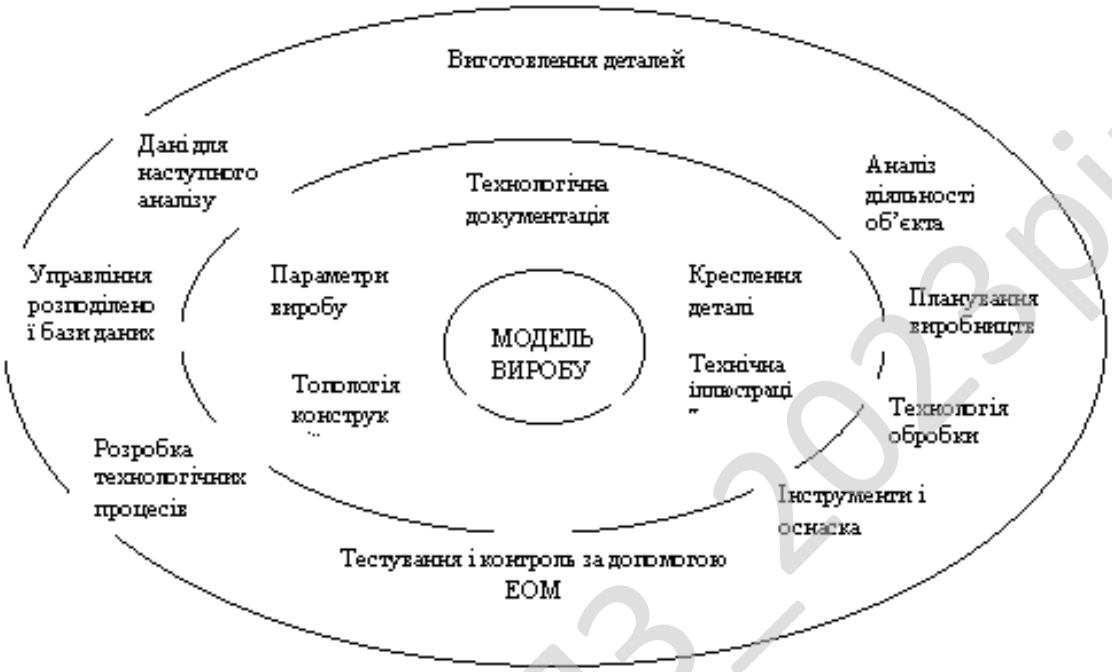

Рисунок. 3.3 - Структура бази даних "Модель виробу"

Однак виникла така проблема: визначити, чи потрібна одна база даних, чи кілька локальних, або взаємозв'язана розподілена база даних, локальні файли чи їх комбінації і т.п. При цьому враховується інформація, що використовується для реалізації багатьох функцій, особливо в оперативному режимі, активна інформація, тобто така, що використовується багаторазово.

Описуючи організацію інформаційної бази (РД 50-34, 698-90), потрібно дати опис логічної і структурної бази даних.

Документ складається з двох частин:

1)опис внутрішньомашинної інформаційної бази;

2)опис зовнішньомашинної інформаційної бази.

Кожна частина складається з таких розділів:

1)логічна структура;

2)фізична структура (для зовнішньомашинної інформаційної бази);

3)організація ведення інформаційної бази.

У розділі "Логічна структура" наводять опис складу даних, їх формати і взаємозв’язки між даними.

У розділі "Фізична структура" наводять опис вибраного варіанта розміщення даних на конкретних машинних носіях даних.

При описі структури внутрішньомашинної інформаційної бази наводять перелік баз даних і масивів та логічні зв’язки між ними. Для масиву інформації вказують логічну структуру масиву чи дають посилання на документ "Опис масиву інформації".

Описуючи структуру зовнішньомашинної інформаційної бази, наводять перелік документів та інших інформаційних повідомлень, використання яких передбачено в системі, із зазначенням автоматизованих функцій, при реалізації яких формується чи використовується цей документ.

Якщо цю інформацію наведено у документах "Перелік вхідних сигналів і даних" і "Перелік вихідних сигналів", можна посилатися на ці документи.

У розділі "Організація ведення інформаційної бази", описуючи внутрішньомашинну базу, наводять послідовність процедур при створенні і обслуговуванні бази із зазначенням в разі потреби регламенту виконання процедур і засобів захисту бази від руйнування і несанкціонованого доступу, а також зв’язків між масивами баз даних і масивами вхідної інформації.

Описуючи зовнішньомашинну інформаційну базу потрібно навести послідовність процедур по маршруту руху груп документів до передачі їх на обробку, а також описати маршрут руху вихідних документів.

Види інформаційних масивів.

При організації раціонального варіанту внутрішньомашинної інформаційної бази даних, яка найбільш повно відбиває специфіку об’єкта управління, перед розробниками постають вимоги до організації масивів, які можуть бути суперечливими. До них належать:

1) повнота подання даних;

2) мінімальний склад даних;

3) мінімізація часу вибірки даних;

4) незалежність структури масивів від програмних засобів їх організації;

5) динамічність структури інформаційної бази.

Останнім часом склалися такі основні підходи до побудови внутрішньомашинної інформаційної бази:

1) проектування масиву як відображання змісту окремого документа;

2) проектування масивів для окремих процесів управління;

3) проектування масивів для комплексів процесів управління, які реалізуються;

4) проектування бази даних;

5) проектування кількох баз даних.

Кожний з цих підходів має свої переваги і недоліки, а вибір залежить від обчислювальної техніки, яка використовується, програмних засобів і специфіки процесів, що автоматизуються.

Основні масиви можуть мати вигляд локальних масивів чи організовані в базу даних (БД) під керуванням системою управління базою даних (СУБД).

Взаємозв'язок користувача з базою даних зображено на рис.3.4.

База даних є сукупність даних, що використовується при функціонуванні ІС, організована за певними правилами, які передбачають загальні принципи опису, зберігання і маніпулювання даними і незалежна від прикладних програм (ГОСТ 24.003-84).

Система управління базами даних - це сукупність програм і мовних засобів, які призначені для управління даними в базі даних і забезпечують взаємодію її з прикладними програмами (ГОСТ 20886- 85).

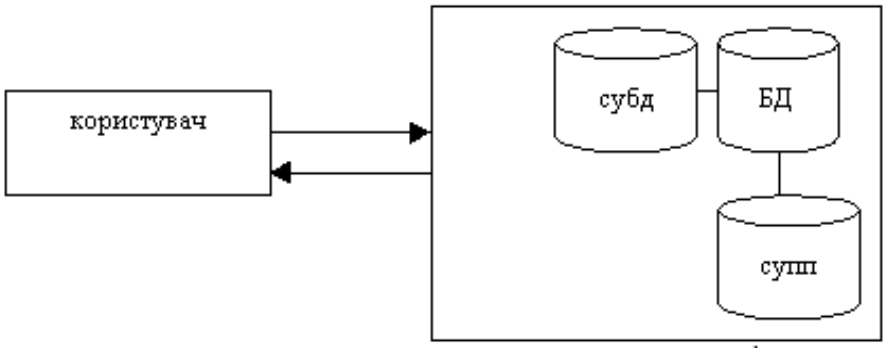

Рисунок 3.4 – Взаємозв’язок користувача з базою даних

Масив даних - це конструкція даних, компоненти якої ідентичні за своїми характеристиками і є значенням функції від фіксованої кількості цілочисельних аргументів (ГОСТ 20886- 85).

Файл - це ідентифікована сукупність примірників повністю описаного в конкретній програмі типу даних, розміщених ззовні програми в зовнішній пам’яті та доступних програмі, за допомогою спеціальних операцій (ГОСТ 20886- 85).

Методика проектування інформаційного забезпечення складається з трьох етапів.

На першому етапі "Розробка рішень по інформаційній базі": визначається склад і обсяг нормативно - довідкової інформації; розробляються пропозиції щодо вдосконалення діючого документообігу; структура бази даних; система збирання і передавання інформації, а також рішення з організації і ведення бази даних; визначається склад і характеристики вихідної і вхідної інформації (сигналів, документів, даних).

На другому етапі "Вибір номенклатури і прив’язка системи класифікації і кодування інформації>: визначається перелік типів інформаційних об’єктів, які підлягають ідентифікації в ІС, перелік необхідних класифікаторів; вибираються й розробляються класифікатори інформаційних об’єктів і системи кодування; визначається система внесення змін і доповнень у класифікатори; розробляються принципи й алгоритми автоматизованого ведення класифікаторів.

На третьому етапі "Розробка рішень щодо забезпечення обміну інформацією в системі" розробляється схема інформаційного забезпечення.

Реалізація криптографічних алгоритмів в системі

Для захисту даних та забезпечення приватності під час обміну електронними документами по мережі Інтернет було вирішено використати алгоритми шифрування RSA та DES.

Розглянемо спочатку систему криптографічного захисту на основі алгоритму RSA. В таких системах для шифрування даних використовується один ключ, а для дешифрування – інший, тому їх також називають асиметричними. Перший ключ є відкритим і може бути опублікований для використання усіма користувачами системи, які шифрують дані. При цьому розшифрувати дані за допомогою відкритого ключа неможливо. Для дешифрування, сторона-отримувач зашифрованої інформації використовує другий ключ, який тримає в таємниці.

Безпека алгоритму RSA заснована на принципі складності факторизації цілих чисел, тобто розкладанні їх на прості співмножники. Алгоритм використовує два ключі – відкритий і закритий (секретний), разом відкритий і відповідний йому секретний ключі утворюють узгоджену пару ключів, вони є взаємно оберненими з точки зору задачі шифрування. Якщо електронний документ було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Процес генерації відкритого і закритого ключів для RSA представляє собою ряд математичних операцій, виконання яких, в кінцевому результаті, дозволить отримати модуль ключа, відкриту та закриту експоненти, з яких, в свою чергу, будуть сформовані відкритий і закритий ключі.

Алгоритм генерації пари ключів такий:

1) обрати два великих простих числа p і q фіксованого розміру;

2) обчислити їх добуток (модуль) n;

3) обчислити функцію Ейлера:

$$\varphi(n) = (p-1)*(q-1); \qquad (4.1)$$

4) підібрати ціле взаємнопросте з φ(n) число *e* таке, що $1 < e < \varphi(n)$;

5) за розширеним алгоритмом Евкліда обчислити число *d*, таке, що

$$e * d \equiv 1 * (\mathrm{mod}\,\varphi(n)). \tag{4.2}$$

Після виконання цих дій будуть отримані наступні параметри для використання алгоритму шифрування: *n* – модуль ключа, *e* – відкрита експонента; *d* – секретна експонента, пара чисел (*n,e*) – відкритий ключ, пара чисел (*n,d*) – закритий ключ. При цьому числа *p* і *q* після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті. При генерації ключів, розмір модуля ключів повинен бути якомога більшим, починаючи від 512 біт і більше. Це гарантуватиме високу стійкість до злому шифру.

Шифрування на основі алгоритму RSA. При шифруванні будь-якої інформації за допомогою алгоритму RSA її потрібно представити в числовому вигляді. Тому перед проведенням шифрування текстової інформації її спочатку перетворюють на набір цілих чисел. Зазвичай це реалізується шляхом перетворення тексту в потік байтів, або ж в набір цілих чисел, що представляють собою ASCII-коди або юнікоди відповідних символів. Ще однією вимогою до інформації, яка підлягає шифруванню є те, що вона повинна бути меншою за модуль ключа. Інакше інформація розбивається на блоки, які є меншими ніж модуль ключа, і операції шифрування та дешифрування проводяться над кожним блоком окремо.

Шифрування інформації відбувається наступним чином:

$$c = m^e \bmod n, \tag{4.3}$$

де *c* – зашифрована інформація; *m* – початкова інформація; *e* – відкритий ключ; *n* – модуль ключа.

Після проведення перерахованих дій буде отримано зашифрована інформація, з якого можна буде отримати початкова інформація, використовуючи секретний ключ.

Дешифрування інформації відбувається так:

$$m = c^d \bmod n, \tag{4.4}$$

де m – відкрита інформація; c – зашифрована інформація; d – закритий ключ; n – модуль ключа.

Основними перевагами асиметричних алгоритмів шифрування, в тому числі і алгоритму RSA є такі:

– відсутність потреби передачі секретного ключа каналами зв'язку;

– порівняно довгий час життя ключів;

– забезпечення високої криптостійкості;

– досить проста програмна реалізація алгоритму.

До недоліків алгоритмів шифрування з відкритим ключем належать:

– використання дуже довгих ключів для забезпечення високого рівня надійності;

– порівняно низька швидкодія алгоритму.

Через низьку швидкодію алгоритму RSA зазвичай використовується гібридна система шифрування, яка полягає в тому, що інформацію шифрують за допомогою продуктивніших симетричних алгоритмів з випадковим (сеансовим) ключем, а за допомогою RSA шифрують лише цей ключ. Після цього зашифрований ключ передається всім учасникам. Такий варіант захисту інформації є достатньо швидкодіючим і ефективним, але при його використанні необхідно вирішити проблему конфіденційного розподілу ключа сесії та гарантування його своєчасності. Для шифрування та дешифрування даних невеликого об'єму можливо використовувати стандартний алгоритм RSA.

Цифровий підпис на основі алгоритму RSA. Для того щоб поставити цифровий підпис електронному документу – треба обчислити хеш-значення документу, а потім зашифрувати його закритим ключем відправника. За необхідності перед цими діями документ слід зашифрувати. У цій роботі перед поставленням цифрового підпису він шифрувався алгоритмом DES. Для перевірки цифрового підпису одержувач повинен обчислити хеш-значення документу, а також розшифрувати його зашифровану відправником хеш-значення і порівняти їх. Якщо обчислена та розшифрована хеш-значення

співпали, то перевірка цифрового підпису успішна – документ справжній і авторство підтверджене. Інакше перевірка цифрового підпису не успішна, і документ підроблено.

Шифрування алгоритмом DES. DES (Data Encryption Standard) – це симетричний алгоритм шифрування, основний принцип роботи якого полягає у застосуванні ітеративного процесу, що включає перестановки та змішування бітів шифрування, які виконуються над блоком вхідних даних фіксованої довжини (64 біти). Основні етапи алгоритму DES наступні:

1. *Перестановка початкового блоку*. Вхідний блок розбивається на дві частини, ліву і праву, які потім переставляються місцями.

2. *Кодування ключа*. Використовується особливий алгоритм для перетворення початкового ключа на 16 раундових підключів.

3. *Раундові перестановки і змішування*. Виконується 16 раундів, в кожному з яких застосовуються операції перестановки та змішування бітів, включаючи використання підключів.

4. *Фінальна перестановка*. Після останнього раунду виконується фінальна перестановка бітів, що формує вихідний зашифрований блок.

Принцип DES полягає у тому, що кожен раунд виконується над попереднім результатом, використовуючи підключі, що постійно змінюються. Цей процес створює складну ітеративну структуру шифрування, яка забезпечує високий рівень безпеки даних при використанні відповідного ключа.

Процес дешифрування в алгоритмі DES відбувається за зворотнім порядком до процесу шифрування.

3.2 Розробка структурної схеми

На рисунку 3.5 показана структурна схема розроблюваної системи.

Відправник хоче послати електронний документ (з гарантованими конфіденційністю, цілісністю, дійсністю даних) користувачеві Одержувач.

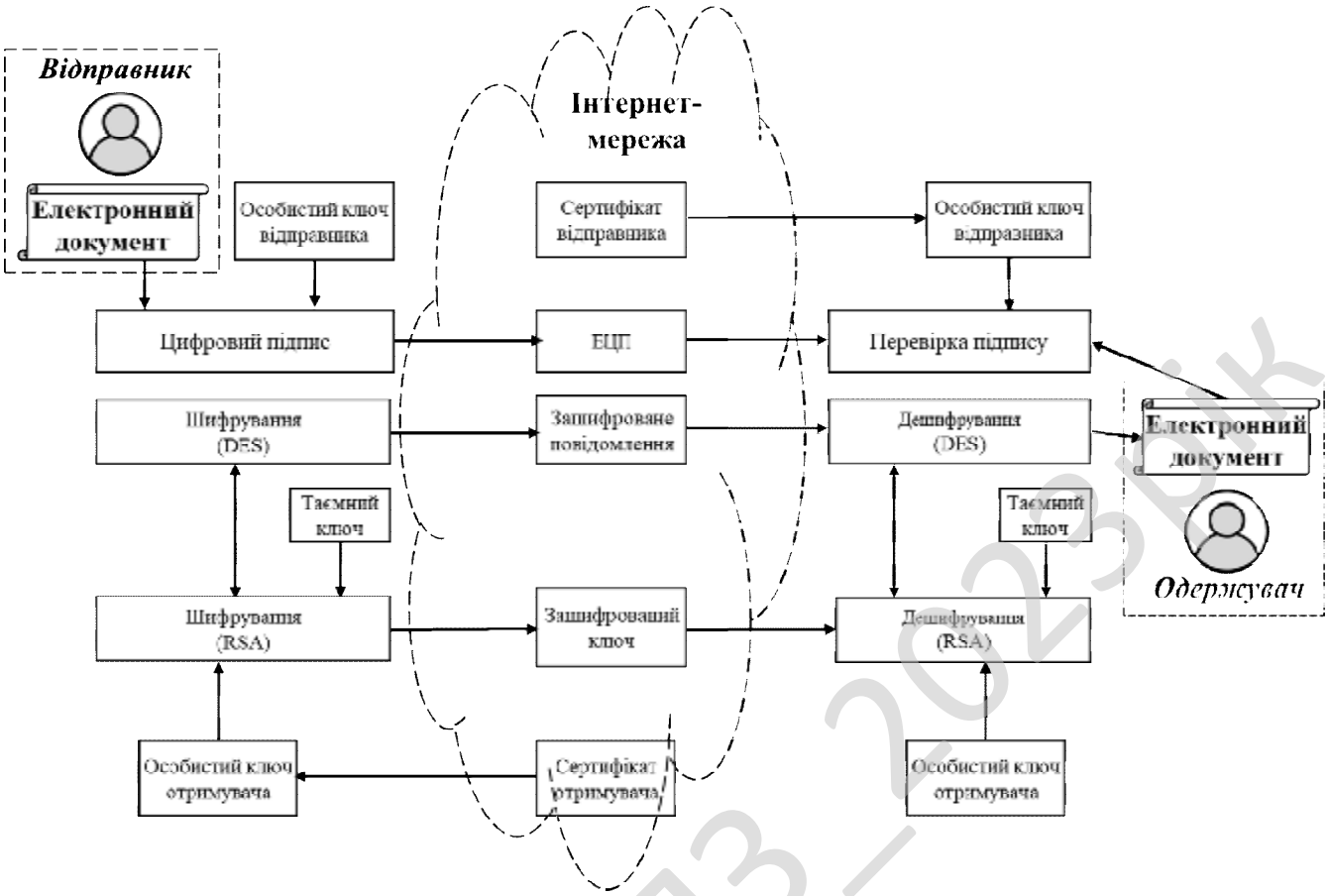

Рисунок 3.5 – Структурна схема системи

Обмін електронними документами складається з наступних кроків:

1. Відправник створює цифрову сигнатуру для електронного документу з використанням свого приватного ключа.

2. Відправник використовує єдиний ключ симетричного шифрування для шифрування повідомлення.

3. Щоб сформувати безпечний канал, у якому буде захищена конфіденційність ключа шифрування при його передачі через загальнодоступний канал зв'язку, Відправник використовує відкритий ключ Одержувача (із сертифіката Одержувача), щоб зашифрувати ключ шифрування. Результат цього етапу – захищений ящик (lockbox), у якому втримується зашифрована копія ключа шифрування.

4. Одержувач використовує свій приватний ключ, щоб розшифрувати захищений ящик (lockbox) та отримати зашифровану копію секретного ключа шифрування електронного документу.

5. Одержувач розшифровує повідомлення за допомогою єдиного ключа шифрування. Тепер Одержувач може прочитати повідомлення.

6. Одержувач використовує відкритий ключ Відправника для перевірки дійсності й цілісності повідомлення, засвідчуючи цифрову сигнатуру, що є в сертифікаті Відправника.

3.3 Розробка функціональної схеми

На рисунку 3.6 зображена функціональна схема розробленого програмного забезпечення. Система розділена на клієнтське і серверне програмне забезпечення.

Розглянемо спочатку клієнтську частину додатку. Вона складається з наступних блоків:

- Модуль головного вікна клієнтської програми;
- Модуль профайлу користувача;
- Модуль кабінету користувача;
- Модуль структури підприємства;
- Модуль журналу подій користувача;
- Модуль кабінету користувача;
- Модуль обробки електронних документів;
- Модуль налаштування клієнтської програми;
- Модуль довідки.

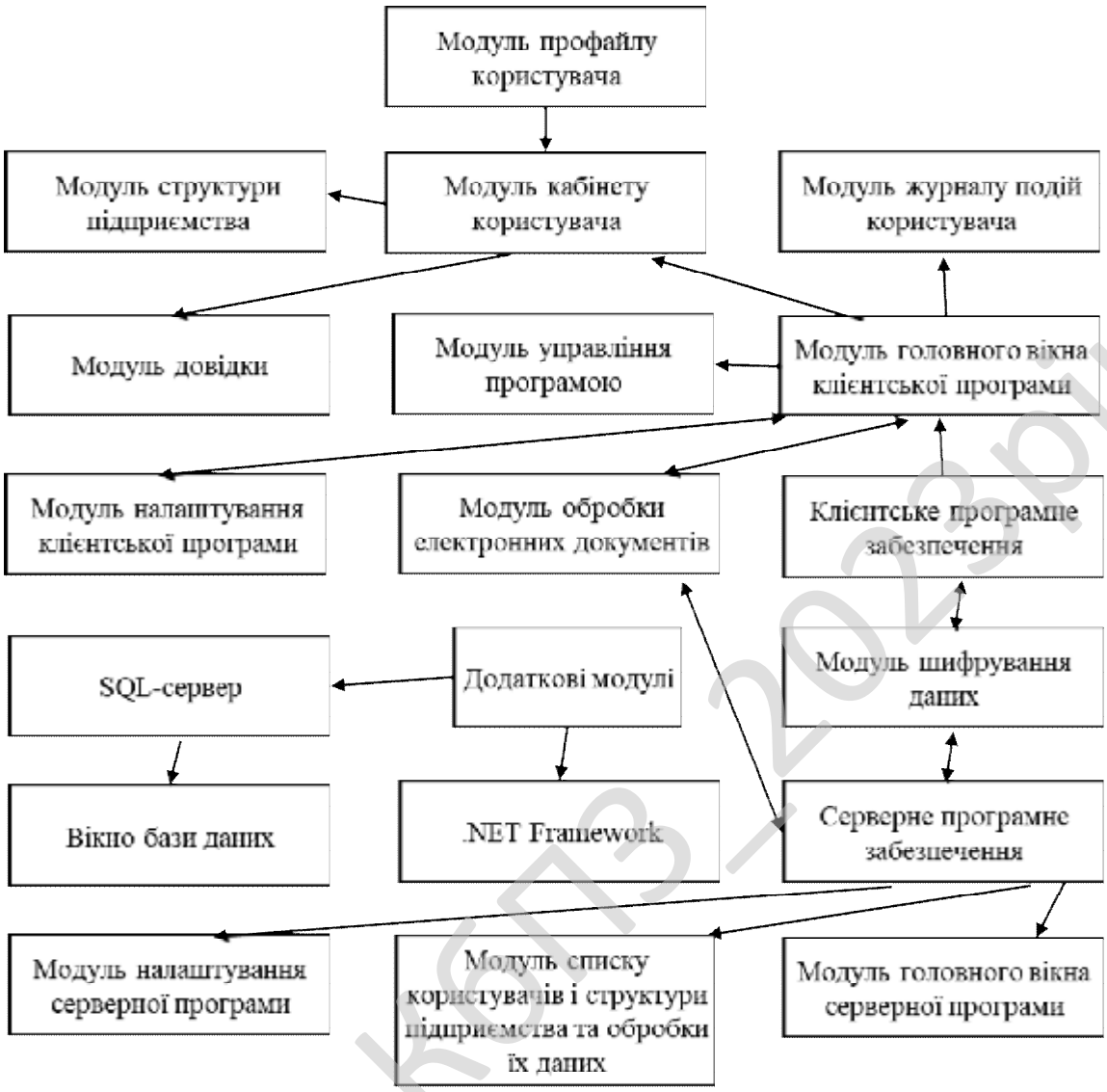

Рисунок 3.6 – Функціональна схема системи

Серверне програмне забезпечення складається з таких блоків:

– Модуль головного вікна серверної програми;

– Модуль списку користувачів і структури підприємства та обробки їх даних;

– Модуль обробки електронних документів;

– Модуль налаштування серверної програми.

Також рекомендовані для встановлення додаткові модулі: .NET Framework та SQL-сервер.

Окремо розроблено модуль шифрування даних, який використовується як серверним, так і клієнтським програмним забезпеченням.

3.4 Розробка діаграми процесів

На рисунку 3.7 зображена діаграма взаємодії процесів системи. Як видно з діаграми, першим процесом, який завантажується у системі, є процес виведення головного вікна. Він взаємодіє з наступними процесами:

– Створення електронного документу.

– Отримання електронного документу.

Процес створення електронного документу взаємодіє з наступними процесами:

– Створення ключа для шифрування.

– Процес отримання електронного документу.

– Генерація сертифікату та ключа для підпису.

– Збереження закритого ключа.

– Публікація сертифікату.

Процес отримання електронного документу взаємодіє з наступними процесами:

– Написання електронного документу.

– Відправка електронного документу.

– Шифрування електронного документу.

– Накладання на електронний документ цифрового підпису.

– Відкриття електронного документу.

– Перегляд електронного документу.

– Дешифрування електронного документу.

– Перевірка цифрового підпису.

Рисунок 3.7 – Діаграма процесів системи

Тож, у даному розділі було розглянуто опис проектних рішень при розробці програмного забезпечення для реалізації мети роботи, його структурну та функціональну схеми, а також діаграму взаємодії процесів. У наступних розділах буде розглянуто блок-схеми алгоритмів роботи розробленого додатку та реалізацію основних функцій розробленої системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема основної програми. З видно, що програма працює наступним чином. Спершу відбувається завантаження основного вікна програми. Після цього можна створювати та приймати електронні документи.

Якщо користувач намагається створити захищений електронний документ, то виводиться вікно створення документа. Далі відбувається перевірка наявності ключа шифрування. Якщо його немає, то він генерується. Якщо ж ключ є то перевіряється наявність ключа та сертифіката для цифрового підпису, яким буде підписуватися електронний документ, уповноваженою особою, що його створює. Якщо ключа немає, то відбувається генерація ключа для цифрового підпису та генерація сертифікату. Після того як, користувач введе назву та зміст документу, він буде зашифрований алгоритмом DES з застосуванням секретного ключа та підписаний цифровим електронним підписом користувача алгоритмами SHA-1 і RSA.

Потім документ буде відправлений одержувачу, після обрання користувачем відповідної дії.

За цим слідує перевірка наявності вхідних електронних документів. Якщо є вхідні електронні документи, то читаються їх назви. Відбувається завантаження вхідних електронних документів. Під час завантаження відбувається перевірка цифрових підписів з використанням відповідних алгоритмів. Після чого виводяться результати перевірки підписів.

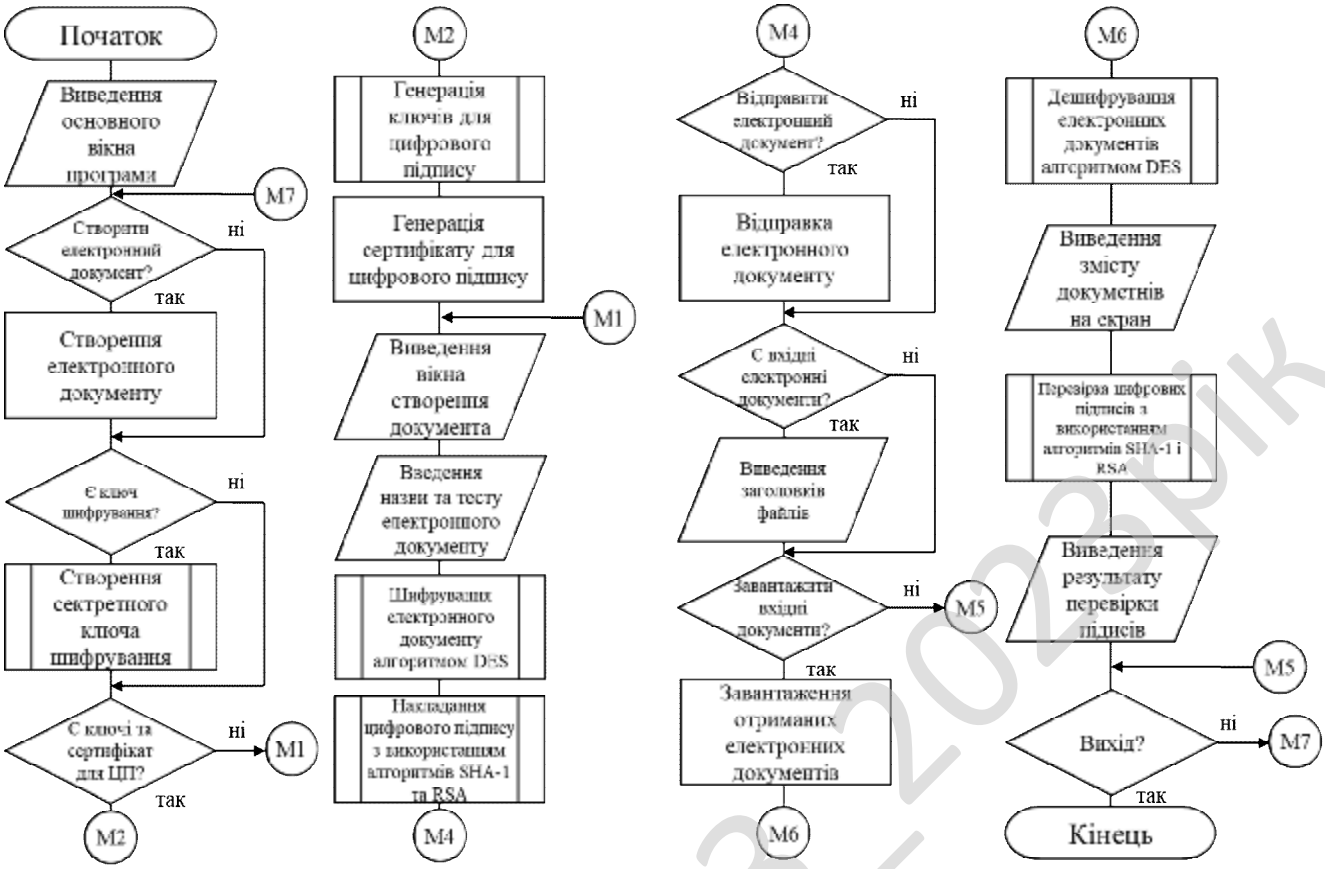

Рисунок 4.1 – Блок-схема основної програми

Накладання ЕЦП здійснюється відправником (підписувачем) документу із використанням його закритого ключа.

Перевірка ЕЦП виконується одержувачем захищеного електронного документу з використанням відкритого ключа підписувача.

Генерація ключів здійснюється генератором псевдовипадкових чисел, після чого програма надсилає запит на сертифікацію відкритого ключа.

Заявка на сертифікацію відкритого ключа складається у формі електронного документа, шифрується відкритим ключем центру сертифікатів та надсилається у зашифрованому вигляді по мережі.

В окремих випадках, при угоді сторін, заявка складається і у вигляді документа на паперовому носієві, підписаного власноручним підписом особи та передається з рук у руки.

Заявка на сертифікацію відкритого ключа повинна містити:

– прізвище та ім’я фізичної особи, номер документа, що засвідчує особу;

– інші ідентифікаційні дані особи, а також інформацію, необхідну для передачі їй повідомлень;

– відкритий ключ, що треба сертифікувати;

– інші відомості, встановлені уповноваженим органом.

Володіння сертифікатом дозволяє користувачеві підписувати свої повідомлення цифровим підписом. Після успішної сертифікації закритий та відкритий ключі зберігаються у файлі на флешці чи іншому електронному носії, а відкритий ключ також заноситься у базу даних цифрових сертифікатів.

Випуск сертифіката відкритого ключа здійснюється центром сертифікації. Створення центрів сертифікації здійснюється з метою впорядкування процесу випуску та відкликання («анулювання») сертифікатів відкритого ключа, у підтримці актуальності бази даних сертифікатів і встановлення відповідальності за правильність внесення в сертифікат необхідних відомостей. Ці обов’язки покладені на адміністраторів центрів сертифікації. Конкретні підстави для одержання сертифіката визначаються окремими документами компанії.

На виданий сертифікат встановлюється термін дії. Існує непряма залежність корисного терміну дії від інформації, яку містить сертифікат. Чим більше інформації містить цей документ, тим менше корисний термін його дії. Це пов’язано з тим, що інформація може змінюватися, й власнику доведеться видати новий сертифікат до того, як вибігає термін дії старого.

Новий сертифікат відкритого ключа може бути виданий у випадку закінчення терміну дії сертифіката, втрати або компрометації діючого сертифіката.

Нижче наведено блок-схеми вже конкретно шифрування/дешифрування та формування/читання цифрового підпису.

На рисунку 4.2 наведено блок-схему створення та перевірки цифрового підпису за допомогою алгоритму RSA.

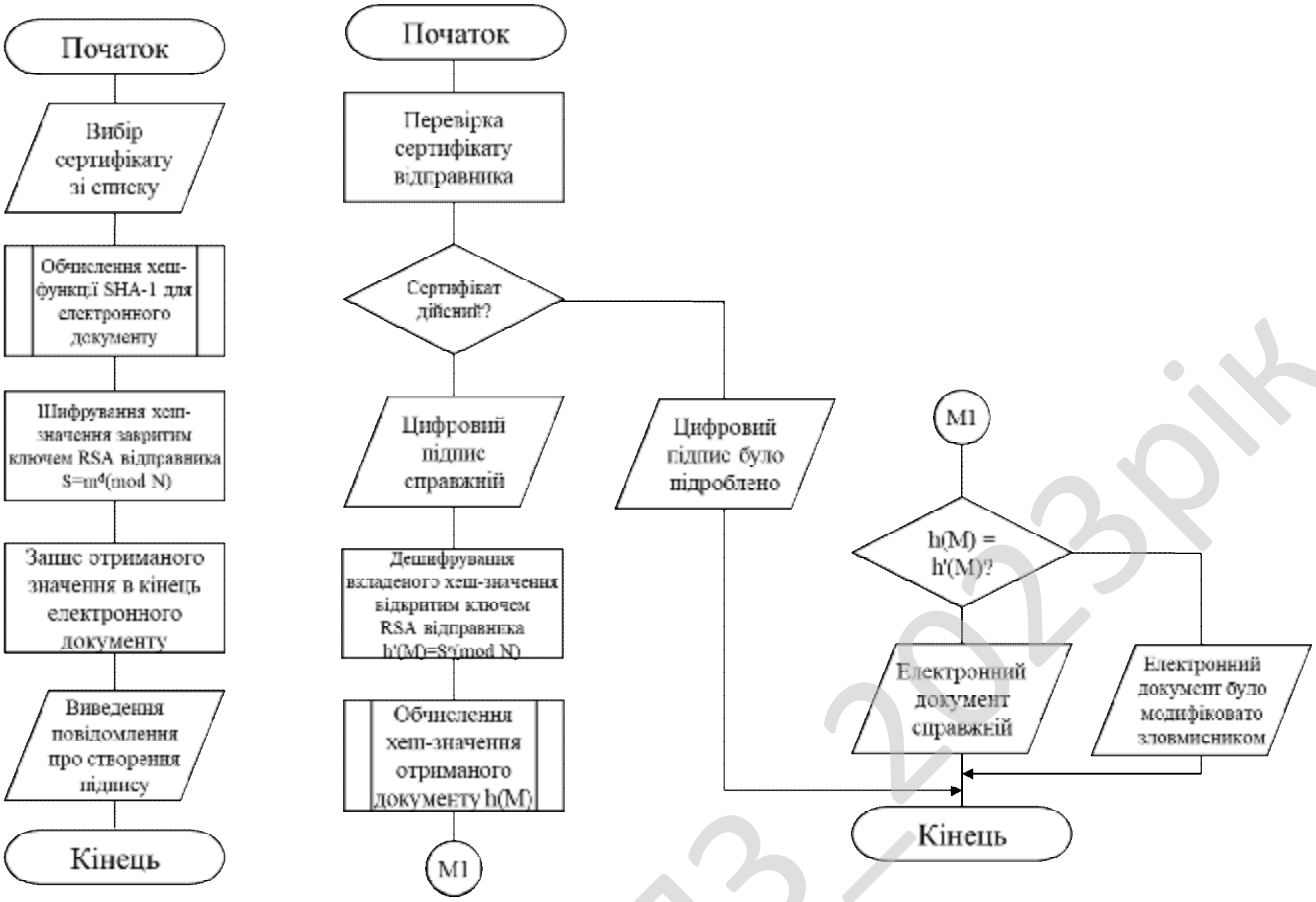

Рисунок 4.2 – Блок-схеми підпрограм роботи цифрового підпису за допомогою алгоритму RSA

З неї видно, що відбуваються наступні дії. Спершу відбувається вибір сертифікату зі списку згенерованих сертифікатів.

Після цього відбувається обчислення хеш-функції по відповідному алгоритму SHA-1. Хеш-значення шифрується відкритим ключем відправника. Отримане значення записується у кінець листа, про що виводиться відповідне повідомлення.

Перевірка відбувається наступним чином. Спершу перевіряється сертифікат відправника, якщо сертифікат дійсний, то вважається, що цифровий підпис справжній, у іншому випадку вважається що цифровий підпис було підроблено, робота програми закінчується. Якщо ж цифровий підпис правильний, то відбувається дешифрування хеш-значення відкритим ключем відправника.

Після цього відбувається обчислення хеш-значення отриманого листа за

формулами наведеними нижче, та перевіряється справжність хеш-функції. Для цього порівнюється хеш функція яку сформував отримувач по листу, та хеш-функція. Яка була отримана у результаті дешифрування. Якщо вони співпадають, то вважається, що лист справжній. У іншому випадку, вважається. що лист несправжній.

Шифрування та дешифрування даних у розробленій системі реалізовано наступним чином:

```
namespace RSA_Encryption_Decryption
{
    public partial class Form1 : Form
    {
        UnicodeEncoding ByteConverter = new UnicodeEncoding();
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
        byte[] plaintext;
        byte[] encryptedtext;

        public Form1()
        {
            InitializeComponent();
        }

        private void EncButton_Click(object sender, EventArgs e)
        {
            plaintext = ByteConverter.GetBytes(PlainTextInput.Text);
            encryptedtext = Encryption(plaintext, RSA.ExportParameters(false),
false);
            CipherText.Text = ByteConverter.GetString(encryptedtext);
        }

        private void DecButton_Click(object sender, EventArgs e)
        {
            byte[] decryptedtext = Decryption(encryptedtext,
RSA.ExportParameters(true), false);
            PlainTextOutput.Text = ByteConverter.GetString(decryptedtext);
        }

        //Шифрування
        static public byte[] Encryption(byte[] Data, RSAParameters RSAKey, bool
DoOAEPPadding)
        {
            try
            {
                byte[] encryptedData;
                using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
                {
                    RSA.ImportParameters(RSAKey); encryptedData = RSA.Encrypt(Data,
DoOAEPPadding);
                } return encryptedData;
            }
            catch (CryptographicException e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }
```

```
        //Дешифрування
        static public byte[] Decryption(byte[] Data, RSAParameters RSAKey, bool
DoOAEPPadding)
        {
            try
            {
                byte[] decryptedData;
                using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
                {
                    RSA.ImportParameters(RSAKey);
                    decryptedData = RSA.Decrypt(Data, DoOAEPPadding);
                }
                return decryptedData;
            }
            catch (CryptographicException e)
            {
                Console.WriteLine(e.ToString());
                return null;
            }
        }
```

Також наведемо функцію для роботи з профайлом користувача – співробітника підприємства, де реалізовано захист електронних документів.

```
        private void Overview_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Overview_page;
            Overview.FlatStyle = FlatStyle.Standard;
            Edu.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
        }

        //=====================================================================
        //Освіта
        //=====================================================================
        private void Education_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Education_page;
            Edu.FlatStyle = FlatStyle.Standard;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
        }
        //=====================================================================
        //Паспортні дані
        //=====================================================================
        private void Passportinfo_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Passportinfo_page;
```

```
        Passportinfo.FlatStyle = FlatStyle.Standard;
        Overview.FlatStyle = FlatStyle.Flat;
        Edu.FlatStyle = FlatStyle.Flat;
        AfterDiploma.FlatStyle = FlatStyle.Flat;
        LastWorkbtn.FlatStyle = FlatStyle.Flat;
        FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
        Positionbtn.FlatStyle = FlatStyle.Flat;
        Vacationsbtn.FlatStyle = FlatStyle.Flat;
    }
    //=========================================================================
    //Післядипломна підготовка
    //=========================================================================
    private void AfterDiploma_Click(object sender, EventArgs e)
    {
        Info_pages.SelectedTab = AfterDiploma_page;
        AfterDiploma.FlatStyle = FlatStyle.Standard;
        Edu.FlatStyle = FlatStyle.Flat;
        Overview.FlatStyle = FlatStyle.Flat;
        Passportinfo.FlatStyle = FlatStyle.Flat;
        LastWorkbtn.FlatStyle = FlatStyle.Flat;
        FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
        Positionbtn.FlatStyle = FlatStyle.Flat;
        Vacationsbtn.FlatStyle = FlatStyle.Flat;
    }
    //=========================================================================
    //Попередня робота
    //=========================================================================
    private void LastWorkbtn_Click(object sender, EventArgs e)
    {
        Info_pages.SelectedTab = LastWork_page;
        LastWorkbtn.FlatStyle = FlatStyle.Standard;
        Edu.FlatStyle = FlatStyle.Flat;
        Overview.FlatStyle = FlatStyle.Flat;
        Passportinfo.FlatStyle = FlatStyle.Flat;
        AfterDiploma.FlatStyle = FlatStyle.Flat;
        FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
        Positionbtn.FlatStyle = FlatStyle.Flat;
        Vacationsbtn.FlatStyle = FlatStyle.Flat;
    }
    //=========================================================================
    //Родинний стан
    //=========================================================================
    private void FamilyStatusbtn_Click(object sender, EventArgs e)
    {
        Info_pages.SelectedTab = FamilyStatus_page;
        FamilyStatusbtn.FlatStyle = FlatStyle.Standard;
        LastWorkbtn.FlatStyle = FlatStyle.Flat;
        Edu.FlatStyle = FlatStyle.Flat;
        Overview.FlatStyle = FlatStyle.Flat;
        Passportinfo.FlatStyle = FlatStyle.Flat;
        AfterDiploma.FlatStyle = FlatStyle.Flat;
        Positionbtn.FlatStyle = FlatStyle.Flat;
        Vacationsbtn.FlatStyle = FlatStyle.Flat;
    }
    //=========================================================================
    //Посада
    //=========================================================================
    private void Positionbtn_Click(object sender, EventArgs e)
    {
        Info_pages.SelectedTab = Position_page;
        Positionbtn.FlatStyle = FlatStyle.Standard;
        FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
        LastWorkbtn.FlatStyle = FlatStyle.Flat;
```

```
            Edu.FlatStyle = FlatStyle.Flat;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
        }
        //=========================================================================
        //Відпустки
        //=========================================================================
        private void Vacationsbtn_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Vacation_page;
            Vacationsbtn.FlatStyle = FlatStyle.Standard;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            Edu.FlatStyle = FlatStyle.Flat;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
        }
        //=========================================================================
        //Вибрати фото
        //=========================================================================
        private void Add_photo_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog open = new OpenFileDialog();
                open.Filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp)|*.jpg;
*.jpeg; *.gif; *.bmp";
                if (open.ShowDialog() == DialogResult.OK)
                {
                    Photo_url = new Bitmap(open.FileName);
                    Photo.Image = Photo_url;
                    PathToPhoto = open.FileName;
                }
            }
            catch (Exception)
            {
                throw new ApplicationException("Не вдалося завантижити фото");
            }
            image = File.ReadAllBytes(PathToPhoto);
        }

        //=========================================================================
        //Підсказки
        //=========================================================================
        private void Speciality_Click(object sender, EventArgs e)
        {
            ToolTip tt = new ToolTip();
            tt.SetToolTip(Speciality, "Спеціальність (професія) за
дипломом(свідоцтвом) відповідно із займаною посадою");
        }

        private void Qualification_Click(object sender, EventArgs e)
        {
            ToolTip tt = new ToolTip();
            tt.SetToolTip(Qualification, "Кваліфікація за дипломом (свідоцтвом)");
        }

        //=========================================================================
        //Завантаження даних
```

```
        //=====================================================================
        public void LoadInformation()
        {
            try
            {

//=====================================================================
                //Загальні відомості

//=====================================================================
                string OverviewQuery = "SELECT * FROM [Worker] WHERE [IdWorker] = '"
+ Program.Global.ChosenWorker + "'";
                SqlDataAdapter OverviewAdapter = new SqlDataAdapter(OverviewQuery,
Program.Global.conn);
                OverviewAdapter.Fill(OverviewSet);

//=====================================================================
                idWorker.Text = OverviewSet.Tables[0].Rows[0]["IdWorker"].ToString();
                SecondName.Text =
OverviewSet.Tables[0].Rows[0]["SecondName"].ToString();
                FirstName.Text =
OverviewSet.Tables[0].Rows[0]["FirstName"].ToString();
                ThirdName.Text =
OverviewSet.Tables[0].Rows[0]["ThirdName"].ToString();
                Birthday.Text = OverviewSet.Tables[0].Rows[0]["Birthday"].ToString();
                Gender.SelectedIndex =
Gender.Items.IndexOf(OverviewSet.Tables[0].Rows[0]["Gender"].ToString());
                Nationality.Text =
OverviewSet.Tables[0].Rows[0]["Nationality"].ToString();
                IPN.Text = OverviewSet.Tables[0].Rows[0]["IPN"].ToString();
                EDRPOU.Text = OverviewSet.Tables[0].Rows[0]["EDRPOU"].ToString();
                MaritalStatus.Text =
OverviewSet.Tables[0].Rows[0]["FamilyStatus"].ToString();
                CompletionDate.Text =
OverviewSet.Tables[0].Rows[0]["CompletionDate"].ToString();
                data = (Byte[])(OverviewSet.Tables[0].Rows[0]["Photo"]);
                MemoryStream mem = new MemoryStream(data);
                Photo.Image = Image.FromStream(mem);


//=====================================================================
                //Паспортні дані

//=====================================================================
                string PassportInfoQuery = "SELECT * FROM [PassportInfo] WHERE
[IdWorker] = '" + Program.Global.ChosenWorker + "'";
                SqlDataAdapter PassportInfoAdapter = new
SqlDataAdapter(PassportInfoQuery, Program.Global.conn);
                PassportInfoAdapter.Fill(PassportInfoSet);

//=====================================================================
                SerieOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["SerieOfPassport"].ToString();
                NumberOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["NumberOfPassport"].ToString();
                DateOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["DateOfPassport"].ToString();
                GiverOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["GiverOfPassport"].ToString();
                RegAdress.Text =
PassportInfoSet.Tables[0].Rows[0]["RegAdress"].ToString();
                FactAdress.Text =
PassportInfoSet.Tables[0].Rows[0]["FactAdress"].ToString();
```

```
//=====================================================================
                //Попередня робота

//=====================================================================
                string LastWorkQuery = "SELECT * FROM [LastWork] WHERE [IdWorker] =
'" + Program.Global.ChosenWorker + "'";
                SqlDataAdapter LastWorkAdapter = new SqlDataAdapter(LastWorkQuery,
Program.Global.conn);
                LastWorkAdapter.Fill(LastWorkSet);

//=====================================================================
                LastPlaceOfWork.Text =
LastWorkSet.Tables[0].Rows[0]["LastPlaceOfWork"].ToString();
                PositionOnLastWork.Text =
LastWorkSet.Tables[0].Rows[0]["PositionOnLastWork"].ToString();
                GenExpDays.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpDays"].ToString();
                GenExpMonths.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpMonths"].ToString();
                GenExpYears.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpYears"].ToString();
                ExpDaysForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpDaysForPremium"].ToString();
                ExpMonthsForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpMonthsForPremium"].ToString();
                ExpYearsForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpYearsForPremium"].ToString();
                DateOfDismissal.Text =
LastWorkSet.Tables[0].Rows[0]["DateOfDismissal"].ToString();
                ReasonOfDismissal.Text =
LastWorkSet.Tables[0].Rows[0]["ReasonOfDismissal"].ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

Для створення налаштувань програмного забезпечення розроблено наступний модуль:

```
    public partial class Settings_Form : Form
    {
        public Settings_Form()
        {
            InitializeComponent();
        }

        private void Settings_Form_Load(object sender, EventArgs e)
        {
            IPadress.Text = Settings.Default["IP_adress"].ToString();
            Port.Text = Settings.Default["Port"].ToString();
            db_login.Text = Settings.Default["Login"].ToString();
            db_password.Text = Settings.Default["Password"].ToString();

        }
```

```
        private void Save_Settings_Click(object sender, EventArgs e)
        {
            Program.Global.conn.Close();
            Program.Global.conn.ConnectionString = "Data Source=" +
IPadress.Text.ToString() + "," + Port.Text.ToString() + ";Initial Catalog=OKDATABASE'
+ ";User ID=" + db_login.Text.ToString() + ";Password=" +
db_password.Text.ToString();
            try
            {
                Program.Global.conn.Open();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
                return;
            }
            ConnectionState ConState = Program.Global.conn.State;
            if (ConState == ConnectionState.Open)
            {
                Settings.Default["IP_adress"] = IPadress.Text;
                Settings.Default["Port"] = Port.Text;
                Settings.Default["Login"] = db_login.Text;
                Settings.Default["Password"] = db_password.Text;
                Settings.Default.Save();
                this.Close();
            }
        }

        private void Cancel_Settings_Click(object sender, EventArgs e)
        {
                this.Close();
        }

        private void Apply_Settings_Click(object sender, EventArgs e)
        {
            Program.Global.conn.Close();
            Program.Global.conn.ConnectionString = "Data Source=" +
IPadress.Text.ToString() + "," + Port.Text.ToString() + ";Initial Catalog=OKDATABASE'
+ ";User ID=" + db_login.Text.ToString() + ";Password=" +
db_password.Text.ToString();
            try
            {
                Program.Global.conn.Open();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
                return;
            }
            ConnectionState ConState = Program.Global.conn.State;
            if (ConState == ConnectionState.Open)
            {
                Settings.Default["IP_adress"] = IPadress.Text;
                Settings.Default["Port"] = Port.Text;
                Settings.Default["Login"] = db_login.Text;
                Settings.Default["Password"] = db_password.Text;
                Settings.Default.Save();
                Apply_Settings.Enabled = false;
            }

        }
```

```
        private void Settings_Form_FormClosing(object sender, FormClosingEventArgs e)
        {
            Main_Form mainfm = new Main_Form();
            mainfm.Refresh();
        }

        private void TextChanged(object sender, EventArgs e)
        {
            Apply_Settings.Enabled = true;
        }
    }
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення було вирішено поширювати по вільній ліцензії. Для захисту вільного програмного забезпечення обрано ліцензію GNU GPL.

Ліцензія GNU General Public License (GNU GPL або просто GPL) є однією з найпопулярніших відкритих ліцензій, яка використовується для захисту програмного забезпечення. Вона розроблена Free Software Foundation (FSF) і є ключовим компонентом проекту GNU, який має на меті створення вільної операційної системи.

Ліцензія GPL створена для того, щоб забезпечити чотири основні свободи користувачам: свободу використовувати програмне забезпечення за будь-яким призначенням, свободу вивчати, як програма працює, і змінювати її, свободу розповсюджувати копії програми, і свободу вносити зміни в програму і розповсюджувати ці зміни відкрито.

GPL використовує концепцію «copyleft», що зобов’язує будь-які модифікації оригінального програмного забезпечення, що розповсюджується під GPL, бути також доступними під тією ж ліцензією. Це означає, що будь-який вихідний код, що виникає з GPL-ліцензованого вихідного коду, повинен також бути вільним. Таким чином, GPL стимулює відкритий обмін знаннями і сприяє розвитку інформаційних технологій.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунках 5.1-5.3 наведено приклади вікон розробленого програмного забезпечення для асиметричного шифрування електронних документів.

На рис. 5.1 зображено вікно налаштувань серверу розробленої системи.

Рисунок 5.1 – Вікно налаштувань серверу розробленої системи

На рис. 5.2 зображено головне вікно клієнтської частини програми.

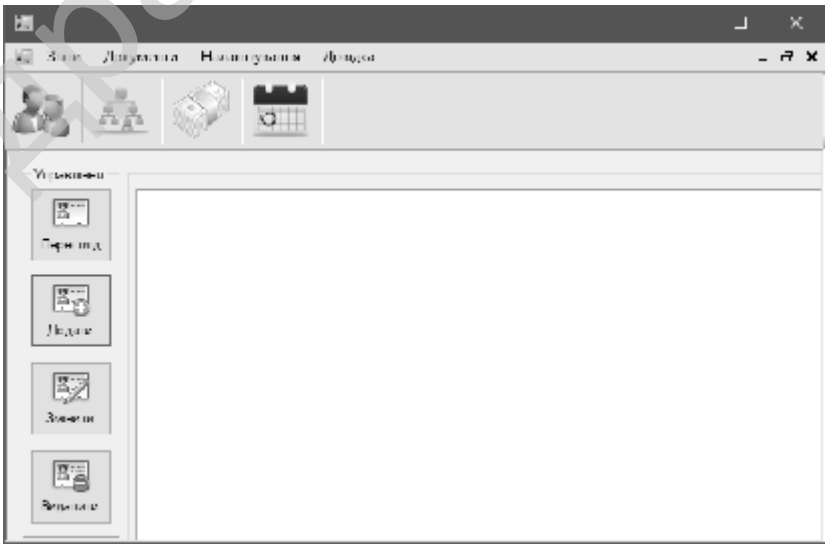

Рисунок 5.2 – Головне вікно програми

На рис. 5.3 зображено вікно одержання ключів та сертифікатів користувачем програмного забезпечення для створення захищених електронних документів методами асиметричного шифрування.

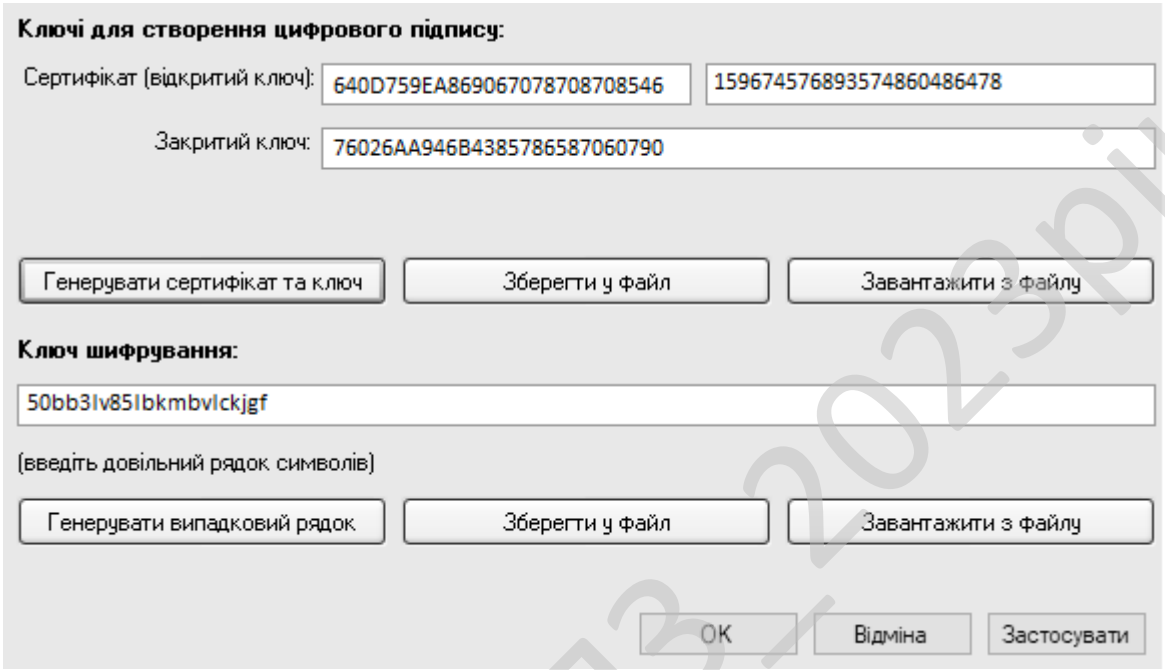

Рисунок 5.3 – Вікно одержання ключів та сертифікатів

6 ОСНОВНІ ВИСНОВКИ

У бакалаврській кваліфікаційній роботі створене програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів.

В рамках обраної теми були вирішені наступні задачі:

– Дослідження існуючих методів асиметричного шифрування та забезпечення інформаційної безпеки електронних документів.

– Розробка алгоритмів для системи кібербезпеки для асиметричного шифрування електронних документів.

– Програмна реалізація системи кібербезпеки для асиметричного шифрування електронних документів.

Розроблені під час виконання роботи алгоритми дозволяють успішно вирішувати завдання створення системи кібербезпеки для асиметричного шифрування електронних документів.

Програма була реалізована на мові програмування рівня C# з застосуванням реляційної бази даних Microsoft SQL Server. Для шифрування даних використано алгоритм DES, для накладання цифрового підпису алгоритми RSA та SHA-1.

Програма призначена для виконання під управлінням ОС Windows 10/11.

У пояснювальній записці надаються принципи функціонування розробленої системи, зокрема, описи структури та принципів роботи алгоритмів.

Розроблене програмне забезпечення поширюється та захищається вільною ліцензією GNU GPL.

Тестування створеного програмного забезпечення підтверджує правильність обраних проектних рішень та ефективність роботи додатку. Створене програмне забезпечення має потенційну можливість для подальшого масштабування та використання у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 p.

2. Al-Odat Z., Abbas A., & Khan S. U. (2019, December). Randomness analyses of the secure hash algorithms, SHA-1, SHA-2 and modified SHA. In 2019 International Conference on Frontiers of Information Technology (FIT) (pp. 316-3165). IEEE.

3. Alsuwaiyel, M. H. (2021). Algorithms: design techniques and analysis (Vol. 15). World Scientific.

4. Andress J. Foundations of Information Security: A Straightforward Introduction. No Starch Press, 2019.

5. Beller M., Gousios G., Panichella A., Proksch S., Amann S., & Zaidman A. (2017). Developer testing in the ide: Patterns, beliefs, and behavior. IEEE Transactions on Software Engineering, 45(3), 261-284.

6. Bhanot R., & Hans R. (2015). A review and comparative analysis of various encryption algorithms. International Journal of Security and Its Applications, 9(4), 289-306.

7. Campbell D. (2022). E-commerce and the Law of Digital Signatures.

8. Cody Lindley. Front-End Developer Handbook 2017. 2017. URL: https://frontendmasters.gitbooks.io/front-end-handbook-2017/content.

9. Comer D. E. (2018). The Internet book: everything you need to know about computer networking and how the Internet works. CRC Press.

10. 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.

11. David Upton. CodeIgniter for Rapid PHP Application Development. Packt Publishing, 2007. 244 p.

12. Deitel P., & Deitel H. (2016). Visual C# how to program. Pearson.

13. Easttom C. (2015). Modern cryptography. McGraw-Hill Education.

14. Forouzan B. A., & Mukhopadhyay D. (2015). Cryptography and network security (Vol. 12). New York, NY, USA:: Mc Graw Hill Education (India) Private Limited.

15. Griffiths I. (2022). Programming C# 10. " O'Reilly Media, Inc.".

16. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 c.

17. Jain H. Data Structures & Algorithms In Go. – Hemant Jain, 2022. – 584 c.

18. Jeremy Thomas. MarkSheet. A free HTML and CSS tutorial. 2015-2017. URL: https://marksheet.io.

19. Katz J., & Lindell Y. (2020). Introduction to modern cryptography. CRC press.

20. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

21. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

22. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

23. Mehlhorn, K., Sanders, P., & Sanders, P. (2008). Algorithms and data structures: The basic toolbox (Vol. 55, p. 56). Berlin: Springer.

24. Menezes A. J., Van Oorschot P. C., & Vanstone S. A. (2018). Handbook of applied cryptography. CRC press.

25. Michail H. E., Athanasiou G. S., Theodoridis G., Gregoriades A., & Goutis C. E. (2016). Design and implementation of totally-self checking SHA-1 and SHA-256 hash functions' architectures. Microprocessors and Microsystems, 45, 227-240.

26. Miles R. Begin to Code with C#. Microsoft Press, 2016.

27. Miles R., Exam Ref 70-483 Programming in C#, 2nd ed. Microsoft Press, 2018.

28. Moriarty K., Kaliski B., Jonsson J., & Rusch A. (2016). PKCS# 1: RSA cryptography specifications version 2.2 (No. rfc8017).

29. Nagel C. (2018). Professional C# 7 and. Net Core 2.0. John Wiley & Sons.

30. Newman M. (2018). Networks. Oxford university press.

31. Patil P., Narayankar P., Narayan D. G., & Meena S. M. (2016). A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and Blowfish. Procedia Computer Science, 78, 617-624.

32. Pelc J. (2015). An IDE for C# Script Development.

33. Rocca La M. Advanced Algorithms and Data Structures. – Manning, 2021. – 768 p.

34. Sharp J. (2018). Microsoft Visual C# Step by Step. Microsoft Press.

35. Sharp J. Microsoft Visual C# Step by Step, 10th ed. Microsoft Press, 2022

36. Skeet J. (2019). C# in Depth. Simon and Schuster.

37. Smith R.E. Elementary Information Security, 2nd ed. Jones & Bartlett Learning, 2015.

38. Stamp M. Information Security: Principles and Practice, 2nd ed. Wiley-IEEE Press, 2011.

39. Stinson D. R., & Paterson M. (2018). Cryptography: theory and practice. CRC press.

40. Ullman J.D., Aho A.V., Hopcroft J.E. The Design and Analysis of Computer Algorithms - International Economy Edition Paperback. – Pearson education, 1905. – 470 p.

41. Whitman M.E. and Mattord H.J. Principles of Information Security, 6th ed. Cengage Learning, 2018.

42. Бушуєв, Р. В. Дослідження та програмна реалізація застосування об’єктно-орієнтованих баз даних в ІС : кваліфікаційна магістерська робота : спец. 123 "Комп’ютерна інженерія" / наук. кер. В. В. Босько ; Центральноукраїн. нац. тех. ун-т. - Кропивницький : ЦНТУ, 2022. - 171 с.

43. Горовая, Н. М. (2021). Двофакторна система аутентифікації

корпоративного середовища університету.

44. Куленко М.Я. Основи графічного дизайну : підручник для студентів вищих навч. закладів / Михайло Куленко; МОНУ; Київський нац. ун-т будівництва і архітектури. – 2-ге вид., виправл. та доп. – Київ : Кондор, 2007. – 492с.

45. 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 – СПД ФО Лисенко В.Ф., 2019. – 156 с.

46. Мелешко, Є. В. Алгоритми та структури даних : навч. посіб. / Є. В. Мелешко, М. С. Якименко, Л. І. Поліщук ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : Лисенко В.Ф., 2019. – 156 с.

47. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Компютерні мережі : навч. посіб. Львів : Магнолія, 2013, 250 с.

48. Основи захисту інформації : метод. вказ. до викон. лаб. робіт для студ. за спец. 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”/ [уклад. : О. А. Смірнов, Є. В. Мелешко, О. К. Конопліцька-Слободенюк, В. Д. Хох, С. А. Смірнов]; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2017. – 53 с.

49. Основи захисту інформації : метод. вказ. до викон. лаб. робіт для студ. за спец. 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”/ [уклад. : О. А. Смірнов, Є. В. Мелешко, О. К. Конопліцька-Слободенюк, В. Д. Хох, С. А. Смірнов] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2017. – 53 с.

50. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення // Навчальний посібник. – Кіровоград: Вид. КНТУ, 2012. – 409 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-125.23.0006.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Генкуленко О.Ю.			Програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Мелешко Є.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КБ-19	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для асиметричного шифрування електронних документів.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №12-02 від 05.01.2023 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для асиметричного шифрування електронних документів.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему кібербезпеки для асиметричного шифрування електронних документів;
– 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 Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мові програмування 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 Перелік документів, що розробляються

– Структурна схема системи.
– Функціональна схема системи.
– Діаграма процесів.
– Блок-схема алгоритму роботи програми.
– Пояснювальна записка.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 20.05.2023 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Є.В. Мелешко

Програмне забезпечення системи кібербезпеки для асиметричного шифрування електронних документів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 47

Літера: РП

Кропивницький – 2023 року

// **Main_Form.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace OK
{
    public partial class Main_Form : Form
    {
        Workers_Form workfm = null; //Форма "Працівники"
        Structure_Form structurefm = null; // Форма "Структура"
        Salary_Form salaryfm = null; //Форма "Список посад та окладів"
        Settings_Form settingsfm = null; //Форма "Налаштування"

        public Main_Form()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Maximized;
            Separator1.Width = this.Width;
            Program.Global.ConnectToDatabase();

        }

        private void Workers_Click(object sender, EventArgs e)
        {
            if (workfm == null || workfm.Text == "")
            {
                workfm = new Workers_Form();
                workfm.MdiParent = this;
                workfm.WindowState = FormWindowState.Maximized;
                workfm.Show();
            }
            else if (CheckOpened(workfm.Text))
            {
                workfm.WindowState = FormWindowState.Maximized;
                workfm.Show();
                workfm.Focus();
            }
        }

        private void Structure_Click(object sender, EventArgs e)
        {
            if (structurefm == null || structurefm.Text == "")
            {
                structurefm = new Structure_Form();
                structurefm.MdiParent = this;
                structurefm.WindowState = FormWindowState.Maximized;
                structurefm.Show();
            }
            else if (CheckOpened(structurefm.Text))
            {
                structurefm.WindowState = FormWindowState.Maximized;
                structurefm.Show();
                structurefm.Focus();
            }
        }
```

```
        private bool CheckOpened(string name)
        {
            FormCollection fc = Application.OpenForms;

            foreach (Form frm in fc)
            {
                if (frm.Text == name)
                {
                    return true;
                }
            }
            return false;
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            if (salaryfm == null || salaryfm.Text == "")
            {
                salaryfm = new Salary_Form();
                salaryfm.MdiParent = this;
                salaryfm.WindowState = FormWindowState.Maximized;
                salaryfm.Show();
            }
            else if (CheckOpened(salaryfm.Text))
            {
                salaryfm.WindowState = FormWindowState.Maximized;
                salaryfm.Show();
                salaryfm.Focus();
            }
        }

        private void ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            settingsfm = new Settings_Form();
            settingsfm.ShowDialog();
        }

        private void Main_Form_FormClosed(object sender, FormClosedEventArgs e)
        {
            Program.Global.conn.Close();
        }

    }
}
```

// **Settings_Form.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using OK.Properties;

namespace OK
{
    public partial class Settings_Form : Form
    {
        public Settings_Form()
        {
            InitializeComponent();
        }

        private void Settings_Form_Load(object sender, EventArgs e)
        {
            IPadress.Text = Settings.Default["IP_adress"].ToString();
            Port.Text = Settings.Default["Port"].ToString();
            db_login.Text = Settings.Default["Login"].ToString();
            db_password.Text = Settings.Default["Password"].ToString();

        }

        private void Save_Settings_Click(object sender, EventArgs e)
        {
            Program.Global.conn.Close();
            Program.Global.conn.ConnectionString = "Data Source=" +
IPadress.Text.ToString() + "," + Port.Text.ToString() + ";Initial
Catalog=OKDATABASE" + ";User ID=" + db_login.Text.ToString() + ";Password=" +
db_password.Text.ToString();
            try
            {
                Program.Global.conn.Open();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
                return;
            }
            ConnectionState ConState = Program.Global.conn.State;
            if (ConState == ConnectionState.Open)
            {
                Settings.Default["IP_adress"] = IPadress.Text;
                Settings.Default["Port"] = Port.Text;
                Settings.Default["Login"] = db_login.Text;
                Settings.Default["Password"] = db_password.Text;
                Settings.Default.Save();
                this.Close();
            }
        }

        private void Cancel_Settings_Click(object sender, EventArgs e)
        {
                this.Close();
        }

        private void Apply_Settings_Click(object sender, EventArgs e)
        {
            Program.Global.conn.Close();
```

```
            Program.Global.conn.ConnectionString = "Data Source=" +
IPadress.Text.ToString() + "," + Port.Text.ToString() + ";Initial
Catalog=OKDATABASE" + ";User ID=" + db_login.Text.ToString() + ";Password=" +
db_password.Text.ToString();
            try
            {
                Program.Global.conn.Open();
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
                return;
            }
            ConnectionState ConState = Program.Global.conn.State;
            if (ConState == ConnectionState.Open)
            {
                Settings.Default["IP_adress"] = IPadress.Text;
                Settings.Default["Port"] = Port.Text;
                Settings.Default["Login"] = db_login.Text;
                Settings.Default["Password"] = db_password.Text;
                Settings.Default.Save();
                Apply_Settings.Enabled = false;
            }

        }

        private void Settings_Form_FormClosing(object sender,
FormClosingEventArgs e)
        {
            Main_Form mainfm = new Main_Form();
            mainfm.Refresh();
        }

        private void TextChanged(object sender, EventArgs e)
        {
            Apply_Settings.Enabled = true;
        }
    }
}
```

// **Worker_Edit.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;
using System.Diagnostics;
using System.Configuration;
using System.Threading;
using Microsoft.Office.Interop.Word;
using Microsoft.Office.Core;
using System.Reflection;
using Word = Microsoft.Office.Interop.Word;

namespace OK
{
    public partial class Worker_Edit : Form
    {
        public Worker_Edit()
        {
            InitializeComponent();
        }
        string CurrentDate = System.DateTime.Today.ToString();
        string ParentDir =
Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName;
        SqlDataAdapter EdInstAdapter = null;
        SqlDataAdapter ADInstAdapter = null;
        SqlDataAdapter FamilyAdapter = null;
        SqlDataAdapter PositionsAdapter = null;
        SqlDataAdapter VacationsAdapter = null;
        System.Data.DataTable EdInstTable = new System.Data.DataTable();
        System.Data.DataTable ADInstTable = new System.Data.DataTable();
        System.Data.DataTable FamilyTable = new System.Data.DataTable();
        System.Data.DataTable PositionsTable = new System.Data.DataTable();
        System.Data.DataTable VacationsTable = new System.Data.DataTable();
        DataSet OverviewSet = new DataSet();
        DataSet PassportInfoSet = new DataSet();
        DataSet LastWorkSet = new DataSet();
        DataGridView EdInstitutes = new DataGridView();
        DataGridView ADInstitutes = new DataGridView();
        DataGridView FamilyGrid = new DataGridView();
        DataGridView PositionsGrid = new DataGridView();
        DataGridView VacationsGrid = new DataGridView();
        Bitmap Photo_url;
        Byte[] data = new Byte[0];
        string PathToPhoto = null;
        string ChosenDiploma = null;
        byte[] image;
        string filename;
        string filenameOfApp;
        string filenameOfDis;
        string filenameOfVac;
        Byte[] byteOfApp = new Byte[0];
        Byte[] byteOfDis = new Byte[0];
        Byte[] byteOfVac = new Byte[0];

        #region Змінні
                    string V_idWorker = "";
                    string V_SecondName = "";
                    string V_FirstName = "";
                    string V_ThirdName = "";
                    string V_Nationality = "";
```

```
                string V_IPN = "";
                string V_EDRPOU = "";
                string V_Gender = "";
                string V_MaritalStatus = "";
                string V_SerieOfPassport = "";
                string V_NumberOfPassport = "";
                string V_GiverOfPassport = "";
                string V_RegAdress = "";
                string V_FactAdress = "";
                string V_Ed_level = "";
                string V_NameOfEdInstitution = "";
                string V_SerieOfDiploma = "";
                string V_NumberOfDiploma = "";
                string V_YearOfGraduation = "";
                string V_Speciality = "";
                string V_Qualification = "";
                string V_FormOfEducation = "";
                string V_TypeOfAfterDiploma = "";
                string V_NameOfEdInAfterDiploma = "";
                string V_SerieOfAfterDiploma = "";
                string V_NumberOfAfterDiploma = "";
                string V_YearOfAfterDiplomaGraduation = "";
                string V_Degree = "";
                string V_AcademicStatus = "";
                string V_LastPlaceOfWork = "";
                string V_PositionOnLastWork = "";
                string V_GenExpDays = "";
                string V_GenExpMonths = "";
                string V_GenExpYears = "";
                string V_ExpDaysForPremium = "";
                string V_ExpMonthsForPremium = "";
                string V_ExpYearsForPremium = "";
                string V_DateOfDismissal = "";
                string V_ReasonOfDismissal = "";
                string V_FullnameOfFamily = "";
                string V_BirthYearOfFamily = "";
                string V_TieOfFamily = "";
                string V_NameOfPosition = "";
                string V_CodeOfPosition = "";
                string V_NameofStructuralUnit = "";
                string V_TypeOfWork = "";
                string V_Category = "";
                string V_ReasonOfAppointment = "";
                string V_NumberOfDecreeOfApp = "";
                string V_ReasonOfDis = "";
                string V_NumberOfDecreeOfDis = "";
                string V_TypeOfVacation = "";
                string V_Period = "";
                string V_ReasonOfVac = "";
                string V_NumberOfDecreeOfVac = "";
        #endregion

        //Метод пошуку та заміни тексту в файлі звіту:
        private void FindAndReplace(Microsoft.Office.Interop.Word.Application
wordApp, object findText, object replaceWithText)
        {
            object matchCase = true;
            object matchWholeWord = true;
            object matchWildCards = false;
            object matchSoundLike = false;
            object nmatchAllForms = false;
            object forward = true;
            object format = false;
            object matchKashida = false;
            object matchDiactitics = false;
            object matchAlefHamza = false;
            object matchControl = false;
            object read_only = false;
            object visible = true;
```

```
            object replace = 2;
            object wrap = 1;

            wordApp.Selection.Find.Execute(ref findText,
                        ref matchCase, ref matchWholeWord,
                        ref matchWildCards, ref matchSoundLike,
                        ref nmatchAllForms, ref forward,
                        ref wrap, ref format, ref replaceWithText,
                        ref replace, ref matchKashida,
                        ref matchDiactitics, ref matchAlefHamza,
                        ref matchControl);
        }
        //Масштабування фотографії
        public static Image ScalePhoto(Image image, int maxWidth, int maxHeight)
        {
            var ratioX = (double)maxWidth / image.Width;
            var ratioY = (double)maxHeight / image.Height;
            var ratio = Math.Min(ratioX, ratioY);

            var newWidth = (int)(image.Width * ratio);
            var newHeight = (int)(image.Height * ratio);

            var newImage = new Bitmap(newWidth, newHeight);
            Graphics.FromImage(newImage).DrawImage(image, 0, 0, newWidth,
newHeight);
            return newImage;
        }
        private void Worker_Edit_Load(object sender, EventArgs e)
        {
            EdInstitutes.SelectionChanged +=EdInstitutes_SelectionChanged;
            ADInstitutes.SelectionChanged +=ADInstitutes_SelectionChanged;
            FamilyGrid.SelectionChanged += FamilyGrid_SelectionChanged;
            PositionsGrid.SelectionChanged += PositionsGrid_SelectionChanged;
            ParameterCase();
            FocusFirstPage();
            Info_pages.Appearance = TabAppearance.FlatButtons;
            Info_pages.ItemSize = new Size(0, 1);
            Info_pages.SizeMode = TabSizeMode.Fixed;
            GenExpOfWorking.Text = "Загальний стаж роботи станом на " +
CurrentDate; // Станом на день заповнення
            Fullname.Text = SecondName.Text.ToString() + " " +
FirstName.Text.ToString() + " " + ThirdName.Text.ToString();
        }

        //=====================================================================
        //Загальні відомості
        //=====================================================================
        public void FocusFirstPage()
        {
            Info_pages.SelectedTab = Overview_page;
            Overview.FlatStyle = FlatStyle.Standard;
            Edu.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            Positionbtn.FlatStyle = FlatStyle.Flat;
        }
        private void Overview_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Overview_page;
            Overview.FlatStyle = FlatStyle.Standard;
            Edu.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
```

```
}

//=========================================================================
//Освіта
//=========================================================================
private void Education_Click(object sender, EventArgs e)
{
    Info_pages.SelectedTab = Education_page;
    Edu.FlatStyle = FlatStyle.Standard;
    Overview.FlatStyle = FlatStyle.Flat;
    Passportinfo.FlatStyle = FlatStyle.Flat;
    AfterDiploma.FlatStyle = FlatStyle.Flat;
    LastWorkbtn.FlatStyle = FlatStyle.Flat;
    FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
    Positionbtn.FlatStyle = FlatStyle.Flat;
    Vacationsbtn.FlatStyle = FlatStyle.Flat;
}
//=========================================================================
//Паспортні дані
//=========================================================================
private void Passportinfo_Click(object sender, EventArgs e)
{
    Info_pages.SelectedTab = Passportinfo_page;
    Passportinfo.FlatStyle = FlatStyle.Standard;
    Overview.FlatStyle = FlatStyle.Flat;
    Edu.FlatStyle = FlatStyle.Flat;
    AfterDiploma.FlatStyle = FlatStyle.Flat;
    LastWorkbtn.FlatStyle = FlatStyle.Flat;
    FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
    Positionbtn.FlatStyle = FlatStyle.Flat;
    Vacationsbtn.FlatStyle = FlatStyle.Flat;
}
//=========================================================================
//Післядипломна підготовка
//=========================================================================
private void AfterDiploma_Click(object sender, EventArgs e)
{
    Info_pages.SelectedTab = AfterDiploma_page;
    AfterDiploma.FlatStyle = FlatStyle.Standard;
    Edu.FlatStyle = FlatStyle.Flat;
    Overview.FlatStyle = FlatStyle.Flat;
    Passportinfo.FlatStyle = FlatStyle.Flat;
    LastWorkbtn.FlatStyle = FlatStyle.Flat;
    FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
    Positionbtn.FlatStyle = FlatStyle.Flat;
    Vacationsbtn.FlatStyle = FlatStyle.Flat;
}
//=========================================================================
//Попередня робота
//=========================================================================
private void LastWorkbtn_Click(object sender, EventArgs e)
{
    Info_pages.SelectedTab = LastWork_page;
    LastWorkbtn.FlatStyle = FlatStyle.Standard;
    Edu.FlatStyle = FlatStyle.Flat;
    Overview.FlatStyle = FlatStyle.Flat;
    Passportinfo.FlatStyle = FlatStyle.Flat;
    AfterDiploma.FlatStyle = FlatStyle.Flat;
    FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
    Positionbtn.FlatStyle = FlatStyle.Flat;
    Vacationsbtn.FlatStyle = FlatStyle.Flat;
}
//=========================================================================
//Родинний стан
//=========================================================================
private void FamilyStatusbtn_Click(object sender, EventArgs e)
{
    Info_pages.SelectedTab = FamilyStatus_page;
    FamilyStatusbtn.FlatStyle = FlatStyle.Standard;
```

```
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            Edu.FlatStyle = FlatStyle.Flat;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
        }
        //=========================================================================
        //Посада
        //=========================================================================
        private void Positionbtn_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Position_page;
            Positionbtn.FlatStyle = FlatStyle.Standard;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            Edu.FlatStyle = FlatStyle.Flat;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
            Vacationsbtn.FlatStyle = FlatStyle.Flat;
        }
        //=========================================================================
        //Відпустки
        //=========================================================================
        private void Vacationsbtn_Click(object sender, EventArgs e)
        {
            Info_pages.SelectedTab = Vacation_page;
            Vacationsbtn.FlatStyle = FlatStyle.Standard;
            Positionbtn.FlatStyle = FlatStyle.Flat;
            FamilyStatusbtn.FlatStyle = FlatStyle.Flat;
            LastWorkbtn.FlatStyle = FlatStyle.Flat;
            Edu.FlatStyle = FlatStyle.Flat;
            Overview.FlatStyle = FlatStyle.Flat;
            Passportinfo.FlatStyle = FlatStyle.Flat;
            AfterDiploma.FlatStyle = FlatStyle.Flat;
        }
        //=========================================================================
        //Вибрати фото
        //=========================================================================
        private void Add_photo_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog open = new OpenFileDialog();
                open.Filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp)|*.jpg;
*.jpeg; *.gif; *.bmp";
                if (open.ShowDialog() == DialogResult.OK)
                {
                    Photo_url = new Bitmap(open.FileName);
                    Photo.Image = Photo_url;
                    PathToPhoto = open.FileName;
                }
            }
            catch (Exception)
            {
                throw new ApplicationException("Не вдалося завантижити фото");
            }
            image = File.ReadAllBytes(PathToPhoto);
        }

        //=========================================================================
        //Підсказки
        //=========================================================================
        private void Speciality_Click(object sender, EventArgs e)
        {
            ToolTip tt = new ToolTip();
```

```
            tt.SetToolTip(Speciality, "Спеціальність (професія) за
дипломом(свідоцтвом) відповідно із займаною посадою");
        }

        private void Qualification_Click(object sender, EventArgs e)
        {
            ToolTip tt = new ToolTip();
            tt.SetToolTip(Qualification, "Кваліфікація за дипломом
(свідоцтвом)");
        }

        //=====================================================================
        //Завантаження даних
        //=====================================================================
        public void LoadInformation()
        {
            try
            {

//=====================================================================
                //Загальні відомості

//=====================================================================
                string OverviewQuery = "SELECT * FROM [Worker] WHERE [IdWorker]
= '" + Program.Global.ChosenWorker + "'";
                SqlDataAdapter OverviewAdapter = new
SqlDataAdapter(OverviewQuery, Program.Global.conn);
                OverviewAdapter.Fill(OverviewSet);

//=====================================================================
                idWorker.Text =
OverviewSet.Tables[0].Rows[0]["IdWorker"].ToString();
                SecondName.Text =
OverviewSet.Tables[0].Rows[0]["SecondName"].ToString();
                FirstName.Text =
OverviewSet.Tables[0].Rows[0]["FirstName"].ToString();
                ThirdName.Text =
OverviewSet.Tables[0].Rows[0]["ThirdName"].ToString();
                Birthday.Text =
OverviewSet.Tables[0].Rows[0]["Birthday"].ToString();
                Gender.SelectedIndex =
Gender.Items.IndexOf(OverviewSet.Tables[0].Rows[0]["Gender"].ToString());
                Nationality.Text =
OverviewSet.Tables[0].Rows[0]["Nationality"].ToString();
                IPN.Text = OverviewSet.Tables[0].Rows[0]["IPN"].ToString();
                EDRPOU.Text =
OverviewSet.Tables[0].Rows[0]["EDRPOU"].ToString();
                MaritalStatus.Text =
OverviewSet.Tables[0].Rows[0]["FamilyStatus"].ToString();
                CompletionDate.Text =
OverviewSet.Tables[0].Rows[0]["CompletionDate"].ToString();
                data = (Byte[])(OverviewSet.Tables[0].Rows[0]["Photo"]);
                MemoryStream mem = new MemoryStream(data);
                Photo.Image = Image.FromStream(mem);


//=====================================================================
                //Паспортні дані

//=====================================================================
                string PassportInfoQuery = "SELECT * FROM [PassportInfo] WHERE
[IdWorker] = '" + Program.Global.ChosenWorker + "'";
                SqlDataAdapter PassportInfoAdapter = new
SqlDataAdapter(PassportInfoQuery, Program.Global.conn);
                PassportInfoAdapter.Fill(PassportInfoSet);

//=====================================================================
                SerieOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["SerieOfPassport"].ToString();
```

```
                NumberOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["NumberOfPassport"].ToString();
                DateOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["DateOfPassport"].ToString();
                GiverOfPassport.Text =
PassportInfoSet.Tables[0].Rows[0]["GiverOfPassport"].ToString();
                RegAdress.Text =
PassportInfoSet.Tables[0].Rows[0]["RegAdress"].ToString();
                FactAdress.Text =
PassportInfoSet.Tables[0].Rows[0]["FactAdress"].ToString();


//=====================================================================
                //Попередня робота

//=====================================================================
                string LastWorkQuery = "SELECT * FROM [LastWork] WHERE
[IdWorker] = '" + Program.Global.ChosenWorker + "'";
                SqlDataAdapter LastWorkAdapter = new
SqlDataAdapter(LastWorkQuery, Program.Global.conn);
                LastWorkAdapter.Fill(LastWorkSet);

//=====================================================================
                LastPlaceOfWork.Text =
LastWorkSet.Tables[0].Rows[0]["LastPlaceOfWork"].ToString();
                PositionOnLastWork.Text =
LastWorkSet.Tables[0].Rows[0]["PositionOnLastWork"].ToString();
                GenExpDays.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpDays"].ToString();
                GenExpMonths.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpMonths"].ToString();
                GenExpYears.Text =
LastWorkSet.Tables[0].Rows[0]["GenExpYears"].ToString();
                ExpDaysForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpDaysForPremium"].ToString();
                ExpMonthsForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpMonthsForPremium"].ToString();
                ExpYearsForPremium.Text =
LastWorkSet.Tables[0].Rows[0]["ExpYearsForPremium"].ToString();
                DateOfDismissal.Text =
LastWorkSet.Tables[0].Rows[0]["DateOfDismissal"].ToString();
                ReasonOfDismissal.Text =
LastWorkSet.Tables[0].Rows[0]["ReasonOfDismissal"].ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
        //=====================================================================
        //Створення таблиць
        //=====================================================================
        public void CreateEdInstitutes()
        {
            EdInstitutes.Columns.Clear();
            EdInstitutes.AutoGenerateColumns = false;
            Education_page.Controls.Add(EdInstitutes);
            EdInstitutes.BackgroundColor = Color.White;
            EdInstitutes.AllowUserToAddRows = false;
            EdInstitutes.AllowUserToDeleteRows = false;
            EdInstitutes.ReadOnly = true;
            EdInstitutes.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            EdInstitutes.Height = 210;
            EdInstitutes.Width = 595;
            EdInstitutes.Location = new System.Drawing.Point(6,358);
            EdInstitutes.Columns.Add("Diploma", "Диплом");
            EdInstitutes.Columns.Add("Ed_level", "Освіта");
```

```
            EdInstitutes.Columns.Add("NameOfEdInstitution", "Назва освітнього
закладу");
            EdInstitutes.Columns.Add("YearOfGraduation", "Рік закінчення");
            EdInstitutes.Columns[0].Width = 70;
            EdInstitutes.Columns[1].Width = 140;
            EdInstitutes.Columns[2].Width = 255;
            EdInstitutes.Columns[3].Width = 87;
            EdInstitutes.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.EdInstitutes_RowP
ostPaint);
        }
        public void CreateADInstitutes()
        {
            ADInstitutes.Columns.Clear();
            ADInstitutes.AutoGenerateColumns = false;
            AfterDiploma_page.Controls.Add(ADInstitutes);
            ADInstitutes.BackgroundColor = Color.White;
            ADInstitutes.AllowUserToAddRows = false;
            ADInstitutes.AllowUserToDeleteRows = false;
            ADInstitutes.ReadOnly = true;
            ADInstitutes.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            ADInstitutes.Height = 256;
            ADInstitutes.Width = 595;
            ADInstitutes.Location = new System.Drawing.Point(6, 315);
            ADInstitutes.Columns.Add("ADDiploma", "Диплом");
            ADInstitutes.Columns.Add("TypeOfAD", "Вид післядипломної
підготовки");
            ADInstitutes.Columns.Add("NameOfADInst", "Назва ВНЗ");
            ADInstitutes.Columns.Add("YearOfADGrad", "Рік закінчення");
            ADInstitutes.Columns[0].Width = 70;
            ADInstitutes.Columns[1].Width = 170;
            ADInstitutes.Columns[2].Width = 210;
            ADInstitutes.Columns[3].Width = 100;
            ADInstitutes.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.ADInstitutes_RowP
ostPaint);
        }
        public void CreateFamilyGrid()
        {
            FamilyGrid.Columns.Clear();
            FamilyGrid.AutoGenerateColumns = false;
            FamilyStatus_page.Controls.Add(FamilyGrid);
            FamilyGrid.BackgroundColor = Color.White;
            FamilyGrid.AllowUserToAddRows = false;
            FamilyGrid.AllowUserToDeleteRows = false;
            FamilyGrid.ReadOnly = true;
            FamilyGrid.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
            FamilyGrid.Height = 374;
            FamilyGrid.Width = 595;
            FamilyGrid.Location = new System.Drawing.Point(6, 197);
            FamilyGrid.Columns.Add("Fullname", "ПІБ");
            FamilyGrid.Columns.Add("BirthYear", "Рік народження");
            FamilyGrid.Columns.Add("TieOfFamily", "Ступінь родинного зв'язку");
            FamilyGrid.Columns[0].Width = 250;
            FamilyGrid.Columns[1].Width = 100;
            FamilyGrid.Columns[2].Width = 200;
            FamilyGrid.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.FamilyGrid_RowPos
tPaint);
        }
        public void CreatePositionsGrid()
        {
            PositionsGrid.Columns.Clear();
            PositionsGrid.AutoGenerateColumns = false;
            Position_page.Controls.Add(PositionsGrid);
            PositionsGrid.BackgroundColor = Color.White;
            PositionsGrid.AllowUserToAddRows = false;
            PositionsGrid.AllowUserToDeleteRows = false;
```

```
            PositionsGrid.ReadOnly = true;
            PositionsGrid.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            PositionsGrid.Height = 119;
            PositionsGrid.Width = 450;
            PositionsGrid.Location = new System.Drawing.Point(6, 448);
            PositionsGrid.Columns.Add("DateOfApp", "Дата");
            PositionsGrid.Columns.Add("PositionName", "Посада");
            PositionsGrid.Columns.Add("StructUnit", "Структурний підрозділ");
            PositionsGrid.Columns[0].Width = 170;
            PositionsGrid.Columns[1].Width = 180;
            PositionsGrid.Columns[2].Width = 150;
            PositionsGrid.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.PositionsGrid_Row
PostPaint);
        }
        public void CreateVacationsGrid()
        {
            VacationsGrid.Columns.Clear();
            VacationsGrid.AutoGenerateColumns = false;
            Vacation_page.Controls.Add(VacationsGrid);
            VacationsGrid.BackgroundColor = Color.White;
            VacationsGrid.AllowUserToAddRows = false;
            VacationsGrid.AllowUserToDeleteRows = false;
            VacationsGrid.ReadOnly = true;
            VacationsGrid.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;
            VacationsGrid.Height = 349;
            VacationsGrid.Width = 595;
            VacationsGrid.Location = new System.Drawing.Point(6, 218);
            VacationsGrid.Columns.Add("TypeOfVacation", "Вид відпустки");
            VacationsGrid.Columns.Add("BeginOfVacation", "Початок");
            VacationsGrid.Columns.Add("EndOfVacation", "Кінець");
            PositionsGrid.Columns[0].Width = 180;
            VacationsGrid.Columns[1].Width = 185;
            VacationsGrid.Columns[2].Width = 185;
            VacationsGrid.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.VacationsGrid_Row
PostPaint);
        }
        //=====================================================================
        //Завантаження даних в таблиці
        //=====================================================================
        public void LoadEducation()
        {
            string ToEdInstGrid = "SELECT * FROM [Education]  WHERE [IdWorker] =
'" + idWorker.Text.ToString() + "'";
            EdInstAdapter = new SqlDataAdapter(ToEdInstGrid,
Program.Global.conn);
            EdInstTable.Clear();
            EdInstAdapter.Fill(EdInstTable);
            EdInstitutes.Columns[0].DataPropertyName = "Diploma";
            EdInstitutes.Columns[1].DataPropertyName = "Ed_level";
            EdInstitutes.Columns[2].DataPropertyName = "NameOfEdInstitution";
            EdInstitutes.Columns[3].DataPropertyName = "YearOfGraduation";
            EdInstitutes.DataSource = EdInstTable;
        }
        public void LoadAD()
        {
            string ToADInstGrid = "SELECT * FROM [AfterDiploma]  WHERE
[IdWorker] = '" + idWorker.Text.ToString() + "'";
            ADInstAdapter = new SqlDataAdapter(ToADInstGrid,
Program.Global.conn);
            ADInstTable.Clear();
            ADInstAdapter.Fill(ADInstTable);
            ADInstitutes.Columns[0].DataPropertyName = "Diploma";
            ADInstitutes.Columns[1].DataPropertyName = "TypeOfAfterDiploma";
            ADInstitutes.Columns[2].DataPropertyName = "NameOfEdInAfterDiploma";
```

```
            ADInstitutes.Columns[3].DataPropertyName =
"YearOfAfterDiplomaGraduation";
            ADInstitutes.DataSource = ADInstTable;
        }
        public void LoadFamily()
        {
            string ToFamilyGrid = "SELECT * FROM [FamilyStatus]  WHERE
[IdWorker] = '" + idWorker.Text.ToString() + "'";
            FamilyAdapter = new SqlDataAdapter(ToFamilyGrid,
Program.Global.conn);
            FamilyTable.Clear();
            FamilyAdapter.Fill(FamilyTable);
            FamilyGrid.Columns[0].DataPropertyName = "FullnameOfFamily";
            FamilyGrid.Columns[1].DataPropertyName = "BirthYearOfFamily";
            FamilyGrid.Columns[2].DataPropertyName = "TieOfFamily";
            FamilyGrid.DataSource = FamilyTable;
        }
        public void LoadPositions()
        {
            string ToPositionsGrid = "SELECT * FROM [Positions]  WHERE
[IdWorker] = '" + idWorker.Text.ToString() + "'";
            PositionsAdapter = new SqlDataAdapter(ToPositionsGrid,
Program.Global.conn);
            PositionsTable.Clear();
            PositionsAdapter.Fill(PositionsTable);
            PositionsGrid.Columns[0].DataPropertyName = "DateOfAppointment";
            PositionsGrid.Columns[1].DataPropertyName = "NameOfPosition";
            PositionsGrid.Columns[2].DataPropertyName = "NameofStructuralUnit";
            PositionsGrid.DataSource = PositionsTable;
        }
        public void LoadVacations()
        {
            string ToVacationsGrid = "SELECT * FROM [Vacations]  WHERE
[IdWorker] = '" + idWorker.Text.ToString() + "'";
            VacationsAdapter = new SqlDataAdapter(ToVacationsGrid,
Program.Global.conn);
            VacationsTable.Clear();
            VacationsAdapter.Fill(VacationsTable);
            VacationsGrid.Columns[0].DataPropertyName = "TypeOfVacation";
            VacationsGrid.Columns[1].DataPropertyName = "BeginOfVacation";
            VacationsGrid.Columns[2].DataPropertyName = "EndOfVacation";
            VacationsGrid.DataSource = VacationsTable;
        }
        //=====================================================================
        //Параметр
        //=====================================================================
        public void ParameterCase()
        {
            switch(Program.Global.Parameter)
            {
                case "View":
                    EdInstitutes.DataSource = null;
                    ADInstitutes.DataSource = null;
                    FamilyGrid.DataSource = null;
                    PositionsGrid.DataSource = null;
                    VacationsGrid.DataSource = null;
                    LoadInformation();
                    CreateEdInstitutes();
                    CreateADInstitutes();
                    CreateFamilyGrid();
                    CreatePositionsGrid();
                    CreateVacationsGrid();
                    LoadEducation();
                    LoadAD();
                    LoadFamily();
                    LoadPositions();
                    LoadVacations();
                    Add_photo.Visible = false;
                    EditingEdInst.Visible = false;
```

```
            EditingADipInst.Visible = false;
            EditingFamily.Visible = false;
            EditingPositions.Visible = false;
            EditingVacations.Visible = false;
            ReasonOfDismissal.Enabled = false;
            //================================
            //Блокування можливості редагування
            //================================
            ChangeReadOnly(true);
            //================================
            //================================
            //Кнопка
            //================================
            CaseButton.Text = "Закрити";
            CaseButton.TextAlign = ContentAlignment.MiddleRight;
            CaseButton.Image = Properties.Resources.View;
            CaseButton.ImageAlign = ContentAlignment.MiddleLeft;
            //================================
            UploadTextOfApp.Enabled = false;
            OpenTextOfApp.Enabled = true;
            UploadTextOfDis.Enabled = false;
            OpenTextOfDis.Enabled = true;
            UploadTextOfVac.Enabled = false;
            OpenTextOfVac.Enabled = true;
            break;
        case "Edit":
            ChangeReadOnly(false);
            EdInstitutes.DataSource = null;
            ADInstitutes.DataSource = null;
            FamilyGrid.DataSource = null;
            PositionsGrid.DataSource = null;
            VacationsGrid.DataSource = null;
            CreateEdInstitutes();
            CreateADInstitutes();
            CreateFamilyGrid();
            CreatePositionsGrid();
            CreateVacationsGrid();
            LoadInformation();
            LoadEducation();
            LoadAD();
            LoadFamily();
            LoadPositions();
            LoadVacations();
            Add_photo.Visible = true;
            EditingEdInst.Visible = true;
            EditingADipInst.Visible = true;
            EditingFamily.Visible = true;
            EditingPositions.Visible = true;
            EditingVacations.Visible = true;
            ReasonOfDismissal.Enabled = true;
            //================================
            //Кнопка
            //================================
            CaseButton.Text = "Змінити";
            CaseButton.TextAlign = ContentAlignment.MiddleRight;
            CaseButton.Image = Properties.Resources.Edit;
            CaseButton.ImageAlign = ContentAlignment.MiddleLeft;
            //================================
            UploadTextOfApp.Enabled = true;
            OpenTextOfApp.Enabled = true;
            UploadTextOfDis.Enabled = true;
            OpenTextOfDis.Enabled = true;
            UploadTextOfVac.Enabled = true;
            OpenTextOfVac.Enabled = true;
            break;
        case "Add":
            EdInstitutes.AutoGenerateColumns = false;
            ADInstitutes.AutoGenerateColumns = false;
            FamilyGrid.AutoGenerateColumns = false;
```

```
PositionsGrid.AutoGenerateColumns = false;
VacationsGrid.AutoGenerateColumns = false;
EdInstitutes.Columns.Clear();
ADInstitutes.Columns.Clear();
FamilyGrid.Columns.Clear();
PositionsGrid.Columns.Clear();
VacationsGrid.Columns.Clear();
EdInstitutes.DataSource = null;
ADInstitutes.DataSource = null;
FamilyGrid.DataSource = null;
PositionsGrid.DataSource = null;
VacationsGrid.DataSource = null;
Add_photo.Visible = true;
EditingEdInst.Visible = true;
EditingADipInst.Visible = true;
EditingFamily.Visible = true;
EditingPositions.Visible = true;
EditingVacations.Visible = true;
ReasonOfDismissal.Enabled = true;
ChangeReadOnly(false);
//================================
CreateEdInstitutes();
CreateADInstitutes();
CreateFamilyGrid();
CreatePositionsGrid();
CreateVacationsGrid();
idWorker.Text = null;
SecondName.Text = null;
FirstName.Text = null;
ThirdName.Text = null;
Nationality.Text = null;
IPN.Text = null;
EDRPOU.Text = null;
Gender.Text = null;
MaritalStatus.Text = null;
Birthday.Text = "01.01.1950";
Photo.Image = null;
SerieOfPassport.Text = null;
NumberOfPassport.Text = null;
DateOfPassport.Text = "01.01.1950";
GiverOfPassport.Text = null;
RegAdress.Text = null;
FactAdress.Text = null;
Ed_level.SelectedIndex = -1;
NameOfEdInstitution.Text = null;
SerieOfDiploma.Text = null;
NumberOfDiploma.Text = null;
YearOfGraduation.Text = null;
Speciality.Text = null;
Qualification.Text = null;
FormOfEducation.SelectedIndex = -1;
TypeOfAfterDiploma.SelectedIndex = -1;
NameOfEdInAfterDiploma.Text = null;
SerieOfAfterDiploma.Text = null;
NumberOfAfterDiploma.Text = null;
YearOfAfterDiplomaGraduation.Text = null;
Degree.SelectedIndex = -1;
AcademicStatus.SelectedIndex = -1;
LastPlaceOfWork.Text = null;
PositionOnLastWork.Text = null;
GenExpDays.Text = null;
GenExpMonths.Text = null;
GenExpYears.Text = null;
ExpDaysForPremium.Text = null;
ExpMonthsForPremium.Text = null;
ExpYearsForPremium.Text = null;
DateOfDismissal.Text = "01.01.1950";
ReasonOfDismissal.SelectedIndex = -1;
FullnameOfFamily.Text = null;
```

```
                    BirthYearOfFamily.Text = null;
                    TieOfFamily.Text = null;
                    NameOfPosition.Text = null;
                    CodeOfPosition.Text = null;
                    NameofStructuralUnit.Text = null;
                    TypeOfWork.SelectedIndex = -1;
                    Category.Text = null;
                    DateOfAppointment.Text = "01.01.2000";
                    ReasonOfAppointment.Text = null;
                    NumberOfDecreeOfApp.Text = null;
                    DateOfDis.Text = "01.01.2000";
                    ReasonOfDis.Text = null;
                    NumberOfDecreeOfDis.Text = null;
                    CompletionDate.Value = System.DateTime.Today;
                    TypeOfVacation.SelectedIndex = -1;
                    Period.Text = null;
                    BeginOfVacation.Value = System.DateTime.Today;
                    EndOfVacation.Value = System.DateTime.Today;
                    ReasonOfVac.Text = null;
                    NumberOfDecreeOfVac.Text = null;
                    //================================
                    UploadTextOfApp.Enabled = true;
                    OpenTextOfApp.Enabled = false;
                    UploadTextOfDis.Enabled = true;
                    OpenTextOfDis.Enabled = false;
                    UploadTextOfVac.Enabled = true;
                    OpenTextOfVac.Enabled = false;
                    //================================
                    //Кнопка
                    //================================
                    CaseButton.Text = "Додати";
                    CaseButton.TextAlign = ContentAlignment.MiddleRight;
                    CaseButton.Image = Properties.Resources.Add;
                    CaseButton.ImageAlign = ContentAlignment.MiddleLeft;
                    //================================
                    break;
                }
            }
            //=====================================================================
            //Номер запису в таблиці
            //=====================================================================
            private void EdInstitutes_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
            {
                using (SolidBrush b = new
SolidBrush(EdInstitutes.RowHeadersDefaultCellStyle.ForeColor))
                {
                    e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 15, e.RowBounds.Location.Y
+ 5);
                }
            }
            private void ADInstitutes_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
            {
                using (SolidBrush b = new
SolidBrush(ADInstitutes.RowHeadersDefaultCellStyle.ForeColor))
                {
                    e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 15, e.RowBounds.Location.Y
+ 5);
                }
            }
            private void FamilyGrid_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
            {
                using (SolidBrush b = new
SolidBrush(FamilyGrid.RowHeadersDefaultCellStyle.ForeColor))
                {
```

```
                e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 15, e.RowBounds.Location.Y
+ 5);
            }
        }
        private void PositionsGrid_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
        {
            using (SolidBrush b = new
SolidBrush(PositionsGrid.RowHeadersDefaultCellStyle.ForeColor))
            {
                e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 15, e.RowBounds.Location.Y
+ 5);
            }
        }
        private void VacationsGrid_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
        {
            using (SolidBrush b = new
SolidBrush(VacationsGrid.RowHeadersDefaultCellStyle.ForeColor))
            {
                e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 15, e.RowBounds.Location.Y
+ 5);
            }
        }
        //=====================================================================
        public void ChangeReadOnly(bool tbstate)
        {
            idWorker.ReadOnly = tbstate;
            SecondName.ReadOnly = tbstate;
            FirstName.ReadOnly = tbstate;
            ThirdName.ReadOnly = tbstate;
            Nationality.ReadOnly = tbstate;
            IPN.ReadOnly = tbstate;
            EDRPOU.ReadOnly = tbstate;
            SerieOfPassport.ReadOnly = tbstate;
            NumberOfPassport.ReadOnly = tbstate;
            GiverOfPassport.ReadOnly = tbstate;
            RegAdress.ReadOnly = tbstate;
            FactAdress.ReadOnly = tbstate;
            NameOfEdInstitution.ReadOnly = tbstate;
            SerieOfDiploma.ReadOnly = tbstate;
            NumberOfDiploma.ReadOnly = tbstate;
            YearOfGraduation.ReadOnly = tbstate;
            Speciality.ReadOnly = tbstate;
            Qualification.ReadOnly = tbstate;
            NameOfEdInAfterDiploma.ReadOnly = tbstate;
            SerieOfAfterDiploma.ReadOnly = tbstate;
            NumberOfAfterDiploma.ReadOnly = tbstate;
            YearOfAfterDiplomaGraduation.ReadOnly = tbstate;
            LastPlaceOfWork.ReadOnly = tbstate;
            PositionOnLastWork.ReadOnly = tbstate;
            GenExpDays.ReadOnly = tbstate;
            GenExpMonths.ReadOnly = tbstate;
            GenExpYears.ReadOnly = tbstate;
            ExpDaysForPremium.ReadOnly = tbstate;
            ExpMonthsForPremium.ReadOnly = tbstate;
            ExpYearsForPremium.ReadOnly = tbstate;
            FullnameOfFamily.ReadOnly = tbstate;
            BirthYearOfFamily.ReadOnly = tbstate;
            TieOfFamily.ReadOnly = tbstate;
            NameOfPosition.ReadOnly = tbstate;
            CodeOfPosition.ReadOnly = tbstate;
            NameofStructuralUnit.ReadOnly = tbstate;
            Category.ReadOnly = tbstate;
            ReasonOfAppointment.ReadOnly = tbstate;
            NumberOfDecreeOfApp.ReadOnly = tbstate;
```

```
            ReasonOfDis.ReadOnly = tbstate;
            NumberOfDecreeOfDis.ReadOnly = tbstate;
            Period.ReadOnly = tbstate;
            ReasonOfVac.ReadOnly = tbstate;
            NumberOfDecreeOfVac.ReadOnly = tbstate;
        }

        private void AddEdInst_Click(object sender, EventArgs e)
        {
            V_idWorker = idWorker.Text.ToString();
            V_SerieOfDiploma =  SerieOfDiploma.Text.ToString();
            V_NumberOfDiploma = NumberOfDiploma.Text.ToString();
            V_Ed_level = Ed_level.SelectedItem.ToString();
            V_NameOfEdInstitution = NameOfEdInstitution.Text.ToString();
            V_YearOfGraduation = YearOfGraduation.Text.ToString();
            if (Ed_level.SelectedIndex.ToString() == "0" ||
Ed_level.SelectedIndex.ToString() == "1")
            {
                V_Speciality = "";
                V_Qualification = "";
                V_FormOfEducation = "";
            }
            else
            {
                V_Speciality = Speciality.Text.ToString();
                V_Qualification = Qualification.Text.ToString();
                V_FormOfEducation = FormOfEducation.SelectedItem.ToString();
            }
            try
            {
                string InsertInstitute = "INSERT INTO [Education]
([IdWorker],[Diploma],[Ed_level],[NameOfEdInstitution],
[YearOfGraduation],[Speciality],[Qualification],[FormOfEducation]) VALUES
(@IdWorker, @Diploma, @Ed_level, @NameOfEdInstitution, @YearOfGraduation,
@Speciality, @Qualification, @FormOfEducation)";
                SqlCommand insInst = new SqlCommand(InsertInstitute,
Program.Global.conn);
                insInst.Parameters.AddWithValue("@IdWorker", V_idWorker);
                insInst.Parameters.AddWithValue("@Diploma", V_SerieOfDiploma + "
" + V_NumberOfDiploma);
                insInst.Parameters.AddWithValue("@Ed_level", V_Ed_level);
                insInst.Parameters.AddWithValue("@NameOfEdInstitution",
V_NameOfEdInstitution);
                insInst.Parameters.AddWithValue("@YearOfGraduation",
V_YearOfGraduation);
                insInst.Parameters.AddWithValue("@Speciality", V_Speciality);
                insInst.Parameters.AddWithValue("@Qualification",
V_Qualification);
                insInst.Parameters.AddWithValue("@FormOfEducation",
V_FormOfEducation);
                insInst.ExecuteNonQuery();
                LoadEducation();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void DeleteEdInst_Click(object sender, EventArgs e)
        {
            try
            {
                string DeleteInstitute = "DELETE FROM [Education] WHERE
[IdWorker] = @IdWorker AND [Diploma] = @Diploma";
                SqlCommand delInst = new SqlCommand(DeleteInstitute,
Program.Global.conn);
                delInst.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
```

```
                delInst.Parameters.AddWithValue("@Diploma",
SerieOfDiploma.Text.ToString() + " " + NumberOfDiploma.Text.ToString());
                delInst.ExecuteNonQuery();
                EdInstTable.Clear();
                EdInstAdapter.Fill(EdInstTable);
                EdInstitutes.DataSource = EdInstTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
        private void EdInstitutes_SelectionChanged(object sender, EventArgs e)
        {
            if (EdInstitutes.CurrentRow != null && EdInstitutes.CurrentRow.Index
>=0)
            {
                int index = EdInstitutes.CurrentRow.Index;
                ShowEdInstitute(index);
            }

        }
        private void ADInstitutes_SelectionChanged(object sender, EventArgs e)
        {
            if (ADInstitutes.CurrentRow != null && ADInstitutes.CurrentRow.Index
>= 0)
            {
                int index = ADInstitutes.CurrentRow.Index;
                ShowADInstitute(index);
            }

        }
        private void FamilyGrid_SelectionChanged(object sender, EventArgs e)
        {
            if (FamilyGrid.CurrentRow != null && FamilyGrid.CurrentRow.Index >=
0)
            {
                int index = FamilyGrid.CurrentRow.Index;
                ShowFamilyGrid(index);
            }
        }
        private void PositionsGrid_SelectionChanged(object sender, EventArgs e)
        {
            if (PositionsGrid.CurrentRow != null &&
PositionsGrid.CurrentRow.Index >= 0)
            {
                int index = PositionsGrid.CurrentRow.Index;
                ShowPositionsGrid(index);
            }
        }
        private void VacationsGrid_SelectionChanged(object sender, EventArgs e)
        {
            if (VacationsGrid.CurrentRow != null &&
VacationsGrid.CurrentRow.Index >= 0)
            {
                int index = VacationsGrid.CurrentRow.Index;
                ShowVacationsGrid(index);
            }
        }
        public void ShowEdInstitute(int index)
        {
            ChosenDiploma =
EdInstitutes.Rows[index].Cells["Diploma"].Value.ToString();

//=====================================================================
            Ed_level.Text = EdInstTable.Rows[index]["Ed_level"].ToString();
            NameOfEdInstitution.Text =
EdInstTable.Rows[index]["NameOfEdInstitution"].ToString();
```

```
            YearOfGraduation.Text =
EdInstTable.Rows[index]["YearOfGraduation"].ToString();
            Speciality.Text = EdInstTable.Rows[index]["Speciality"].ToString();
            Qualification.Text =
EdInstTable.Rows[index]["Qualification"].ToString();
            FormOfEducation.Text =
EdInstTable.Rows[index]["FormOfEducation"].ToString();

//=================================================================
            if (ChosenDiploma != "")
            {
                string[] SerieAndNumber =
EdInstTable.Rows[index]["Diploma"].ToString().Split(' ');
                SerieOfDiploma.Text = SerieAndNumber[0];
                NumberOfDiploma.Text = SerieAndNumber[1];
            }
        }
        public void ShowADInstitute(int index)
        {
            string ChosenDiploma =
ADInstitutes.Rows[index].Cells["ADDiploma"].Value.ToString();

//=================================================================
            TypeOfAfterDiploma.Text =
ADInstTable.Rows[index]["TypeOfAfterDiploma"].ToString();
            NameOfEdInAfterDiploma.Text =
ADInstTable.Rows[index]["NameOfEdInAfterDiploma"].ToString();
            YearOfAfterDiplomaGraduation.Text =
ADInstTable.Rows[index]["YearOfAfterDiplomaGraduation"].ToString();
            Degree.Text = ADInstTable.Rows[index]["Degree"].ToString();
            AcademicStatus.Text =
ADInstTable.Rows[index]["AcademicStatus"].ToString();

//=================================================================
            if (ChosenDiploma != "")
            {
                string[] SerieAndNumber =
ADInstTable.Rows[index]["Diploma"].ToString().Split(' ');
                SerieOfAfterDiploma.Text = SerieAndNumber[0];
                NumberOfAfterDiploma.Text = SerieAndNumber[1];
            }
        }
        public void ShowFamilyGrid(int index)
        {

//=================================================================
            FullnameOfFamily.Text =
FamilyTable.Rows[index]["FullnameOfFamily"].ToString();
            BirthYearOfFamily.Text =
FamilyTable.Rows[index]["BirthYearOfFamily"].ToString();
            TieOfFamily.Text =
FamilyTable.Rows[index]["TieOfFamily"].ToString();

//=================================================================
        }
        public void ShowPositionsGrid(int index)
        {

//=================================================================
            NameOfPosition.Text =
PositionsTable.Rows[index]["NameOfPosition"].ToString();
            CodeOfPosition.Text =
PositionsTable.Rows[index]["CodeOfPosition"].ToString();
            NameofStructuralUnit.Text =
PositionsTable.Rows[index]["NameofStructuralUnit"].ToString();
            TypeOfWork.Text =
PositionsTable.Rows[index]["TypeOfWork"].ToString();
            Category.Text = PositionsTable.Rows[index]["Category"].ToString();
```

```
            DateOfAppointment.Text =
PositionsTable.Rows[index]["DateOfAppointment"].ToString();
            ReasonOfAppointment.Text =
PositionsTable.Rows[index]["ReasonOfAppointment"].ToString();
            NumberOfDecreeOfApp.Text =
PositionsTable.Rows[index]["NumberOfDecreeOfApp"].ToString();
            DateOfDis.Text = PositionsTable.Rows[index]["DateOfDis"].ToString();
            ReasonOfDis.Text =
PositionsTable.Rows[index]["ReasonOfDis"].ToString();
            NumberOfDecreeOfDis.Text =
PositionsTable.Rows[index]["NumberOfDecreeOfDis"].ToString();

//=====================================================================
        }
        public void ShowVacationsGrid(int index)
        {

//=====================================================================
            TypeOfVacation.Text =
VacationsTable.Rows[index]["TypeOfVacation"].ToString();
            Period.Text = VacationsTable.Rows[index]["Period"].ToString();
            BeginOfVacation.Text =
VacationsTable.Rows[index]["BeginOfVacation"].ToString();
            EndOfVacation.Text =
VacationsTable.Rows[index]["EndOfVacation"].ToString();
            ReasonOfVac.Text =
VacationsTable.Rows[index]["ReasonOfVac"].ToString();
            NumberOfDecreeOfVac.Text =
VacationsTable.Rows[index]["NumberOfDecreeOfVac"].ToString();

//=====================================================================
        }
        private void AddAfterDiplomaInst_Click(object sender, EventArgs e)
        {
            V_idWorker = idWorker.Text.ToString();
            V_SerieOfAfterDiploma = SerieOfAfterDiploma.Text.ToString();
            V_NumberOfAfterDiploma = NumberOfAfterDiploma.Text.ToString();
            V_TypeOfAfterDiploma = TypeOfAfterDiploma.SelectedItem.ToString();
            V_NameOfEdInAfterDiploma = NameOfEdInAfterDiploma.Text.ToString();
            V_YearOfAfterDiplomaGraduation =
YearOfAfterDiplomaGraduation.Text.ToString();
            V_Degree = Degree.Text.ToString();
            V_AcademicStatus = AcademicStatus.Text.ToString();
            try
            {
                string InsertADInstitute = "INSERT INTO [AfterDiploma]
([IdWorker],[Diploma],[TypeOfAfterDiploma],[NameOfEdInAfterDiploma],
[YearOfAfterDiplomaGraduation],[Degree],[AcademicStatus]) VALUES (@IdWorker,
@Diploma, @TypeOfAfterDiploma, @NameOfEdInAfterDiploma,
@YearOfAfterDiplomaGraduation, @Degree, @AcademicStatus)";
                SqlCommand insADInst = new SqlCommand(InsertADInstitute,
Program.Global.conn);
                insADInst.Parameters.AddWithValue("@IdWorker", V_idWorker);
                insADInst.Parameters.AddWithValue("@Diploma",
V_SerieOfAfterDiploma + " " + V_NumberOfAfterDiploma);
                insADInst.Parameters.AddWithValue("@TypeOfAfterDiploma",
V_TypeOfAfterDiploma);
                insADInst.Parameters.AddWithValue("@NameOfEdInAfterDiploma",
V_NameOfEdInAfterDiploma);

insADInst.Parameters.AddWithValue("@YearOfAfterDiplomaGraduation",
V_YearOfAfterDiplomaGraduation);
                insADInst.Parameters.AddWithValue("@Degree", V_Degree);
                insADInst.Parameters.AddWithValue("@AcademicStatus",
V_AcademicStatus);
                insADInst.ExecuteNonQuery();
                LoadAD();
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show(ex.Message);
            }
            /*ADInstTable.Clear();
            ADInstAdapter.Fill(ADInstTable);
            ADInstitutes.DataSource = ADInstTable;  */
        }

        private void DeleteAfterDiplomaInst_Click(object sender, EventArgs e)
        {
            try
            {
                string DeleteADInstitute = "DELETE FROM [AfterDiploma] WHERE
[IdWorker] = @IdWorker AND [Diploma] = @Diploma";
                SqlCommand delADInst = new SqlCommand(DeleteADInstitute,
Program.Global.conn);
                delADInst.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
                delADInst.Parameters.AddWithValue("@Diploma",
SerieOfAfterDiploma.Text.ToString() + " " +
NumberOfAfterDiploma.Text.ToString());
                delADInst.ExecuteNonQuery();
                ADInstTable.Clear();
                ADInstAdapter.Fill(ADInstTable);
                ADInstitutes.DataSource = ADInstTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void AddFamily_Click(object sender, EventArgs e)
        {
            V_idWorker = idWorker.Text.ToString();
            V_FullnameOfFamily = FullnameOfFamily.Text.ToString();
            V_BirthYearOfFamily = BirthYearOfFamily.Text.ToString();
            V_TieOfFamily = TieOfFamily.Text.ToString();
            try
            {
                string InsertFamily = "INSERT INTO [FamilyStatus]
([IdWorker],[FullnameOfFamily],[BirthYearOfFamily],[TieOfFamily]) VALUES
(@IdWorker, @FullnameOfFamily, @BirthYearOfFamily, @TieOfFamily)";
                SqlCommand insFamily = new SqlCommand(InsertFamily,
Program.Global.conn);
                insFamily.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
                insFamily.Parameters.AddWithValue("@FullnameOfFamily",
FullnameOfFamily.Text.ToString());
                insFamily.Parameters.AddWithValue("@BirthYearOfFamily",
BirthYearOfFamily.Text.ToString());
                insFamily.Parameters.AddWithValue("@TieOfFamily",
TieOfFamily.Text.ToString());
                insFamily.ExecuteNonQuery();
                LoadFamily();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            /*FamilyTable.Clear();
            FamilyAdapter.Fill(FamilyTable);
            FamilyGrid.DataSource = FamilyTable;*/
        }

        private void DeleteFamily_Click(object sender, EventArgs e)
        {
            try
            {
```

```
                string DeleteFamily = "DELETE FROM [FamilyStatus] WHERE
[IdWorker] = @IdWorker AND [FullnameOfFamily] = @FullnameOfFamily AND
[BirthYearOfFamily] = @BirthYearOfFamily AND [TieOfFamily] = @TieOfFamily";
                SqlCommand delFamily = new SqlCommand(DeleteFamily,
Program.Global.conn);
                delFamily.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
                delFamily.Parameters.AddWithValue("@FullnameOfFamily",
FullnameOfFamily.Text.ToString());
                delFamily.Parameters.AddWithValue("@BirthYearOfFamily",
BirthYearOfFamily.Text.ToString());
                delFamily.Parameters.AddWithValue("@TieOfFamily",
TieOfFamily.Text.ToString());
                delFamily.ExecuteNonQuery();
                FamilyTable.Clear();
                FamilyAdapter.Fill(FamilyTable);
                FamilyGrid.DataSource = FamilyTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        private void CaseButton_Click(object sender, EventArgs e)
        {
            switch (Program.Global.Parameter)
            {
                case "View":
                    this.Close();
                    break;
                case "Edit":
                    V_idWorker = idWorker.Text.ToString();
                    V_SecondName = SecondName.Text.ToString();
                    V_FirstName = FirstName.Text.ToString();
                    V_ThirdName = ThirdName.Text.ToString();
                    V_Gender = Gender.SelectedItem.ToString();
                    V_Nationality = Nationality.Text.ToString();
                    V_IPN = IPN.Text.ToString();
                    V_EDRPOU = EDRPOU.Text.ToString();
                    V_MaritalStatus = MaritalStatus.SelectedItem.ToString();
                    V_NameOfPosition = NameOfPosition.Text.ToString();
                    V_SerieOfPassport = SerieOfPassport.Text.ToString();
                    V_NumberOfPassport = NumberOfPassport.Text.ToString();
                    V_GiverOfPassport = GiverOfPassport.Text.ToString();
                    V_RegAdress = RegAdress.Text.ToString();
                    V_FactAdress = FactAdress.Text.ToString();
                    V_LastPlaceOfWork = LastPlaceOfWork.Text.ToString();
                    V_PositionOnLastWork = PositionOnLastWork.Text.ToString();
                    V_GenExpDays = GenExpDays.Text.ToString();
                    V_GenExpMonths = GenExpMonths.Text.ToString();
                    V_GenExpYears = GenExpMonths.Text.ToString();
                    V_ExpDaysForPremium = ExpDaysForPremium.Text.ToString();
                    V_ExpMonthsForPremium = ExpMonthsForPremium.Text.ToString();
                    V_ExpYearsForPremium = ExpYearsForPremium.Text.ToString();
                    V_DateOfDismissal = DateOfDismissal.Value.ToString();
                    V_ReasonOfDismissal = ReasonOfDismissal.Text.ToString();
                    V_NameOfPosition = NameOfPosition.Text.ToString();
                    V_CodeOfPosition = CodeOfPosition.Text.ToString();
                    V_NameofStructuralUnit =
NameofStructuralUnit.Text.ToString();
                    V_TypeOfWork = TypeOfWork.Text.ToString();
                    V_Category = Category.Text.ToString();
                    V_ReasonOfAppointment = ReasonOfAppointment.Text.ToString();
                    V_NumberOfDecreeOfApp = NumberOfDecreeOfApp.Text.ToString();
                    V_ReasonOfDis = ReasonOfDis.Text.ToString();
                    V_NumberOfDecreeOfDis = NumberOfDecreeOfDis.Text.ToString();

                   try
                   {
```

```
                        MemoryStream ms = new MemoryStream();
                        Photo.Image.Save(ms, Photo.Image.RawFormat);
                        byte[] a = ms.GetBuffer();
                        ms.Close();
                        string UpdateWorker = "UPDATE [Worker] SET
[SecondName]=@SecondName,[FirstName]=@FirstName,[ThirdName]=@ThirdName,[Birthday
]=@Birthday,[Gender]=@Gender,[Nationality]=@Nationality,[IPN]=@IPN,[EDRPOU]=@EDR
POU,[Photo]=@Photo,[FamilyStatus]=@FamilyStatus,[CompletionDate]=@CompletionDate
, [Position]=@Position WHERE [IdWorker]=@IdWorker";
                        SqlCommand updWorker = new SqlCommand(UpdateWorker,
Program.Global.conn);
                        updWorker.Parameters.AddWithValue("@IdWorker",
V_idWorker);
                        updWorker.Parameters.AddWithValue("@SecondName",
V_SecondName);
                        updWorker.Parameters.AddWithValue("@FirstName",
V_FirstName);
                        updWorker.Parameters.AddWithValue("@ThirdName",
V_ThirdName);
                        updWorker.Parameters.AddWithValue("@Birthday",
Birthday.Value.ToString());
                        updWorker.Parameters.AddWithValue("@Gender", V_Gender);
                        updWorker.Parameters.AddWithValue("@Nationality",
V_Nationality);
                        updWorker.Parameters.AddWithValue("@IPN", V_IPN);
                        updWorker.Parameters.AddWithValue("@EDRPOU", V_EDRPOU);
                        updWorker.Parameters.AddWithValue("@Photo", a);
//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                        updWorker.Parameters.AddWithValue("@FamilyStatus",
V_MaritalStatus);
                        updWorker.Parameters.AddWithValue("@CompletionDate",
CompletionDate.Value.ToString());
                        updWorker.Parameters.AddWithValue("@Position",
V_NameOfPosition);
                        updWorker.ExecuteNonQuery();

//================================================================================
=======
                        string UpdatePassportInfo = "UPDATE [PassportInfo] SET
[SerieOfPassport]=@SerieOfPassport,[NumberOfPassport]=@NumberOfPassport,[DateOfP
assport]=@DateOfPassport,[GiverOfPassport]=@GiverOfPassport,[RegAdress]=@RegAdre
ss,[FactAdress]=@FactAdress WHERE [IdWorker]=@IdWorker";
                        SqlCommand updPassportInfo = new
SqlCommand(UpdatePassportInfo, Program.Global.conn);
                        updPassportInfo.Parameters.AddWithValue("@IdWorker",
V_idWorker);

updPassportInfo.Parameters.AddWithValue("@SerieOfPassport", V_SerieOfPassport);

updPassportInfo.Parameters.AddWithValue("@NumberOfPassport",
V_NumberOfPassport);

updPassportInfo.Parameters.AddWithValue("@DateOfPassport",
DateOfPassport.Value.ToString());

updPassportInfo.Parameters.AddWithValue("@GiverOfPassport", V_GiverOfPassport);
                        updPassportInfo.Parameters.AddWithValue("@RegAdress",
V_RegAdress);
                        updPassportInfo.Parameters.AddWithValue("@FactAdress",
V_FactAdress);
                        updPassportInfo.ExecuteNonQuery();

//================================================================================
=======
                        if (checkLastWork.Checked == true)
                        {
                            string UpdateLastWork = "UPDATE [LastWork] SET
[LastPlaceOfWork]=@LastPlaceOfWork,[PositionOnLastWork]=@PositionOnLastWork,[Gen
ExpDays]=@GenExpDays,[GenExpMonths]=@GenExpMonths,[GenExpYears]=@GenExpYears,[Ex
```

```
pDaysForPremium]=@ExpDaysForPremium,[ExpMonthsForPremium]=@ExpMonthsForPremium,[
ExpYearsForPremium]=@ExpYearsForPremium,[DateOfDismissal]=@DateOfDismissal,[Reas
onOfDismissal]=@ReasonOfDismissal WHERE [IdWorker]=@IdWorker";
                            SqlCommand updLastWork = new
SqlCommand(UpdateLastWork, Program.Global.conn);
                            updLastWork.Parameters.AddWithValue("@IdWorker",
V_idWorker);

updLastWork.Parameters.AddWithValue("@LastPlaceOfWork", V_LastPlaceOfWork);

updLastWork.Parameters.AddWithValue("@PositionOnLastWork",
V_PositionOnLastWork);
                            updLastWork.Parameters.AddWithValue("@GenExpDays",
V_GenExpDays);
                            updLastWork.Parameters.AddWithValue("@GenExpMonths",
V_GenExpMonths);
                            updLastWork.Parameters.AddWithValue("@GenExpYears",
V_GenExpMonths);

updLastWork.Parameters.AddWithValue("@ExpDaysForPremium", V_ExpDaysForPremium);

updLastWork.Parameters.AddWithValue("@ExpMonthsForPremium",
V_ExpMonthsForPremium);

updLastWork.Parameters.AddWithValue("@ExpYearsForPremium",
V_ExpYearsForPremium);

updLastWork.Parameters.AddWithValue("@DateOfDismissal", V_DateOfDismissal);

updLastWork.Parameters.AddWithValue("@ReasonOfDismissal", V_ReasonOfDismissal);
                            updLastWork.ExecuteNonQuery();
                        }

//=============================================================================
=======
                        string UpdatePositions = "UPDATE [Positions] SET
[NameOfPosition]=@NameOfPosition,[CodeOfPosition]=@CodeOfPosition,[NameofStructu
ralUnit]=@NameofStructuralUnit,[TypeOfWork]=@TypeOfWork,[Category]=@Category,[Da
teOfAppointment]=@DateOfAppointment,[ReasonOfAppointment]=@ReasonOfAppointment,[
NumberOfDecreeOfApp]=@NumberOfDecreeOfApp,[DateOfDis]=@DateOfDis,[ReasonOfDis]=@
ReasonOfDis, [NumberOfDecreeOfDis]=@NumberOfDecreeOfDis WHERE
[IdWorker]=@IdWorker";
                        SqlCommand updPositions = new
SqlCommand(UpdatePositions, Program.Global.conn);
                        updPositions.Parameters.AddWithValue("@IdWorker",
V_idWorker);
                        updPositions.Parameters.AddWithValue("@NameOfPosition",
V_NameOfPosition);
                        updPositions.Parameters.AddWithValue("@CodeOfPosition",
V_CodeOfPosition);

updPositions.Parameters.AddWithValue("@NameofStructuralUnit",
V_NameofStructuralUnit);
                        updPositions.Parameters.AddWithValue("@TypeOfWork",
V_TypeOfWork);
                        updPositions.Parameters.AddWithValue("@Category",
V_Category);

updPositions.Parameters.AddWithValue("@DateOfAppointment",
DateOfAppointment.Value.ToString());

updPositions.Parameters.AddWithValue("@ReasonOfAppointment",
V_ReasonOfAppointment);

updPositions.Parameters.AddWithValue("@NumberOfDecreeOfApp",
V_NumberOfDecreeOfApp);
                        updPositions.Parameters.AddWithValue("@DateOfDis",
DateOfDis.Value.ToString());
```

```
                        updPositions.Parameters.AddWithValue("@ReasonOfDis",
V_ReasonOfDis);

updPositions.Parameters.AddWithValue("@NumberOfDecreeOfDis",
V_NumberOfDecreeOfDis);
                        updPositions.ExecuteNonQuery();
                    }
                    catch(Exception ex)
                   {
                       MessageBox.Show(ex.Message);
                   }
                    break;
                case "Add":
                    V_idWorker = idWorker.Text.ToString();
                    V_SecondName = SecondName.Text.ToString();
                    V_FirstName = FirstName.Text.ToString();
                    V_ThirdName = ThirdName.Text.ToString();
                    V_Gender = Gender.SelectedItem.ToString();
                    V_Nationality = Nationality.Text.ToString();
                    V_IPN = IPN.Text.ToString();
                    V_EDRPOU = EDRPOU.Text.ToString();
                    V_MaritalStatus = MaritalStatus.SelectedItem.ToString();
                    V_NameOfPosition = NameOfPosition.Text.ToString();
                    V_SerieOfPassport = SerieOfPassport.Text.ToString();
                    V_NumberOfPassport = NumberOfPassport.Text.ToString();
                    V_GiverOfPassport = GiverOfPassport.Text.ToString();
                    V_RegAdress = RegAdress.Text.ToString();
                    V_FactAdress = FactAdress.Text.ToString();
                    V_LastPlaceOfWork = LastPlaceOfWork.Text.ToString();
                    V_PositionOnLastWork = PositionOnLastWork.Text.ToString();
                    V_GenExpDays = GenExpDays.Text.ToString();
                    V_GenExpMonths = GenExpMonths.Text.ToString();
                    V_GenExpYears = GenExpMonths.Text.ToString();
                    V_ExpDaysForPremium = ExpDaysForPremium.Text.ToString();
                    V_ExpMonthsForPremium = ExpMonthsForPremium.Text.ToString();
                    V_ExpYearsForPremium = ExpYearsForPremium.Text.ToString();
                    V_DateOfDismissal = DateOfDismissal.Value.ToString();
                    V_ReasonOfDismissal = ReasonOfDismissal.Text.ToString();
                    V_NameOfPosition = NameOfPosition.Text.ToString();
                    V_CodeOfPosition = CodeOfPosition.Text.ToString();
                    V_NameofStructuralUnit =
NameofStructuralUnit.Text.ToString();
                    V_TypeOfWork = TypeOfWork.Text.ToString();
                    V_Category = Category.Text.ToString();
                    V_ReasonOfAppointment = ReasonOfAppointment.Text.ToString();
                    V_NumberOfDecreeOfApp = NumberOfDecreeOfApp.Text.ToString();
                    V_ReasonOfDis = ReasonOfDis.Text.ToString();
                    V_NumberOfDecreeOfDis = NumberOfDecreeOfDis.Text.ToString();
                    try
                    {
                        MemoryStream ms2 = new MemoryStream();
                        Photo.Image.Save(ms2, Photo.Image.RawFormat);
                        byte[] b = ms2.GetBuffer();
                        ms2.Close();
                        string InsertWorker = "INSERT INTO [Worker]
([IdWorker],[SecondName],[FirstName],[ThirdName],[Birthday],[Gender],[Nationalit
y],[IPN],[EDRPOU],[Photo],[FamilyStatus],[CompletionDate],[Position]) VALUES
(@IdWorker,
@SecondName,@FirstName,@ThirdName,@Birthday,@Gender,@Nationality,@IPN,@EDRPOU,@P
hoto,@FamilyStatus,@CompletionDate,@Position)";
                        SqlCommand insWorker = new SqlCommand(InsertWorker,
Program.Global.conn);
                        insWorker.Parameters.AddWithValue("@IdWorker",
V_idWorker);
                        insWorker.Parameters.AddWithValue("@SecondName",
V_SecondName);
                        insWorker.Parameters.AddWithValue("@FirstName",
V_FirstName);
```

```
                        insWorker.Parameters.AddWithValue("@ThirdName",
V_ThirdName);
                        insWorker.Parameters.AddWithValue("@Birthday",
Birthday.Value.ToString());
                        insWorker.Parameters.AddWithValue("@Gender", V_Gender);
                        insWorker.Parameters.AddWithValue("@Nationality",
V_Nationality);
                        insWorker.Parameters.AddWithValue("@IPN", V_IPN);
                        insWorker.Parameters.AddWithValue("@EDRPOU", V_EDRPOU);
                        insWorker.Parameters.AddWithValue("@Photo", b);
                        insWorker.Parameters.AddWithValue("@FamilyStatus",
V_MaritalStatus);
                        insWorker.Parameters.AddWithValue("@CompletionDate",
CompletionDate.Value.ToString());
                        insWorker.Parameters.AddWithValue("@Position",
V_NameOfPosition);
                        insWorker.ExecuteNonQuery();

//=============================================================================
=======
                        string InsertPassportInfo = "INSERT INTO [PassportInfo]
([IdWorker],[SerieOfPassport],[NumberOfPassport],[DateOfPassport],[GiverOfPasspo
rt],[RegAdress],[FactAdress]) VALUES (@IdWorker,
@SerieOfPassport,@NumberOfPassport,@DateOfPassport,@GiverOfPassport,@RegAdress,@
FactAdress)";
                        SqlCommand insPassportInfo = new
SqlCommand(InsertPassportInfo, Program.Global.conn);
                        insPassportInfo.Parameters.AddWithValue("@IdWorker",
V_idWorker);

insPassportInfo.Parameters.AddWithValue("@SerieOfPassport", V_SerieOfPassport);

insPassportInfo.Parameters.AddWithValue("@NumberOfPassport",
V_NumberOfPassport);

insPassportInfo.Parameters.AddWithValue("@DateOfPassport",
DateOfPassport.Value.ToString());

insPassportInfo.Parameters.AddWithValue("@GiverOfPassport", V_GiverOfPassport);
                        insPassportInfo.Parameters.AddWithValue("@RegAdress",
V_RegAdress);
                        insPassportInfo.Parameters.AddWithValue("@FactAdress",
V_FactAdress);
                        insPassportInfo.ExecuteNonQuery();

//=============================================================================
=======
                        string InsertLastWork = "INSERT INTO [LastWork]
([IdWorker],[LastPlaceOfWork],[PositionOnLastWork],[GenExpDays],[GenExpMonths],[
GenExpYears],[ExpDaysForPremium],[ExpMonthsForPremium],[ExpYearsForPremium],[Dat
eOfDismissal],[ReasonOfDismissal]) VALUES (@IdWorker,
@LastPlaceOfWork,@PositionOnLastWork,@GenExpDays,@GenExpMonths,@GenExpYears,@Exp
DaysForPremium,@ExpMonthsForPremium,@ExpYearsForPremium,@DateOfDismissal,@Reason
OfDismissal)";
                        SqlCommand insLastWork = new SqlCommand(InsertLastWork,
Program.Global.conn);
                        insLastWork.Parameters.AddWithValue("@IdWorker",
V_idWorker);
                        insLastWork.Parameters.AddWithValue("@LastPlaceOfWork",
V_LastPlaceOfWork);

insLastWork.Parameters.AddWithValue("@PositionOnLastWork",
V_PositionOnLastWork);
                        insLastWork.Parameters.AddWithValue("@GenExpDays",
V_GenExpDays);
                        insLastWork.Parameters.AddWithValue("@GenExpMonths",
V_GenExpMonths);
                        insLastWork.Parameters.AddWithValue("@GenExpYears",
V_GenExpMonths);
```

```
insLastWork.Parameters.AddWithValue("@ExpDaysForPremium", V_ExpDaysForPremium);

insLastWork.Parameters.AddWithValue("@ExpMonthsForPremium",
V_ExpMonthsForPremium);

insLastWork.Parameters.AddWithValue("@ExpYearsForPremium",
V_ExpYearsForPremium);
                        insLastWork.Parameters.AddWithValue("@DateOfDismissal",
DateOfDismissal.Value.ToString());

insLastWork.Parameters.AddWithValue("@ReasonOfDismissal", V_ReasonOfDismissal);
                        insLastWork.ExecuteNonQuery();

//==========================================================================
=======
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                    break;

            }
        }

        private void TextOfApp_Click(object sender, EventArgs e)
        {
                        try
                        {
                            OpenFileDialog openfile = new OpenFileDialog();
                            openfile.Filter = "Text Files (*.doc; *.docx; *.xls;
*.xlsx; *.pdf) |*.doc; *.docx; *.xls; *.xlsx; *.pdf";
                            openfile.ShowDialog();
                            string filePath = openfile.FileName;
                            filenameOfApp = Path.GetFileName(filePath);

                            FileStream fs = new FileStream(filePath,
FileMode.Open);
                            BinaryReader br = new BinaryReader(fs);
                            byteOfApp = br.ReadBytes((Int32)fs.Length);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                        }

        }

        private void OpenTextOfApp_Click(object sender, EventArgs e)
        {
            try
            {
                byte[] bytes;
                string fileName;
                string strQuery = "select NameOfDocOfApp,DataOfDocOfApp from
Positions where IdWorker=@IdWorker and DateOfAppointment=@DateOfAppointment and
NumberOfDecreeOfApp=@NumberOfDecreeOfApp";
                SqlCommand cmd = new SqlCommand(strQuery, Program.Global.conn);
                cmd.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
                cmd.Parameters.AddWithValue("@DateOfAppointment",
DateOfAppointment.Text.ToString());
                cmd.Parameters.AddWithValue("@NameOfPosition",
NameOfPosition.Text.ToString());
                cmd.Parameters.AddWithValue("@NumberOfDecreeOfApp",
NumberOfDecreeOfApp.Text.ToString());
                using (SqlDataReader sdr = cmd.ExecuteReader())
                {
```

```
                    sdr.Read();
                    bytes = (byte[])sdr["DataOfDocOfApp"];
                    fileName = sdr["NameOfDocOfApp"].ToString();
                }
                string path = ParentDir + "\\Documents\\Decrees\\Appointment\\"
+ fileName.ToString();
                if (File.Exists(path))
                {
                    Process.Start(path);
                }
                else
                {
                    MessageBox.Show("Файл тексту наказу відсутній\n\nБудь ласка
збережіть файл");
                    SaveFileDialog savefile = new SaveFileDialog();
                    savefile.FileName = fileName;
                    savefile.InitialDirectory = ParentDir +
"\\Documents\\Decrees\\Appointment";
                    savefile.Filter = "Документ Word (*.doc) | *.doc |Документ
Word (*.docx) | *.docx | PDF File (*.pdf) | *.pdf";
                    savefile.ShowDialog();
                    BinaryWriter writer = new
BinaryWriter(File.Open(savefile.FileName, FileMode.Create));
                    writer.Write(bytes);
                    writer.Close();
                    Process.Start(path);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void UploadTextOfDis_Click(object sender, EventArgs e)
        {
                        try
                        {
                            OpenFileDialog openfile = new OpenFileDialog();
                            openfile.Filter = "Text Files (*.doc; *.docx; *.xls;
*.xlsx; *.pdf) |*.doc; *.docx; *.xls; *.xlsx; *.pdf";
                            openfile.ShowDialog();
                            string filePath = openfile.FileName;
                            filenameOfDis = Path.GetFileName(filePath);

                            FileStream fs = new FileStream(filePath,
FileMode.Open);
                            BinaryReader br = new BinaryReader(fs);
                            byteOfDis = br.ReadBytes((Int32)fs.Length);
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                        }
        }

        private void OpenTextOfDis_Click(object sender, EventArgs e)
        {
            try
            {
                byte[] bytes;
                string fileName;
                string strQuery = "select NameOfDocOfDis,DataOfDocOfDis from
Positions where IdWorker=@IdWorker and DateOfDis=@DateOfDis and
NumberOfDecreeOfDis=@NumberOfDecreeOfDis";
                SqlCommand cmd = new SqlCommand(strQuery, Program.Global.conn);
                cmd.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
```

```
                cmd.Parameters.AddWithValue("@DateOfDis",
DateOfDis.Text.ToString());
                cmd.Parameters.AddWithValue("@NumberOfDecreeOfDis",
NumberOfDecreeOfDis.Text.ToString());
                using (SqlDataReader sdr = cmd.ExecuteReader())
                {
                    sdr.Read();
                    bytes = (byte[])sdr["DataOfDocOfDis"];
                    fileName = sdr["NameOfDocOfDis"].ToString();
                }
                string path = ParentDir + "\\Documents\\Decrees\\Dismissal\\" +
fileName.ToString();
                if (File.Exists(path))
                {
                    Process.Start(path);
                }
                else
                {
                    MessageBox.Show("Файл тексту наказу відсутній\n\nБудь ласка
збережіть файл");
                    SaveFileDialog savefile = new SaveFileDialog();
                    savefile.FileName = fileName;
                    savefile.InitialDirectory = ParentDir +
"\\Documents\\Decrees\\Dismissal";
                    savefile.Filter = "Документ Word (*.doc) | *.doc |Документ
Word (*.docx) | *.docx | PDF File (*.pdf) | *.pdf";
                    savefile.ShowDialog();
                    BinaryWriter writer = new
BinaryWriter(File.Open(savefile.FileName, FileMode.Create));
                    writer.Write(bytes);
                    writer.Close();
                    Process.Start(path);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void CreateWordDocument(object filename, object savaAs)
        {
            object missing = Missing.Value;

            Word.Application wordApp = new Word.Application();

            Word.Document aDoc = null;

            if (File.Exists((string)filename))
            {
                DateTime today = DateTime.Now;

                object readOnly = false; //default
                object isVisible = false;

                wordApp.Visible = false;

                aDoc = wordApp.Documents.Open(ref filename, ref missing, ref
readOnly,
                                              ref missing, ref missing, ref
missing,
                                              ref missing, ref missing, ref
missing,
                                              ref missing, ref missing, ref
missing,
                                              ref missing, ref missing, ref
missing, ref missing);

                aDoc.Activate();
```

```
                //Find and replace:
                string idWorker =
OverviewSet.Tables[0].Rows[0]["IdWorker"].ToString();
                string SecondName =
OverviewSet.Tables[0].Rows[0]["SecondName"].ToString();
                string FirstName =
OverviewSet.Tables[0].Rows[0]["FirstName"].ToString();
                string ThirdName =
OverviewSet.Tables[0].Rows[0]["ThirdName"].ToString();
                string[] Birthday =
OverviewSet.Tables[0].Rows[0]["Birthday"].ToString().Split(' ');
                string Gender =
OverviewSet.Tables[0].Rows[0]["Gender"].ToString();
                string Nationality =
OverviewSet.Tables[0].Rows[0]["Nationality"].ToString();
                string IPN = OverviewSet.Tables[0].Rows[0]["IPN"].ToString();
                string EDRPOU =
OverviewSet.Tables[0].Rows[0]["EDRPOU"].ToString();
                string MaritalStatus =
OverviewSet.Tables[0].Rows[0]["FamilyStatus"].ToString();
                string[] CompletionDate =
OverviewSet.Tables[0].Rows[0]["CompletionDate"].ToString().Split(' ');

//=============================================================================
=============
                string SerieOfPassport =
PassportInfoSet.Tables[0].Rows[0]["SerieOfPassport"].ToString();
                string NumberOfPassport =
PassportInfoSet.Tables[0].Rows[0]["NumberOfPassport"].ToString();
                string[] DateOfPassport =
PassportInfoSet.Tables[0].Rows[0]["DateOfPassport"].ToString().Split(' ');
                string GiverOfPassport =
PassportInfoSet.Tables[0].Rows[0]["GiverOfPassport"].ToString();
                string RegAdress =
PassportInfoSet.Tables[0].Rows[0]["RegAdress"].ToString();
                string FactAdress =
PassportInfoSet.Tables[0].Rows[0]["FactAdress"].ToString();

//=============================================================================
=============
                string LastPlaceOfWork =
LastWorkSet.Tables[0].Rows[0]["LastPlaceOfWork"].ToString();
                string PositionOnLastWork =
LastWorkSet.Tables[0].Rows[0]["PositionOnLastWork"].ToString();
                string GenExpDays =
LastWorkSet.Tables[0].Rows[0]["GenExpDays"].ToString();
                string GenExpMonths =
LastWorkSet.Tables[0].Rows[0]["GenExpMonths"].ToString();
                string GenExpYears =
LastWorkSet.Tables[0].Rows[0]["GenExpYears"].ToString();
                string ExpDaysForPremium =
LastWorkSet.Tables[0].Rows[0]["ExpDaysForPremium"].ToString();
                string ExpMonthsForPremium =
LastWorkSet.Tables[0].Rows[0]["ExpMonthsForPremium"].ToString();
                string ExpYearsForPremium =
LastWorkSet.Tables[0].Rows[0]["ExpYearsForPremium"].ToString();
                string[] DateOfDismissal =
LastWorkSet.Tables[0].Rows[0]["DateOfDismissal"].ToString().Split(' ');
                string ReasonOfDismissal =
LastWorkSet.Tables[0].Rows[0]["ReasonOfDismissal"].ToString();

//=============================================================================
=============
                string[] Ed_level = new string[4];
                string[] NameOfEdInstitution = new string[4];
                string[] YearOfGraduation = new string[4];
                string[] Speciality = new string[4];
                string[] Qualification = new string[4];
```

```
                string[] FormOfEducation = new string[4];
                string[] SerieAndNumber = new string[4];

//=============================================================================
=============
                string[] TypeOfAfterDiploma = new string[3];
                string[] NameOfEdInAfterDiploma = new string[3];
                string[] YearOfAfterDiplomaGraduation = new string[3];
                string[] Degree = new string[3];
                string[] AcademicStatus = new string[3];
                string[] ADDiploma = new string[3];

//=============================================================================
=============
                string[] FullnameOfFamily = new string[4];
                string[] BirthYearOfFamily = new string[4];
                string[] TieOfFamily = new string[4];

//=============================================================================
=============
                for (int index = 0; index < EdInstTable.Rows.Count; index++)
                {
                    Ed_level[index] =
EdInstTable.Rows[index]["Ed_level"].ToString();
                    NameOfEdInstitution[index] =
EdInstTable.Rows[index]["NameOfEdInstitution"].ToString();
                    YearOfGraduation[index] =
EdInstTable.Rows[index]["YearOfGraduation"].ToString();
                    Speciality[index] =
EdInstTable.Rows[index]["Speciality"].ToString();
                    Qualification[index] =
EdInstTable.Rows[index]["Qualification"].ToString();
                    FormOfEducation[index] =
EdInstTable.Rows[index]["FormOfEducation"].ToString();
                    SerieAndNumber[index] =
EdInstTable.Rows[index]["Diploma"].ToString();
                }

//=============================================================================
=============
                for (int index = 0; index < ADInstTable.Rows.Count; index++)
                {
                    TypeOfAfterDiploma[index] =
ADInstTable.Rows[index]["TypeOfAfterDiploma"].ToString();
                    NameOfEdInAfterDiploma[index] =
ADInstTable.Rows[index]["NameOfEdInAfterDiploma"].ToString();
                    YearOfAfterDiplomaGraduation[index] =
ADInstTable.Rows[index]["YearOfAfterDiplomaGraduation"].ToString();
                    Degree[index] =
ADInstTable.Rows[index]["Degree"].ToString();
                    AcademicStatus[index] =
ADInstTable.Rows[index]["AcademicStatus"].ToString();
                    ADDiploma[index] =
ADInstTable.Rows[index]["Diploma"].ToString();
                }

//=============================================================================
=============
                for (int index = 0; index < FamilyTable.Rows.Count; index++)
                {
                    FullnameOfFamily[index] =
FamilyTable.Rows[index]["FullnameOfFamily"].ToString();
                    BirthYearOfFamily[index] =
FamilyTable.Rows[index]["BirthYearOfFamily"].ToString();
                    TieOfFamily[index] =
FamilyTable.Rows[index]["TieOfFamily"].ToString();
                }
```

```
/*//=============================================================================
===============
                NameOfPosition.Text =
PositionsTable.Rows[index]["NameOfPosition"].ToString();
                CodeOfPosition.Text =
PositionsTable.Rows[index]["CodeOfPosition"].ToString();
                NameofStructuralUnit.Text =
PositionsTable.Rows[index]["NameofStructuralUnit"].ToString();
                TypeOfWork.Text =
PositionsTable.Rows[index]["TypeOfWork"].ToString();
                Category.Text =
PositionsTable.Rows[index]["Category"].ToString();
                DateOfAppointment.Text =
PositionsTable.Rows[index]["DateOfAppointment"].ToString();
                ReasonOfAppointment.Text =
PositionsTable.Rows[index]["ReasonOfAppointment"].ToString();
                NumberOfDecreeOfApp.Text =
PositionsTable.Rows[index]["NumberOfDecreeOfApp"].ToString();
                DateOfDis.Text =
PositionsTable.Rows[index]["DateOfDis"].ToString();
                ReasonOfDis.Text =
PositionsTable.Rows[index]["ReasonOfDis"].ToString();
                NumberOfDecreeOfDis.Text =
PositionsTable.Rows[index]["NumberOfDecreeOfDis"].ToString();

//=============================================================================
=============
                TypeOfVacation.Text =
VacationsTable.Rows[index]["TypeOfVacation"].ToString();
                Period.Text = VacationsTable.Rows[index]["Period"].ToString();
                BeginOfVacation.Text =
VacationsTable.Rows[index]["BeginOfVacation"].ToString();
                EndOfVacation.Text =
VacationsTable.Rows[index]["EndOfVacation"].ToString();
                ReasonOfVac.Text =
VacationsTable.Rows[index]["ReasonOfVac"].ToString();
                NumberOfDecreeOfVac.Text =
VacationsTable.Rows[index]["NumberOfDecreeOfVac"].ToString();

//=============================================================================
============= */

                this.FindAndReplace(wordApp, "<CompDate>", CompletionDate[0]);
                this.FindAndReplace(wordApp, "<idWorker>", idWorker);
                this.FindAndReplace(wordApp, "<IPN>", IPN);
                this.FindAndReplace(wordApp, "<Gender>", Gender);
                //this.FindAndReplace(wordApp, "<TypeOfWork>", typeofwork);
                this.FindAndReplace(wordApp, "<EDRPOU>", EDRPOU);
                this.FindAndReplace(wordApp, "<SecondName>", SecondName);
                this.FindAndReplace(wordApp, "<FirstName>", FirstName);
                this.FindAndReplace(wordApp, "<ThirdName>", ThirdName);
                this.FindAndReplace(wordApp, "<Birthday>", Birthday[0]);
                this.FindAndReplace(wordApp, "<Nationality>", Nationality);
                //this.FindAndReplace(wordApp, "<Ed_level>", edlevel);
                this.FindAndReplace(wordApp, "<MaritalStatus>", MaritalStatus);
                this.FindAndReplace(wordApp, "<FactAdress>", FactAdress);
                this.FindAndReplace(wordApp, "<RegAdress>", RegAdress);
                this.FindAndReplace(wordApp, "<SOP>", SerieOfPassport);
                this.FindAndReplace(wordApp, "<NOP>", NumberOfPassport);
                this.FindAndReplace(wordApp, "<GiverOfPassport>",
GiverOfPassport);
                this.FindAndReplace(wordApp, "<DateOfPassport>",
DateOfPassport[0]);
                this.FindAndReplace(wordApp, "<LastPlaceOfWork>",
LastPlaceOfWork);
                this.FindAndReplace(wordApp, "<PositionOnLastWork>",
PositionOnLastWork);
                this.FindAndReplace(wordApp, "<GED>", GenExpDays);
```

```
                this.FindAndReplace(wordApp, "<GEM>", GenExpMonths);
                this.FindAndReplace(wordApp, "<GEY>", GenExpYears);
                this.FindAndReplace(wordApp, "<EDFP>", ExpDaysForPremium);
                this.FindAndReplace(wordApp, "<EMFP>", ExpMonthsForPremium);
                this.FindAndReplace(wordApp, "<EYFP>", ExpYearsForPremium);
                this.FindAndReplace(wordApp, "<DateOfDismissal>",
DateOfDismissal[0]);
                this.FindAndReplace(wordApp, "<ReasonOfDismissal>",
ReasonOfDismissal);
                for (int i = 0; i < 4; i++)
                {
                    this.FindAndReplace(wordApp, "<Ed_level>" + i, Ed_level[i]);
                    this.FindAndReplace(wordApp, "<NameOfEdInstitution>" + i,
NameOfEdInstitution[i]);
                    this.FindAndReplace(wordApp, "<Diploma>" + i,
SerieAndNumber[i]);
                    this.FindAndReplace(wordApp, "<YearOfGrad>" + i,
YearOfGraduation[i]);
                    this.FindAndReplace(wordApp, "<Speciality>" + i,
Speciality[i]);
                    this.FindAndReplace(wordApp, "<Qualification>" + i,
Qualification[i]);
                    this.FindAndReplace(wordApp, "<FormOfEd>" + i,
FormOfEducation[i]);
                }
                for (int i = 0; i < 3; i++)
                {
                    this.FindAndReplace(wordApp, "<TypeOfAD>" + i,
TypeOfAfterDiploma[i]);
                    this.FindAndReplace(wordApp, "<NameOfEdInAfterDiploma>" + i,
NameOfEdInAfterDiploma[i]);
                    this.FindAndReplace(wordApp, "<YOADG>" + i,
YearOfAfterDiplomaGraduation[i]);
                    this.FindAndReplace(wordApp, "<Degree>" + i, Degree[i]);
                    this.FindAndReplace(wordApp, "<AS>" + i, AcademicStatus[i]);
                    this.FindAndReplace(wordApp, "<AD>" + i, ADDiploma[i]);
                }
                for (int i = 0; i < 4; i++)
                {
                    this.FindAndReplace(wordApp, "<FullnameOfFamily>" + i,
FullnameOfFamily[i]);
                    this.FindAndReplace(wordApp, "<BirthYearOfFamily>" + i,
BirthYearOfFamily[i]);
                    this.FindAndReplace(wordApp, "<TieOfFamily>" + i,
TieOfFamily[i]);
                }

                //вставити зображення:
                string imageName = "Photo.jpg";
                string temppath = ParentDir + "\\Reports" + imageName;
                var newImage = ScalePhoto(Photo.Image, 135, 180);
                newImage.Save(temppath,
System.Drawing.Imaging.ImageFormat.Jpeg);
                Object oLinkToFile = false;  //за замовчуванням
                Object oSaveWithDocument = true;// за замовчуванням
                Object left = 436;
                Object top = 10;
                Object width = Type.Missing;
                Object height =Type.Missing;
                Object range = Type.Missing;
                aDoc.Shapes.AddPicture(temppath, ref oLinkToFile, ref
oSaveWithDocument, ref left, ref top, ref width, ref height, ref range);
                File.Delete(temppath);
            }
            else
            {
                MessageBox.Show("Файл не знайдено");
                return;
            }
```

```
            //Зберегти як...
            aDoc.SaveAs2(ref savaAs, ref missing, ref missing, ref missing,
                    ref missing, ref missing, ref missing,
                    ref missing, ref missing, ref missing,
                    ref missing, ref missing, ref missing,
                    ref missing, ref missing, ref missing);

            //Закрити документ
            aDoc.Close(ref missing, ref missing, ref missing);
            //MessageBox.Show("Звіт створений");
        }
        private void Print_Click(object sender, EventArgs e)
        {
            filename = ParentDir + "\\Reports\\Templates\\" +
"Personal_card.doc";
            string SaveFileName = idWorker.Text.Replace(" ",string.Empty) + "_"
+ SecondName.Text.ToString() + "_" + FirstName.Text.ToString() + ".doc";
            string InitDir = ParentDir + "\\Reports\\"+SaveFileName;
            CreateWordDocument(filename, InitDir);
            Process.Start(InitDir);
        }

        private void AddPosition_Click(object sender, EventArgs e)
        {
            V_idWorker = idWorker.Text.ToString();
            V_NameOfPosition = NameOfPosition.Text.ToString();
            V_CodeOfPosition = CodeOfPosition.Text.ToString();
            V_NameofStructuralUnit = NameofStructuralUnit.Text.ToString();
            V_TypeOfWork = TypeOfWork.SelectedItem.ToString();
            V_Category = Category.Text.ToString();
            V_ReasonOfAppointment = ReasonOfAppointment.Text.ToString();
            V_NumberOfDecreeOfApp = NumberOfDecreeOfApp.Text.ToString();
            V_ReasonOfDis = ReasonOfDis.Text.ToString();
            V_NumberOfDecreeOfDis = NumberOfDecreeOfDis.Text.ToString();
            if (Dismissed.Checked == true)
            {
                try
                {
                    string InsertPositions = "INSERT INTO [Positions]
([IdWorker],[NameOfPosition],[CodeOfPosition],[NameofStructuralUnit],[TypeOfWork
],[Category],[DateOfAppointment],[ReasonOfAppointment],[NumberOfDecreeOfApp],[Da
teOfDis],ReasonOfDis],
[NumberOfDecreeOfDis],[NameOfDocOfApp],[DataOfDocOfApp],[NameOfDocOfDis],[DataOf
DocOfDis]) VALUES (@IdWorker,
@NameOfPosition,@CodeOfPosition,@NameofStructuralUnit,@TypeOfWork,@Category,@Dat
eOfAppointment,@ReasonOfAppointment,@NumberOfDecreeOfApp,@DateOfDis,@ReasonOfDis
,
@NumberOfDecreeOfDis,@NameOfDocOfApp,@DataOfDocOfApp,@NameOfDocOfDis,@DataOfDocO
fDis)";
                    SqlCommand insPositions = new SqlCommand(InsertPositions,
Program.Global.conn);
                    insPositions.Parameters.AddWithValue("@IdWorker",
V_idWorker);
                    insPositions.Parameters.AddWithValue("@NameOfPosition",
V_NameOfPosition);
                    insPositions.Parameters.AddWithValue("@CodeOfPosition",
V_CodeOfPosition);

insPositions.Parameters.AddWithValue("@NameofStructuralUnit",
V_NameofStructuralUnit);
                    insPositions.Parameters.AddWithValue("@TypeOfWork",
V_TypeOfWork);
                    insPositions.Parameters.AddWithValue("@Category",
V_Category);
                    insPositions.Parameters.AddWithValue("@DateOfAppointment",
DateOfAppointment.Value.ToString());
                    insPositions.Parameters.AddWithValue("@ReasonOfAppointment",
V_ReasonOfAppointment);
```

```
                    insPositions.Parameters.AddWithValue("@NumberOfDecreeOfApp",
V_NumberOfDecreeOfApp);
                    insPositions.Parameters.AddWithValue("@DateOfDis",
DateOfDis.Value.ToString());
                    insPositions.Parameters.AddWithValue("@ReasonOfDis",
V_ReasonOfDis);
                    insPositions.Parameters.AddWithValue("@NumberOfDecreeOfDis",
V_NumberOfDecreeOfDis);
                    insPositions.Parameters.AddWithValue("@NameOfDocOfApp",
filenameOfApp);
                    insPositions.Parameters.AddWithValue("@DataOfDocOfApp",
byteOfApp);
                    insPositions.Parameters.AddWithValue("@NameOfDocOfDis",
filenameOfDis);
                    insPositions.Parameters.AddWithValue("@DataOfDocOfDis",
byteOfDis);
                    insPositions.ExecuteNonQuery();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
            else
            {
                string InsertPositions = "INSERT INTO [Positions]
([IdWorker],[NameOfPosition],[CodeOfPosition],[NameofStructuralUnit],[TypeOfWork
],[Category],[DateOfAppointment],[ReasonOfAppointment],[NumberOfDecreeOfApp],[Na
meOfDocOfApp],[DataOfDocOfApp]) VALUES (@IdWorker,
@NameOfPosition,@CodeOfPosition,@NameofStructuralUnit,@TypeOfWork,@Category,@Dat
eOfAppointment,@ReasonOfAppointment,@NumberOfDecreeOfApp,@NameOfDocOfApp,@DataOf
DocOfApp)";
                SqlCommand insPositions = new SqlCommand(InsertPositions,
Program.Global.conn);
                insPositions.Parameters.AddWithValue("@IdWorker", V_idWorker);
                insPositions.Parameters.AddWithValue("@NameOfPosition",
V_NameOfPosition);
                insPositions.Parameters.AddWithValue("@CodeOfPosition",
V_CodeOfPosition);
                insPositions.Parameters.AddWithValue("@NameofStructuralUnit",
V_NameofStructuralUnit);
                insPositions.Parameters.AddWithValue("@TypeOfWork",
V_TypeOfWork);
                insPositions.Parameters.AddWithValue("@Category", V_Category);
                insPositions.Parameters.AddWithValue("@DateOfAppointment",
DateOfAppointment.Value.ToString());
                insPositions.Parameters.AddWithValue("@ReasonOfAppointment",
V_ReasonOfAppointment);
                insPositions.Parameters.AddWithValue("@NumberOfDecreeOfApp",
V_NumberOfDecreeOfApp);
                insPositions.Parameters.AddWithValue("@NameOfDocOfApp",
filenameOfApp);
                insPositions.Parameters.AddWithValue("@DataOfDocOfApp",
byteOfApp);
                insPositions.ExecuteNonQuery();
            }
            LoadPositions();
        }

        private void DeletePosition_Click(object sender, EventArgs e)
        {
            try
            {
                string DeletePosition = "DELETE FROM [Positions] WHERE
[IdWorker]=@IdWorker AND [NameOfPosition]=@NameOfPosition
AND[CodeOfPosition]=@CodeOfPosition AND
[NameofStructuralUnit]=@NameofStructuralUnit AND [TypeOfWork]=@TypeOfWork AND
[Category]=@Category AND [DateOfAppointment]=@DateOfAppointment AND
[ReasonOfAppointment]=@ReasonOfAppointment AND
```

```
[NumberOfDecreeOfApp]=@NumberOfDecreeOfApp AND [DateOfDis]=@DateOfDis AND
[ReasonOfDis]=@ReasonOfDis AND [NumberOfDecreeOfDis]=@NumberOfDecreeOfDis";
                SqlCommand delPosition = new SqlCommand(DeletePosition,
Program.Global.conn);
                delPosition.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
                delPosition.Parameters.AddWithValue("@NameOfPosition",
NameOfPosition.Text.ToString());
                delPosition.Parameters.AddWithValue("@CodeOfPosition",
CodeOfPosition.Text.ToString());
                delPosition.Parameters.AddWithValue("@NameofStructuralUnit",
NameofStructuralUnit.Text.ToString());
                delPosition.Parameters.AddWithValue("@TypeOfWork",
TypeOfWork.SelectedItem.ToString());
                delPosition.Parameters.AddWithValue("@Category",
Category.Text.ToString());
                delPosition.Parameters.AddWithValue("@DateOfAppointment",
DateOfAppointment.Value.ToString());
                delPosition.Parameters.AddWithValue("@ReasonOfAppointment",
ReasonOfAppointment.Text.ToString());
                delPosition.Parameters.AddWithValue("@NumberOfDecreeOfApp",
NumberOfDecreeOfApp.Text.ToString());
                delPosition.Parameters.AddWithValue("@DateOfDis",
DateOfDis.Value.ToString());
                delPosition.Parameters.AddWithValue("@ReasonOfDis",
ReasonOfDis.Text.ToString());
                delPosition.Parameters.AddWithValue("@NumberOfDecreeOfDis",
NumberOfDecreeOfDis.Text.ToString());
                delPosition.ExecuteNonQuery();
                PositionsTable.Clear();
                PositionsAdapter.Fill(PositionsTable);
                PositionsGrid.DataSource = PositionsTable;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void OpenTextOfVac_Click(object sender, EventArgs e)
        {
            byte[] bytes;
            string fileName;
            string strQuery = "select NameOfDocOfVac,DataOfDocOfVac from
Vacations where IdWorker=@IdWorker and
NumberOfDecreeOfVac=@NumberOfDecreeOfVac";
            SqlCommand cmd = new SqlCommand(strQuery, Program.Global.conn);
            cmd.Parameters.AddWithValue("@IdWorker", idWorker.Text.ToString());
            cmd.Parameters.AddWithValue("@NumberOfDecreeOfVac",
NumberOfDecreeOfVac.Text.ToString());
            using (SqlDataReader sdr = cmd.ExecuteReader())
            {
                sdr.Read();
                bytes = (byte[])sdr["DataOfDocOfVac"];
                fileName = sdr["NameOfDocOfVac"].ToString();
            }
            string path = ParentDir + "\\Documents\\Decrees\\Vacation" +
fileName.ToString();
            if (File.Exists(path))
            {
                Process.Start(path);
            }
            else
            {
                MessageBox.Show("Файл тексту наказу відсутній\n\nБудь ласка
збережіть файл");
                SaveFileDialog savefile = new SaveFileDialog();
                savefile.FileName = fileName;
```

```
                savefile.InitialDirectory = ParentDir +
"\\Documents\\Decrees\\Vacation";
                savefile.Filter = "Документ Word (*.doc) | *.doc |Документ Word
(*.docx) | *.docx | PDF File (*.pdf) | *.pdf";
                savefile.ShowDialog();
                BinaryWriter writer = new
BinaryWriter(File.Open(savefile.FileName, FileMode.Create));
                writer.Write(bytes);
                writer.Close();
                Process.Start(path);
            }
        }

        private void UploadTextOfVac_Click(object sender, EventArgs e)
        {
            try
            {
                OpenFileDialog openfile = new OpenFileDialog();
                openfile.Filter = "Text Files (*.doc; *.docx; *.xls; *.xlsx;
*.pdf) |*.doc; *.docx; *.xls; *.xlsx; *.pdf";
                openfile.ShowDialog();
                string filePath = openfile.FileName;
                filenameOfVac = Path.GetFileName(filePath);

                FileStream fs = new FileStream(filePath, FileMode.Open);
                BinaryReader br = new BinaryReader(fs);
                byteOfVac = br.ReadBytes((Int32)fs.Length);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        private void AddVacation_Click(object sender, EventArgs e)
        {
            V_idWorker = idWorker.Text.ToString();
            V_TypeOfVacation = TypeOfVacation.SelectedItem.ToString();
            V_Period = Period.Text.ToString();
            V_ReasonOfVac = ReasonOfVac.Text.ToString();
            V_NumberOfDecreeOfVac = NumberOfDecreeOfVac.Text.ToString();
            try
            {
                string InsertVacation = "INSERT INTO [Vacations]
([IdWorker],[TypeOfVacation],[Period],[BeginOfVacation],[EndOfVacation],[ReasonO
fVac],[NumberOfDecreeOfVac],[NameOfDocOfVac],[DataOfDocOfVac]) VALUES
(@IdWorker,@TypeOfVacation,Period,@BeginOfVacation,@EndOfVacation,@ReasonOfVac,@
NumberOfDecreeOfVac,@NameOfDocOfVac,@DataOfDocOfVac)";
                SqlCommand insVacation = new SqlCommand(InsertVacation,
Program.Global.conn);
                insVacation.Parameters.AddWithValue("@IdWorker", V_idWorker);
                insVacation.Parameters.AddWithValue("@TypeOfVacation",
V_TypeOfVacation);
                insVacation.Parameters.AddWithValue("@Period", V_Period);
                insVacation.Parameters.AddWithValue("@BeginOfVacation",
BeginOfVacation.Value.ToString());
                insVacation.Parameters.AddWithValue("@EndOfVacation",
EndOfVacation.Value.ToString());
                insVacation.Parameters.AddWithValue("@ReasonOfVac",
V_ReasonOfVac);
                insVacation.Parameters.AddWithValue("@NumberOfDecreeOfVac",
V_NumberOfDecreeOfVac);
                insVacation.Parameters.AddWithValue("@NameOfDocOfVac",
filenameOfVac);
                insVacation.Parameters.AddWithValue("@DataOfDocOfVac",
byteOfVac);
                insVacation.ExecuteNonQuery();
                LoadVacations();
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void DeleteVacation_Click(object sender, EventArgs e)
        {
            string DeleteVacation = "DELETE FROM [Vacations] WHERE
IdWorker=@IdWorker and NumberOfDecreeOfVac=@NumberOfDecreeOfVac";
            SqlCommand delVacation = new SqlCommand(DeleteVacation,
Program.Global.conn);
            delVacation.Parameters.AddWithValue("@IdWorker",
idWorker.Text.ToString());
            delVacation.Parameters.AddWithValue("@NumberOfDecreeOfVac",
NumberOfDecreeOfVac.Text.ToString());
            delVacation.ExecuteNonQuery();
            VacationsTable.Clear();
            VacationsAdapter.Fill(VacationsTable);
            VacationsGrid.DataSource = VacationsTable;
        }

        private void Ed_level_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (Ed_level.SelectedIndex.ToString() == "0" ||
Ed_level.SelectedIndex.ToString() == "1")
            {
                Speciality.ReadOnly = true;
                Qualification.ReadOnly = true;
                FormOfEducation.Enabled = false;
            }
            else
            {
                Speciality.ReadOnly = false;
                Qualification.ReadOnly = false;
                FormOfEducation.Enabled = true;
            }
        }


    }
}
```

// **Workers_Form.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Threading;
using Microsoft.Office.Interop.Word;
using Microsoft.Office.Core;
using Word = Microsoft.Office.Interop.Word;
using System.Reflection;

namespace OK
{
    public partial class Workers_Form : Form
    {
        //Метод пошуку та заміни
        private void FindAndReplace(Microsoft.Office.Interop.Word.Application
wordApp, object findText, object replaceWithText)
        {
            object matchCase = true;
            object matchWholeWord = true;
            object matchWildCards = false;
            object matchSoundLike = false;
            object nmatchAllForms = false;
            object forward = true;
            object format = false;
            object matchKashida = false;
            object matchDiactitics = false;
            object matchAlefHamza = false;
            object matchControl = false;
            object read_only = false;
            object visible = true;
            object replace = 2;
            object wrap = 1;

            wordApp.Selection.Find.Execute(ref findText,
                       ref matchCase, ref matchWholeWord,
                       ref matchWildCards, ref matchSoundLike,
                       ref nmatchAllForms, ref forward,
                       ref wrap, ref format, ref replaceWithText,
                       ref replace, ref matchKashida,
                       ref matchDiactitics, ref matchAlefHamza,
                       ref matchControl);
        }

//=====================================================================
        public Workers_Form()
        {
            InitializeComponent();
        }
        Worker_Edit workered = new Worker_Edit();
        SqlDataAdapter WorkerAdapter;
        System.Data.DataTable WorkerTable = new System.Data.DataTable();
        DataView WorkersView;


        private void View_Click(object sender, EventArgs e)
        {
            Program.Global.Parameter = "View";
            //Program.Global.ChosenWorker =
WorkersGrid.Rows[WorkersGrid.CurrentRow.Index].Cells[1].Value.ToString();
            workered.ShowDialog();
```

```
        }

        private void Workers_Form_Load(object sender, EventArgs e)
        {
            string ToWorkersGrid = "SELECT
[IdWorker],[SecondName],[FirstName],[ThirdName], [Position] FROM [Worker]";
            WorkerAdapter = new SqlDataAdapter(ToWorkersGrid,
Program.Global.conn);
            WorkerAdapter.Fill(WorkerTable);
            WorkersGrid.Columns[0].DataPropertyName = "IdWorker";
            WorkersGrid.Columns[1].DataPropertyName = "SecondName";
            WorkersGrid.Columns[2].DataPropertyName = "FirstName";
            WorkersGrid.Columns[3].DataPropertyName = "ThirdName";
            WorkersGrid.Columns[4].DataPropertyName = "Position";
            WorkersView = new DataView(WorkerTable);
            WorkersGrid.DataSource = WorkersView;
            this.WorkersGrid.RowPostPaint += new
System.Windows.Forms.DataGridViewRowPostPaintEventHandler(this.WorkersGrid_RowPo
stPaint);
            if (WorkersGrid.Rows.Count != 0)
            {
                Program.Global.ChosenWorker =
WorkersGrid.Rows[WorkersGrid.CurrentRow.Index].Cells[1].Value.ToString();
            }
        }
        //=====================================================================
        //Номер запису в таблиці
        //=====================================================================
        private void WorkersGrid_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
        {
            using (SolidBrush b = new
SolidBrush(WorkersGrid.RowHeadersDefaultCellStyle.ForeColor))
            {
                e.Graphics.DrawString((e.RowIndex + 1).ToString(),
e.InheritedRowStyle.Font, b, e.RowBounds.Location.X + 10, e.RowBounds.Location.Y
+ 5);
            }
        }

        private void Add_Click(object sender, EventArgs e)
        {
            Program.Global.Parameter = "Add";
            workered.ShowDialog();
        }

        private void Edit_Click(object sender, EventArgs e)
        {
            Program.Global.Parameter = "Edit";
            //Program.Global.ChosenWorker =
WorkersGrid.Rows[WorkersGrid.CurrentRow.Index].Cells[1].Value.ToString();
            workered.ShowDialog();
        }

        private void Delete_Click(object sender, EventArgs e)
        {

//=====================================================================
            //Видалити з Worker

//=====================================================================
            string DeleteWorker = "DELETE FROM [Worker] WHERE [IdWorker] =
@IdWorker";
            SqlCommand delWor = new SqlCommand(DeleteWorker,
Program.Global.conn);
            delWor.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delWor.ExecuteNonQuery();
```

```
//=========================================================================
            //Видалити з PassportInfo

//=========================================================================
            string DeletePassportInfo = "DELETE FROM [PassportInfo] WHERE
[IdWorker] = @IdWorker";
            SqlCommand delPI = new SqlCommand(DeletePassportInfo,
Program.Global.conn);
            delPI.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delPI.ExecuteNonQuery();

//=========================================================================
            //Видалити з LastWork

//=========================================================================
            string DeleteLastWork = "DELETE FROM [LastWork] WHERE [IdWorker] =
@IdWorker";
            SqlCommand delLW = new SqlCommand(DeleteLastWork,
Program.Global.conn);
            delLW.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delLW.ExecuteNonQuery();

//=========================================================================
            //Видалити з FamilyStatus

//=========================================================================
            string DeleteFamilyStatus = "DELETE FROM [FamilyStatus] WHERE
[IdWorker] = @IdWorker";
            SqlCommand delFS = new SqlCommand(DeleteFamilyStatus,
Program.Global.conn);
            delFS.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delFS.ExecuteNonQuery();

//=========================================================================
            //Видалити з Education

//=========================================================================
            string DeleteEducation = "DELETE FROM [Education] WHERE [IdWorker] =
@IdWorker";
            SqlCommand delEd = new SqlCommand(DeleteEducation,
Program.Global.conn);
            delEd.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delEd.ExecuteNonQuery();

//=========================================================================
            //Видалити з AfterDiploma

//=========================================================================
            string DeleteAfterDiploma = "DELETE FROM [AfterDiploma] WHERE
[IdWorker] = @IdWorker";
            SqlCommand delAD = new SqlCommand(DeleteAfterDiploma,
Program.Global.conn);
            delAD.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delAD.ExecuteNonQuery();

//=========================================================================
            //Видалити з Positions

//=========================================================================
            string DeletePositions = "DELETE FROM [Positions] WHERE [IdWorker] =
@IdWorker";
            SqlCommand delPos = new SqlCommand(DeletePositions,
Program.Global.conn);
```

```
            delPos.Parameters.AddWithValue("@IdWorker",
Program.Global.ChosenWorker);
            delPos.ExecuteNonQuery();

            WorkerTable.Clear();
            WorkerAdapter.Fill(WorkerTable);
            WorkersGrid.DataSource = WorkerTable;
        }
        private void SortingRB_CheckedChanged(object sender, EventArgs e)
        {
            if (SortById.Checked == true)
            {
                WorkersGrid.Sort(WorkersGrid.Columns["IdWorker"],
ListSortDirection.Ascending);
            }
            if (SortByName.Checked == true)
            {
                WorkersGrid.Sort(WorkersGrid.Columns["SecondName"],
ListSortDirection.Ascending);
            }
            if (SortByPosition.Checked == true)
            {
                WorkersGrid.Sort(WorkersGrid.Columns["NameOfPosition"],
ListSortDirection.Ascending);
            }
        }

        private void SearchId_TextChanged(object sender, EventArgs e)
        {
            WorkersView.RowFilter = "IdWorker like '%" + SearchId.Text + "%'";
            WorkersGrid.DataSource = WorkersView;
        }

        private void SearchSndName_TextChanged(object sender, EventArgs e)
        {
            WorkersView.RowFilter = "SecondName like '%" + SearchSndName.Text +
"%'";
            WorkersGrid.DataSource = WorkersView;
        }

        private void SearchFstName_TextChanged(object sender, EventArgs e)
        {
            WorkersView.RowFilter = "FirstName like '%" + SearchFstName.Text +
"%'";
            WorkersGrid.DataSource = WorkersView;
        }

        private void WorkersGrid_SelectionChanged(object sender, EventArgs e)
        {
            if (WorkersGrid.Rows.Count != 0)
            {
                Program.Global.ChosenWorker =
WorkersGrid.Rows[WorkersGrid.CurrentRow.Index].Cells[1].Value.ToString();
            }
        }

        private void Print_Click(object sender, EventArgs e)
        {

        }

    }
}
```

// **RSA_Encryption.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Security.Cryptography;

namespace RSA_Encryption_Decryption
{
    public partial class Form1 : Form
    {
        UnicodeEncoding ByteConverter = new UnicodeEncoding();
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
        byte[] plaintext;
        byte[] encryptedtext;

        public Form1()
        {
            InitializeComponent();
        }

        private void EncButton_Click(object sender, EventArgs e)
        {
            plaintext = ByteConverter.GetBytes(PlainTextInput.Text);
            encryptedtext = Encryption(plaintext, RSA.ExportParameters(false),
false);
            CipherText.Text = ByteConverter.GetString(encryptedtext);
        }

        private void DecButton_Click(object sender, EventArgs e)
        {
            byte[] decryptedtext = Decryption(encryptedtext,
RSA.ExportParameters(true), false);
            PlainTextOutput.Text = ByteConverter.GetString(decryptedtext);
        }

        //Шифрування
        static public byte[] Encryption(byte[] Data, RSAParameters RSAKey, bool
DoOAEPPadding)
        {
            try
            {
                byte[] encryptedData;
                using (RSACryptoServiceProvider RSA = new
RSACryptoServiceProvider())
                {
                    RSA.ImportParameters(RSAKey); encryptedData =
RSA.Encrypt(Data, DoOAEPPadding);
                } return encryptedData;
            }
            catch (CryptographicException e)
            {
                Console.WriteLine(e.Message);
                return null;
            }
        }

        //Дешифрування
        static public byte[] Decryption(byte[] Data, RSAParameters RSAKey, bool
DoOAEPPadding)
        {
            try
```

```
            {
                byte[] decryptedData;
                using (RSACryptoServiceProvider RSA = new
RSACryptoServiceProvider())
                {
                    RSA.ImportParameters(RSAKey);
                    decryptedData = RSA.Decrypt(Data, DoOAEPPadding);
                }
                return decryptedData;
            }
            catch (CryptographicException e)
            {
                Console.WriteLine(e.ToString());
                return null;
            }
        }

    }
}
```