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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи централізованого розподілу ключів”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-20МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Окаєвич Ю.О.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Тетяна СМІРНОВА
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Центр *Заочної та дистанційної освіти*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"Інформаційні технології"*_____

Спеціальність *122 "Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

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

Окаєвич Юлії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи централізованого розподілу ключів*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 40-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи централізованого розподілу ключів*

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

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

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

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Окаєвич Ю.О. Дослідження та програмна реалізація системи централізованого розподілу ключів. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи централізованого розподілу ключів.

Метою розробки є дослідження та програмна реалізація системи централізованого розподілу ключів.

Об'єктом дослідження є процес централізованого розподілу ключів.

Предметом дослідження є методи централізованого розподілу ключів.

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

Результат роботи – програмна реалізація системи централізованого розподілу ключів.

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

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

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: Комп'ютерні науки, розподіл ключів

ABSTRACT

Okaievych Yu.O. Research and software implementation of the centralized key distribution system. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of the centralized distribution of keys is developed.

The purpose of development is research and software implementation of the centralized key distribution system.

The object of research is the process of centralized distribution of keys.

The subject of research is the methods of centralized key distribution.

Research methods are based on methods of information security theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of a centralized key distribution system.

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

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC PC with Windows XP / Vista / 7/8/10.

The program is developed in the Delphi 10.4.1 environment.

Keywords: Computer science, key distribution

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІБ – інформаційна безпека

ІС – інформаційна система

НСД – несанкціонований доступ

ОБК – одноразовий багатоалфавітний кодер

ЦСФРК – центр сертифікації, формування й розподілу ключів

СА – сертифікаційний центр

PIN-код – персональний ідентифікаційний код

PKI – інфраструктура відкритого ключа

ВСТУП

Актуальність теми. На сучасному етапі розвитку систем передачі даних, особливо гостро стоїть питання забезпечення захисту інформації, що передається по каналах зв'язку цих систем. У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти пропонується до розгляду захист інформації в банківській мережі. Для забезпечення захищеності даних, які передаються по даній мережі, у банківських мережах регулярно відбувається зміна ключів. Формуванню системи централізованого розподілу ключів й присвячена випускна кваліфікаційна робота за другим (магістерським) рівнем вищої освіти.

Ця проблема є дуже актуальною та важливою, в зв’язку з тим, що в банківський мережі за день проходить кілька десятків тисяч транзакцій, і якщо зловмисник зможе взломати систему розподілу ключів, то банк понесе колосальні втрати [1]. При цьому виникає питання не тільки фінансових втрат, а й питання втрати іміджу банку. Тобто, якщо з вини служби безпеки банку, яка не зуміла забезпечити необхідний рівень захисту системи розподілу ключів, відбудеться взлом цієї системи, то у банка буде підмочена репутація і клієнти оберуть інший банк для проведення операцій з готівковими та безготівковими операціями.

Для забезпечення стійкості системи необхідно забезпечити захист наступних компонент цієї системи: центру сертифікації, формування й розподілу ключів (ЦСФРК), серверів розподіленої обробки й користувальницькі пристрої [2].

При цьому необхідно комплексно вирішувати проблему захисту інформації, тобто, враховувати не тільки загрози зі сторони взлому програмного забезпечення, а й забезпечувати фізичний, правовий та технічний рівень захисту вищеперерахованих елементів системи [3-6].

Розроблена в випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти система повинна забезпечувати можливість ефективного, з гарантованою надійністю обміну закритою інформацією. Кожний сертифікований

користувач, звернувшись до центру сертифікації, формування й розподілу ключів, повинен змогти обмінюватися закритою інформацією з будь -яким сервером або користувачем банківської мережі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи централізованого розподілу ключів.

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

– Огляд існуючих систем централізованого розподілу ключів.

– Дослідження системи централізованого розподілу ключів.

– Програмна реалізація системи централізованого розподілу ключів.

Об'єктом дослідження є процес централізованого розподілу ключів.

Предметом дослідження є методи централізованого розподілу ключів.

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

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

– Удосконалено метод централізованого розподілу ключів.

– Розроблено вітчизняний продукт централізованого розподілу ключів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі централізованого розподілу ключів.

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

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

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

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Словосполучення "інформаційна безпека" у різних контекстах може мати різний сенс. У Доктрині інформаційної безпеки України термін "інформаційна безпека" використовується в широкому змісті. Мається на увазі стан захищеності національних інтересів в інформаційній сфері, обумовлених сукупністю збалансованих інтересів особистості, суспільства й держави [7].

У Законі України "Про інформацію" інформаційна безпека визначається аналогічним образом – як стан захищеності інформаційного середовища суспільства, що забезпечує її формування, використання й розвиток в інтересах громадян, організацій, держави.

Під інформаційною безпекою (ІБ) ми будемо розуміти захищеність інформації й підтримуючої інфраструктури від випадкових або навмисних впливів природного або штучного характеру, які можуть завдати неприйнятної шкоди суб'єктам інформаційних відносин, у тому числі власникам і користувачам інформації й підтримуючої інфраструктури [8].

Захист інформації – це комплекс заходів, спрямованих на забезпечення інформаційної безпеки.

Таким чином, правильний з методологічної точки зору підхід до проблем інформаційної безпеки починається з виявлення суб'єктів інформаційних відносин і інтересів цих суб'єктів, пов'язаних з використанням інформаційних систем (ІС). Погрози інформаційної безпеки – це зворотний бік використання інформаційних технологій.

Із цього положення можна вивести два важливих наслідки:

1. Трактування проблем, пов'язаних з інформаційною безпекою, для різних категорій суб'єктів може істотно розрізнятися. Для ілюстрації досить

зіставити режимні державні організації й навчальні інститути. У першому випадку "нехай краще все зламається, ніж ворог довідається хоч один секретний біт", у другому – "так немає в нас ніяких секретів, аби тільки все працювало".

2. Інформаційна безпека не зводиться винятково до захисту від несанкціонованого доступу до інформації, це принципово більш широке поняття. Суб'єкт інформаційних відносин може постраждати (зазнати збитків і/або одержати моральний збиток) не тільки від несанкціонованого доступу, але й від поломки системи, що викликало перерву в роботі. Більше того, для багатьох відкритих організацій (наприклад, навчальних) властиво захист від несанкціонованого доступу до інформації стоїть по важливості аж ніяк не на першому місці.

Вертаючись до питань термінології, відзначимо, що термін "комп'ютерна безпека" (як еквівалент або замінник ІБ) представляється нам занадто вузьким. Комп'ютери – тільки одна зі складових інформаційних систем, і хоча наша увага буде зосереджено в першу чергу на інформації, що зберігається, обробляється й передається за допомогою комп'ютерів, її безпека визначається всією сукупністю складових і, у першу чергу, самою слабкою ланкою, якою в переважній більшості випадків виявляється людина (записавший, наприклад, свій пароль на "гірчичнику", приліпленому до монітора).

Звернемо увагу, що у визначенні ІБ перед іменником "збиток" стоїть прикметник "неприйнятний". Очевидно, застрахуватися від всіх видів збитку неможливо, тим більше неможливо зробити це економічно доцільним способом, коли вартість захисних засобів і заходів не перевищує розмір очікуваного збитку. Виходить, із чимсь доводиться миритися й захищатися треба тільки від того, із чим упокоритися ніяк не можна. Іноді таким неприпустимим збитком є нанесення шкоди здоров'ю людей або стану навколишнього середовища, але частіше поріг неприйнятності має матеріальне (грошове) вираження, а метою захисту інформації стає зменшення розмірів збитку до припустимих значень.

1.2 Область застосування

Інформаційна безпека – багатогранна, можна навіть сказати, багатомірна область діяльності, у якій успіх може принести тільки систематичний, комплексний підхід.

Спектр інтересів суб'єктів, пов'язаних з використанням інформаційних систем, можна розділити на наступні категорії: забезпечення **доступності**, **цілісності** й **конфіденційності** інформаційних ресурсів і підтримуючої інфраструктури.

Іноді в число основних складових ІБ включають захист від несанкціонованого копіювання інформації, але, на наш погляд, це занадто специфічний аспект із сумнівними шансами на успіх, тому ми не станемо його виділяти.

Пояснимо поняття доступності, цілісності й конфіденційності.

Доступність – це можливість за прийнятний час одержати необхідну інформаційну послугу. Під цілісністю мається на увазі актуальність і несуперечність інформації, її захищеність від руйнування й несанкціонованої зміни. Нарешті, конфіденційність – це захист від несанкціонованого доступу до інформації.

Інформаційні системи створюються для одержання певних інформаційних послуг. Якщо по тим або інших причинах надати ці послуги користувачам стає неможливо, це, мабуть, завдає шкоди всім суб'єктам інформаційних відносин. Тому, не протиставляючи доступність іншим аспектам, ми виділяємо її як найважливіший елемент інформаційної безпеки.

Особливо яскраво провідна роль доступності проявляється в різного роду системах керування – виробництвом, транспортом і т.п. Зовні менш драматичні, але також досить неприємні наслідки – і матеріальні, і моральні – може мати тривала неприступність інформаційних послуг, якими користується велика

кількість людей (продаж залізничних і авіаквитків, банківські послуги й т.п.). В банківських мережах доступність повинна бути зведена до мінімуму.

Цілісність можна підрозділити на статичну (що розуміється як незмінність інформаційних об'єктів) і динамічну (стосовну до коректного виконання складних дій (транзакцій)). Засоби контролю динамічної цілісності застосовуються, зокрема, при аналізі потоку фінансових повідомлень із метою виявлення крадіжки, з або дублювання окремих повідомлень.

Цілісність виявляється найважливішим аспектом ІБ у тих випадках, коли інформація служить "керівництвом до дії". Рецептура ліків, запропоновані медичні процедури, набір і характеристики комплектуючих виробів, хід технологічного процесу – все це приклади інформації, порушення цілісності якої може виявитися в буквальному значенні смертельним. Неприємно й перекручування офіційної інформації, будь те текст закону або сторінка Web-сервера якої-небудь урядової організації, або розпорядження банківського керівництва. Конфіденційність – самий пророблений у нас у країні аспект інформаційної безпеки. На жаль, практична реалізація мер по забезпеченню конфіденційності сучасних інформаційних систем натрапляє в Україні на серйозні труднощі. По-перше, відомості про технічні канали витоку інформації є закритими, так що більшість користувачів позбавлені можливості скласти уявлення про потенційні ризики. По-друге, на шляху користувальницької криптографії як основного засобу забезпечення конфіденційності стоять численні законодавчі перепони й технічні проблеми.

Якщо повернутися до аналізу інтересів різних категорій суб'єктів інформаційних відносин, то майже для всіх, хто реально використовує ІС, на першому місці стоїть доступність. Практично не уступає їй по важливості цілісність – який зміст в інформаційній послузі, якщо вона містить перекручені відомості?

Нарешті, конфіденційні моменти є також у багатьох організацій (навіть у згадуваних вище навчальних інститутах намагаються не розголошувати відомості про зарплату співробітників) і окремих користувачів (наприклад, паролі).

Важливість і складність проблеми інформаційної безпеки

Інформаційна безпека є одним з найважливіших аспектів інтегральної безпеки, на якому би рівні ми не розглядали останню – національному, галузевому, корпоративному або персональному.

При аналізі проблематики, пов'язаної з інформаційною безпекою, необхідно враховувати специфіку даного аспекту безпеки, що складає в тім, що інформаційна безпека є складова частина інформаційних технологій – області, що розвивається безпрецедентно високими темпами. Тут важливі не стільки окремі рішення (закони, навчальні курси, програмно-технічні вироби), що перебувають на сучасному рівні, скільки механізми генерації нових рішень, що дозволяють жити в темпі технічного прогресу.

На жаль, сучасна технологія програмування не дозволяє створювати безпомилкові програми, що не сприяє швидкому розвитку засобів забезпечення ІБ. Варто виходити з того, що необхідно конструювати надійні системи (інформаційної безпеки) із залученням ненадійних компонентів (програм). У принципі, це можливо, але вимагає дотримання певних архітектурних принципів і контролю стану захищеності на всьому протязі **життєвого циклу ІС**.

Збільшення числа атак – ще не сама більша неприємність. Гірше те, що постійно виявляються нові уразливі місця в програмному забезпеченні (вище ми вказували на обмеженість сучасної технології програмування) і, як наслідок, з'являються нові види атак. У таких умовах системи інформаційної безпеки повинні вміти протистояти різноманітним атакам, як зовнішнім, так і внутрішнім, атакам автоматизованим і скоординованим. Іноді напад триває частки секунди; часом промацування уразливих місць ведеться повільно й розтягується на годинники, так що підозріла активність практично непомітна.

Метою зловмисників може бути порушення всіх складових ІБ – доступності, цілісності або конфіденційності.

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

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Симетричні алгоритми розподілу ключів

При симетричному шифруванні два учасники, які хочуть обмінюватися конфіденційною інформацією, повинні мати однаковий ключ. Частота зміни ключа повинна бути досить великою, щоб у зловмисника не вистачило часу для повного перебору ключів. Отже, стійкість будь-якої криптосистеми багато в чому залежить від технології розподілу ключа. Цей термін означає передачу ключа двом учасникам, які хочуть обмінюватися даними, таким способом, щоб ніхто інший не міг змінити цей ключ. Для двох учасників *A* й *B* розподіл ключа може бути виконано одним з наступних способів.

1. Ключ може бути створений *A* й фізично переданий *B*.
2. Третя сторона може створити ключ і фізично передати його *A* та *B*.
3. *A* й *B* мають попередньо створений і недовго використовуваний ключ, один учасник може передати новий ключ іншому, застосувавши для шифрування старий ключ.
4. Якщо *A* й *B* кожного мають безпечне з'єднання із третім учасником *C*, *C* може передати ключ по цьому безпечному каналу *A* й *B*.

Перший і другий способи називаються ручним розподілом ключа. Це самі надійні способи розподілу ключа, однак у багатьох випадках користуватися ними незручно й навіть неможливо. У банківській системі будь-який хост або сервер повинен мати можливість обмінюватися конфіденційною інформацією з багатьма автентифікованими хостами й серверами. Таким чином, кожний хост повинен

мати набір ключів, підтримуваний динамічно. Проблема особливо актуальна в великих банківських системах.

Кількість необхідних ключів залежить від числа учасників, які повинні взаємодіяти. Якщо виконується шифрування на мережному або IP-рівні, то ключ необхідний для кожної пари хостів у мережі. Таким чином, якщо є N хостів, то необхідне число ключів $[N\ (N-1)]/2$. Якщо шифрування виконується на прикладному рівні, то ключ потрібний для кожної пари прикладних процесів, яких набагато більше, ніж хостів.

Третій спосіб розподілу ключів може застосовуватися на будь-якому рівні стека протоколів, але якщо зловмисник одержує можливість доступу до одного ключа, то вся послідовність ключів буде розкрита. Більше того, однаково повинне бути проведене первісне поширення великої кількості ключів.

Тому в великих банківських системах широко застосовуються різні варіанти четвертого способу. У цій схемі передбачається існування так званого центра розподілу ключів, що відповідає за розподіл ключів для хостів, процесів і додатків. Використання центра розподілу ключів засновано на використанні ієрархії ключів. Як мінімум використовується два типи ключів: майстер-ключі й ключі сесії.

Для забезпечення конфіденційного зв'язку між кінцевими системами використовується тимчасовий ключ, називаний ключем сесії. Звичайно ключ сесії використовується для шифрування транспортного з'єднання й потім знищується. Кожний ключ сесії повинен бути отриманий по мережі із центра розподілу ключів. Ключі сесії передаються в зашифрованому виді, використовуючи майстер-ключ, що розділяється між центром розподілу ключів і кінцевою системою. Ці майстер-ключі також повинні розподілятися деяким безпечним способом. Однак при цьому істотно зменшується кількість ключів, що вимагають ручного розподілу. Якщо існує N учасників, які хочуть установлювати з'єднання, то в кожний момент часу необхідно $[N\ (N-1)]/2$ ключів сесії. Але потрібно тільки N майстер-ключів, по одному для кожного учасника. Час життя ключа сесії

як правило дорівнює часу життя самої сесії. Чим частіше міняються ключі сесії, тим більш безпечними вони є, тому що зловмисник має менше часу для взламування даного ключа сесії. З іншого боку, розподіл ключів сесії затримує початок будь-якого обміну й завантажує мережу. Політика безпеки повинна збалансувати ці умови для визначення оптимального часу життя конкретного ключа сесії.

Якщо з'єднання має довгий час життя, то повинна існувати можливість періодично міняти ключ сесії. Для протоколів, що не підтримують з'єднання, таких як протокол, орієнтований на транзакції, немає явної ініціалізації або переривання з'єднання. Отже, неясно, як часто треба міняти ключ сесії. Більшість підходів ґрунтується на використанні нового ключа сесії для кожного нового обміну. Найбільше часто застосовується стратегія використання ключа сесії тільки для фіксованого періоду часу або тільки для певної кількості транзакцій.

Розподіл ключів на основі PKI

Метою PKI (Інфраструктури Відкритого Ключа) є надання довіреного й дійсного відкритого ключа учасника, а також керування всім життєвим циклом сертифіката відкритого ключа.

Одним з вимог до алгоритмів цифрового підпису є вимога, щоб було рахунково неможливо, знаючи відкритий ключ K_B, визначити закритий ключ $K_З$. Здавалося б, відкритий ключ K_B можна поширювати по небезпечних мережах і зберігати в небезпечних репозиторіях. Але при цьому варто пам'ятати, що при використанні цифрового підпису необхідно бути впевненим, що суб'єкт, з яким здійснюється взаємодія з використанням алгоритму відкритого ключа, є власником відповідного закритого ключа. У противному випадку можлива атака, коли опонент заміняє відкритий ключ законного учасника своїм відкритим ключем, залишивши при цьому ідентифікатор законного учасника без зміни. Це дозволить йому створювати підписи від імені законного учасника й читати зашифровані повідомлення, послані законному учасникові, використовуючи для цього свій закритий ключ, що відповідає підміненому відкритому ключу. Для

запобігання такої ситуації варто використовувати сертифікати, які є структурами даних, що зв'язують значення відкритого ключа із суб'єктом. Для зв'язування необхідна наявність довіреного що засвідчує (або сертифікаційного) центра (Certification Authority – CA), що перевіряє ідентифікацію суб'єкта й підписує його відкритий ключ і деяку додаткову інформацію своїм закритим ключем.

Основним поняттям PKI є поняття сертифіката.

Сертифікат учасника, створений CA, має наступні характеристики:

1. Будь-який учасник, що має відкритий ключ CA, може відновити відкритий ключ учасника, для якого CA створив сертифікат.

2. Ніхто іншої, крім даного центра, що засвідчує, не може модифікувати сертифікат без виявлення цього стороною, що перевіряє.

Часто використовується наступна нотація для позначення сертифіката: CA <<A>> – сертифікат користувача *A*, виданий сертифікаційним центром CA.

CA підписує сертифікат своїм закритим ключем. Якщо відповідний відкритий ключ відомий користувачеві, то користувач може перевірити, що сертифікат, підписаний CA, дійсний.

Тому що сертифікати не можуть бути змінені без виявлення цього, їх можна розмістити в загальнодоступній директорії й пересилати по відкритих каналах зв'язку без побоювання, що хтось може їх змінити.

У кожному разі якщо *B* має сертифікат *A*, *B* упевнений, що повідомлення, що він розшифровує відкритим ключем *A*, ніхто не міг переглянути, і що повідомлення, підписане закритим ключем *A*, не змінювалося. При великій кількості користувачів нерозумно підписувати сертифікати всіх користувачів в одного CA. Крім того, якщо існує єдиний CA, що підписує сертифікати, кожний користувач повинен мати копію відкритого ключа CA, щоб перевіряти підпису. Цей відкритий ключ повинен бути переданий кожному користувачеві абсолютно безпечним способом (із забезпеченням цілісності й автентифікації), щоб користувач був упевнений у підписані їм сертифікатах. Таким чином, у випадку

великої кількості користувачів краще мати кілька СА, кожний з яких безпечно надає свій відкритий ключ деякій підмножині користувачів.

Тепер припустимо, що *A* одержав сертифікат від уповноваженого органа X_1, і *B* одержав сертифікат від уповноваженого органа X_2. Якщо *A* не знає безпечним способом відкритий ключ X_2, то сертифікат В, отриманий від X_2, для нього невірний. *A* може прочитати сертифікат *B*, але не в змозі перевірити підпис. Проте, якщо два СА можуть безпечно обмінюватися своїми відкритими ключами, можлива наступна процедура для одержання *A* відкритого ключа *B*.

1. *A* одержує з директорії сертифікат X_2, підписаний X_1. так як *A* знає відкритий ключ X_1 надійним способом, *A* може одержати відкритий ключ X_2 з даного сертифіката й перевірити його за допомогою підпису X_1 у сертифікаті.

2. Потім *A* вертається назад у директорію й одержує сертифікат *B*, підписаний X_2. так як *A* тепер має відкритий ключ X_2 надійним способом, *A* може перевірити підпис і безпечно одержати відкритий ключ *B*.

Для одержання відкритого ключа *B A* використовує ланцюжок сертифікатів. У наведеній вище нотації цей ланцюжок виглядає в такий спосіб:

$$X_1 << X_2 >> X_2 << B >>$$

Аналогічно В може одержати відкритий ключ А с допомогою такого ж ланцюжка:

$$X_2 << X_1 >> X_1 << A >>$$

Дана схема не обов'язково обмежена ланцюжком із двох сертифікатів. Для одержання ланцюжка може використовуватися шлях СА довільної довжини. Ланцюжок, що містить N елементів, виглядає в такий спосіб:

$$X_1 << X_2 >> X_2 << X_3 >> \ldots X_N << B >>$$

У цьому випадку кожна пара СА у ланцюжку (X_i , X_{i+1}) повинна створити сертифікати друг для друга.

Всі ці сертифікати СА необхідно розмістити в директорії, і користувачі повинні мати інформацію про те, як вони зв'язані один з одним, щоб одержати шлях до сертифіката відкритого ключа іншого користувача. Це визначає PKI

Розподіл ключів на основі протоколу Kerberos

Kerberos є автентифікаційним сервісом і вирішує наступне завдання: припустимо, що існує відкрита банківська мережа, у якій користувачі, що працюють за своїми комп'ютерами, хочуть одержати доступ до розподілених в мережі серверів. Сервери повинні мати можливість надавати доступ тільки авторизованим користувачам, тобто автентифікувати запити на надання тих або інших послуг. У банківській мережі робоча станція не може бути довіреною системою, що коректно ідентифікує своїх користувачів для доступу до мережних серверів. Зокрема, існують наступні погрози:

1. Зловмисник може одержати фізичний доступ до якої-небудь робочої станції й спробувати ввійти під чужим ім'ям.

2. Зловмисник може змінити мережну адресу своєї робочої станції, щоб запити, що посилаються з невідомої робочої станції, приходили з відомої робочої станції.

3. Зловмисник може переглядати трафік і використовувати replay-атаку для одержання доступу на сервер або розриву з'єднання законних користувачів.

У всіх цих випадках неавторизований користувач може одержати доступ до сервісів і даних, не маючи на те права. Для того щоб не вбудовувати ретельно розроблені протоколи автентифікації на кожний сервер, Kerberos створює централізований автентифікуючий сервер, у чиї функції входить автентифікація користувачів для серверів і серверів для користувачів. На відміну від більшості схем автентифікації, Kerberos застосовує винятково симетричне шифрування, не використовуючи шифрування з відкритим ключем.

Якщо користувачі використовують не з'єднані в мережу комп'ютери, то користувальницькі й системні ресурси й файли можна вберегти від зловмисників, забезпечивши фізичний захист кожного комп'ютера. Коли користувачі обслуговуються централізованою системою поділу часу, безпеку повинна забезпечувати саме вона. Операційна система може проводити політику

керування доступом на основі ідентифікатора користувача й підтримувати строго певну процедуру входу для ідентифікації й автентифікації користувача.

В умовах мережної взаємодії подібні сценарії неприйнятні. Найбільш загальним випадком є розподілена архітектура, що складається з робочих станцій користувача (клієнтів) і розподілених серверів. У подібному оточенні можуть використовуватися три підходи до забезпечення безпеки:

1. Автентифікація користувача на своїй робочій станції.

2. Автентифікація користувача на кожному сервері, до якого він повинен мати доступ.

3. Автентифікація користувача на спеціальному сервері, що видає відповідний квиток (Ticket) для доступу на сервер.

У невеликих закритих мережах, у яких всі системи працюють у єдиному відділені банку, першої й, можливо, другої стратегії виявляється досить. Але в великих банківських мережах, де підтримуються мережні з'єднання між комп'ютерами, для захисту інформації й ресурсів користувачів необхідний третій підхід. Цей третій підхід підтримується Kerberos. Kerberos припускає наявність розподіленої архітектури клієнт/сервер і використовує один або більше серверів Kerberos для надання автентифікаційних сервісів.

Kerberos повинен задовольняти наступним вимогам:

1. Безпека.

2. Надійність.

3. Прозорість.

4. Масштабованість.

Для реалізації цих вимог Kerberos використовує тристоронній автентифікаційний діалог, заснований на протоколі Нидхема й Шредера. Така система є довіреною в тому випадку, якщо клієнти й сервери довіряють Kerberos бути посередником при автентифікації. Цей автентифікаційний діалог безпечний тією самою мірою, у якій безпечний сам сервер Kerberos.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняють ся з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

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

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64 -bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

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

забезпечення, яке призначено для системи централізованого розподілу ключів.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Синтез одноразової системи з відкритою передачею ключів

Характерна риса одноразової системи шифрування – одноразове використання ключової послідовності. Така система шифрує вихідний відкритий текст X у шифротекст Y з використанням одноразової випадкової ключової послідовності K. Для її реалізації іноді використовують одноразовий блокнот, складений з відривних сторінок; на кожній з них надрукована таблиця з випадковими числами (ключами) K_i. Блокнот виконується у двох екземплярах: один використовується відправником, а інший – одержувачем. Для кожного символу X_i повідомлення є свій ключ K_i з таблиці одержувача. Після того, як таблиця використана, її необхідно видалити із блокнота й знищити. Шифрування нового повідомлення починається з нової сторінки.

Абсолютна надійність одноразової системи доведена Клодом Шенноном. Одноразові системи нерозкриваємі, оскільки їх шифротекст не містить достатньої інформації для відновлення відкритого тексту. Однак можливості використання одноразових систем на практиці обмежені. Ключова послідовність довжиною не менш довжини повідомлення повинна передаватися одержувачеві повідомлення заздалегідь або окремо по деякому секретному каналі, що практично нездійсненно в сучасних банківських системах, де потрібно шифрувати багато мільйонів символів і забезпечувати засекречений зв'язок для безлічі абонентів. Ці недоліки усунуті в способі синтезу одноразових систем шифрування з відкритим поширенням ключа.

Розглянемо процес передачі інформації з лінії зв'язку, що з'єднує сервер банківської мережі (сторона A) та клієнтську ЕОМ банківської мережі (сторона B). Пропонований спосіб побудови одноразової системи дає можливість

передавати практично необмежений обсяг інформації з використанням випадкової перестановки тільки однієї таблиці ключів.

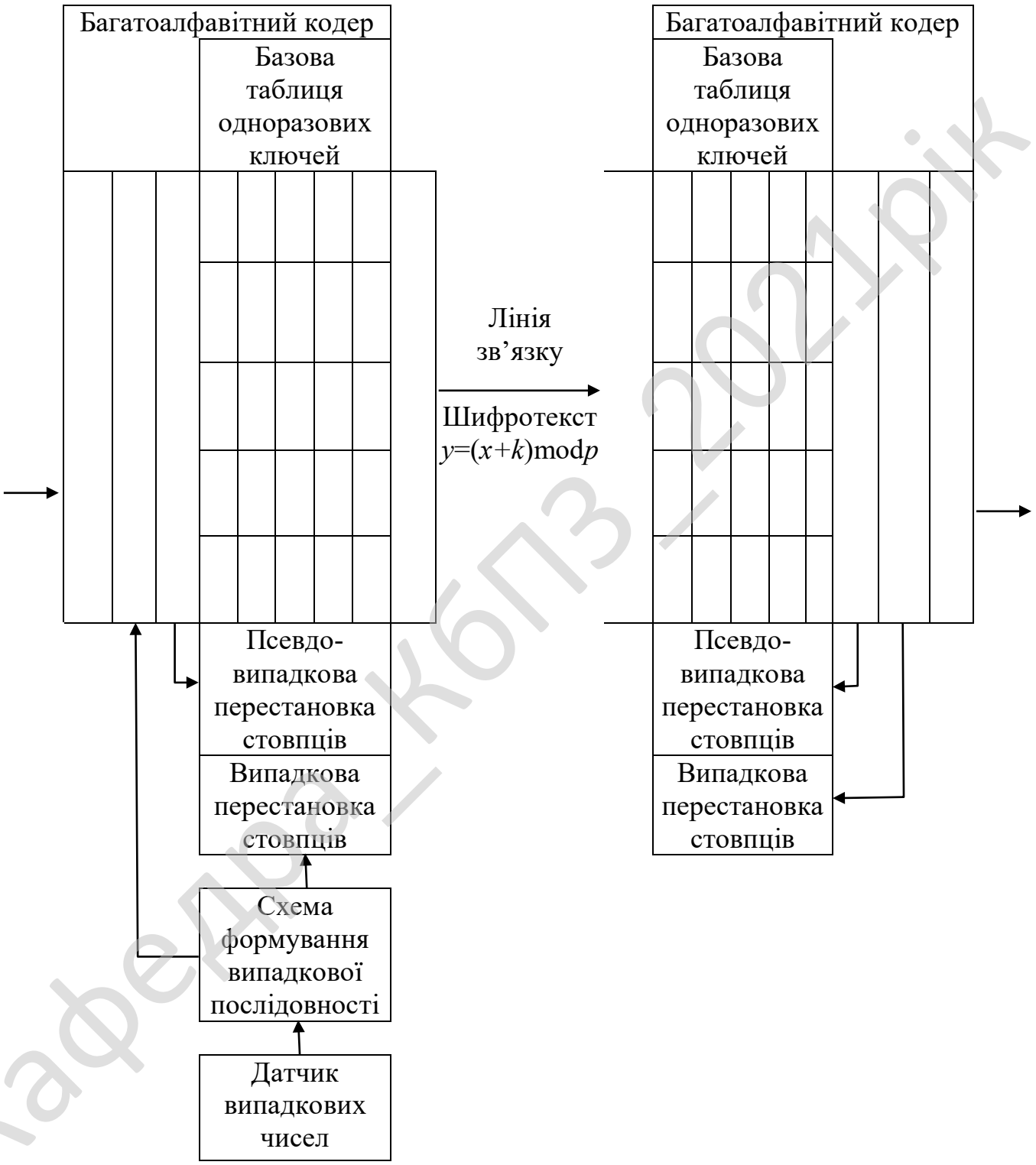


Рисунок 3.1 – Одноразовий багатоалфавитний кодер

У якості базового елемента, що шифрує, для системи з відкритою передачею ключів розроблений одноразовий багатоалфавитний кодер (ОБК). Система містить ОБК, датчик випадкових чисел, схему формування випадкової перестановки на стороні А и багатоалфавитний декодер на стороні В ОБК реалізується процес стохастичного кодування (рисунок 3.1).

До складу ОБК входить базова таблиця одноразових ключів, регістр перестановки інтерфейсу, регістри випадкової й псевдовипадкової перестановок рядків і стовпців базової таблиці. Аналогічний состав має й багатоалфавитний декодер. Регістри випадкових і псевдовипадкових перестановок рядків і таблиці інтерфейсу декодера містять комбінації, зворотні стосовно відповідних перестановок кодера.

Базова таблиця одноразових ключів на стороні A и на стороні B має розмір n x n. Кожний і-ий рядок таблиці містить випадкову ключову комбінацію, у яку входять всі можливі різні значення K_{ij} довжиною m біт. (Для таблиці кодів ASCII $m = 8$, $n = 256$, тому для шифрування тексту використовують таблицю розміром 256 x 256.)

$$K_i = K_{i0}, K_{i1}, ..., K_{in-1} \ (i = 1,...,n)$$

У результаті роботи датчика випадкових чисел i схеми формування випадкової перестановки генерується відповідна перестановка. В отриманій перестановці стовпці задають відповідність між вхідними значеннями (верхній рядок) і вихідними (нижній рядок).

Базова таблиця одноразових ключів на стороні A виконує дві функції:

– генерацію віртуальної змінної таблиці одноразових ключів з випадковою перестановкою стовпців і рядків;

– реалізацію логічного виводу, що забезпечує перетворення секретної перестановки в несекретну, застосовувану для відкритої передачі ключа.

Із цією метою кожний стовпець базової таблиці можна представити у вигляді вертикально розташованої перестановки. При цьому регістр псевдовипадкової перестановки, підключений до даної таблиці, у сполученні з

попередньою випадковою перестановкою, що передана на сторону В, забезпечує вибір стовпців таблиці для формування їхніх одноразових комбінацій. Названі комбінації стовпців застосовуються в процесі логічного виводу. Усього може бути сформоване $N = n!$ різних комбінацій стовпців.

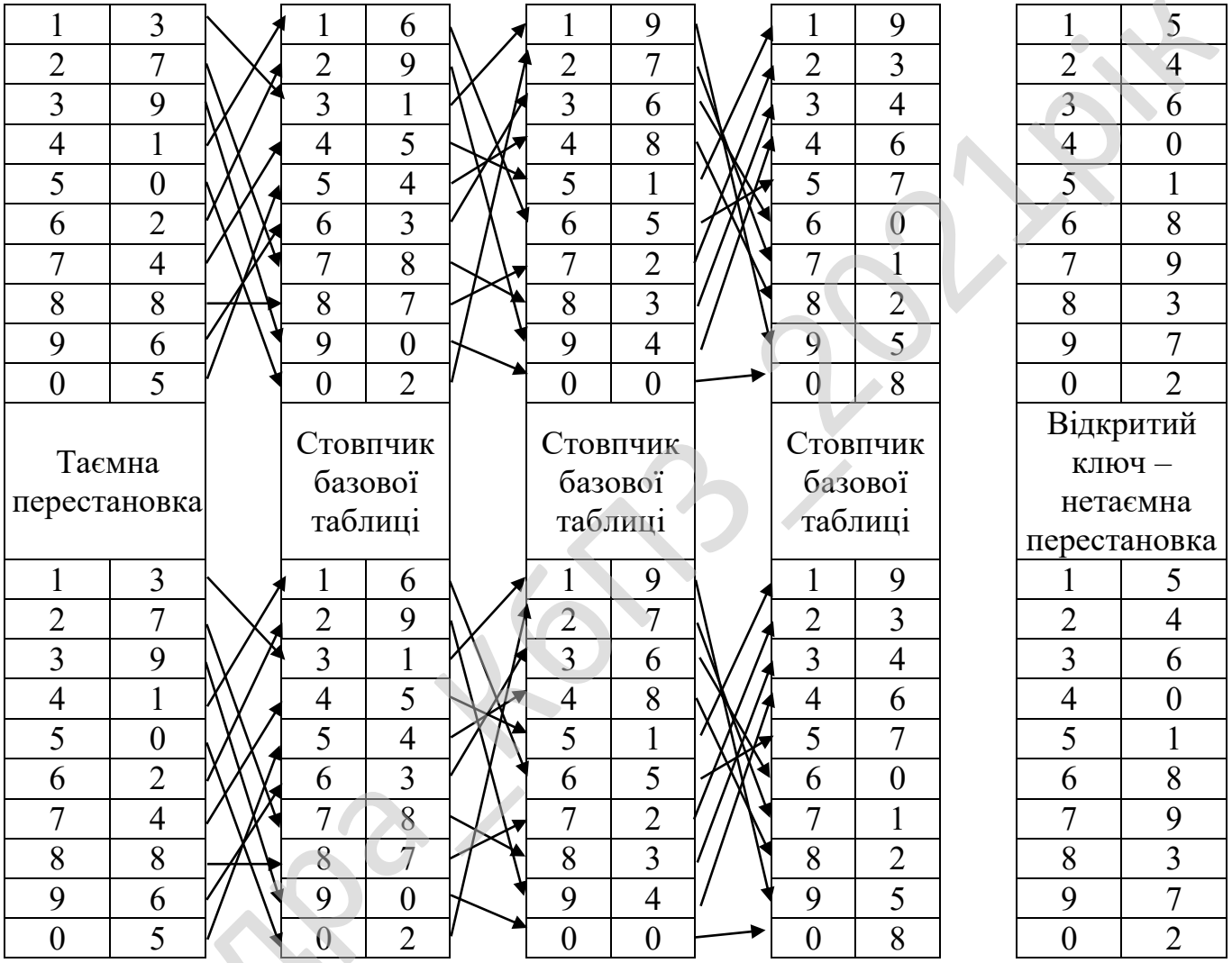


Рисунок 3.2 – Процес логічного виводу для формування відкритих ключів.

Логічний вивід, схематично представлений на рисунку 3.2, реалізує односпрямовану функцію $Y = F(x)$, що дозволяє на основі секретної перестановки, записаної в лівий регістр базової таблиці одноразових ключів, одержати несекретну перестановку, формовану у вихідному блоці ОБК. Тут x – значення секретної перестановки, F – функціональні зв'язки, формовані в процесі логічного

виводу з використанням чергової комбінації стовпців-перестановок, Y – відносна несекретна перестановка. Знаючи x і формуючи функціональні зв'язки F, легко одержати Y. Однак за відомим значенням Y, не знаючи всієї схеми функціональних зв'язків базової таблиці, не можна відновити вихідну секретну перестановку. Для цього необхідно зробити повний перебір на безлічі $V = n!$ всіх значень результуючих перестановок, одержуваних у ході логічного виводу, – свого роду ефект лабіринту, у центр якого поміщають людину із зав'язаними очами й, знявши пов'язку, пропонують шляхом випадкового перебору всіх можливих варіантів проходу знайти вихід.

У результаті виконання n процедур логічного виводу буде сформована таблиця несекретної перестановки, яка буде використовуватися в процесі шифрування для відкритої передачі ключа.

Таким чином, одночасно з передачею й шифруванням інформації на стороні користувача A генерується чергова випадкова перестановка. Потім за допомогою описаного алгоритму логічного виводу формується відповідна їй несекретна перестановка. Вона передається на сторону B у початку обміну інформацією й після передачі по лінії зв'язку n блоків шифротексту довжиною n символів кожний. На основі цієї перестановки на стороні B с допомогою базової таблиці, ідентичній базовій таблиці A, виконується процедура зворотного логічного виводу з метою одержання відповідної секретної перестановки (рис. 3.2). Ця процедура описується вираженням $x = F^{-1}(Y)$, де F^{-1} – функція зворотного логічного виводу, реалізованого за допомогою базової таблиці сторони B. Сформована секретна перестановка записується в регістри випадкових перестановок стовпців і рядків багатоалфавитного декодера. Шляхом використання зазначених регістрів у декодері відбувається утворення віртуальних таблиць одноразових ключів відповідно до отриманої випадкової перестановки. У результаті на сторонах A и B щораз будуть одночасно сформовані нові випадкові віртуальні таблиці одноразових ключів, ідентичних по змісту. Ці таблиці застосовуються при передачі зашифрованої інформації.

Розглянемо цей процес докладніше. Вихідний текст надходить на вхід регістра перестановки інтерфейсу ОБК, що забезпечує перестановку таблиці кодів ASCII. Так здійснюється перший етап перетворення вихідної інформації. Потім перетворений текст проходить через регістр випадкової перестановки рядків, що у сполученні з випадковою перестановкою стовпців реалізує чергову віртуальну таблицю одноразового ключа. При цьому застосування випадкових і псевдовипадкових перестановок забезпечує для кожної чергової комбінації вихідного тексту $X_i = (X_{i0}, X_{i1}, ..., X_{in-1})$ $(i = 1,...,n)$ формування унікальної одноразової ключової послідовності $K_i = K_{i0}, K_{i1}, ..., K_{in-1}$ $(i = 1,..,n)$. Усього для даної віртуальної таблиці, обумовленою черговою випадковою перестановкою, може бути утворено n таких ключових послідовностей. У результаті зроблених перестановок і замін у багатоалфавитному кодері символів кожної чергової послідовності X_i, а також циклічних зрушень стовпців таблиці, процес шифрування аналогічний класичній одноразовій системі. У декодері спочатку реалізується процедура ідентифікації символів шифротекста шляхом включення відповідних стовпців базової таблиці, а потім виробляються відповідні циклічні зрушення стовпців і за допомогою регістрів перестановок рядків виконуються зворотні перестановки, що забезпечують перетворення шифротекста у вихідний текст.

Після передачі $i = n$ чергових комбінацій шифротекста реалізується описаний процес відкритої передачі ключа (чергової секретної перестановки). За рахунок цього виробляється постійна (із заданим періодом) випадкова модифікація віртуальної таблиці багатоалфавитних кодера й декодера для одержання нових таблиць одноразових ключів. Потім триває передача, шифрування й дешифрування інформації з використанням нових таблиць одноразових ключів. При цьому передача несекретної перестановки реалізує функцію відкритої передачі ключів, виробленої після видачі кожних n блоків зашифрованої інформації. У результаті забезпечується гарантована надійність шифрування. Дійсно, самі базові таблиці одноразових ключів супротивникові

невідомі при будь-яких видах атак на дану систему шифрування (у явн ому виді вони не беруть участь у процесі шифрування інформації), тому формовані віртуальні таблиці одноразових ключів випадкові й непередбачені. З огляду на односпрямованість функції $Y = F(x)$ одержання несекретної перестановки, безліч варіантів модифікації віртуальних таблиць на сторонах *A* и *B* шляхом випадкової перестановки стовпців і рядків виміряється числом $V = n!$ Так, при використанні таблиці кодів ASCII із зазначеними параметрами m і n одержимо величину $V > 10^{500}$. Для більших значень *n* даний спосіб дозволяє передавати практично необмежені обсяги зашифрованої інформації в режимі одноразового ключа з гарантованим рівнем надійності, обумовленим числом $V = n!$ всіх можливих значень результуючих перестановок, які одержують у ході логічного виводу. Відзначимо, що в цьому випадку застосовується одна таблиця одноразових ключів розміром $n \times n$ і функція відкритої передачі ключів з використанням випадкової несекретної перестановки довжиною *n* байт. Важко навіть указати, скільки часу буде потрібно на перебори всіх варіантів перестановок на реальному комп'ютері. При цьому функція відкритої передачі ключів може періодично використовуватися для відновлення базової таблиці шляхом передачі нових значень її стовпців (перестановок). Зазначені значення стовпців генеруються за допомогою датчика випадкових чисел і схеми формування випадкових перестановок. У результаті після n циклів відновлення на сторонах *A* и *B* будуть отримані нові базові таблиці, використовувані далі при шифруванні.

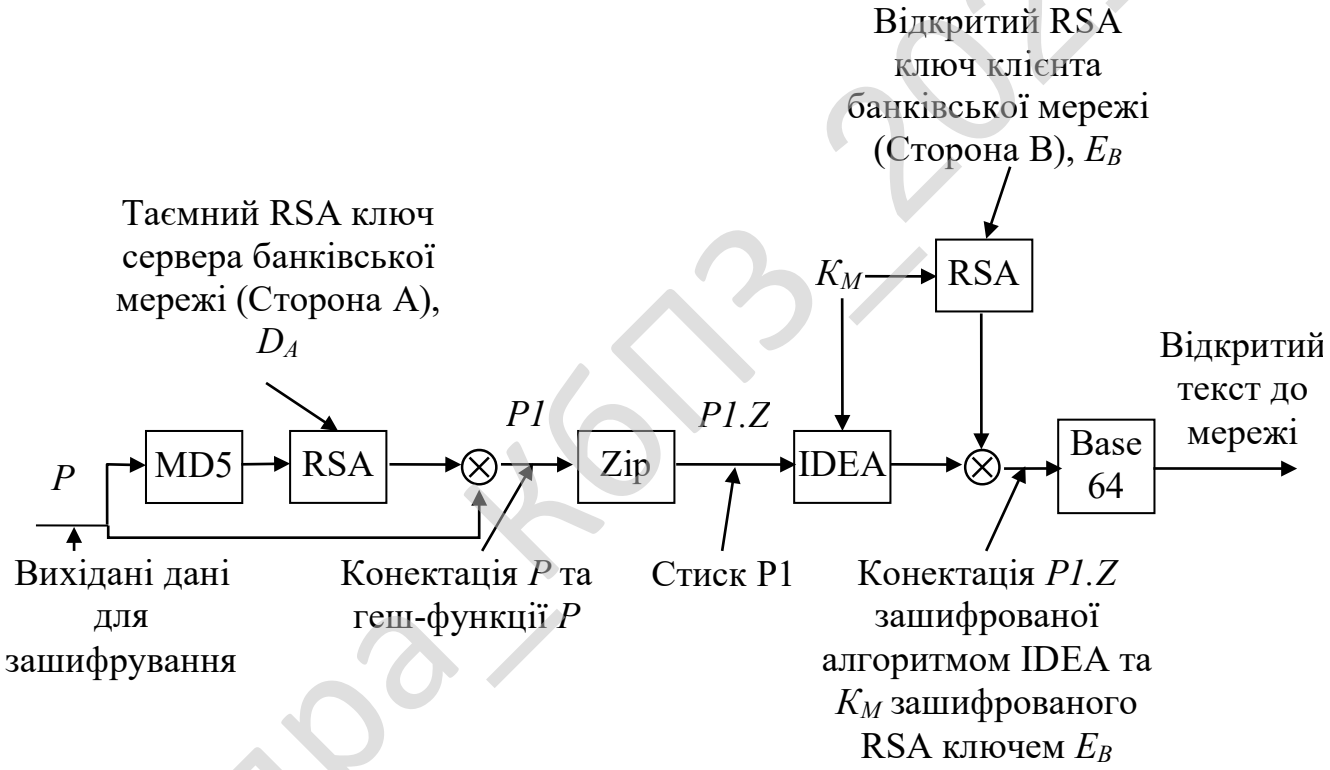
Процес кодування в ОБК практично не знижує швидкість передачі інформації з каналу зв'язку. Це дозволяє реалізувати швидкісні одноразові шифри для роботи в банковських мережах. Є ефективні технології забезпечення цілісності інформації, а також ідентифікації й автентифікації користувачів, перевірки дійсності повідомлень.

Алгоритм розподілу ключів на основі PGP

На основі алгоритму RSA розроблена програма PGP (Pretty Good Privacy). Це повний пакет безпеки, що включає засобу конфіденційності, установлення

дійсності, електронного підпису, стиску й все це в зручній для використання формі. Завдяки цьому система працює як на платформі Unix, так і MS-DOS/Windows, Macintosh і поширюється безкоштовно, тому вона одержала дуже широке поширення.

PGP використовує алгоритми шифрування RSA, IDEA і MD5. PGP підтримує компресію, переданих даних, їхню таємність, електронний підпис і засоби управління доступу до ключів. Схема роботи PGP показана на рисунку 3.3. На цьому рисунку – D_A, D_B особисті (закриті) ключі A и B відповідно, а E_A , E_B – їхні відкриті ключі.



K_M – одноразовий ключ для зашифрування повідомлення IDEA

⊗ – конектація

Рисунок 3.3 – Алгоритм шифрування при розподілу ключей PGP

Відзначимо, що секретний ключ для IDEA будується автоматично по ходу роботи PGP на стороні A и називається ключем сесії – K_M, що потім шифрується алгоритмом RSA з відкритим ключем користувача B. Так само варто звернути

увагу на те, що повільний алгоритм RSA використовується для шифрування коротких фрагментів тексту: 128 біт MD5 і 128 біт IDEA ключа.

PGP підтримує три довжини ключів:

– Звичайний – 314 біт (може бути розкритий за рахунок більших витрат).

– Комерційний – 512 біт (може бути розкритий спеціалізованими організаціями, назви яких, як правило, складається із трьох букв).

– Військовий – 1024 біта (не може бути розкритий поки ніким на землі).

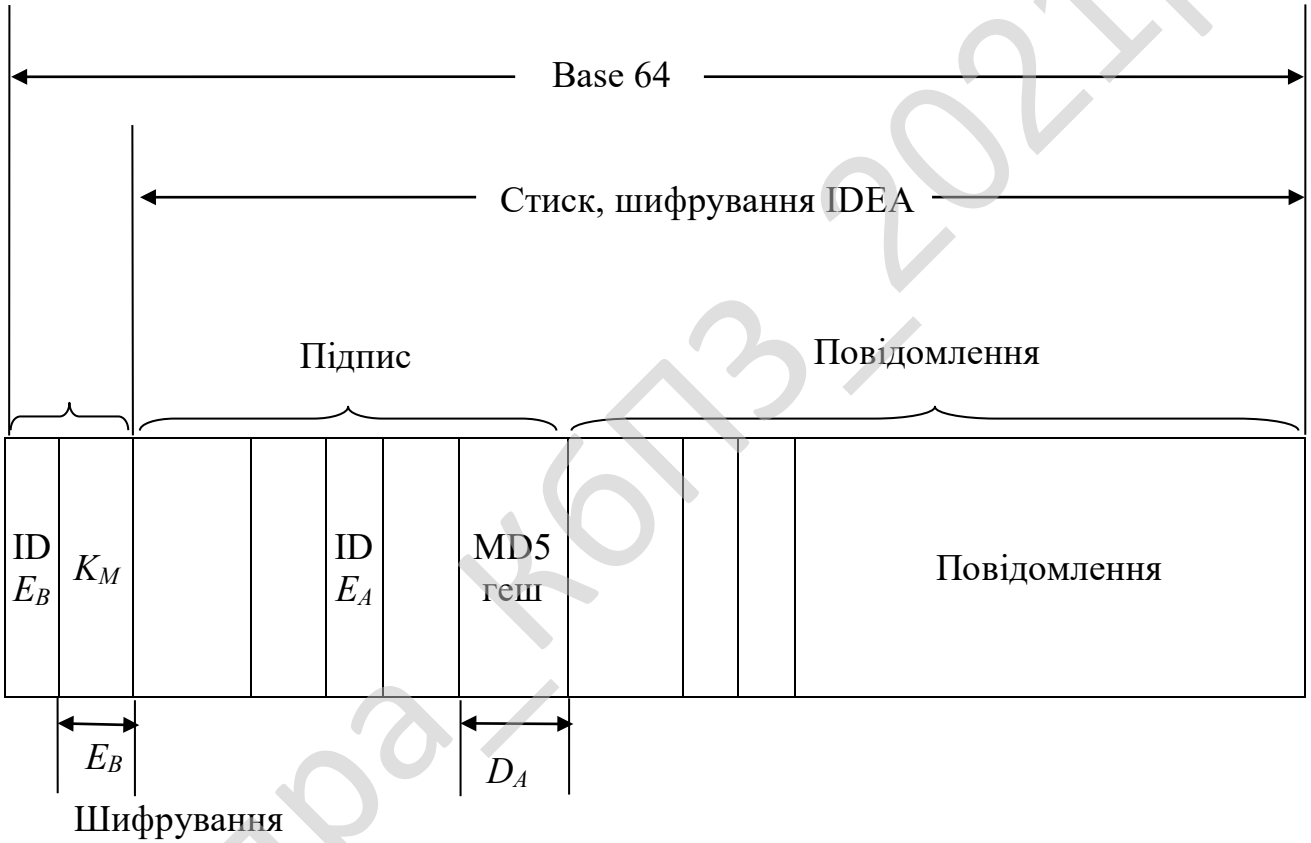


Рисунок 3.4 – Структурна схема PGP

Розподіл ключів. Web of Trust

Кожний користувач PGP може самостійно згенерувати свій ключ. Кожний користувач може привласнити ключу будь-яке ім'я й будь-який e-mail-адресу. Все це відкриває широкий простір для шахрайства й атак по методу "людина у середині". Щоб переконатися, що конкретний ключ дійсно належить

передбачуваному власникові, потрібно якось це перевірити. Це неважко, якщо ви особисто знайомі з людиною, у противному ж випадку може виявитися досить складним. Основний з механізмів, пропонованих PGP для рішення цієї проблеми – це *цифрові сертифікати ключів* і модель відносин довіри *Web of Trust.*

Сертифікат відкритого ключа PGP – це форма посвідчення, що несе ідентифікацію користувача (тобто об'єктивний спосіб його впізнавання), і пов'язана з певним відкритим ключем за допомогою підтверджувального підпису третьої сторони – підпису поручителя. На сьогоднішній день не існує устояного визначення підтверджувального підпису. Такий підпис на сертифікаті ключа може мати приблизно наступне значення: *"Я поручаюся в тім, що підписаний мною ключ дійсно належить особі, зазначеній у відомостях (ідентифікації) сертифіката".* Таке значення, на жаль, не можна вважати доста тнім і повним.

Тому більш точне значення підтверджувального підпису звучить так: *"Я поручаюся, ґрунтуючись на своїй особистій безпосередній переконаності й об'єктивних підтверджувальних свідченнях, у тім, що підписаний мною відкритий ключ і пов'язаний з ним закритий ключ дійсно належать особі, чиє ім'я, e-mail і інші ідентифікаційні відомості зазначені в сертифікаті ключа".* Щоб дати такий підтверджувальний підпис, поручитель зобов'язаний упевнитися в наступному (у наведеній послідовності): ключ індивідуальний і унікальний. Для цього власник ключа повинен повідомити його цифровий відбиток; зазначений у сертифікаті ключа людина (ім'я, фото) є тої, за кого себе видає. Звичайно із цією метою перевіряють персональні документи державного зразка або інше надійне посвідчення особи з фотографією; власник відкритого ключа має відповідний закритий ключ. Якщо він може розшифрувати зашифрований текст даним відкритим ключем і згенерувати цифровий підпис, що звіряється даним відкритим ключем, значить він володіє й відповідним закритим ключем; у сертифікаті зазначені приналежному власникові ключа контактні координати.

Він повинен довести, що має повний доступ до цих координат для одержання й відправлення повідомлень.

Але навіть якщо поручитель завірив своїм підписом сертифікат і ключ, маючи на увазі саме таке значення своєго підтверджуючого підпису, однаково ряд питань залишаються відкритими: яким документом власник ключа засвідчив свою особистість, чи надійний цей документ, чи було посвідчення особи справжнім, чи не був закрит ий ключ скомпрометований і викрадений? Ці питання впритул підводять нас до критерію довіри в середовищі асиметричних криптосистем і до розподіленої моделі довіри PGP *Web of Trust.*

Моделі довіри в криптосистемах з відкритим ключем діляться на два великих види: централізовані й розподілені. У централізованих, або ієрархічних, моделях всі користувачі системи покладаються на довіру до одного кореневого джерела, що підтверджує вірогідність всіх відкритих ключів. Така модель звичайно застосовується в корпоративному середовищі (де єдине джерело довіри обов'язкове) і в системах центрів, що засвідчують, на базі стандарту X.509, хоча бувають і виключення, наприклад, центр, що засвідчує, Thawte Consulting, що реалізує, як схему з єдиним сервером-зберігачем ключів, так і розподілену модель PGP, також називану мережею довіри.

У такій системі немає єдиного джерела сертифікації, навпроти, кожний користувач самостійно вирішує, кому він довіряє, а кому не довіряє в посвідченні інших відкритих ключів, створюючи тим самим особисту мережу поручителів. Такий підхід забезпечує гнучкість і стійкість системи до будь-якого зловмисного впливу: можна вплинути на один вузол розподіленої системи, але тисячі інших вузлів збережуть повну надійність.

Загальна проблема всієї асиммтеричной криптографії – складність перевірки автентичності відкритих ключів. Непросто з достатньою точністю визначити, що конкретний відкритий ключ є справжнім і належить передбачуваному власникові, і ще сутужніше це в середовищі PGP, де немає єдиного джерела сертифікації ключів, як в умовах інфраструктур PKI (Personal

Key Infrastructure). З іншого боку, розподілена природа моделі довіри PGP має й свої переваги.

Наведена в PGP схема досить складна й багата на несполучені ланцюжки сертифікації. Несполученими ланцюжками називаються такі, які не мають загальних ланок поручителів. Чим більше таких паралельних шляхів поручництв, тим менше ймовірність, що підозрілий ключ недостовірний: малоймовірно, щоб відразу кілька людей у такому випадку поручилися за його дійсність.

Щоб мати можливість знаходити такі комплексні ланцюжки, дуже важливо, щоб користувачі перевіряли дійсність ключів своїх кореспондентів, завіряли їхніми підписами й обновляли на сервері. В остаточному підсумку це буде благом для всіх. Такі взаємні поручництва утворять свого роду мережа, саме тому модель довіри PGP називається Web of Trust – Мережа довіри.

Усякий відкритий ключ на своєму зв'язуванні ви можете наділити деяким ступенем довіри в сертифікації інших ключів. Це значить, що якщо до вас у руки потрапить ключ *Б*, підписаний ключем А, що підтверджує підпису якого ви цілком довіряєте, ключ *Б* буде розцінений споконвічно достовірним, рятуючи вас від необхідності перевіряти його дійсність самостійно.

Інтегральним показником критеріїв довіри є ранг ключа в Мережі довіри, заснований на індексі MSD. Всім цим процедура наділення довірою істотно відрізняється від сертифікації ключа, коли ви повинні оцінити тільки його взаємозв'язок з передбачуваним власником, але не особистісні якості власника ключа.

Розглядаючи схему несполучених ланцюжків, намагаючись визначити дійсність підозрілого ключа, обов'язково переконаєтесь, що в складі проміжних ланок є хоча б кілька людей, яким ви довіряєте в сертифікації ключів. Якщо ланцюжок сертифікації не містить довірених поручителів, ви не повинні покладатися на неї як на оцінний критерій.

Чим більше взаємних перехресних підписів, що сертифікують, буде акумульовано в Мережі довіри, тим коротше почнуть ставати ланцюжки

сертифікації, і тем вище стане загальна переконаність у дійсності всякого ключа. Ключі, надійно зв'язані безліччю коротких ланцюжків, що засвідчують, називаються міцним набором. Кількість цих ключів у цей час становить приблизно 25 тисяч, і саме вони утворять стрижень і ядро всієї Мережі довіри, або зв'язкового набору – ключів, що мають хоча б один ланцюжок сертифікації від міцного набору. Зв'язний набір вичерпується приблизно 70 тисячами ключів, при цьому в названі 70 тисяч входять 25 тисяч ключів міцного набору. Ключі зі зв'язного набору, що не входять у міцний набір, називають периферійним набором (порядку 45 тисяч).

На суспільних серверах Інтернету зберігається біля двох мільйонів відкритих ключів, які утворять "міцні зв'язування". Більшість із них не має підписів, що сертифікують; деякі мають підпис, але не від ключів, що входять у мережу довіри банку. Такі невеликі групи взаємопідписаних ключів і ключі, що не мають підтверджувальних підписів, називаються ізольованим набором. Вони не враховуються в статистику мережі довіри банку, і визначити їхню дійсність по методу аналізу ланцюжків сертифікації неможливо. Якщо ж будь-який власник ключа зі зв'язного набору (клієнт банку) перевірить надійність одного з ізольованих ключів і підпише його, що підтверджує підпис виявиться сполучною ланкою, що веде до серця міцного набору, і ця група ізольованих ключів відразу увіллється в мережу довіри.

Сервери, або депозитарії, ключів OpenPGP, – це відкриті бази даних, прості сховища сертифікатів. Щоб спростити клієнтським частинам банківських мереж завдання знаходження відкритого ключа, можно завантажити його на сервер. Аналогічно, коли виникне потреба відправити на банківський сервер зашифрований лист, просто користуються формою пошуку для знаходження його відкритого ключа в базі.

Крім завантаження й пошуку ключів, зведена статистика мережі довіри PGP Web of Trust, а також базовані на ній спеціальні механізми відстеження шляхів сертифікації ключів і оцінки "авторитетності" і "ваги" будь-якого ключа в

системі Web of Trust. Механізм відстеження шляхи сертифікації дозволяє встановити ланцюжок підписань від ключа банківського сервера до ключа клієнтської частини банківської мережі, виявляючи всі проміжні ланки поручителів. Це один з допоміжних методів визначення дійсності й вірогідності конкретного ключа. Аналіз і оцінка положення ключа в системі Web of Trust також дозволяє визначити, як давно ключ циркулює в банківській мережі, наскільки вагомим можна вважати його підпис, що сертифікує, і т.д.

3.2 Розробка структурної схеми

Розглянемо основні положення які реалізовані в даній роботі – структурна схема роботи системи зображена на рисунку 3.4.

До складу системи з відкритим розподілом ключів входять центр сертифікації, формування й розподілу ключів (ЦСФРК), сервери розподіленої обробки й користувальницькі пристрої.

Основними завданнями ЦСФРК є підключення користувальницьких пристроїв і серверів до системи захисту, їхня сертифікація, формування й розподіл закритих і відкритих ключів між користувальницькими пристроями й серверами розподіленої обробки даних.

Схема автоматизованого централізованого управління ключами

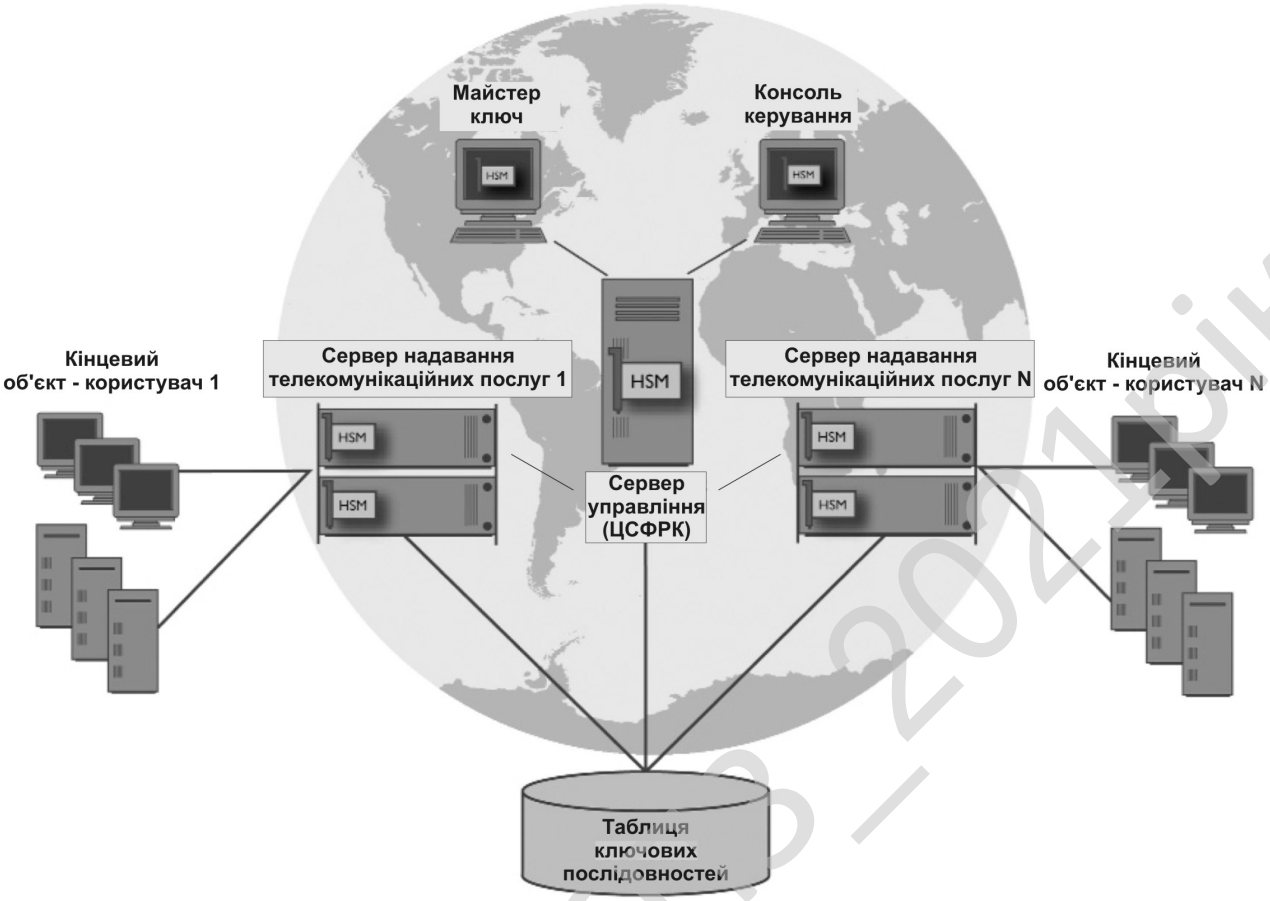


Рисунок 3.4 – Структурна схема роботи системи

3.3 Розробка функціональної схеми

У ЦСФРК генерується й зберігається головний ключ системи (майстер-ключ), що являє собою випадково заповнену кодами таблицю розміром n x n.

На основі таблиці головного секретного ключа в ЦСФРК шляхом випадкової перестановки стовпців і рядків формується безліч різних таблиць початкових секретних ключів для користувачів. При цьому кожній отриманій таблиці початкового секретного ключа ставиться у відповідність застосована перестановка стовпців і рядків таблиці головного секретного ключа. Потім для кожної таблиці початкового секретного ключа шляхом випадкових перестановок його стовпців і рядків створюються таблиці внутрішнього секретного ключа й

зовнішнього секретного ключа. Кожній отриманій таблиці ставляться у відповідність використані випадкові перестановки стовпців і рядків таблиці початкового секретного ключа, таким чином реалізована функціональна схема розробленого програмного забезпечення (рисунок 3.5).

Отримані таблиці початкового ключа й випадкові перестановки стовпців і рядків для формування таблиць внутрішнього секретного ключа, а також зовнішнього секретного ключа застосовуються при підготовці носіїв для сертифікованих користувачів. Формується носій даних – смарт-карта, копія якої зберігається в центрі сертифікації. Вона містить повну таблицю початкового ключа, а також набір секретних ключів-перестановок для таблиць внутрішнього й зовнішнього ключів користувача. Також у смарт-карту записується PIN-код і значення геш-функції пароля даного користувача.

Щоб одержати систему ключів, користувач вводить інформацію зі смарт-карти; при доступі до функцій системи захисту по команді користувача в користувальницькому пристрої на основі таблиці початкового ключа й секретних перестановок, введених зі смарт-карти, виробляється формування таблиць внутрішнього секретного ключа, а потім таблиці зовнішнього секретного ключа. Аналогічні процедури виконуються й на сервері. При цьому таблиця зовнішнього секретного ключа застосовується для заповнення базової таблиці одноразових ключів ОБК, що служить для організації зовнішнього зашифрованого зв'язку з іншими користувачами або серверами банківської мережі.

Рисунок 3.5 – Функціональна схема системи

Після завершення процесу формування ключових таблиць користувач може звернутися із запитом до ЦСФРК для організації закритого зв'язку з необхідним сервером розподіленої обробки або з іншим користувачем. Цьому повинна передувати відповідна домовленість, досягнута по відкритому зв'язку. Відповідно до даного запиту ЦСФРК забезпечує генерацію й розподіл відкритих ключів між користувачами.

Формування відкритих ключів засновано на застосуванні односпрямованої функції, що використовує логічний вивід на перестановках. У ЦСФРК зберігаються всі ключі-перестановки стовпців і рядків, що дозволяють із таблиці головного ключа сформувати для кожного користувача таблиці початкових, внутрішніх і зовнішнього секретних ключів. Після завантаження системи всі ці таблиці, включаючи таблицю зовнішніх секретних ключів, для різних користувачів будуть асиметричні. З метою організації закритого зв'язку між користувачами А и В необхідно привести їхні таблиці зовнішніх секретних ключів у симетричний стан. Це досягається завдяки наявності в ЦСФРК всіх зазначених функціонально зв'язаних секретних перестановок таблиць. При цьому за допомогою логічного виводу на послідовності транзитивного зв'язку між рядками таблиць секретних перестановок визначаються відносні перестановки для користувачів А и В, які дозволяють привести таблиці зовнішніх секретних ключів в ідентичний стан. Зазначені відносні перестановки є відкритими ключами; з їхньою допомогою користувачі можуть перевести таблиці зовнішніх секретних ключів в ідентичний стан для організації симетричного закритого зв'язку.

Відзначимо, що функція формування відкритих ключів з вико ристанням відносної перестановки є односпрямованою для будь-якого користувача системи. На основі отриманих відкритих ключів у користувальницькому пристрої А и сервері В розподіленої обробки створюють таблиці симетричних зовнішніх секретних ключів. Ці таблиці записуються в одноразовий багатоалфавітний кодер (декодер) з метою встановлення закритого симетричного зв'язку між користувачами. При цьому в процесі шифрування на основі генерації випадкових перестановок таблиць зовнішніх секретних ключів реалізується описаний режим одноразової системи з відкритою передачею ключів, що забезпечує необхідний гарантований рівень захисту інформації. Після завершення сеансу закритого зв'язку ЦСФРК посилає користувачам А и В відкриті ключі перестановки для генерації нових асиметричних таблиць зовнішніх секретних ключів.

Вище представлена послідовність дій оброблена для графічного представлення та винесена на функціональну схему алгоритму роботи програми.

Основні кроки виконання алгоритму:

1. Реєстрація користувача.
2. Генерація запиту користувача.
3. Верифікація запиту користувача.
4. Випуск сертифіката.
5. Поширення.
6. Зберігання.
7. Використання.
8. Припинення дії.
9. Відновлення.
10. Відкликання.

3.4 Розробка діаграми процесів

Важливим критерієм при розробці програмного забезпечення це є грамотна розробка структури роботи системи. На діаграмі процесів зображеній на рисунку 3.6 можна точно зрозуміти як працює і взаємодіє розроблене програмне забезпечення. в цілому.

Починається і закінчується програма в першому блоці це є основною крапкою відрахунку діаграми. При переміщенні по стрілках можна побачити загальну схему взаємодії блоків і їх входження один в одного.

Як можна побачити з основного блоку програми управляння переходить спочатку до інтерфейсу операційної системи та потім до вікна розробленого програмного забезпечення. Далі через модуль захисту програмного забезпечення формується та потім зберігається головний ключ системи. Після цього через той самий блок формується за допомогою головного ключа системи таблиці

секретного ключа та таблиці внутрішнього та зовнішнього секретного ключа для користувача.

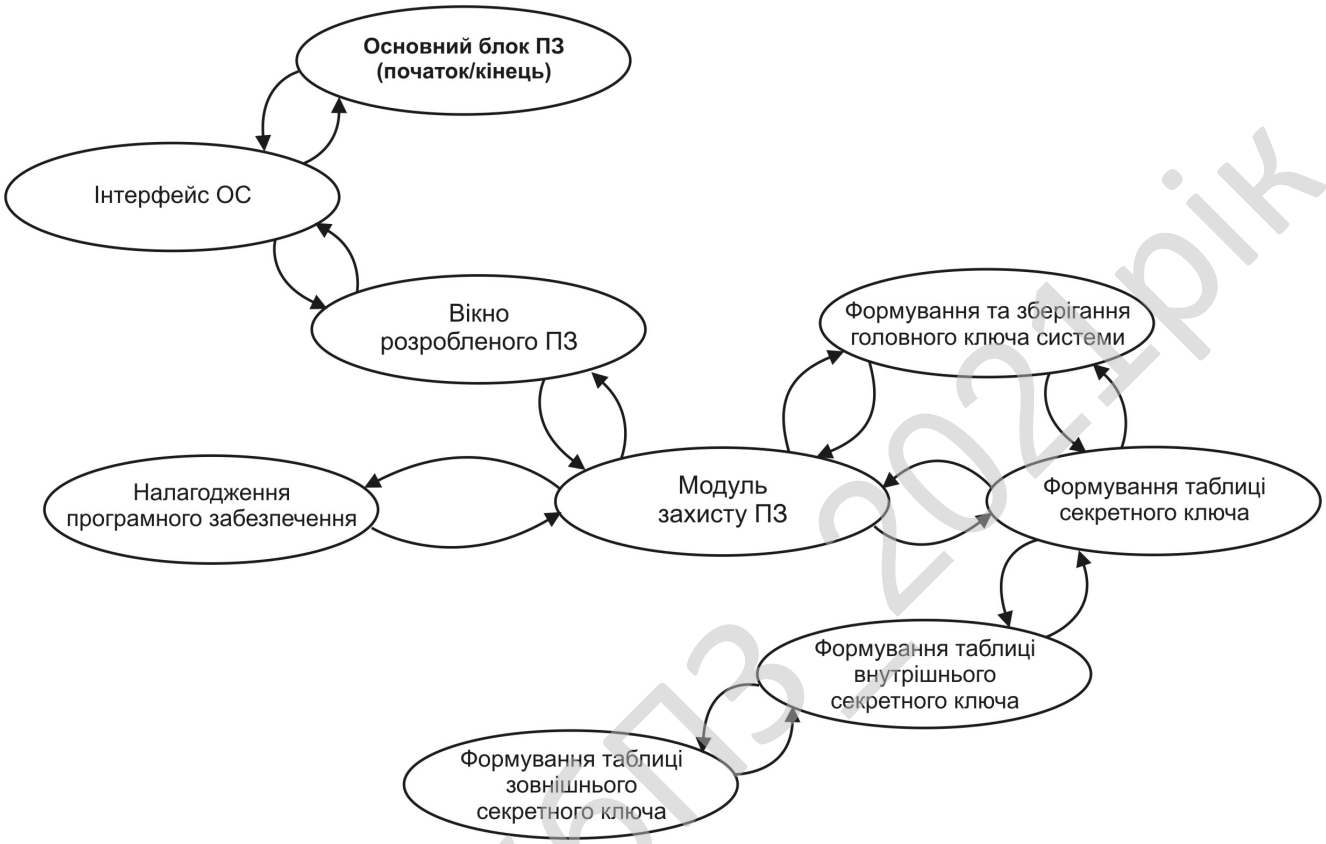


Рисунок 3.6 – Схема взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунках 4.1, 4.2, 4.3 зображена розроблена блок-схема програми. Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно є розробка блок-схем.

З основного алгоритму програми винесені дві основних підпрограми це підпрограма генерації та формування таблиці секретних ключів (рисунок 4.2) та підпрограма відхилення запиту генерації та формування таблиці секретних ключів (рисунок 4.3).

Виділимо у розробленій блок-схемі основні блоки (рисунок 4.1).

Початок роботи та перевірки:

– Виділення ресурсів програми.

– Підключення бази даних користувачів – закодований текстовий файл осіб шо допускаються до запиту секретних ключів (кодових комбінацій).

– Перевірка цілісності бази даних і доступу до ресурсів ОС.

– Перевірка успішності здійснення цілісності бази даних і доступу до ресурсів ОС.

– Виведення головного вікна ПЗ на екран.

Робота програми:

– Очікування запиту від програмного забезпечення.

– Прийом запиту.

– Розбір пакету запиту.

– Перевірка ID користувача.

– Перевірка дозволу та вхід в підпрограми.

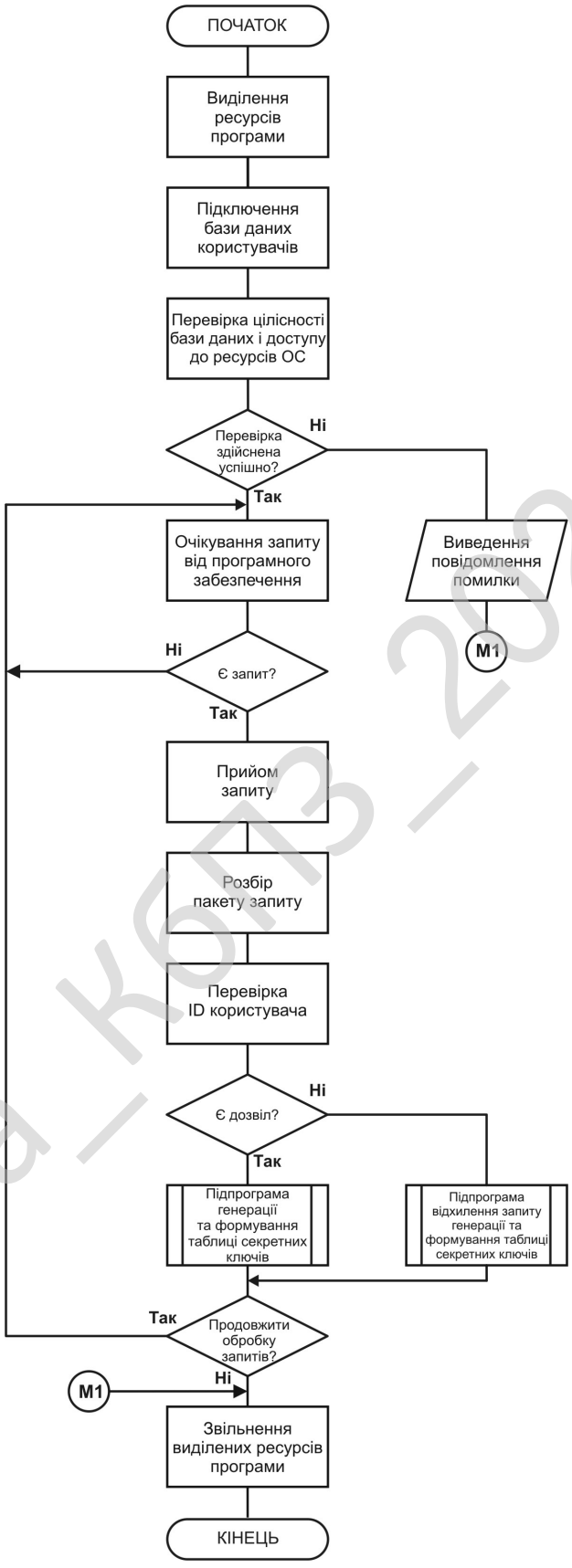


Рисунок 4.1 – Блок-схема основної програми

Робота підпрограми – генерації та формування таблиці секретних ключів (рисунок 4.2).

– Формування таблиці секретного ключа.

– Встановлення параметрів довжини секретного ключа.

– Встановлення параметрів блокових характеристик секретного ключа.

– Налагодження модуля створення кодової комбінації.

– Генерація таблиці секретного ключа.

– Генерація таблиці внутрішнього секретного ключа.

– Генерація таблиці зовнішнього секретного ключа.

– Формування носіїв даних та відправлення адресату.

– Перевірка відправлення носія даних.

– Запис в журнал роботи програми.

Робота підпрограми – відхилення запиту генерації та формування таблиці секретних ключів (рисунок 4.3).

– Формування повідомлення відмови запиту секретного ключа.

– Заповнення текстових полів з причиною відхилення запиту.

– Заповнення полів службовою інформацією.

– Запис у БД даних про запит оновлення.

– Заповнення полів пакету службовою інформацією.

– Відправлення адресату інформації.

– Інформація відправлена.

– Повернення позитивного результату.

– Запис в журнал роботи програми.

Завершення роботи:

– Вихід з підпрограм.

– Запит на продовження обробки запитів.

– Звільнення виділених ресурсів програми.

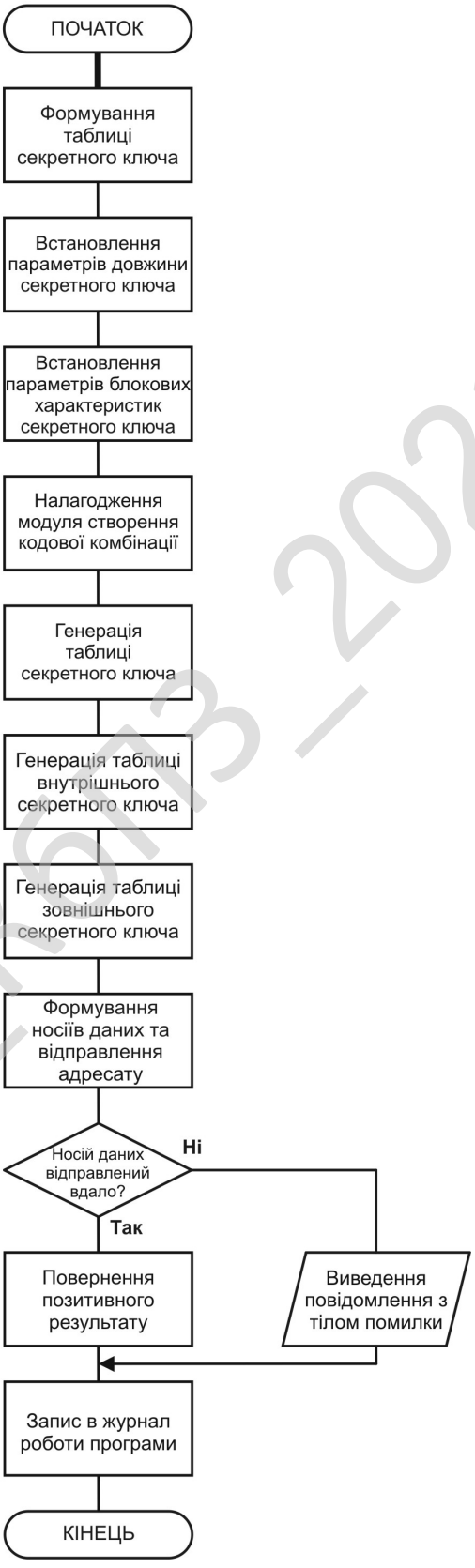


Рисунок 4.2 – Блок-схема підпрограми генерації та формування таблиці секретних ключів

Підпрограма відхилення запиту генерації та формування таблиці секретних ключів

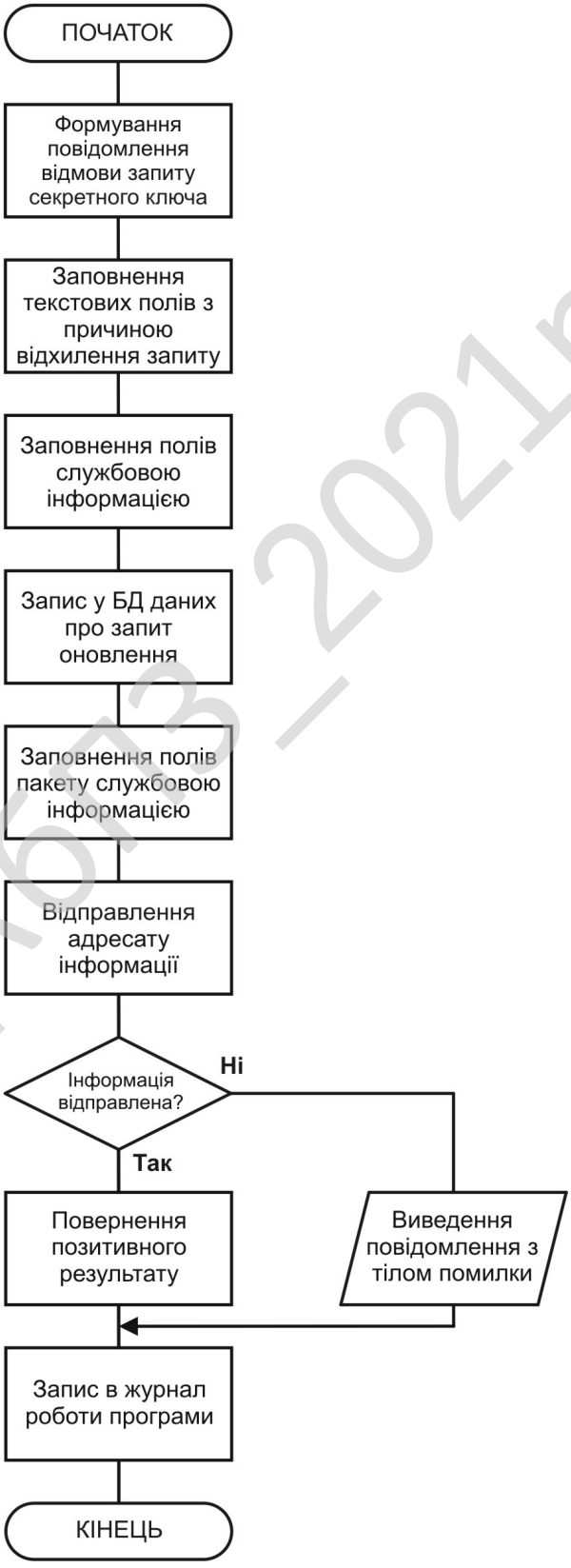


Рисунок 4.3 – Блок-схема підпрограми відхилення запиту генерації та формування таблиці секретних ключів

Опис розроблено програмного забезпечення

Метод ПЗ полягає у винесенні модуля генерації секретних ключових послідовностей з розробленого програмного забезпечення в окремий мережний модуль, що дозволяє швидко реагувати на підміну ключів доступу.

Розроблене програмне забезпечення дозволяє створювати і контролювати секретні ключові послідовності.

Для впровадження програмного забезпечення, фірма впроваджує систему захисту з врахуванням модуля генерації секретних ключів (послідовностей) та кілька алгоритмів зміни ключів і їх характеристик.

Розроблений у випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти програмний пакет, забезпечує універсальну систему захисту за допомогою розповсюдження ключів гарантованої стійкості.

Програма призначена для дрібних і середніх фірм розроблювачів програмного забезпечення, банківських мереж, дозволяючи автоматизувати процес заміни ключових комбінацій, є ідеальним рішенням при розробці великої кількості програмних продуктів.

Не юридичні особи (фізичні) такі як: наукові співробітники, молодші і середні менеджери, які використовують мови програмування для особистих специфічних цілей також знайдуть для себе ідеальний спосіб створення ключових комбінацій для своїх розробок.

Також програма буде корисна студентам, які розробляють безкоштовні або напів-безкоштовні програмні продукти, з авторським правом розроблювача.

Для повного представлення розробленого програмного забезпечення, розглянемо головний складальний файл програми.

```
// авторське право ПЗ
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
// назва програми
program Project_MASTER_KEYS;
uses
//модулі, що підключаються
// підключення бібліотек та компонентів ті ін.
```

```
  Forms,
  SysUtils,
//підключення вікна MAIN
  frmMAIN in 'frmMAIN.pas' {Form1},
//підключення вікна Settings
  frmSettings in 'frmSettings.pas' {Form2},
//підключення вікна KEYGEN
  frmKEYGEN in 'frmKEYGEN.pas' {Form3},
//підключення вікна LOG
  frmLOG in 'frmLOG.pas' {Form4},
  frmKEYSTAT in 'frmKEYSTAT.pas' {Form5},
//підключення вікна KEYSTAT
  frmSPLASH in 'frmSPLASH.pas' {U_Form_Splash},
//підключення вікна About
  frmAbout in 'frmAbout.pas' {AboutBox};
//підключення вікна Unit1
frmUnit1 in 'frmUnit.pas' {Unit6};

// файл ресурсів
{$R *.res}
begin
try
// Створення та виведення на екран вікна заставки

U_Form_Splash:=TU_Form_Splash.Create(Application);
    U_Form_Splash.Show;
    U_Form_Splash.Update;
    U_Form_Splash.Label2.Caption:='1';
    U_Form_Splash.Update;
    U_Form_Splash.Label2.Caption:='2';
    U_Form_Splash.Update;

  Application.HintPause:=200;//ms
  Application.HintHidePause:=7000;//ms
  Application.HintShortPause:=25;//ms
// Ініціалізація програмного забезпечення
  Application.Initialize;
// Створення головного вікна програми
  Application.CreateForm(TForm1, Form1);
// Створення вікна Form2
  Application.CreateForm(TForm2, Form2);
// Створення вікна Form3
```

```
  Application.CreateForm(TForm3, Form3);
// Створення вікна Form4
  Application.CreateForm(TForm4, Form4);
// Створення вікна Form5
  Application.CreateForm(TForm5, Form5);
// Створення вікна Form6
  Application.CreateForm(Tform6, Form6);
// Створення вікна AboutBox
  Application.CreateForm(TAboutBox, AboutBox);
  finally
//звільнення вікна заставки
    U_Form_Splash.free;
end;
// запуск програмного забезпечення
  Application.Run;
end.
```

Реалізація методу обробки запитів користувачів

Для реалізації задачі обробки запитів від кінцевих користувачів (для розподілення секретних ключів) необхідно використовувати мережні компоненти.

Розглянемо використаний компонент TNMHTTP (NetMasters HTTP), який знаходиться на вкладці FastNet палітри компонентів DELPHI та розглянемо коди підпрограм які дозволяють реалізувати поставлені задачі (винесені у різні програми для зменшення коду програми):

```
{... ПОЧАТОК йде заголовок файлу і визначення форми (TForm1) і її екземпляра
Form1. У формі е кнопка TButton і одне поле TEdit. При натисненні на кнопку
викликається обробник події OnClick – Button1Click. }

procedure Button1Click(Sender: TObject);
begin
 {Намагаємося отримати заголовок}
 NMHTTP1.Head(Edit1.Text);
 {Якщо адрес невірний, то тут вискочить помилка}
end;
```

```
…
//Викачування вказаної URL в папку розробленої
//програми
```

```
{... ПОЧАТОК Йде заголовок файлу і визначення форми TForm1 і її екземпляра Form1.
У формі є кнопка TButton і три поля TEdit. При натисненні на кнопку викликається
обробник події OnClick – Button1Click. Перед цим в перший TEdit введен  адресу
URL, в другій – ім'я файлу для заголовка, а в третій – ім'я файлу для тіла html}

procedure Button1Click(Sender: TObject);
begin
    {Намагаємося отримати http-документ}
    {Результат треба записати у файли}
    NMHTTP1.InputFileMode := True;
    { указуємо в які саме файли}
    NMHTTP1.Header := Edit2.Text;
    NMHTTP1.Body := Edit3.Text;
    NMHTTP1.Get(Edit1.Text);
end;
```

….

Розгляд викачування відразу декількох URL одночасно необхідно розглянути більш детально. У DELPHI дуже легко створювати окремі, підлеглі процеси за допомогою базового класу TThread.

Розглянемо приклад реалізований в програмному забезпеченні – одночасне викачування вказаних URL в заданий каталог.

```
….
{... ПОЧАТОК Йде заголовок файлу і визначення форми TForm1 і її екземпляра
Form1,опис класу окремого процесу}
Type
    THTTPThread = class(TThread)
    private
{Для кожного процесу – створюємо свій компонент TNMHTTP}
        FHTTP: TNMHTTP;
    Protected
{Execute викликається при запуску процесу; override – замінюємо існуючу
процедуру базового класу TThread}
    procedure Execute; override;
    {DoWork – створена нами функція, виконання якої синхронізується в Execute}
    procedure DoWork;
    public
    {URL – створена нами рядок, вказуючий процесу, який URL йому потрібно
викачати}
        URL: string;
    end;
```

```
{У форму потрібно помістити три кнопки TButton, одне поле TEdit і один список TListBox. При натисненні на кнопку Button1 викликається обробник події OnClick – Button1Click. Перед цим в TEdit потрібно ввести шлях до каталога, в якому зберігатимуться викачані файли, а ListBox1 потрібно заповнити списком URL-ов для викачування (за допомогою кнопок Add (Button2) і Delete (Button3))}
    procedure TForm1.Button3Click(Sender: TObject);
    begin
        {Видалення виділеного URL із списку}
        if ListBox1.ItemIndex >= 0 then
            ListBox1.Items.Delete(ListBox1.ItemIndex);
    end;

    procedure TForm1.Button2Click(Sender: TObject);
        var s: string;
    begin
        {Додавання URL в список}
        s := InputBox('Input','Введення URL:','');
        if s <> '' then ListBox1.Items.Add(s);
    end;

    procedure TForm1.Button1Click(Sender: TObject);
      var i: Integer;
    begin
        {Перевірка на існування каталога}
        if Length(Edit1.Text)> 0 then
            if not DirectoryExists(Edit1.Text) then
                MkDir(Edit1.Text);
     {Далі йде створення для кожного URL в списку свого процесу}
        for i := 0 to ListBox1.Items.Count-1 do begin
            with THTTPThread.Create(True) do begin
    {Створюємо припинену задачу, указуємо їй її URL і запускаємо її}
                URL := ListBox1.Items[i];
                Resume;
            end;
        end;
    end;
    {Оператори процесу THTTPThread}
    procedure THTTPThread.Execute;
    begin
    {Робимо так, щоб кожний процес виконувався одночасно з іншими (синхронізація)}
        Synchronize(DoWork);
```

```
end;
procedure THTTPThread.DoWork;
    var i: Integer;
begin
    {Створюємо компонент TNMHTTP}
    FHTTP := TNMHTTP.Create(Form1);
    {Результат записуємо у файли}
    FHTTP.InputFileMode := True;
    {Підбираємо імена для файлів}
    i:= 1;
    while FileExists(Form1.Edit1.Text+'\page'+IntToStr(i)+'.htm') do
        Inc(i);
    {Указуємо, в які саме файли класти результат}
    FHTTP.Body := Form1.Edit1.Text+'\body'+IntToStr(i)+'.htm';
    FHTTP.Header := Form1.Edit1.Text+'\header'+IntToStr(i)+'.txt';
    {Намагаємося послати запит}
    FHTTP.Get(URL);
    FHTTP.Free;
end;
```

Реалізація програмного опитування користувачів

При роботі розробленого програмного продукту в мережі в деяких випадках необхідно знати поточний стан як локального, так і видалених хостів (чи має локальний хост в даний момент можливість виходу в мережу Інтеренет, чи доступний якийсь видалений хост і т.д.)

Загальновідомо, що для вказаної мети використовується утиліта ping. Принцип роботи ping-а заснований на використовуванні протоколу ICMP – Internet Control Message Protocol (протокол керівників, або контрольних, повідомлень).

За допомогою ICMP хост в мережі обмінюються різною службовою інформацією (інформацією про зміну маршруту, зменшення швидкості передачі, неприступність якої-небудь адреси і т.д.)

В основі протоколу ICMP лежить поняття повідомлень. Повідомлення ICMP протоколу, як правило, оповіщають про помилки, що виникають при обробці датаграмм. ICMP використовує основні властивості протоколу IP,

неначебто він був протоколом більш високого рівня. На самій же справі ICMP є складовою частиною IP.

Одним з типів повідомлень протоколу є т.н. "луна-запит". Отримавши "луна-запит" хост зобов'язаний відповісти тому, що послав "луна-відповіддю".

По суті, "луна-запит" та "луна-відповідь" відрізняються лише адресами відправника і одержувача і кодом типу повідомлення (тип 8 – "лунає-запит, тип 0 – "лунає-відповідь).

Реалізації утиліти ping на різних платформах істотно відрізняються. Так, в Ос UNIX використовуються RAW sockets (необроблені, "сирі" сокети), а в Ос Windows всіх версій – т.з. ICMP API.

На практиці, у всіх версіях Windows є і використовується бібліотека icmp.dll. Отже, на даний момент, можна її використовувати що було зроблено у випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

Визначення типів і прототипи функцій для Microsoft icmp.dll:

```
type
// Інформація заголовка IP (Наповнення)
ip_option_information = packed record
Ttl : byte; // Час життя (використовується traceroute-ом)
Tos : byte; // Тип обслуговування, звичайно 0
Flags : byte; // Прапори заголовка IP, звичайно 0
// Розмір даних в заголовку, звичайно 0, максимум 40
OptionsSize : byte;
OptionsData : Pointer; // Покажчик на дані
end;
icmp_echo_reply = packed record
        Address : u_long; // Адреса відповідаючого
        Status : u_long; // IP_STATUS
        // Час між луна-запитом і луна-відповіддю
        RTTime : u_long; // в мілісекундах
        DataSize : u_short; // Розмір повернених даних
        Reserved : u_short; // Зарезервовано
        Data : Pointer; // Покажчик на повернені дані
        // Інформація із заголовка IP
        Options : ip_option_information;
    end;
```

```
            PIPINFO = ^ip_option_information;
            PVOID = Pointer;
        function IcmpCreateFile():THandle;stdcall;external 'ICMP.DLL';
        function IcmpCloseHandle(IcmpHandle : THandle): BOOL; stdcall; external
'ICMP.DLL'  name 'IcmpCloseHandle';
        function IcmpSendEcho(IcmpHandle : THandle;
        // Адреса одержувача (в мережному порядку)
        DestAddress : u_long;
        RequestData : PVOID; // Покажчик на посилані дані
        RequestSize : Word; // Розмір посиланих даних
        RequestOptns : PIPINFO;// Покажчик на посилану структуру
        //ip_option_information (може бути nil)
        //Покажчик на буфер, що містить відповіді.
        ReplyBuffer:PVOID;
        ReplySize : DWORD; //Розмір буфера відповідей
        Timeout : DWORD //Час очікування відповіді в мілісекундах
        ) : DWORD; stdcall; external 'ICMP.DLL' name 'IcmpSendEcho';
```

Функція IcmpSendEcho() посилає ICMP луна-запит за заданою IP адресою і поміщає всі відповіді, отримані за час заданого таймауту, в буфер відповідей (ReplyBuffer). Перед використовуванням функцій Icmp.dll необхідно викликати функцію WSAStartup() з Winsock.

Якщо цього не зробити, то перший же виклик функції IcmpSendEcho() приведе до помилки 10091 (WSASYSNOTREADY).

Слід помітити, що відповіді, поміщені в буфер, необов'язково будуть повідомленнями луна-відповідь. Можливо, що у відповідь на луна-запит прийдуть повідомлення ICMP про виниклі помилки.

Природно, ці повідомлення так само будуть поміщені в ReplyBuffer.

Розмір буфера відповідей повинен бути достатнім для прийому хоча б однієї відповіді, будь то луна-відповідь або повідомлення про помилку. Звідси, вказаний розмір складається з розміру самої структури icmp_echo_reply плюс Max(RequestSize, 8), оскільки повідомлення ICMP про помилку складає 8 байт.

Функція IcmpSendEcho() повертає кількість відповідей (або структур icmp_echo_reply) в масиві ReplyBuffer. Якщо функція повернула 0, то для більш

детальної діагностики виниклої помилки використовується функція GetLastError().

Нижче приведений базовий фрагмент коду який реалізован у програмі, необхідний для однократного посилання луна-запиту:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    hIP : THandle;
    pingBuffer : array [0..31] Char;
    pIpe : ^icmp_echo_reply;
    pHostEn : PHostEnt;
    wVersionRequested : WORD;
    lwsaData : WSAData;
    error : DWORD;
    destAddress : In_Addr;
begin
    // Створюємо handle
    hIP := IcmpCreateFile();
GetMem( pIpe, sizeof(icmp_echo_reply)+ sizeof(pingBuffer));
    pIpe.Data := @pingBuffer;
    pIpe.DataSize := sizeof(pingBuffer);
    wVersionRequested := MakeWord(1,1);
    error := WSAStartup(wVersionRequested,lwsaData);
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Error in call to '+'WSAStartup().');
Memo1.Lines.Add('Error code: '+IntToStr(error));
    Exit;
    end;
       pHostEn := gethostbyname;
    error := GetLastError();
    if (error <> 0) then
    begin
Memo1.SetTextBuf('Error in call to'+'gethostbyname().');
Memo1.Lines.Add('Error code: '+IntToStr(error));
    Exit;
    end;
destAddress := PInAddr(pHostEn^.h_addr_list^)^;
// Посилаємо ping-пакет
Memo1.Lines.Add('Pinging'+pHostEn^.h_name+' ['+
inet_ntoa(destAddress)+'] '+ ' with '+
```

```
            IntToStr(sizeof(pingBuffer))+
        'bytes data:');
        IcmpSendEcho(hIPdestAddress.S_addr, pingBuffer
sizeof(pingBuffer), Nil, pIpe sizeof(icmp_echo_reply)+ sizeof(pingBuffer), 5000);
        error := GetLastError();
            if (error <> 0) then
            begin
        Memo1.SetTextBuf('Error in call to '+'IcmpSendEcho()');
        Memo1.Lines.Add('Error code: '+IntToStr(error));
            Exit;
            end;
             // Дивимося деякі з даних, що повернулися
        Memo1.Lines.Add('Reply from '+ IntToStr(LoByte(LoWord(pIpe^.Address)))+'.'+
        IntToStr(HiByte(LoWord(pIpe^.Address)))+'.'+
        IntToStr(LoByte(HiWord(pIpe^.Address)))+'.'+
        IntToStr(HiByte(HiWord(pIpe^.Address))));
        Memo1.Lines.Add('Reply time:'+IntToStr(pIpe.RTTime)+' ms');
            IcmpCloseHandle(hIP);
            WSACleanup();
            FreeMem(pIpe);
        end;
```

Для роботи цього фрагмента коду необхідно:

– створити форму TForm1;

– включити в розділ Uses юнит WinSock;

– помістити на форму компонент Memo1:TMemo і кнопку Button1:TButton;

– в розділі Implementation помістити описи типів і функцій Icmp.dll вище);

– зіставити події OnClick кнопки приведену функцію.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох

великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення за модулем 2^{32}-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3) і 32 -бітовими підблоками ключу (k_0, k_1, k_2, k_3). Це спрощує реалізацію алгоритму на сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються за модулем 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція *f* виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються за модулем 4. Операція множення на кроці 1 виконується по модулі 2^{32}-1. Спеціальний випадок для даного

алгоритму: якщо другий операнд дорівнює 2^{32}-1, результат теж дорівнює 2^{32}-1. В алгоритмі використовуються наступні константи:

$$C = 2aaaaaaa,\ c_0 = 025f1cdb,\ c_1 = 2*c_0,\ c_2=2^3*c_0,\ c_3=2^7*c_0.$$

Константа C – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи c_0 є інші особливі характеристики. Константи c_1, c_2 і c_3 – зрушені версії c_0, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість c_i використовується c_i^{-1}. Значення c_0^{-1} = 0dad4694.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення системи централізованого розподілу ключів рекомендується для впровадження в будь-яких організаціях де гостро стає проблема розповсюдження ключів для систем захисту.

Разом з тим, при впровадженні системи необхідно вирішити ряд проблем, які дозволять розпочати промислову експлуатацію розробленої програмної моделі.

Це, перш за все, розробка організаційного та методичного забезпечення. Методичне забезпечення впровадження програми у експлуатацію включає слідуючи документи:

– інструкції по використанню програмного забезпечення;

– інструкції користувача, який буде працювати з програмним забезпеченням;

– правила по забезпечення системи безпеки при експлуатації програмного забезпечення;

– документи з описом методики вивчення методу послідовного аналізу;

– таблиці інформації.

Організаційне забезпечення впровадження програмної моделі включає в себе наступні документи:

– накази та розпорядження по регламентації взаємодії підрозділів, які приймають участь в експлуатації створеної програмного забезпечення;

– накази та розпорядження, які визначають функції підрозділів по експлуатації та обслуговуванню програмного забезпечення та технічного обладнання;

– затверджений перелік матеріально-технічного забезпечення підсистеми;

– графік регламентних робіт;

– журнали для регламентації роботи користувачів, обліку машинного часу, обліку проведених регламентних робіт та інші;

– штатний розклад обслуговуючого персоналу.

Організаційна система впровадження також передбачає розробку плану впровадження і організацію підготовки та навчання персоналу, налагодження системи на підприємстві, приймальню здавальні іспити, промислову експлуатацію.

План впровадження системи встановлює:

– обсяг фінансування проектних та налагоджувальних робіт;

– терміни розробки проектної документації по елементах системи, терміни постачання технічних заходів.

План впровадження також передбачає організацію підготовки кадрів, експлуатаційного персоналу і аналіз функціонування програмного забезпечення програмної моделі після виконання всіх робіт з її впровадження.

Крім конкретних термінів виконання робіт, у плані впровадження зазначені підрозділи, які відповідають за виконання робіт в повному обсязі із забезпеченням якісних показників.

Приклади розробленої програми приведені на рисунках 5.1, 5.2, 5.3, 5.4.

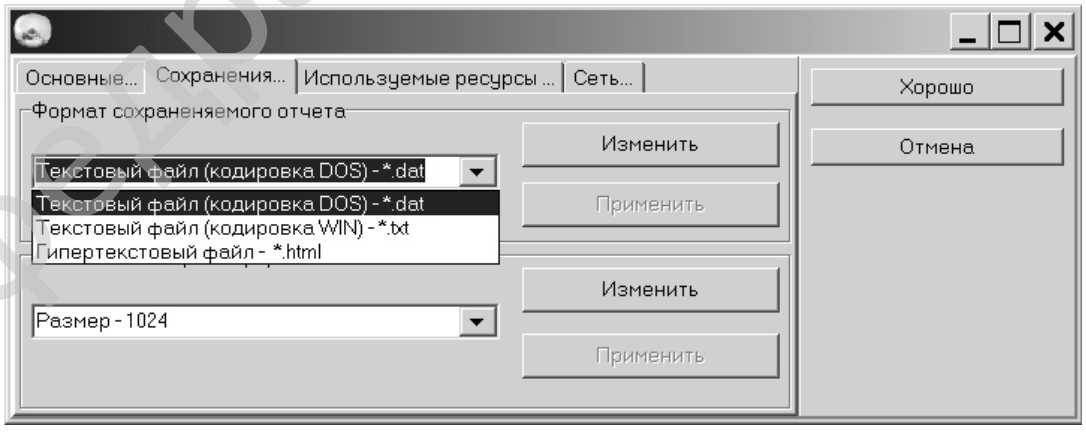


Рисунок 5.1 – Вікно налагодження параметрів програми, вибір файлу типу звіту

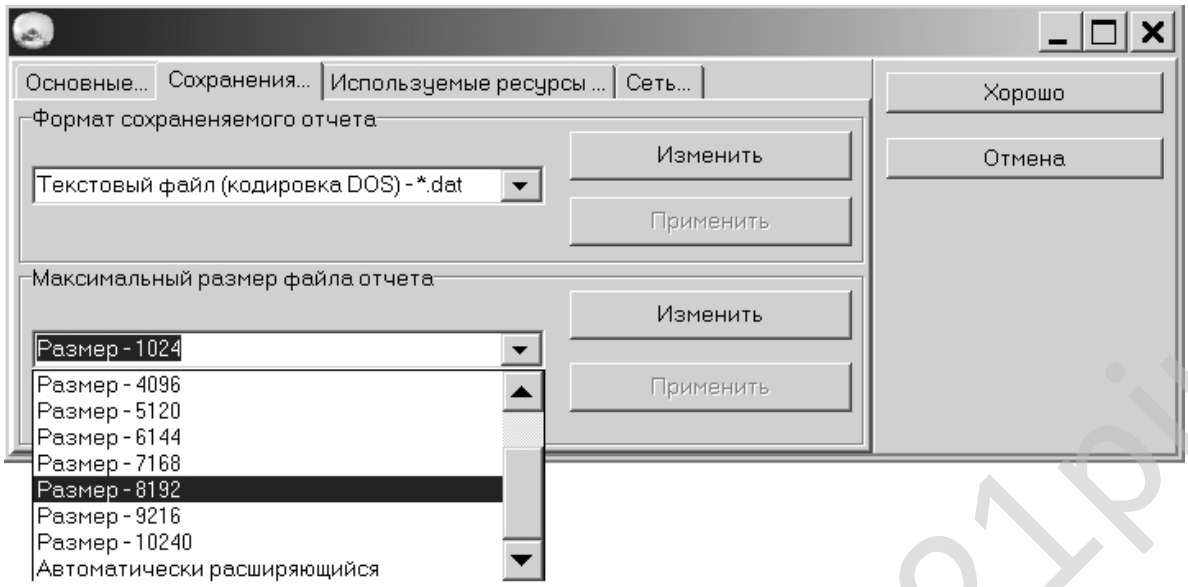


Рисунок 5.2 – Вікно налагодження параметрів програми, вибір максимального розміру файлу звіту

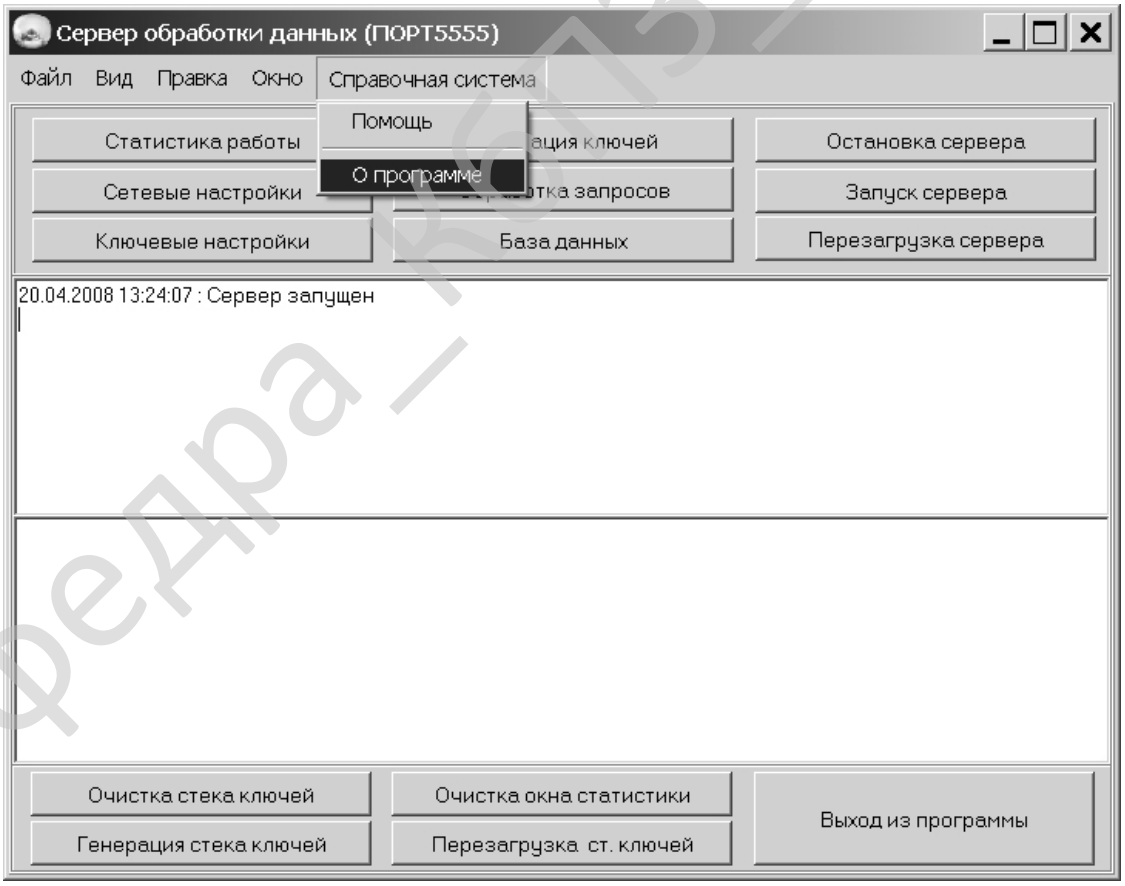


Рисунок 5.3 – Головне вікно програми

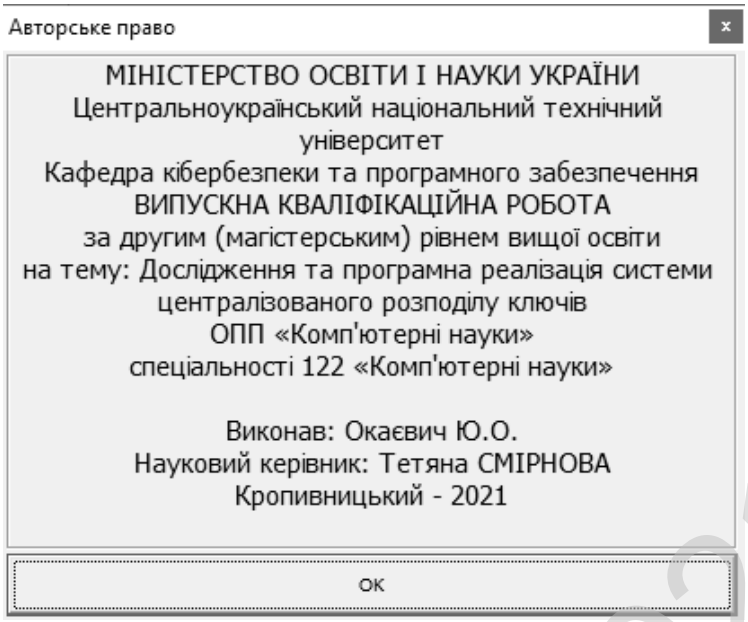


Рисунок 5.4 – Авторське вікно програми

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи централізованого розподілу ключів.

Метою розробки є дослідження та програмна реалізація системи централізованого розподілу ключів.

Об'єктом дослідження є процес централізованого розподілу ключів.

Предметом дослідження є методи централізованого розподілу ключів.

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

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

– Удосконалено метод централізованого розподілу ключів.

– Розроблено вітчизняний продукт централізованого розподілу ключів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи централізованого розподілу ключів.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	120 (варіант № 98)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	120000 (варіант № 98)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	37
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

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

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

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

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

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

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

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

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

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

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

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

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

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

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

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

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

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

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

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

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 567,5/(60·8) = 1,2 ставки.

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6000	21600
Інженер-системотехнік	0,25	6000	4500
Адміністратор мережі	0,5	6000	9000
Системний програміст	0,25	6000	4500
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6000	4500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 198300

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 401 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

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

$S_у$ – питома площа на одне робоче місце, $м^2$;

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

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

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

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

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

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Brain за 26.10.21 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i3 540 (3.067Ghz 4Mb_cache Clarkdale, 73W, socket1156) box	1750
Системна плата	MB MSI H55M-E33 s1156 mATX (H55M E33)	1200
Відеокарта	VC VTX Radeon HD6570 1GB GDDR3 128bit, 650 MHz/1334 MHz, PCI-E 2.1, DVI HDMI, VGA (VX6570 1GBK3-H)	750
Жорсткий диск	HDD: 320 Gb 7200 Serial ATA WD 16MB	1200
Оперативна пам’ять	DIMM 2048Mb DDR3 PC3-12800 Patrio 1600Mhz, CL9, (9-9-9-28), 1.5V, Reta (PSD32G16002H) 2 модуля	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silve body material – 0.6mm), 80mm fan (rear 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 401 \cdot 209/120 = 698 \text{ грн.}$$

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

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

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

$$З_д = 698 \cdot 10 \cdot 0{,}01 = 70 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ від суми основної та додаткової зарплати:

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

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

$$С_{оц} = 0{,}01 \cdot 37(698+70) = 288 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

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

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

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

Згідно виданих викладачем норм приймаємо одну пачку паперу на три місяці розробки (n_p=0,33). Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 105 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

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

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

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

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

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

$$З_{М2} = 120 \cdot 12 = 1440 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (105 + 1440 + 1702)/120 = 27 \text{ грн.}$$

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

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

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

$$О_п = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 120 прим.):

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

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

$$A_м = 190286 \cdot 3/(120 \cdot 12) = 396 \text{ грн.}$$

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

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

$$C_п = 698+70+288+105+27+105+396 = 1689 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	698
2. Додаткова зарплата виконавців	$З_д$	70
3. Відрахувння на соціальні потреби	$C_{оц}$	288
4. Загальногосподарські витрати	$Г_{осп}$	105
5. Витрати на матеріали	$З_М$	27
6. Освоєння нових операційних систем, мов програмування	$О_п$	105
7. Амортизація основних фондів	$A_м$	396
8. Повна собівартість програмного забезпечення	$C_п$	1689
9. Плановий прибуток	$П_р$	845
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	2534
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	506,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3040,8

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1689 = 845 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3041
Всього капітальних витрат	–	3041

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	8138	4069
2. Витрати на електроенергію	$З_{ел}$	311	155
3. Витрати на амортизацію	$З_{ам}$	0	760
Всього витрат за рік	$І$	8449	4984

Витрати на обслуговування роботи системи:

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

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

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

Після купівлі нового програмного забезпечення кількість годин на обслуговування системи зменшилось з 300 год до 150 год на рік.

$$З_{р\,баз} = 300 \cdot 18 \cdot 1{,}1 \cdot 1{,}37 = 8137{,}8 \text{ грн},$$

$$З_{р\,нов} = 150 \cdot 18 \cdot 1{,}1 \cdot 1{,}37 = 4068{,}9 \text{ грн}.$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації, %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3041	–	760,25
Всього відрахувань	-	–	3041	–	760,25

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

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

$$З_{ел\ баз} = 0{,}545 \cdot 300 \cdot 1{,}9 = 311 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 150 \cdot 1{,}9 = 155 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (2534 - 1689) \cdot 120 - (0{,}05 \cdot 1408000 + 0{,}4 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 120000) \cdot 3/12 = 53828{,}5 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{1760367}{(2534 - 1689) \cdot 120 \cdot 12 / 3} = 4 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

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

$$E_{сп} = (8449\text{-}4984)\text{-}0{,}25 \cdot 3041 = 2705 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	120
2. Повна собівартість розробленої програми	Грн.	1689
3. Ціна розробленої програми	Грн.	2534
4. Плановий прибуток від реалізації розробленої програми	Грн.	845
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	101400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	53828,5
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3041
11.Величина економічного ефекту у користувача програмної продукції	Грн.	2705
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,8

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

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

$$T_{сп} = \frac{3041}{8449 - 4984} = 0,8 \ року .$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя. Охорона праці є складовою частиною безпеки життєдіяльності.

Охорона праці − це:

Система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності.

Законом України “Про охорону праці” [1] регламентуються загальні

Законом України “Про охорону праці” регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні

електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

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

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	8,5*
Довжина	13*
Висота	2,9

* вказано загальні розміри поєднаного приміщення, де загалом працюють 16 людей, а фактично у наявності є дві кімнати, розділених перестінком.

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

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Обсяг, V	$м^3$	не менше 20.0	20

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

У зазначеному приміщенні працює 16 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку л юдина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких:

вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	23-24	55-70	0,14

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер *HP Laser 135a*, електродвигуни вентиляторів ЕОМ.

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

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

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

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

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 8,5 м, довжина 13 м, висота 2,9 м.

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

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, *Лм*;

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

S – площа освітлюваного приміщення (у нашому випадку *S*= 8,5 х 13 = 110,5 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення

залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку $K = 1,5$);

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

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

Обчислимо індекс приміщення за формулою:

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

де:

S – площа приміщення, $S = 110,5$ м^2;

h – розрахункова висота підвісу, $h = 2,9$ м (співпадає з висотою стелі, т.я. світильники освітлення закріплюються на стелі);

A – ширина приміщення, $A = 8,5$ м;

B – довжина приміщення, $B = 13$ м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=1,77.$$

Знаючи індекс приміщення (i), за знаходимо $n = 0,57$ (з табличних даних коефіцієнтів використання світлового потоку (n) світильників [2]). Підставимо всі значення у формулу, визначемо світловий потік: F=95960 *Лм*.

Для штучного освітлення приміщення використовуються світильники *LED 54W-5000K*, світловий потік яких $F_л = 5940$ *Лм*.

Кількість світильників визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекс приміщення:

$$N= 95960/5940=16,15 \text{ шт.}$$

Для забезпечення нормованої мінімальної освітленості приймаємо необхідну кількість світильників 17 шт.

8.6 Висновки до розділу

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи централізованого розподілу ключів.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів централізованого розподілу ключів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем централізованого розподілу ключів.

– Досліджена система централізованого розподілу ключів.

– На основі отриманих результатів досліджень створена програмна реалізація системи централізованого розподілу ключів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання централізованого розподілу ключів.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ММВ.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 2705 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,8 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Окаєвич Ю.О. Дослідження та програмна реалізація системи централізованого розподілу ключів // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.
2. Лапонина О.Р. Основи сетевой безопасности: криптографические алгоритми и протоколи взаимодействия Интернет-университет информационних технологий – ИНТУИТ.ру, 2005
3. Хаулет Т. Защитные средства с открытими исходними текстами. БИНОМ. Лаборатория знаний, Интернет-университет информационних технологий – ИНТУИТ.ру, 2007
4. Галатенко В.А. Основи информационной безопасности. Интернет-университет информационних технологий – ИНТУИТ.ру, 2005
5. Галатенко В.А. Стандарти информационной безопасности. Интернет-университет информационних технологий – ИНТУИТ.ру, 2005
6. В. Столлингс Криптография и защита сетей. Принципи и практика 2-е изд. 2001г., Издательский дом «Вильямс», 672 с.
7. Б. Шнайер Прикладная криптография. Протоколи, алгоритми и исходние тексти на язике С 2-е изд. 2003г.
8. М.А. Иванов Криптографические методи защити информации в компьютерних системах и сетях 2001г., «Кудиц-образ», 386с.
9. RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile 2002г. 129с.
10. RFC 3281 An Internet Attribute Certificate Profile for Authorization 2002г. 40с.
11. RFC 2510 Internet X.509 Public Key Infrastructure Certificate Management Protocols 1999г. 72с.
12. RFC 2511 Internet X.509 Certificate Request Message Format 1999г. 25с.

13. RFC 2560 X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP 1999г. 23с.

14. Internet Security Association and Key Management Protocol (ISAKMP) RFC 2408 1998г. 86с.

15. The Internet Key Exchange (IKE) RFC 2409 1998г. 41с.

16. Information Technology Security Evaluation Criteria (ITSEC). Harmonized Criteria of France – Germany – the Netherlands – the United Kingdom Department of Trade and Industry, London, 1991.

17. Security Architecture for Open Systems Interconnection for CCITT Applications. Recommendation X.800 CCITT, Geneva, 1991.

18. Балакирский В.Б Безопасность электронних платежей. Защита информации Конфидент, № 5, 1996

19. Бернет С., Пэйн С Криптография. Официальное руководство RSA Security М.: Бином-Пресс, 2002

20. Бруно Л Certificate Authorities: Кому Ви доверяете? Data Communications (Russian edition). № 3, 1998

21. Вьюкова Н Сервер аутентификации Kerberos

22. Галатенко В.А Информационная безопасность. Обзор основних положений Jet Info. № 1-3, 1996

23. Галатенко В.А Стандарти в области безопасности распределенних систем Jet Info, № 5, 1999

24. Горбатов В.С., Полянская О.Ю Доверенние центри как звено системи обеспечения безопасности корпоративних информационних ресурсов Информационний бюллетень Jet Info, № 11 (78), 1999

25. Горбатов В.С., Полянская О.Ю Основи технологии PKI М.: Горячая линия – Телеком, 2003

26. Горбатов В.С., Полянская О.Ю Программная поддержка инфраструктури с откритими ключами Безопасность информационних технологий, Вип. 2, МИФИ, 2001

27. Закон Украини "Об электронной цифровой подписи"

28. Карве А Защищенний обмен сообщениями LAN / Журнал сетевих решений, № 12, 1998

29. Карпов А Г Удостоверяющий центр в системе электронного документооборота. Опит построения открити систем

30. Лукацкий А.В Как обеспечить подлинность электронних документов

31. Мэтью С Инфраструктура открити ключей: состояние и перспективи

32. Полянская О.Ю Проблеми и риски в работе удостоверяющих центров

33. Полянская О.Ю Стандарти и спецификации в области инфраструктур открити ключей. Безопасность информационних технологий, Вип. 1 М.: МИФИ, 2003

34. Полянская О.Ю Методи распространения информации в инфраструктурах открити ключей, Безопасность информационних технологий, Вип. 4 М.: МИФИ, 2004

35. Рапоза Д Незнакомая PKI

36. Семенов Г Не только шифрование, или Обзор криптотехнологий Jet Infosystems, № 3 (94), 2001

37. Симонович П.С Регулирование электронной цифровой подписи нормами права: международний опит

38. Харли Х Конфиденциальность сообщений: будь начеку LAN/Журнал сетевих решений, № 7-8,1999

39. Циммерман Ф.Р PGP: концепция безопасности и уязвимие места

40. Шабат В Каталоги LDAP и мета каталоги

41. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

42. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов,

Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

43. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

44. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

45. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

46. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

47. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

48. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam,

S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

49. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

50. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

51. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. – практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

52. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

53. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

54. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

55. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев,

С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

56. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

57. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

58. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

59. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

60. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

61. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных

сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

62. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

63. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

64. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

65. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

66. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та

взаємодії» (IT & I): зб. тез II міжнар. наук. -практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

67. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

68. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

69. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

70. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

71. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

72. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук. -практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

73. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

74. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня – 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

75. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

76. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

77. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов,

С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

78. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая – 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

79. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. – техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

80. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

81. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

82. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

83. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-122.21.0098.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Окаєвич Ю.О.			Дослідження та програмна реалізація системи централізованого розподілу ключів	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнова Т.В.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КН-20М3	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи централізованого розподілу ключів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 40-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи централізованого розподілу ключів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи централізованого розподілу ключів;

– 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 XP/Vista/7/8/10 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 XP/Vista/7/8/10.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.1.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 110 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 23.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнова Т.В.

Дослідження та програмна реалізація системи централізованого розподілу ключів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 43

Літера: РП

Кропивницький – 2021 року

Клієнтська частина програмного забезпечення
Файл Project_MASTER_KEYS.dpr

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
program Project_MASTER_KEYS;
uses
Forms,SysUtils,
frmMAIN in 'frmMAIN.pas' {Form1},
frmSettings in 'frmSettings.pas' {Form2},
frmKEYGEN in 'frmKEYGEN.pas' {Form3},
frmLOG in 'frmLOG.pas' {Form4},
frmKEYSTAT in 'frmKEYSTAT.pas' {Form5},
frmSPLASH in 'frmSPLASH.pas' {U_Form_Splash},
frmAbout in 'frmAbout.pas' {AboutBox};
frmUnit1 in 'frmUnit.pas' {Unit6};

{$R *.res}
begin
try
U_Form_Splash:=TU_Form_Splash.Create(Application);
U_Form_Splash.Show;
U_Form_Splash.Update;
U_Form_Splash.Label2.Caption:='Підключення модулів';
U_Form_Splash.Update;
U_Form_Splash.Label2.Caption:='Налагодження інтерфейсів';
U_Form_Splash.Update;
Application.HintPause:=200;
Application.HintHidePause:=7000;
Application.HintShortPause:=25;
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.CreateForm(TForm2, Form2);
Application.CreateForm(TForm3, Form3);
Application.CreateForm(TForm4, Form4);
Application.CreateForm(TForm5, Form5);
Application.CreateForm(Tform6, Form6);
Application.CreateForm(TAboutBox, AboutBox);
finally
U_Form_Splash.free;
end;
Application.Run;
end.
```

Файл frmKEYGEN.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmKEYGEN;

interface

uses


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

type
  TForm3 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

end.
```

Файл frmKEYSTAT.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmKEYSTAT;

interface

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

type
  TForm5 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Button1: TButton;
    Button2: TButton;
    ListBox1: TListBox;
    SaveDialog1: TSaveDialog;
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.dfm}

procedure TForm5.Button2Click(Sender: TObject);
begin

if SaveDialog1.Execute then
begin
ListBox1.Items.SaveToFile(SaveDialog1.FileName);
end;

end;

end.
```

Файл Unit1.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, RxMenus, RXShell, ComCtrls, ExtCtrls, StdCtrls, ScktComp,
ToolWin,WinSock;
const
PORT_NUM=5555;
type
TForm1 = class(TForm)
RxTrayIcon1: TRxTrayIcon;
RxPopupMenu1: TRxPopupMenu;
Showform1: TMenuItem;
N1: TMenuItem;
About1: TMenuItem;
N2: TMenuItem;
Exit1: TMenuItem;
Panel1: TPanel;
StatusBar1: TStatusBar;
Panel3: TPanel;
Panel2: TPanel;
Memo1: TMemo;
Panel4: TPanel;
GroupBox1: TGroupBox;
TrackBar1: TTrackBar;
StaticText1: TStaticText;
GroupBox2: TGroupBox;
Button4: TButton;
Button5: TButton;
GroupBox3: TGroupBox;
Edit1: TEdit;
GroupBox4: TGroupBox;
Edit2: TEdit;
Timer1: TTimer;
GroupBox5: TGroupBox;
Button1: TButton;
Button2: TButton;
Button3: TButton;
GroupBox6: TGroupBox;
Edit3: TEdit;
Edit4: TEdit;
StaticText2: TStaticText;
StaticText3: TStaticText;
procedure RxTrayIcon1Click(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Showform1Click(Sender: TObject);
procedure TrackBar1Change(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button2Click(Sender: TObject);
private A:TClientSocket;
procedure Add_Memo(s:string);
procedure ApplicationMinimize(Sender : TObject);
procedure ApplicationRestore(Sender : TObject);
function GetIP:string;
function Get_User: string;
procedure Accept(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientDisconnect(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientError(Sender: TObject; Socket: TCustomWinSocket;
ErrorEvent: TErrorEvent; var ErrorCode: Integer);
procedure ClientRead(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientWrite(Sender: TObject; Socket: TCustomWinSocket);
procedure GetThread(Sender: TObject;
ClientSocket: TServerClientWinSocket;
```

```
var SocketThread: TServerClientThread);
procedure Listen(Sender: TObject; Socket: TCustomWinSocket);
procedure ThreadEnd(Sender: TObject; Thread: TServerClientThread);
procedure ThreadStart(Sender: TObject; Thread: TServerClientThread);
public
{-----------}
DT:TDateTime;
procedure Connect(Sender: TObject; Socket: TCustomWinSocket);
procedure Read(Sender: TObject; Socket: TCustomWinSocket);
procedure Write(Sender: TObject; Socket: TCustomWinSocket);
{-----------}
function
PACKED_SBOR(LOGIN:string;PASS:string;exclusion:string;RASMER:integer):string;
procedure PACKED_RASBOR(s:string);
end;

var
Form1: TForm1;

implementation
var
Login,Pass,Excl:string;
COL:integer;
{$R *.dfm}

procedure TForm1.RxTrayIcon1Click(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
if (IsWindowVisible(Form1.Handle)=false) then
begin
Application.Restore;
Form1.show;
Form1.BringToFront;
end
else
begin
Form1.hide;
end;end;

procedure TForm1.Showform1Click(Sender: TObject);
begin
Application.ProcessMessages;
Form1.Show; Form1.BringToFront;

Application.Restore;
Application.BringToFront;
end;

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
StaticText1.caption:=inttostr(TrackBar1.position)+' сек.';
Timer1.Interval:=TrackBar1.position*1000;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
A:=TClientSocket.Create(self);
A.OnRead:=Read; A.OnWrite:=Write;
A.OnConnect:=Connect;
A.ClientType:=ctNonBlocking;
A.Host:=Edit1.text;
A.Port:=strtoint(Edit2.text);
A.Open; Timer1.Enabled:=true;
end;

procedure TForm1.Connect(Sender: TObject; Socket: TCustomWinSocket);
begin

end;
```

```
procedure TForm1.Read(Sender: TObject; Socket: TCustomWinSocket);
begin


Add_memo('Read '+Socket.ReceiveText);


end;

procedure TForm1.Write(Sender: TObject; Socket: TCustomWinSocket);
begin
Add_memo('Write');
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnMinimize := ApplicationMinimize;
Application.OnRestore := ApplicationRestore;
Add_Memo('Запуск клиента');

DT:=now;
end;

procedure TForm1.Add_Memo(s: string);
begin
Memo1.lines.Add(DateTimeToStr(now)+' : '+s);
end;

procedure TForm1.ApplicationMinimize(Sender: TObject);
begin
Application.ProcessMessages;
ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.ApplicationRestore(Sender: TObject);
begin
Application.ProcessMessages;
ShowWindow(Application.Handle, SW_HIDE);
end;

function TForm1.GetIP: string;
var
WSAData : TWSAData;
p : PHostEnt;
Name : array [0..$FF] of Char;
s:string;
begin
WSAStartup($0101, WSAData);
GetHostName(name, $FF);
p := GetHostByName(Name);
s:=inet_ntoa(PInAddr(p.h_addr_list^)^);
WSACleanup;
result:=s;
end;

function TForm1.Get_User: string;
var
Buffer: array[0..MAX_PATH] of Char;
sz:DWord;
begin
sz:=MAX_PATH-1;
if windows.GetUserName(Buffer,sz)
then begin
if sz>0 then dec(sz);
SetString(Result,Buffer,sz);
end else begin
Result:='Error '+inttostr(GetLastError);
end;
```

```
end;

procedure TForm1.Accept(Sender: TObject; Socket: TCustomWinSocket);
begin

end;

procedure TForm1.ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
begin
Add_memo('ХОСТ:'+Socket.RemoteHost+' АДРЕС:'+Socket.RemoteAddress+'
ПОРТ:'+inttostr(Socket.RemotePort));
end;

procedure TForm1.ClientDisconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
Add_memo('Сервер');
end;

procedure TForm1.ClientError(Sender: TObject; Socket: TCustomWinSocket;
ErrorEvent: TErrorEvent; var ErrorCode: Integer);
begin
Add_memo('Client error. Code = '+IntToStr(ErrorCode));
MessageDlg('Client error. Code = '+IntToStr(ErrorCode),mtInformation,[mbOK],0);
SysErrorMessage(GetLastError);
end;

procedure TForm1.ClientRead(Sender: TObject; Socket: TCustomWinSocket);
begin
Add_memo('COMAND:'+Socket.ReceiveText);
end;

procedure TForm1.ClientWrite(Sender: TObject; Socket: TCustomWinSocket);
begin
end;

procedure TForm1.Listen(Sender: TObject; Socket: TCustomWinSocket);
begin

end;

procedure TForm1.ThreadEnd(Sender: TObject; Thread: TServerClientThread);
begin

end;


procedure TForm1.ThreadStart(Sender: TObject; Thread: TServerClientThread);
begin

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
if (Edit3.Text<>'')and(Edit4.Text<>'') then
begin
A.Socket.SendText(PACKED_SBOR(Edit3.Text,Edit4.Text,'324-02',10));
 end;
end;

procedure TForm1.PACKED_RASBOR(s: string);
var
A:string;
i:integer;
Point:integer;
begin
A:=s;
MessageDlg(A,mtInformation,[mbOK],0);
if (s<>'') then
```

```
begin
Point:=pos('*',A);
Login:=copy(A,0,point-1);
MessageDlg(Login,mtInformation,[mbOK],0);
A:=copy(A,point+1,Length(A)-point);

Point:=pos('*',A);
Pass:=copy(A,0,point-1);;
MessageDlg(Pass,mtInformation,[mbOK],0);

A:=copy(A,point+1,Length(A)-point);
Point:=pos('*',A);
Excl:=copy(A,0,point-1);;
MessageDlg(Excl,mtInformation,[mbOK],0);

A:=copy(A,point+1,Length(A)-point);
COL:=strtoint(A);
MessageDlg(A,mtInformation,[mbOK],0);
end
else MessageDlg('No DATA!!!',mtInformation,[mbOK],0);
end;


function TForm1.PACKED_SBOR(LOGIN, PASS, exclusion: string;
RASMER: integer): string;
begin
result:=LOGIN+'*'+PASS+'*'+exclusion+'*'+inttostr(RASMER);
end;

end.
```

Файл frmAbout.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmAbout;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, jpeg;

type
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

implementation

{$R *.dfm}

end.
```

Файл frmLOG.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmLOG;

interface

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

type
  TForm4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Memo1: TMemo;
    Button1: TButton;
    Button2: TButton;
    SaveDialog1: TSaveDialog;
    procedure FormShow(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form4: TForm4;

implementation

uses frmMAIN;
{$R *.dfm}

procedure TForm4.FormShow(Sender: TObject);
begin
      Memo1.Clear;
      Memo1.Lines.Assign(Form1.Memo1.Lines);
end;
procedure TForm4.Button1Click(Sender: TObject);
begin
      form4.Hide;
      Form1.Show;
end;
procedure TForm4.Button2Click(Sender: TObject);
begin
      if SaveDialog1.Execute then
      begin
            Memo1.Lines.SaveToFile(SaveDialog1.FileName);
      end;
end;
end.
```

Файл frmMAIN.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmMAIN;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, StdCtrls, ExtCtrls,ScktComp, Buttons;
const
PORT_NUM=5555;
 type
TForm1 = class(TForm)
Panel1: TPanel;
MainMenu1: TMainMenu;
Fil1: TMenuItem;
N1: TMenuItem;
N2: TMenuItem;
N3: TMenuItem;
N4: TMenuItem;
Panel2: TPanel;
Memo1: TMemo;
Panel3: TPanel;
Panel5: TPanel;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
BitBtn6: TBitBtn;
BitBtn7: TBitBtn;
BitBtn8: TBitBtn;
BitBtn9: TBitBtn;
BitBtn10: TBitBtn;
BitBtn11: TBitBtn;
BitBtn12: TBitBtn;
BitBtn13: TBitBtn;
BitBtn14: TBitBtn;
ListBox2: TListBox;
procedure FormCreate(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure BitBtn9Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn13Click(Sender: TObject);
procedure BitBtn12Click(Sender: TObject);
procedure BitBtn10Click(Sender: TObject);
procedure N4Click(Sender: TObject);

private
CLIENT_CONNECT:integer;
A:TServerSocket;
public
procedure Add_Memo(s:string);
procedure Listen(Sender : TObject;Socket: TCustomWinSocket);
procedure Accept(Sender: TObject;Socket: TCustomWinSocket);
procedure ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientDisconnect(Sender: TObject;Socket: TCustomWinSocket);
procedure ClientError(Sender: TObject;Socket: TCustomWinSocket; ErrorEvent:
TErrorEvent; var ErrorCode: Integer);
procedure ClientRead(Sender: TObject; Socket: TCustomWinSocket);
procedure ClientWrite(Sender: TObject;Socket: TCustomWinSocket);
procedure GetThread(Sender: TObject;ClientSocket: TServerClientWinSocket;var
SocketThread: TServerClientThread);
procedure ThreadEnd(Sender: TObject;Thread: TServerClientThread);

procedure ThreadStart(Sender: TObject; Thread: TServerClientThread);
procedure ADD_CLIENT_DATA(A:string);
```

```
procedure PACKED_RASBOR(s:string);
procedure PACKED_processing;
end;

var
Form1: TForm1;
implementation

uses frmLOG, frmKEYSTAT;
var
Login,Pass,Excl:string;
COL:integer;
{$R *.dfm}

procedure TForm1.Accept(Sender: TObject; Socket: TCustomWinSocket);
begin

end;

procedure TForm1.ADD_CLIENT_DATA(A: string);
begin

end;

procedure TForm1.ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
begin
Memo1.lines.add('Clietnts:'+Socket.RemoteAddress+'Хост:'+Socket.RemoteHost+' ->
CONNECT');
ListBox2.Items.Add('ADD:');
ListBox2.Items.Add(' Klient:'+Socket.RemoteAddress);
ListBox2.Items.Add(' Хост:'+Socket.RemoteHost);
end;

procedure TForm1.ClientDisconnect(Sender: TObject;
Socket: TCustomWinSocket);
begin
Memo1.lines.add('Klient:'+Socket.RemoteAddress+'Хост:'+Socket.RemoteHost+' ->
DISCONNECT');
ListBox2.Items.Add('Diskonect:');
ListBox2.Items.Add('Klient:'+Socket.RemoteAddress);
ListBox2.Items.Add('Хост:'+Socket.RemoteHost);
end;

procedure TForm1.ClientError(Sender: TObject; Socket: TCustomWinSocket;
ErrorEvent: TErrorEvent; var ErrorCode: Integer);
begin

end;

procedure TForm1.ClientRead(Sender: TObject; Socket: TCustomWinSocket);
begin
PACKED_RASBOR(Socket.ReceiveText);
PACKED_processing;
Add_Memo('Name='+Login);
Add_Memo('Pass='+Pass);
Add_Memo('Excl='+Excl);
Add_Memo('Dlinna='+inttostr(COL));
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
A:=TServerSocket.Create(self);
A.ServerType:=stNonBlocking;
A.OnListen:=Listen;
A.OnAccept:=Accept;
A.OnClientConnect:=ClientConnect;
A.OnClientDisconnect:=ClientDisconnect;
A.OnClientError:=ClientError;
A.OnClientRead:=ClientRead;
```

```
A.OnClientWrite:=ClientWrite;
A.OnGetThread:=GetThread;
A.OnThreadEnd:=ThreadEnd;
A.OnThreadStart:=ThreadStart;
A.Port:=PORT_NUM;
A.Open;
CLIENT_CONNECT:=0;
Form1.Caption:='ADD DATA(ПОРТ'+inttostr(PORT_NUM)+')';
Add_Memo('Сервер запущен');
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
A.Close;
A.Destroy;
end;

procedure TForm1.Add_Memo(s: string);
begin
Memo1.lines.Add(DateTimeToStr(now)+' : '+s);
end;

procedure TForm1.PACKED_RASBOR(s: string);
var
A:string;
Point:integer;
begin
A:=s;
MessageDlg(A,mtInformation,[mbOK],0);
if (s<>'') then
begin
Point:=pos('*',A);
Login:=copy(A,0,point-1);
MessageDlg(Login,mtInformation,[mbOK],0);
A:=copy(A,point+1,Length(A)-point);
Point:=pos('*',A);
Pass:=copy(A,0,point-1);;
MessageDlg(Pass,mtInformation,[mbOK],0);
A:=copy(A,point+1,Length(A)-point);
Point:=pos('*',A);
Excl:=copy(A,0,point-1);;
MessageDlg(Excl,mtInformation,[mbOK],0);
A:=copy(A,point+1,Length(A)-point);
COL:=strtoint(A);
MessageDlg(A,mtInformation,[mbOK],0);
end;
end;
procedure TForm1.BitBtn9Click(Sender: TObject);
begin
      form1.Close;
end;
procedure TForm1.PACKED_processing;
begin
      A.Socket.Connections[0].SendText('JHJKTT');
      A.Socket.SendText('OK');
      A.Socket.SendText('OK');
      A.Socket.SendText('OK');}
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
      Form4.show;
      Form1.Hide;
end;
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
      Memo1.Lines.Clear;
      ListBox2.Items.Clear;
end;
procedure TForm1.BitBtn12Click(Sender: TObject);
```

```
begin
      MessageDlg('Clear',mtInformation,[mbOK],0);
      Form5.ListBox1.Items.Clear;
end;
procedure TForm1.BitBtn10Click(Sender: TObject);
begin
      MessageDlg('Ok',mtInformation,[mbOK],0);
end;
procedure TForm1.N4Click(Sender: TObject);
begin
      MessageDlg('Betta vershion 1.0',mtInformation,[mbOK],0);
end;
end.
```

Файл frmSettings.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmSettings;

interface

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

type
  TForm2 = class(TForm)
    Panel2: TPanel;
    Panel3: TPanel;
    Button1: TButton;
    Button2: TButton;
    Panel4: TPanel;
    PageControl2: TPageControl;
    TabSheet6: TTabSheet;
    GroupBox4: TGroupBox;
    TabSheet7: TTabSheet;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    TabSheet8: TTabSheet;
    GroupBox3: TGroupBox;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    CheckBox3: TCheckBox;
    CheckBox4: TCheckBox;
    CheckBox5: TCheckBox;
    TabSheet9: TTabSheet;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation
{$R *.dfm}
end.
```

Файл frmSPLASH.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit frmSPLASH;

interface

uses


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

type
  TU_Form_Splash = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Animate1: TAnimate;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  U_Form_Splash: TU_Form_Splash;

implementation

{$R *.dfm}

end.
```

Розроблена бібліотека файл Logics.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit Logics;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
glLogics,
glSBox, StdCtrls, glGrBox, ExtCtrls, ComCtrls, dsgnintf, ToolWin, ImgList,
richEdit,
glPage, Tabs;

type
TchGroupBoxPlus = class;

TchLogicsComponentEditor = class(TComponentEditor)
procedure ExecuteVerb(Index: Integer); override;
function GetVerb(Index: Integer): string; override;
function GetVerbCount: Integer; override;
private
procedure ShowEditor(LogicProducer: TLogicProducer);
end;

TchLogicsEditor = class(TForm)
SB: TchScrollBox;
Panel1: TPanel;
iPKey: TImage;
iFKey: TImage;
Label2: TLabel;
cbMode: TComboBox;
SBar: TStatusBar;
iLink: TImage;
cbNext: TComboBox;
Label4: TLabel;
eStepName: TEdit;
Label1: TLabel;
ImageList1: TImageList;
ToolBar1: TToolBar;
ToolButton1: TToolButton;
tbNew: TToolButton;
ToolButton3: TToolButton;
ToolButton4: TToolButton;
cbNextFalse: TComboBox;
Label5: TLabel;
Image3: TImage;
Label7: TLabel;
Label8: TLabel;
Shape1: TShape;
ImageList: TImageList;
ToolButton2: TToolButton;
ToolButton5: TToolButton;
ToolButton6: TToolButton;
ToolButton7: TToolButton;
cbIgnoreSpaces: TCheckBox;
ToolButton8: TToolButton;
pLeft: TPanel;
pLog: TPanel;
Splitter1: TSplitter;
tbStop: TToolButton;
Panel2: TPanel;
Splitter2: TSplitter;
reReslt: TRichEdit;
PC: TchPageControl;
tsLog: TTabSheet;
mLog: TMemo;
tsDictionary: TTabSheet;
```

```
mDictionary: TMemo;
Shape2: TShape;
Shape3: TShape;
Shape4: TShape;
Shape5: TShape;
Shape6: TShape;
Shape7: TShape;
Shape8: TShape;
Shape9: TShape;
Shape10: TShape;
Shape11: TShape;
Shape12: TShape;
Shape13: TShape;
TabSet1: TTabSet;
procedure SBEraseBkgndEvent(Sender: TObject; DC: HDC);
procedure cbNextChange(Sender: TObject);
procedure cbModeChange(Sender: TObject);
procedure tbNewClick(Sender: TObject);
procedure cbNextFalseChange(Sender: TObject);
procedure eStepNameChange(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure ToolButton5Click(Sender: TObject);
procedure ToolButton7Click(Sender: TObject);
procedure cbIgnoreSpacesClick(Sender: TObject);
procedure pLeftDockOver(Sender: TObject; Source: TDragDockObject; X,
Y: Integer; State: TDragState; var Accept: Boolean);
procedure pLeftUnDock(Sender: TObject; Client: TControl;
NewTarget: TWinControl; var Allow: Boolean);
procedure pLeftDockDrop(Sender: TObject; Source: TDragDockObject; X,
Y: Integer);
procedure ToolButton8Click(Sender: TObject);
procedure tbStopClick(Sender: TObject);
procedure TabSet1Change(Sender: TObject; NewTab: Integer;
var AllowChange: Boolean);
private
FActiveBox: TchGroupBoxPlus;
LogicProducer: TLogicProducer;
Logics: TLogics;

procedure MouseDown_(Sender: TObject; Button: TMouseButton; Shift: TShiftState;
X, Y: Integer);
procedure MouseMove_(Sender: TObject; Shift: TShiftState; X, Y: Integer);
procedure MouseUp_(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X,
Y: Integer);
procedure DblClick_(Sender: TObject);
procedure SetActiveBox(const Value: TchGroupBoxPlus);
procedure UpdateView;
procedure AddBox(LogicElement_: TLogicElement);
procedure AddShape(CommentArea: TCommentArea);

procedure OnTraceMessage(Sender: TLogics; fStepResult: boolean; const
StepResult, ParsedResult, Msg: string);
public
function Execute(LogicProducer: TLogicProducer): boolean;
property ActiveBox: TchGroupBoxPlus read FActiveBox write SetActiveBox;
end;

TchGroupBoxPlus = class(TchGroupBox)
public
pt: TPoint;
Selected: boolean;
LogicElement: TLogicElement;
fAsLogical: boolean;
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
procedure Paint; override;
end;

TchShapePlus = class(TShape)
```

```
public
pt: TPoint;
Selected: boolean;
CommentArea: TCommentArea;
procedure Paint; override;
end;

var
glLogicsEditor: TchLogicsEditor;

implementation
uses glTypes, glUtils, geLogicItemEditor, clipbrd;
{$R *.DFM}

function TchLogicsEditor.Execute(LogicProducer: TLogicProducer): boolean;
var
i: integer;
begin
fLogicItemEditor := TfLogicItemEditor.Create(nil);

mDictionary.Lines.Assign(LogicProducer.Dictionary);
try

self.LogicProducer := LogicProducer;
self.Logics := LogicProducer.Logics;

Logics.OnTraceMessage := OnTraceMessage;

cbNext.Items.Clear;
cbNextFalse.Items.Clear;
cbNext.Items.Add('');
cbNextFalse.Items.Add('');

for i := 0 to Logics.Count-1 do AddBox(Logics[i]);

for i := 0 to LogicProducer.CommentAreas.Count-1 do
AddShape(LogicProducer.CommentAreas[i]);

Result := ShowModal = mrOK;
finally
fLogicItemEditor.Free;
end;

pLog.Dock(pLeft, rect(1,1,10,10));
pLog.Dock(pLeft, rect(1,1,10,10));
pLeft.Dock(pLog, rect(1,1,1,1));
end;

procedure TchLogicsEditor.AddBox(LogicElement_: TLogicElement);
var
Box: TchGroupBoxPlus;
begin
Box := TchGroupBoxPlus.Create(self);
with Box do
begin
Parent := SB;
Left := LogicElement_.Left;
Top := LogicElement_.Top;
Width := 100;
Height := 50;
Caption := LogicElement_.Caption;
Options := Options - [fgoCanCollapse];
CaptionAlignment := fcaWidth;
CaptionBorder.Inner := bvRaised;
CaptionBorder.Outer := bvLowered;
Colors.Caption := clBtnShadow;
Colors.TextActive := clBtnHighlight;
OnMouseDown := MouseDown_;
OnMouseMove := MouseMove_;
```

```
OnMouseUp := MouseUp_;
OnDblClick := DblClick_;
Border.Inner := bvRaised;
Border.Outer := bvNone;
Colors.Client := clWhite;
Colors.ClientActive := clWhite;
Box.LogicElement := LogicElement_;

cbNext.Items.AddObject(LogicElement_.Caption, LogicElement);
cbNextFalse.Items.AddObject(LogicElement_.Caption, LogicElement);

end;
end;

procedure TchLogicsEditor.AddShape(CommentArea: TCommentArea);
var
Shape: TchShapePlus;
begin
Shape := TchShapePlus.Create(self);
Shape.CommentArea := CommentArea;
with Shape do
begin
Parent := SB;
Left := CommentArea.Left;
Top := CommentArea.Top;
Width := CommentArea.Width;
Height := CommentArea.Height;
Pen.Style := psDashDot;
Brush.Style := bsClear;

OnMouseDown := MouseDown_;
OnMouseMove := MouseMove_;
OnMouseUp := MouseUp_;
OnDblClick := DblClick_;
end;

end;

procedure TchLogicsEditor.MouseDown_(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
var i: integer;
begin
if Sender is TchShapePlus then with Sender as TchShapePlus do
begin
pt.X := X; pt.Y := Y;
pt := ClientToScreen(pt);
Selected := true;
Tag := 1;
if (X>=Width-5)and(X<Width)and(Y>=Height-5)and(Y<Height) then Tag := 2;
exit;
end;

with TchGroupBoxPlus(Sender) do
begin
pt.X := X; pt.Y := Y;
pt := ClientToScreen(pt);
pt.Y := pt.Y+ SB.VertScrollBar.ScrollPos;
Tag := 1;
Options := Options + [fgoDelineatedText];
Colors.Caption := clBtnHighlight;
Colors.TextActive := clBlack;
Font.Style := [fsBold];
Selected := true;
ActiveBox := TchGroupBoxPlus(Sender);
end;

for i:=0 to SB.ControlCount-1 do
if (SB.Controls[i] is TchGroupBoxPlus) then with TchGroupBoxPlus(SB.Controls[i])
do
```

```
begin
if ActiveBox = TchGroupBoxPlus(SB.Controls[i]) then continue;
Options := Options - [fgoDelineatedText];
Colors.Caption := clBtnShadow;
Colors.TextActive := clBtnHighlight;
Font.Style := [];
Selected := false;
Repaint;
end;
end;

procedure TchLogicsEditor.MouseMove_(Sender: TObject; Shift: TShiftState; X, Y:
Integer);
var
pt_new: TPoint;
begin
if Sender is TchShapePlus then with Sender as TchShapePlus do
begin
case Tag of
1:
begin
pt_new.X := X; pt_new.Y := Y;
pt_new := ClientToScreen(pt_new);
Left := Left + pt_new.X - pt.X;
Top := Top + pt_new.Y - pt.Y;
CommentArea.Left := Left + SB.HorzScrollBar.ScrollPos;
CommentArea.Top := Top + SB.VertScrollBar.ScrollPos;
end;
2:
begin
pt_new.X := X; pt_new.Y := Y;
pt_new := ClientToScreen(pt_new);
Width := Width + pt_new.X - pt.X;
Height := Height + pt_new.Y - pt.Y;
if Width < 50 then Width := 50; if Height < 50 then Height := 50;
CommentArea.Width := Width;
CommentArea.Height := Height;
end;
end;
pt.X := pt_new.X; pt.Y := pt_new.Y;
exit;
end;

with TchGroupBoxPlus(Sender) do
begin
if bool(Tag) then
begin
pt_new.X := X; pt_new.Y := Y;
pt_new := ClientToScreen(pt_new);
pt_new.Y := pt_new.Y+ SB.VertScrollBar.ScrollPos;
Left := Left + pt_new.X - pt.X;
Top := Top + pt_new.Y - pt.Y;

LogicElement.Left := Left + SB.HorzScrollBar.ScrollPos;
LogicElement.Top := Top + SB.VertScrollBar.ScrollPos;

SBar.SimpleText := IntToStr(Left) + ':' + IntToStr(Top);

UpdateView;
end;
pt.X := pt_new.X; pt.Y := pt_new.Y;
end;
end;

procedure TchLogicsEditor.UpdateView;
var
DC: HDC;
begin
DC := GetDC(SB.Handle);
```

```
SendMessage(SB.Handle, WM_EraseBkgnd, WPARAM(DC),0);
ReleaseDC(SB.Handle, DC);
end;

procedure TchLogicsEditor.MouseUp_(Sender: TObject; Button: TMouseButton; Shift:
TShiftState; X, Y: Integer);
var i: integer;
begin
TControl(Sender).Tag := 0;
for i := 0 to SB.ControlCount-1 do
if (SB.Controls[i] is TchShapePlus) then (SB.Controls[i] as TchShapePlus).Paint;
end;

procedure TchLogicsEditor.DblClick_(Sender: TObject);
var str: string;
begin
TControl(Sender).Tag := 0;
if Sender is TchShapePlus then with Sender as TchShapePlus do
begin
str := CommentArea.Text;
if InputQuery('Caption', 'Comments', str) then CommentArea.Text := str;
PostMessage(TWinControl(Parent).Handle, WM_LBUTTONUP, 1, 1);
exit;
end;

fLogicItemEditor.Execute(Logics, TchGroupBoxPlus(Sender).LogicElement);
PostMessage(TWinControl(Sender).Handle, WM_LBUTTONUP, 1, 1);
end;


procedure TchLogicsEditor.SetActiveBox(const Value: TchGroupBoxPlus);
var
i, Index: integer;
Box: TchGroupBoxPlus;
begin
FActiveBox := Value;

Index := cbNext.Items.IndexOfObject(Value.LogicElement.NextElement);
if Index <> -1 then cbNext.ItemIndex := Index else cbNext.ItemIndex := 0;

Index := cbNextFalse.Items.IndexOfObject(Value.LogicElement.NextFalseElement);
if Index <> -1 then cbNextFalse.ItemIndex := Index else cbNextFalse.ItemIndex :=
0;

cbMode.ItemIndex := integer(FActiveBox.LogicElement.IsFirst);
eStepName.Text := FActiveBox.LogicElement.Caption;
end;

procedure TchLogicsEditor.OnTraceMessage(Sender: TLogics; fStepResult: boolean;
const StepResult, ParsedResult, Msg: string);
begin
mLog.Lines.Add(Msg);
if reReslt.Text = '' then reReslt.Tag := 0;

if length(ParsedResult) = 0 then exit;
tag := 1 - tag;
reReslt.Lines.BeginUpdate;
reReslt.Text := reReslt.Text + ParsedResult;

reReslt.SelStart := length(reReslt.Text) - length(ParsedResult);
reReslt.SelLength := length(ParsedResult);


if tag = 0 then reReslt.SelAttributes.Color:=clRed else
reReslt.SelAttributes.Color:=clGreen;
reReslt.SelAttributes.Color := RGB(100+Random(100), 100+Random(100),
100+Random(100));

reReslt.SelLength := 0;
```

```
reReslt.Lines.EndUpdate;
end;

procedure TchGroupBoxPlus.Paint;
var
i: integer;
R: TRect;
str: string;
begin
inherited;
ChangeBitmapColor((Owner as TchLogicsEditor).iLink.Picture.Bitmap,
GetPixel((Owner as TchLogicsEditor).iLink.Picture.Bitmap.Canvas.Handle, 0,0),
IIF(LogicElement.NextElement<>nil, clGreen, clRed));
BitBlt(Canvas.handle, 100-14-3, 3, 14, 13, (Owner as
TchLogicsEditor).iLink.Picture.Bitmap.Canvas.Handle, 0, 0, SRCCOPY);

Canvas.Font.Color := clTeal;
 Canvas.Font.Style := [];
str := LogicElement.Expression + ' ' + LogicRuleLabels[LogicElement.Rule] + ' '
+ LogicElement.Value;
R := Bounds(3, 20, Width-6, Height-22);
DrawText(Canvas.handle, PChar(str), length(str), R, DT_WORDBREAK or
DT_END_ELLIPSIS or DT_MODIFYSTRING);
end;

procedure TchLogicsEditor.SBEraseBkgndEvent(Sender: TObject; DC: HDC);
var
Canvas: TCanvas;
LogicElement: TLogicElement;
i: integer;
PenFalse, Pen, PenTrue, OldPen, PenGrid: HPen;
Brush, OldBrush: HBrush;
NextBox, PrevBox, PrevFalseBox: TchGroupBoxPlus;
bmp: TBitmap;

function FindBox(LogicElement: TLogicElement): TchGroupBoxPlus;
var
i: integer;
begin
Result := nil;
if LogicElement = nil then exit;
for i := 0 to SB.ControlCount-1 do
if SB.Controls[i] is TchGroupBoxPlus then
if TchGroupBoxPlus(SB.Controls[i]).LogicElement = LogicElement then Result :=
TchGroupBoxPlus(SB.Controls[i]);
end;
procedure Line(X, Y, X2, Y2: integer; isTrueLine: boolean);
const R = 5;
begin
if isTrueLine then SelectObject(DC, PenTrue) else SelectObject(DC, PenFalse);
MoveToEx(DC, X, Y, nil); LineTo(DC, X2, Y2);
SelectObject(DC, Pen);
if (abs(X2-X) < 500)and(abs(Y2-Y) < 500) then
MoveToEx(DC, X, Y+1, nil); LineTo(DC, X2, Y2+1);
Ellipse(DC, X-R-2, Y-R-2, X+R*2-2, Y+R*2-2);
Ellipse(DC, X2-R-2, Y2-R-2, X2+R*2-2, Y2+R*2-2);
end;
procedure DrawGrid;
var i, j: integer;
const step = 14;
begin
FillRect(DC, SB.ClientRect, Brush);
for i:=1 to SB.Width div step do
begin
j := i*14;
MoveToEx(DC, j, 0, nil); LineTo(DC, j, SB.Height);
end;
for i:=1 to SB.Height div step do
begin
```

```
j := i*14;
MoveToEx(DC, 0, j, nil); LineTo(DC, SB.Width, j);
end;
end;
begin
try
Brush := CreateSolidBrush(clWhite);
Pen := CreatePen( PS_SOLID, 1, clBlack );
PenGrid := CreatePen( PS_SOLID, 1, $E0E0E0 );
PenLong := CreatePen( PS_DASHDOT, 1, $E0E0E0 );
PenTrue := CreatePen( PS_SOLID, 1, $FF9090 );
PenFalse := CreatePen( PS_SOLID, 1, $009090 );
OldPen := SelectObject( DC, PenGrid );
OldBrush := SelectObject( DC, Brush );
DrawGrid;
for i := 0 to SB.ControlCount-1 do
begin
if not(SB.Controls[i] is TchGroupBoxPlus) then continue;
LogicElement := TchGroupBoxPlus(SB.Controls[i]).LogicElement;
if LogicElement = nil then exit;
if LogicElement.IsFirst then
begin
MoveToEx(DC, 0, 0, nil);
LineTo(DC, SB.Controls[i].Left, SB.Controls[i].Top);
end;
PrevBox := FindBox(LogicElement.NextElement);
if Assigned(PrevBox) then
begin
Line(SB.Controls[i].Left + SB.Controls[i].Width, SB.Controls[i].Top,
PrevBox.Left, PrevBox.Top, true);
DeleteObject( SelectObject( DC, OldPen ) );
end;
PrevFalseBox := FindBox(LogicElement.NextFalseElement);
if Assigned(PrevFalseBox) then
begin
Line(SB.Controls[i].Left + SB.Controls[i].Width, SB.Controls[i].Top +
SB.Controls[i].Height, PrevFalseBox.Left, PrevFalseBox.Top, false);
DeleteObject( SelectObject( DC, OldPen ) );
end;
bmp := TBitmap.Create;
if LogicElement.NextElement <> nil then
begin
ImageList.GetBitmap(0, bmp);
BitBlt(DC, SB.Controls[i].Left + SB.Controls[i].Width-3, SB.Controls[i].Top,
bmp.width, bmp.height, bmp.Canvas.Handle, 0, 0, SRCCOPY);
end;
if LogicElement.NextFalseElement <> nil then
begin
ImageList.GetBitmap(1, bmp);
BitBlt(DC, SB.Controls[i].Left + SB.Controls[i].Width-3, SB.Controls[i].Top +
SB.Controls[i].Height-17, bmp.width, bmp.height, bmp.Canvas.Handle, 0, 0,
SRCCOPY);
end;
if LogicElement.IsFirst then
begin
ImageList.GetBitmap(2, bmp);
BitBlt(DC, SB.Controls[i].Left - 20, SB.Controls[i].Top, bmp.width, bmp.height,
bmp.Canvas.Handle, 0, 0, SRCCOPY);
end;
bmp.Free;
end;

finally
SelectObject(DC, Pen);
DeleteObject(SelectObject(DC, OldPen));
DeleteObject(PenTrue);
DeleteObject(PenFalse);
DeleteObject(PenGrid);
DeleteObject(Brush);
```

```
end;

for i := 0 to SB.ControlCount-1 do

if (SB.Controls[i] is TchShapePlus) then (SB.Controls[i] as TchShapePlus).Paint;
end;

procedure TchLogicsEditor.cbNextChange(Sender: TObject);
begin
if FActiveBox = nil then exit;
if FActiveBox.LogicElement <> cbNext.items.Objects[cbNext.ItemIndex] then
begin
FActiveBox.LogicElement.NextElement :=
TlogicElement(cbNext.items.Objects[cbNext.ItemIndex]);
end;
UpdateView;
end;

procedure TchLogicsEditor.cbModeChange(Sender: TObject);
begin
if FActiveBox = nil then exit;
FActiveBox.LogicElement.IsFirst := cbMode.ItemIndex = 1;
end;

procedure TchLogicsComponentEditor.ExecuteVerb(Index: Integer);
begin
inherited;
ShowEditor(TLogicProducer(Component));
end;

function TchLogicsComponentEditor.GetVerb(Index: Integer): string;
begin
case Index of
0: Result := 'Edit component...';
end;
end;

function TchLogicsComponentEditor.GetVerbCount: Integer;
begin
Result := 1;
end;

procedure TchLogicsComponentEditor.ShowEditor(LogicProducer: TLogicProducer);
var
glLogicsEditor: TchLogicsEditor;
Logics: TLogics;
begin
Logics := LogicProducer.Logics;
with Logics.Add do
begin
Left := 10; Top := 10;
IsFirst := true;
end;
with Logics.Add do
begin
Left := 200; Top := 30;
PrevElementID := Logics[0].ID;
end;
Logics[0].NextElementID := Logics[1].ID;
with Logics.Add do
begin
Left := 200; Top := 100;
end;

try
glLogicsEditor := TchLogicsEditor.Create(nil);
glLogicsEditor.Execute(LogicProducer);
finally
FreeAndNil(glLogicsEditor);
```

```
end;
end;


procedure TchLogicsEditor.tbNewClick(Sender: TObject);
var
LogicElement: TLogicElement;
begin
LogicElement := Logics.Add;
with LogicElement do
begin
Left := SB.Width div 2; Top := SB.Height div 2;
AddBox(LogicElement);
end;
end;

procedure TchLogicsEditor.cbNextFalseChange(Sender: TObject);
begin
if FActiveBox = nil then exit;
if FActiveBox.LogicElement <> cbNextFalse.items.Objects[cbNextFalse.ItemIndex]
then
begin
FActiveBox.LogicElement.NextFalseElement :=
TlogicElement(cbNextFalse.items.Objects[cbNextFalse.ItemIndex]);
end;
UpdateView;
end;

procedure TchLogicsEditor.eStepNameChange(Sender: TObject);
begin
if not Assigned(ActiveBox) then exit;
ActiveBox.LogicElement.Caption := eStepName.Text;
ActiveBox.Caption := eStepName.Text;
end;

procedure TchLogicsEditor.FormShow(Sender: TObject);
begin
SB.BufferedDraw := true;
end;

procedure TchLogicsEditor.ToolButton5Click(Sender: TObject);
var i: integer;
begin
pLog.Visible := true;

mLog.Lines.Clear;
reReslt.Text := '';

Logics.Analyze;

for i := 0 to SB.ControlCount-1 do
if (SB.Controls[i] is TchGroupBoxPlus) then
if TchGroupBoxPlus(SB.Controls[i]).LogicElement.IsTrue then
TchGroupBoxPlus(SB.Controls[i]).Colors.Caption := clGreen;

end;

procedure TchLogicsEditor.ToolButton7Click(Sender: TObject);
var
CommentArea: TCommentArea;
begin
CommentArea := LogicProducer.CommentAreas.Add;
with CommentArea do
begin
Left := SB.Width div 2; Top := SB.Height div 2;
Width := 100; Height := 100;
AddShape(CommentArea);
end;
end;
```

```
procedure TchShapePlus.Paint;
var
i: integer;
R: TRect;
str: string;
begin
inherited;

Canvas.Font.Color := clBlue;
Canvas.Font.Style := [fsBold];
str := CommentArea.Text;
R := Bounds(3, 2, Width, Height);
DrawText(Canvas.handle, PChar(str),length(str), R, DT_WORDBREAK);
end;

procedure TchLogicsEditor.cbIgnoreSpacesClick(Sender: TObject);
begin
LogicProducer.IgnoreSpaces := cbIgnoreSpaces.Checked;
end;

procedure TchLogicsEditor.pLeftDockOver(Sender: TObject; Source:
TDragDockObject; X, Y: Integer; State: TDragState; var Accept: Boolean);
begin
Accept := true;
end;

procedure TchLogicsEditor.pLeftUnDock(Sender: TObject; Client: TControl;
NewTarget: TWinControl; var Allow: Boolean);
begin
(Sender as TWinControl).Height := 6;
end;

procedure TchLogicsEditor.pLeftDockDrop(Sender: TObject; Source:
TDragDockObject; X, Y: Integer);
begin
(Sender as TWinControl).Height:=100;
SBar.Top:=1500;
end;

procedure TchLogicsEditor.ToolButton8Click(Sender: TObject);
var i: integer;
begin
if Logics.TraceItem = nil then
begin
Logics.StartAnalyze;
reReslt.Text := '';
end;
tbStop.Enabled :=true;
Logics.AnalyzeStep;
for i := 0 to SB.ControlCount-1 do
if (SB.Controls[i] is TchGroupBoxPlus) then
if TchGroupBoxPlus(SB.Controls[i]).LogicElement.IsTrue then
TchGroupBoxPlus(SB.Controls[i]).Colors.Caption := clGreen;
ShowMessage(Logics.Result);
end;

procedure TchLogicsEditor.tbStopClick(Sender: TObject);
var i: integer;
begin
      Logics.TraceItem := nil;
      tbStop.Enabled :=false;
      for i := 0 to SB.ControlCount-1 do
      if (SB.Controls[i] is TchGroupBoxPlus) then
      if TchGroupBoxPlus(SB.Controls[i]).LogicElement.IsTrue then
      TchGroupBoxPlus(SB.Controls[i]).Colors.Caption := clBtnShadow;
end;
```

```
procedure TchLogicsEditor.TabSet1Change(Sender: TObject; NewTab: Integer; var
AllowChange: Boolean);
begin
      PC.ActivePageIndex := NewTab;
end;
end.
```

Розроблена бібліотека файл HShape.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit HShape;

interface
{$I DEF.INC}
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls, glTypes, glUtils, glCommCl;

type
TRGNCombineMode = ( cmAND, cmCOPY, cmDIFF, cmOR, cmXOR );
THoleShapeType = (stRectangle, stSquare, stRoundRect, stRoundSquare, stEllipse,
stCircle);
TchHoleShape = class(TGraphicControl)
private
FShape            : THoleShapeType;
FShapeBitmap      : TBitmap;
FBevelInner       : TPanelBevel;
FBevelOuter       : TPanelBevel;
FBoldInner        : boolean;
FBoldOuter        : boolean;
FRectEllipse      : TPointClass;
FBevelOffset      : integer;
fNeedUpdateRGN    : boolean;
fDestroyed        : boolean;
fRunOnce          : boolean;
fNeedRebuildBitmapShape : boolean;
OldX,OldY,OldW,OldH : integer;
procedure SetEnabled( Value: boolean );
procedure SetEnabledDT( Value: boolean );
procedure SetShape( Value: THoleShapeType );
procedure SetShapeBitmap( Value: TBitmap );
procedure SetBevelInner( Value: TPanelBevel );
procedure SetBevelOuter( Value: TPanelBevel );
procedure SetBoldInner( Value: boolean );
procedure SetBoldOuter( Value: boolean );
procedure SetCombineMode( Value: TRGNCombineMode );
procedure SetBevelOffset( Value: integer );

procedure Update_;
procedure CalcRGNs;
procedure SmthChanged(Sender: TObject);
procedure SayAllDTEnabledState( EnabledDT: boolean );
protected
procedure Paint; override;
public
RGNOuter, RGNInner      : HRGN;
FCombineMode      : TRGNCombineMode;
FEnabledDT        : boolean;
FEnabled          : boolean;
constructor Create( AOwner : TComponent ); override;
destructor Destroy; override;
procedure UpdateRGN;
procedure Loaded; override;
published
property Align;
property ShowHint;
property ParentShowHint;
property PopupMenu;
property Visible;
property Enabled: boolean read FEnabled write SetEnabled
default true;
property EnabledAllInDesignTime: boolean read FEnabledDT write SetEnabledDT
default true;
```

```
property Shape: THoleShapeType read FShape write SetShape
default stEllipse;
property BevelInner: TPanelBevel read FBevelInner write SetBevelInner
default bvNone;
property BevelOuter: TPanelBevel read FBevelOuter write SetBevelOuter
default bvLowered;

property BevelInnerBold: boolean read FBoldInner write SetBoldInner
default true;
property BevelOuterBold: boolean read FBoldOuter write SetBoldOuter
default true;
property CombineMode: TRGNCombineMode read FCombineMode write SetCombineMode
default cmDIFF;
property BevelOffset: integer read FBevelOffset write SetBevelOffset
default 0;
property RectEllipse: TPointClass read FRectEllipse write FRectEllipse;
property ShapeBitmap: TBitmap read FShapeBitmap write SetShapeBitmap;
end;

procedure Register;

implementation
const
aCombMode : array[0..4] of integer = (RGN_AND, RGN_COPY, RGN_DIFF, RGN_OR,
RGN_XOR );
procedure Register;
begin
RegisterComponents('Proba', [TchHoleShape]);
end;
constructor TchHoleShape.Create( AOwner : TComponent );
begin
inherited;
FShapeBitmap:=TBitmap.Create;
FEnabled := (Owner is TWinControl);

ControlStyle := ControlStyle - [csOpaque];
FEnabledDT:=FEnabled;
fDestroyed:=false;
FRectEllipse:=TPointClass.Create;
FRectEllipse.x:=30; FRectEllipse.y:=30;
FRectEllipse.OnChanged:=SmthChanged;
FShape:=stEllipse;
FBevelOuter:=bvLowered;
FBevelInner:=bvNone;
FCombineMode:=cmDIFF;
FBoldInner:=true; FBoldOuter:=true;
FRectEllipse.y:=45; FRectEllipse.x:=45;
FBevelOffset:=0;
Width:=112; Height:=112;
fNeedUpdateRGN:=false;
fRunOnce:=true;
end;

destructor TchHoleShape.Destroy;
begin
FShapeBitmap.Free;
FRectEllipse.Free;
if not (csDestroying in Owner.ComponentState) then
begin FEnabledDT:=false; FEnabled:=false; UpdateRGN(); end;
inherited;
end;

procedure TchHoleShape.Paint;
var
r             :TRect;
H,W,EH,EW,i   :integer;

procedure DrawShape( Bevel: TPanelBevel; fBold, fRect: boolean );
procedure SetPenAndBrush( c: Tcolor );
```

```
begin
Canvas.Pen.Color:=c;
if fRect and ((EW and EH)=0) then
begin Canvas.Brush.Style:=bsClear; end
else begin Canvas.Brush.Color:=c; end
end;
begin
Canvas.Brush.Style:=bsClear;//bsSolid bsClear
i := integer( fBold );
with Canvas do
case Bevel of
bvLowered:
begin
SetPenAndBrush( clBtnHighlight );
if fRect then RoundRect( R.Left, R.Top, R.Right, R.Bottom, EW, EH)
else Ellipse( R.Left, R.Top, R.Right, R.Bottom );
SetPenAndBrush( clBtnShadow );
if fRect then RoundRect( R.Left, R.Top, R.Right-1, R.Bottom-1, EW, EH)
else Ellipse( R.Left, R.Top, R.Right-1, R.Bottom-1 );
if FBold then begin
SetPenAndBrush( cl3DDkShadow );
if fRect then RoundRect( R.Left+1, R.Top+1, R.Right-1, R.Bottom-1, EW, EH)
else Ellipse( R.Left+1, R.Top+1, R.Right-1, R.Bottom-1 );
end;
InflateRect( R, -1, -1 ); inc(R.Left,i); inc(R.Top,i);
end;
bvRaised:
begin
SetPenAndBrush( clBtnHighlight );
if fRect then RoundRect( R.Left, R.Top, R.Right, R.Bottom, EW, EH)
else Ellipse( R.Left, R.Top, R.Right, R.Bottom );
if FBold then begin
SetPenAndBrush( cl3DDkShadow );
if fRect then RoundRect( R.Left+1, R.Top+1, R.Right, R.Bottom, EW, EH)
else Ellipse( R.Left+1, R.Top+1, R.Right, R.Bottom );
end;
SetPenAndBrush( clBtnShadow );
if fRect then RoundRect( R.Left+1, R.Top+1, R.Right-i, R.Bottom-i, EW, EH)
else Ellipse( R.Left+1, R.Top+1, R.Right-i, R.Bottom-i );
InflateRect( R, -1, -1 ); dec(R.Right,i); dec(R.Bottom,i);
end;
else
begin
Brush.Color:=clBlack;
FrameRect( Rect(Left, Top, Left+W, Top+H) );
end;
end;
SetPenAndBrush( clBtnFace );

end;

begin
fNeedUpdateRGN := fNeedUpdateRGN
or(OldX<>Left)or(OldY<>Top)or(OldW<>Width)or(OldH<>Height);

if fNeedUpdateRGN then UpdateRGN();
OldX:=Left; OldY:=Top; OldW:=Width; OldH:=Height;

if IsItAFilledBitmap( FShapeBitmap ) then
begin
BitBlt( Canvas.handle, -1,-1, Width, Height, FShapeBitmap.Canvas.handle, 0,0,
SRCCopy);
exit;
end;

case FShape of
stRectangle, stRoundRect, stEllipse:
begin H:=Height; W:=Width; end
else
```

```
begin H:=min(Height,Width); W:=H; end;
end;
R := Bounds( 0, 0, W, H );
with Canvas do
case FShape of
stRectangle, stSquare, stRoundRect, stRoundSquare:
begin
if (FShape = stRectangle)or(FShape = stSquare) then
begin EW:=0; EH:=0; end;
if (FShape = stRoundRect)or(FShape = stRoundSquare) then
begin EW:=FRectEllipse.x; EH:=FRectEllipse.y; end;
DrawShape( FBevelOuter, FBoldOuter, true );
InflateRect( R, -FBevelOffset, -FBevelOffset );
DrawShape(FBevelInner, FBoldInner, true );
Pen.Color:=clBtnFace;
Rect( R.Left, R.Top, R.Right, R.Bottom );
end;
stEllipse, stCircle:
begin
DrawShape( FBevelOuter, FBoldOuter, false );
InflateRect( R, -FBevelOffset, -FBevelOffset );
DrawShape(FBevelInner, FBoldInner, false );
end;
end;
end;
//-------------------------------------------------------
procedure TchHoleShape.CalcRGNs;
var
H, W, xOffs, yOffs      :integer;
R                 :TRect;

BmpInfo           :Windows.TBitmap;
BorderStyle: TFormBorderStyle;
procedure CalcShape( Bevel: TPanelBevel; fBold: boolean );
var
i: integer;
begin
i := integer( fBold );
case Bevel of
bvLowered: begin InflateRect( R, -1, -1 ); inc(R.Left,i); inc(R.Top,i); end;
bvRaised: begin InflateRect( R, -1, -1 ); dec(R.Right,i); dec(R.Bottom,i); end;
end;
end; procedure CalcBmpRgn(var rgn: HRGN );
var
i,j   :integer;
rgn2: HRGN;
TransparentColor: TColor;
begin
TransparentColor := FShapeBitmap.Canvas.Pixels[0, FShapeBitmap.Height-1];
for j:=0 to FShapeBitmap.Height do
for i:=0 to FShapeBitmap.Width do
begin
if FShapeBitmap.Canvas.Pixels[i,j] <> TransparentColor then continue;
RGN2 := CreateRectRgn(i, j, i+1, j+1);
CombineRgn( RGN, RGN2, RGN, RGN_OR );
DeleteObject( RGN2 );
end;
end;
begin
if not FShapeBitmap.Empty then
begin
if fNeedRebuildBitmapShape then} with FShapeBitmap do
begin
GetObject( FShapeBitmap.Handle, sizeof(Windows.TBitmap), @BmpInfo );
if RGNOuter <> 0 then DeleteObject( RGNOuter );
if RGNInner <> 0 then DeleteObject( RGNInner );
RGNInner := CreateRectRgn(0, 0, 0, 0);
CalcBmpRgn(RGNInner);
fNeedRebuildBitmapShape := false;
```

```
end;
end
else
begin
case FShape of
stRectangle, stRoundRect, stEllipse:
begin H:=Height; W:=Width; end
else
begin H:=min(Height,Width); W:=H; end;
end;
R := Bounds( 0, 0, W, H );
if RGNOuter <> 0 then DeleteObject( RGNOuter );
if RGNInner <> 0 then DeleteObject( RGNInner );

if FBevelOffset <> 0 then
begin
CalcShape( FBevelOuter, FBoldOuter );
OffsetRect(R,1,1);
end;
case FShape of
stRectangle, stSquare:
RGNOuter := CreateRectRgn( R.Left, R.Top, R.Right, R.Bottom );
stRoundRect, stRoundSquare:
RGNOuter := CreateRoundRectRgn( R.Left, R.Top, R.Right, R.Bottom,
FRectEllipse.x, FRectEllipse.y );
stEllipse, stCircle:
RGNOuter := CreateEllipticRgn( R.Left, R.Top, R.Right, R.Bottom );
end;
if FBevelOffset=0 then CalcShape( FBevelOuter, FBoldOuter );
InflateRect( R, -FBevelOffset, -FBevelOffset );
if FBevelOffset=0 then CalcShape( FBevelInner, FBoldInner )
else OffsetRect(R,-1,-1);
case FShape of
stRectangle, stSquare:
RGNInner := CreateRectRgn( R.Left+1, R.Top+1, R.Right+1, R.Bottom+1 );
stRoundRect, stRoundSquare:
RGNInner := CreateRoundRectRgn( R.Left+1, R.Top+1, R.Right+2, R.Bottom+2,
FRectEllipse.x, FRectEllipse.y );
stEllipse, stCircle:
RGNInner:=CreateEllipticRgn(R.Left+1,R.Top+1, R.Right+2, R.Bottom+2 );
end;
end;

if Owner is TForm then
begin
if csDesigning in ComponentState then BorderStyle := bsSizeable
     else BorderStyle := TForm(Owner).BorderStyle;
case BorderStyle of
bsSizeable:
begin
xOffs := GetSystemMetrics(SM_CXFRAME)-1;
yOffs := GetSystemMetrics(SM_CYFRAME)-1;
inc( yOffs, GetSystemMetrics(SM_CYCAPTION) );
end;
bsDialog:
begin
xOffs := GetSystemMetrics(SM_CXDLGFRAME)-1;
yOffs := GetSystemMetrics(SM_CYDLGFRAME)-1;
inc( yOffs, GetSystemMetrics(SM_CYCAPTION) );
end;
bsSingle:
begin
xOffs := GetSystemMetrics(SM_CXBORDER);
yOffs := GetSystemMetrics(SM_CYBORDER);
inc( yOffs, GetSystemMetrics(SM_CYCAPTION) );
end;
bsToolWindow:
begin
```

```
xOffs := GetSystemMetrics(SM_CXBORDER);
yOffs := GetSystemMetrics(SM_CYBORDER);
inc( yOffs, GetSystemMetrics(SM_CYSMCAPTION) );
end;
bsSizeToolWin:
begin
xOffs := GetSystemMetrics(SM_CXSIZEFRAME);
yOffs := GetSystemMetrics(SM_CYSIZEFRAME);
inc( yOffs, GetSystemMetrics(SM_CYSMCAPTION) );
end;
else
begin
xOffs := -1;
yOffs := -1;
end;
end;

OffsetRgn( RGNInner, Left+xOffs, Top+yOffs );
OffsetRgn( RGNOuter, Left+xOffs, Top+yOffs );
end;

fRunOnce:=false;
end;
//-------------------------------------------------------
procedure TchHoleShape.SayAllDTEnabledState( EnabledDT: boolean );
var
i: integer;
begin
for i:=0 to TWinControl(Owner).ControlCount-1 do with TWinControl(Owner) do
begin
if (Controls[i] is TchHoleShape) then
begin
TchHoleShape(Controls[i]).FEnabledDT := EnabledDT;
end;
end;

end;
//-------------------------------------------------------
procedure TchHoleShape.UpdateRGN;
var
i: integer;
NewRGN: HRGN;
begin
if not(Owner is TWinControl) then exit;
NewRGN := CreateRectRgn( 0, 0, 2000, 1000 );

for i:=0 to TWinControl(Owner).ControlCount-1 do with TWinControl(Owner) do
begin
if Controls[i] is TchHoleShape then
with TchHoleShape(Controls[i])do
if ((csDesigning in ComponentState)and FEnabledDT)
or ((not(csDesigning in ComponentState))and FEnabled) then
begin
CalcRGNs;
CombineRgn( NewRGN, NewRGN, RGNInner, aCombMode[ integer(FCombineMode) ] )
end;
end;

SetWindowRgn( TWinControl(Owner).Handle, NewRGN, true );
fNeedUpdateRGN:=false;
end;

procedure TchHoleShape.Update_;
begin
      if csLoading in ComponentState then exit;
      UpdateRGN();
      Refresh;
end;
procedure TchHoleShape.SmthChanged(Sender: TObject);
```

```
      begin
      Update_;
end;
procedure TchHoleShape.SetEnabled( Value: boolean );
begin
      if (FEnabled = Value)or not(Owner is TWinControl) then exit;
      FEnabled := Value; Update_;
end;
procedure TchHoleShape.SetEnabledDT( Value: boolean );
begin
      if (FEnabledDT = Value)or not(Owner is TWinControl) then exit;
      FEnabledDT := Value; SayAllDTEnabledState( FEnabledDT );
      Update_;
end;
procedure TchHoleShape.SetShape( Value: THoleShapeType );
begin
      if FShape = Value then exit;
      FShape := Value; Update_;
end;
procedure TchHoleShape.SetShapeBitmap( Value: TBitmap );
begin
      if FShapeBitmap = Value then exit;
      fNeedRebuildBitmapShape := true;
      FShapeBitmap.Assign(Value);
      if Assigned(FShapeBitmap) then
      begin
            Width := FShapeBitmap.Width;
            Height := FShapeBitmap.Width;
      end;
      Update_();
end;
procedure TchHoleShape.SetBevelInner( Value: TPanelBevel );
begin
      if FBevelInner = Value then exit;
      FBevelInner := Value; Update_;
end;
procedure TchHoleShape.SetBevelOuter( Value: TPanelBevel );
begin
      if FBevelOuter = Value then exit;
      FBevelOuter := Value; Update_;
end;
procedure TchHoleShape.SetBoldInner( Value: boolean );
begin
      if FBoldInner = Value then exit;
      FBoldInner := Value; Update_;
end;
procedure TchHoleShape.SetBoldOuter( Value: boolean );
begin
      if FBoldOuter = Value then exit;
      FBoldOuter := Value; Update_;
end;
procedure TchHoleShape.SetCombineMode( Value: TRGNCombineMode );
begin
      if FCombineMode = Value then exit;
      FCombineMode := Value; Update_;
end;
procedure TchHoleShape.SetBevelOffset( Value: integer );
begin
      if (FBevelOffset = Value)or(Value < 0) then exit;
      if (Value > width-2)or(Value > height-2) then Value:=min(width,height)-2;
      FBevelOffset := Value; Update_;
end;
procedure TchHoleShape.Loaded;
begin
      inherited;
      fNeedRebuildBitmapShape := true;
      UpdateRGN(); Refresh;
end;
end.
```

Розроблена бібліотека файл glPage.pas

```
{Central Ukrainian National Texnical University
Okaievych Yu.O., 2021 year}
unit glPage;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

ComCtrls, CommCtrl, glTypes, glUtils, DrawTab, TabComm, ExtCtrls, glCommCl
const
TCM_SETTEXTCOLOR  =     (TCM_FIRST + 36);
type
TchPageControl = class (TPageControl)
private
FGlyphs           : TImageList;
FSingleGlyph      : boolean;
FTabStyle           : TTabStyle;
FTabSelectedStyle  : TTabStyle;
FWallpaper  : TTabsWallpaper;
FDrawGlyphsOption : TchWallpaperOption;
FLookLikeButtons  : boolean;
FTabsPosition     : TchSide;
FOptions          : TchTabOptions;
FFontDirection    : TchLabelDir;
FOnGetItemColor    : TchOnGetItemColorEvent;
FOnGetItemFontColor: TchOnGetItemColorEvent;
FOnGetGradientColors: TchOnGetGradientColors;

GlyphsChangeLink  : TChangeLink;
DrawTabStr  : TDRAWTABSTRUCT;
GlyphTmpBitmap    : TBitmap;
FontNormal         : TFont;
FontSelected       : TFont;
fNotFirst   : boolean;
aTabColors         : array[0..100] of TColor;

function GeTchyphIndex(Index: Integer): Integer;
procedure SeTchyphIndex(Index: Integer; imgIndex: Integer);
procedure SeTchyphs(Value: TImageList);
procedure SetSingleGlyph(Value: boolean);
procedure SetDrawGlyphsOption(Value: TchWallpaperOption);
procedure SetLookLikeButtons(Value: boolean);
procedure SetTabsPosition(Value: TchSide);
procedure SetOptions(Value: TchTabOptions);
procedure SetFontDirection(Value: TchLabelDir);
function GetFont: TFont;
procedure SetFont(Value: TFont);
function GetTabColor( Index: integer ): TColor;
procedure SetTabColor(Index: integer; Value: TColor);
procedure SmthChanged(Sender: TObject);
procedure FontsChanged(Sender: TObject);
procedure DrawItem(lpDrawItemStr:PDRAWITEMSTRUCT);
procedure CNDrawItem(var Message: TWMDrawItem); message CN_DRAWITEM;
procedure CMFontChanged(var Message: TMessage); message CM_FONTCHANGED;
procedure SetTabStyle(const Value: TTabStyle);
procedure SetTabSelectedStyle(const Value: TTabStyle);
protected
procedure GlyphsListChanged(Sender: TObject);
procedure WndProc(var Message: TMessage); override;
procedure CreateParams(var Params: TCreateParams); override;
procedure Loaded; override;
procedure Notification(AComponent: TComponent; Operation: TOperation); override;
public
fSupressDraw       : boolean;
procedure RemakeFonts;
constructor Create (AOwner: TComponent); override;
```

```
destructor Destroy; override;
property GlyphIndex[Index: Integer]: Integer read GeTchyphIndex write
SeTchyphIndex;
property TabColor[ Index: integer ]: TColor read GetTabColor write SetTabColor;
property GlyphState[Index: Integer]: Integer read GeTchyphState write
SeTchyphState;
published
property Glyphs: TImageList read FGlyphs write SeTchyphs;
property SingleGlyph: boolean read FSingleGlyph write SetSingleGlyph
default false;
property TabStyle: TTabStyle read FTabStyle write SetTabStyle;
property TabSelectedStyle: TTabStyle read FTabSelectedStyle write
SetTabSelectedStyle;
property Wallpaper: TTabsWallpaper read FWallpaper write FWallpaper;
property DrawGlyphsOption: TchWallpaperOption
read FDrawGlyphsOption write SetDrawGlyphsOption default fwoNone;
property LookLikeButtons: boolean read FLookLikeButtons write SetLookLikeButtons
default false;
property TabsPosition: TchSide read FTabsPosition write SetTabsPosition
default fsdTop;
property Options: TchTabOptions read FOptions write SetOptions;
property FontDirection: TchLabelDir
read FFontDirection write SetFontDirection default fldLeftRight;
property Font: TFont read GetFont write SetFont;
property OnGetItemColor: TchOnGetItemColorEvent read FOnGetItemColor write
FOnGetItemColor;
property OnGetItemFontColor: TchOnGetItemColorEvent read FOnGetItemFontColor
write FOnGetItemFontColor;
property OnGetGradientColors: TchOnGetGradientColors read FOnGetGradientColors
write FOnGetGradientColors;
end;

procedure Register;

implementation
const FontDirs:array[TchSide]of TchLabelDir
=(fldDownUp,fldLeftRight, fldUpDown, fldLeftRight );
procedure Register;
begin
RegisterComponents('Proba', [TchPageControl]);
end;
constructor TchPageControl.Create (AOwner: TComponent);
begin
inherited Create(AOwner);
TabStop:=false;
FTabStyle        := TTabStyle.Create(self);
with FTabStyle do
begin
BackgrColor := clBtnShadow;
Font.Color := clBtnHighlight;
CaptionHAlign := fhaCenter;
end;
FTabSelectedStyle := TTabStyle.Create(self);
with FTabSelectedStyle do
begin
BackgrColor := clBtnFace;
Font.Color := clBtnText;
CaptionHAlign := fhaCenter;
end;

FWallpaper        := TTabsWallpaper.Create;
FontNormal            := TFont.Create;
FontSelected          := TFont.Create;
DrawTabStr.Font_      := TFont.Create;

FTabStyle.Font.Name:='Arial';
FTabSelectedStyle.Font.Name:='Arial';

GlyphTmpBitmap    := TBitmap.Create;
```

```
GlyphsChangeLink  := TChangeLink.Create;
GlyphsChangeLink.OnChange:=GlyphsListChanged;
DrawTabStr.Gradient := TGradient.Create;
FSingleGlyph:=false;
FDrawGlyphsOption:=fwoNone;
FTabsPosition:=fsdTop;
FOptions:=[ftoAutoFontDirection,ftoExcludeGlyphs];
FFontDirection:=fldLeftRight;
FTabStyle.OnChanged             := SmthChanged;
FTabSelectedStyle.OnChanged   := SmthChanged;
FTabStyle.OnFontChanged := FontsChanged;
FTabSelectedStyle.OnFontChanged := FontsChanged;
FWallpaper.OnChanged          := SmthChanged;
FillMemory( @aTabColors, sizeof(aTabColors), $FF );
end;

destructor TchPageControl.Destroy;
begin
FTabStyle.Free;
FTabSelectedStyle.Free;
GlyphTmpBitmap.Free;
FWallpaper.Free;
GlyphsChangeLink.Free;
FontNormal.Free;
FontSelected.Free;
DrawTabStr.Font_.Free;
if Assigned(DrawTabStr.Gradient) then DrawTabStr.Gradient.Free;
inherited;
end;

procedure TchPageControl.SmthChanged;
begin Invalidate; end;


procedure TchPageControl.FontsChanged;
begin RemakeFonts; Invalidate; end;

procedure TchPageControl.CreateParams(var Params: TCreateParams);
const PosStyles : array[TchSide]of DWORD =
( TCS_VERTICAL, 0, TCS_VERTICAL or TCS_RIGHT, TCS_BOTTOM or TCS_SCROLLOPPOSITE
or TCS_BUTTONS );
begin
inherited CreateParams(Params);
with Params do
begin
if LookLikeButtons then Style := Style or TCS_BUTTONS;
Style := Style or TCS_OWNERDRAWFIXED or PosStyles[FTabsPosition];
end;
end;

procedure TchPageControl.Loaded;
begin
inherited Loaded; RemakeFonts;
if Assigned(Wallpaper.Bitmap)and(not Wallpaper.Bitmap.Empty)
then Wallpaper.bmp := Wallpaper.Bitmap;
end;

procedure TchPageControl.Notification(AComponent: TComponent; Operation:
TOperation);
begin
inherited Notification(AComponent, Operation);
if Assigned(Wallpaper)and(AComponent = Wallpaper.Image) and (Operation =
opRemove) then Wallpaper.Image := nil;
if (AComponent = FGlyphs) and (Operation = opRemove) then Glyphs := nil;
end;

procedure TchPageControl.CNDrawItem(var Message: TWMDrawItem);
begin
DrawItem(Pointer(Message.DrawItemStruct));
```

```
end;

procedure TchPageControl.WndProc(var Message: TMessage);
var
GlyphID:integer;
begin
inherited WndProc(Message);
with Message do
case Msg of
TCM_INSERTITEM:
begin  result:=0;
if not Assigned(FGlyphs) then exit;
GlyphID:=-1;
if FSingleGlyph then GlyphID:=0
else if wParam < FGlyphs.Count then GlyphID:=wParam;
if GlyphID=-1 then exit;
TTCItem(Pointer(Message.lParam)^).iImage:=GlyphID;
TTCItem(Pointer(Message.lParam)^).Mask:=TCIF_IMAGE;

SendMessage( handle, TCM_SETITEM, wParam, lParam );
end;
TCM_DELETEITEM: begin end;
TCM_DELETEALLITEMS: begin end;
end;
end;

procedure TchPageControl.GlyphsListChanged(Sender: TObject);
begin
if HandleAllocated then SendMessage(Handle, TCM_SETIMAGELIST, 0,
Longint(TImageList(Sender).Handle));
end;

procedure TchPageControl.DrawItem(lpDrawItemStr:PDRAWITEMSTRUCT);
var
FontColor: TColor;
begin
if fSupressDraw then exit;
with lpDrawItemStr^ do
if CtlType=ODT_TAB then
begin
DrawTabStr.lpDrawItemStr := lpDrawItemStr;
DrawTabStr.Caption:=Tabs[ItemID];

if GlyphIndex[ItemID]<>-1 then
begin
FGlyphs.GetBitmap( GlyphIndex[ItemID], GlyphTmpBitmap );
DrawTabStr.Glyph:=GlyphTmpBitmap;
end else DrawTabStr.Glyph:=nil;

if (itemState and ODS_DISABLED)<>0 then
begin
DrawTabStr.BoxStyle := FTabStyle;
DrawTabStr.Font_.Assign(FontNormal);
end
else if (itemState and ODS_SELECTED)<>0 then begin
DrawTabStr.BoxStyle := FTabSelectedStyle;
DrawTabStr.Font_.Assign(FontSelected);
end
else begin
DrawTabStr.BoxStyle := FTabStyle;
DrawTabStr.Font_.Assign(FontNormal);
end;

if Assigned(OnGetItemFontColor) then
begin
OnGetItemFontColor( self, ItemID, FontColor );
DrawTabStr.Font_.Color := FontColor;
end;
DrawTabStr.GlyphOption := FDrawGlyphsOption;
```

```
DrawTabStr.Wallpaper:=FWallpaper;
DrawTabStr.ClientR:=ClientRect;
DrawTabStr.TabsCount:=Tabs.Count;
DrawTabStr.fButton:=LookLikeButtons;
DrawTabStr.Position:=TabsPosition;
DrawTabStr.Options:=Options;
DrawTabStr.FontDirection:=FontDirection;

if Assigned(OnGetGradientColors) then OnGetGradientColors( self, ItemID,
DrawTabStr.Gradient);

if Assigned(OnGetItemColor) then OnGetItemColor( self, ItemID,
DrawTabStr.BackgrColor_) else
if aTabColors[ItemID] <> -1 then DrawTabStr.BackgrColor_ := aTabColors[ItemID]
else DrawTabStr.BackgrColor_ := DrawTabStr.BoxStyle.BackgrColor;
DrawOwnTab( DrawTabStr );
end;
end;

procedure TchPageControl.CMFontChanged(var Message: TMessage);
begin
inherited;
if ftoInheriteTabFonts in Options then
begin
FTabStyle.Font.Assign(inherited Font);
FTabSelectedStyle.Font.Assign(inherited Font);
Disabled.Assign(inherited Font);
RemakeFonts;
end;
end;

procedure TchPageControl.RemakeFonts;
const
RadianEscapments:array [TchlabelDir] of integer = (0,-1800,-900,900);
begin
if csReading in ComponentState then exit;
if fNotFirst then DeleteObject( FTabStyle.Font.Handle );
fNotFirst:=true;

FontNormal.Handle := CreateRotatedFont( FTabStyle.Font,
RadianEscapments[FFontDirection]);
FontNormal.Color := FTabStyle.Font.Color;
FontSelected.Handle := CreateRotatedFont( FTabSelectedStyle.Font,
RadianEscapments[FFontDirection]);
FontSelected.Color := FTabSelectedStyle.Font.Color;
end;

procedure TchPageControl.SeTchyphs(Value: TImageList);
var i: integer;
label SkipAutoGlypsSet;
begin
if Assigned(FGlyphs) then  FGlyphs.UnregisterChanges(GlyphsChangeLink);
FGlyphs := Value;
if Assigned(FGlyphs) then
begin
FGlyphs.RegisterChanges(GlyphsChangeLink);
SendMessage(Handle, TCM_SETIMAGELIST, 0, Longint(FGlyphs.Handle));
for i:=0 to min( Tabs.Count-1, FGlyphs.Count-1) do

if GlyphIndex[i]<>-1 then goto SkipAutoGlypsSet;
SetSingleGlyph(FSingleGlyph);
SkipAutoGlypsSet:
end
else SendMessage(Handle, TCM_SETIMAGELIST, 0, Longint(0));
end;

procedure TchPageControl.SeTchyphIndex( Index: Integer; imgIndex: Integer);
var
r     : TRect;
```

```
Item  : TTCItem;
begin
Item.iImage := imgIndex;
Item.mask := TCIF_IMAGE;
SendMessage( Handle, TCM_SETITEM, Index, Longint(@Item) );
SendMessage( Handle, TCM_GETITEMRECT, Index, Longint(@r) );
InvalidateRect( Handle, @r, true );
end;

function TchPageControl.GeTchyphIndex( Index: Integer ): Integer;
var
imgItem: TTCItem;
begin
if Assigned(FGlyphs) then
begin
imgItem.mask := TCIF_IMAGE;
SendMessage( Handle, TCM_GETITEM, Index, Longint(@imgItem) );
Result := imgItem.iImage;
end
else Result := -1;
end;

procedure TchPageControl.SetSingleGlyph(Value: boolean);
var i: integer;
begin
FSingleGlyph:=Value;
if (Tabs=nil)or(FGlyphs=nil) then exit;
if FSingleGlyph then
for i:=0 to Tabs.Count-1 do GlyphIndex[i]:=0
else
for i:=0 to Tabs.Count-1 do
if FGlyphs.Count >= i then GlyphIndex[i]:=i else break;
end;

procedure TchPageControl.SetDrawGlyphsOption(Value: TchWallpaperOption);
begin
if FDrawGlyphsOption = Value then exit;
FDrawGlyphsOption := Value; Invalidate;
end;

procedure TchPageControl.SetLookLikeButtons(Value: boolean);
begin

if FLookLikeButtons = Value then exit;
FLookLikeButtons := Value;
RecreateWnd;
end;

procedure TchPageControl.SetTabsPosition(Value: TchSide);
begin
if FTabsPosition = Value then exit;
FTabsPosition := Value; RecreateWnd;
if (ftoAutoFontDirection in FOptions)and not(csLoading in ComponentState) then
FontDirection := FontDirs[TabsPosition];
end;

procedure TchPageControl.SetOptions(Value: TchTabOptions);
begin
if FOptions = Value then exit; FOptions := Value;
if ftoAutoFontDirection in FOptions then
FontDirection := FontDirs[TabsPosition];
Invalidate;
end;

procedure TchPageControl.SetFontDirection(Value: TchlabelDir);
begin
if FFontDirection = Value then exit;
FFontDirection := Value; RemakeFonts;
Invalidate;
```

```
end;

function TchPageControl.GetFont: TFont;
begin Result := inherited Font; end;

procedure TchPageControl.SetFont(Value: TFont);
begin
inherited Font := Value;
if ftoInheriteTabFonts in Options then
begin
FTabStyle.Font.Assign(inherited Font);
FTabSelectedStyle.Font.Assign(inherited Font);
end;
end;


function TchPageControl.GetTabColor( Index: integer ): TColor;
begin
if Index<100 then Result := aTabColors[Index] else Result := -1;
end;

procedure TchPageControl.SetTabColor(Index: integer; Value: TColor);
var
TCItem: TTCItem;
begin
if (Index<100)and(TabColor[Index] <> Value) then aTabColors[Index] := Value
else exit;
if not fSupressDraw then
begin
Repaint;
TCItem.mask := TCIF_TEXT;
TCItem.pszText := PChar(Tabs[Index]);
SendMessage( Handle, TCM_SETITEM, Index, Longint(@TCItem));
end;
end;

procedure TchPageControl.SetTabStyle(const Value: TTabStyle);
begin
FTabStyle := Value;
RemakeFonts;
end;

procedure TchPageControl.SetTabSelectedStyle(const Value: TTabStyle);
begin
FTabSelectedStyle := Value;
RemakeFonts;
end;

end.
```