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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Микитенко Д.Ю.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Микитенку Даниїлу Юрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій*

2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій*

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

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

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Дрєєв О.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Микитенко Д.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Микитенко Д.Ю. Програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

Метою розробки є програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

Результат роботи – програмна реалізація системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерна інженерія, керування параметрами структур корпоративних мереж

ABSTRACT

Mykytenko D.Yu. Software for managing the parameters of corporate network structures using cloud technologies. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software was developed, which is intended for the system of managing the parameters of corporate network structures using cloud technologies.

The purpose of the development is software for the management of parameters of corporate network structures using cloud technologies.

The result of the work is the software implementation of the parameter management system of corporate network structures using cloud technologies.

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

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

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10 environment.

Keywords: computer engineering, management of parameters of corporate network structures

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ПЗ	–	програмне забезпечення
QoS	–	механізми контролю й керування якістю

ВСТУП

Актуальність теми. Важливою умовою підвищення конкурентоспроможності української економіки в умовах ринку є впровадження на вітчизняних підприємствах інформаційних технологій (ІТ). Основою інфраструктури сучасних підприємств є корпоративні мережі передачі даних, що забезпечують транспорт для переносу інформації між різними додатками інформаційних систем. На них опираються підсистеми телефонії, охорони, відеоспостереження й ін.

У цей час на зміну спеціалізованим мережам (телефонним, охоронним і ін.) приходять мультисервісні корпоративні мережі. Нові технології (NGN і MPLS) дозволяють створювати ефективні, надійні й безпечні мережі будь-якого масштабу. Для забезпечення зростаючих потреб українських підприємств вимоги до мультисервісної корпоративної мережі, як до середовища передачі інформації для забезпечення роботи різних додатків, безупинно зростають. Великого значення набуває час реакції додатків – при динамічному ринку для успішної боротьби з конкурентами рішення необхідно приймати в реальному масштабі часу, що вимагає відповідної організації корпоративної мережі і її додатків. Вимоги роботи в реальному часі стали для багатьох підприємств нагальною потребою й одним з основних вимог, пропонованих до корпоративних мереж і корпоративних додатків.

У той же час у реальній корпоративній мережі забезпечити гарний час реакції особливо складно – цьому заважає висока інтенсивність і розмаїтість потоків даних, створюваних сотнями й тисячами співробітників корпорації, необхідність робити пошук даних у базах великої розмірності, складна взаємодія розподілених додатків, невисока швидкість глобальних ліній зв'язку між відділеннями корпорації, зі швидкості взаємодії в шлюзах, що погодять неоднорідні компоненти різних підмереж.

Як показує світовий досвід, рішення зазначених проблем неможливо без створення й впровадження ефективних систем керування, що дозволяють підтримувати на заданому рівні мережні ресурси, необхідні для надання якісних послуг. На зазначені цілі у світі витрачатися до 20% від вартості устаткування корпоративних мереж передачі даних. При цьому необхідно враховувати, що в сучасних мулитисервісних корпоративних мережах використовується складне багатофункціональне комунікаційне устаткування, що забезпечує підтримку спеціальних механізмів контролю й керування якістю – QoS, розвинених інструментів реалізації корпоративної політики інформаційної безпеки.

Комплексне рішення завдань керування корпоративною мережею представляє складну наукову проблему, пов'язану з розробкою науково-обгрунтованих методів створення систем, що забезпечують підтримку заданої якості обслуговування, адміністрування й адаптивного керування.

Найважливішим параметром корпоративної мережі є її структура, що багато в чому визначає характеристики мережі. У зв'язку із цим структура мережі може розглядатися як об'єкт керування, вплив на який дозволяє управляти потоками даних, що є основним завданням керування мережею.

У цей час створена й експлуатується велика кількість потужних систем керування корпоративними мережами, що дозволяє узагальнити результати їхньої роботи й виділити загальні для них переваги й недоліки. Як показує аналіз, часто основні переваги багатьох систем керування – універсальність і багатофункціональність стають у корпоративних системах і їх основними недоліками. Це зв'язано, як правило, з необхідністю враховувати специфіку роботи корпоративної системи, що вимагає відповідних налаштувань корпоративної мережі й методів керування її роботою. Таким чином, існує й постійно заглиблюється розрив між зростаючими універсальними можливостями систем керування й реальних потреб при керуванні, орієнтованому на конкретні додатки.

У зв'язку із цим насущною проблемою є розробка нових концептуальних підходів до керування корпоративними мережами, орієнтованих на рішення заданого набору прикладних завдань, і забезпечення необхідної якості їхнього рішення при застосуванні універсальних багатофункціональних систем керування. Рішення проблеми засноване на розробці підходів до керування корпоративною мережею, що сполучають облік специфіки розв'язуваних завдань і можливості наявних систем керування. Дослідження, спрямовані на створення й теоретичне обґрунтування зазначених підходів актуальні як у цей час, так і на доступну для огляду перспективу розвитку корпоративних мереж.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

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

– Огляд існуючих систем керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

– Дослідження системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

– Програмна реалізація системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації програмного забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій**.** Метою бакалаврського проекту є створення концептуального підходу до керування корпоративними мережами, орієнтованого на ефективне виконання додатків і комплексів, що включає, математичні моделі і правила їхнього застосування для рішення завдань аналізу й формування структури мережі, що виникають при керуванні мережею. Розробка методики практичної реалізації створеного підходу, адаптованої для різних напрямків застосування корпоративних мереж. Для досягнення зазначених цілей під час бакалаврського проектування були вирішені наступні завдання:

– проведено аналіз стану й напрямків розвитку корпоративних мереж;

– проведено аналіз сучасних і перспективних засобів і методів керування корпоративними мережами;

– розроблено дворівневий метод аналізу структури корпоративної мережі, що дозволяє проводити розрахунок параметрів потоків даних у корпоративній мережі, що включає дослідження інформаційної й технічної структур;

– розроблено комплекс математичних моделей для розрахунку параметрів потоків даних в інформаційній і технічній структурах корпоративної мережі;

– розроблено правила формування простору станів і множини керуючих параметрів корпоративної мережі;

– розроблено дворівневий метод керування мережею, заснований на послідовному рішенні завдань налаштування й оперативного керування;

– сформульовано завдання налаштування й завдання оперативного керування корпоративною мережею;

– розроблено математичні моделі для рішення часткових завдань керування: формування структури мережі, розподіл смуги пропущення каналу зв'язку, керування навантаженням серверів;

– розроблено методику застосування математичних моделей при керуванні корпоративною мережею;

– розроблено архітектуру системи керування корпоративною мережею, що включає елементи штучного інтелекту;

– розроблено підсистему прогнозування стану корпоративної мережі в нейромережному логічному базисі;

– розроблено архітектуру апаратно-програмної системи налагодження й випробування програмного забезпечення систем керування корпоративними мережами.

1.2 Область застосування

Областю застосування системи є корпоративні мережі. Корпоративна мережа – це мережа, головним призначенням якої є підтримка роботи конкретного підприємства, що володіє даною мережею. Користувачами корпоративної мережі є тільки співробітники даного підприємства. На відміну від мереж операторів зв'язку, корпоративні мережі, в загальному випадку, не надають послуг стороннім організаціям або користувачам. Залежно від масштабу підприємства, а також від складності і різноманіття вирішуваних завдань розрізняють мережі відділу, мережі кампусу і корпоративні мережі (термін «корпоративні» в даній класифікації набуває вузького значення – мережу великого підприємства). Будь-яка організація – це сукупність взаємодіючих елементів (підрозділів), кожен з яких може мати свою структуру. Елементи зв'язані між собою функціонально, тобто вони виконують окремі види робіт в рамках єдиного бізнес процесу, а також інформаційно, обмінюючись документами, факсами, письмовими і усними розпорядженнями і так далі крім

того, ці елементи взаємодіють із зовнішніми системами, причому їх взаємодія також може бути як інформаційною, так і функціональною. І ця ситуація справедлива практично для всіх організацій, яким би видом діяльності вони не займалися – для урядової установи, банку, промислового підприємства, комерційної фірми і так далі. Такий загальний погляд на організацію дозволяє сформулювати деякі загальні принципи побудови корпоративних інформаційних систем, тобто інформаційних систем в масштабі всієї організації. Корпоративною мережею вважається будь-яка мережа, що працює по протоколу TCP/IP. і використовує комунікаційні стандарти Інтернету, а також сервісні застосування, що забезпечують доставку даних користувачам мережі. Наприклад, підприємство може створити сервер Web для публікації оголошень, виробничих графіків і інших службових документів. Службовці здійснюють доступ до необхідних документів за допомогою засобів (коштів) проглядання Web. Сервери Web корпоративної мережі можуть забезпечити користувачам послуги, аналогічні послугам Інтернету, наприклад роботу з гіпертекстовими сторінками (що містять текст, гіперпосилання, графічні зображення і звукозаписи), надання необхідних ресурсів по запитах клієнтів Web, а також здійснення доступу до баз даних. У цьому керівництві всі служби публікації називаються “Службами Інтернету” незалежно від того, де вони використовуються (у Інтернеті або корпоративній мережі). Корпоративна мережа, як правило, є територіально розподіленою, тобто об'єднуючою офіси, підрозділи і інші структури, що знаходяться на значному віддаленні один від одного. Принципи, по яких будується корпоративна мережа, досить сильно відрізняються від тих, що використовуються при створенні локальної мережі. Це обмеження є принциповим, і при проектуванні.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Майже кожному адміністраторові мережі доводилося відповідати на неприємні дзвінки клієнта або, гірше того, начальника, що бажає з'ясувати, чому та або інша служба функціонує повільно або зовсім зупинена? Інструменти керування мережею допоможуть уникнути подібної ситуації й першим довідатися про несправності мережі. У даному розділі представлені чотири рішення для керування мережею, два з яких – безкоштовні. Одне із цих рішень, швидше за все, підійде для компаній з якою завгодно обчислювальною інфраструктурою й будь-яким бюджетом.

Spiceworks

Відсутність плати виділяє Spiceworks серед інших продуктів, але користувачам прийдеться примиритися з показом у програмі ненав'язливих рекламних оголошень. Якщо реклама дратує або заборонена правилами компанії, від її можна позбутися, заплативши 45 дол. на місяць. Якщо договір укладається на рік, надається невелика знижка (495 дол. замість 540). Інша відмітна риса Spiceworks – повнофункціональна служба підтримки.

Важливу роль грає й активне співтовариство користувачів Spiceworks. Наприклад, можна встановити контакт із місцевими користувачами Spiceworks через службу SpiceCorps або брати участь у традиційних форумах на сайті співтовариства (community.spiceworks.com).

Установити Spiceworks просто. Після завершення процедури установки потрібно створити початковий обліковий запис і пароль. Потім з'являється основний запит відносно початку роботи. Вибрати можна із трьох варіантів: Inventory (інвентаризація), Help Desk (служба підтримки) і Configuration Backup (резервне копіювання конфігурації).

З розділу Inventory запускається IP-сканування мережі. Майстер допоможе підготувати підходящі облікові дані для Windows, UNIX, Apple і інших серверів, доступних через оболонку Secure Shell (SSH), а також для принтерів, комутаторів або інших пристроїв SNMP. Для пошуку, виконання процедури реєстрації й інвентаризації всіх мережних пристроїв у моїй тестовій мережі треба було кілька хвилин.

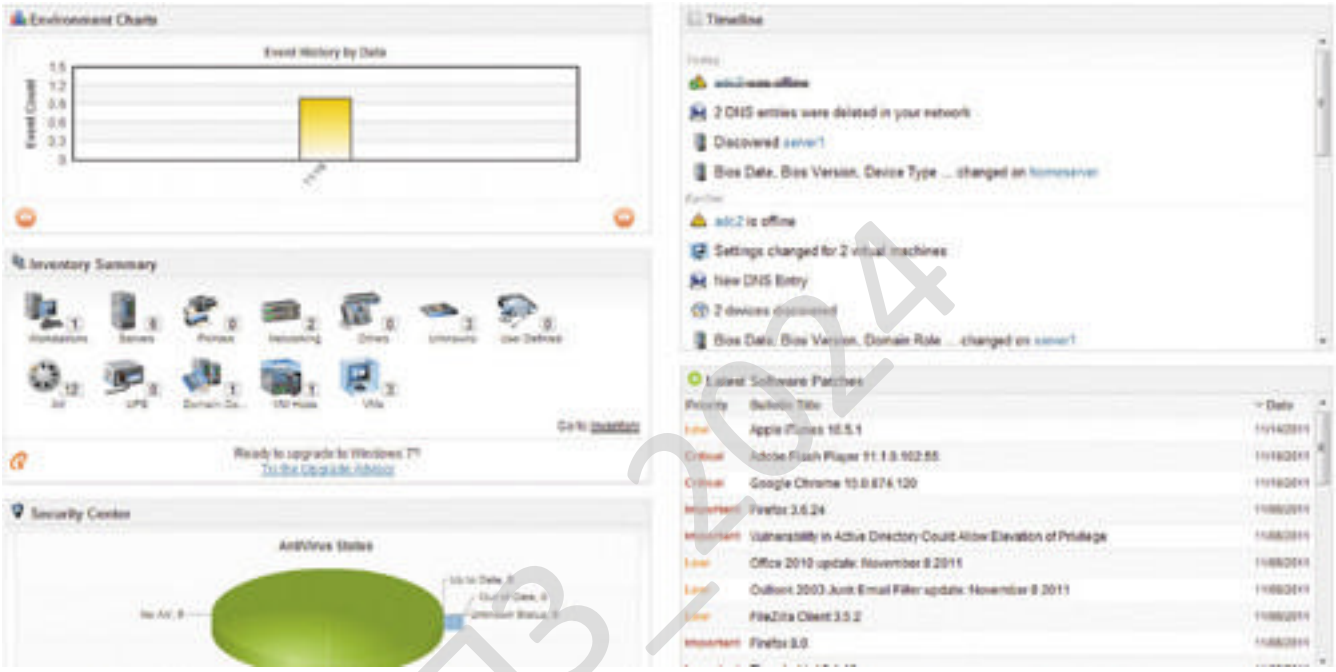

Рисунок 2.1 – Інтерфейс користувача Spiceworks

Відразу після завершення сканування від Spiceworks по електронній пошті прийшло повідомлення з докладним описом результатів. На головній сторінці інвентаризації є поточний журнал із вказівкою виявлених об'єктів і примітками щодо неполадок. Наприклад, Spiceworks виявив сервер VMware ESXi, але не зміг одержати докладні відомості про нього, тому що уведені мною ім'я користувача й пароль були неправильними. По клацанню на сервері ESXi було видано меню, за допомогою якого вдалося усунути проблему перевірки дійсності. У ході чергового сеансу сканування сервер ESXi був виявлений, виконані реєстрація й дані інвентаризації оновлені. Spiceworks не тільки надав інформацію про кожну

віртуальну машину, але й перелічив імена сховищ даних VMware із вказівкою вільного простору в кожному з них. Завдяки інтуїтивно зрозумілому інтерфейсу ви зможете без праці знаходити й виправляти типові помилки конфігурації.

Я якийсь час спостерігала за приладовою панеллю, показаній на рисунку 2.1. За замовчуванням на ній показаний загальний вид мережі й відображаються відомості з 12 основних категорій, у тому числі стан антивірусного захисту, дані Microsoft Exchange Server, зведення інвентаризації, строки завершення дії гарантій і попередження. Інтерфейс користувача відрізняється гнучкістю налаштування, наприклад можна додавати, видаляти й переміщати розділи.

Я також протестувала службу підтримки Spiceworks. Як адміністраторові служби технічної підтримки мені було цікаво подивитися, наскільки це рішення придатне для виконання масштабних завдань. У допомогу початківцем є чотири заздалегідь підготовлених квитки, які дозволяють познайомитися з організацією служби підтримки і її налаштувань. Я швидко переконалася, що служба підтримки елементарна. Наприклад, у ній немає групових функцій, типів запитів і шляхів переходу на більше високий рівень – всі ці можливості дуже важливі для великої IT-організації.

Одне з достоїнств служби підтримки – добування даних для квитків з повідомлень електронної пошти. Користувачі просто відсилають свої запити по електронній пошті, а служба підтримки збирає інформацію з повідомлень і автоматично формує квиток. Інша вдала знахідка – довідковий портал, через який користувач може зайти на веб-сторінку (з перевіркою дійсності Active Directory) і представити свої квитки.

Така служба підтримки ідеальна для компаній з декількома користувачами. Пам’ятайте тільки, що вона погано піддається масштабуванню при збільшенні числа користувачів або адміністраторів.

В Spiceworks представлений 21 шаблоновий звіт, у тому числі про інформацію у квитках служби підтримки, списки комп'ютерів без антивірусних

програм і комп'ютерів з невеликою кількістю вільного простору на дисках. Можна також завантажити підготовлені співтовариством звіти із сайту співтовариства. Моя увага залучили два звіти: в одному втримувалася інформація про використання Exchange (наприклад, час останньої реєстрації користувача, розмір поштової скриньки користувача й загальна кількість об'єктів), а в іншому перераховані локально підключені принтери.

Завершує перелік компонентів цієї повнофункціональної платформи мобільний додаток Spiceworks. Воно дозволяє перевіряти квитки служби підтримки, стежити за попередженнями щодо інвентаризації, читати новітні коментарі в співтоваристві Spiceworks і т.д. Якщо доводиться обслуговувати кілька сайтів, можна підготувати профіль для кожного сайту, щоб не запам'ятовувати комбінації ім'я користувача/пароль/адреса.

Spiceworks – чудовий інструмент. Він повнофункціональний і надається безкоштовно. Навіть при платі за відсутність реклами 45 дол. на місяць або 495 дол. у рік покупка буде дуже вигідною. Це оптимальний варіант для малих компаній, що не бідують у дорогому рішенні.

Foglight Network Management System (NMS)

Другий продукт даного огляду, Foglight Network Management System (NMS) компанії Quest Software, також надається безкоштовно. За допомогою цієї програми з потужною функціональністю можна безкоштовно відслідковувати до 100 пристроїв у мережі. Якщо потрібно контролювати більше пристроїв, купите повну версію Foglight NMS, заплативши 99 дол. за кожний пристрій понад сотню. Тому, наприклад, якщо в мережі 201 пристроїв, прийде заплатити 9 900 дол. (100 х 99 дол.). При цьому ви зможете використовувати три модулі розширення повної версії (Traffic Analysis, IPSLA-VoIP і Remote Site Monitoring with Pollers) з усіма пристроями. Можна також придбати два інших модулі розширення: для моніторингу продуктивності Performance Monitoring і керування конфігурацією Configuration Management.

На жаль, на сайті Quest нелегко знайти інформацію про п'ять модулів розширення. Зацікавленим користувачам варто безпосередньо звернутися в компанію Quest.

До складу як безкоштовної, так і повної версій Foglight NMS входять модулі Network Flow Analyzer Module, Remote Agent Module, VoIP Monitoring Module і Wireless Monitoring Module. У продажі є й додаткові компоненти, у тому числі vFoglight для віртуальних машин.

Як і Spiceworks, Quest користується активною підтримкою співтовариства. У продукті є посилання Community, зв'язана прямо із сайтом співтовариства (www.quest.com/communities), де можна задати питання колегам і навіть довести свої пропозиції до групи розроблювачів. Однак більшість форумів орієнтована строго на продукти Quest, а загальні теми обговорюються не занадто активно. Наприклад, форум SharePoint був порожній, а на форумі Oracle я знайшов усього дев'ять публікацій, більшість із яких з'явилося більше шести місяців назад.

Foglight NMS варто встановлювати на окремому комп'ютері або віртуальній машині. У ході повністю автоматизованого процесу встановлюються необхідні продукти, у тому числі Microsoft.NET Framework 4.0 і SQL Server Compact. Як відзначається в документації по системних вимогах, SQL Server Compact використовується при апробуванні Foglight NMS або завантаженню певної кількості пристроїв. У виробничих умовах необхідно використовувати SQL Server Standard або Enterprise Edition.

Після завершення установки я виконала реєстрацію в Foglight NMS Studio і зареєстрував продукт. На даному етапі варто вибрати безкоштовну версію або ввести ліцензійний ключ, якщо потрібно відслідковувати більше 100 пристроїв. Хоча в мене був ліцензійний ключ, я розглянув безкоштовну версію.

На екрані-заставці приводяться посилання на статті в довідковій системі, у яких описуються кроки по початковому налаштуванню системи моніторингу. Контролювати пристрою можна через протокол SNMP, за допомогою інструментів керування Windows (WMI), розгортаючи вилучені агенти або збираючи дані системного журналу, NetFlow або пасток SNMP.

Я вирішила використовувати SNMP і WMI для моніторингу пристроїв у тестовій мережі. Я нажав кнопку Add Device(s), і на рисунку 2.з'явився майстер, за допомогою якого можна додавати пристрою через функцію виявлення мережі SNMP або розгортання агентів. Функція виявлення мережі SNMP успішно виявила в тестовій мережі всі пристрої. Після цього в меню ліворуч було показано, що 11 пристроїв перебувають у стані Credential Not Set (облікові дані не задані). Клацаючи елементи цього меню, можна одержати список пристроїв, у які програмі Foglight NMS не вдалося виконати вхід. Серед них були сервер Apple, мережні пристрої й Windows Home Server. Із цього списку мені вдалося призначити дійсні облікові дані для кожного пристрою. Інтерфейс настільки інтуїтивно зрозумілий, що мені не довелося користуватися довідковою системою або шукати допомоги в Інтернеті. Після призначення облікових даних Foglight NMS активно відслідковував всю тестову мережу.

Хоча продукт надається безкоштовно, він має у своєму розпорядженні широку функціональність. Зокрема, я ознайомився з функцією створення карти мережі. Спочатку потрібно ввести піктограми, що представляють пристрої, а потім об'єднати їхню конфігурацію, що відповідає фізичній структурі мережі, у графічному інтерфейсі на основі Adobe Flash Player. Можна підготувати більше однієї карти, щоб показати кілька рівнів. Наприклад, одну карту можна накласти на зображення території США. Кількість деталей на кожній наступній карті може збільшуватися.

У ході роботи я помітила, що сервер ESXi був невірно визначений програмою Foglight NMS. Можливо, причина в тому, що в мене була стара версія (3.x). Крім того, в 2010 році ESXi був перейменований в vSphere Hypervisor. Настроїти Foglight NMS вручну для розпізнавання сервера як пристрій VMware було неважко. Після уведення вірних облікових даних програма негайно приступилася до збору статистики про використання процесора, пам'яті й т.д. Є статистика навіть для віртуальних машин.

Якщо в мережі є кілька фізичних сайтів, корисно мати сервер Foglight NMS скрізь. У цьому випадку можна підготувати різні сайти, щоб було простіше організувати відслідковуються устройства, що. Потім ці сайти передають дані центральному серверу Foglight NMS.

Можна призначити попередження, які не тільки сповіщають адміністратора, але й запускають певні дії, наприклад зупиняють службу або запускають спеціальний сценарій.

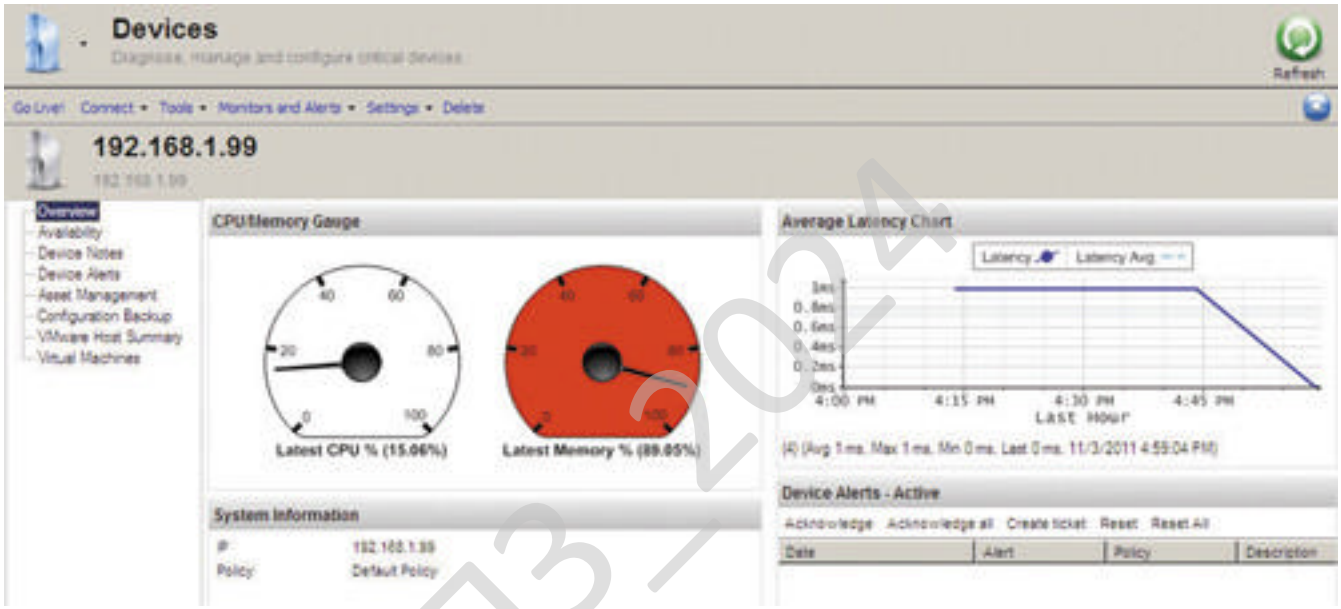

Рисунок 2.2 – Інтерфейс користувача Foglight NMS

В Foglight передбачена можливість збору статистики NetFlow. Вибираючи пункт NetFlow у меню Tools у перший раз, користувач одержує пропозицію завантажити пакет PacketTrap Tool Suite (розмір 10 Мбайт). Це пробна версія, але термін дії – до 2031 року, тому часу для ознайомлення досить. Як повідомляє Келли О'двайер-Мануэль, старший аналітик і фахівець зі зв'язків із громадськістю компанії Quest, у недалекому майбутньому інструмент буде перейменований в Quest Free Network Tools, і пробна версія просто стане безкоштовним додатком. Однак таке додавання не робить сильного враження. Властиво Foglight NMS –

відмінна програма, але спосіб інтеграції PacketTrap із продуктом не можна визнати зробленим.

Foglight NMS – додаток із продуманою структурою й відмінною приладовою панеллю, як показано на рисунку 2.2. Якщо є служба підтримки, який потрібно переглянути стан мережі, то через сайт можна сформувати надання, доступне тільки для читання. Для цього досить додати порт 5053 до ім'я сервера (наприклад, https://Foglight:5053).

У цілому Foglight NMS – інтуїтивно зрозумілий і простій для використання продукт. Середнім компаніям варто звернути увагу, що він надається безкоштовно для обслуговування 100 пристроїв. Мені, наприклад, потрібний безкоштовний мережний монітор для домашньої мережі, і, схоже, я знайшов його.

WhatsUp Gold

WhatsUp Gold компанії Ipswitch – зрілий продукт, що існує вже більше 10 років, і це наочно проявляється в його глибокій і потужній функціональності. Фахівці Ipswitch прислухалися до користувачів і внесли поліпшення на підставі їхніх коментарів і пропозицій.

Випускається три редакції WhatsUp Gold: Standard, Premium і Distributed. До складу Premium Edition входять усе компоненти Standard Edition, а також функції моніторингу через WMI, моніторингу UNIX і Linux, моніторингу бездротових вузлів доступу й т.д. Редакція Distributed доповнена функціональністю для мереж, розподілених по більших територіях. Докладний список компонентів кожної редакції опублікований на сайті WhatsUp Gold. Якщо виникає потреба переходу на більше високий рівень, повторна установка не обов'язкова. Досить увести новий ліцензійний ключ, і продукт автоматично обновляється до нової версії.

Я познайомився з WhatsUp Gold v15 Premium Edition. Процес установки простий, тому що в ньому передбачені всі умови, зокрема установка. NET Framework 4.0, IIS і SQL Server Express 2015. Власники великої мережі, можливо,

зволіють використовувати екземпляр SQL Server 2018 або SQL Server 2015 замість SQL Server Express. У пакеті установки втримуються дві надбудови: WhatsConnected і WhatsConfigured. Для даного огляду я встановила тільки WhatsUp Gold.

Після установки майстер проводить користувача по етапах налаштування оповіщень SMTP і облікових даних для різних пристроїв, підключених до мережі. У процесі сканування мережі можна спостерігати, як WhatsUp Gold виявляє кожний пристрій, перетворить його ім'я вузла й намагається визначити його роль (наприклад, вузол VMware, сервер Windows).

WhatsUp Gold успішно виявив і ідентифікував пристрою, а помилки у визначенні було легко виправити. Достоїнство продукту – можливість налаштування будь-яких параметрів з консолі Windows або веб-консолі.

У великих мережах можуть бути присутнім сотні й навіть тисячі пристроїв. В WhatsUp Gold вони групуються, щоб адміністраторові було простіше знайти потрібні пристрої. Існує два види груп: нединамічні й динамічні. Перші управляються вручну. Наприклад, якщо є два місця розташування, можна створити групу «Київ» і групу «Кіровоград», а потім перенести пристрій за допомогою методу drag-and-drop у відповідні групи.

За замовчуванням в WhatsUp Gold сформовано 20 динамічних груп, а в базі дані довідки є ще 14. Вони заповнюються з використанням інструкцій SQL. Наприклад, щоб помістити всі пристрої Cisco в одну динамічну групу, інструкція SQL може мати такий вигляд:

```
SELECT DISTINCT Device.nDeviceID
FROM Device
WHERE Device.bRemoved = 0 AND
Device.sSnmpOID LIKE N'1.3.6.1.4.1.9%'
```

Цих прикладів досить, щоб навіть адміністратори з поверхневими знаннями про SQL могли створити власні динамічні групи. Але такий метод здасться незручним для людей, які рідко звертаються до SQL. Хотілося б, щоб розроблювачі WhatsUp Gold зберегли можливість працювати вручну з функціями

SQL (для користувачів, знайомих з SQL), але додали майстер, що допомагає будувати інструкції SQL.

Компанія Ipswitch публікує схему бази даних на своєму сайті, що спрощує підготовку запитів і створення спеціалізованих звітів. Природно, запис ззовні додатка WhatsUp Gold неможлива, але дозволяється доступ для читання, якщо це допоможе зібрати потрібні дані.

Крім звичайної перевірки за допомогою команди ping, WhatsUp Gold контролює характеристики продуктивності (наприклад, використання процесора й пам'яті) і служби (наприклад, DNS, HTTP). Є також «пасивні монітори» для контролю пасток SNMP, системних журналів і журналів подій Windows.

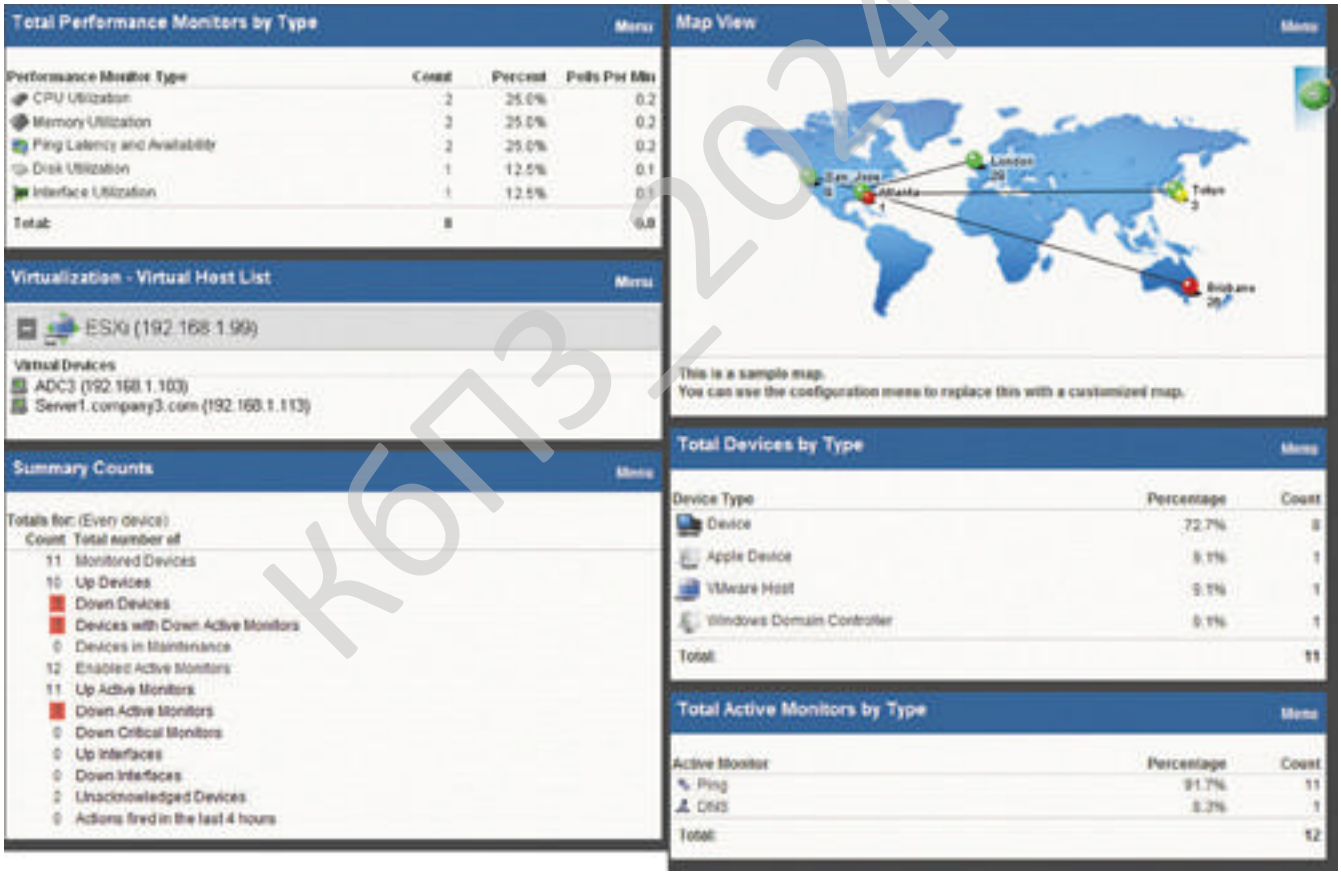

Рисунок 2.3 – Інтерфейс користувача WhatsUp Gold

При установці й налаштуванні інструмента моніторингу зручніше користуватися консоллю Windows, а у веб-консолі краще показаний стан кожного

пристрою. У випадку відмови пристрою відповідна піктограма стає жовтої, і у веб-консолі відображається вказівка, який монітор (наприклад, ping або DNS) видає помилку й протягом якого часу. Якщо пристрій як і раніше не відповідає, колір піктограми стає червоним. Клацнувши на пристрої, можна одержати додаткові відомості, що полегшують усунення неполадок.

Налаштування приладової панелі WhatsUp Gold дуже гнучкі. Не становить праці додавати, видаляти елементи й змінювати приладову панель відповідно до власних переваг, як показано на рисунку 2.3.

Компанія Ipswitch працює давно й заслужено має авторитет. Новітня версія WhatsUp Gold мене не розчарувала.

Orion Network Performance Monitor (NPM)

Orion Network Performance Monitor (NPM) компанії SolarWinds – ще один зрілий продукт. Його ціна висока в порівнянні з безкоштовними продуктами, але витрати окупаються завдяки широким можливостям.

Я встановила Orion NPM на виділеному сервері Windows Server 2018, члені домену тестової мережі. Установка на контролері домена (DC) не підтримується. У ході установки забезпечуються всі необхідні умови, у тому числі встановлюється. NET Framework 3.5 з пакетом відновлення SP1 і IIS. Зібрані дані зберігаються в обов'язковій базі даних на сервері. У невеликих і тестових мережах можна використовувати SQL Server Express 2015, але рекомендується SQL Server 2015 або більше нова версія.

При першому звертанні до адміністративного веб-консолі майстер допоможе задати облікові дані для SNMP, VMware і Windows, і вказати IP-Адреси для перевірки. Потім результати імпортуються в базу даних.

Всі пристрої в моїй мережі були виявлені без проблем, у тому числі сервер ESXi. Коли я розгорнула значок ESXi, на рисунку 2.з'явився повний список розміщених на сервері віртуальних машин. Провівши курсором миші над будь-якою віртуальною машиною, можна одержати зведення її стану. Імена неактивних віртуальних машин показані сірим кольором.

Замість установки агентів для контролю пристроїв використовується протокол Internet Control Message Protocol (ICMP – або команди ping), SNMP, WMI, системний журнал або процедура реєстрації в пристрої. Відповідно до керівництва адміністратора Orion NPM, методи без агентів використовуються по наступних причинах:

NPM не задіє служби, що відбирають ресурси у важливих прикладних програм;

NPM не встановлює ніяких програм на контрольованих мережних пристроях; некеровані або застарілі програми можуть стати уразливим місцем у мережі.

На мій погляд, це ясна відповідь на питання про переваги моніторингу з використанням агентів або без них.

Рисунок 2.4 – Інтерфейс користувача Orion NPM

Структура приладової панелі Orion NPM добре продумана й зручна для переміщень. Наприклад, на рисунку 2.4 показані зручні вкладки у верхній частині панелі, такі як Top10, Alerts (попередження), Syslog (системний журнал) і Events (події). Клацаючи кожний контрольований пристрій, подібно серверу ESXi і його віртуальним машинам, можна перейти на екран з більше докладною інформацією. На таких екранах розміщені численні «спідометри», на яких динамічно відображаються статистичні дані, у тому числі середній час відгуку, втрати пакетів, середня швидкість процесора й використана пам'ять. Клацнувши на спідометрі, можна одержати графік зі значеннями раніше зроблених вимірів. Є графік, що налаштовується також, на якому показаний стан пристрою за кілька тижнів або місяців.

В Orion NPM можна імпортувати інструмент Network Atlas. З його допомогою ви можете створити карту мережі в чотири етапи: виберіть тло, помістите об'єкти на карту, з'єднаєте об'єкти з об'єктами внутрішньої бази даних і зробіть необхідні налаштування. Компанія SolarWinds надає більше 30 карт, у тому числі карти миру й окремих континентів. Менш чим за мінуту я підготував карту США, додав піктограму й зв'язав піктограму на карті з комутатором своєї тестової мережі.

SolarWinds застосовує цікаву модель ліцензування. Замість єдиної ціни за один пристрій загальна вартість розраховується по найбільшому числу інтерфейсів, вузлів або томів. Наприклад, якщо є два 48-портових комутатори, 100 серверів/вузлів і 20 томів на жорстких дисках, споживач платить тільки за найбільше число, у даному прикладі – 100 серверів/вузлів. Я ввів цю інформацію в калькулятор ліцензій, доступний через Інтернет, і одержав ціну 3595 дол. Інші контрольовані пристрої виявляються «безкоштовними». Але перш ніж оформити покупку в Інтернеті, подзвоните в компанію й переконаєтеся, що не придбаєте зайві ліцензії.

Трапляється, що, призначивши сигнал тривоги для певного пристрою, адміністратор згодом не одержує попередження про аварію, тому що сигнал був

настроєний невірно. До складу Orion NPM входить інструмент Test Fire Alerts, за допомогою якого можна перевірити правильність спрацьовування тривоги у випадку відмови пристрою.

Одна з особливостей Orion NPM – спосіб реалізації окремих інструментів. Замість того щоб помістити всі інструменти в один виконується файл, що, багато хто з них представлені окремими програмами, які потрібно встановлювати. Я нарахував 15 таких програм.

Orion Network – перевірена програма керування мережею, потужна й надійна. Але розібратися в незвичайній моделі ліцензування буває нелегко, а ціна може виявитися занадто високої для деяких компаній.

Кожний із представлених продуктів має у своєму розпорядженні відмінні функції керування. Кожний оптимальний для певних застосувань. Spiceworks відмінно підходить для малих підприємств із обмеженим бюджетом. Він також гарний, якщо компанія обслуговує Ит-Інфраструктуру багатьох дрібних організацій.

Foglight NMS – ідеальне рішення для середніх компаній, що бідують у недорогому інструменті керування мережею. Можна безкоштовно контролювати до 100 пристроїв. Не становить праці додавати пристрою й розширювати Foglight NMS у міру росту потреб компанії.

Orion NPM заслуговує на увагу великих компаній, що прагнуть до досконалості в керуванні мережею.

WhatsUp Gold – бездоганна програма для керування мережею. При істотно більше високій ціні, чим у двох безкоштовних продуктів, її можливості також набагато вище. Після багатьох годин роботи з WhatsUp Gold я усе ще виявляю цікаві функції, які необхідно випробувати. WhatsUp Gold – ідеальний вибір для власників більших мереж з багатьма пристроями, якщо їм потрібний бездоганний інструмент для контролю й керування мережею.

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 звільняються 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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи керування параметрами структур

корпоративних мереж з застосуванням хмарних технологій.

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

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

б) вибрати та обґрунтувати методику побудови системи 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

Під час реалізації бакалаврського проекту були проведені дослідження й аналіз стану, перспектив і тенденцій розвитку корпоративних мереж. Дано визначення корпоративної мережі з погляду структурного, системно-технічного й функціонального аспектів. Виконано порівняльний аналіз структур корпоративних мереж: розподіленої й багатошарової. Відзначено, що перший тип структури дозволяє проектувати корпоративні з погляду ефективності розподілу функцій мережі між рівнями. Це дає можливість оцінити функціональність кожного рівня, але не дозволяє оцінити надаваний сервіс і механізми забезпечення якості послуг. Це можна зробити за допомогою багатошарових структур корпоративної мережі. Відзначено, що ієрархічні розподілені структури дозволяють ефективно проектувати великомасштабні мережі, забезпечують наскрізну якість обслуговування на всіх рівнях мережі, інформаційну безпеку, дозволяють впроваджувати інтелектуальні сервіси, системи IP-телефонії й відеоконференцзв’язку.

Проведений аналіз особливостей проектування корпоративних мереж, що показав, що проект самої корпоративної мережі створюється під функціональну модель підприємства. Тому якість керування корпоративною мережею буде впливати на якість функціонування підприємства.

Розглянуто переваги й недоліки типових рішень по об'єднанню локальних комп'ютерних мереж у корпоративну мережу організації. До типових рішень відносяться:

– корпоративні мережі на основі технології VPN;

– корпоративні мережі на основі виділених каналів зв'язку;

– корпоративні мережі на основі змішаних технологій, орендованих ресурсів мереж зв'язку сервіс провайдерів.

Проведено аналіз організаційної інфраструктури українських корпоративних мереж: відомчих, науково-освітніх та бібліотечних.

Аналіз розглянутих прикладів корпоративних мереж дозволив визначити загальні тенденції розвитку корпоративних мереж.

У розділі розглянуті спеціалізовані інфраструктури підприємства:

– телефонні мережі загального призначення для забезпечення прямого спілкування віддалених корпоративних абонентів, для передачі факсимільних повідомлень, для організації багатобічних телефонних конференцій і різних форм обробки й переадресації вхідних телефонних викликів;

– мережі ефірного й кабельного телебачення для забезпечення віщання й прийому державних і комерційних телевізійних інформаційних каналів і відеопрограм, для організації відеоконференцій;

– мережі передачі даних і доступу до інформаційних корпоративних ресурсів і ресурсів загального користування для прийому й передачі файлів, оперативний обмін текстовими повідомленнями й повідомленнями електронної пошти.

Процес конвергенції цих телекомунікаційних інфраструктур, що представляє собою взаємне проникнення мереж різного призначення шляхом використання єдиних компонентів і сполучення виконуваних функцій, привів до появи нового класу корпоративних мереж – до мультисервісних корпоративних мереж – мереж нового покоління NGN (Next Generation Network). Відзначено, що базовими принципами мультисервісних корпоративних мереж є поділ функцій переносу й комутації, функцій керування викликом і функцій керування послугами. Ключовим поняттям у мережах нового покоління стала "послуга", тобто та функціональність, що потрібно корпоративному користувачеві й потенційно може бути йому надана. Саме перелік послуг, рівень гарантованої якості їхнього надання, забезпечення безпеки доступу до послуги в майбутньому

будуть визначати перспективність мультисервісних корпоративних мереж нового покоління. Концепція NGN передбачає підтримку необмеженого набору послуг із гнучкими можливостями по їхньому керуванню, реалізацію універсальної транспортної мультипротокольної мережі з розподіленою комутацією, інтеграцію із традиційними мережами зв'язку.

Аналіз сучасних корпоративних мереж і тенденцій їхнього розвитку дозволив виділити вимоги до корпоративних мереж нового покоління:

– «мультисервісність» – незалежність технологій надання послуг від транспортних технологій;

– «широкополосність» – можливість гнучкої й динамічної зміни швидкості передачі інформації в широкому діапазоні залежно від поточних потреб користувача;

– «мультимедійність» – здатність мережі передавати багатокомпонентну інформацію (мова, дані відео, аудіо) з необхідною синхронізацією цих компонентів у реальному часі й використанням складних конфігурацій з'єднань;

– «інтелектуальність» – можливість керування послугою, викликом і з'єднанням з боку користувача або постачальника послуг;

– «інваріантність доступу» – можливість організації доступу до послуг незалежно від використовуваної технології;

– «розширюваність» – простота інтеграції окремих компонентів мережі (користувачів, комп'ютерів, додатків, служб) і можливість легко нарощувати довжини сегментів кабелів і заміняти існуючу апаратуру могутнішою;

– «масштабованість» – збільшення кількості вузлів і довжини зв'язків у дуже широких межах, при цьому продуктивність мережі не погіршується.

Таким чином, проведені огляд і аналіз стану, перспектив і тенденцій розвитку корпоративних мереж дозволили зробити вивід про необхідність узагальнення відомих результатів і проведення досліджень по створенню методів підвищення ефективності корпоративних мереж і методів і засобів інтелектуального керування корпоративними мережами.

Розглянемо різні схеми класифікації трафіку корпоративних мереж. Найбільш детальною є класифікація ATM, що лежить в основі типових вимог до параметрів і механізмів забезпечення якості обслуговування в сучасних корпоративних мережах. Але найбільш повна схема класифікації запропонована фірмою Cisco, що включає 11 класів трафіку. У цей час не багато підприємств використовують велику кількість класів трафіку. Як правило, використовується не більше чотирьох основних класів трафіку:

– Критичний – трафік критично важливих додатків.

– Транзакційний/інтерактивний – трафік клієнт-серверних додатків.

– Об'ємний – трафік додатків, що передають більші обсяги даних: передача більших файлів, синхронізація й реплікація баз даних.

– Не критичний – клас за замовчуванням для всього не призначеного трафіку (як мінімум приділяється 25 % смуги пропущення каналу).

У розділі розглянута загальна модель служби забезпечення якості обслуговування й структурних моделей сервісів QoS:

– модель сервісу DiffServ;

– модель сервісу IntServ.

Показано, що загальними важливими елементами є механізми обслуговування черг. Тому правильний вибір і налаштування алгоритмів обслуговування черг, оцінка можливої довжини черг в узах комутації й маршрутизації дозволить також оцінити параметри якості обслуговування при відомих характеристиках трафіку. Поводження черг являє собою імовірнісний процес, на який впливає багато факторів, особливо при складних алгоритмах обробки черг, що використовують пріоритети або зважене обслуговування різних потоків. Відомі моделі масового обслуговування дозволяють дати кількісну оцінку тільки для простих ситуацій, що не відповідають реальним умовам роботи корпоративної мережі. Тому ефективне рішення цього завдання можливо методом імітаційного моделювання алгоритмів обслуговування черг із метою оцінки складності реалізованого алгоритму, визначення його ефективності.

Для мультисервісних корпоративних мереж, у яких по єдиному середовищу передачі даних передається трафік з різними характеристиками, важливим є забезпечення якості обслуговування критичного трафіку. З урахуванням міжнародного стандарту ISO/IEC 9126 виділені наступні параметри якості обслуговування:

– пропускна здатність мережі;

– реакція на характеристики класу трафіку;

– кількість загублених і перекручених пакетів;

– час доставки пакета;

– нерівномірність часу доставки пакетів;

– затримка передачі пакетів.

Далі докладно розглянуті показники функціонування й завдання керування корпоративними мережами, серед яких виділені шість функціональних груп:

– 1 група: керування конфігурацією мережі й іменуванням;

– 2 група: обробка помилок;

– 3 група: аналіз продуктивності й надійності;

– 4 група: керування безпекою;

– 5 група: облік роботи мережі;

– 6 група: ідентифікація й прогнозування.

Важливим завданням для корпоративних мереж є прогнозування стану, якісне рішення якого дозволить адміністраторові вчасно підготуватися до критичної ситуації або уникнути її. Проведений аналіз базових методів математичного прогнозування дозволив обґрунтувати вибір методу нейромережного прогнозування.

Виділено п'ять базових архітектур систем керування, що вирішують перераховані вище завдання:

– однорівнева архітектура;

– ієрархічна архітектура;

– коміркова архітектура;

– платформна архітектура;

– розподілена архітектура на основі WEB-технології.

Відзначено, що найбільш популярними є платформна, котра використовується, наприклад, в HP OpenView і SunNet Manager, і розподілена архітектура на основі WEB-технології.

Виконаний аналіз архітектур перспективних систем керування корпоративними мережами дозволив сформулювати основні вимоги, запропоновані до сучасних засобів керування:

– автоматичне виявлення мережних пристроїв і визначення відносин між ними;

– збір і зберігання ключових параметрів вузлів мережі, що дозволяє прогнозувати проблеми й швидко знаходити першопричину несправності у випадку її виникнення;

– автоматична реакція на події;

– створення аналітичних звітів для аналізу й планування розвитку мережі;

– підтримка баз даних пристроїв для керування IT-активами;

– керування за допомогою стратегій;

– керування великими мережами за допомогою розподіленої архітектури з будь-якої точки мережі.

Системи керування оцінювалися за наступними критеріями:

– визначення імені вузла по його адресі через сервер DNS;

– можливість модифікації привласненого імені вузла;

– розпізнавання мережних топологій, підтримувані бази даних;

– формат звітів;

– підтримувані Web-сервери;

– можливість рішення завдань на підставі неповних даних;

– здатність до самонавчання;

– здатність прогнозувати роботу корпоративної мережі;

– можливість описати хід одержання рішення.

Таким чином, проведений аналіз показав, що:

– для збору й обробки інформації, необхідної для ідентифікації станів корпоративної мережі й прогнозування поводження корпоративної мережі потрібне створення багатофункціональних засобів керування корпоративними мережами;

– для реалізації засобів керування корпоративними мережами доцільно використовувати архітектури, що забезпечують багатоплатформеність менеджерів і агентів, універсальність доступу до менеджера, єдиний прикладний програмний інтерфейс;

– модель адміністрування зі стеком протоколів TCP/IP дозволяє реалізовувати надійні й ефективні засоби керування корпоративними мережами.

Результати проведених досліджень дозволили зробити вивід, що при всьому різноманітті підходів до побудови й керування роботою корпоративних мереж, основними завданнями є формування структури й визначення її параметрів.

Дослідимо завдання керування корпоративною мережею. Приведемо загальну постановку завдання керування, визначимо параметри й простір станів корпоративної мережі, цілі й параметри керування. Наведемо постановки загальних і часткових завдань керування.

Множина параметрів мережі – $\mathbf{SN} = \{\mathbf{SI}, \mathbf{PSI}(\mathbf{SI}), \mathbf{ST}, \mathbf{PST}(\mathbf{ST})\}$. розділяється на дві непересічні підмножини, що задають первинні й вторинні параметри. $\mathbf{SN} = \mathbf{SN}_0 \cup \mathbf{SN}_1$, де $\mathbf{SN}_0$ – множина первинних параметрів, $\mathbf{SN}_1$ – множина вторинних параметрів. Вторинні параметри залежать від первинних параметрів, які міняються адміністратором мережі при керуванні мережею, звідси:

$$\mathbf{SN}_1 = \{\mathbf{PSI}^*(\mathbf{SI}), \mathbf{PST}^*(\mathbf{ST})\},$$

де:

$$\mathbf{PSI}^*(\mathbf{SI}) = \{\mathbf{Z}_k, \mathbf{A}_k (k = 1,2,..., L), \mathbf{A}, \mathbf{B}^*, \mathbf{\Phi}\} \subseteq \mathbf{PSI}(\mathbf{SI}),$$

$$\mathbf{PST}^*(\mathbf{ST}) = \{\mathbf{A}_1^*, \mathbf{A}_2^*, \mathbf{A}_3^*, \boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \boldsymbol{\lambda}_3^*, \boldsymbol{\gamma}_1^*\} \subseteq \mathbf{PST}(\mathbf{ST}).$$

Також у множини параметрів мережі виділені базові параметри, які задаються при створенні мережі й визначають розвиток мережі. Ці параметри характерні тим, що вони є постійними на досить тривалому інтервалі часу. Базові параметри мережі це завжди первинні параметри.

Стан мережі визначається множиною параметрів $\mathbf{STN}$, так, що $\mathbf{STN} \subset \mathbf{SN}$.

Стан мережі, що відповідає певному набору базових параметрів, називається базовим станом. Цей стан відповідає початку функціонування мережі. Подальше функціонування мережі відбувається при незмінних базових параметрах. Множина базових параметрів – $\mathbf{BSN} \subseteq \mathbf{SN}_0$. Наприклад, $\mathbf{BSN} = \{\mathbf{SI}, \mathbf{ST}\}$. Реконфігурація (розвиток) мережі, зміна прикладних завдань, зміна состава користувачів і устаткування приводить до зміни множини базових параметрів мережі.

Простір станів мережі можна розбити на непересічні підпростори, кожен з яких характеризується своїм базовим (початковим) станом, тобто своїм набором базових параметрів. Показано, що кожний базовий підпростір є зв'язковим, а самі базові підпростори зв'язуються тільки через початкові базові стани. Це значить, що перехід з одного підпростору в інший можливий тільки при попередній зміні базових параметрів. Таким чином, переходи зі стану в стан при незмінних значеннях базових параметрів можливі тільки в межах одного базового підпростору.

Для адміністратора мережі такий вивід означає, що можна управляти мережею тільки при постійних базових параметрах, якими є, як правило, вихідні дані для побудови мережі.

Визначено множина можливих керувань мережею. Елементами цієї множини є варіюємі первинні параметри мережі, змінюючи які можна змінювати стан мережі – параметри керування. Такими параметрами можуть бути:

– параметри мережного устаткування й каналів зв'язку;

– кількісні параметри, що визначають состав мережі;

– параметри, що визначають розміщення прикладних програмних засобів у мережі;

– параметри, що задають структуру мережі.

Особливістю цих параметрів є можливість їхньої зміни адміністратором мережі і їхній вплив на стан мережі і якість її роботи. Базові параметри мережі не є параметрами керування. Якщо – **UN** множина параметрів керування мережею, то $\mathbf{UN} \subseteq \mathbf{SN}_0$ й $\mathbf{UN} \cap \mathbf{BSN} = \varnothing$.

Кроком керування є однократна й одночасна (разова) зміна заданого набору параметрів керування мережею.

Відзначено властивість мережі, що полягає в тому, що усередині одного базового підпростору станів будь-які переходи можливі за один крок керування. Це обумовлено тим, що на практиці допускається будь-яка однократна зміна параметрів керування мережею, коли адміністратор мережі змінює відразу всі необхідні параметри керування.

Оскільки перехід з одного базового підпростору станів в інше можливий тільки через початковий стан цього базового підпростору, то переходи між станами мережі усередині всього простору станів можливий мінімум за два кроки керування. На практиці, дійсно, спочатку змінюються базові параметри, а потім мережа переводиться в необхідний стан заданого підпростору.

З наведених даних треба, що концептуально природно представити процес керування корпоративною мережею як дворівневий процес керування, де на першому рівні формується множина базових параметрів, а на другому, при фіксованих базових параметрах, відбувається керування варіюємими параметрами.

При керуванні роботою мережі необхідно враховувати, що мережа призначена для рішення прикладних завдань (виконання додатків), кожне з яких повинне відповідати певним вимогам до якості одержуваних результатів. Забезпечення виконання комплексних вимог до якості рішення прикладних завдань є однієї з основних цілей керування. Основна мета керування містить ряд часткових цілей, кожна з яких може бути пов'язана з певним прикладним завданням. Уведено загальний (уніфікований) набір показників якості рішення прикладних завдань на мережі, що враховує специфіку кожного завдання:

$\mathbf{QT} = \{QT_1, QT_2, ..., QT_Q\}$, де Q – загальна кількість показників якості рішення прикладних завдань. QT_i – і-й показник. Завдання k має набір показників якості рішення, обумовлена вектором $\mathbf{q}_k = (q_{1k}, q_{2k}, .. q_{Qk})$, де $(k = 1,2,..., L)$. Тоді набір показників якості для цього завдання: $\mathbf{QT}_k = \{q_{1k}QT_1, q_{2k}QT_2, ..., q_{Qk}QT_Q\}$. Використання єдиної системи показників для оцінки якості рішення прикладних завдань дозволяє визначити множина часткових цілей керування мережею, як сукупність наступних функцій: $\mathbf{GT}^* = \{\underset{\mathbf{UN}}{opt}(F[QT_{ik}(\{\mathbf{p}_k, \mathbf{d}_k, \mathbf{u}_k, \mathbf{W}_k\})])\}$.

Оптимізація часткових показників якості рішення кожного окремого завдання не завжди може забезпечити оптимальну роботу мережі за рішенням усього комплексу завдань у цілому. Крім того, процеси, що програмно реалізують додатки, як правило, конкурують за ресурси мережі й одночасне досягнення оптимальних результатів по кожному завданню не завжди можливо. Тому пропонується використовувати інтегральні показники якості рішення завдань, для чого уведена множина ваг показників $\mathbf{A}_k = \{a_{ik} > 0\}$ для кожного завдання, і множина ваг завдань $\mathbf{B} = \{b_k\}$, $(i = 1,2,..., Q; k = 1,2,..., L)$.

Отримано узагальнену формулу для обчислення інтегрального (оптимального) показника якості роботи мережі за рішенням заданого набору завдань, що має вигляд:

$$GT^*(\mathbf{UN}) = \underset{\mathbf{UN}}{opt}\left(\sum_{k=1}^{L} GT_k(\mathbf{UN})\right) = \underset{\mathbf{UN}}{opt}\left(\sum_{k=1}^{L} b_k \sum_{i=1}^{Q} a_{ik} q_{ik}(QT_{ik})\right),$$

де $GT_k(\mathbf{UN}), (k = 1,2,.., L)$ – значення інтегральних показників якості роботи мережі при рішенні кожного завдання окремо.

Відзначено, що керування роботою корпоративної мережі, як правило, має на увазі керування потоками даних у мережі, що відбивається й у наборі параметрів керування **UN**. Проведені дослідження дозволили виділити основні фактори, що впливають на потоки даних у мережі, завантаження каналів зв'язку й мережного устаткування, які враховувалися при розробці моделей і постановці завдань.

У реальній корпоративній мережі інтенсивності потоків запитів, состав користувачів і состав розв'язуваних завдань можуть мінятися згодом, крім того, з розвитком мережі міняється состав устаткування і його параметри – тобто міняються базові параметри мережі. Все це викликає необхідність зміни (корекції) базових параметрів мережі для досягнення необхідної ефективності її роботи. Така зміна параметрів мережі – налаштування мережі – є один з основних процесів керування мережею. При цьому, природно, необхідно забезпечувати необхідні значення показників якості роботи мережі, пов'язаних з рішенням прикладних завдань.

Концепція дворівневого керування мережею зводиться до керування на рівні базових параметрів і керуванню на рівні варіюємих параметрів. Керування мережею на всіх рівнях повинне забезпечувати оптимальні значення показників якості роботи мережі.

У зв'язку із цим розрізняють два види керування мережею, застосовуваних на практиці, і відповідним зазначеним вище рівням керування:

– налаштування мережі (рівень базових параметрів);

– оперативне керування (приватним і найбільш важливим випадком якого, є керування потоками даних) (рівень варіюємих параметрів).

Налаштування мережі потрібно або при первинному (початковому) запуску мережі після її створення при заданому наборі базових параметрів, або, при зміні якого-небудь із базових параметрів мережі.

Оперативне керування застосовується постійно при роботі мережі і є керуванням на рівні варіюємих параметрів. Необхідність оперативного керування пов'язана з виникненням у мережі ситуацій, коли, наприклад, з'являються тимчасові зміни інтенсивностей потоків даних, і налаштувань устаткування, викликані виробничою необхідністю.

Множина показників якості роботи мережі **QT** розділимо на дві підмножини так, що $\mathbf{QT} = \mathbf{QT}_0 \cup \mathbf{QT}_1$ й $\mathbf{QT}_0 \cap \mathbf{QT}_1 \neq \varnothing$. Підмножину $\mathbf{QT}_0$ становлять показники якості, що обчислюються на етапі налаштування мережі, а

підмножину $\mathbf{QT}_1$ становлять показники якості, що обчислюються при оперативному керуванні мережею.

Сформульовано загальне завдання налаштування мережі, як завдання оптимізації на множини $\mathbf{QT}_0$. Рішенням завдання налаштування мережі буде набір базових параметрів $\mathbf{UN}_0^*$, що забезпечують оптимальне значення показника якості роботи мережі.

Сформульовано загальне завдання оперативного керування мережею, як завдання оптимізації на множини $\mathbf{QT}_1$, що вирішується після рішення завдання налаштування**.** Рішенням завдання оперативного керування буде оптимальний набір параметрів оперативного керування $\mathbf{UN}_1^*$.

Оскільки корпоративна мережа, як правило, має ієрархічну структуру, керування мережею повинне враховувати особливості керування ієрархічними системами. Основними принципами керування ієрархічною корпоративною мережею в нашім випадку є:

– принцип декомпозиції, що передбачає поділ корпоративної мережі на ряд підмереж;

– принцип координації керування роботою підмереж, коли керування для кожної підмережі виробляються з урахуванням стану інших підмереж;

– принцип узгодження цілей керування підмережами, при якому часткові (локальні) цілі керування підмережами повинні забезпечувати досягнення глобальної цілі керування всією корпоративною мережею.

Декомпозиція завдання керування передбачає декомпозицію корпоративної мережі на множина часткових мереж (підмереж) і керування рішенням групи однотипних часткових завдань. Це багато в чому відповідає принципам керування в VLAN і VPN. Пропоновані правила декомпозиції мережі забезпечують виконання наступних умов:

– розбивка на підмережі забезпечує можливість автономного оперативного керування підмережами, а якість роботи підмережі визначається тільки параметрами цієї підмережі;

– потоки даних між підмережами не залежать від оперативного керування.

Показано, що перераховані умови можуть виконуватися, якщо декомпозиція мережі проводиться на рівні базових параметрів, що треба із властивостей простору станів мережі.

Для даного випадку конкретизовані загальні завдання керування. Рішенням завдання налаштування мережі буде набір базових параметрів $\mathbf{UN}_0^*$, що визначає розбивку вузлів мережі на підмережі, що забезпечує оптимальне значення показника якості роботи мережі. Елементом множини $\mathbf{UN}_0^*$ є, наприклад, матриця $\mathbf{C}_1$. Ще одним результатом рішення завдання налаштування повинне бути визначення состава підмножин $\mathbf{UN}_{1i}$, де i – номер підмережі.

Загальне завдання оперативного керування корпоративною мережею може бути розбита на ряд завдань оперативного керування підмережами. Сформульовано загальне завдання оперативного керування підмережею. Рішенням завдання оперативного керування буде оптимальний набір параметрів оперативного керування $\mathbf{UN}_{1i}^*$ на кожному кроці керування.

Запропонований підхід має наступні переваги:

1. Скорочується розмірність завдання налаштування мережі, оскільки в порівнянні із загальним завданням керування скорочується число обмежень і спрощується цільова функція.

2. Проводиться декомпозиція завдання оперативного керування на завдання оперативного керування підмережами, що дає можливість скоротити розмірність кожного завдання.

3. З'являється можливість незалежного рішення завдань оперативного керування підмережами, змінюючи для кожного завдання свої набори показників якості й параметрів керування, а також алгоритми керування.

Декомпозиція завдання керування припускає скоординоване керування підмережами. Координація керування підмережами в цьому випадку повинна забезпечувати погоджене за часом керування підмережами. Необхідність

узгодження зв'язана як з різницею кроків керування підмережами по тривалості, так і з обмеженням автономності керування кожної підмережею, що не завжди дозволяє вибирати початок кроку керування незалежно від стану інших підмереж. Проведені дослідження дозволили сформулювати правила координації керування підмережами:

– при зміні базових станів мережі процеси оперативного керування в підмережах повинні бути зупинені доти, поки не буде вирішене завдання налаштування мережі при нових базових параметрах;

– оперативне керування підмережами не міняє параметри потоків даних.

Перераховані правила визначають дії адміністратора мережі.

Особливістю рішення завдань оперативного керування підмережами є необхідність обліку конкуренції процесів за поділювані ресурси усередині підмережі. Тому узгодження цілей при оперативному керуванні можна забезпечити шляхом розподілу ресурсів між завданнями. Для цього визначена матриця розподілу ресурсів підмережі $\mathbf{RS} = \|rs_{ij}\|$, $(i = 1,2,...,T;\ j = 1,2,...,R)$, ($0 \le rs_{ij} \le 1$ – частка ресурсу j, виділеного завданням типу i), яку можна використовувати як параметр оперативного керування. Показано, що загальне завдання оперативного керування тоді ставиться як завдання оптимізації:

$$GT_i^*(\mathbf{RS}) = \underset{\mathbf{RS}}{opt}\left(\sum_{k=1}^{T} b_{1k}\left(\sum_{j=1}^{Q} a_{1ijk} q_{jk} QT_{1ijk}(\mathbf{S}_{ik}) + \sum_{m=1}^{R} c_{km}(rt_m)(rs_{km})\right)\right),$$

при заданій системі обмежень. Результатом рішення даного завдання буде розподіл ресурсів підмережі номер і між завданнями, розв'язуваними на цієї підмережі.

Дослідимо можливі методи рішення завдань оптимізації як завдань математичного програмування.

Далі отримані рішення для найбільше що часто зустрічаються на практиці часткових завдань керування: формування структури мережі; керування розподілом смуги пропущення каналу зв'язку; розподіл ресурсів багатосерверного вузла обробки інформації. Показано можливі підходи до

рішення цих завдань. На основі розроблених методів, моделей створена методика застосування отриманих результатів при керуванні корпоративною мережею, що визначає порядок рішення завдань аналізу й керування. Проведено узагальнення результатів для випадку мережі з декількома типами потоків даних, що вимагає використання технології мудьтисервісних мереж і QoS.

3.2 Розробка структурної схеми

Розглянемо структурну схему архітектури інтелектуальної системи керування корпоративною мережею, алгоритм її функціонування, наведені експериментальні результати реалізації нейромережного методу прогнозування стану корпоративної мережі.

Пропонована структурна схема архітектури системи керування, представлена на рисунку 3.1, задовольняє всім вимогам, застосованим до сучасних систем керування. Модуль ідентифікації й прогнозування станів корпоративної мережі, що входить до складу системи керування, реалізований у нейромережному логічному базисі.

Архітектура системи керування корпоративною мережею побудована за модульним принципом й складається із семи основних підсистем:

– підсистема доступу до устаткування корпоративної мережі – надає доступ до інформації про стан об'єкта по корпоративній мережі;

– підсистема пошуку об'єктів корпоративної мережі – здійснює пошук об'єктів корпоративної мережі й відтворить її топологію;

– підсистема збору статистики – здійснює збір статистичних даних про кожний знайдений об'єкт;

– база даних статичної інформації корпоративної мережі – використовується для зберігання зібраних даних про об'єкти мережі;

– підсистема формування архітектури нейронної мережі – за допомогою параметрів налаштування (кількість внутрішніх шарів, початкові значення ваг) визначає конструктивні особливості нейромережного комплексу;

– підсистема керування й прогнозування завантаженості корпоративної мережі – здійснює контроль над всіма елементами системи й формує прогноз про стан корпоративної мережі на основі даних, одержуваних з вихідного шару нейронної мережі;

– середовище візуалізації – являє собою графічний інтерфейс користувача.

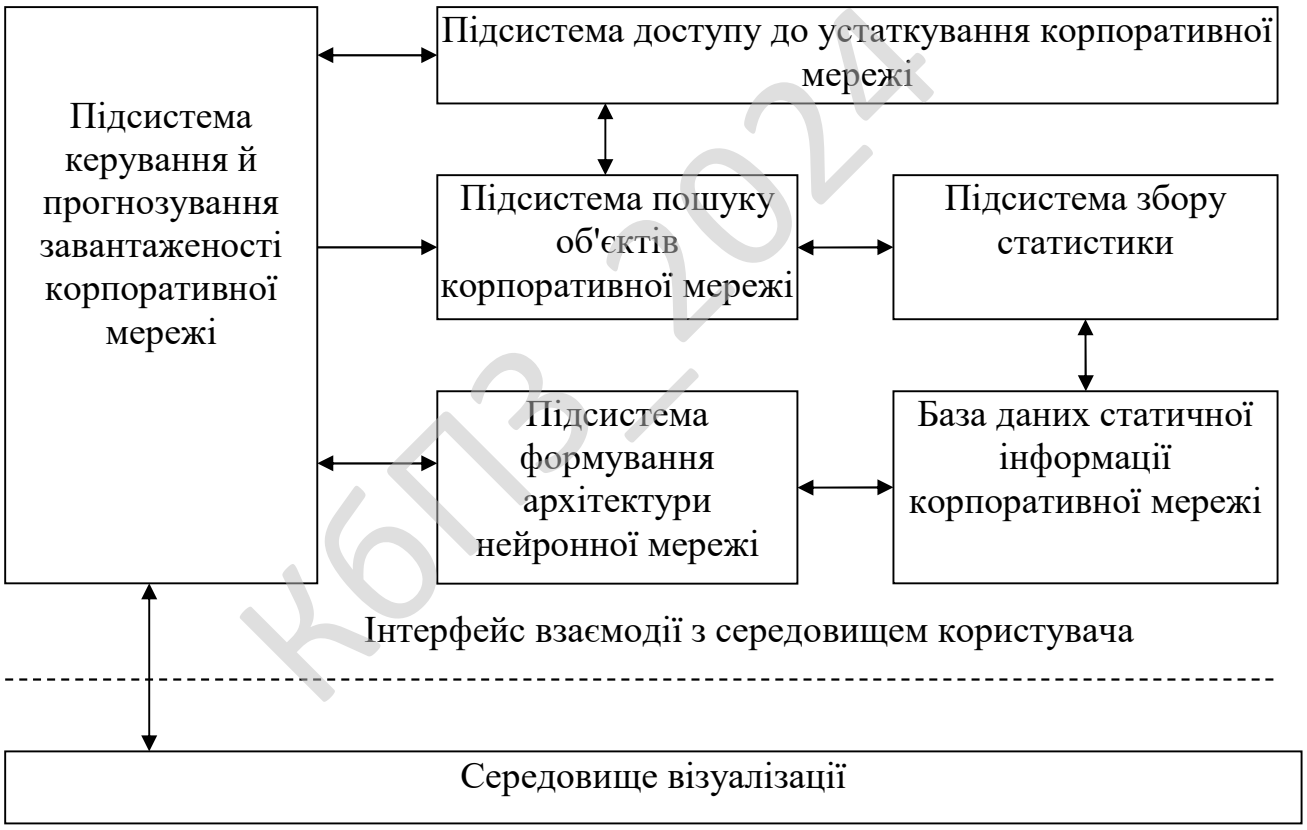

Рисунок 3.1 – Структурна схема системи

Розроблено алгоритм функціонування системи, якому можна розділити на чотири етапи:

– ініціалізація;

– збір статистики;

– формування нейронної мережі;

– контроль над корпоративною мережею.

Перший етап – ініціалізаційний, виконується за три основних кроки:

– початкова ініціалізація – на цьому кроці підсистема керування й прогнозування налаштовує всі елементи системи;

– пошук об'єктів корпоративної мережі – початковий пошук і формування списку об'єктів корпоративної мережі; на цьому етапі запускається підсистема пошуку об'єктів мережі;

– формування бази даних – формування бази даних статистичної інформації на основі списку об'єктів мережі.

Другий етап полягає в зборі статистичної інформації, необхідної для навчання нейронної мережі. Етап виконується за один крок, під час роботи якого запускається підсистема збору статистики, у результаті чого поповнюється база даних.

Третій етап – формування й навчання нейронної мережі на основі списку об'єктів мережі й накопиченої в базі даних статистичної інформації. Етап реалізується за один крок.

Перші три етапи є підготовчими етапами. На останньому, четвертому етапі відбувається використання навченої раніше нейронної мережі для формування прогнозу стану об'єктів корпоративної мережі й стани корпоративної мережі в цілому. На основі отриманого прогнозу, система дає висновок про проблемні місця корпоративної мережі.

Більшість адміністраторів для прогнозування використовують невелику кількість інтегральних параметрів – обсяг трафіку, затримки у вузлах комутації, кількість загублених пакетів, що не дозволяє одержати якісний прогноз і підходить в основному для великих магістралей мереж. Для підвищення якості прогнозу пропонується використовувати значення параметрів MIB пристроїв. У корпоративній мережі це можливо, тому що всі ресурси належать підприємству й доступ до пристроїв і мережних вузлів в адміністраторів не обмежений.

Далі опишемо розроблену підсистема прогнозування стану корпоративної мережі й наведені результати експериментального дослідження вузла RUNNet. За результатами технічного аналізу з параметрів MIB були обрані 32 змінних: 25 вхідні і 7 вихідні. Вибір визначався тим, що саме ці змінні варіюються динамічно протягом доби. Виміри проводилися щогодини протягом декількох місяців. Як метод прогнозування був обґрунтовано обраний нейромережний метод. Для досліджень можливостей розробленої підсистеми прогнозування NeuroNet були проведені різні експерименти, основна мета яких – підібрати значення параметрів налаштування нейромережного комплексу, при яких підсумкові результати її роботи містили найменшу кількість помилок у прогнозі. Експерименти проводилися на даних, отриманих від системних адміністраторів корпоративної мережі RUNNnet. При побудові прогнозу статистика була отримана для вузла мережі RUNNet, структура якого представлена на рисунку 3, протягом 2 місяців, включаючи критичні, з періодом в 1годину.

Для оцінки наскільки ефективності підсистеми прогнозування, як експеримент ставилися досвіди по прогнозуванню трафіку мережі по інтегральних параметрах, і проводилося порівняння підсумків з тими, які вийшли в результаті досвідів прогнозування по параметрах MIB пристроїв.

Проведено розрахунок характеристик структури корпоративної мережі NanoNet, що дозволив спрогнозувати й визначити навантаження на канали зв'язку й опорні вузли при заданих параметрах телекомунікаційного устаткування з урахуванням інтенсивності роботи учасників національної мережі.

Для налагодження програмного забезпечення систем керування й вибору й налаштування алгоритмів обслуговування черг у вузлах комутації й маршрутизації з урахуванням тимчасових параметрів трафіку розроблена архітектура апаратно-програмної системи комплексного динамічного налагодження й випробування. Дана система дозволяє реєструвати й обчислювати наступні параметри: час надходження пакета у вузол, час трансляції пакета вузлом у мережу, час затримки пакета у вузлі, зміна затримки, кількість загублених пакетів, зміна довжин черг і їхніх поточних значень.

До складу системи входять:

– комп'ютер розроблювача;

– імітатор мережного вузла;

– пастка подій;

– синхронізатор процесів імітації;

– програмне забезпечення засобів налагодження й випробування.

В основу реалізації архітектури системи налагодження й випробування покладене принцип, що використовує стартостопний режим і додаткові апаратно-програмні засоби моделювання мережного вузла, керовані комп'ютером розроблювача, у середовищі операційної системи якого функціонує відлагожувальне ПЗ (ВПЗ) системи керування корпоративною мережею. Основна особливість системи налагодження й випробування полягає в тому, що в реальному масштабі часу працює тільки ВПЗ, а при рішенні допоміжних завдань (підготовка даних для імітації, моделювання, обробка результатів і т.д.) відбувається останов "реального" часу. Останов "реального" часу здійснюється по подіях, до яких відносяться:

– "запит на обслуговування черги";

– "видача керуючого впливу";

– "читання стану черги".

Пастка подій ідентифікує кожну подію й інформує про нього програму синхронізації процесів імітації, що зупиняє "реальне" час тільки на періоди рішення допоміжних завдань. Синхронізація функціонування комп'ютера розроблювача з лічильником реального часу, що досягається при цьому, забезпечує повний збіг ходу ПЗ системи керування корпоративною мережею при повторенні експериментів з однаковими вихідними даними, що істотно розширює можливість пошуку помилок у процесі налагодження.

3.3 Розробка функціональної схеми

Перейдемо до розгляду функціональної схеми. Сформулюємо й вирішимо завдання аналізу функціональної структури корпоративної мережі. Для цього розроблена концепція проведення аналізу, вирішені завдання формального опису структури й обчислення її характеристик: навантаження на канали зв'язку, вузли й структуроутворююче устаткування мережі.

Основною метою аналізу структури є визначення параметрів потоків даних, що проходять по каналах зв'язку мережі й вступні на вузли мережі. Однак, завдання структури мережі тільки як сукупності вузлів і зв'язків між ними, не дозволяє досліджувати потоки даних, оскільки потоки даних формуються розв'язуваними на мережі завданнями, точніше додатками, які запускаються на вузлах мережі й обмінюються між собою даними. Отже, для аналізу мережі необхідно відомості про функціональну структуру доповнити відомостями про додатки і їхнє розміщення в мережі.

Результатами аналізу повинні стати математичні залежності, що дозволяють обчислювати чисельні значення характеристик мережі з урахуванням особливостей конкретної структури мережі.

Запропоновано концептуальний підхід до аналізу функціональної структури мережі, заснований на дослідженні взаємодії додатків (завдань) як незалежних джерел і приймачів даних. У цьому випадку спочатку визначаються параметри потоків даних між додатками при виконанні всього комплексу завдань (будується інформаційна модель), а потім, залежно від розміщення додатків по вузлах мережі й використовуваного устаткування, визначаються параметри потоків даних між вузлами мережі (будується технічна модель). При цьому не тільки повністю враховуються всі взаємодії між додатками, але й з'являється можливість проведення аналізу складних ієрархічних мережних структур, шляхом декомпозиції на підмережі, що часто застосовується в технологіях VLAN і VPN. Даний підхід розвивається й застосовується в бакалаврському проекті.

Відзначимо, що отримані результати аналізу потоків даних, можна надалі використовувати для проведення більше глибоких досліджень із застосуванням відомих методів і моделей, наприклад, СеМО.

Для аналізу функціональної структури й розрахунку характеристик мережі визначені правила її формального опису, що однозначно задають властивості додатків і структуру мережі, що дозволяють будувати моделі для проведення розрахунків.

Відповідно до концептуального підходу до проведення аналізу функціональної структури корпоративної мережі виділені дві складові її структури інформаційна й технічна:

– Інформаційна структура визначає інформаційні потоки між інформаційними вузлами, на яких установлене програмне забезпечення й представляє сукупність вузлів і інформаційних ресурсів, розміщених на цих вузлах. Маючи у своєму розпорядженні дані про інформаційну структуру мережі можна приймати рішення про організацію каналів зв'язку між вузлами мережі, визначати необхідні параметри телекомунікаційної системи, формувати структуру мережі.

– Технічна структура – сукупність структуроутворюючого устаткування, технічних вузлів мережі й каналів зв'язку. Технічному вузлу можуть відповідати трохи інформаційних.

Таким чином, для повноцінного аналізу функціональної структури реальної мережі необхідно провести аналіз складових її інформаційної й технічної структур і зв'язати результати аналізу.

Зв'язування результатів аналізу інформаційної й технічної структур має на увазі відображення характеристик інформаційної структури в характеристики технічної структури й визначення параметрів технічної структури на основі параметрів і характеристик інформаційної структури.

На рисунку 3.2 представлена функціональна схема системи, які відповідає запропонованому підходу.

Рисунок 3.2 – Функціональна схема системи

Інформаційна структура мережі задається набором параметрів:

$$\mathbf{SI}=\{N,M,D,L,R,\mathbf{S}_k(k=1,2,\ldots,L),\mathbf{A}_{km}(k=1,2,\ldots,L;m=1,2,\ldots,D),\mathbf{G},\mathbf{H},\mathbf{S}\},$$

де:

– N – число працюючих користувачів;

– M – число задіяних вузлів;

– D – число використовуваних додатків;

– L – число розв'язуваних завдань;

– R – число баз даних;

– $\mathbf{S}_k = \{\mathbf{p}_k, \mathbf{d}_k, \mathbf{u}_k, \mathbf{W}_k\}$, ($k = 1,2,...L$) – набір даних, що описують розв'язувані завдання, де:

– $\mathbf{p}_k = (p_{k1}, p_{k2}, ..., p_{kD})$, – вектор-рядок, що визначає завданням k додатка, що запускаються;

– $\mathbf{d}_k = (d_{k1}, d_{k2}, ..., d_{kR})$ – вектор-рядок, що визначає використовувані завданням k бази даних (сховища даних);

– $\mathbf{u}_k = (u_{k1}, u_{k2}, ..., u_{kN})$, – вектор-рядок, що визначає завдання k користувачів, що запускають $\mathbf{W}_k = \left\| w_{kij} \right\|$ ($i = 1,2,..., D$; $j = 1,2,..., D$) – матриця, що встановлює послідовність запуску додатків завданням k;

– $\mathbf{A}_{km} = \{\mathbf{v}_{km}, \mathbf{b}_{km}\}$, ($k = 1,2,...,L$; $m = 1,2,...,D$) – набір даних, що описують додатки, використовувані завданнями, де:

– $\mathbf{v}_{km} = (v_{km1}, v_{km2}, ..., v_{kmR})$, – вектор-рядок, що задає обсяги даних, якими обмінюється додаток m з базами даних, при рішенні завдання k;

– $\mathbf{b}_{km} = (b_{km1}, b_{km2}, ..., b_{kmD})$ – вектор-рядок, що задає обсяги даних, якими обмінюється додаток m з іншими додатками, при рішенні завдання k;

– **G** матриця розміщення додатків по вузлах;

– **H** – матриця підключення користувачів до вузлів;

– **S** – матриця розміщення баз даних по вузлах.

Набір однозначно визначає інформаційну структуру мережі.

Для заданого набору **SI** отримані результати, що дозволяють визначити параметри потоків даних між вузлами мережі.

При цьому використовувалася матриця інтенсивностей потоків запитів користувачів на запуск завдань $\mathbf{\Lambda} = \left\| \lambda_{ij} \right\|$, ($i = 1,2,..., N$; $j = 1,2,..., L$).

Показано, що матриця інтенсивностей потоків даних між вузлами мережі при рішенні завдання k обчислюється за формулою: $\mathbf{A}_k = \lambda_k \mathbf{Z}_k$, ($k = 1,2,...,L$). , де $\mathbf{Z}_k$ – матриця обсягів даних, переданих між вузлами, при рішенні завдання. Обчислені також матриця інтенсивностей потоку запитів на запуск додатка номер j, на вузлі номер i – $\mathbf{B}^*$ і матриця сумарної інтенсивності потоку запитів до бази даних номер j, на вузлі номер i – $\mathbf{\Phi}$.

Таким чином, визначена множина параметрів потоків даних інформаційної структури мережі: $\mathbf{PSI(SI)} = \{\mathbf{\Lambda}, \mathbf{Z}_k, \mathbf{A}_k(k = 1,2,...,L), \mathbf{A}, \mathbf{B}^*, \mathbf{\Phi}\}$.

Розроблено моделі для аналізу ієрархічної трьохрівневої інформаційної структури мережі, що дозволяють визначити завантаження каналів зв'язку й мережного устаткування в мережі, що складається із сукупності підмереж і корпоративних серверів. Отримано наступні результати:

– матриці інтенсивностей потоків даних між групами й усередині груп кожного рівня $\mathbf{A}_1(\mathbf{C}_1) = \mathbf{C}_1\mathbf{A}(\mathbf{C}_1)^T$, $\mathbf{A}_2(\mathbf{C}_2) = \mathbf{C}_2[\mathbf{A}_1(\mathbf{C}_1) - dg(\mathbf{A}_1(\mathbf{C}_1))](\mathbf{C}_2)^T$;

– матриці інтенсивностей потоків даних утворених кожним завданням $\mathbf{A}_{1k}(\mathbf{C}_1)$ і $\mathbf{A}_{2k}(\mathbf{C}_1)$, ($\mathbf{A} = \sum_{k=1}^{L} \mathbf{A}_k$).

Досліджено властивості матриць, що відбивають взаємозв'язки потоків даних окремих завдань, сформульовані у вигляді доведених тверджень.

Як міра ефективності ієрархічної структури запропоновані коефіцієнти поглинання інтенсивностей потоків даних на кожному рівні. Коефіцієнт поглинання на кожному рівні відбиває частку сумарної інтенсивності потоків, що локалізується усередині рівня.

Результати аналізу інформаційної структури дозволяють визначати параметри потоків даних між логічними об'єднаннями вузлів мережі. Інформаційна структура реалізується конкретними технічними засобами й втілюється у вигляді технічної структури. При цьому можливо невідповідність інформаційної й технічної структур, пов'язане з технічними характеристиками й

можливостями апаратури. Наприклад, на одному технічному вузлі можуть бути встановлені кілька інформаційних вузлів. У зв'язку із цим розроблені методи й засоби аналізу технічної структури корпоративної мережі, створюваної на базі інформаційної структури.

Для аналізу роботи реальної мережі розроблений метод формального опису її технічної структури. Структура реальної мережі формується із застосуванням структуроутворюючого устаткування (комутатори, маршрутизатори), до якого підключаються вузли мережі, при цьому створюється багаторівнева (часто трьохрівнева) мережа. Такий підхід застосовується, як правило, при створенні корпоративної мережі на базі технології VLAN. При цьому групи першого рівня інформаційної структури становлять віртуальні локальні мережі. Другий і третій рівні інформаційної структури призначені для з'єднання цих мереж між собою.

Число комутаторів, використовуваних для з'єднання вузлів технічної структури корпоративної мережі при створенні груп першого рівня (комутатори першого рівня), визначається особливостями реальної мережі, технічними можливостями комутаторів. Позначимо це число K_1^*, ($K_1^* \geq K_1 \geq 1$). Число комутаторів, для з'єднання комутаторів першого рівня й створення груп другого рівня (комутатори другого рівня) – K_2^*, ($K_2^* \geq K_2 \geq 1$). Число комутаторів третього рівня для з'єднання комутаторів другого рівня – $K_3^* \geq 0$.

Технічна структура мережі задається множиною **ST** , елементами якого є:

– $\mathbf{Y}_1^* = \left\| y_{1ij}^* \right\|$, ($i = 1,2,...,M;\ j = 1,2,...,K_1^*$) матриця з'єднань технічних вузлів мережі (робітники станції, сервери) з комутаторами першого рівня;

– $\mathbf{Y}_2^* = \left\| y_{2ij}^* \right\|$, ($i = 1,2,...,K_1^*;\ j = 1,2,...,K_2^*$) матриця з'єднань комутаторів першого рівня з комутаторами другого рівня;

– $\mathbf{Y}_3^* = \left\| y_{3ij}^* \right\|$, ($i = 1,2,...,K_2^*;\ j = 1,2,...,K_3^*$) матриця з'єднань комутаторів третього рівня з комутаторами другого рівня;

– $\mathbf{X}_1^* = \left\| x_{1ij}^* \right\|$, ($i, j = 1,2,..., K_1^*$) матриця з'єднань комутаторів першого рівня;

– $\mathbf{X}_2^* = \left\| x_{2ij}^* \right\|$, ($i, j = 1,2,..., K_2^*$) матриця з'єднань комутаторів другого рівня;

– $\mathbf{X}_3^* = \left\| x_{3ij}^* \right\|$, ($i, j = 1,2,..., K_3^*$) матриця з'єднань комутаторів третього рівня.

Для кожного рівня задані матриці, що задають пропускні здатності каналів зв'язку, використовуваних при побудові мереж:

$$\mathbf{C}_1^*(\mathbf{Y}_1^*), \mathbf{C}_2^*(\mathbf{Y}_2^*), \mathbf{C}_3^*(\mathbf{Y}_3^*), \mathbf{C}_1^*(\mathbf{X}_1^*), \mathbf{C}_2^*(\mathbf{X}_2^*), \mathbf{C}_3^*(\mathbf{X}_3^*).$$

Для заданої множини **ST** отримані наступні результати:

– матриця інтенсивностей інформаційних потоків, між комутаторами першого рівня й усередині груп технічних вузлів, підключених до комутаторів першого рівня: $\mathbf{A}_1^*(\mathbf{Y}_1^*) = (\mathbf{Y}_1^*)^T \mathbf{A}\mathbf{Y}_1^*$;

– матриця сумарних інтенсивностей інформаційних потоків для комутаторів другого рівня: $\mathbf{A}_2^*(\mathbf{Y}_2^*) = (\mathbf{Y}_2^*)^T [\mathbf{A}_1^*(\mathbf{Y}_1^*) - dg(\mathbf{A}_1^*(\mathbf{Y}_1^*))]\mathbf{Y}_2^*$;

– матриця інтенсивностей інформаційних потоків, між і усередині комутаторів третього рівня: $\mathbf{A}_3^*(\mathbf{Y}_3^*) = (\mathbf{Y}_3^*)^T [\mathbf{A}_2^*(\mathbf{Y}_2^*) - dg(\mathbf{A}_2^*(\mathbf{Y}_2^*))]\mathbf{Y}_3^*$;

– вектори інтенсивностей інформаційних потоків, що надходять на комутатори всіх рівнів: $\boldsymbol{\lambda}_1^* = (\lambda_{11}^*, \lambda_{12}^*, ..., \lambda_{1K_1^*}^*)$, $\boldsymbol{\lambda}_2^* = (\lambda_{21}^*, \lambda_{22}^*, ..., \lambda_{2K_2}^*)$, $\boldsymbol{\lambda}_3^* = (\lambda_{31}^*, \lambda_{32}^*, ..., \lambda_{3K_3}^*)$;

– вектор сумарних інтенсивностей потоків даних, переданих по каналах зв'язку $\boldsymbol{\gamma}_1^* = (\gamma_{11}^*, \gamma_{12}^*, ..., \gamma_{1M}^*)$, ($i = 1,2,..., M$).

Отримані також формули для обчислення параметрів потоків даних кожного завдання.

Оскільки в корпоративних мережах великої розмірності використовується маршрутизація, то для обчислення параметрів потоків необхідно застосовувати відповідні алгоритми маршрутизації на графах. Для цього побудована матриця графа зв'язків між комутаторами $\mathbf{\Omega}$. Показано, що матриця $\mathbf{\Omega}$ може бути представлена у вигляді:

$$\mathbf{\Omega} = \begin{pmatrix} \mathbf{X}_1^* & \mathbf{Y}_2^* & \mathbf{0} \\ (\mathbf{Y}_2^*)^T & \mathbf{X}_2^* & \mathbf{Y}_3^* \\ \mathbf{0} & (\mathbf{Y}_3^*)^T & \mathbf{X}_3^* \end{pmatrix}.$$

Таким чином, параметри потоків даних у мережі для заданої технічної структури визначаються множиною:

$$\mathbf{PST(ST)} = \{\mathbf{A}_1^*(\mathbf{Y}_1^*), \mathbf{A}_2^*(\mathbf{Y}_2^*), \mathbf{A}_3^*(\mathbf{Y}_3^*), \boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \boldsymbol{\lambda}_3^*, \boldsymbol{\gamma}_1^*, \mathbf{\Omega}\}.$$

Множина **PST(ST)** визначає параметри, як сумарних, так і часток потоків даних для окремих завдань (додатків), переданих по каналах зв'язку.

Якість рішення завдань залежить від того, яка частина пропускної здатності каналу (смуга) виділена для кожного завдання, що звичайно досягається шляхом застосування режиму гарантованої якості обслуговування (QoS). Тому при рішенні завдань розрахунку потоків із застосуванням систем з гарантованою якістю обслуговування проводиться розширення множини **PST(ST)** шляхом додавання множини коефіцієнтів поділу каналів – **PKST** . Це дозволяє визначити залежність характеристик мережі від смуги пропущення в каналі (комутаторі), що відводиться для завдання.

Показано, як застосовувати отримані результати при аналізі структури VLAN і VPN мереж.

Також проведене узагальнення отриманих результатів для випадку мультисервісної мережі й для випадку, коли для кожної групи завдань застосовуються свої правила передачі даних.

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

3.4 Розробка діаграми процесів

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

Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ (Середовище візуалізації) далі можна провести пошук об'єктів корпоративної мережі з подальшим переглядом статистика роботи корпоративної мережі та підсистеми доступу до устаткування корпоративної мережі. У подальшому через обробник помилок можна потрапити до база даних статичної інформації корпоративної мережі, звідки через формування архітектури нейронної мережі провести перегляд сформованої нейронної мережі та створення журналу роботи нейронної мережі.

Рисунок 3.3 – Діаграма взаємодії процесів

Далі можна проводити моніторинг корпоративної мережі з формуванням звіту та збереження в БД, переглядом параметрів потоків даних між хостами корпоративної мережі та переглядом параметрів потоків даних між вузлами.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

– Ініціалізація динамічних бібліотек.

– Виділення пам`яті ПЗ.

– Виведення на екран вікна завантаження ПЗ.

– Завантаження модулів корпоративної мережі.

– Виведення на екран головного вікна ПЗ.

– Перевірка наявності файлу формування нейронної мережі.

– Файли знайдено?

– Підключення файлу формування нейронної мережі.

– Всі модулі та бібліотеки підключено?

– Пошук об'єктів корпоративної мережі.

– Створення списку об'єктів.

– Читання та формування архітектури нейронної мережі.

– Виведення на екран схеми мережі.

– Отримання та аналіз мережних даних.

– Запис до журналу роботи ПЗ.

– Запит формування архітектури нейронної мережі?

– Формування початкових значень.

– Підпрограма формування нейронної мережі.

– Запит збору статистики?

– Підпрограма збору статистики корпоративної мережі.

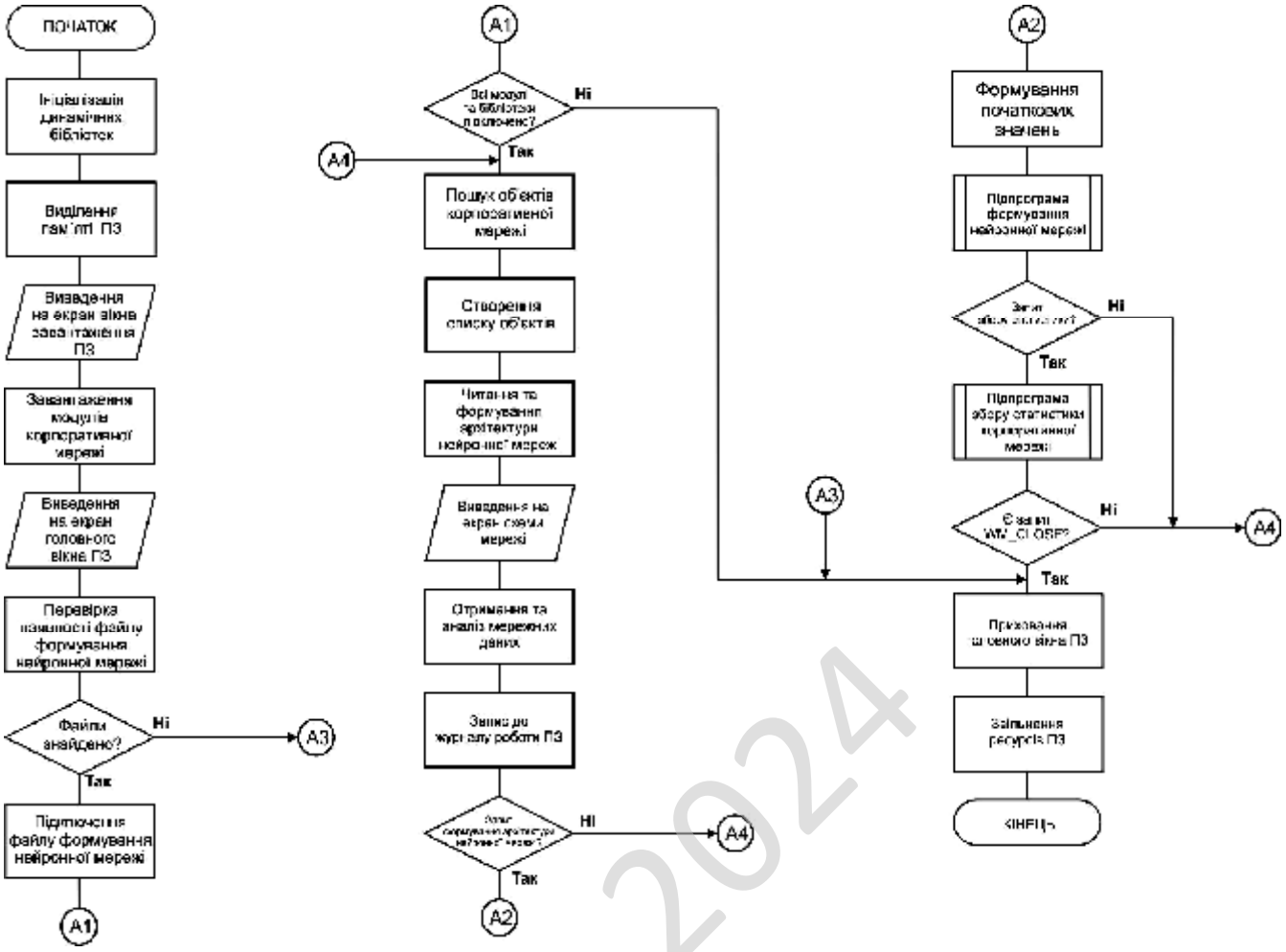

Рисунок 4.1 – Блок-схема основної програми

– Є запит WM_CLOSE?

– Приховання головного вікна ПЗ.

– Звільнення ресурсів ПЗ.

На рисунку 4.2 наведено блок-схему підпрограми збору статистики корпоративної мережі. Її робота складається з виконання наступних кроків:

– Приховання головного вікна ПЗ.

– Виведення вікна статистики роботи корпоративної мережі.

– Сканування корпоративної мережі.

– Збій роботи корпоративної мережі?

– Отримання кількості помилок вузлів корпоративної мережі.

– Формування корегуючого пакету.

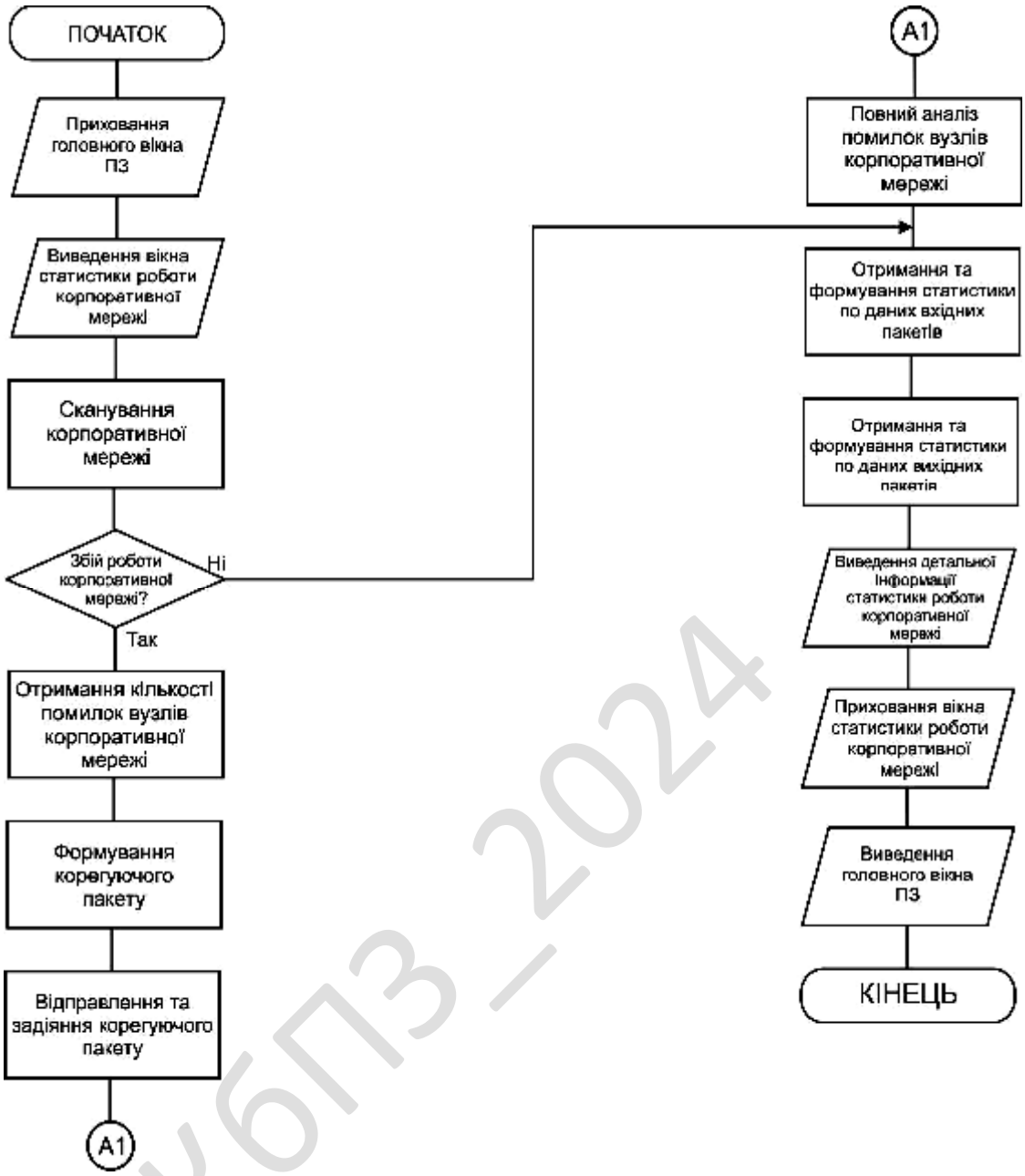

Рисунок 4.2 – Блок-схема підпрограми

– Відправлення та задіяння корегуючого пакету.

– Повний аналіз помилок вузлів корпоративної мережі.

– Отримання та формування статистики по даних вхідних пакетів.

– Отримання та формування статистики по даних вихідних пакетів.

– Виведення детальної інформації статистики роботи корпоративної мережі.

– Приховання вікна статистики роботи корпоративної мережі.

– Виведення головного вікна ПЗ.

Опис алгоритмів функціонування системи.

Розглянемо розроблені процедури та функції у бакалаврському проекті. Розглянемо процедуру що надає інформацію про локальні ресурси й можливості їх контролю.

DIPData – за допомогою цієї функції отримаємо дані про всі спільні ресурси. Оголошення функції для Windows 10/11:

```
DIPData :function (ServerData:Pwchar; Level:DWORD; Bufptr:Pointer; Prefmaxlen:DWORD; Entriesread, Totalentries, resume_handle:LPDWORD): DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера, на якому повинна виконаться функція, якщо виконуємо у себе, то даному параметрові можна привласнити NIL.

– Lvl повинен містити ідентифікатор структури.

– Bufptr повинен містити адресу покажчика на масив структур.

– Prefmaxlen повинен містити максимальну довжину поверненних даних у байтах, якщо не ставити обмеження, то даному параметрові потрібно привласнити DWORD(– 1) .

– Entriesread повинен містити покажчик на змінну, у яку запишеться кількість спільних ресурсів доступних на даний момент.

Результати виконання будуть збережені в масиві структур переданих функції при її виклику. Існує 6 типів структур переданих функції DIPData:

– SHARE_INFO_0 – тільки Windows 10/11 SP0.

– SHARE_INFO_1 – тільки Windows 10/11 SP1.

– SHARE_INFO_1 – тільки Windows 9x – Me.

– SHARE_INFO_2 – тільки Windows 10/11 SP2.

– SHARE_INFO_50 – тільки Windows 9x – Me .

– SHARE_INFO_502 – тільки Windows 10/11 SP3.

Структура SHARE_INFO_50, оголошення структури:

```
type
 TSHAREINFO50 = packed record
  DD_DIPname : array [0..12] of Char;
  DD_type:Byte;
  DD_flags:Word;
  DD_remark: Pchar;
  DD_path:Pchar;
  DD_rw_password:array [0..8] of Char;
DD_ro_password: array [0..8] of Char;
end;
```

Розроблені поля:

– DD_DIPname містить рядок з мережним ім'ям ресурсу;

– DD_type визначає тип ресурсу;

– DD_flags містить інформацію про права доступу до ресурсу;

– DD_remark покажчик на рядок необов'язкових коментарів;

– DD_path містить локальне розташування ресурсу;

– DD_rw_password містить пароль на запис – читання;

– DD_ro_password містить пароль на читання.

Розглянемо розроблену функцію DIPshareadd. Відкриття локального ресурсу. Оголошення функції для Windows 10/11:

```
var
DIPshareadd: function (ServerData: Pwidechar; level: DWORD; buf:Pointer;
parm_err: LPDWORD): DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера на якім повинна виконається функція, якщо відкриваємо локальний ресурс то даному параметру потрібно привласнити NIL;

– Lvl повинен містити ідентифікатор структури;

– buf повинен містити покажчик на структуру;

– parm_err містить покажчик помилки.

У функції для XP також використовується покажчик а не адреса покажчика. Розглянемо вихідний код виклику функції:

```
function Tmainform.Selectdirectory: String;
var
```

```
  lpitemid : Pitemidlist;
  Browseinfo : Tbrowseinfo;
  Displayname : array[0..MAX_PATH] of Char;
  Temppath : array[0..MAX_PATH] of Char;
begin
  Fillchar(Browseinfo, sizeof(Tbrowseinfo), #0);
  Browseinfo.hwndowner := Handle;
  Browseinfo.pszdisplayname := @Displayname;
  Browseinfo.lpsztitle := '';
  Browseinfo.ulflags := BIF_RETURNONLYFSDIRS;
  lpitemid := Shbrowseforfolder(Browseinfo);
  if Assigned(lpitemid) then begin
    Shgetpathfromidlist(lpitemid, Temppath);
    Globalfreeptr(lpitemid);
  end else Result := '';
  Result := String(Temppath);
end;
```

Для виконання функції необхідно додати в розділ Uses модулі Shelldipapi і Shldipobj. Ось сам код відкриття ресурсу:

```
procedure Tmainform.btnaddsharesclick(Sender: Tobject);
const
  STYPE_DISKTREE = 0;  ACCESS_ALL = 258;  DDF_FULL = 258;
var
  Flibhandle : Thandle;  Share9x : Tshareinfo50;  Sharent : Tshareinfo2;
  Tmpdir, Tmpname: String; Tmpdirnt, Tmpnament: Pwchar; OS: Boolean;
  Tmplength: Integer;
begin
  Tmpdir := Selectdirectory; //Визначаємо шлях до майбутнього ресурсу
//Визначаємо мережне ім'я
  Tmpname := Inputbox('Share name','Enter name','Test');
  if Tmpdir = '' then Exit;
  if not OS(OS) then Close; // З'ясовуємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
@DIPshareaddnt:=Getprocaddress(Flibhandle,'DIPshareadd');
    if not Assigned(DIPshareaddnt) then
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
```

```
    Tmplength := Sizeof(Widechar)*256; //Визначаємо необхідний розмір
    Getmem(Tmpnament, Tmplength); //Конвертуємо в Pwchar
    Stringtowidechar(Tmpname, Tmpnament, Tmplength);
    Sharent.SS2_DIPname := Tmpnament; //Ім'я
    Sharent.SS2_type := STYPE_DISKTREE; //Тип ресурсу
    Sharent.SS2_remark := ''; //Коментар
    Sharent.SS2_permissions := ACCESS_READ; //Доступ
    //Кількість максимальних підключень
    Sharent.SS2_max_uses := DWORD( - 1);
    //Кількість поточних підключень
    Sharent.SS2_current_uses := 0;
    Getmem(Tmpdirnt, Tmplength);
    Stringtowidechar(Tmpdir, Tmpdirnt, Tmplength);
    Sharent.SS2_path := Tmpdirnt; //Шлях до ресурсу
    Sharent.SS2_passwd := nil; //Пароль
    DIPshareaddnt(nil,2,@Sharent,nil); //Додаємо ресурс
    Freemem (Tmpnament); //звільняємо пам'ять
    Freemem (Tmpdirnt);
  end else begin //Код для 9x
    Flibhandle := Loadlibrary('PZ_2.DLL');
    if Flibhandle = 0 then Exit;
    @DIPshareadd := Getprocaddress(Flibhandle,'DIPshareadd');
    if not Assigned(DIPshareadd) then
    begin
      Freelibrary(Flibhandle);
      Close;
    end;
    Fillchar(Share9x.DD_DIPname, Sizeof(Share9x.DD_DIPname), #0);
    move(Tmpname[1],Share9x.DD_DIPname[0],Length(Tmpname)); //Ім'я
    Share9x.DD_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.DD_flags := DDF_RDONLY; //Доступ
    Fillchar(Share9x.DD_remark,
      Sizeof(Share9x.DD_remark), #0); //Коментар
    Fillchar(Share9x.DD_path,
      Sizeof(Share9x.DD_path), #0);
    Share9x.DD_path := Pansichar(Tmpdir); //Шлях до ресурсу
    Fillchar(Share9x.DD_rw_password,
      Sizeof(Share9x.DD_rw_password), #0);
//Пароль повного доступу
    Fillchar(Share9x.DD_ro_password,
    Sizeof(Share9x.DD_ro_password), #0); //Пароль для читання
    DIPshareadd(nil,50,@Share9x,Sizeof(Share9x));
```

```
  end;
  Freelibrary(Flibhandle);
end;
```

Змінні Tmpnament і Tmpdirnt звільняються тільки після виконання функції. Це критично, а якщо ні, то структура передана функції буде із двома "незаповненими" полями.

Даний код відкриває новий ресурс на повний доступ.

Одержання списку поточних сесій. Для визначення користувачів підключених до комп'ютера була розроблена функція PZ_DIPsession. Оголошення функції для Windows 10/11:

```
PZ_DIPsession: function(ServerData, Uncclientname, Username: Pwchar;
                        Level: DWORD; bufptr: Pointer; prefmaxlen:DWORD;
               entriesread, totalentries,
                        resume_handle: LPDWORD):DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція, якщо дивимося в себе то даному параметрові потрібно привласнити NIL.

– Uncclientname містить покажчик на рядок утримуючу ім'я сесії.

– Username – містить покажчик на рядок утримуючу ім'я користувача.

– Lvl – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адреса покажчика на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах.

– Entriesread повинен містити покажчик на змінну в яку запишеться кількість загальних ресурсів доступних на даний момент.

Для чіткого представлення часу роботи в бакалаврському проекті була розроблена функція, завдання якої перетворювати кількість секунд у більш звичну форму відображення.

```
function Tmainform.Cardinaltotimestr(Value:Cardinal):String;
var d,h,m,s: Real;
begin
    d:=0; h:=0; m:=0; s:=Value;
```

```
            if s > 59 then begin
                m:=int(s / 60);
                s:=s - (m*60);
            end;
            if m > 59 then begin
                h:=int(m/60);
                m:=m - (h*60);
            end;
            if h > 23 then begin
                d:=int(h/24);
                h:=h - (d*24);
            end;
            Result:='';
            if (d>0) then Result:=Result+floattostr(d)+' d. ';
            if (h<9) then Result:=Result+'0'+floattostr(h)+':' else
Result:=Result+floattostr(h)+':';
            if (m<9) then Result:=Result+'0'+floattostr(m)+':' else
Result:=Result+floattostr(m)+':';
            if (s<9) then Result:=Result+'0'+floattostr(s)     else
Result:=Result+floattostr(s);
        end;
```

Завершення сесій. Для завершення відкритих сесій розроблена функція PZ_DIPsessiondel. Оголошення функції для Windows 10/11:

```
        Var PZ_DIPsessiondel: function(ServerData, Uncclientname,
Username:Pwchar):DWORD;
```

Параметри:

ServerData – повинен містити ім'я віддаленого комп'ютера на якому повинна виконається функція;

uncclientname – повинен містити ім'я клієнта чия сесія завершується;

username – повинен містити ім'я користувача чия сесія завершується, якщо параметр NIL, завершаться всі сесії указані в параметрі uncclientname. Розглянемо вихідний код виклику функції:

```
        procedure Tmainform.btnclosesessionclick(Sender: Tobject);
        var
        OS: Boolean; Flibhandle : Thandle; Cnament: Pwidechar; Cname9x: Pansichar;
          Key:Smallint; i: Integer;
        begin
          if not OS(OS) then Close; //З'ясовуємо тип системи
```

```
  if not Assigned(lvsessions.Selected) then Exit;
  i:= lvsessions.Selected.Index; //Визначаємо номер обраної сесії
  if OS then begin
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
@PZ_DIPsessiondelnt := Getprocaddress(Flibhandle, 'PZ_DIPsessiondel');
    if not Assigned(PZ_DIPsessiondelnt) then
    begin Freelibrary(Flibhandle); Exit;
    end;
    //Перетворимо дані в необхідний вигляд
    Cnament := Pwchar(Widestring('\\'+lvsessions.Items.Item[i].Caption));
    PZ_DIPsessiondelnt(nil,Cnament,nil);
  end else begin
    Flibhandle := Loadlibrary('PZ_2.DLL');
    if Flibhandle = 0 then Exit;
@PZ_DIPsessiondel:=Getprocaddress(Flibhandle, 'PZ_DIPsessiondel');
    if not Assigned(PZ_DIPsessiondel) then
    begin Freelibrary(Flibhandle); Exit;  end;
    //Перетворимо дані в необхідний вигляд
    Cname9x := Pansichar(lvsessions.Items.Item[i].Caption);
    key := Sessionclosekey[i]; //Беремо ключ із масиву
    PZ_DIPsessiondel(nil,Cname9x,Key);
  end;
  Freelibrary(Flibhandle);
end;
```

Я реалізувала у бакалаврському проекті – оголошення структури Tshareinfo2Array = array [0..512] of Tshareinfo2; Тут пам'ять уже виділена.

Розглянемо розроблену функцію DIPsharedel яка дозволить закрити обраний загальний ресурс.

```
Var DIPsharedel:function (pszserver, pszDIPname:Pchar;
                          usreserved:Word):DWORD;
```

Параметри:

– ServerData повинен містити ім'я віддаленого комп'ютера, якщо закриваємо свої ресурси то даному параметру потрібно присвоїти NIL;

– DIPname покажчик на рядок утримуючу ім'я ресурсу, що закривається.

Обидві функції не використовують ніяких структур. У якості імені передається не шлях до ресурсу, а саме ім'я ресурсу яке я визначив за допомогою

коду (розглянутого вище). У випадку успішного виконання функцій, їх результат буде дорівнює нулю.

```
procedure Tmainform.btnclosesharesclick(Sender: Tobject);
var
  OS:Boolean;
  Flibhandle : Thandle;
  Name9x:array [0..12] of Char;
  Nament:Pwchar;
  i:Integer;
  Sharename: String;
begin
  if not OS(OS) then Close; //Визначаємо тип системи
  if lbxshares.Items.Count = 0 then Exit;
  for i:= 0 to lbxshares.Items.Count  - 1 do
    if lbxshares.Selected[i] then Break;  //Шукаємо обраний елемент
//Якщо не знайдений завершення роботи  ПЗ
  if not lbxshares.Selected[i] then Exit;
  Sharename := lbxshares.Items.Strings[i];
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
    @DIPsharedelnt := Getprocaddress(Flibhandle,'DIPsharedel');
    if not Assigned(DIPsharedelnt) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    i:= Sizeof(Widechar)*256;
    Getmem(Nament,i);  //Виділяємо пам'ять під змінну
    Stringtowidechar(Sharename,Nament,i); //Перетворимо в Pwidechar
    DIPsharedelnt(nil,Nament,0);  //Видаляємо ресурс
    Freemem(Nament); //Звільняємо пам'ять
  end else begin
    Flibhandle := Loadlibrary('PZ_2.DLL');
    if Flibhandle = 0 then Exit;
    @DIPsharedel := Getprocaddress(Flibhandle,'DIPsharedel');
    if not Assigned(DIPsharedel) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
```

```
        Fillchar(Name9x, Sizeof(Name9x), #0); //Очищаємо масив
        move(Sharename[1],Name9x[0],Length(Sharename));
      //Заповнюємо масив
        DIPsharedel(nil,@Name9x,0); //Видаляємо ресурс
      end;
      Freelibrary(Flibhandle);
    end;
```

Розглянемо розроблену функцію, яка визначає тип системи.

```
function Tmainform.OS(var Value: Boolean): Boolean;
    var Ver: Tosversioninfo;
        Bres: Boolean;
    begin
      Ver.dwosversioninfosize := Sizeof(Tosversioninfo);
      Bres := Getversionex(Ver);
      if not Bres then
      begin
        Result := False;
        Exit;
      end else
        Result := True;
      case Ver.dwplatformid of
        VER_PLATFORM_WIN32_XP: Value := True;
    //Windows 10/11  -  підходить
        VER_PLATFORM_WIN32_WINDOWS : Value := False;
    //Windows 9x - Me - не підходить
        VER_PLATFORM_WIN32s : Result := False;
    //Windows 3.x  - не підходить
      end;
    end;
```

Якщо дана функція повернула результат True, виходить, визначення версії системи пройшло успішно, і тип системи буде зберігатися в змінній Value, а якщо ні, то визначення типу системи, у цьому випадку прийде завершити виконання програми.

Дана функція буде практично найголовнішою, тому що вона буде вказувати у бакалаврському проекті яку частину коду виконувати.

Визначаємо, тип системи, завантажуємо необхідну бібліотеку, одержуємо адреси функцій і виконуємо функцію.

```
    procedure Tmainform.btngetsharesclick(Sender: Tobject);
```

```
var
  i:Integer;
  Flibhandle : Thandle;
  Sharent : Pshareinfo2Array;
  entriesread,totalentries:DWORD;
  Share : array [0..512] of Tshareinfo50;
  pcentriesread,pctotalavail:Word;
  OS: Boolean;
begin
  lbxshares.Items.Clear;
  if not OS(OS) then Close; //Визначаємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('_R1.DLL'); //Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
//Зв'язуємо функцію
    @DIPDatant := Getprocaddress(Flibhandle,'DIPData');
    if not Assigned(DIPDatant) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Sharent := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if DIPDatant(nil,2,@Sharent,DWORD( - 1),
             @entriesread,@totalentries,nil) <> 0 then
    begin //Якщо виклик невдалий вивантажуємо бібліотеку
      Freelibrary(Flibhandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread  -  1 do
            lbxshares.Items.Add(String(Sharent[i].SS2_DIPname));
  end else begin
    Flibhandle := Loadlibrary('PZ_2.DLL');
//Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
    //Зв'язуємо функцію
    @DIPData := Getprocaddress(Flibhandle,'DIPData');
    if not Assigned(DIPData) then
//Перевірка
    begin
      Freelibrary(Flibhandle);
```

```
        Exit;
      end;
      if DIPData(nil,50,@Share,Sizeof(Share),
              @pcentriesread,@pctotalavail) <> 0 then //Виклик функції
      begin //Якщо виклик невдалий вивантажуємо бібліотеку
        Freelibrary(Flibhandle);
        Exit;
      end;
      if pcentriesread > 0 then //Обробка результатів
      for i:= 0 to pcentriesread  -  1 do
           lbxshares.Items.Add(String(Share[i].DD_DIPname));
    end;
    Freelibrary(Flibhandle); // вивантажити бібліотеку
  end;
```

При виконанні цього коду Listbox заповниться назвами загальних ресурсів, які беруться з масиву структур. Масив заповнюється в результаті виконання функції.

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7564:2014 («Купина»). В Україні на основі консервативного підходу із залученням відомих і добре досліджених конструкцій була розроблена ґеш-функція, що базується на новому блоковому шифрі „Калина” (ДСТУ 7624:2014).

Національний стандарт ДСТУ 7564:2014 визначає криптографічну функцію ґешування „Купина”, додатковий режим її застосування для формування коду автентифікації повідомлення (імітовставки), а також значення для перевірки реалізацій.

Для скорочення обсягу тексту національного стандарту була застосована математична нотація, що дозволяє отримати точний і компактний запис.

Водночас, такий підхід може ускладнювати сприйняття сутності алгоритму для фахівців, що не мають фундаментальної криптологічної освіти.

У роботі наводиться розгорнутий альтернативний опис функції ґешування „Купина” із позначеннями, традиційними для галузі комп’ютерних наук.

Термінологія та позначення

Вектор ініціалізації – бітова послідовність фіксованої довжини (512 або 1024 біта), що використовується як початкове значення при обчисленні ґеш-значення.

Внутрішній стан – бітова послідовність фіксованої довжини (512 або 1024 біта), що є проміжним значенням на кожній ітерації перетворення функції ґешування, а також вхідним та вихідним значеннями перетворень P і Q; для цих перетворень внутрішній стан подається як матриця розміром $8 \times c$ байт.

Ґеш-значення (ґеш-вектор) – бітова послідовність фіксованої довжини ($n = 8 \cdot s$, $s \in \{1,2,...,64\}$), що є результатом роботи функції ґешування.

Доповнення – вставка додаткових біт у кінець повідомлення для отримання кратності довжини бітової послідовності довжині внутрішнього стану функції ґешування.

Повідомлення – бітова послідовність довжини від 0 біт (порожній рядок) до 2^{96}-1 біт.

Функція стиснення – ітеративне перетворення, що відображає l-бітний блок повідомлення та l-бітне значення, отримане функцією стиснення на попередньому кроці, у нове l-бітне значення.

Далі використовуються наступні позначення:

– $\oplus$ – додавання за модулем 2 (XOR);

– 0x – префікс числа, що записане у шістнадцятковій системі числення;

– $a \bmod b$ – ціле невід'ємне число, що дорівнює залишку від ділення цілого числа a на натуральне число b;

– B_i – i -й байт вхідної послідовності;

– C^i – константа перетворення XORRoundKey або Add64RoundKey для i -го циклу;

– c – кількість стовпців внутрішнього стану в матричному поданні;

– φ – функція стиснення;

– H – визначена у стандарті функція ґешування;

– $H(M)$ – результат обчислення функції ґешування для повідомлення M (ґешзначення);

– IV – вектор ініціалізації;

– l – розмір внутрішнього стану функції ґешування (у бітах), $l \in \{512,1024\}$;

– M – повідомлення;

– m_i – i -й блок повідомлення M;

– n – довжина обчисленого ґеш-значення;

– N – довжина повідомлення M без доповнення;

– P, Q – складові перетворення функції стиснення;

– P_{512} – перетворення P для 512-бітного внутрішнього стану;

– P_{1024} – перетворення P для 1024-бітного внутрішнього стану;

– Q_{512} – перетворення Q для 512-бітного внутрішнього стану;

– Q_{1024} – перетворення Q для 1024-бітного внутрішнього стану;

– r – кількість ітерацій у перетвореннях P і Q ($r = t$ в ДСТУ 7564:2014);

– S – внутрішній стан ґеш-функції;

– t – кількість блоків m, з яких складається повідомлення M, включаючи доповнення;

– v_i – i -й біт вхідної послідовності;

– Ω – завершальне перетворення;

– Купина-n – режим використання функції ґешування з усіченням обчисленого ґешзначення до розміру n біт.

Загальні положення

Під функцією ґешування H розуміється залежне від вектора ініціалізації IV відображення послідовності біт M у ґеш-значення $H(M)$ фіксованої довжини n.

ДСТУ 7564:2014 визначає функцію ґешування, яка виконує перетворення «Купина-256» або «Купина-512», що забезпечують обчислення ґеш-значення з довжинами 256 або 512 біт відповідно.

Ґеш-значення довжиною 256 бітів додатково може бути усічено до бітової послідовності довжиною від 8 до 248 біт з кроком у 8 біт, 512 бітів може бути усічене до бітової послідовності довжиною від 264 до 504 біт з кроком у 8 біт.

Режим роботи для формування ґеш-значення довжиною n біт позначається як «Купина-n».

Основними режимами роботи функції ґешування, що рекомендуються до застосування, є «Купина-256», «Купина-384» і «Купина-512».

Структура перетворення

Функція ґешування, визначена в ДСТУ 7564:2014, формує ґеш-значення для повідомлення, що складається з бітової послідовності довжини від 0 біт (порожній рядок) до 2^{96}-1 біт.

При формуванні ґеш-значення повідомлення доповнюється, далі поділяється на l-бітні блоки $m_0,\ldots, m_t$, після чого виконується обробка кожного блоку шляхом ітеративного виконання функції стиснення φ.

При цьому формуються значення $h_i = \varphi(h_{i-1}, m_i)$ де $i = 1,...,t$, а початкове значення $h_0 = \text{IV}$.

Після обробки останнього блоку повідомлення результуюче ґеш-значення обчислюється як $H(M) = \Omega(h_t)$, де Ω – завершальне перетворення, що повертає n - бітне значення, кратне 8 ($0 < n \le l/2$).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню яке складається з: Формат; Налаштування; Довідка; Автор розробки.

– Поле відображення IP адреси та порту.

– Поточного стану системи.

– Апаратне обрання мережної карти.

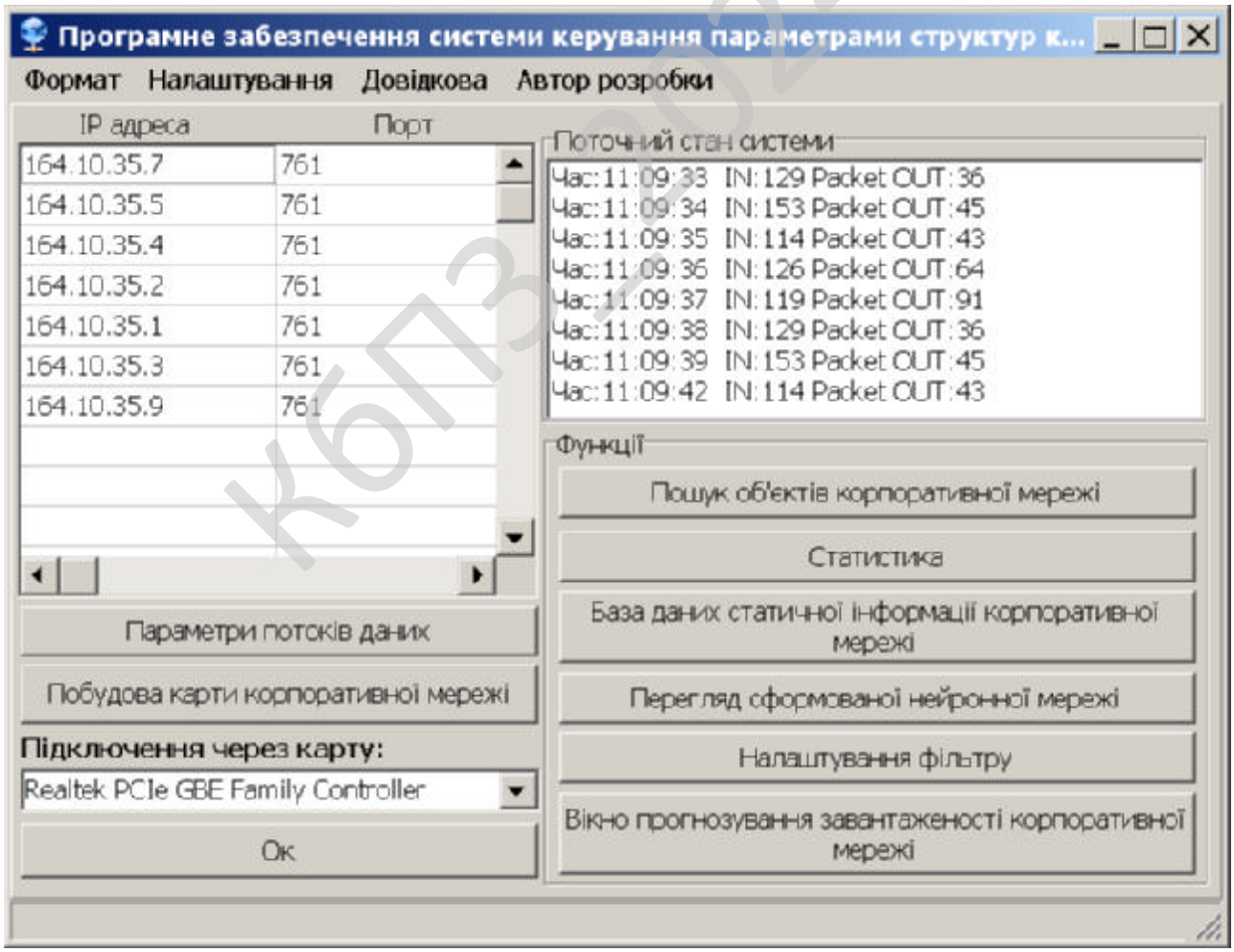

Рисунок 5.1 – Головне вікно програми

– Функціоналу: Параметри потоків даних; Побудова карти корпоративної мережі; Пошук об'єктів корпоративної мережі; Статистика; База даних статичної інформації корпоративної мережі; Перегляд сформованої нейронної мережі; Налаштування фільтру; Вікно прогнозування завантаженості корпоративної мережі.

На рисунку 5.2 зображено форму авторського права. Було обрано Shareware умову розповсюдження. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

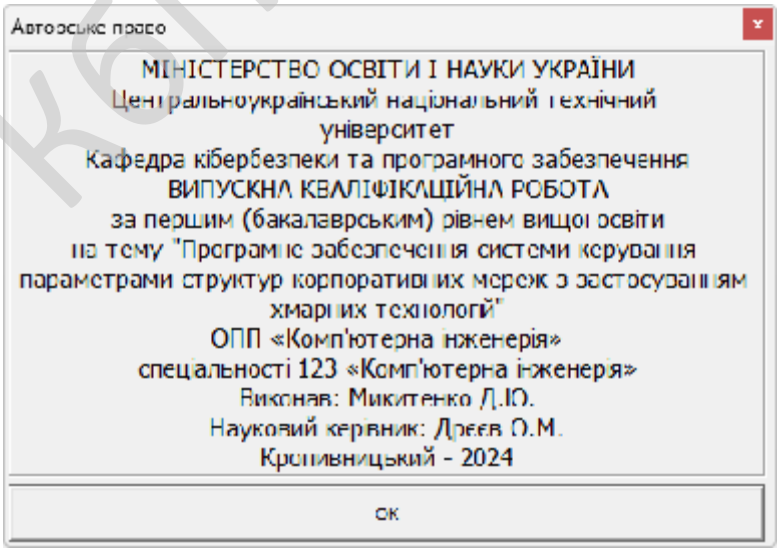

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

– Досліджена система керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

– На основі отриманих результатів досліджень створена програмна реалізація системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

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. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 7564:2014.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

2. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

4. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

5. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

6. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

7. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

8. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

9. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

10. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

11. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of

Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

12. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

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

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

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

16. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

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

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

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

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

24. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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

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

27. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

28. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

30. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

31. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

32. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду

абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

33. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

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

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

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

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

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

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

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

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

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

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

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

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

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

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

51. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

52. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій;

– 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 86 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 14.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Дрєєв О.М.

Програмне забезпечення системи керування параметрами структур корпоративних мереж з застосуванням хмарних технологій

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 48

Літера: РП

Кропивницький – 2024 року

Файл Main.pas – основне тіло розробленого ПЗ

```
unit Main; // назва модулю
{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Тема диплому: Програмне забезпечення системи керування
параметрами структур корпоративних мереж з застосуванням хмарних технологій
Виконав: студент 4 курсу, групи КІ-20
напряму підготовки (спеціальності) 123 Комп'ютерна інженерія
Микитенко Даниїл Юрійович
2024 рік
Керівник: Дреєв О.М.
}
interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
```

```
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
         ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
   function OpenEnum(const NetContainerToOpen: PNetResource;
         ResScope, ResType, ResUsage: DWORD): THandle;
    function EnumResources(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
```

```
    Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
   Sesi50_username    : PChar;
   sesi50_key         : Cardinal;
   sesi50_num_conns   : Word;
   sesi50_num_opens   : Word;
   sesi50_time        : Cardinal;
   sesi50_idle_time   : Cardinal;
   sesi50_protocol    : Byte;
   pad1               : Byte;
 end;

type
 TFileInfo3 = packed record
   fi3_id          : DWORD;
   fi3_permissions : DWORD;
   fi3_num_locks   : DWORD;
   fi3_pathname    : PWChar;
   fi3_username    : PWChar;
 end;
 PFileInfo3 = ^TFileInfo3;
 TFileInfo3Array = array[0..512] of TFileInfo3;
 PFileInfo3Array = ^TFileInfo3Array;

type
 TFileInfo50 = packed record
   fi50_id          : Cardinal;
   fi50_permissions : WORD;
   fi50_num_locks   : WORD;
   fi50_pathname    : PChar;
   fi50_username    : PChar;
   fi50_sharename   : PChar;
 end;

type
  TMibIfRow = packed record
    wszName           : array[0..255] of WideChar;
    dwIndex           : DWORD;
    dwType            : DWORD;
    dwMtu             : DWORD;
    dwSpeed           : DWORD;
    dwPhysAddrLen     : DWORD;
    bPhysAddr         : array[0..7] of Byte;
    dwAdminStatus     : DWORD;
    dwOperStatus      : DWORD;
    dwLastChange      : DWORD;
    dwInOctets        : DWORD;
    dwInUcastPkts     : DWORD;
    dwInNUCastPkts    : DWORD;
    dwInDiscards      : DWORD;
    dwInErrors        : DWORD;
    dwInUnknownProtos : DWORD;
    dwOutOctets       : DWORD;
    dwOutUCastPkts    : DWORD;
    dwOutNUCastPkts   : DWORD;
    dwOutDiscards     : DWORD;
    dwOutErrors       : DWORD;
    dwOutQLen         : DWORD;
    dwDescrLen        : DWORD;
    bDescr            : array[0..255] of Char;
  end;
  TMibIfArray = array [0..512] of TMibIfRow;
```

```
  PMibIfRow = ^TMibIfRow;
  PMibIfArray = ^TMibIfArray;

type
 TMibIfTable = packed record
   dwNumEntries     : DWORD;
   Table            : TMibIfArray;
 end;
 PMibIfTable = ^TMibIfTable;
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
```

```
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;

var
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function(pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

////////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
//  щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
//  Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана
```

```
  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої корпоративної
мережі
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin
//Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
//Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil;
//Очищаємо покажчик на масив структур
//Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                  @entriesread,@totalentries,nil) <> 0 then
    begin
//Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then
//Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
//Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
```

```
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle);
//Не забуваємо вивантажити бібліотеку
end;

//  Закриття загального ресурсу

procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close;
//Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;
//Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit;
//Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);
//Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i);
//Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);
//Видаляємо ресурс
    FreeMem(NameNT);
//Звільняємо пам' ять
  end else begin
//Код для 9 x-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then
//Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0);
//Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName));
//Заповнюємо масив
    NetShareDel(nil,@Name9x,0);
//Видаляємо ресурс
  end;
```

```
  FreeLibrary(FLibHandle);
end;
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
end;

//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory;
//Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' );
//Визначаємо ім' я під яким він буде видний у корпоративної мережі
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close;
//З'ясовуємо тип системи

  if OS then begin
// Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256;
//Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength);
//Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
```

```
    ShareNT.shi2_netname := TmpNameNT;
//Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE;
//Тип ресурсу
    ShareNT.shi2_remark := ' ' ;
//Коментар
    ShareNT.shi2_permissions := ACCESS_ALL;
//Доступ
    ShareNT.shi2_max_uses := DWORD(-1);
// Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0;
// Кіл-ть поточних підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT;
//Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE;
//Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL;
//Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0);
//Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0);
//Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

//
//  активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати Кіл-ть секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
```

```
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

{
  Одержання списку сесій досліджуємої корпоративної мережі
}

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname);
//Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);
//Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //
//Час активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin
```

```
//Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname);
//Ім' я комп' ютера досліджуємої корпоративної мережі
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //
//Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time));
//Час активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key;
//Унікальний ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
```

```
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

{
  Одержання списку відкритих файлів досліджуємої корпоративної мережі
}

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
```

```
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

{
  Визначаємо вхідний / вихідний трафік досліджуємої корпоративної мережі
}

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
```

```
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' DAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //
// Усього прийнято байт з досліджуємої корпоративної мережі  (DATA)
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));
//Усього відправлено байт у досліджуєму мережу
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true;
//Не забуваємо активувати таймер
end;


 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                NetContainerToOpen, hNetEnum))
 then ShowMessage(' Помилка!' )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
```

```
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                                  @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!' );
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
```

```
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл About.pas – довідкової інформації

```
unit About;

interface

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

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```

Файл data.pas- статистика роботи корпоративної мережі

```
unit Stat;

interface

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

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
```

```
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadD then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл DAPI.pas - обробка API функцій розробленої системи

```
unit DAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] = ( ' Невизначений' , ' Передача' ,
' Рівень до рівня' , ' ' , ' Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'    );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , ' loopback' ,
' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
    ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , ' tokenring'
,
     ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' PPP' ,     ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
// Кінець змін в типі адаптерів

//-------для інших MS заготовочних файлів------------------
```

```
  MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
  MAXLEN_PHYSADDR = 8; { iprtrmib.h }
  MAXLEN_IFDESCR = 256; {-"--     }

//-----------------------------------

type
  TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

//---IP адресні структури-------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record // для використання у зв' язних списках
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;

//-----Fixed Info структура----------------------

  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
    DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

{ структура мережного інтерфейсу досліджуємої корпоративної мережі
(DATA)
}

//////////////////////
//                                                                  //
// Наступне є діючими станами для WAN да LAN інтерфейсів. //
// Порядок станів створений для визначення. Для      //
// стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED //
// може передавати деякі дані. Стан < DISCONNECTED може   //
// не передавати дані.                                              //
// карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
// причин. Крім невдачі з'єднання.                                  //
//                                                                  //
// NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не працює
//
//                      або не з'єднується з картою.        //
// UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      не з'єднується за потрібний час.
//
// DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
//                      не з'єднується.
//
// CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
//                      з сайтом, якого немає.              //
// CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт      //
//                      з'єднується.                                  //
// OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
//                      в праці.                                    //
```

```
//                                                                    //
// Усі дії користувачів записуються до  MIB-II значення    //
// можуть бути використовані                                          //
//                                                                    //
//////////////////////////

const
// дані додані до ipifcons.h
  IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  IF_OPER_STATUS_UNREACHABLE = 1 ;
  IF_OPER_STATUS_DISCONNECTED = 2 ;
  IF_OPER_STATUS_CONNECTING = 3 ;
  IF_OPER_STATUS_CONNECTED = 4 ;
  IF_OPER_STATUS_OPERATIONAL = 5 ;

  MIB_IF_TYPE_OTHER = 1 ;
  MIB_IF_TYPE_ETHERNET = 6 ;
  MIB_IF_TYPE_TOKENRING = 9 ;
  MIB_IF_TYPE_FDDI = 15 ;
  MIB_IF_TYPE_PPP = 23 ;
  MIB_IF_TYPE_LOOPBACK = 24 ;
  MIB_IF_TYPE_SLIP = 28 ;

  MIB_IF_ADMIN_STATUS_UP = 1 ;
  MIB_IF_ADMIN_STATUS_DOWN = 2 ;
  MIB_IF_ADMIN_STATUS_TESTING = 3 ;

  MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
  MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
  MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
  MIB_IF_OPER_STATUS_CONNECTING = 3 ;
  MIB_IF_OPER_STATUS_CONNECTED = 4 ;
  MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

type
  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;      // дивись MIB_IF_TYPE
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
    dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;

 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIfRow;
  end;
```

```
//---ADAPTER INFO структура-------------------------

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       // дані
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;    //
дані
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
    SpareStuff: array [1..200] of char ;   // дані- простір для списку IP адрес
досліджуємої корпоративної мережі (DATA)
  end;

//--------TCP структура---------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА --------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
```

```
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
```

```
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ----------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до DAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
```

```
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі помилки при використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    DDLL = ' DAPI.DLL' ;
var
    DModule: THandle;

    function LoadD: Boolean;

implementation

function LoadD: Boolean;
begin
    Result := True;
    if DModule <> 0 then Exit;

// відкрити DLL
    DModule := LoadLibrary (DDLL);
    if DModule = 0 then
    begin
        Result := false;
        exit ;
    end ;
    GetAdaptersInfo := GetProcAddress (DModule, ' GetAdaptersInfo' ) ;
    GetNetworkParams := GetProcAddress (DModule, ' GetNetworkParams' ) ;
    GetTcpTable := GetProcAddress (DModule, ' GetTcpTable' ) ;
    GetTcpStatistics := GetProcAddress (DModule, ' GetTcpStatistics' ) ;
    GetUdpTable := GetProcAddress (DModule, ' GetUdpTable' ) ;
    GetUdpStatistics := GetProcAddress (DModule, ' GetUdpStatistics' ) ;
    GetIpStatistics := GetProcAddress (DModule, ' GetIpStatistics' ) ;
    GetIpNetTable := GetProcAddress (DModule, ' GetIpNetTable' ) ;
    GetIpAddrTable := GetProcAddress (DModule, ' GetIpAddrTable' ) ;
    GetIpForwardTable := GetProcAddress (DModule, ' GetIpForwardTable' ) ;
    GetIcmpStatistics := GetProcAddress (DModule, ' GetIcmpStatistics' ) ;
```

```
    GetRTTAndHopCount := GetProcAddress (DModule, ' GetRTTAndHopCount' ) ;
    GetIfTable := GetProcAddress (DModule, ' GetIfTable' ) ;
    GetIfEntry := GetProcAddress (DModule, ' GetIfEntry' ) ;
    GetFriendlyIfIndex := GetProcAddress (DModule, ' GetFriendlyIfIndex' ) ;
end;

initialization
    DModule := 0 ;
finalization
    if DModule <> 0 then
    begin
        FreeLibrary (DModule) ;
        DModule := 0 ;
    end ;

end.
```

Файл DataIP.pas- функції роботи з IP-протоколом

```
unit DataIP;


interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, DApi;

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { File Transfer Protocol- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol       }
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service           }
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),
{ Протокол Simple Netw. Management Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
    );
```

```
//------перетворення ICMP кодів помилок до рядків-------------

const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ПОМИЛКА' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


//-----Перетворення різних перерахованих величин у рядки---------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , '  MIL-STD-1778' ,
    ' Van Jacobson' , ' інший'  );

  IPForwTypes   : array[1..4] of string =
    ( ' інший' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для DNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для DAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
```

```
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

function DAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows): integer ;
procedure Get_AdaptersInfo( List: TStrings );
function DNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function DTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function DUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function DIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function DIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function DIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// функції перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
```

```
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

//-----------------------------------
{ перетворення числового MAC-адреса  до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := '  00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2) + '-' ;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення IP-адреси в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, '.'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
```

```
  Result := Swap( WORD( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в мережний байт типу string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;

//-----------------------------------
{ перетворення номеру порту в сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку, якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;

//-----------------------------------
{ голона частина, фіксація мережний параметрів досліджуємої корпоративної мережі
(DATA)}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := DNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function DNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
```

```
    result := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

//-----------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//-----------------------------------

// включення байтів у/з для кожного адаптера

function DIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadD then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
```

```
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := DIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає '  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
               ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // дані, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function DIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadD then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------
```

```
{ інформація про інстальовані адаптери }

function DAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows): integer
;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                  AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
                begin
                     SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                     SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
```

```
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

    // беремо список IP адрес для GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.GatewayList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].GatewayList) <= I then
                        SetLength (AdpRows [AdpTot].GatewayList, I -2) ;
        end ;
        AdpRows [AdpTot].GatewayTot := I ;

    // беремо список IP адрес для  GatewayList
        I := 0 ;
        PIpAddr := @AdapterInfo^.DHCPServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].DHCPServer) <= I then
                        SetLength (AdpRows [AdpTot].DHCPServer, I -2) ;
        end ;
        AdpRows [AdpTot].DHCPTot := I ;

    // беремо список IP адрес для PrimaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].PrimWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].PrimWINSTot := I ;

   // беремо список IP адрес для SecondaryWINSServer
        I := 0 ;
        PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                        SetLength (AdpRows [AdpTot].SecWINSServer, I -2) ;
        end ;
        AdpRows [AdpTot].SecWINSTot := I ;

        AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
        AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                    SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
      end ;
      SetLength (AdpRows, AdpTot) ;
    end ;
finally
    FreeMem( pBuf );
end ;
```

```
end ;

procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := DAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' данних немає '  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |'  + Description ); // jpt : не
використовується
            List.Add( Format(' %8.8x | %6s | %16s | %2d | %16s | %16s | %16s' ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : не використовується
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /'  + IPMaskList [J] + '
| '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '    '  ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//-----------------------------------
{ моніторимо час доступу до IP досліджуємої корпоративної мережі (DATA)}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
```

```
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
```

```
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо останій запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s | %15s : %-7s | %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) =-1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadD then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + IntToStr( dwRTOMax ) +
'  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
```

```
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function DTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-----------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
```

```
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  // перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації досліджуємої корпоративної мережі
(DATA); }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
```

```
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   // дані
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadD then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
```

```
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована   : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function DIPStatistics (var IPStats: TMibIPStats): integer ;      // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)    : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів         : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
```

```
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function DUdpStatistics (UdpStats: TMibUDPStats): integer ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Прийнято повідомлень    : ' + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                 : ' + IntToStr( dwErrors ) );
  ICMPIn.Add( ' Розташування недосягенно  : ' + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищений         : ' + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами     : ' + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено       : ' + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено           : ' + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит       : ' + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo відповідь           : ' + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : ' + IntToStr( dwTimeStamps ) );
    ICMPIn.Add( ' Відповідь мітки часу    : ' + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адрес : ' + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : ' + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : ' + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : ' + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : ' + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : ' + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : ' + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : ' + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено            : ' + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Exo запит        : ' + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo відповідь         : ' + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : ' + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : ' + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: ' + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : ' + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

//-----------------------------------
```

```
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- обробка з'єднань TCP/IP досліджуємої корпоративної мережі

```
unit TCP_IP;

interface

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

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
```

```
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ‘  Час запиту ‘
      + inttostr( rtt ) + ‘  ms, ‘
      + inttostr( HopCount )
      + ‘  hops to : ‘  + edtRTTI.Text
      )
  else
    ShowMessage( ‘ Відбулася помилка:’  + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadD then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
  ShowMessage( ‘ Інтернет помічник DLL не є доступним, або не підтримується’ ) ;
end;

end.
```