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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL”

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

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

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

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

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

Купчину Дмитру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Купчин Д.Ю. Дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Метою розробки є дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Об'єктом дослідження є процес управління маршрутизацією у центрах обробки даних за технологією TRILL.

Предметом дослідження є методи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерні науки, управління маршрутизацією

ABSTRACT

Kupchyn D.Yu. Research and software implementation of the routing management system in data processing centers based on TRILL technology. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the routing management system in data processing centers based on TRILL technology.

The purpose of the development is the research and software implementation of the routing management system in data centers based on the TRILL technology.

The object of the study is the routing management process in data processing centers using the TRILL technology.

The subject of research is routing management methods in data processing centers based on TRILL technology.

Research methods are based on methods of cloud technologies, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the routing management system in data processing centers based on TRILL technology.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: computer science, routing management

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ПЗ	–	програмне забезпечення
ЦОД	–	центр обробки даних
QoS	–	механізми контролю й керування якістю

ВСТУП

Актуальність теми. На зміну архітектурі «клієнт-сервер», яка багато років домінувала в IT, приходить сервіс-орієнтована архітектура. Якщо раніше більшість процесів серверної частини виконувалися на одному фізичному сервері, а тому при взаємодії між ними мережа не задіялася, то зараз для підвищення ефективності використання обчислювальних ресурсів окремі процеси розподіляються по різних серверах, що істотно підвищує навантаження на мережу. У сервіс-орієнтованій архітектурі більшість даних залишається усередині центрів обробки даних (ЦОД, дата центри), і передається між його встаткуванням, тоді як частка трафіку між клієнтами й серверами в загальному обсязі даних, що пересилаються по мережах, знижується. На першій стадії розвитку технологій віртуалізації вона використовувалася для підвищення ефективності використання ресурсів окремих серверів, що лише незначно збільшувало навантаження на мережу. З появою рішень на зразок VMotion стала можливою міграція віртуальних машин (без переривання роботи додатків) для підвищення ефективності використання ресурсів серверного парку в цілому. Це привело до різкого росту трафіку по горизонталі. Один тільки факт «розвороту на 90°» основного напрямку передачі трафіку вже робить малоефективними як традиційну ієрархічну архітектуру мереж (доступ – агрегація – ядро), так і логічні структури на зразок «дерева», які були оптимізовані для пересилання трафіку від «кореня» до «листів» і назад, тобто по вертикалі. До цьому варто додати ріст інтересу замовників до конвергенції мереж – до впровадження технології FCoE, для якої потрібна гарантована передача трафіку без втрат, а також до повноцінного віртуалізації мережної інфраструктури для підтримки вже віртуалізованих серверів і переходу до хмарної моделі надання/одержання IT-сервісів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

– Огляд існуючих систем управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Дослідження системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Об'єктом дослідження є процес управління маршрутизацією у центрах обробки даних за технологією TRILL.

Предметом дослідження є методи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Розроблено вітчизняний продукт управління маршрутизацією у центрах обробки даних за технологією TRILL, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління маршрутизацією у центрах обробки даних за технологією TRILL.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями

параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для управління маршрутизацією у центрах обробки даних за технологією TRILL**.** TRILL (Transparent Interconnection of Lots of Links) – мережна технологія, що дозволяє здійснювати т.зв. “маршрутизацію” на рівні L2, на відміну від стандартної маршрутизації, що здійснюється на рівні L3.

TRILL працює по такій же логіці як і стандартна маршрутизація, ухвалюючи рішення щодо виборі маршруту за результатами обчислення найкоротшого шляху, але робить це не для IP, а для MAC-адрес.

TRILL – один з Ethernet-стандартів, що підвищує можливості мережних технологій у хмарних обчисленнях:

– збільшення комутаційної ємності;

– кращий шлях для досягнення призначення;

– необхідна підтримка мобільних серверів у середовищі віртуалізації;

– більше швидку збіжність при перегонах напруги й виникненні помилок.

Комутатори, що підтримують технологію TRILL, називають маршрутизуючими мостами, або RBridge.

TRILL призначена для Layer 2 мереж великого масштабу, дата-центрів, підключення до мережі хмарних обчислень, спрощуючи міграцію віртуальних машин, а також більше ефективного використання мережі й значного поліпшення стабільності серверних рішень дата-центрів.

Для обчислення найкращого шляху до пункту призначення комутатори RBridge використовують протокол IS-IS, заснований на відомому алгоритмі Shortest Path First (SPF). Комутатор, що перебуває на вході в хмару TRILL, за допомогою IS-IS відразу визначає 16-розрядний ідентифікатор комутатора на виході. Кожний наступний комутатор (транзитний вузол) у хмарі пересилає

трафік на основі цього ідентифікатора, завдяки чому усередині хмари не потрібно підтримувати таблицю зовнішніх MAC-адрес. Вузли оперують дуже невеликим обсягом адресної інформації, що спрощує їхнє завдання, зокрема, по розподілі трафіку по множині маршрутів. У технології TRILL уводиться такий важливий параметр, як «час життя» – Time To Live (TTL): при проходженні фреймом кожного вузла в мережі TRILL, значення цього параметра зменшується. Цей механізм відсутній у класичній технології Ethernet, що багато в чому і є причиною зациклення трафіку – без поля TTL кадр Ethernet може нескінченно довго «подорожувати» по мережі, поки не досягне адресата.

1.2 Область застосування

Областю застосування системи є центри обробки даних (ЦОД). Нові рішення, які в англомовній літературі одержали найменування Fabric, на думку більшості експертів, дозволять зняти проблеми, що нагромадилися, і зробити мережі ефективною транспортною основою для віртуалізованих ЦОД. У галузі вже сформувалося загальне розуміння того, які повинні бути основні характеристики «фабрик»:

- підтримка всіх ліній зв'язку в активному стані (без властивому протоколу STP блокувань);
- використання найкоротших шляхів пересилання;
- забезпечення низької затримки і її варіації;
- повна відказостійкість при відсутності точок загальносистемної відмови.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

З розвитком Інтернет-ринку усе більше затребуваними є послуги з розміщення й оренди серверів. Тому останнім часом українські телекомунікаційні оператори кинулися створювати й удосконалювати центри обробки даних (ЦОД).

Ринок

По оцінках фахівців «Голден Телекому», річний оберт українського ринку послуг ЦОД становить до 5 млн. дол. у рік. Йому є куди рости, оскільки світовий ринок – близько 15 млрд. дол., тобто Україна займає на ньому мікроскопічно малу частку – усього 0,033%. Ще якихось пари років тому дешевше було орендувати сервер у США, ніж в Україні. Сьогодні з падінням тарифів на закордонний трафік і зниженням тарифів на хостинг і колокацію, обсяги даних щорічно ростуть більш ніж на 50%, на таку ж величину росте й ринок ЦОД. У цей час витрати компаній на зберігання інформації становлять більше 15% IT-бюджетів.

В Україні даний вид бізнесу тільки починає активно розвиватися, що основному обумовлено деяким відставанням українського сегмента Інтернет, що є основним каталізатором росту попиту на ЦОД.

В Україні багато провайдерів, що надають послуги хостингу, але великих операторів, що демонструють серйозний підхід до організації бізнесу ЦОД і які мають частку ринку близько 50%, – не більше 5-8 компаній. Не дозволяє вийти на цей ринок висока планка входу – середня вартість облаштованості приміщень

ЦОД, де потрібно розгорнути потужне сучасне встаткування, що становить близько 1,5-2,5 тис. дол. за м².

Сучасні ЦОД повинні відповідати основним критеріям. Вони повинні мати систему безперебійного електроживлення, мати незалежні джерела електроживлення, а також як мінімум одне джерело автономного живлення, на випадок якщо відбудеться відключення від загальних мереж. Також повинна бути потужна система кондиціювання повітря, також забезпечена резервом. У ЦОД повинні дотримуватися міри безпеки: твердий пропускний режим і цілодобова охорона (останнім часом виникали безпрецедентні випадки, коли здійснювалися захватів серверів). Повинне бути налагоджене резервування всіх систем, безперебійний режим надання послуг, і задіяне високоякісне професійне промислове встаткування. Для ЦОД важливі також швидкісні канали включення в Інтернет. Для українських ЦОД – це крім міжнародних ще й гігабітні включення в UA-IX – точку обміну трафіком.

Канали

Серед самих потужних в Україні ЦОД компаній «Воля», «Колоколл», «Укртелеком», «Голден-Телекому». Але «Укртелеком» в основному використовує ЦОД для власних потреб.

Самі широкі канали сьогодні – в «Воля» – закордонний 1,7 Гбіт/с, а в UA-IX – аж 10 Гбіт/с. При цьому залишене старе з'єднання з UA-IX на швидкості 2 Гбіт/с у якості резервного. Пропускна здатність українських каналів ЦОД «Голден Телекому» становить більше 1Гбіт/с, закордонних – 300 Мбіт/с. «Колоколл» має пропускну потужність за рубіж – 500 Мбіт/с. Якщо донедавна ріст сумарного трафіку усередині мережі UA-IX збільшувався в 2-4 рази щорічно й на момент переходу на 10 Гбіт/с його обсяг становив 7-8 Гбіт/с. У той же час постійно ростуть швидкості підключень учасників. Так якщо наприкінці минулого року 36 компаній працювали на швидкості 1 Гбіт/с, того 1 вересня поточного року вже 54 компанії працювали на цій швидкості. І перші компанії почали підключатися на швидкості 10 Гбіт/с.

Недавно була відкрита нова технічна площадка ЦОД «Голден-Телеком». На облаштованість ЦОД було витрачено більше 200 тис. дол. Плануються ще інвестиції в інфраструктуру в розмірі 150 тис. дол. У компанії хочуть створити кращий ЦОД на українському ринку. Але зробити це буде непросто, оскільки конкуренти не дрімають.

Ціни

Сьогодні орендувати сервер можна за ціною від 199 грн. на місяць. По колишньому вартість послуг залежить від кількості споживаного закордонного трафіку. І лімітується в першу чергу вхідний закордонний трафік. У такий спосіб оператори створили захисний механізм від зловживань, пов'язаних з перегонкою трафіку й файлообміном. При цьому обмежується й активність дрібних провайдерів, які по суті через сервер можуть організовувати доступ в Інтернет або створювати проксі-сервера. Також у такий спосіб створені перешкоди для міжнародного файлообміна.

Незважаючи на те, що в тарифах «Голден Телеком» 1 ГБ закордонного трафіку поки що оцінюється в 10 доларів, у компанії обіцяють у найближчому майбутньому конкурентні тарифи. Це в першу чергу стосується тарифів без обмеження трафіку: на швидкості 2 Мбіт/с – 190 дол./мес., а 1 Мбіт/с – за 140 дол. У ЦОД «Волі» з 1 листопада вводиться тариф з 1 Мбіт/із для вхідні й 20 Мбіт/с – для вихідного закордонного трафіку за 1199 грн./мес. А ще оператор надає послугу «Телепорт» за 9999 грн./мес., у якій надається в оренду ціла стійка в ЦОД, а вхідний закордонний трафік лімітується швидкістю аж 10 Мбіт/с.

Перспективи

По-справжньому великих ЦОД в Україні поки немає. По світових мірках існуючі ЦОД належать до корпоративного рівня.

Великий ЦОД повинен бути по площі як великий гіпермаркет з капіталізацією не менш 5 млн. дол. Саме в такому виді бізнес стає високорентабельним.

Це повинні бути більші склади на великій території з інженерною інфраструктурою, підведенням великих потужностей електроживлення, установкою потужних генераторів. Зараз в Україні поки створюються ЦОД у містах, у житлових будинках, іноді орендуються приміщення в офісних будинках. Також постійно виникають складності з електроживленням, оскільки доводиться підключатися до звичайних трансформаторних станцій. Електричні потужності – головний фактор, що стримує ріст ЦОД. Тому перспективніше встановлювати ЦОД у колишніх промзонах, де хоча б підведені потужності, розраховані на енергоємне виробництво. Але потрібно враховувати, що більшість трансформаторних підстанцій уже морально й фізично застаріло, оскільки встановлені ще з радянських часів.

Так, у Києві недавно був бум невеликих ЦОД. І коли початку відчуватися недостача електроенергії, такі ЦОД масово відключалися, і їм не допомагали ніякі генератори.

У США, де будуються наймогутніші ЦОД, зовсім інший підхід до бізнесу. Посередині пустелі розвертаються величезні комплекси, що включають відповідну інфраструктуру, альтернативні джерела енергії, із сонячними батареями, вітровими станціями. Є навіть невеликі «зелені» ЦОД, що працюють повністю на альтернативних джерелах енергії. А великі компанії будують навіть під свої потрібні величезні корпоративні ЦОД. Так, недавно Google почав будівництво ЦОД з покупки електростанції.

Такого рівня ЦОД в Україні виникнуть тоді, коли прийдуть професійні гравці в області ЦОД або місцеві компанії доростуть до професійної гри в ЦОД. Швидше за все це будуть паралельні процеси.

Лідери українського ринку є основними претендентами, які можуть вирости до світового рівня. Але для цього потрібно створювати інфраструктуру в нових місцях, наприклад, будувати потужні ЦОД у південних степах, де можна було б використовувати вітрову й сонячну енергію. Плюс така робота буде сприяти створенню нових робочих місць. Але й весь Інтернет-ринок повинен

вирости ще на порядок, щоб українським операторам було вигідно будувати вилучені ЦОД. Адже коли близько 60% користувачів і трафіку знаходяться в Києві, немає необхідності виводити потужності так далеко.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за

допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL,

включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

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

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

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

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній

формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Приведемо опис технології TRILL (Transparent Interconnection of Lots of Links).

TRILL дозволяє легко конфігурувати Ethernet. Концепція обробки пакетів при відомому місці призначення:

– RBridges запускає протокол стану зв'язків, за допомогою якого маршрутизатору відомо про всіх RBridges у мережі й стани лінків між ними. Використовуючи цей протокол, кожний RBridges розраховує коротку відстань між собою й кожним RBridges у мережі, а так само дерева для доставки multidestination трафіку.

– Коли RBridges1 посилає Ethernet фрейм від кінцевого вузла A, кінцевому вузлу B, що перебуває за іншим RBridges2, то RBridges1 інкапсулює фрейм в TRILL заголовок і пересилає RBridges2. Заголовок TRILL складається з полів "ingress RBridge", "egress RBridge" і лічильника.

– Коли RBridges2 приймає інкапсульований пакет, RBridges2 знімає TRILL заголовок і відправляє пакет кінцевому вузлу B.

Заголовок TRILL

Основне поля заголовка TRILL – ingress RBridge nickname (16 bits), egress RBridge nickname (16 bits), hop count (6 bits), й multidestination flag bit (1 bit). Довжина поля призначення 16 біт, тому може бути складена таблиця для простого пошуку вихідного порту, у відмінності від Ethernet 6-ти байтового значення, що вимагає хешування, або довгих префіксів, що погодять, IP.

Learning End-Node Locations

За замовчуванням механізм навчання перепискою між ingress RBridge, source MAC address і коли egress RBridge декапсулює пакет. Якщо RBridge1 не

знає де перебуває dstMAC-адреса, то RBridge1 інкапсулює пакет у заголовок TRILL й встановлює multidestination flag, указуючи тим самим, що пакет може бути переданий через все дерево RBridge.

Додатковим опціональним механізмом є End-Station Address Distribution Information (ESADI). ESADI дозволяє RBridge1 представляти декілька або всі кінцеві вузли, приєднані до нього. Обидва механізми (подання й прослуховування) є опціональними. Вони мають ряд переваг:

– Пакети ESADI можуть мати криптографічний захист.

– RBridge має більше підстав знати, що конкретний вузол прикріплений до нього, ніж звичайний перегляд заголовка пакета.

– На RBridge може бути реалізований таймер для перевірки кінцевих вузлів.

Крім того можна створити каталог, у якому перераховані не тільки (RBridge nickname, {set of attached end-node MAC addresses}), але також {(end-node IP address, end-node MAC address)}. RBridge1 або hypervisor, або процес кінцевого вузла можуть запросити інформацію про вузол призначення й інкапсульованих пакетах замість flooding, що дозволяє обійти використання протоколів ARP (IPv4) і ND (IPv6).

Link State Protocols

Це маршрутизуючий протокол, у якому кожний маршрутизатор визначає своїх сусідів розсилаючи broadcastsLink State Packet (LSP). Всі маршрутизатори мають трохи LSPdatabase, тому що вони всі одержують і зберігають останнє згенероване LSP кожного іншого маршрутизатора. LSPdatabase подає повну інформацію, необхідну для обчислення шляху. І цієї інформації досить для всіх маршрутизаторів для розрахунку сполучного дерева без необхідності використання алгоритму STP.

Acquiring Nicknames

З огляду на, що останній сформований пакет поширюється й зберігається на RBridge, те за допомогою цієї функції можна поширювати й іншу інформацію,

таку як робота протоколу для вибору унікального імені маршрутизатора в мережі. Кожний маршрутизатор вибирає ім'я випадково, уникаючи вже існуючих у мережі імен. Якщо 2 маршрутизатори вибрали однакові імена, то суперечка дозволяється пріоритетом маршрутизатора й 6-byte system ID.

Mixing RBridges with Bridges

TRILL улаштований так, що будь-яка підмножина мостів у мережі можуть бути замінені на RBridges. Набір каналів з'єднаних мостами буде сприйматися як RBridges одна загальна сполучна ланка. Усередині міст буде поводитися як звичайний міст, створюючи каркас для пересилання пакетів.

У доповненні TRILL заголовка, коли RBridge1 пересилає TRILL-інкапсульований фрейм до RBridge2 може бути додане поле типу з'єднання. Хоча в основному TRILL працює з Ethernet фреймами, але він так само може працювати з PPP, IPSecurity tunnel, MPLS. Якщо лінк Ethernet, то зовнішній заголовок Ethernet, якщо PPP, то PPP і т.д.

Зовнішні заголовки потрібні для двох цілей:

– Якщо це міст на з'єднанні, то він буде сприймати пакет як звичайний Ethernet пакет і пересилати його через сполучне дерево. Вивчаюча таблиця мосту на лінку буде бачити тільки адресу RBridge на цьому лінку.

– Це дозволяє R1 при відправленні фрейму з multiple neighbors, указувати, який з них обраний для пересилання фрейму. Для приклада: якщо однакова вартість шляху до пункту призначення від R2 і R3, то R1 потрібно вказати який R2 або R3 буде пересилати фрейм, у противному випадку фрейм буде дубльований.

У такий спосіб інкапсульований фрейм може мати три заголовки:

– Зовнішній заголовок або заголовок hop-by-hop, що знімається на кожному хопі і є специфічним для типу ланки.

– Заголовок TRILL, що за аналогією третього рівня залишається на місці, поки фрейм пересилається від першого RBridge (інкапсулюючого) до останнього (деінкапсулюючого).

– Внутрішній заголовок, що визначає адреси кінцевих вузлів (srcи dst).

Appointed Forwarders

Якщо два RBridges на лінку підключені до одного кінцевого вузла, то тільки один з них повинен інкапсулювати пакет в TRILL. Проте, якщо кінцевий вузол передасть мультикастовий пакет або пакет unknown destination, те R1 інкапсулює пакет і передасть у мережу, R2 одержить цей пакет, і деінкапсулює його. Деінкапсульований пакет знову буде переданий у мережу, прийнятий R1 і знову інкапсульован.

Вартість пройдених хопів не вирішить цю проблему, тому що вона не існує, поки пакет не інкапсульован.

У протоколі IS-IS один з RBridges вибирається Designated RBridge (DRB). DRB може делегувати іншим RBridges роботу з інкапсуляції/деінкапсуляції пакетів з певним VLAN. Тим самим R2 інкапсулює пакети з VLANx, R3 з VLANy та R1 з VLANz.

3.2 Розробка структурної схеми

Технологія TRILL визначена в серії документів організації IETF (RFC 5556, 6325, 6327, 6349), але деякі механізми перебувають тільки в стадії розгляду. Часто неї називають маршрутизацією на рівні L2. Як відомо, класична маршрутизація виконується на підставі інформації рівня L3, при цьому рішення про вибір маршруту здійснюється за результатами обчислення найкоротшого шляху. TRILL реалізує схожу логіку, але тільки не для IP-, а для MAC-адрес. Не дивно, що «мовою» TRILL підтримуючу цю технологію комутатори називаються маршрутизуючими мостами, або RBridge.

Для обчислення найкращого шляху до пункту призначення комутатори RBridge використовують протокол IS-IS, заснований на відомому алгоритмі Shortest Path First (SPF). Комутатор, що перебуває на вході в хмару TRILL, за допомогою IS-IS відразу визначає 16-розрядний ідентифікатор комутатора на

виході. Кожний наступний комутатор (транзитний вузол) у хмарі пересилає трафік на основі цього ідентифікатора, завдяки чому усередині хмари не потрібно підтримувати таблицю зовнішніх MAC-адрес. Вузли оперують дуже невеликим обсягом адресної інформації, що спрощує їхнє завдання, зокрема, по розподілі трафіку по множині шляхів. У технології TRILL уводиться такий важливий параметр, як «час життя» – Time To Live (TTL): при проходженні кадром кожного вузла в мережі TRILL значення цього параметра зменшується. Цей механізм відсутній у класичній технології Ethernet, що багато в чому і є причиною зациклення трафіку – без поля TTL кадр Ethernet може нескінченно довго «подорожувати» по мережі, якщо не досягне адресата.

У цей час кілька виробників при описі своїх рішень згадують про технологію TRILL. Зокрема, Cisco називає свою технологію FabricPath, підтримувану пристроями серій Nexus 5000 і 7000, сумісної з TRILL. Однак незалежні експерти відзначають ряд відступів від стандарту – зокрема, інший формат кадру, що використовується для передачі трафіку між комутаторами. Але оскільки Cisco бере активну участь у триваючій стандартизації TRILL, висока ймовірність, що фірмові функції згодом стануть частиною стандартів. Властиво, таке вже багаторазово відбувалося при формуванні стандартів на інші мережні технології.

На рисунку 3.1 наведена структурна схема системи у вигляді ЦОД, у якому застосовується комбінація технологій Cisco FabricPath і FEX. Sxxx – це номера (ідентифікатори) комутаторів, використовувані для доставки кадру усередині мережі FabricPath. Так, у кадр, відправлений вузлом з MAC-Адресою A вузлу з MAC адресом C, на вході в мережу FabricPath додається заголовок, де як номер вихідного комутатора вказується S300, і подальша передача до виходу з мережі FabricPath буде здійснюватися на підставі цього номера.

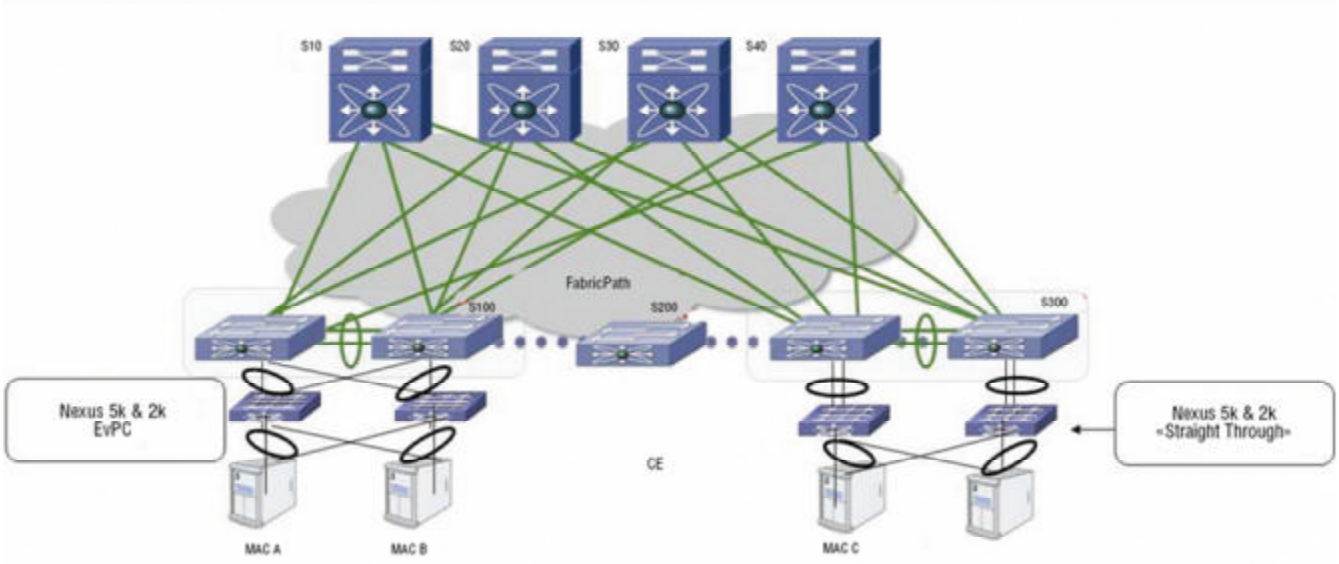

Рисунок 3.1 – Структурна схема системи

Показані на схемі S100 і S300 – це так звані емульовані комутатори (або домени vPC+), тобто пари комутаторів, підключення до яких здійснюється за технологією Multi-Chassis PortChannel. Кожний з комутаторів у парі має також свій власний ідентифікатор FabricPath (скажемо, S101 і S102, S301 і S302), а емульований комутатор з погляду логічної топології FabricPath (як вона видна в IS-IS) виглядає як такий, що знаходиться «за ними»: S100 досяжний через S101 і S102, а S300 – через S301 і S302. Тим самим пристрої, підключені до правого за схемою парі через канали vPC+ (безпосередньо або, як на схемі, через FEX), виявляються досяжні з «рівною вартістю» через S301 і S302, що забезпечує розподіл навантаження між всіма оптимальними каналами.

На схемі також зазначені два варіанти підключення виносів FEX – наскрізне (Straight-Through) і EvPC. Оскільки винос FEX – це «продовження» головного пристрою, найбільш простий варіант – підключення FEX тільки до одному «материнському» комутатору. У цьому випадку все настроювання й комутація здійснюється на одному комутаторі, а для резервування на випадок його відмови від серверів організуються канали vPC до виносів FEX,

підключених до різних комутаторів у парі vPC. Такий варіант історично називається Straight-Through.

Альтернативний, більше складний варіант – підключення FEX відразу до двох головних комутаторів. У цьому випадку комутація відбувається відразу на обох за рахунок організації підключень vPC від FEX до пари комутаторів, а узгодження настроювань для портів FEX здійснюється, наприклад, за допомогою автоматичної синхронізації конфігурацій. У такій схемі можуть використовуватися підключення vPC і до серверів, що дозволяє говорити про дворівневий vPC – від комутаторів до FEX і від FEX до серверів. Дана схема одержала назву Enhanced VPC (EvPC).

Але повернемося до реалізацій технології TRILL. Вона покладена й в основу рішення Virtual Cluster Switching – «фабрики» Ethernet, розробленою компанією Brocade. Правда, замість протоколу IS-IS у рішенні компанії використовується протокол Fabric Shortest Path First (FSPF), запозичений з миру Fibre Channel. Як указують фахівці компанії, протокол FSPF дозволяє кожному комутатору VCS одночасно «бачити» всі вхідні в «фабрику» пристрою й вибирати маршрути з урахуванням стану всієї топології. Комутація трафіку між двома кінцевими пристроями у фабриці здійснюється в режимі балансування навантаження, при якому використовуються всі можливі еквівалентні шляхи з однаковими мінімальними вагами між кінцевими комутаторами. Фізичні канали Ethernet, що зв'язують два суміжних комутатори в Ethernet фабриці, автоматично поєднуються в одну логічну групу Brocade Fabric Trunk.

Восени 2014 року Brocade представила VDX 8770 – перший модульний комутатор у лінійці пристроїв VDX, призначених для побудови Ethernet-фабрик (до цього будівельникам таких «фабрик» були доступні тільки пристрої з фіксованою конфігурацією). Представлений комутатор істотно збільшує масштабованість і продуктивність Ethernet-фабрики, у якій може налічуватися більше 8000 портів. Як повідомляють у компанії, рішення VDX зараз застосовуються в більш ніж 700 інсталяцій по усьому світі.

3.3 Розробка функціональної схеми

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

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

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

Результатами аналізу повинні стати математичні залежності, що дозволяють обчислювати чисельні значення характеристик центру обробки даних за технологією TRILL з урахуванням особливостей конкретної структури центру обробки даних за технологією TRILL.

Запропоновано концептуальний підхід до аналізу функціональної структури центру обробки даних за технологією TRILL, заснований на дослідженні взаємодії додатків (завдань) як незалежних джерел і приймачів даних. У цьому випадку спочатку визначаються параметри потоків даних між

додатками при виконанні всього комплексу завдань (будується інформаційна модель), а потім, залежно від розміщення додатків по вузлах центру обробки даних за технологією TRILL й використовуваного устаткування, визначаються параметри потоків даних між вузлами центру обробки даних за технологією TRILL (будується технічна модель).

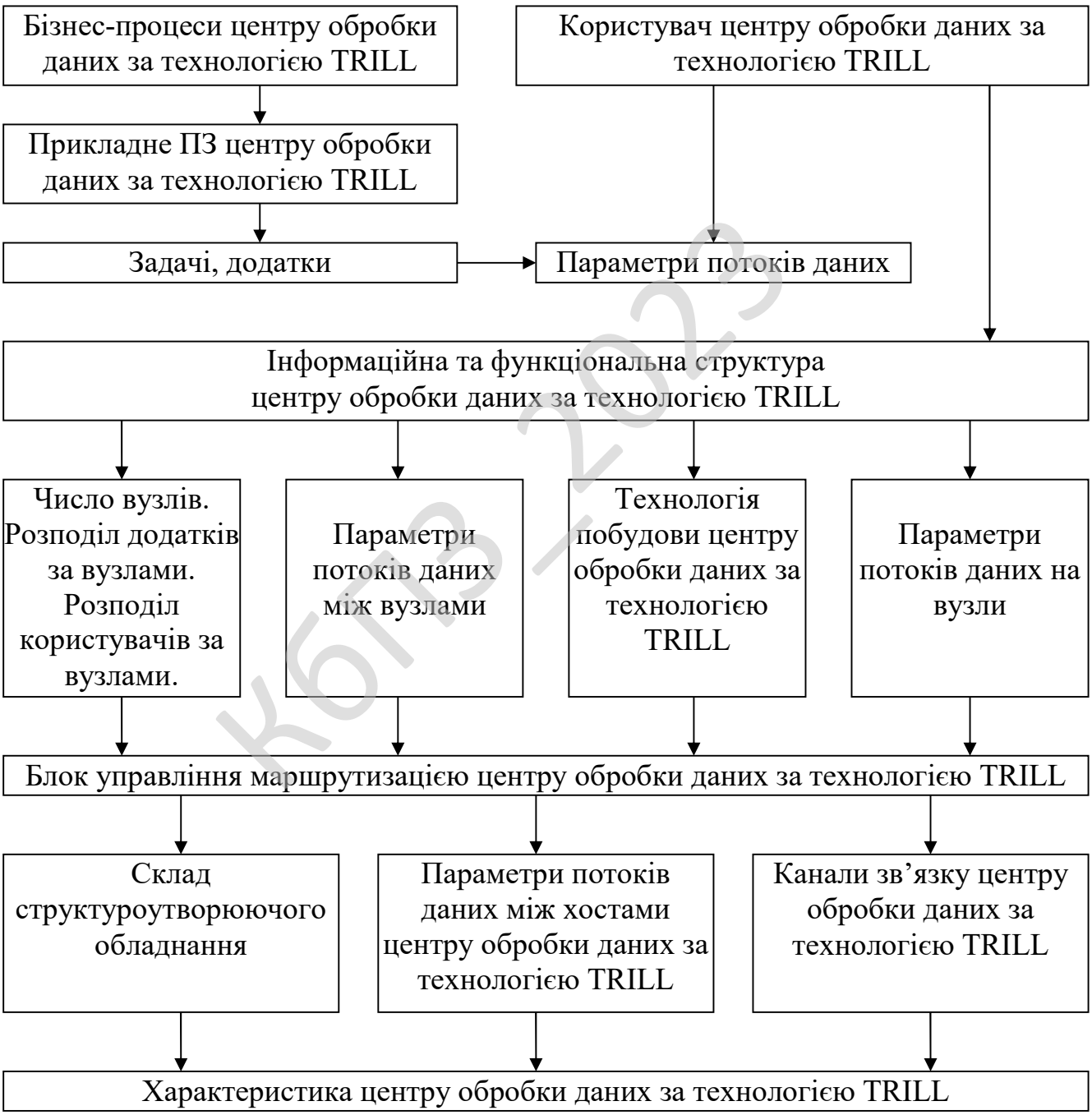

Рисунок 3.2 – Функціональна схема системи

При цьому не тільки повністю враховуються всі взаємодії між додатками, але й з'являється можливість проведення аналізу складних ієрархічних мережних структур, шляхом декомпозиції на підмережі центру обробки даних за технологією TRILL, що часто застосовується в технологіях VLAN і VPN. Даний підхід розвивається й застосовується в даній роботі.

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

Для аналізу функціональної структури й розрахунку характеристик центру обробки даних за технологією TRILL визначені правила її формального опису, що однозначно задають властивості додатків і структуру центру обробки даних за технологією TRILL, що дозволяють будувати моделі для проведення розрахунків.

Відповідно до концептуального підходу до проведення аналізу функціональної структури центру обробки даних за технологією TRILL виділені дві складові її структури інформаційна й технічна:

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

– Технічна структура – сукупність структуроутворюючого устаткування, технічних вузлів центру обробки даних за технологією TRILL й каналів зв'язку. Технічному вузлу можуть відповідати декілька інформаційних.

Таким чином, для повноцінного аналізу функціональної структури реальної центру обробки даних за технологією TRILL необхідно провести аналіз складових її інформаційної й технічної структур і зв'язати результати аналізу.

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

Інформаційна структура центру обробки даних за технологією TRILL задається набором параметрів:

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

де:

– N – число працюючих користувачів;

– M – число задіяних вузлів;

– D – число використовуваних додатків;

– L – число розв'язуваних завдань;

– R – число баз даних;

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

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

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

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

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

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

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

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

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

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

Набір однозначно визначає інформаційну структуру центру обробки даних за технологією TRILL.

Для заданого набору **SI** отримані результати, що дозволяють визначити параметри потоків даних між вузлами центру обробки даних за технологією TRILL.

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

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

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

Розроблено моделі для аналізу ієрархічної трьохрівневої інформаційної структури центру обробки даних за технологією TRILL, що дозволяють визначити завантаження каналів зв'язку й мережного устаткування в центру обробки даних за технологією TRILL, що складається із сукупності підмереж і корпоративних серверів.

Отримано наступні результати:

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

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

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

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

Результати аналізу інформаційної структури дозволяють визначати параметри потоків даних між логічними об'єднаннями вузлів центру обробки даних за технологією TRILL. Інформаційна структура реалізується конкретними технічними засобами й втілюється у вигляді технічної структури. При цьому можливо невідповідність інформаційної й технічної структур, пов'язане з технічними характеристиками й можливостями апаратури. Наприклад, на одному технічному вузлі можуть бути встановлені кілька інформаційних вузлів. У зв'язку із цим розроблені методи й засоби аналізу технічної структури центру обробки даних за технологією TRILL, створюваної на базі інформаційної структури.

Для аналізу роботи реальної центру обробки даних за технологією TRILL розроблений метод формального опису її технічної структури. Структура реальної центру обробки даних за технологією TRILL формується із застосуванням структуроутворюючого устаткування (комутатори, маршрутизатори), до якого підключаються вузли центру обробки даних за технологією TRILL, при цьому створюється багаторівнева (часто трьохрівнева) мережа. Такий підхід застосовується, як правило, при створенні центру обробки даних за технологією TRILL на базі технології VLAN. При цьому групи першого

рівня інформаційної структури становлять віртуальні локальні центру обробки даних за технологією TRILL. Другий і третій рівні інформаційної структури призначені для з'єднання цих мереж між собою.

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

Технічна структура центру обробки даних за технологією TRILL задається множиною **ST**, елементами якого є:

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

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

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

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

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

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

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

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

Для заданої множини $\mathbf{ST}$ отримані наступні результати:

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

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

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

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

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

Отримані також формули для обчислення параметрів потоків даних кожного завдання.

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

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

Таким чином, параметри потоків даних у центру обробки даних за технологією TRILL для заданої технічної структури визначаються множиною:

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

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

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

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

3.4 Розробка діаграми процесів

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

– Головне вікно ПЗ.

– Налаштування ПЗ.

– Налаштування підсистеми Transparent Interconnection of Lots of Links.

– Налаштування підсистеми статистики.

– Налаштування підсистеми доступу до устаткування мережі.

– Обробник помилок.

Рисунок 3.3 – Діаграма взаємодії процесів

– Управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Формування архітектури нейронної мережі.

– Перегляд сформованої нейронної мережі.

– Створення журналу роботи нейронної мережі.

– Моніторинг мережі.

– Формування звіту та збереження в БД.

– Перегляд параметрів потоків даних між хостами мережі.

– Перегляд параметрів потоків даних між вузлами.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

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.

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.

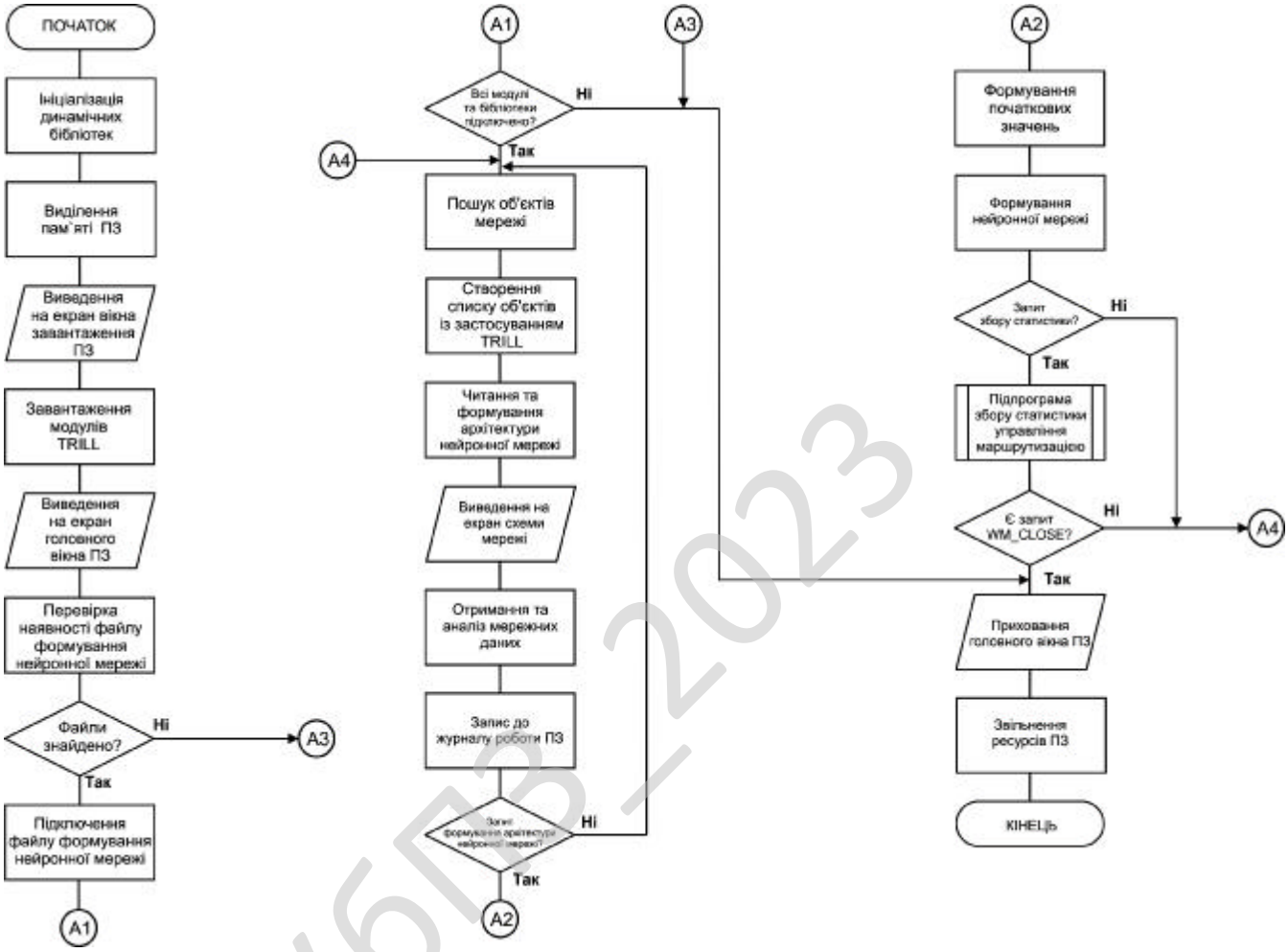

Рисунок 4.1 – Блок схема основної програми

Опис алгоритмів функціонування системи

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

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

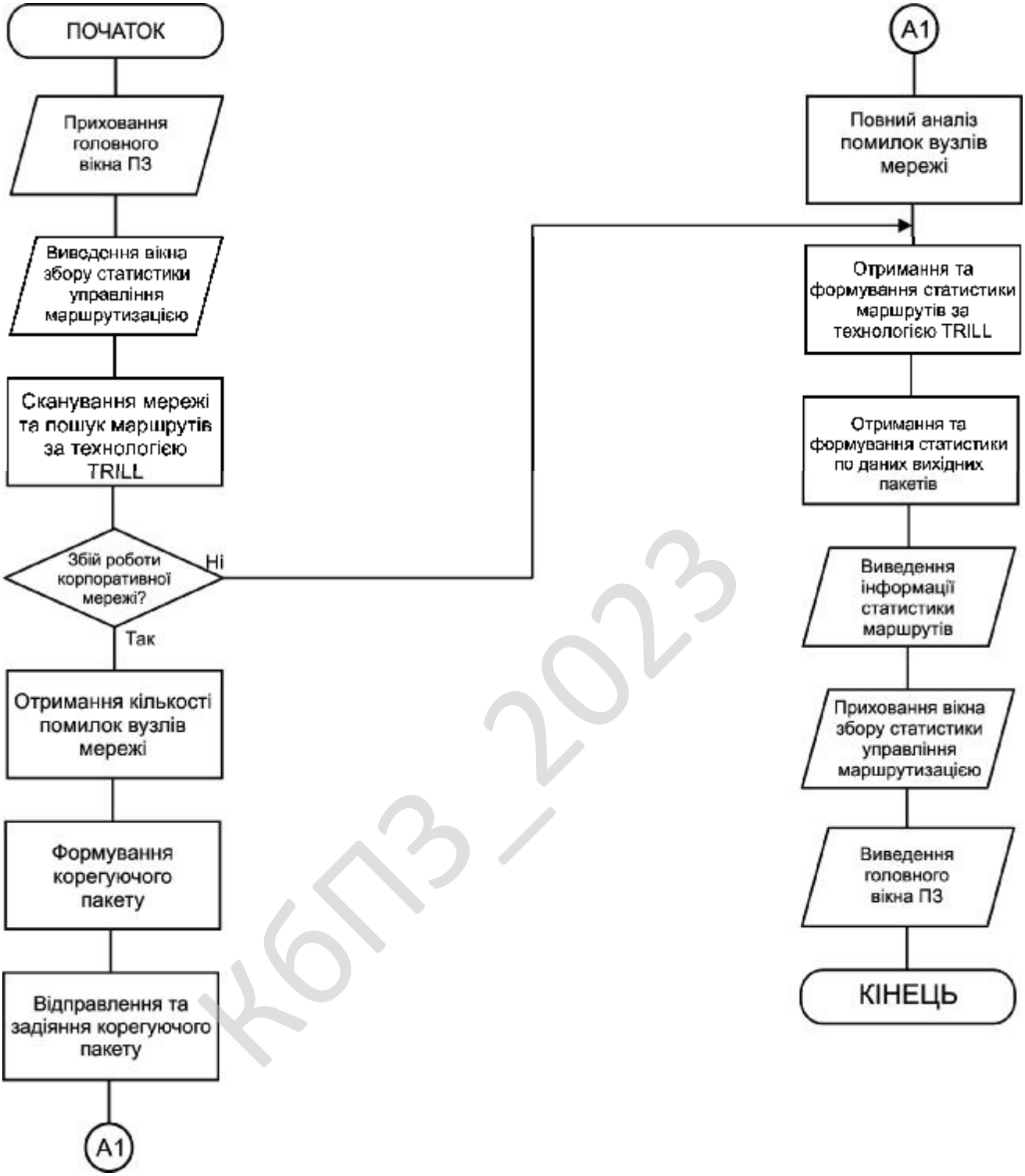

Рисунок 4.2 – Блок схема підпрограми

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача.

Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик.

Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

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

Розглянемо розроблені процедури та функції у дипломному проекті.

Завершення сесій. Для завершення відкритих сесій розроблена функція PZ_sessiondel. Оголошення функції для Windows 7/8/10:

```
Var // об’ява змінних ПЗ
PZ_sessiondel: function(ServerData, Uncclientname,
                                Username:Pwchar):DWORD;
```

Параметри:

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

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

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

```
procedure Tm.btnclosesessionclick(Sender: Tobject);
var // об’ява змінних ПЗ
OS: Boolean; Flibhandle : Thandle; Cnament: Pwidechar; Cname9x: Pansichar;
  Key:Smallint; i: Integer;
begin
  if not OS(OS) then Close; //З'ясовуємо тип системи
  if not Assigned(lvsessions.Selected) then Exit;
  i:= lvsessions.Selected.Index; //Визначаємо номер обраної сесії
  if OS then begin
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
@PZ_sessiondelnt := Getprocaddress(Flibhandle, 'PZ_sessiondel');
    if not Assigned(PZ_sessiondelnt) then
    begin Freelibrary(Flibhandle); Exit;
    end;
    //Перетворимо дані в необхідний вигляд
```

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

Розглянемо процедуру що надає інформацію про локальні ресурси й можливості їх контролю.

PZData – за допомогою цієї функції отримаємо дані про всі спільні ресурси. Оголошення функції для Windows 7/8/10:

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

Параметри:

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

– Lvl повинен містити ідентифікатор структури.

– Bufptr повинен містити адресу покажчика на масив структур.

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

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

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

– SHARE_INFO_0 – тільки Windows 7/8/10 SP0.

– SHARE_INFO_1 – тільки Windows 7/8/10 SP1.

– SHARE_INFO_1 – тільки Windows 9x – Me.

– SHARE_INFO_2 – тільки Windows 7/8/10 SP2.

– SHARE_INFO_50 – тільки Windows 9x – Me .

– SHARE_INFO_502 – тільки Windows 7/8/10 SP3.

Структура SHARE_INFO_50, оголошення структури:

```
type
 TSHAREINFO50 = packed record
  A_PZname : array [0..12] of Char;
  A_type:Byte;
  A_flags:Word;
  A_remark: Pchar;
  A_path:Pchar;
  A_rw_password:array [0..8] of Char;
A_ro_password: array [0..8] of Char;
end;
```

Розроблені поля:

– A_PZname містить рядок з мережним ім'ям ресурсу;

– A_type визначає тип ресурсу;

– A_flags містить інформацію про права доступу до ресурсу;

– A_remark покажчик на рядок необов'язкових коментарів;

– A_path містить локальне розташування ресурсу;

– A_rw_password містить пароль на запис – читання;

– A_ro_password містить пароль на читання.

Розглянемо розроблену функцію PZshareadd. Відкриття локального ресурсу. Оголошення функції для Windows 7/8/10:

```
var // об’ява змінних ПЗ
PZshareadd: function (ServerData: Pwidechar; level: DWORD; buf:Pointer;
parm_err: LPDWORD): DWORD;
```

Параметри:

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

– Lvl повинен містити ідентифікатор структури;

– buf повинен містити покажчик на структуру;

– parm_err містить покажчик помилки.

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

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

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

```
procedure Tm.btnaddsharesclick(Sender: Tobject);
const
  STYPE_DISKTREE = 0;  ACCESS_ALL = 258;  DDF_FULL = 258;
var // об’ява змінних ПЗ
  Flibhandle : Thandle;   Share9x : Tshareinfo50;  Sharent : Tshareinfo2;
  Tmpdir, Tmpname: String; Tmpdirnt, Tmpnament: Pwchar; OS: Boolean;
  Tmplength: Integer;
begin
  Tmpdir := Selectdirectory; //Визначаємо шлях до майбутнього ресурсу
```

```
//Визначаємо мережне ім'я
  Tmpname := Inputbox('Share name','Enter name','Test');
  if Tmpdir = '' then Exit;
  if not OS(OS) then Close; // З'ясовуємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
@PZshareaddnt:=Getprocaddress(Flibhandle,'PZshareadd');
    if not Assigned(PZshareaddnt) then
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Tmplength := Sizeof(Widechar)*256; //Визначаємо необхідний розмір
    Getmem(Tmpnament, Tmplength); //Конвертуємо в Pwchar
    Stringtowidechar(Tmpname, Tmpnament, Tmplength);
    Sharent.SS2_PZname := Tmpnament; //Ім'я
    Sharent.SS2_type := STYPE_DISKTREE; //Тип ресурсу
    Sharent.SS2_remark := ''; //Коментар
    Sharent.SS2_permissions := ACCESS_READ; //Доступ
    //Кількість максимальних підключень
    Sharent.SS2_max_uses := DWORD( - 1);
    //Кількість поточних підключень
    Sharent.SS2_current_uses := 0;
    Getmem(Tmpdirnt, Tmplength);
    Stringtowidechar(Tmpdir, Tmpdirnt, Tmplength);
    Sharent.SS2_path := Tmpdirnt; //Шлях до ресурсу
    Sharent.SS2_passwd := nil; //Пароль
    PZshareaddnt(nil,2,@Sharent,nil); //Додаємо ресурс
    Freemem (Tmpnament); //звільняємо пам'ять
    Freemem (Tmpdirnt);
  end else begin //Код для 9x
    Flibhandle := Loadlibrary('PZ_2.DLL');
    if Flibhandle = 0 then Exit;
    @PZshareadd := Getprocaddress(Flibhandle,'PZshareadd');
    if not Assigned(PZshareadd) then
    begin
      Freelibrary(Flibhandle);
      Close;
    end;
    Fillchar(Share9x.A_PZname, Sizeof(Share9x.A_PZname), #0);
    move(Tmpname[1],Share9x.A_PZname[0],Length(Tmpname)); //Ім'я
```

```
    Share9x.A_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.A_flags := DDF_RDONLY; //Доступ
    Fillchar(Share9x.A_remark,
      Sizeof(Share9x.A_remark), #0); //Коментар
    Fillchar(Share9x.A_path,
      Sizeof(Share9x.A_path), #0);
    Share9x.A_path := Pansichar(Tmpdir); //Шлях до ресурсу
    Fillchar(Share9x.A_rw_password,
      Sizeof(Share9x.A_rw_password), #0);
//Пароль повного доступу
    Fillchar(Share9x.A_ro_password,
    Sizeof(Share9x.A_ro_password), #0); //Пароль для читання
    PZshareadd(nil,50,@Share9x,Sizeof(Share9x));
  end;
  Freelibrary(Flibhandle);
end;
```

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

Даний код відкриває новий ресурс на повний доступ.

Одержання списку поточних сесій. Для визначення користувачів підключених до комп'ютера була розроблена функція PZ_session. Оголошення функції для Windows 7/8/10:

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

Параметри:

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

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

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

– Lvl – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адреса покажчика на масив структур.

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

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

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

```
function Tm.Cardinaltotimestr(Value:Cardinal):String;
var // об’ява змінних ПЗ
 d,h,m,s: Real;
begin
    d:=0; h:=0; m:=0; s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:=s - (m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:=m - (h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:=h - (d*24);
    end;
    Result:='';
    if (d>0) then Result:=Result+floattostr(d)+' d. ';
    if (h<9) then Result:=Result+'0'+floattostr(h)+':' else
Result:=Result+floattostr(h)+':';
    if (m<9) then Result:=Result+'0'+floattostr(m)+':' else
Result:=Result+floattostr(m)+':';
    if (s<9) then Result:=Result+'0'+floattostr(s)     else
Result:=Result+floattostr(s);
end;
```

Реалізовано в дипломному проекті – оголошення структури Tshareinfo2Array = array [0..512] of Tshareinfo2; Тут пам'ять уже виділена.

Розглянемо розроблену функцію PZsharedel яка дозволить закрити обраний загальний ресурс.

```
Var // об’ява змінних ПЗ
 PZsharedel:function (pszserver, pszPZname:Pchar;
                              usreserved:Word):DWORD;
```

Параметри:

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

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

Обидві функції не використовують ніяких структур. У якості імені передається не шлях до ресурсу, а саме ім'я ресурсу яке я визначив за допомогою коду (розглянутого вище). У випадку успішного виконання функцій, їх результат буде дорівнює нулю.

```
procedure Tm.btnclosesharesclick(Sender: Tobject);
var // об’ява змінних ПЗ
  OS:Boolean;
  Flibhandle : Thandle;
  Name9x:array [0..12] of Char;
  Nament:Pwchar;
  i:Integer;
  Sharename: String;
begin
  if not OS(OS) then Close; //Визначаємо тип системи
  if lbxshares.Items.Count = 0 then Exit;
  for i:= 0 to lbxshares.Items.Count - 1 do
    if lbxshares.Selected[i] then Break;  //Шукаємо обраний елемент
//Якщо не знайдений завершення роботи  ПЗ
  if not lbxshares.Selected[i] then Exit;
  Sharename := lbxshares.Items.Strings[i];
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('PZ_1.DLL');
    if Flibhandle = 0 then Exit;
    @PZsharedelnt := Getprocaddress(Flibhandle,'PZsharedel');
    if not Assigned(PZsharedelnt) 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
```

```
        PZsharedelnt(nil,Nament,0);  //Видаляємо ресурс
        Freemem(Nament); //Звільняємо пам'ять
      end else begin
        Flibhandle := Loadlibrary('PZ_2.DLL');
        if Flibhandle = 0 then Exit;
        @PZsharedel := Getprocaddress(Flibhandle,'PZsharedel');
        if not Assigned(PZsharedel) then //Перевірка
        begin
          Freelibrary(Flibhandle);
          Exit;
        end;
        Fillchar(Name9x, Sizeof(Name9x), #0); //Очищаємо масив
        move(Sharename[1],Name9x[0],Length(Sharename));
    //Заповнюємо масив
        PZsharedel(nil,@Name9x,0); //Видаляємо ресурс
      end;
      Freelibrary(Flibhandle);
    end;
```

Розглянемо розроблену функцію, яка визначає тип системи.

```
function Tm.OS(var Value: Boolean): Boolean;
    var // об’ява змінних ПЗ
     Ver: Tosversioninfo;
        Bres: Boolean;
    begin
      Ver.dwosversioninfosize := Sizeof(Tosversioninfo);
      Bres := Getversionex(Ver);
      if not Bres then
      begin
        Result := False;
        Exit;
      end else
        Result := True;
      case Ver.dwplatformid of
        VER_PLATFORM_WIN32_XP: Value := True;
    //Windows 7/8/10 -  підходить
        VER_PLATFORM_WIN32_WINDOWS : Value := False;
    //Windows 9x - Me - не підходить
        VER_PLATFORM_WIN32s : Result := False;
    //Windows 3.x -  не підходить
      end;
    end;
```

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

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

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

```
procedure Tm.btngetsharesclick(Sender: Tobject);
var // об’ява змінних ПЗ
  i:Integer;
  Flibhandle : Thandle;
  Sharent : Pshareinfo2Array;
  entriesread,totalentries:DWORD;
  Share : array [0..512] of Tshareinfo50;
  pcentriesread,pctotalavail:Word;
  OS: Boolean;
begin
  lbxshares.Items.Clear;
  if not OS(OS) then Close; //Визначаємо тип системи
  if OS then begin //Код для XP
    Flibhandle := Loadlibrary('_R1.DLL'); //Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
//Зв'язуємо функцію
    @PZDatant := Getprocaddress(Flibhandle,'PZData');
    if not Assigned(PZDatant) then //Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    Sharent := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if PZDatant(nil,2,@Sharent,DWORD( – 1),
             @entriesread,@totalentries,nil) <> 0 then
    begin //Якщо виклик невдалий вивантажуємо бібліотеку
      Freelibrary(Flibhandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
```

```
    for i:= 0 to entriesread -  1 do
            lbxshares.Items.Add(String(Sharent[i].SS2_PZname));
  end else begin
    Flibhandle := Loadlibrary('PZ_2.DLL');
//Завантажуємо бібліотеку
    if Flibhandle = 0 then Exit;
    //Зв'язуємо функцію
    @PZData := Getprocaddress(Flibhandle,'PZData');
    if not Assigned(PZData) then
//Перевірка
    begin
      Freelibrary(Flibhandle);
      Exit;
    end;
    if PZData(nil,50,@Share,Sizeof(Share),
            @pcentriesread,@pctotalavail) <> 0 then //Виклик функції
    begin //Якщо виклик невдалий вивантажуємо бібліотеку
      Freelibrary(Flibhandle);
      Exit;
    end;
    if pcentriesread > 0 then //Обробка результатів
    for i:= 0 to pcentriesread -  1 do
         lbxshares.Items.Add(String(Share[i].A_PZname));
  end;
  Freelibrary(Flibhandle); // вивантажити бібліотеку
end;
```

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

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації.

Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи: Поточний стан системи; Апаратне обрання мережної карти; Поле відображення IP адреси та порту; Меню (Налаштування, Довідка, Автор розробки); Функціонал. Який складається з: Параметри потоків даних; Побудова карти корпоративної мережі; Пошук об'єктів корпоративної мережі; Статистика; База даних статичної інформації корпоративної мережі.

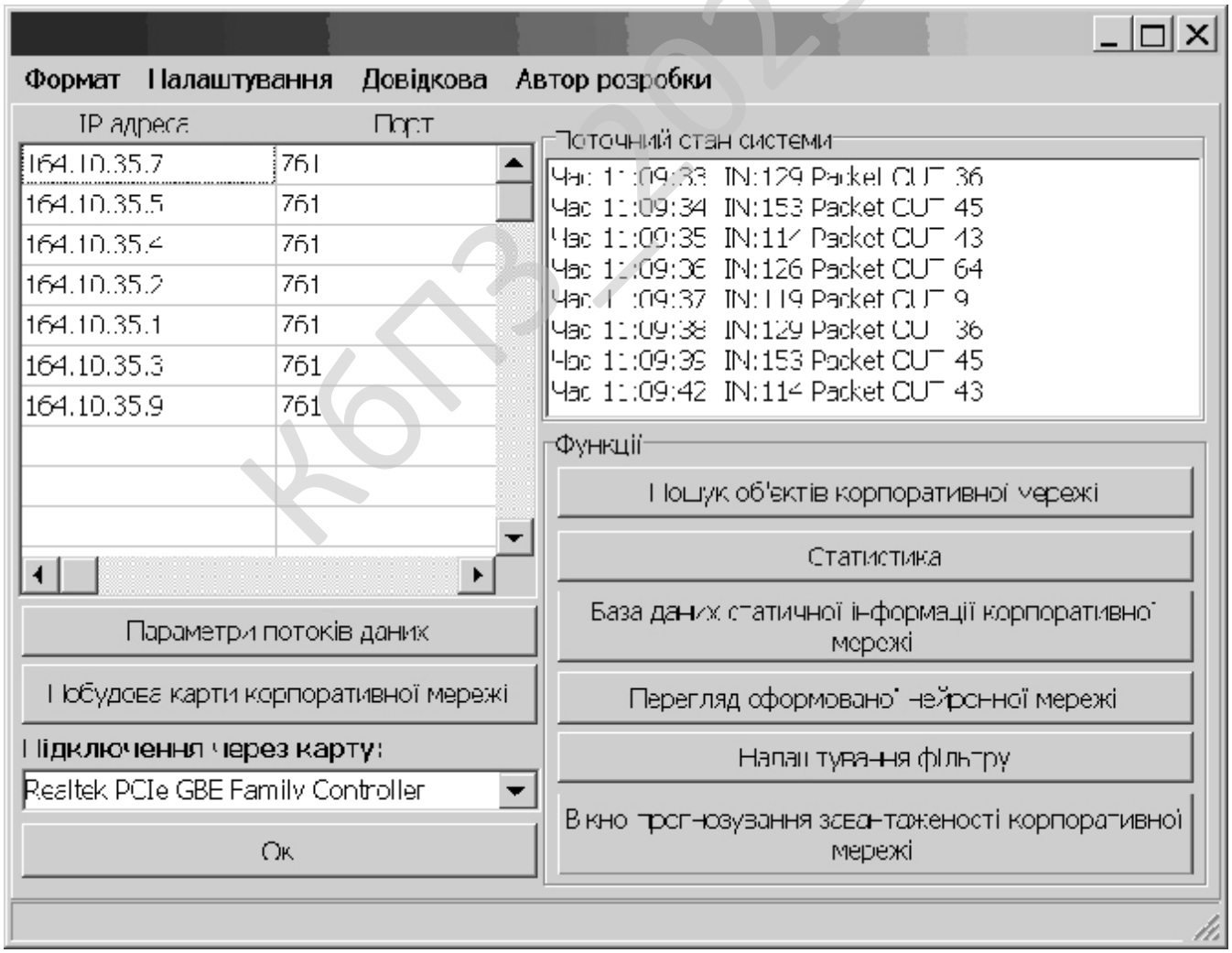

Рисунок 5.1 – Головне вікно програми

Центр обробки даних представляється у вигляді корпоративної мережі.

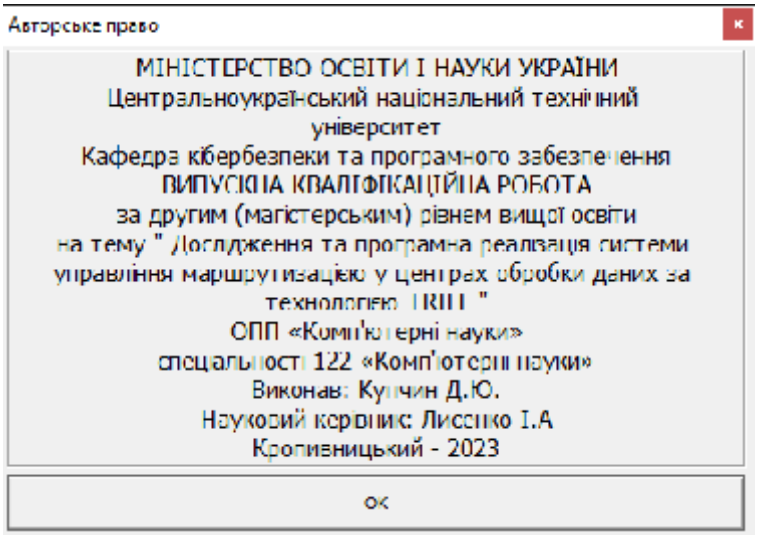

Рисунок 5.2 – Авторське право

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Метою розробки є дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Об'єктом дослідження є процес управління маршрутизацією у центрах обробки даних за технологією TRILL.

Предметом дослідження є методи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Розроблено вітчизняний продукт управління маршрутизацією у центрах обробки даних за технологією TRILL, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	99
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	100000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

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

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

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

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 33 = 56 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	56	Ф 7.1-7.4
Впровадження	13	Д13
Всього	97	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{97 \cdot 1}{60 - 5} = 1{,}8 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	5	450	7,5
Монітор	60	5	300	5
Клавіатура	30	5	150	2,5
Маніпулятор «мишка»	30	5	150	2,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	32,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{32,49 \cdot 3}{1,2} = 81,23 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 81,23/(60 \cdot 8) = 0,2 \text{ ставки.}$$

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

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

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

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

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13440	40320
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	1,8	14000	75600
Інженер - електронщик	0,2	12000	7200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$ = 5,25	-	$Ф_{роб}$ = 204120

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{204120}{5,25 \cdot 60} = 648 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 29000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 29000 = 1858000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 185800 грн.

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Суперкомп за 07.11.23 – джерело https://supercomp.kiev.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	INTEL Celeron G5905 (BX80701G5905 1200, 2 ядра, 2 потоки, 3.5 GHz, TDP - 58 Вт 14nm BOX	-
Системна плата	ASUS PRIME H510M-K сокет - 1200, DDR4 3200 MHz, LAN - 1 Гбіт/с, D-Sub (VGA) HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s Micro-ATX	-
Відеокарта	Intel UHD Graphics 610	-
Жорсткий диск	SSD M.2 2280 512GB LEVEN (JP600PCIE512GB) Серія - JP600, 512 GB 3D TLC NAND, M.2, PCI Express 3.0 x4	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Fury Beast Blac Kingston Fury (ex.HyperX) (KF432C16BB/8)	-
Блок живлення	Gamemax 500W (GM-500B) ATX 12V v2.3 500 Вт, 20+4 pin, CPU - 4+4pin, GPU - 1x pin, SATA - 3, Peripheral - 2, +12V1 - 20A 1x120 мм, 150 x 140 x 86 мм	-
Корпус	Vinga CS210B, Miditower, ATX, Micro ATX, Mini - ITX	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" LG 22MP58VQ-P 5 мс IPS 1920x108 250/1000M:1 178/178 D-Sub+HDMI+DVI	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

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

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

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1858000	-	-
2. Передавальні пристрої	185800	-	-
Всього по групі	2043800	5	102190
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
4. Нематеріальні активи	100000	10	10000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	37857,75
Разом	$К_р$ = 2523008		$А_р$ = 249636,25

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 648 \cdot 97/99 = 635 \text{ грн.}$$

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

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

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

$$З_д = 635 \cdot 10 \cdot 0{,}01 = 64 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

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

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

$$С_{оц} = 0{,}01 \cdot 22(635+64) = 154 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 635 \cdot 15 \cdot 0{,}01 = 95 \text{ грн.}$$

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

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

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

Згідно виданих норм приймаємо 1/6 пачку паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 206 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

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

$$З_{М1} = 206 \cdot 3 \cdot 1/6 = 309 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 45 примірників:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 27 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 27 грн./шт.

$$З_{М2} = 27 \cdot 45 = 1215 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (309 + 1215 + 1702)/99 = 33 \text{ грн.}$$

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

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

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

$$О_п = 635 \cdot 15 \cdot 0{,}01 = 95 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 99 прим.):

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

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

$$A_м = 249636 \cdot 3/(99 \cdot 12) = 631 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	635
2. Додаткова зарплата виконавців	$З_д$	64
3. Відрахування на соціальні потреби	$С_{оц}$	154
4. Загальногосподарські витрати	$Г_{осп}$	95
5. Витрати на матеріали	$З_М$	33
6. Освоєння нових операційних систем, мов програмування	$О_п$	95
7. Амортизація основних фондів	$А_м$	631
8. Повна собівартість програмного забезпечення	$С_п$	1707
9. Плановий прибуток	$П_р$	940
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2647
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	529,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3176,4

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

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

$$C_п = 635+64+154+95+33+95+631 = 1707 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 55 \cdot 1707 = 940 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3176
Всього капітальних витрат	–	3176

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	32208	20130
2. Витрати на електроенергію	$З_{ел}$	410	257
3. Витрати на амортизацію	$З_{ам}$	0	794
Всього витрат за рік	$І$	32618	21181

Витрати на обслуговування системи:

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

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

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 32208 \text{ грн},$$

до:

$$З_{р\,нов} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 20130 \text{ грн.}$$

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

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

$$З_{ел\,баз} = 0{,}45 \cdot 240 \cdot 3{,}8 = 410 \text{ грн.}$$

$$З_{ел\,нов} = 0{,}45 \cdot 150 \cdot 3{,}8 = 257 \text{ грн.}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3176	–	794
Всього відрахувань	-	–	3176	–	794

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (2647\text{-}1707) \cdot 99\text{-}(0{,}05 \cdot 2043800 + 0{,}5 \cdot 199177 + 0{,}25 \cdot 37031 +$$
$$+0{,}1 \cdot 100000 + 0{,}2 \cdot 143000) \cdot 3/12 = 30651 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{479208}{(2647\text{-}1707)\cdot 99\cdot 12/3} = 1{,}3 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	99
2. Повна собівартість розробленої програми	Грн.	1707
3. Ціна розробленої програми	Грн.	2647
4. Плановий прибуток від реалізації розробленої програми	Грн.	940
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2523008
7. Загальний прибуток від реалізації програмної продукції	Грн.	93060
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	30651
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,3
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3176
11.Величина економічного ефекту у користувача програмної продукції	Грн.	10643
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,3

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

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

$$E_{сп} = (32618\text{-}21181)\text{-}0{,}25 \cdot 3176 = 10643 \text{ грн.}$$

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

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

$$T_{сп} = \frac{3176}{32618 - 21181} = 0{,}3 \text{ року}.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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

Ці шкідливі фактори можуть привести до професійних захворювань.

Керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18 розглянемо шкідливі чинники роботи персоналу.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

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

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

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м'язовий апарат;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці.

Результати досліджень показали, що найбільшою мірою негативний фізіологічний вплив на операторів ПК пов'язано з дискомфортними зоровими умовами через неправильно спроектованого освітлення: пряма і відбита від екранів бляклість, несприятливий розподіл яскравості в полі зору, невірна орієнтація робочого місця щодо світлоприймачів.

Розташовувати обладнане дисплеєм робоче місце необхідно таким чином, щоб в поле зору оператора не потрапляли вікна або освітлювальні прилади. Вони не повинні перебувати і безпосередньо за спиною оператора. Слід домагатися зменшення відображень на екрані від різних джерел штучного і денного світла. Коли штучне світло змішується з природним, рекомендується використовувати лампи, по спектрального складу найбільш близькі до сонячного світла. Співвідношення яскравості екрана і безпосередніх найближчого оточення не повинно перевищувати 3: 1.

Оптимальні значення температури повітря в приміщенні повинні бути 19-23 С. Швидкість руху повітря не більше 0,1 м / с. Рекомендована відносна вологість повітря 55%.

8.3 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів. З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту. До засобів гасіння пожежі, призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в який встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно оснащувати переносними вуглекислотними з розрахунку 2 шт. на кожні 20 $м^2$ в приміщеннях. Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

Електроустановки (можливість їх застосування, монтаж, накладка експлуатація) повинні відповідати вимогам чинних правил улаштування електроустановок, правил технічної експлуатації, електроустановок та інших нормативних документів.

Ймовірність виникнення пожежі від електротехнічного та іншого одиничного виробу не повинна перевищувати 10^{-6} на рік. При короткому замиканні в місцях з'єднання проводів опір практично дорівнює нулю, звідси величина струму досягає дуже великих значень.

Персональні комп'ютери після закінчення роботи повинні відключатися від мережі не рідше 1 разу на квартал, необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами. Не дозволяється розміщувати комп'ютерні зали ЕОМ у підвалах; проводити ремонт вузлів (блоків) ЕОМ безпосередньо у залах, де знаходяться ПК (персональні комп'ютери), залишати без нагляду ввімкнену в мережу електронну апаратуру, яка використовується для контролю ЕОМ.

Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і

незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин електроустаткування або порушення заземлення треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

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

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахунок занулення

Розрахунок занулення складається з трьох частин:

1. Розрахунок на відключаючу спроможність;

2. Визначєння максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус;

3. Розрахунок робочого і повторного заземлювачів

Початкові дані

1. Потужність електроприладів, які підлягають зануленню:

$$P = 7 \text{ кВт.}$$

2. Довжина магістрального кабеля: L_M=45 м.

3. Довжина розгалудження (від розподільчого щита до електроприладів): l=25 м.

4. Материал провідників кабеля−алюміній.

5. Лінійна напруга U=380 В.

6. Фазна напруга Uф=220 В.

Розрахунок проводиться відповідно до схеми електромережі, яка зображенана рис. 8.1.

У результаті розрахунку отримали:

– номінальна силу струму апарата захисту 33 А;

– найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання 100 А;

– площа перерізу магістрального кабеля (провідника) 9 мм^{2};

– максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус 31,2 В.

«Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні

правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

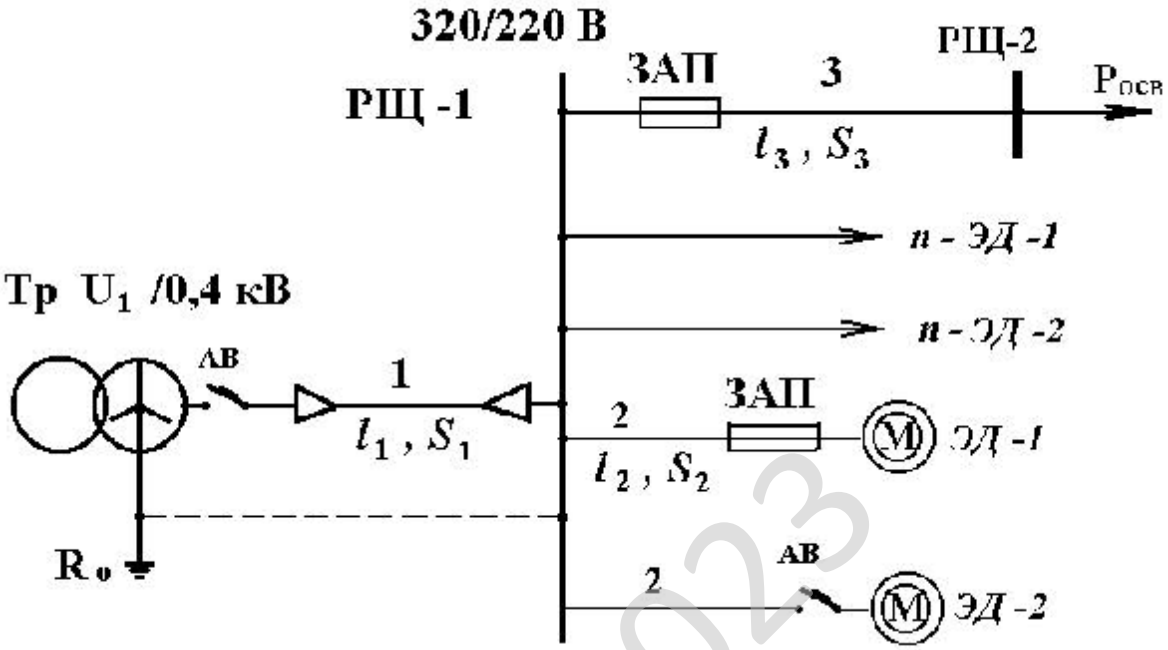

Рисунок 8.1 – Схема електромережі

Тр – трансформатор; РЩ – розподільчий щит; АВ – автоматичний вимикач; ЗАП – запобіжник; 1 – магістральний кабель; 2 – розгалуження до електроприладів.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів. Ці шкідливі фактори можуть привести до професійних захворювань.

Керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18 розглянемо шкідливі чинники роботи персоналу.

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

Список використаних джерел інформації

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

3. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Мін-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с. URI: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів управління маршрутизацією у центрах обробки даних за технологією TRILL.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління маршрутизацією у центрах обробки даних за технологією TRILL.

– Досліджена система управління маршрутизацією у центрах обробки даних за технологією TRILL.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління маршрутизацією у центрах обробки даних за технологією TRILL.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 10643 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,3 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Купчин Д.Ю. Дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

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. 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.

10. 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.

11. 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.

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

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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).

34. 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

35. 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.

36. 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.

37. 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.

38. 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.

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

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

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

42. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

43. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

44. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

45. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

46. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

47. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

48. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

49. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів

вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

50. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.0925 «Автоматизація й комп'ютерно-інтегровані технології». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 1.12.2011 року № 1/11-11258. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи управління маршрутизацією у центрах обробки даних за технологією TRILL;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 94 аркуші.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 14.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Лисенко І.А

Дослідження та програмна реалізація системи управління маршрутизацією у центрах обробки даних за технологією TRILL

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 68

Літера: РП

Кропивницький – 2023 року

Файл DataIP.pas- функції роботи з IP-протоколом

```
unit DataIP;

{
Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Кафедра кібербезпеки та програмного забезпечення
Тема: Дослідження та програмна реалізація системи управління маршрутизацією у
центрах обробки даних за технологією TRILL
Виконав: студент 2 курсу магістратури, групи КН-22Мз
Купчин Дмитро Юрійович
2023 рік
Керівник: Лисенко І.А.
}

interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

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

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


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

```
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),
{ Протокол Simple Netw. Management Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
    );



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

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


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

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

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

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

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

type

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

```
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для DAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


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

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

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

```
implementation

var
  RecentIPs     : TStringList;

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

{ отримання наступного "токена" з рядка }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

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

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

//-----------------------------------
{ перетворення крапкової десяткової IP-адреси в мережний байт типу DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
```

```
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;

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

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

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

//-----------------------------------
{ головна частина, (DATA)}

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

```
end ;

//-------------------------------------//
function DNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadD then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                 // дані
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if NetworkParams.DnsServerNames [0] <> ' ' then
                                  NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                   PDnsServer^.IPAddress ;  // дані
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;
end;

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

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


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

// включення байтів у/з для кожного адаптера

function DIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
```

```
  if NOT LoadD then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам’ яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

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

```
end ;

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

//-----------------------------------
{ інформація про інстальовані адаптери }

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

        // беремо список IP адрес та конвертуємо в IPAddressList
```

```
        I := 0 ;
        PIpAddr := @AdapterInfo^.IPAddressList ;
        while (PIpAddr <> Nil) do
        begin
            AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
            AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
            PIpAddr := PIpAddr.Next ;
            inc (I) ;
            if Length (AdpRows [AdpTot].IPAddressList) <= I then
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

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

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

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

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

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

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

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

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

//-----------------------------------
{ моніторимо час доступу до IP центрів обробки даних за технологією TRILL
(DATA)}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
```

```
  end
  else
    Result := NO_ERROR;
end;

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

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


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
```

```
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

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

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

//-----------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadD then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
```

```
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі : '  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мінімальний час виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу        : '  + IntToStr( dwRTOMax ) +
'  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення            : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних       : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

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

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

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

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

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
```

```
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' немає даних.'  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

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

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

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

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації центрів обробки даних за технологією
TRILL (DATA); }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

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

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

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

```
//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadD then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена    : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос          : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана     : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

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

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
```

```
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)     : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів         : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)     : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

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

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                : '  + IntToStr( dwErrors ) );
  ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' 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- обробка з'єднань TCP/IP центрів обробки даних за технологією TRILL

```
unit TCP_IP;

interface

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

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

var
  Form2: TForm2;

implementation

{$R *.dfm}

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

procedure TForm2.DOIpStuff;
begin

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

end;

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

procedure TForm2.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
```

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

end;

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

procedure TForm2.FormCreate(Sender: TObject);
begin

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

end.
```

Файл Main.pas – основне тіло розробленого ПЗ

```
unit Main; // назва модулю
interface

// опис бібліотек

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

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
```

```
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


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

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

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

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
   Sesi50_username    : PChar;
   sesi50_key         : Cardinal;
   sesi50_num_conns   : Word;
   sesi50_num_opens   : Word;
```

```
   sesi50_time        : Cardinal;
   sesi50_idle_time   : Cardinal;
   sesi50_protocol    : Byte;
   pad1               : Byte;
 end;

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

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

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

type
 TMibIfTable = packed record
   dwNumEntries     : DWORD;
   Table            : TMibIfArray;
 end;
 PMibIfTable = ^TMibIfTable;
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
```

```
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

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

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

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

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

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

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

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

var
NetSessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

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

var
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
```

```
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

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

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

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

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

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

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

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

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

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів центрів обробки даних за
технологією TRILL
```

```
//

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

  if OS then begin
//Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
//Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil;
//Очищаємо покажчик на масив структур
//Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin
//Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then
//Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
//Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
//Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle);
//Не забуваємо вивантажити бібліотеку
end;

//  Закриття загального ресурсу
```

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

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

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

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
```

```
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
end;

//
//  Додавання загального ресурсу
//

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

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

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

    GetMem(TmpNameNT, TmpLength);
//Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT;
//Ім' я

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

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

    ShareNT.shi2_passwd := ' ' ; //Пароль

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

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

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
```

```
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

{
  Одержання списку сесій центрів обробки даних за технологією TRILL
}

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 центрів обробки даних за технологією TRILL
        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 центрів обробки даних за технологією TRILL
```

```
}

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 центрів обробки даних за технологією
TRILL
}

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

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

```
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

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


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

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

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                               @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
```

```
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
       Exit;
       else ShowMessage(Помилка!' );
       Result:=1;
       Exit;
      end;
     end;
    for i:=1 to EntriesToGet do
     begin
      NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
      if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
      then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
      Application.ProcessMessages;
     end;
  end;
end;

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

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

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
```

```
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

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

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

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

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

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

end.
```

Файл About.pas – довідкової інформації

```
unit About;

interface

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

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

var
  Form1: TForm1;

implementation

{$R *.dfm}

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

end.
```

Файл data.pas- статистика роботи корпоративної мережі

```
unit Stat;

interface

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

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

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

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

end;

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

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

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

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

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

end.
```

Файл DAPI.pas - обробка API функцій розробленої системи

```
unit DAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

{ структура мережного інтерфейсу центрів обробки даних за технологією TRILL
(DATA)
}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

var

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

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

```
        DWORD; stdcall;

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

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

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

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

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

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

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

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

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

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

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

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

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


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

    function LoadD: Boolean;

implementation

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

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

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

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

end.
```

Файл DataIP.pas- функції роботи з IP-протоколом

```
unit DataIP;


interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

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

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


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

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

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


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

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

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

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

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

type

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

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

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

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

  TAdaptorRows = array of TAdaptorInfo ;


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

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

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

implementation

var
  RecentIPs     : TStringList;

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

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

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

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

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

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

end;

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

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

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

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

//-----------------------------------
{ голона частина, фіксація мережний параметрів центрів обробки даних за
технологією TRILL (DATA)}

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

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

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

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

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


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

// включення байтів у/з для кожного адаптера

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

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

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

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

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

```
{ інформація про інстальовані адаптери }

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

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

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

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

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

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

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

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

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

```
end ;

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

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

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

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

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


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

```
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

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

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

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

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

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

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

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

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

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

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

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

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

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

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації центрів обробки даних за технологією
TRILL (DATA); }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- обробка з'єднань TCP/IP центрів обробки даних за технологією TRILL

```
unit TCP_IP;

interface

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

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

var
  Form2: TForm2;

implementation

{$R *.dfm}

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

procedure TForm2.DOIpStuff;
begin

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

end;

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

procedure TForm2.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
```

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

end;

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

procedure TForm2.FormCreate(Sender: TObject);
begin

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

end.
```