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 автоматизованого керування кабельною інфраструктурою АІМ”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Усатенко К.О.
«____» _____ 2024 р.

Керівник проекту
докт. техн. наук, професор
_____ Коваленко О.В.
«____» _____ 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

Усатенку Костянтину Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи автоматизованого керування кабельною інфраструктурою AIM*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 132-02 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 автоматизованого керування кабельною інфраструктурою AIM*

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
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Усатенко К.О.
(прізвище та ініціали)

АНОТАЦІЯ

Усатенко К.О. Програмне забезпечення системи автоматизованого керування кабельною інфраструктурою AIM. 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 автоматизованого керування кабельною інфраструктурою AIM.

Метою розробки є програмне забезпечення системи автоматизованого керування кабельною інфраструктурою AIM.

Результат роботи – програмна реалізація системи автоматизованого керування кабельною інфраструктурою AIM.

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 RAD Studio Delphi 10.

Ключові слова: комп'ютерна інженерія, керування кабельною інфраструктурою

ABSTRACT

Usatenko K.O. AIM cable infrastructure automated management system software. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the system of automated management of cable infrastructure AIM.

The goal of the development is the software of the AIM cable infrastructure automated management system.

The result of the work is the software implementation of the AIM cable infrastructure automated management system.

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

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 RAD Studio Delphi 10 environment.

Keywords: computer engineering, cable infrastructure management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ЛОМ	–	локальна обчислювальна мережа
ASP		Active Server Pages – активні серверні сторінки
DHCP	–	Dynamic Host Configuration Protocol – протокол динамичної конфігурації вузла
HTTP	–	HyperText Transfer Protocol – протокол передачі гіпер тексту
IMAP	–	Internet Message Access Protocol – протокол доступу до електроної пошти Інтернету
ICMP	–	Internet Control Message Protocol – міжмережний протокол керуючих повідомлень
MMC	–	Microsoft Management Console
POP3	–	Post Office Protocol Version 3 – протокол поштового відділення, версія 3
SQL	–	Structured Query Language – мова структурованих запросів
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
SNMP	–	Simple Network Management Protocol – простий протокол керування мережею
Syslog	–	стандарт відправки повідомлень про зміни які відбуваються в мережі
UDP	–	User Datagram Protocol – протокол користувальницьких дейтаграм

ВСТУП

Актуальність теми. Стандарт для систем автоматизованого керування кабельною інфраструктурою (Automated Infrastructure Management, AIM) розроблений з метою підвищення надійності мереж, збільшення терміну служби встаткування, зниження працезатрат і зменшення часу простою. При цьому повинні підтримуватися всі мережні сервіси й з'єднання.

Крім вимог до видимості й контролюємості в реальному часі всіх фізичних мережних компонентів, стандарт визначає загальні процеси в частині поліпшеного документування мережі, автоматичних повідомлень і попереджень, а також удосконаленого керування переміщеннями, додаваннями й змінами (MAC). Керування волокном високої щільності, з'єднаннями однієї крапки з декількома й архітектурами з розподіленим ядром (leaf-spine) високої зв'язності жадає від ваших кабельних систем безпрецедентних наочності, інтелектуальності й керованості. Завдяки інтелектуальному апаратному й програмному забезпеченню, а також панелі керування на базі веб-технологій дозволяє користувачам здійснювати моніторинг кабельної інфраструктури, вести облік подій, що відбуваються в ній, і відслідковувати місце розташування всіх підключених мережних пристроїв у реальному часі.

– Керування на фізичному рівні з оглядом всіх підключених серверів, комутаторів і пристроїв зберігання даних у реальному часі.

– Підвищення ефективності IT-інфраструктури завдяки швидкому виявленню невикористовуваних IT-ресурсів і можливості повторного використання окремих елементів кабельної інфраструктури.

– Спрощення й прискорення робочих процесів завдяки потужній системі автоматизації процесів і функціям звітності.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи автоматизованого керування кабельною інфраструктурою AIM.

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

– Огляд існуючих систем автоматизованого керування кабельною інфраструктурою АІМ.

– Дослідження системи автоматизованого керування кабельною інфраструктурою АІМ.

– Програмна реалізація системи автоматизованого керування кабельною інфраструктурою АІМ.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматизованого керування кабельною інфраструктурою АІМ.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизованого керування кабельною інфраструктурою АІМ, 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

На початку 2015 року Міжнародна організація по стандартизації (International Standardization Organization; ICO; ISO) і Міжнародна електротехнічна комісія (International Electrotechnical Commission; IEC; МЕК) опублікували новий добровільний міжнародний стандарт на основі консенсусу на системи автоматичного керування інфраструктурою. Поширення цього документа повинне сприяти подальшому розвитку ринку комплектів програмного забезпечення для керування інфраструктурою центра обробки даних (Data Center Infrastructure Management; DCIM). Після появи універсального стандарту софтверні компанії зможуть привести свої продукти у відповідність вимогам цього документа й тим самим вселити в клієнтів упевненість щодо надійності свого ПЗ. Документ називається ICO / МЕК 18598 «Інформаційні технології – Системи автоматизованого керування інфраструктурою (Automated Infrastructure Management; AIM) – Вимоги, обмін даними й додатка». Роботу над ним ведуть експерти з 29 країн, що входять до складу підкомітету (ПК) 25 «Реалізація можливості обміну інформацією між IT-Устаткуванням» при спільному технічному комітеті (СТК) 1 «Інформаційні технології», сформованому ICO й МЕК. Стандарт створюється за підтримкою Асоціації телекомунікаційної промисловості США (Telecommunications Industry Association; TIA)

1.2 Область застосування

В тексті стандарту ICO / МЕК 18598 на системи автоматизованого керування інфраструктурою будуть серед іншого присутні рекомендації щодо проектування й розробки інструментів, призначених для відстеження яких-

небудь змін фізичних характеристик мережної інфраструктури і її елементів, включаючи комутатори й патч-панелі.

Незважаючи на те, що присутні в документі рекомендації будуть стосуватися будь-яких інструментів керування мережною інфраструктурою, заточений стандарт буде саме під центр обробки даних.

Завдяки новому стандарту значно спроститься механізм збору в режимі реального часу необхідної DCIM-рішенням технічної інформації відносно стану кабельної інфраструктури, керування активами й незапланованими змінами. Таким чином, розроблювачі, які раніше робили ставку переважно на інструменти для керування інфраструктурою охолодження й електроживлення ЦОД, після випуску стандарту зможуть із легкістю розширити функціонала своїх DCIM-рішень за рахунок додавання в них модулів керування активами й мережною інфраструктурою.

Експерт, що також бере участь у роботі над новим стандартом, відзначив, що творці DCIM-рішень уже давно мали потребу в стандарті начебто ICO / МЕК 18598, тому що їм найчастіше доводилося уступати ринкові частки безпосередньо виробникам допоміжного устаткування для ЦОД (систем охолодження й електропостачання, а також стійкових рішень), що просувають свої власні запатентовані технології для керування інфраструктурою.

ICO / МЕК 18598 – перший добровільний міжнародний стандарт на основі консенсусу, що містить термін «DCIM». Стандарт серед іншого спростив механізми формування замовлень на виконання робіт для інженерів, що здійснюють переміщення, додавання й зміну тих або інших елементів інфраструктури ЦОД. Всі тому, що наведені в документі рекомендації дозволять розроблювачам DCIM-рішень швидко створювати й впроваджувати відповідні інструменти, що враховують обмеження, що накладаються електроенергетичною інфраструктурою, доступним простором і несучою здатністю фальшполу стосовно до будь-якому машзалу їх ЦОД.

Слід зазначити, що з моменту початку роботи над стандартом ICO / МЕК 18598 до проекту приєдналися багато компаній. Вони об'єднали сили й поділилися досвідом, щоб додати в документ якнайбільше корисної інформації, а також з метою внесення в текст стандарту рекомендацій щодо оптимізації DCIM-рішень для ефективної взаємодії із системами керування будинком (Building Management System BMS), системами освітлення, електропостачання, кондиціювання й вентиляції.

Необхідність таких стандартів є досить очевидної, з огляду на безперервне підвищення складності мережних архітектур, збільшення кількості різних елементів інфраструктури в машзалах серверних ферм і підвищення розмаїтості інтерфейсів передачі технічної інформації. Наша зростаюча залежність від ЦОД робить їхні простої усе більше «дорогим задоволенням».

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизованого керування кабельною інфраструктурою AIM, 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

Інтелектуальна система R&MinteliPhy

Інтелектуальна система R&MinteliPhy відкриває нову еру для ефективного централізованого керування фізичної IT інфраструктурою ЦОД, кампусної або корпоративної мережі.

Повна автоматизація керування інфраструктурою AIM (Automated Infrastructure Management) у реальному часі без застосування спеціалізованих пасивних компонентів (кабелів, комутаційних панелей, сполучних шнурів) дозволяє, використовуючи безконтактну сенсорну технологію RFID, здійснювати надійний гарантований контроль і оптимізацію керування кожного як мідного, так і оптичного порту кабельної системи.

Складові R&MinteliPhy AIM системи:

– R&MinteliPhy Monitor.

– R&MinteliPhy Manage.

R&MinteliPhy Monitor складається з невеликого числа взаємозамінних компонентів.

RFID замінні в польових умовах мітки для стандартних комутаційних шнурів. Відображають інформацію про кабель, з'єднання.

Підтримувані типи інтерфейсів:

– RJ45 (кат 5e, кат 6, кат 6A).

– LC duplex/E2000 simplex/SC simplex/MPO.

Сенсорні рейки для комутаційних панелей

Зчитують інформацію, що втримується в RFID мітках комутаційних шнурів. Записують з'єднання, тип з'єднання. LED індикація на кожному порту для можливості простої комутації, ідентифікації й керування з'єднань. Установлюються на всі стандартні HD комутаційні панелі R&M.

Аналізатор для шафи

Управляє декількома шафами, зчитуючи й поєднуючи дані від декількох сенсорів. Взаємодіє з AIM сервером. Передбачені 19" 1U і 0U варіанти рішення.

AIM сервер

Клієнтоорієнтований сервер із централізованою конфігуруємою у реальному часі Базою Даних. Доступний, як автономний, так і хмарний сервер SaaS (Software as a Service) із численними автоматизованими можливостями, функціями, інструментами по плануванню із впровадженими бібліотеками для керування кабельною системою. Має широкі асортименти автоматизованих інструментів і інтерфейсів для керування й взаємодії з іншими системами.

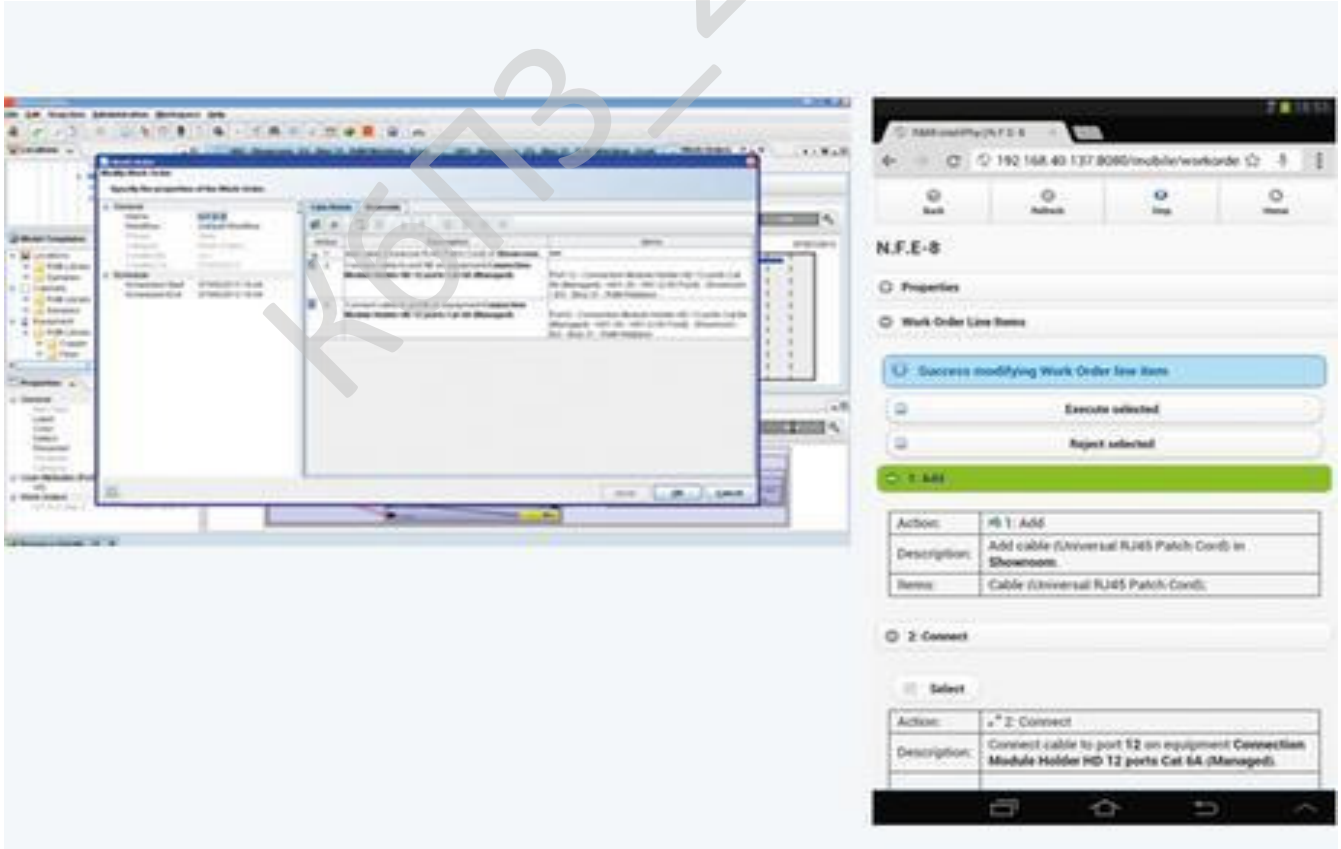

Рисунок 2.1 – Інтерфейс користувача AIM сервер

AIM Клієнт

Графічний користувальницький інтерфейс (GUI) з можливістю віддаленого доступу, що забезпечує адміністрування у вигляді ієрархічного дерева, географічної мережної карти. Доступ до всіх функцій: від автоматичної маршрутизації до виконання завдань по «Русі/Додаванню/Зміні» з візуалізацією станів.

Незалежний від операційної системи.

Масштабована, модульна архітектура системи R&MinteliPhy забезпечує можливість автоматизації й керування необхідної замовникові частини пасивною інфраструктурою з будь-якої крапки тоді, коли це необхідно й здійснювати віддалене ефективне керування якістю й ризиками.

R&M FREENET

Пропоновані в даний момент рішення компанією Райхле й Де-Массари в рамках системи R&M freenet дозволяють будувати кабельні системи класу D, E і F с інтеграцією технологій передачі мови й даних які відповідають всім існуючим вимогам і стандартам: національним, європейським і міжнародним. Система містить у собі компоненти категорій 5, 6 і 7 для передачі інформації з волоконо-оптичних або мідних каналів зв'язку. Модульність системи дозволяє забезпечити її нарощування. Пропоноване колірне маркування дозволяє легко ідентифікувати подавані на робоче місце сервіси. Високий вихідний контроль на всіх етапах виробництва гарантує найвищу надійність всієї системи в цілому.

R&M freenet забезпечує великий резерв по продуктивності, що гарантує роботу нових додатків у майбутньому. Спеціальні способи механічного захисту утрудняють або ж виключають можливість, як несанкціонованого доступу, так і неправильного включення або перекомутації з'єднань. Волоконо-оптичні коннектори SC, LC, SC-RJ і E2000 дозволяють забезпечити високий ступінь інтеграції оптичних портів у СКС і повну взаємозамінність із загальновідомим стандартним портом RJ45.

На ринку України R&M, базуючись на корпоративних угодах з фірмою THORSMAN, пропонує комплексні рішення, що включають у себе прикінцеве термінальне встаткування, різні стійки й люки, електромонтажні короби з алюмінію, сталі й пластика із широкою гамою декоративної обробки.

Разом з повним спектром устаткування R&M пропонує повний пакет кваліфікованого сервісу, що також містить у собі багаторівневу програму гарантій: 5 років на компоненти, 20 років на працездатність системи й гарантія на весь строк життя для всіх існуючих додатків.

Кроссове встаткування для мереж загального користування

Сімейство кросів для мідного кабелю фірми R&M є комплексним рішенням для кабельної частини телекомунікаційної мережі, як аналогової, так і цифровий. Завдяки використанню передових технологій з'єднання (IDC) і широкого набору сполучних елементів забезпечується побудова надійної кабельної інфраструктури. Філософія системи базується на широко відомій технології твердого урізного контакту. Ми розрізняємо наступні компоненти системи:

– Окремо варті вертикально-горизонтальні кроси.

– Окремо варті вертикально-вертикальні кроси.

– Настінні й пристінні кроси різних конфігурацій.

– Розподільні шафи (настінні, пристінні й ті, які знаходяться окремо).

– Розподільні коробки (металеві й пластикові).

– 19" кошика для інтеграції рішень.

Характерними рисами всіх кросів R&M є:

– Твердий урізний контакт.

– Унікальні технології урізних контактів.

– Діаметронезалежний контакт.

– Можливість використання багатожильного кабелю.

– Паралельне підключення ліній.

– Відсутність пайки.

– Відповідність Cat. 5e.

– Ідеальний для xDSL- додатків, швидкість потоку до 8 Mbit/s.

– Простота в обслуговуванні й експлуатації.

– Компактність установки.

– Економія часу й місця.

– Захист по струму й напрузі.

– Робота із кросом без відключення захисту.

– Можливість вимірів без обриву зв'язку.

– Безліч рішень маркування.

– Універсальність компонентів.

– Широкий набір аксесуарів.

R&M робить три типи кросів для мідних кабелів, що дозволяє замовникам знайти найбільш підходяще рішення:

VS Compact

Систему відрізняє мінімальні вимоги до простору, необхідному для монтажу й оптимізована технологія закладення кабелю. Крім цього характерною рисою є більше широкий діапазон ємності плінтів – 5, 8, 10, 16, 20 і 25 пара.

VS Standard

Серед відмінностей даної технології широкий вибір настінних і окремих конструктивів. Застосовуються 10 і 8 парні модулі урізними ножами VS Standard.

VS Modular

Модульний дизайн системи дозволяє індивідуальну адаптацію до всіх конструктивних і кабельних структур. Найбільш помітною відмінністю даної системи є максимальна простота в монтажі й експлуатації.

Волоконо-оптичні рішення. Кабелі й кроссове встаткування

Лінійка продукції для волоконо-оптичних каналів зв'язку дозволяє реалізовувати пасивні розподільні елементи телекомунікаційних мереж будь-якого розміру й призначення. Це досягається завдяки наявності різноманітних асортиментів компонентів і аксесуарів. Модульність і взаємосумістність дозволяє

будувати рішення, як для операторів магістральних мереж, так і локальні мережі для установ.

Волоконо-оптичні кабелі

Самий широкий вибір У кабелів на складі в Україні для побудови магістральних каналів зв'язку й структурованих кабельних мереж у будинку. Відповідність всім національним вимогам і приписанням. Можливість розробки й поставки самонесучих ADSS, грозозахисних OPGW, морських і інших спеціальних У кабелів.

Волоконо-оптичні муфти

У муфти R&M були спроектовані з метою забезпечення універсальності застосування. Залежно від розмірів, вони містять до 144 У з'єднань. Можуть розміщатися як під землею так і на відкритому повітрі. Механічний спосіб герметизації корпуса муфт полегшує їхню експлуатацію.

ВО комутаційні панелі

Різноманіття варіантів 19" ВО комутаційних панелей дозволяє оптимально підібрати необхідне рішення для вартих завдань. R&M пропонує економічні панелі SpliceBox, зручні панелі для операторів зв'язку UniRack, а також самі популярні В панелі Sub-Rack для ВО модулів Fiberliner.

Настінні волоконо-оптичні розподільники

R&M пропонує широкий спектр настінних У розподільників. Металевий і пластиковий, більший і малі, ступінь захисту до IP54. Продуманість конструкції, можливість розмежування роботи монтажних і експлуатаційних груп – полегшує роботу й забезпечує надійність функціонування об'єкта.

ВО розподільник R&M Foccos

Відмінна риса R&M Foccos – модульність, що дозволяє будувати саме той конструктив, що необхідний. Висота кросу від 10U до 46U, глибина 300 і 600 мм. Наповнення кросу може виробляється як стандартними В комутаційними панелями UniRack, так і спеціальними тримачами кабелю.

ВО розподільник FOMsystem

Рішення FOMsystem використовується там, де мають місце підвищені вимоги до надійності з'єднань і зручності експлуатації. Систему відрізняють оптимальне укладання кабелів і спеціальні касети індивідуального зберігання ділянок зварювання волокна. Поряд з іншими особливостями це дозволяє досягти операторського класу надійності – 99.99%.

Зовнішні розподільники R&M FiberCurb

Одним з можливих варіантів застосувань R&M FiberCurb є побудова PON (Passive Optical Network) або ONU (Optical Network Unit). R&M FiberCurb II.

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.

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 автоматизованого керування кабельною інфраструктурою АІМ.

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

Розгортання комплекту програмного забезпечення для керування інфраструктурою ЦОД у тої або іншій серверній фермі дозволяє скоротити експлуатаційні витрати на 20-30 відсотків. Крім того, ефективне керування тією же мережною інфраструктурою підвищує безпека даних, втрата яких може нанести більша втрата, чим відхід дата-центра в офлайн: такий інцидент може значний погіршити репутацію власника ЦОД серед поточних і потенційних партнерів.

Основними проблемами, з якими зіштовхуються сучасні ІТ, є збільшення обсягів використовуваної й збереженої інформації й ріст швидкості передачі даних. До того ж користувачів не цікавить сам процес обробки інформації, вони просто бажають якнайшвидшого одержання результатів. Яскравим прикладом реалізації такого підходу при вищевказаних тенденціях можна вважати використання хмарних обчислень. Однак ІТ-хмари, на відміну від однойменного природного явища, завжди мають конкретне місце прописки, оскільки реалізуються на базі центрів обробки даних (ЦОД), і доступ до них здійснюється в тому числі й по корпоративній ЛОМ. У справжній статті хотілося б звернути увагу на один з менш помітних, але від цього не менш важливих сегментів хмари – інфраструктуру передачі даних

У порівнянні з активним устаткуванням, що у загальному міняється кожні 3-5 років, структурована кабельна система (СКС) припускає експлуатацію протягом довгого часу. Про це свідчить і строк гарантійних зобов'язань виробників СКС: 20-25 років. Але наскільки створені 5-10 років тому мережі відповідають сьогоднішнім вимогам і чи будуть вони відповідати їм ще через 10 років? Спробуємо відповісти на це питання, розглянувши використовувані й

проектовані протоколи передачі даних, а також продукти, що існують і розробляються під ці протоколи виробниками СКС.

Традиційно СКС виконуються з використанням мідних або оптичних волокон. Мідь застосовується в основному для доставки даних безпосередньо до робочого місця, а оптичні лінії завдяки кращій якості передачі інформації вибираються для з'єднання сегментів ЛОМ і високошвидкісних додатків. У такому порядку їх і розглянемо.

Історично для передачі даних у ЛОМ служить протокол Base-T, зокрема, найпоширеніші його 100- і 1000-мегабитні версії. Саме для підтримки цих протоколів і створені найпоширеніші категорії СКС: 5 й 6 (по TIA/ EIA-942-A, або клас D і E по ISO/Genelec; надалі будемо використовувати позначення TIA/EIA), на основі яких побудоване переважна більшість ЛОМ за минулі 15 років. Однак в останні кілька років наявних резервів стало не вистачати. І кабельні, і бездротові технології переступили рубіж 1 Гбіт/с і вийшли з лабораторій виробників на широкий ринок. Практично всі виробники СКС зараз пропонують кабелі категорій 6a, 7 і 7a, що забезпечують передачу даних на швидкостях до 10 Гбіт/с. З них найбільше поширення одержує категорія 6a, що забезпечує дану швидкість поряд з адекватною вартістю й технологічністю, на відміну від інших, у яких використовується не тільки екранування провідників, але й нестандартні рознімання типу TERA.

Розглянемо основні фактори, що впливають на зростаючі вимоги до інфраструктури.

Wi-Fi

Прийнятий в 2014 р. стандарт технології Wi-Fi 802.11ac дає можливість організувати обмін даними з бездротовими пристроями на швидкостях, що перевищують 1G. І якщо роботу в діапазоні швидкостей до 2G можливо забезпечити за допомогою Link Aggregation (так, крапки доступу 802.11ac "першої хвилі" мають 2 гігабітних uplink-a), то для "другої хвилі" цього вже недостатньо. А попереду нас чекають ще більш високошвидкісні протоколи, що

перебувають у розробці, наприклад 802.11ax. Таким чином, бездротові технології сприяють розширенню застосування в мережах високошвидкісних кабелів категорії 6а.

PoE

Ще одним приводом для використання категорії 6а є планований до прийняття в 2017 р. новий стандарт PoE – IEEE 802.3 bt. Він висуває підвищені вимоги до ЛОМ на основі кручений пари. Зокрема, значно, практично в 2 рази (до 90 Вт), піднімається потужність, що доставляється до прикінцевого пристрою. А температурний режим, що передбачається цим стандартом, перевершує можливості кабелів категорії 5е.

Продовження життя категорії 5е і 6

Однак з багатьох причин не всі користувачі можуть дозволити собі перехід на високошвидкісну офісну ЛОМ. Для їхньої підтримки 28 жовтня 2014 р. компанії Cisco, Freescale, Xilinx і Aquantia сформували альянс NBASE-T з метою розробити стандарти Ethernet 2.5G і 5G для підтримки 802.11ac "другої хвилі", а також інших додатків в інсталяціях категорії 5е і 6. У такий спосіб вийде адаптувати використання існуючої провідної інфраструктури під нові бездротові технології введенням двох "проміжних" стандартів передачі даних – 2,5 і 5 Гбіт/с.

По задуму учасників альянсу ці два протоколи будуть працювати на всіх стандартних інстальованих системах на базі кабелю категорії 5е до 100 м, хоча подібні очікування здаються малоймовірними. На гігабітних протоколах категорія 5е працює на межі своїх можливостей, і її подальший розвиток на 2,5 і 5 Гбіт/с викликає питання.

ЦОД

Для нових офісів або реконструкцій у даний момент як економічно, так і технологічно виправданим є вибір СКС не менш категорії 6, а краще 6а. Перехід на більше високошвидкісні категорії кабелів не тільки диктується збереженням інвестицій у ЛОМ, але й підтримується з боку виробників активного встаткування. За інформацією компанії Dell'Oro Group, що посилається на дані по

відвантаженнях замовникам в 2014 р., кількість 10 G-Портів активного встаткування для ЦОД перевищило кількість 1 G-Портів за той же період.

Для інфраструктури ж ЦОД категорія 6а є обов'язковою у відповідності зі стандартами TIA/ EIA-942-A, EN 50173-5 і ISO 24764. Однак і 10 Гбіт/с може виявитися недостатньо – очікуваний ріст потреби в пропускній здатності ЦОД і мереж зберігання, а також недостатня економічна ефективність попередніх інфраструктурних рішень стимулюють розробку специфікації кабельних систем нового покоління.

– Була: категорія 5e/6; 100/1 G-Base.

– Зараз: категорія 5e/6/6a; 100/1G/10 G-Base.

– Буде: категорія 5e/6/6a/8;100/1/2,5/5/10/25/40 G-Base.

На зміну стандарту CR4 розробляється стандарт 40GBase. Він покликаний, з одного боку, допомогти уникнути проблем недоліку пропускної здатності, а з іншого боку – забезпечити більш економічно вигідне рішення для топологій End-of-Row (EOR) і Middle-of-Row (MOR).

В оптичній частині основні нововведення зв'язані по більшій частині з інфраструктурою дата-центрів, оскільки існуючі категорії оптичних провідників – OM4 для багатомода й OS2 для одномода – цілком задовольняють запитам офісних ЛОМ.

У дата-центрах вимоги до швидкості передачі даних помітно зростають. Щоб їм відповідати, в 2010 р. був прийнятий стандарт IEEE802.3ba, що описує роботу зі швидкістю 40 і 100 Гбіт/сі шляхом паралельної передачі сигналів по декількох волокнах. Одним з нововведень у даному стандарті стала можливість застосування багатоволоконого оптичного коннектора MPO, що не тільки забезпечує високу щільність оптоволоконих ліній, але й дає можливість легкої модернізації інфраструктури для підтримки високих швидкостей.

Як і для офісних додатків, для дата-центрів розглядається можливість введення проміжного протоколу 25 Гбіт/с. Очікується, що 25 GBASE-T поряд з попередніми стандартами дасть можливість для економічного з'єднання

активного встаткування на швидкості понад 10 Гбіт/с по одному оптичному волокну. Таке рішення буде вигідно для топологій End-of-Row (EOR) і Middle-of-Row (MOR).

У жовтні 2014 р. компанія CommScope разом з декількома виробниками волокна, приймачепередатчиків і постачальниками системних рішень у рамках TIA ініціювала проект по розробці стандарту на волокно, що забезпечувало б смугу пропущення OM4 до 950 нм, тим самим дозволяючи домогтися швидкості в 100 Гбіт/с по одному оптичному волокну. Оскільки нове широкополосне волокно збереже продуктивність OM4 на довжині хвилі 850 нм, те воно буде сумісно з усіма існуючими додатками, при цьому забезпечить передачу даних у режимі WDM з використанням недорогих лазерів VCSEL.

Одночасно із цим збільшується швидкість передачі даних у середовищі Fiber Channel, призначеної для роботи із системами зберігання даних, і розробляються нові протоколи для Fiber Channel over Ethernet.

Саме в оптичній підсистемі зосереджують основні зусилля по досягненню високих швидкостей передачі даних: розробляються не тільки нові протоколи, але й нові типи провідників і рішень. Так, все більшу популярність набирають претерміновані оптичні рішення з використанням 12-волоконих коннекторов MPO, які не тільки дозволяють ущільнити інфраструктуру й прискорити її розгортання, але й забезпечують універсальність застосування в різних додатках, а також збереження інвестицій при модернізації систем

При побудові сучасної інфраструктури передачі даних бажано враховувати й інші тенденції.

Підхід до облаштованості робочого місця останнім часом перетерпів більші зміни, пов'язані зі змінами в поводженні співробітників і способами використання робочих місць. Ці зміни вимагають більшої гнучкості при проектуванні робочих місць, спільно використовуваних площ, переговірних кімнат і інших приміщень. Основними причинами цих змін є все більше поширення як провідних, так і бездротових технологій. Ефективність рішень

інтелектуального будинку усе більше залежить від інтеграції систем IT і систем автоматизації будинку. Зміни в організації робочого місця спричиняють зміни в організації телекомунікацій: якщо раніше архітектура будувалася навколо підключення робочої станції, то зараз потрібно брати до уваги можливість включення в інфраструктуру й інші пристрої.

Інтелектуальна інфраструктура

Інфраструктура офісів і дата-центрів значно ускладнюється з ростом швидкості обміну й обробки інформації, і для більше ефективного контролю над інфраструктурою рекомендується постачити її часткою інтелекту. Практичні всі провідні виробники СКС пропонують такого роду рішення, і зараз розробляються відповідні стандарти автоматизованого керування інфраструктурою (Automated Infrastructure Management) TIA 606-B, ISO/IEC 18598 і ISO/IEC 14763-2.

Подібного роду системи є єдиним джерелом знань про стан фізичної інфраструктури, даючи можливість:

– одержувати інформацію про підключені пристрої в реальному часі;

– здійснювати контроль за ресурсами – невикористовувані порти, планування робіт, акуратність і звітність;

– робити інтеграції з іншими додатками, наприклад DCIM.

Отже, що ж варто врахувати при плануванні інфраструктури передачі даних?

Необхідність у збільшенні швидкості передачі даних вирішується за рахунок розробки нових протоколів. Однак середовища передачі даних для реалізації поточних і розроблювальних високошвидкісних протоколів доступні вже зараз. Існуючі інсталяції можуть не забезпечувати повноцінну роботу активних мережних пристроїв нового покоління. Щоб не виявитися на узбіччі прогресу, варто готовити свою інфраструктуру для роботи із цими протоколами.

Категорія 5e не може розглядатися як основне середовище передачі даних для нових офісів і ЦОД, оскільки не відповідає вимогам сучасних і майбутніх

високошвидкісних додатків. Для нових інсталяцій варто віддавати перевагу мінімум категорії 6, а для дата-центрів – категорії 6а.

Використання претермінованих рішень (дозволяють уже зараз впровадити 10, 40 і 100 Гбіт/с по оптичних лініях) дає можливість не тільки підвищити швидкість передачі даних, але й легко робити реконфігурацію мережі й мігрувати на нові протоколи. Інтелектуальні системи керування СКС полегшують контроль і експлуатацію системи при росту кількості з'єднань.

Хоча рішення з функціональністю AIM доступні вже багато років, є п'ять причин, по яких стандарт приймають саме зараз.

1. Збільшення складності мереж. Розвиток таких тенденцій в IT, як мобільні обчислення, серверна віртуалізація й публічні хмари, робить середовище ЦОД усе більше динамічним. У той же час постійно зростаюча щільність устаткування й з'єднань у центрах обробки даних ще більше утрудняє реєстрацію мережних з'єднань і топологій.

Комплексне автоматизоване документування всіх компонентів і з'єднань, включаючи канали Fibre Channel, комутацію в мережі зберігання даних (SAN) і з'єднання з віддаленими вузлами, сприяє більше якісному керуванню складними мережами.

2. Зростаючі працезатрати. У багатьох випадках керування фізичною інфраструктурою здійснюється підручними засобами – наприклад, за допомогою таблиць Excel і графіки Visio. Дорогоцінний час витрачається на спроби документування високодинамічних мережних топологій, а тим часом діагностика й усунення проблем у мережі стають усе більше складними завданнями, що подовжує строки їхнього виконання.

Крім того, без автоматизованих рішень додавання або переміщення компонентів мережі вимагає значних зусиль і спричиняє усе більше помилок. При наявності AIM вся мережна інфраструктура представлена в систематизованій базі даних, що подає точну інформацію про поточний стан мережі й майбутніх вимог у реальному часі.

Системи AIM забезпечують автоматизоване керування порядком виконання робіт, а механізми індикації й мобільних додатків дозволяють технічному персоналу швидше виконувати поточні завдання й вносити зміни.

3. Неефективне використання ресурсів. Без інвентаризації компонентів мережі в реальному часі виникає ризик багаторазового призначення того самого порту або ситуації, коли організація продовжує платити за невикористовувані пристрої й з'єднання, які не виведені з експлуатації належним чином.

Завдяки інтелектуальній системі керування при виводі з експлуатації серверів або комутаторів можна передбачити процеси звільнення проміжних з'єднань і відповідних портів для майбутнього використання.

AIM дозволяє компаніям домогтися підвищення економічної ефективності в масштабі ЦОД протягом усього його життєвого циклу – від проектування, планування інсталяції й розгортання до повсякденного керування змінами (MAC) і модернізації.

4. Дорогі простої. По даним Ponemon Institute, середня вартість простою становить близько 5600 доларів у мінуту, або 336 000 доларів у годину, а серед причин простоїв на першому місці як і раніше людський фактор. AIM здійснює моніторинг компонентів мережі, на ранній стадії попереджає про проблеми й перевантаження, які можуть в остаточному підсумку привести до простою або зниження продуктивності. У результаті підприємство одержує добре керовану, більше стабільну й надійну високопродуктивну мережу.

5. Скомпрометована безпека. Мережі можуть бути уразливими на багатьох рівнях – від атак шкідливого програмного коду до крадіжок робітників даних і несанкціонованого доступу. Усе більше виробничих операцій безпосередньо пов'язане з використанням інформаційних мереж, а зі збільшенням кількості підключених до них пристроїв росте число можливих «крапок входу».

У відповідальних мережних комунікаціях AIM додає додатковий рівень безпеки, запобігаючи несанкціонованому підключенню, відключення, а також переміщення й зміни в мережі.

AIM забезпечує повний контроль у реальному часі за всіма компонентами мережі і їхніх підключень, завдяки чому компанії одержують більше надійну мережну інфраструктуру – з високою доступністю сервісів і досить продуктивну. Завдяки автоматичному документуванню всіх мережних компонентів і їхніх з'єднань підвищується керованість мережі, вона стає по-справжньому інтелектуальною.

За інформацією Gartner, інтелектуальна мережна інфраструктура, розгорнута в ЦОД, допомагає знизити операційні витрати на 20-30%, скоротити простої, оптимізувати енергоспоживання й займані площі, прискорити розгортання сервісів і поліпшити захист.

3.2 Розробка структурної схеми

Моніторинг журналу Security Log

Адміністраторам, які мають потребу в інструменті для моніторингу журналів подій Windows і розсилання оповіщень (по електронній пошті або на пейджер) при виявленні подій, що відповідають заданим критеріям, я рекомендую створити спеціалізований механізм попереджень із використанням VBScript і Windows Management Instrumentation (WMI). Раніше в журналі ми вже приводили приклади сценаріїв для моніторингу певних подій і пересилання поштового повідомлення на зазначену адресу при кожній такій події. Сценарій не витрачає ресурси центрального процесора під час очікування події, і адміністратори може легко змінити WMI-запит, що описує події, оброблювані сценарієм. В WMI-запитах використовується команда SQL Select, аналогічна командам, з яких побудовані запити Microsoft Access. Проблема полягає у відборі повідомлень, які повинні генерувати попередження. Якщо не дотримувати обережності, пейджер або вхідна поштова скринька будуть переповнені не занадто важливими повідомленнями.

Приступаючи до підготовки критерію для відправлення попереджень, необхідно використовувати інструмент для збору інформації з журналів подій. Кращий інструмент для опитування журналів подій – Log Parser компанії Microsoft. Log Parser дозволяє направляти запити до журналу подій з використанням оператора Select, схожого на запит WMI, але більш зручного для опитування існуючих журналів. Задача адміністратора – скласти один або кілька запитів для журналу подій кожного типу (наприклад, Application, Security, System), щоб відфільтрувати незначні події, але повідомляти про надзвичайні події. Я рекомендую витягати з журналу Security найбільш типові й важливі події. Зразковий вид запиту:

```
logparser "select
        TimeGenerated,EventID,Message
from \\mtg1\security
where EventID in
        (675;676;681;642;624;644;617;632;660;636)"
```

Моніторинг інших журналів

Для інших журналів подій (наприклад, Application, System) я рекомендую складати звіт, починаючи з пошуку попереджень і помилок, ігноруючи інформаційні повідомлення. І навіть після цього залишається багато незначних попереджень і повідомлень про помилки. Наприклад, я одержую багато помилок із джерела MRxSmb, який можна сміло ігнорувати. Тому наступний крок – ідентифікувати події, які являють собою «шум», і виключити їх за допомогою пропозиції Where у запиті Log Parser. У наступну команду додане вираження, що виключає подію ID 3019 для джерела MRxSmb і інформаційні повідомлення з будь-якого іншого джерела (EventType < 4):

```
logparser "select
        TimeGenerated,EventID,Message
from system where
        EventID<>3019 and SourceName
        <>'MRxSmb' and EventType < 4"
```

Варто пам'ятати, що журнали подій досягають дуже великих розмірів. Крім того, EVT-файли не мають індексації, що забезпечує швидкий пошук

інформації без переглядів усього журналу. На підготовку звіту Log Parser може піти чимало часу, так як програмі щораз доводиться переглядати весь журнал подій. На щастя, завдяки контрольним крапкам Log Parser пам'ятає, де закінчився минулий перегляд.

Задача моніторингу журналів подій Windows ускладнюється через те, що в кожного комп'ютера є власний набір журналів і не існує природного способу зібрати їх воєдино. Якщо немає інструмента, що поєднував би журнали в одній центральній базі даних для розсилання попереджень і моніторингу, існує два варіанти рішення проблеми. Якщо має бути управляти лише нечисленними серверами й пристроями, то можна просто призначити попередження й звіти для кожної машини. Більше централізований підхід – використовувати можливість Log Parser опитувати кілька комп'ютерів. Необхідно лише перелічити журнали подій кожного комп'ютера в пропозиції From. Наприклад, що випливає команда опитує журнал System на машинах server1, server2 і server3.

```
logparser "select
        TimeGenerated,EventID,Message
from \\server1\system,
        \\server2\system,
        \\server3\system"
```

Після того як будуть складені один або кілька звітів для журналу кожного типу, можна готовити їх щодня або щотижня за розкладом. Досить перенаправляти вихід Log Parser у текстовий файл, додавши до наведеного вище команду: «> C:\path\file.txt» а потім регулярно переглядати цей файл. Ще краще пересилати отриманий файл по електронній пошті адміністраторові. Для цього варто додати один рядок у командний файл, для виклику Blat. За допомогою загальнодоступної утиліти Blat зручно пересилати файли по електронній пошті через SMTP.

Настроївши звіти, можна приступати до проектування критеріїв попередження. Було б дуже добре, якби всі постачальники додатків і пристроїв документували події й повідомлення, генеруємі їхніми продуктами. У такому випадку адміністратори могли б зробити усвідомлений вибір, але в житті нічого

не дається легко, тому доводиться діяти «на дотик». Вибір критерію нагадує процес складання звітів, описаний вище. Однак варто бути більше розбірливим при виборі попереджень, що посилаються на пейджер або у свою поштову скриньку. У більшості програм попередження можна будувати як фільтри Exclude, Include, а потім застосовувати їх послідовно, видаляючи дрібні події. Якщо використовується WMI, то можна задіяти речення Where для фільтрації більшості непотрібних подій. Якщо можливостей речення Where виявляється недостатньо, можна побудувати додатковий фільтр із застосуванням сценарію VBScript.

Я рекомендую використовувати Log Parser для підготовки критеріїв попередження, замість того щоб призначати попередження, а потім набудовувати їх доти, поки на пейджер не буде надходити тільки потрібна інформація. Використовуючи інформацію про події, уже зібрану в журналах, варто почати з тих же критеріїв, які застосовуються для звітів, і уточнювати їх, відкидаючи менш значимі події, що не заслуговують попередження. Якщо зібрані дані приблизно за місяць, то такий підхід, по суті, дозволяє моделювати обробку попереджень за тривалий період часу, щоб оцінити кількість вступників попереджень. Після того як буде знайдений прийнятний критерій у формі звітів Log Parser (або іншого інструмента підготовки звітів), можна перетворити критерій у запити WMI усередині сценаріїв VBScript. Щоб сценарії попереджень були постійно активні, можна настроїти їх на запуск у процесі початкового завантаження комп'ютера. Сценарії початкового завантаження можна розмістити в будь-якому об'єкті групової політики Group Policy Object (GPO) у розділі настроювання політик Computer Configuration\WindowsSettings\Startup and Shutdown Scripts. Сценарії, активізуємі при початковому запуску, працюють у контексті локального облікового запису System, тому в них є всі необхідні повноваження.

Ключ до підготовки ефективних звітів і критеріїв попереджень – відтинати непотрібну інформацію, який не повинне бути у звітах або попередженнях, а не

вивуджувати корисні дані. Через численність джерел подій і поганої документованості більшості додатків і мережних пристроїв не існує способу перелічити всі важливі події й повідомлення. Якщо обмежити критерій пошуком відомих найважливіших подій, то виникає реальна погроза, що в результаті неполадки пристрою або комп'ютера не буде отримане повідомлення про подію, не враховану в критеріях. Для балансу варто підготувати кілька попереджень або звітів, орієнтованих на конкретні критерії. Корисно підготувати такі попередження й звіти для вже знайомих проблем. Але в цілому при проектуванні критеріїв рекомендується додержуватися принципу виключення, а не включення.

Моніторинг текстових журналів

Більшість серверних продуктів Microsoft і компонентів Windows протоколюють будь-які важливі події в журналах System або Application, але кожна служба або основний компонент операційної системи (наприклад, IIS, DHCP, SMTP, Internet Authentication Service, IAS) записують більш докладну інформацію у власний текстовий журнал у спеціальному форматі. Для моніторингу інформації, наявної тільки в цих текстових файлах, зручно використовувати Log Parser. Log Parser розпізнає будь-який формат текстового файлу з розмежувачами, наприклад із символами табуляції або комами (CSV), і дозволяє задіяти ту ж команду SQL Select для опитування текстових файлів журналів.

Таким чином, за допомогою Log Parser можна вирішити задачу підготовки звітів на основі текстових журналів. Але що робити із попередженнями про критичні події, що поступають у реальному часі, інформація про які зберігається в текстових файлах? Я рекомендую скористатися інструментом tail, запозиченим з UNIX. Tail відслідковує додавання нових рядків у зазначені користувачем текстові файли. Як тільки виявляються нові дані, tail посилає їх у стандартний вихідний потік (stdout). Вихідні дані tail можна направити в сценарій, що аналізує нові записи в міру їхнього протоколювання й при необхідності генерує попередження. Наприклад:

```
tail /f logfile.txt |LoopOnNewMessages.cmd
```

виявляє нові повідомлення, додані у файл logfile.txt, і направляє їх в LoopOnNewMessages.cmd. Loop OnNewMessages.cmd передає кожне повідомлення за адресою rsmith@ultimatewindowssecurity.com, але замість нього можна вказати будь-яку іншу поштову адресу. Щоб не пересилати не занадто істотні повідомлення, можна доповнити сценарій фільтруючою логікою.

Tail

Витягти інформацію з журнальних файлів або інших текстових файлів можна за допомогою широко розповсюдженої програми grep. Але є й інший корисний інструмент із миру UNIX, гідний зайняти місце в інструментальному наборі адміністратора Windows, – програма tail. По суті, tail показує останні кілька рядків текстового файлу – це особливо корисно при аналізі файлів журналів. Наприклад, якщо адміністратор становить нові правила для брандмауера, tail покаже, як правила відбиваються на файлі журналу.

Автономне використання Tail

При використанні поза комбінацією з іншими програмами, tail показує кілька останніх рядків текстового файлу. За допомогою декількох параметрів можна змінити подання інформації на екрані. Особливо корисний параметр follow (-f), що дозволяє безупинно відслідковувати й виводити на екран зміни в текстовому файлі. Наприклад, команда

```
tail -f ex050410.log
```

показує останні 10 рядків журнального файлу з ім'ям ex050410.log і буде відслідковувати й відображати нові записи в міру їхньої появи. Якщо файл являє собою журнал Web-служби Microsoft IIS і хто-небудь звертається до Web-вузла, IIS зробить у журналі новий запис. Нові додавання негайно відображаються на консолі, у якій працює tail. Цей параметр спрощує діагностику, дозволяючи негайно побачити нові записи.

Спільне застосування Tail і Grep

Як відомо, grep – програма, що веде пошук зазначених послідовностей символів у цільовому текстовому файлі. Наприклад, при діагностиці комп'ютера,

що працює з Windows Firewall, потрібно відшукати в журналі брандмауера дії, зроблені в певний день. Журнал не розділений по датах і досить великий.

За допомогою grep можна витягти рядки даних від 7 березня 2009 року й записати їх у новий текстовий файл:

```
grep « 2009-03-07» p-firewall.log
> 030705 p-firewall.log
```

Як щодо tail? Команду можна використовувати для обробки журналів брандмауера в процесі діагностики або відстеження атак у реальному часі. Але можна застосувати tail разом з grep, щоб виводити на екран тільки певні дані.

Для початку варто настроїти брандмауер на запис журналів у текстовий файл. Всі системи UNIX використовують syslog для протоколювання подій; більшість комерційних брандмауерів також підтримують syslog. Якщо на системі UNIX використовуються grep і tail, то варто настроїти брандмауер на пересилання даних syslog у хост-машину syslog. Користувачі Windows можуть установити й працювати із сервером syslog на базі Windows. Я рекомендую Kiwi Syslog Daemon фірми Kiwi Enterprises, відмінний інструмент для збереження даних syslog у текстовому файлі.

Потім потрібно побудувати шаблон на основі синтаксису постачальника брандмауера. Наприклад, адміністратор використовує брандмауера Cisco PIX і хоче одержувати оповіщення щораз, коли хтось звертається до Web-служб через брандмауера. За допомогою tail і grep можна в реальному часі виявляти в журналах символи «/80» (представляють Web-трафік у журналі PIX), наприклад:

```
tail -f pix.log | grep «/80»
```

Більш вдалий підхід – використовувати метасимволи регулярних виражень, які забезпечують більше складну фільтрацію, чим звичайні текстові рядки:

```
tail -f pix.log | grep /80[[:space:]]
```

Освоєння регулярних виражень вимагає часу, але в нагороду ви одержуєте бібліотеку корисних і ефективних шаблонів, які можна використовувати для пошуку майже будь-яких даних, – безсумнівно, це виправдує витрачені зусилля.

Ускладнений Tail

Grep і tail – прості у використанні й дуже гнучкі програми. При роботі з консольними додатками обидва інструменти значно спрощують аналіз журнальних файлів і повсякденне адміністрування. Версія командного рядка tail – швидка й проста в експлуатації, і, імовірно, прихильники строгих правил нададуть їй перевагу завдяки простоті й можливості пересилати вихідні дані в інші програми, такі як grep. Але існують версії tail із графічним інтерфейсом Windows, причому деякі з них наділені більш складними функціями, наприклад кольоровим виділенням співпадаючих послідовностей. Таке форматування допомагає відзначати важливі файли.

Зразок безкоштовної графічної програми tail для Windows – BareTail компанії Bare Metal Software (її можна завантажити за адресою *http://www.baremetalsoft.com/baretail*). Як і tail, BareTail відображає текстовий файл і відслідковує доповнення до файлу, але оскільки BareTail працює із графічним інтерфейсом, вона має у своєму розпорядженні функції виділення.

Завдяки таким функціям простіше виявити певний текст (наприклад, конкретну IP-адресу або порт) «на ходу», спостерігаючи за журналом брандмауера. Можна також змінити шрифт, без праці скопіювати рядок тексту й відкрити недавно переглянуті файли журналів за допомогою списку недавно використаних файлів Windows.

Моніторинг SNMP і журналів Syslog

Контролювати телеметричні джерела SNMP і Syslog легко завдяки безкоштовній версії програми Kiwi Syslog Daemon компанії Kiwi Enterprises. Цей диспетчер серверів автоматизованого керування кабельною інфраструктурою AIM, Windows Syslog і SNMP дозволяє зібрати всі телеметричні дані про мережні пристрої в одній програмі. Із графічного інтерфейсу програми можна настроїти фільтри для збору повідомлень, що відповідають певним критеріям, а потім указати одне або кілька дій, що вживаються у відповідь на повідомлення. Можна побудувати фільтри для видалення непотрібних повідомлень і вказати, що

повідомлення, що залишилися, повинні генерувати попередження або зберігатися в базі даних для наступних звітів. За допомогою Kiwi Syslog Daemon можна фільтрувати повідомлення за часом дня, днем тижня, пристроям, рівню, IP-адресі звітного агента або рядкам у повідомленні. Крім того, інструмент може виконувати різноманітні дії – оповіщення по електронній пошті, збереження в базі даних по ODBC, запуск програми й інші – у відповідь на зазначені події.

Безкоштовна версія Kiwi Syslog Daemon інтерактивно працює в настільному комп'ютері, тому адміністратор повинен зареєструватися, щоб контролювати пристрої. Але розширена версія продукту функціонує як служба, а її вартість – усього 100 дол. для одного сервера. Якщо активізувати моніторинг пасток SNMP, необхідно також указати поля Facility і Level, які використовуються інструментом при перетворенні пастки в повідомлення Syslog. Наприклад, можна вказати пастки SNMP як Facility Local4 і Level 3 Error. Потім можна скласти правила розсилання попереджень спеціально для пасток SNMP шляхом фільтрації повідомлень Local4.

Отже, існують ресурси для моніторингу різноманітних джерел телеметричних даних. Перш ніж почати проектувати власне рішення для моніторингу, корисно познайомитися з інструментами, які є на ринку. Вони доступні й повнофункціональні. Однак не можна одержати повне рішення, просто здобуваючи інструмент. Потрібно визначити критерії для звітів і попереджень, щоб не одержувати занадто багато повідомлень про незначні події, але не слід упадати в іншу крайність і задавати настільки строгі критерії, що рішення моніторингу може перешкодити виконанню тої самої задачі, для якої воно призначалося. Для досягнення балансу варто становити критерії, відтинаючи незначні, а не вибираючи важливі події. Єдине виключення із цього правила – журнал Security, що набагато коротше, а крім того, краще документовано. Повна база даних подій журналу Security і їхніх значень опублікована в Security Log Encyclopedia на сайті Ultimate Windows Security (*www.ultimatewindowssecurity.com*).

Для підготовки ефективної й вичерпної процедури моніторингу потрібно прикласти певні зусилля, але вони не пропадуть даром. Немає нічого гірше, ніж довідатися про проблему від користувачів і після перегляду журналів виявити, що попередження надходили трьома днями раніше. Вимоги бізнесу й законодавства щодо інформаційної безпеки й звітності дуже високі. Порушення безпеки можливі, але адміністратор і його компанія набагато успішніше переборють труднощі, якщо добре підготуються до критичної ситуації. Ефективному моніторингу немає повноцінної заміни.

Структурна схема системи зображена на рисунку 3.1.

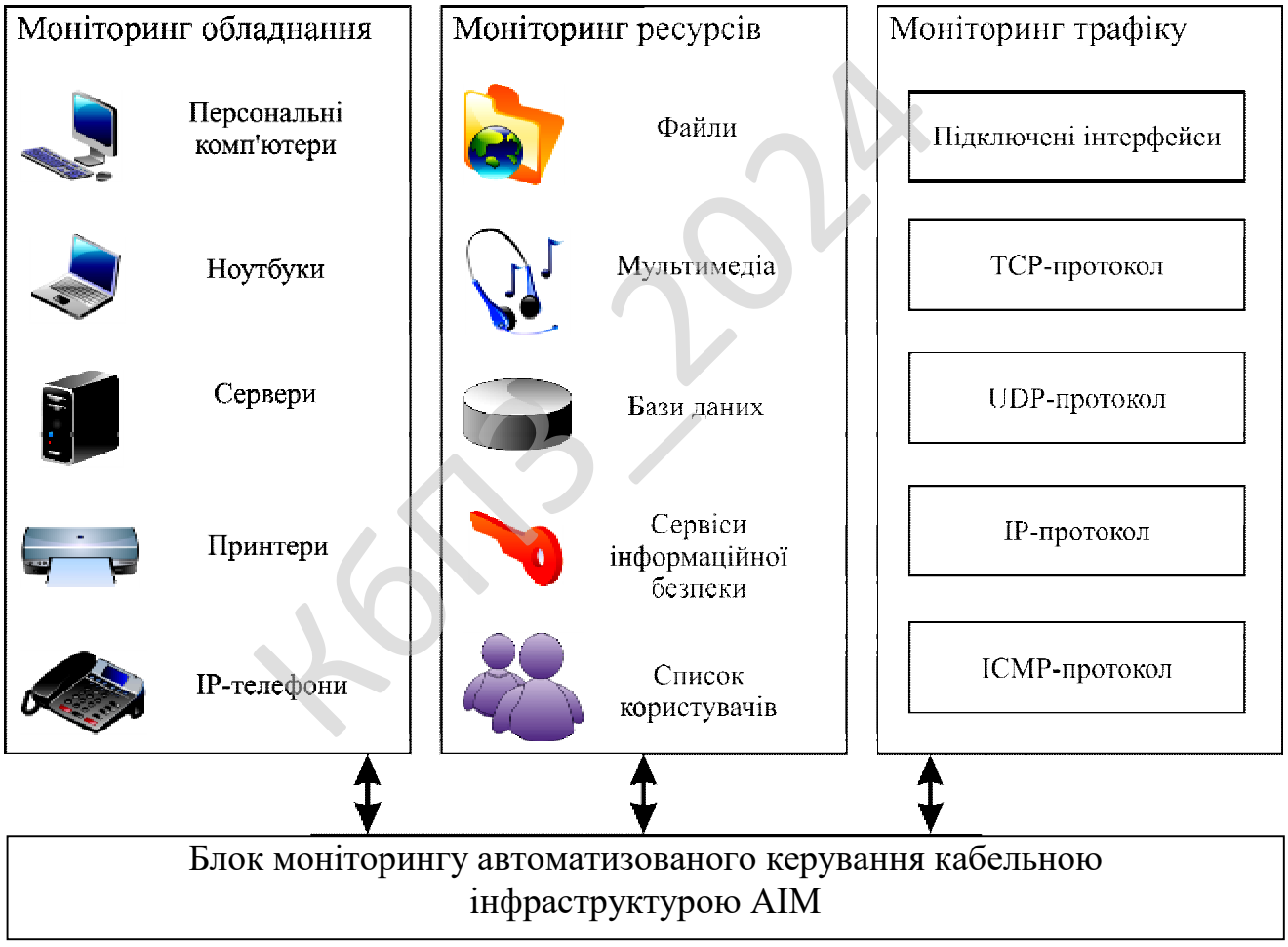

Рисунок 3.1 – Структурна схема системи

З рисунку видно, що моніторинг локальної мережі здійснюється по трьох напрямках:

– Моніторинг обладнання автоматизованого керування кабельною інфраструктурою AIM.

– Моніторинг ресурсів автоматизованого керування кабельною інфраструктурою AIM.

– Моніторинг трафіку автоматизованого керування кабельною інфраструктурою AIM.

Переваги створення ЦОД:

- підвищення ефективності й надійності експлуатації обчислювальних ресурсів;
- надання відказостійких інфраструктурних сервісів у режимі 24 години х 7 днів у тиждень х 365 днів у році;
- просте й прозоре централізоване адміністрування;
- зниження витрат на надання інженерних комунікацій;
- високий рівень захисту інформаційної системи;
- централізоване керування й облік ресурсів ЦОД;
- контроль доступу до ЦОД;
- просте й зручне масштабування обчислювальних ресурсів.

3.3 Розробка функціональної схеми

Функціональна схема системи зображена на рисунку 3.2. З рисунку видно, що розроблена система складається з наступних блоків:

– Моніторинг трафіку автоматизованого керування кабельною інфраструктурою AIM.

– Робота з файлами автоматизованого керування кабельною інфраструктурою AIM.

– Монітор з'єднань автоматизованого керування кабельною

інфраструктурою AIM.

– Статистика подій автоматизованого керування кабельною інфраструктурою AIM.

– Робота з ресурсами автоматизованого керування кабельною інфраструктурою AIM.

– Робота з сесіями автоматизованого керування кабельною інфраструктурою AIM.

– Функції для роботи з центрами обробки даних, побудованих з використанням автоматизованого керування кабельною інфраструктурою AIM.

Розглянемо детальніше кожний з блоків.

Робота з ресурсами автоматизованого керування кабельною інфраструктурою AIM включає в себе:

– Визначення доступних ресурсів.

– Закриття локального ресурсу.

– Відкриття локального ресурсу.

– Приховання й показ ресурсів.

Система відображає наявні у мережі ресурси у вигляді дерева. Пошук ресурсів можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо.

Можна додавати до загальних ресурсів мережі свої власні, а також закривати їх потім.

Робота з сесіями автоматизованого керування кабельною інфраструктурою AIM включає в себе:

– Одержання списку поточних сесій.

– Завершення сесій.

Програма дозволяє переглянути список відкритих сесій, що включає в себе: назву сесії, користувача, що її розпочав, номер сесії, час роботи та час очікування.

Рисунок 3.2 – Функціональна схема системи

Моніторинг трафіку автоматизованого керування кабельною інфраструктурою AIM включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку.

Розроблена система відображає всі інтерфейси приєднані до комп'ютера, на якому запущена програма, їх MAC-адреси та вхідний і вихідний трафік на кожному з них.

Робота з файлами автоматизованого керування кабельною інфраструктурою AIM включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

Можна переглянути, які з Ваших файлів, що Ви відкрили для загального доступу, переглядають по мережі. Програма відобразить список файлів та користувачів, які їх переглядають. Також можна відкрити чи закрити файл.

Монітор з'єднань автоматизованого керування кабельною інфраструктурою AIM включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

Система фіксує всі підключення по TCP- та UDP-протоколу, та виводить їх на екран у форматі *IP-адреса:порт_призначення*.

Статистика подій автоматизованого керування кабельною інфраструктурою AIM включає в себе відстеження подій в наступних протоколах:

– TCP-протокол.

– UDP-протокол.

– IP-протокол.

– ICMP-протокол.

Статистика ведеться по цілому ряду параметрів. Наприклад для TCP-протоколу фіксується: Тип алгоритму повторної передачі, мінімальний тайм-аут, максимальний тайм-аут, максимальна кількість помилок з'єднання, активні з'єднання, пасивні з'єднання, невдалі спроби відкриття, скидання встановлених з'єднань, отримані сегменти, надіслані сегменти, повторно передані сегменти, помилки тощо.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

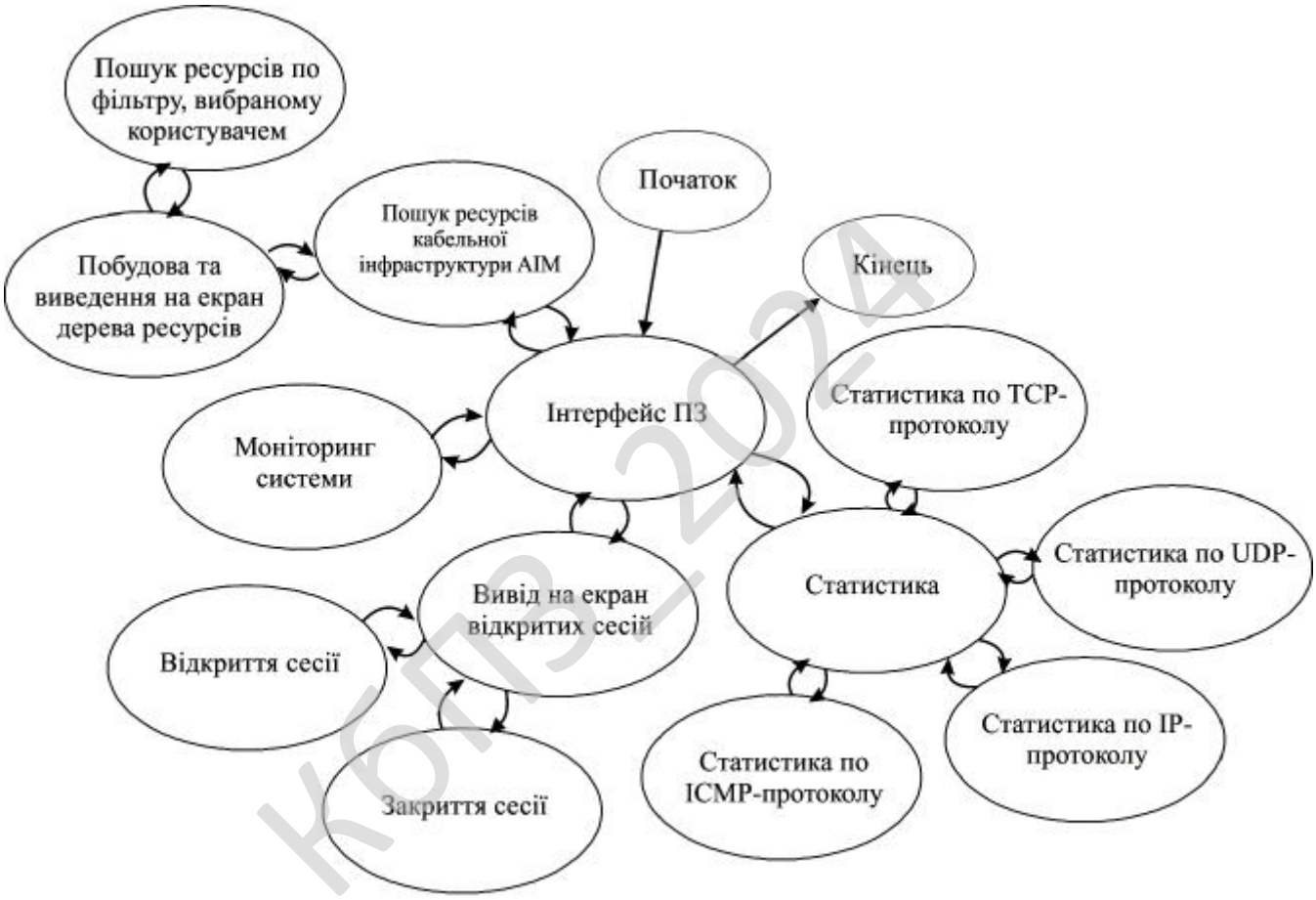

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю автоматизованого керування кабельною інфраструктурою АІМ. При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

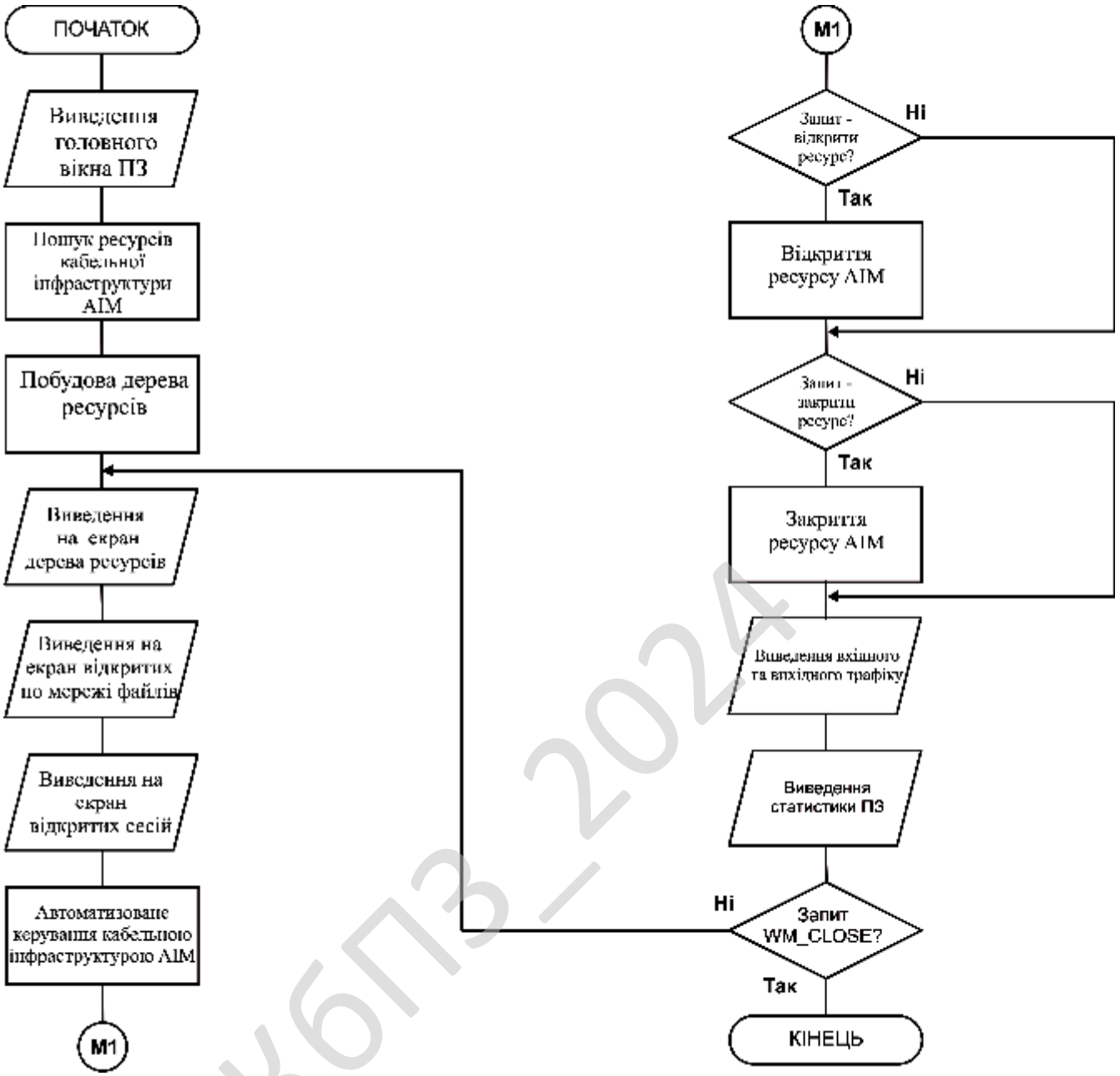

Рисунок 4.1 – Блок-схема основної програми

Перед розглядом алгоритму роботи основної програми розглянемо як була реалізована робота з точки зору UML.

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

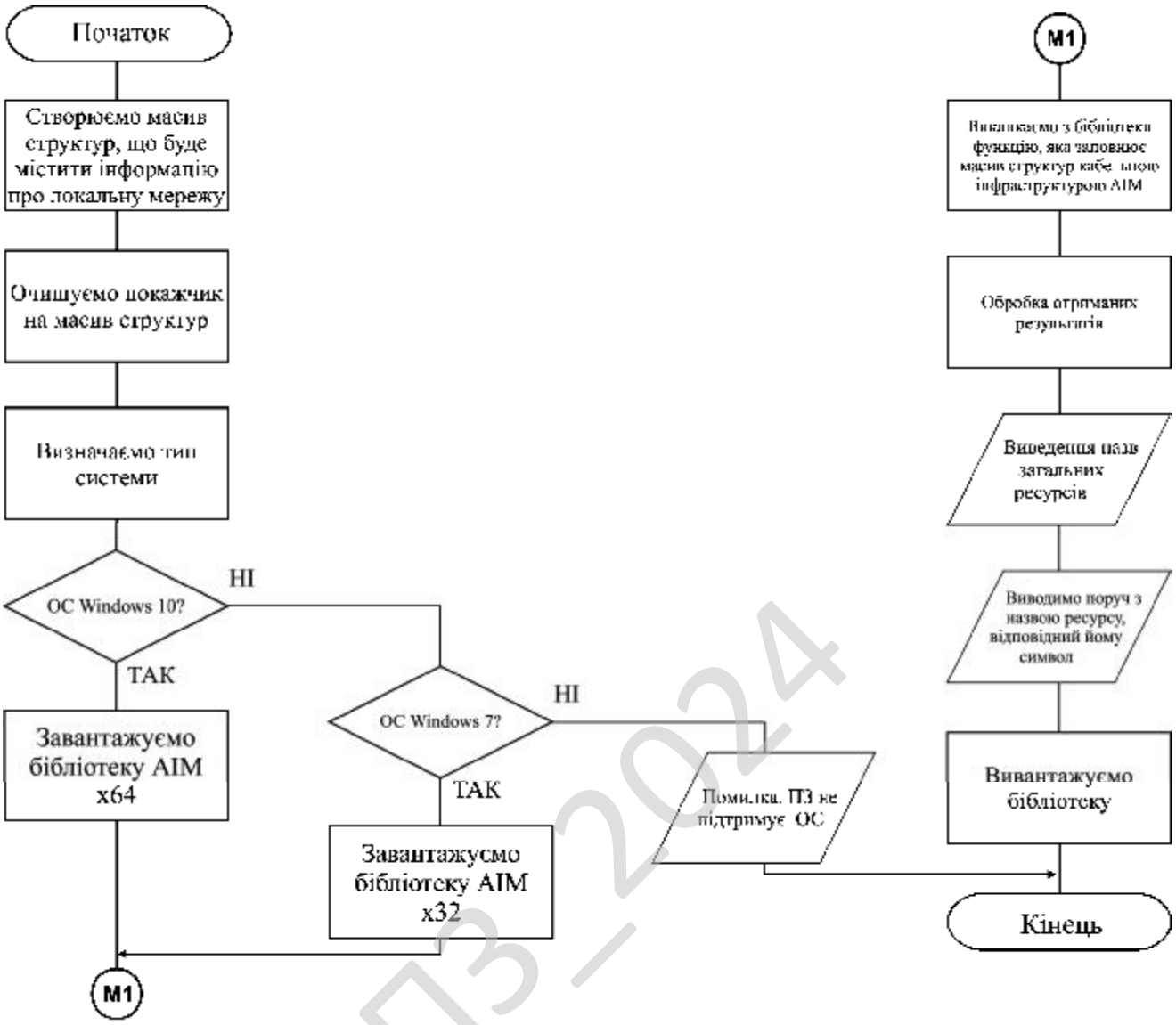

Рисунок 4.2 – Блок-схема роботи підпрограми

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати

певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: Діаграма класів; Діаграма компонент; Діаграма об'єктів.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає

необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Розглянемо код, що дозволяє вибрати папку, загальний доступ до якої ми хочемо відкрити:

```
procedure TMainForm.btnA(Sender: TObject);
const
  STYPE_DISKTREE = 0; ACCESS_ALL = 258; SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share8x : TShareInfo50;
  ShareWIN : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirWIN, TmpNameWIN: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory;
//Визначаємо шлях до майбутнього ресурсу
  TmpName := InputBox('Share name','Enter name','Test');
//Визначаємо ім'я під яким його буде видно у мережі
  if TmpDir = '' then Exit;
  if not IsWIN(OS) then Close;
//З'ясовуємо тип системи
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
```

```
    @NetShareAddWIN := GetProcAddress(FLibHandle,'NetShareAdd');
    if not Assigned(NetShareAddWIN) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256;
//Визначаємо необхідний розмір
    GetMem(TmpNameWIN, TmpLength);
//Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameWIN, TmpLength);
    ShareWIN.shi2_netname := TmpNameWIN;
//Ім'я
    ShareWIN.shi2_type := STYPE_DISKTREE;
//Тип ресурсу
    ShareWIN.shi2_remark := ''; //Коментар
    ShareWIN.shi2_permissions := ACCESS_READ;
//Доступ
    ShareWIN.shi2_max_uses := DWORD(-1);
// Кількість макс. підключ.
    ShareWIN.shi2_current_uses := 0;
// Кількість підкл.
    GetMem(TmpDirWIN, TmpLength);
    StringToWideChar(TmpDir, TmpDirWIN, TmpLength);
    ShareWIN.shi2_path := TmpDirWIN;
//Шлях до ресурсу
    ShareWIN.shi2_passwd := nil;
//Пароль
    NetShareAddWIN(nil,2,@ShareWIN,nil);
//Додаємо ресурс
    FreeMem (TmpNameWIN);
//звільняємо пам'ять
    FreeMem (TmpDirWIN);
  end else begin //Код для 8х
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,'NetShareAdd');
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
```

```
    FillChar(Share8x.shi50_netname, SizeOf(Share8x.shi50_netname), #0);
    move(TmpName[1],Share8x.shi50_netname[0],Length(TmpName));
//Ім'я
    Share8x.shi50_type := STYPE_DISKTREE;
//Тип ресурсу
    Share8x.shi50_flags := SHI50F_RDONLY;
//Доступ
    FillChar(Share8x.shi50_remark,
      SizeOf(Share8x.shi50_remark), #0);
//Коментар
    FillChar(Share8x.shi50_path,
      SizeOf(Share8x.shi50_path), #0);
    Share8x.shi50_path := PAnsiChar(TmpDir);
//Шлях до ресурсу
    FillChar(Share8x.shi50_rw_password,
      SizeOf(Share8x.shi50_rw_password), #0);
//Пароль повного доступу
    FillChar(Share8x.shi50_ro_password,
      SizeOf(Share8x.shi50_ro_password), #0);
//Пароль для читання
    NetShareAdd(nil,50,@Share8x,SizeOf(Share8x));
  end;
  FreeLibrary(FLibHandle);
end;
```

Змінні TmpNameWIN і TmpDirWIN звільняються тільки після виконання функції. Це критично, у противному випадку структура передана функції буде із двома "незаповненими" полями.

Також слід звернути увагу на те, що у версії коду для Windows 8/10 всі поля, що являють собою масив Char елементів, спочатку очищаються функцією FillChar, щоб уникнути перекручування даних (знак закінчення рядка дорівнює #0). Даний код відкриває новий ресурс на повний доступ.

Приховання й показ ресурсів. Сховати ресурс можна простим додаванням до його імені значка долара. Активувати, оберненою операцією. Це не зовсім правильно, але працює. Другий варіант сховати ресурс, це видалити його й створити його копію але із правами SHI50F_SYSTEM або вказівкою shi502_security_descriptor (для цього потрібно використовувати структуру SHARE_INFO_502).

Не можна проводити маніпуляції над наступними системними ресурсами:

IPC$, ADMIN$, PRINT$, WWWROOT$, BIOSINFO$, A$, B$, C$, D$, E$, F$, G$, H$, I$, J$, K$, L$, M$, N$, O$, P$, Q$, R$,S$, T$, U$, V$, W$, X$, Y$, Z$. Відкриття доступу до цих ресурсів робить беззахисною операційну систему.

Одержання списку поточних сесій. Для визначення користувачів підключених до нашого комп'ютера скористаємося функцією NetSessionEnum.

Оголошення функції для Windows 8/10:

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

Параметри:

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

– sLevel – повинен містити ідентифікатор структури;

– pbBuffer – повинен містити покажчик на масив структур;

– cbBuffer – повинен містити розмір масиву структур;

– pcEntriesRead – покажчик на змінну утримуючу загальну кількість структур;

– pcTotalAvial – не використовується.

Оголошення функції для Windows:

```
Var NetSessionEnum:function(ServerName, UncClientName, Username: PWChar; Level:
DWORD; bufptr: Pointer; prefmaxlen: DWORD; entriesread, totalentries,
resume_handle: LPDWORD):DWORD; stdcall;
```

Параметри:

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

– UncClientName – містить покажчик на рядок утримуючий ім'я сесії, про яку ми хочемо одержати інформацію, якщо потрібно переглянути всі сесії параметру потрібно привласнити NIL.

– UserName – містить покажчик на рядок утримуючий ім'я користувача, про яке ми хочемо одержати інформацію, якщо потрібно переглянути всі, параметру потрібно привласнити NIL.

– Level – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адресу покажчика на масив структур.

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

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

– Totalentries – не використовується.

– Resume_handle – не використовується, повинен бути NIL.

Існує також 6 типів структур, що передаються функції NetSessionEnum.

Структура session_info_50. Опис структури:

```
type
TSessionInfo50 = packed record
S50_cname       : PChar;
S50_username    : PChar;
S50_key         : Cardinal;
S50_num_conns   : Word;
S50_num_opens   : Word;
S50_time        : Cardinal;
S50_idle_time   : Cardinal;
S50_protocol    : Byte;
pad1              : Byte;
end;
```

Поля:

– S50_cname – містить покажчик на рядок утримуючий ім'я комп'ютера, який встановив сесію;

– S50_username – містить покажчик на рядок утримуючий ім'я користувача, який встановив сесію;

– S50_key – містить значення за допомогою якого ми будемо завершувати сесію;

– S50_num_conns – містить число підключень зроблених під час сесії;

– S50_num_opens – містить кількість файлів відкритих під час сесії;

– S50_time – містить час в секундах, протягом якого сесія була активна;

– S50_idle_time – містить час у секундах протягом якого сесія була неактивна;

– S50_protocol – містить ім'я протоколу, за допомогою якого клієнт зв'язується із сервером;

– Pad1 – невеликий вирівнювач структури, не використовується.

Структура session_info_502. Опис структури:

```
type
TSessionInfo502 = packed record
   S502_cname: PWideChar;
   S502_username: PWideChar;
   S502_num_opens: DWORD;
   S502_time: DWORD;
   S502_idle_time: DWORD;
   S502_user_flags: DWORD;
   S502_cltype_name: PWideChar;
   S502_transport: PWideChar;
End;
PSessionInfo502 = ^TSessionInfo502;
TSessionInfo502Array = array[0..512] of TSessionInfo502;
PSessionInfo502Array = ^TSessionInfo502Array;
```

Поля:

– S502_cname – містить покажчик на рядок утримуючий ім'я комп'ютера, який встановив сесію;

– S502_username – містить покажчик на рядок утримуючий ім'я користувача, який встановив сесію;

– S502_num_opens – містить кількість файлів відкритих під час сесії;

– S502_time – містить час у секундах протягом якого сесія була активна;

– S502_idle_time – містить час у секундах протягом якого сесія була неактивна;

– S50_user_flags – значення, що описує як користувач установив сесію;

– S502_cltype_name – тип клієнта, який встановив сесію, не використовується;

– S502_transport – містить ім'я протоколу, за допомогою якого клієнт зв'язується із сервером.

Розглянемо детальніше значення полів time і idle_time. Що означає коли сесія не активна? Наприклад Ви відкрили якийсь ресурс на віддаленій машині, просто подивитися, що в ньому знаходиться (наприклад імена файлів). У той час коли Ви нічого не робите, не копіюєте, не запускаєте, не відкриваєте файли, сесія не активна, вона чекає ваших дій. Як тільки Ви починаєте копіювати файл з віддаленої машини, сесія стає активною до закінчення копіювання.

Слід звернути увагу на поле key структури TSessionInfo50, воно містить унікальний ідентифікатор за допомогою якого можна завершити сесію.

Код одержання поточних сесій:

```
procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadWIN: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvSessions.Items.Clear;
  if not IsWIN(OS) then Close;
//З'ясовуємо тип системи
  if OS then begin  //Код для WIN
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionEnumWIN := GetProcAddress(FLibHandle, 'NetSessionEnum');
    if not Assigned(NetSessionEnumWIN) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
```

```
          if            NetSessionEnumWIN(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadWIN, @totalentries, nil)=0 then
          for i:=0 to EntriesReadWIN-1 do
          begin
            with lvSessions.Items.Add do
       //Заповнення даними зі структури
            begin
              Caption := string(SessionInfo502^[i].S502_cname);
       //Ім'я комп'ютера
              SubItems.Add(SessionInfo502^[i].S502_username);
       //Ім'я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].S502_num_opens));
       //Відкритих ресурсів
       SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].S502_Time));
       //Час                                                          активн.
SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].S502_idle_time));
       //Час не активний
            end;
          end;
         end else begin //Код для Windows 8/10
          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].S50_cname);
       //Ім'я комп'ютера
               SubItems.Add(SessionInfo50[i].S50_username);
       //Ім'я користувача
           SubItems.Add(IntToStr(SessionInfo50[i].S50_num_opens));
       //Відкритих ресурсів
       SubItems.Add(CardinalToTimeStr(SessionInfo50[i].S50_Time));
       //Час активн.
```

```
SubItems.Add(CardinalToTimeStr(SessionInfo50[i].S50_idle_time));
//Час не активн.
        SessionCloseKey[i]:= SessionInfo50[i].S50_key;
//Унікальний ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;
```

Ключі для закриття сесій слід заносити в масив у тому порядку, в якому були отримані самі сесії. Це зроблено для простоти. Якщо в списку, що відображає сесії, не застосовувати сортування, то порядковий номер виділеної для закриття сесії і її ідентифікатор у масиві збіжаться.

Завершення сесій. Для завершення відкритих сесій будемо використовувати функцію NetSessionDel.

Оголошення функції для Windows 8/10:

```
Var NetSessionDel:function(pszServer: PChar; PszClientName: PChar; SReserved:
SmallInt):DWORD; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо завершується сесія в себе, то даному параметру потрібно привласнити NIL;

– pszClientName – повинен містити ім'я клієнта, чия сесія завершується;

– sReserved – повинен містити унікальний ключ для завершення сесії (той який ми одержали попередньою функцією).

Оголошення функції для Windows:

```
Var NetSessionDel:function(ServerName, UncClientName, Username:PWChar):DWORD;
stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо завершується сесія в себе, то даному параметру потрібно привласнити NIL;

– uncClientName – повинен містити ім'я клієнта, чия сесія завершується,

якщо параметр NIL, завершаться всі сесії зазначені в параметрі username;

– username – повинен містити ім'я користувача, чия сесія завершується, якщо параметр NIL, завершаться всі сесії зазначені в параметрі uncClientName.

Як можна помітити, ніяких структур дані функції не використовують. Напишемо процедуру яка буде завершувати обрану нами в lvSessions сесію по ім'ю клієнта. От як цей код виглядає:

```
procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameWIN: PWideChar;
  CName8x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsWIN(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;
    @NetSessionDelWIN := GetProcAddress(FLibHandle, 'NetSessionDel');
    if not Assigned(NetSessionDelWIN) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameWIN := PWChar(WideString('\\'+lvSessions.Items.Item[i].Caption));
    NetSessionDelWIN(nil,CNameWIN,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
        CName8x := PAnsiChar(lvSessions.Items.Item[i].Caption);
        key := SessionCloseKey[i]; //Беремо ключ із масиву
        NetSessionDel(nil,CName8x,Key);
      end;
      FreeLibrary(FLibHandle);
    end;
```

Одержання списку відкритих файлів. Розглянемо функцію NetFileEnum. Оголошення функції для Windows 8/10:

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

Параметри:

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

– pszBasePath – повинен містити каталог, відкриті файли з якого ми хочемо подивитися, якщо ми хочемо переглянути всі відкриті файли параметру потрібно привласнити NIL;

– sLevel – повинен містити ідентифікатор структури;

– pbBuffer – повинен містити покажчик на масив структур;

– cbBuffer – повинен містити розмір структури;

– pcEntriesRead – покажчик на змінну утримуючу загальну кількість структур;

– pcTotalAvial – не використовується.

Оголошення функції для Windows:

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

Параметри:

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

– BasePath – повинен містити каталог, відкриті файли з якого ми хочемо подивитися, якщо ми хочемо переглянути всі відкриті файли параметру потрібно привласнити NIL.

– UserName – містить покажчик на рядок утримуючий ім'я користувача, про яке ми хочемо одержати інформацію, якщо потрібно переглянути всі, параметру потрібно привласнити NIL.

– Level – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адресу покажчика на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах, якщо не ставити обмеження те даному параметру потрібно привласнити DWORD(-1).

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

– Totalentries – не використовується.

– Resume_handle – не використовується, повинен бути NIL.

Структура file_info_50. Опис структури:

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

Поля:

– Fi50_id – містить ідентифікаційний номер відкритого файлу.

– Fi50_permissions – містить рівень доступу, з яким відкритий файл.

– Fi50_num_locks – містить кількість блокувань файлу.

– Fi50_pathname – містить повний шлях до відкритого файлу.

– Fi50_username – містить ім'я користувача або комп'ютера открывшего файл.

– Fi50_sharename – містить ім'я загального ресурсу, в якому перебуває відкритий файл.

Структура FILE_INFO_3. Опис структури:

```
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;
```

Поля:

– Fi3_id – містить ідентифікаційний номер відкритого файлу.

– Fi3_permissions – містить рівень доступу, з яким відкритий файл.

– Fi3_num_locks – містить кількість блокувань файлу.

– Fi3_pathname – містить повний шлях до відкритого файлу.

– Fi3_username – містить ім'я користувача або комп'ютера, який відкрив файл.

Тепер напишемо код який покаже нам список відкритих файлів на нашому комп'ютері:

```
procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoWIN: PFileInfo3Array;
  FileInfo8x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadWIN: 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 WIN
          FLibHandle := LoadLibrary('NETAPI32.DLL');
          if FLibHandle = 0 then Exit;
          @NetFileEnumWIN := GetProcAddress(FLibHandle, 'NetFileEnum');
          if not Assigned(NetFileEnumWIN) then
          begin
            FreeLibrary(FLibHandle);
            Exit;
          end;
          FileInfoWIN := nil;
          if NetFileEnumWIN(nil,nil,nil,3,@FileInfoWIN,DWORD(-1),@entriesreadWIN,
@totalentries, nil)=0 then
          for i:=0 to EntriesReadWIN-1 do
          begin
            with lvFiles.Items.Add do
      //Заповнення даними зі структури
            begin
              Caption := string(IntToStr(FileInfoWIN^[i].fi3_id));
      //Ідентифікатор
              SubItems.Add(FileInfoWIN^[i].fi3_pathname);
      //Шлях до файлу
              SubItems.Add(FileInfoWIN^[i].fi3_username);
      //Ім'я користувача
            end;
          end;
        end else begin
          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,@FileInfo8x,SizeOf(FileInfo8x),@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(FileInfo8x[i].fi50_id));
        //Ідентифікатор
               SubItems.Add(FileInfo8x[i].fi50_pathname);
        //Шлях до файлу
               SubItems.Add(FileInfo8x[i].fi50_username);
        //Ім'я користувача
             end;
          end;
         end;
         FreeLibrary(FLibHandle);
        end;
```

Закриття відкритого файлу. Тепер розглянемо функції NetFileClose і NetFileClose2, за допомогою яких будемо закривати відкриті по мережі файли.

Функція NetFileClose2 – використовується тільки в Windows 8/10:

```
Var NetFileClose2:function(pszServer:PChar; UlFileId:LongWord):DWORD; stdcall;
```

Поля:

– pszServer – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо в себе, то даному параметру потрібно привласнити NIL.

– UlFileId – повинен містити унікальний ідентифікатор відкритого файлу.

Функція NetFileClose – використовується тільки в Windows:

```
Var NetFileClose:function(ServerName:PWideChar; FileId:DWORD):DWORD; stdcall;
```

Поля:

– ServerName – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо в себе, то даному параметру потрібно привласнити NIL.

– FileId – повинен містити унікальний ідентифікатор відкритого файлу.

Як перший параметр передамо NIL (закриваємо в себе), у якості другого, ідентифікатор файлу, що відображається у полі ID. От приклад коду:

```
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 WIN
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, 'NetFileClose');
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));
//Закриваємо
  end else begin //Код для Windows 8/10
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, 'NetFileClose2');
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
//Закриваємо
  end;
  FreeLibrary(FLibHandle); end;
```


4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій. Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа і-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Висновок

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

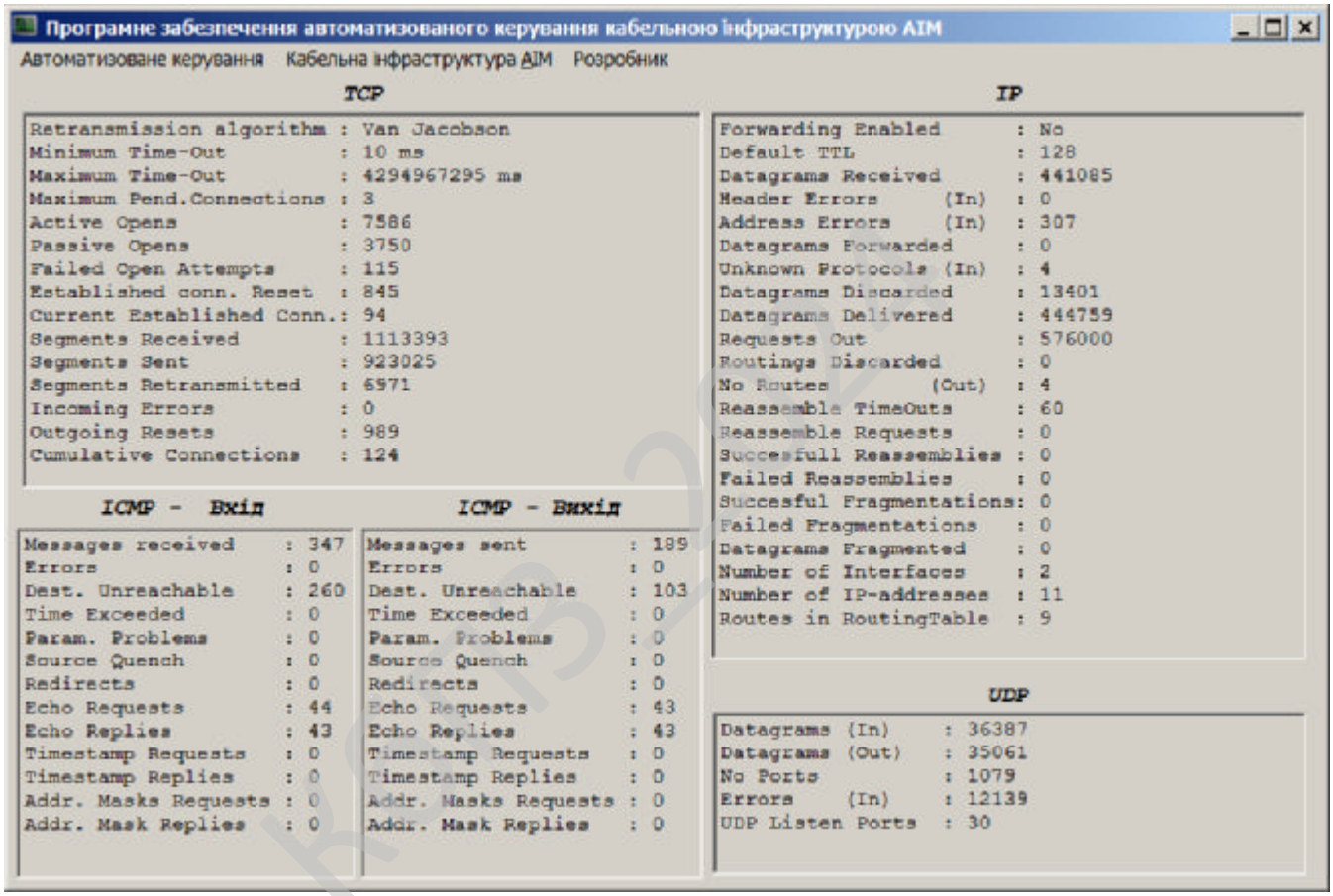

Рисунок 5.1 – Головне вікно розробленого ПЗ

Розроблене програмне забезпечення автоматизованого керування кабельною інфраструктурою AIM складається з наступних функціональних блоків:

– Навігаційне меню: Автоматизованого керування; Кабельна інфраструктура AIM; Розробник.

– Інформаційних панелей: TCP; IP; ICMP; UDP.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

ПЗ дозволяє переглядати всі з'єднання, що відбуваються у мережі з можливістю проведення автоматизованого керування кабельною інфраструктурою АІМ. ПЗ складається з двох полів, в першому показані з'єднання по TCP-протоколу, в тругому – по UDP-протоколу. Також при необхідності можна переглянути вікно статистики.

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

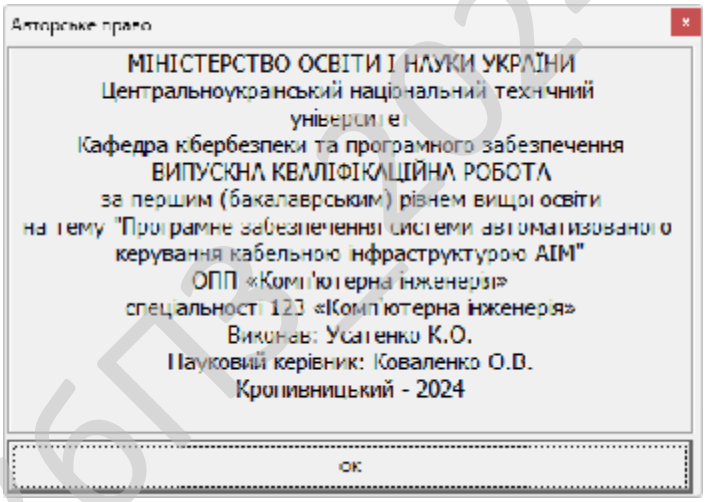

Рисунок 5.2 – Вікно розробника ПЗ

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи автоматизованого керування кабельною інфраструктурою AIM.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем автоматизованого керування кабельною інфраструктурою AIM.

– Досліджена система автоматизованого керування кабельною інфраструктурою AIM.

– На основі отриманих результатів досліджень створена програмна реалізація системи автоматизованого керування кабельною інфраструктурою AIM.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання автоматизованого керування кабельною інфраструктурою AIM.

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 RAD Studio Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи автоматизованого керування кабельною інфраструктурою AIM. 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 MISTY1.

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. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

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

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

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

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

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

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

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

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

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

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

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

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

34. 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.

35. 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.

36. 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.

37. 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.

38. 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

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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).

46. 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

47. 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.

48. 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.

49. 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.

50. 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.

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи автоматизованого керування кабельною інфраструктурою АІМ.

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. № 132-02 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 автоматизованого керування кабельною інфраструктурою АІМ.

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 автоматизованого керування кабельною інфраструктурою AIM;

– 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 RAD Studio 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 – 82 аркуші.

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 6.06.2024 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

_____ Коваленко О.В.

Програмне забезпечення системи автоматизованого керування кабельною інфраструктурою АІМ

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2024 року

Основна програма

Файл Control_AIM.dpr основної програми

```
program Control_AIM;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

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- 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
//

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 х-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;

////////////////////////////
//
//  Показу 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 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-У максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-У тік 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;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати кіл-у 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
//

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
        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
//

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
//

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 про всякий випадок таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' IPHLPAPI.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
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт
      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 о у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES данні на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES данні 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 IPHLPAPI.pas- обробка API функцій

```
unit IPHLPAPI;

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;

 found in 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;    // данні
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // данні
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // данні
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати данні зразу. Стан >= DISCONNECTED
//
      // може передавати деякі данні. 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
      // данні 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;       //
данні
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// данні
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // данні
          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 ;   // данні- простір для списку IP
адрес
        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 IPHLPAPI.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 баги при використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
    Result := True;
    if IpHlpModule <> 0 then Exit;

// відкрити DLL
```

```
          IpHlpModule := LoadLibrary (IpHlpDLL);
          if IpHlpModule = 0 then
          begin
              Result := false;
              exit ;
          end ;
          GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
          GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' )
;
          GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
          GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
          GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
          GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
          GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
          GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
          GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
          GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
          GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
          GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
          GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
          GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
          GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
      end;

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

      end.
```

Файл IPHelper.pas - функції роботи з IP-протоколом

```
unit IPHelper;


interface

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

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- данні}
    ( 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 IpHlpNetworkParams
  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 IpHlpAdaptersInfo
  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 данні---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (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 IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (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 IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(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 }

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (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 IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // данні
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // данні
begin
```

```
    InfoSize := 0 ;   // данні
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // данні
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // данні
    GetMem (FixedInfo, InfoSize) ;                    // данні
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // данні
    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 ;  // данні
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // данні
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // данні
    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 IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // данні
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // данні
  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 := IpHlpIfTable (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] )  // данні, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp 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 IpHlpAdaptersInfo(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 := IpHlpAdaptersInfo(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 ;

//-----------------------------------
{ моні торимо час доступу до IP }
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 );   // данні
  //
  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 ) ); // бермо 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 );  // данні
  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 LoadIpHlp 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 IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp 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 );  // данні
  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; }
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 ;   // данні
          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 LoadIpHlp 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 ) );
// данні
      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 IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp 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 IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp 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( ' Ехо запит        : ' + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо 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( 'Ехо запит        : ' + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо 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- монітор TCP/IP з'єднань

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, 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 LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ‘ Інтернет помічник DLL не є доступним, або не підтримується’
) ;
end;

end.
```

Файл Stat.pas- статистика мережі

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, 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 LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл About.pas- довідка

```
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.
```