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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Фролов Б.В.
«____» _____ 2021 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Олександр ДОРЕНСЬКИЙ
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

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

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

Фролову Богдану Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД*

2. Керівник роботи *Доренський Олександр Павлович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Фролов Б.В. Дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Legrand Cabling System 3 для ЦОД.

Метою розробки є дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД.

Об'єктом дослідження є процес Legrand Cabling System 3 для ЦОД.

Предметом дослідження є методи Legrand Cabling System 3 для ЦОД.

Методи дослідження базуються на методах теорії побудови комп'ютерних систем та мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи Legrand Cabling System 3 для ЦОД.

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

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

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

Програму розроблено в середовищі Delphi 10.3.2.

Ключові слова: комп'ютерна інженерія, Legrand Cabling System 3, ЦОД

ABSTRACT

Frolov B.V. Research and software implementation of Legrand Cabling System 3 for data centers. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work for the second (master's) level of higher education, software has been developed for the Legrand Cabling System 3 for data centers.

The purpose of development is research and software implementation of Legrand Cabling System 3 for data centers.

The object of research is the process of Legrand Cabling System 3 for data centers.

The subject of the study is the methods of Legrand Cabling System 3 for data centers.

The research methods are based on the methods of the theory of construction of computer systems and networks, methods of mathematical statistics, methods of software development.

The result is a software implementation of the Legrand Cabling System 3 for data centers.

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

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

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

The program is developed in the Delphi 10.3.2 environment.

Keywords: computer engineering, Legrand Cabling System 3, data center

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи 6
1.2 Область застосування 7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 8
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 8
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 12
2.3 Розгорнута постановка завдання 16
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 18
3.1 Опис функціонування системи 18
3.2 Розробка структурної схеми 40
3.3 Розробка функціональної схеми 50
3.4 Розробка діаграми процесів 53
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 55
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 55
4.2 Захист розробленого програмного забезпечення 71
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 73
6 НАУКОВА НОВИЗНА 79

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 80
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. 80
7.2 Розрахунок трудомісткості розробки програмної продукції 82
7.3 Визначення чисельності виконавців і планового фонду зарплати 84
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 89
7.5 Визначення собівартості розробки та ціни програмної продукції. 93
7.6 Визначення об'єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 96
7.7 Визначення експлуатаційних витрат..... 97
7.8 Визначення економічної ефективності програмної продукції..... 98
7.9 Висновок. 100
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 101
8.1 Вступ 101
8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером 102
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста . 103
8.4 Розробка заходів з умов поліпшення охорони праці..... 104
8.5 Розрахункова частина 106
8.6 Висновки до розділу..... 109
9 ОСНОВНІ ВИСНОВКИ..... 110
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 112

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Legrand представила третє покоління своєї кабельної системи (Legrand Cabling System 3, LCS3) своїм українським партнерам. Нова, радикально оновлена система, крім кабельних компонентів, включає шафи й PDU і розрахована на застосування як у локальних мережах, так і в ЦОД. Крім мідних і оптичних кабельних компонентів до складу системи ввійшли також серверні й телекомунікаційні шафи й розеточні блоки. За допомогою нової системи Legrand розраховує розширити свою присутність у центрах обробки даних.

LCS3 – усього лише третє покоління СКС за ті 35 років, які компанія працює на цьому ринку. Окремі компоненти (роз'єми RJ-45 і комутаційні панелі) компанія початку пропонувати ще в 1993 році, однак перша закінчена система з'явилася в неї лише десять років через, в 2003 році, а наступне друге покоління – в 2009 році. Нова, радикально оновлена система розрахована на більше широку область застосування, чим її попередниця – від малих офісів до ЦОД.

LCS3 покликана задовольнити потребу ринку в продуктивних, масштабованих і (економічно) ефективних рішеннях. Мідна кабельна підсистема LCS3 Категорії 8 може підтримувати швидкості до 40 Гбіт/с, а оптична – до 100 Гбіт/с. Полки високої щільності дозволяють заощадити місце в шафах і ефективно масштабувати рішення, а без інструментальне закладення коннекторів і інші інновації в дизайні компонентів спрощують експлуатацію й обслуговування.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД.

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

– Огляд існуючих систем Legrand Cabling System 3 для ЦОД.

– Дослідження системи Legrand Cabling System 3 для ЦОД.

– Програмна реалізація системи Legrand Cabling System 3 для ЦОД.

Об'єктом дослідження є процес Legrand Cabling System 3 для ЦОД.

Предметом дослідження є методи Legrand Cabling System 3 для ЦОД.

Методи дослідження базуються на методах теорії побудови комп'ютерних систем та мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод Legrand Cabling System 3 для ЦОД.

– Розроблено вітчизняний продукт Legrand Cabling System 3 для ЦОД , який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Legrand Cabling System 3 для ЦОД.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД , 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

Ще десять років тому на мідні кабельні системи доводилося близько 90% всіх з'єднань у ЦОД. Однак потреба в підтримки все зростаючих швидкостей і зниження ціна оптичні рішення (насамперед активне встаткування) привели до того, що оптика усе ширше застосовується в центрах обробки даних. Волоконо-оптична система включає повний комплекс рішень – від коннекторів і кабелів до інструментів. Один тільки асортименти оптичних кабелів включає 113 позицій.

З'єднувачі представляють основу будь-якої кабельної системи, оскільки саме від них у першу чергу залежать її характеристики В LCS3 з'явилося відразу два нових удосконалений коннектора – оптичний MTP і мідний RJ-45. Як затверджують в Legrand, MTP є самим швидкісним на ринку, а рівень внесених перешкод в оптичних касетах на його основі не перевищує 0,35 дБ. Legrand запатентувала конструкцію нового RJ-45, що може встановлюватися на кабель без інструмента. Він розрахований на застосування в системах Категорій 5e, 6, 6A и 8 і передачу живлення PoE++.

В LCS3 з'явилися нарешті-те полки високої щільності, причому як оптичні, так і мідні. Legrand пропонує оптичні полки високої й надвисокої щільності з коннекторами LC на 96 портів і 114 портів і мідні полки високої щільності на 48 порто висотою 1U. Крім того, компанія пропонує модульні касетні полки, у які можна встановлювати як оптичні, так і мідні касети. Можливість безінструментальної фіксації й добування полиці із шафи /стійки дозволяє прискорити процес обслуговування й відмовитися від використання монтажних гвинтів.

1.2 Область застосування

Серверні й мережні шафи мають єдиний конструктив і мають високу навантажувальну здатність (до 1,5 т), при цьому вони важать на 20% легше, ніж аналоги. Крім шаф в асортименти також входять стійки, настінні шафи й універсальні аксесуари до них. Для ізоляції потоків холодного й гарячого повітря Legrand пропонує власну запатентовану модульну систему. Як затверджується, організація холодного коридору Next Generation Corridor дозволяє заощаджувати до 30% енергії на охолодженні.

За останні кілька років Legrand придбала декілька виробників і постачальників блоків розподілу живлення – Raritan, Server Technology і Shenzhen Clever Electronic – і тепер може запропонувати різноманітний асортименти інтелекульних PDU – горизонтальні й вертикальні блоки, трифазні, однофазні, на 16 і 32 А. Монтаж PDU може вироблятися без інструментів.

Лінійка включає три моделі. Моделі з посиленим захистом мають автоматичний вимикач для захисту встаткування від перевантажень і коротких замикань, моделі з обмежником перенапруги забезпечують захисту від імпульсних стрибків, а модель із амперметром відображає доступну потужність у шафі. Система захисту від випадкового від'єднання шнурів CoreLocking System дозволяє запобігти ненавмисному відключенню навантаження.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД , 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

У світі існує величезна кількість виробників устаткування для СКС. Це такі відомі німецькі й американські компанії як Siemon, Panduit, Shneider, GE на які так люблять будувати мережі. Це й величезна кількість виробників з Китаю й України, що часто маскуються під солідний бренд – а на ділі, що випускають алюмінієвий (замість мідного) кабель UTP непостійної товщини.

Hyperline

Асортименти продукції являє собою повний набір пасивних елементів, необхідних для побудови СКС: комутаційні панелі, розетки, волоконо-оптичні сполучні шнури, кроссове встаткування, модулі Keystone Jack, патч-корди, роз'єми – це лише невеликий перелік компонентів, що поставляються.

Розмаїтість продукції Hyperline забезпечує користувачам можливість прокладати СКС будь-якої складності. Висока якість і широкий спектр компонентів гарантує клієнтам стабільну роботу системи, а також повну сумісність зі СКС інших виробників. При цьому на всю продукцію Hyperline, включаючи компоненти категорії 6, підтримуються конкурентоспроможні ціни.

Інша настільки ж важлива частина продуктової лінійки Hyperline – це виробу із пластмаси, необхідні для маркування, структурування, захисту кабельних трас і компонентів СКС у цілому. Широкий спектр продукції із пластмаси представлений різними видами стяжок, хомутів, пластикових скоб, що самоклеяться площадок, кабельних вводів і т.п. При цьому асортименти даних виробів незмінно поповнюється новими моделями.

Здійснюється повний контроль якості на всіх стадіях розробки й виробництва продукції, починаючи з підготовки сировини для забезпечення відповідності кожного компонента внутрішнім і міжнародним стандартам і закінчуючи готовими виробами.

Компанія Hyperline є членом міжнародних організацій і має міжнародні сертифікати на продукцію.

ZPAS

Продукція компанії ZPAS знаходить застосування в таких галузях, як телекомунікації, енергетика, гірничодобувна промисловість, металургія, хімічна промисловість, охорона навколишнього середовища, а також в областях, що вимагає використання сучасних методів автоматизації й модернізації виробничих процесів.

Монтажні шафи й стійки виробництва ZPAS використовуються на безлічі підприємств у Польщі.

Продукція компанії ZPAS користується попитом і за межами Польщі. Так, устаткуванням виробництва ZPAS оснащені електростанції в Туреччині, Саудівській Аравії, Китаї. Компанія також поставляє за договором у Німеччину серію компонентів для розподільних шаф, комутаційні панелі для залізниці, а також пульти керування й корпуса, виготовлені з нержавіючої сталі.

Legrand

Компанія Legrand – світовий лідер у виробництві продукції електротехнічного призначення. Компанія поєднує понад 22000 співробітників на п'яти континентах, має 31 філію й більше 20 централізованих офісів в усьому світі.

Lergrand робить широкий спектр устаткування для офісів, промислових і житлових приміщень: пластикові й алюмінієві кабель-канали, настановні аксесуари до них, електроустановочні вироби, монтажні аксесуари, силове встаткування, компоненти систем автоматичного керування електроживленням.

Продукція Legrand відповідає найвищим вимогам і стандартам європейської якості. Основні характерні риси встаткування Legrand – бездоганна надійність, зручність і простота монтажу, безпека, довговічність, сучасний дизайн. Виробляється з використанням новітніх технологій і досягнень в області електротехніки.

На українському ринку Legrand працює більше 35 років. Із самого початку роботи компанії на радянському, (потім і українському ринку) виробу, вироблені Legrand, були рекомендовані до установки тільки на елітних об'єктах і об'єктах, що мають стратегічне значення, тому що завжди випереджали час за всіма показниками, пропонованим до виробів низьких напруг. Сьогодні вироби Legrand установлюють як в урядових закладах, так і житлових приміщеннях, на промислових підприємствах і офісах.

Siemon

The Siemon Company, заснована в 1903, початку свою діяльність із розробки состава матеріалу й технології виробництва ручок для столових ножів. Незабаром з'ясувалося, що цей матеріал може витримувати вплив високих температур, води що кипить, й має гарні діелектричні властивості. Це дало можливість компанії Siemon укласти контракт із AT&T (Western Electric) на виробництво 3-полюсних комутаційних блоків, які й стали її першим компонентом, зробленим для телекомунікаційних кабельних систем. Незважаючи на це, компанія кілька десятиліть випускала продукцію, що не має відносини до кабельних систем, – грамофонні пластинки й столові прилади, при цьому підтримуючи контракт із AT&T на виробництво комутаційних блоків.

Пізніше, в 60-х The Siemon Company придбала в AT&T ліцензію на технологію виробництва комутаційного встаткування на основі блоків типу 66, що й привело компанію на ринок кабельних компонентів. Компанія вирішила сконцентрувати свою діяльність на виробництві пасивних телекомунікаційних компонентів, зробивши кабельні системи й коннектори своїм основним бізнесом. З тих пор The Siemon Company випу стила кілька ліній комутаційного

встаткування, компонентів кабельних систем, тестуючих приладів і створила інженерно-технічний підрозділ, у завдання якого входить розробка нової передової продукції й сприяння в проектуванні й монтажі кабельних систем Siemon Cabling System.

Сьогодні The Siemon Company проектує й робить найбільш широкий діапазон продукції для комутації, захисту, адміністрування, маркування й тестування кабельних систем на основі UTP і ScTP 100 Ом і оптичного волокна. Компанія володіє більш ніж 200 діючими патентами у своїй області, підтверджуючи цим прагнення до світового технологічного лідерства. У число основних ліній продукції компанії Siemon входять три типи комутаційних блоків – S110, S66 і Multiflex, а також безліч ліній модульної сполучної продукції, включаючи серії CT, HD5, SM і нову серію MAX.

The Siemon Company здійснює контроль якості на всіх стадіях розробки й виробництва продукції, починаючи з підготовки сировини для забезпечення відповідності або переваги кожним компонентом внутрішніх і міжнародних стандартів. Кабельна система Siemon Cabling System підтримує діючі й нові мережні додатки, включаючи передачу мови, даних, відеозображень, TP-PMD,100 BASE-T, Gigabit Ethernet, ATM 155 Мбіт/с і ATM 622 Мбіт/с.

Розмаїтість продукції марки Siemon і її "відкрита" конструкція дає користувачам можливість вибирати кабелі й компоненти, ґрунтуючись на їхній ціні й робочих характеристиках, на відміну від інших систем, де потрібне проходження твердій схемі й пов'язаної з нею лінії компонентів. Компанія поставляє на ринок кілька ліній продукції, що відповідає вимогам стандартів ANSI / TIA / EIA-568-A і ANSI / TIA / EIA 606, ISO / IEC 11801 і інших промислових стандартів.

Molex

Molex Incorporated – світовий лідер по виробництву електронних, електричних і волоконо-оптичних компонентів, а також комплексних рішень в області СКС (підрозділ Molex Premise Networks). У числі партнерів компанії такі

фірми як CISCO, Ford, Intel, Nokia, Phillips, Siemens, 3Com і інші. Компанія Molex заснована в 1938 році в США (штат Іллінойс), на сьогоднішній день має оберт в 3 млрд. доларів США. 55 заводів і фабрик компанії розташовані в 19 країнах миру, на них працює більше 32000 чоловік.

У липні 1994 року корпорація Molex придбала компанію Mod-Tap з метою більше активної участі в розвитку й просуванні на ринок рішень для структурованих кабельних систем. Це злиття дало можливість компанії Molex розширити спектр своїх продуктів і пропонувати не тільки окремі компоненти виробництва Molex, але й комплексні рішення від Molex Premise Networks.

Корпорація Molex приділяє особливу увагу ринкам Східної Європи, де одним з перспективних є ринок України.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з

іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення,

дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності. Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windows або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв’язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв’язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Вводу тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії

InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

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 Legrand Cabling System 3 для ЦОД.

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

Організації стандартизації й базові стандарти СКС

Абревіатура СКС – структуровані кабельні системи – уже настільки поширена, що не вимагає пояснень для більшості користувачів персональних комп'ютерів. Під цим терміном розуміють телекомунікаційну інфраструктуру будівель або, інакше кажучи, середовище передачі будь-яких слабкострумових сигналів у межах (комплексу) житлових, офісних і промислових будівель.

Ця стандартизована основа локальних комп'ютерних і офісних телефонних мереж завойовує все більше визнання завдяки ряду переваг – універсальності, зручності експлуатації й надійності. Успіх даної технології залежить у чималому ступені від організацій, що розвивають, що впроваджують і використовують СКС, і, у тому числі, від організацій стандартизації.

Розробку загальновизнаних стандартів СКС ведуть у США, у Європейській і Міжнародній організаціях стандартизації.

США

У середині 80-х років ряд компаній, що представляють телекомунікаційну й комп'ютерну індустрію США, виступили з ініціативою розробки стандартів кабельної інфраструктури будівель. У рамках Асоціації електронної промисловості (EIA) було створено кілька робочих груп. Проект Інженерного комітету TR-41, одержав назву TR-41.8. Комітет заснував кілька робочих груп по наступних напрямках стандартизації:

– TR-41.8.1 Робоча група кабельної проводки комерційних і промислових будівель.

– TR-41.8.2 Робоча група кабельної проводки житлових будівель і малих офісів.

– TR-41.8.3 Робоча група телекомунікаційних каналів і приміщень / Адміністрування.

– TR-41.8.4 Робоча група магістралей житлових будівель і малих офісів.

– TR-41.8.5 Робоча група визначень.

– TR-41.7.2 Робоча група по заземленню й електричним з'єднанням.

– TR-41.7.2 Робоча група по рекомендаціях електромагнітної сумісності.

Технічні комітети Асоціації електронної промисловості (EIA) і координаційні комітети Правління EIA розробляють стандарти спільно. Члени комітетів працюють добровільно без якої-небудь компенсації. Компанії, які вони представляють, не обов'язково є членами Асоціації. Таким чином, прийняті документи є результатом домовленості зацікавлених фахівців і відбивають їхній різнобічний досвід, у даній області, наявний до моменту твердження стандартів.

В 1998 році Сектор телекомунікацій Асоціації електронної промисловості був перетворений в Асоціацію телекомунікаційної промисловості (TIA), що підкоряється Технічній раді. Ставши незалежною організацією, Асоціація продовжує здійснювати діяльність по стандартизації разом з Асоціацією електронної промисловості (EIA). Ці організації представлені в назвах стандартів як ANSI / TIA / EIA.

ANSI / TIA / EIA переглядає більшість стандартів кожні п'ять років. У результаті стандарти можуть бути підтверджені, доповнені або змінені. Зміни, які передбачається внести, направляються Голові комітету або в секретаріат ANSI / TIA / EIA.

Міжнародні організації стандартизації

Міжнародна організація стандартизації (ISO) і Міжнародна електротехнічна комісія (IEC) утворять орган стандартизації, визнаний в усьому світі. Національні організації – члени ISO і IEC беруть участь у розробці стандартів у складі Технічних комітетів. Комітети, створені галузевими організаціями, взаємодіють один з одним у суміжних областях. У спільній роботі беруть участь інші міжнародні, урядові й неурядові організації.

Міжнародна організація стандартизації й Міжнародна електротехнічна комісія заснували Об'єднаний технічний комітет ISO / IEC JTC 1, що спеціалізується в області інформаційних технологій. Проекти міжнародного стандарту, схвалені Об'єднаним технічним комітетом, передаються в національні організації стандартизації для голосування. Для прийняття стандарту потрібно не менш 75% голосів.

Європейські організації стандартизації

Європейський комітет стандартизації електротехніки (CENELEC) діє регіонально в тісній координації з Міжнародною організацією стандартизації. Країни, що входять в CENELEC, приймають європейські стандарти в якості національних без яких-небудь виправлень. Європейські стандарти публікуються на трьох офіційних мовах – англійській, французькій й німецькій. Переклади на інші мови, зроблені членами CENELEC, і завірені в Центральному секретаріаті, одержують статус офіційних версій.

Базові стандарти СКС

Базовими стандартами структурованих кабельних систем є: ANSI / TIA / EIA-568-A. Стандарт телекомунікаційних кабельних систем комерційних будівель. Жовтень 1995 року; ISO / IEC 11801. Інформаційні технології. Структурована кабельна система для приміщень замовників. Липень 1995 року; EN 50173:1995. Інформаційні технології. Структуровані кабельні системи. Липень 1995 року.

Стандарти покликані служити суспільним інтересам, усуваючи непорозуміння між виробниками й споживачами, забезпечуючи взаємозамінність і універсальну якість продукції поряд з її доступністю й грамотним використанням. Стандарти телекомунікаційної інфраструктури будівель повинні забезпечити роботу різнотипного встаткування будь-яких виробників, створення кабельні системи на етапі будівництва будівель і їхню тривалу експлуатацію.

Стандарт ANSI / TIA / EIA-568-A замінив ANSI / TIA / EIA-568, що діяв з липня 1991 року. У нову редакцію ввійшли доповнення, прийняті у формі

технічних бюлетенів: EIA / TIA TSB 36, TIA / EIA TSB 40 і TIA / EIA TSB 40a. Бюлетені містили параметри категорій 3, 4 і 5 для кабелів типа незахищена кручена пари (UTP) і роз'ємів. У стандарт додані специфікації Проекту TSB 53 захищеної кабельної системи із хвильовим опором 150 ом, багатомодового оптоволокна 62,5 / 125 мкм, одномодового волокна, ОВ роз'ємів і обмежень для оптоволоконого середовища передачі. Системи категорії 1 і 2 виключені з даного стандарту.

Міжнародний стандарт ISO / IEC 1180 був підготовлений Підкомітетом 25 ISO / IEC JTC 1 "Підключення встаткування інформаційних технологій". Європейський стандарт EN 50173 був прийнятий Технічним комітетом 115 "Електротехнічні аспекти телекомунікаційного встаткування". На додаток до американського стандарту, що визначає як альтернативне середовище передачі захищені системи із хвильовим опором 150 ом (розробка IBM) визначені параметри незахищених чотирьохпарних систем із хвильовим опором 120 ом (розробка Alcatel). Характеристики універсальних 100-омних систем розрізняються незначно.

Базові міжнародні і європейські стандарти збігаються практично буквально. Однак ISO / IEC і CENELEC розробляють власні стандарти в суміжних областях. У Європі, наприклад, існує Директива EMC, визначені власні параметри екранованих і оптоволоконих кабелів. Міжнародна організація стандартизації веде розробку стандартів проектування, монтажу, адміністрування, вимірів і впровадження застосунків. Назви взаємозалежних діючих і розроблювальних стандартів приводяться в додатках до кожного документа.

Україна бере участь у роботі Міжнародної організації стандартизації (ISO), але не входить в CENELEC. Тому в даному огляді за основу взяті положення й термінологія міжнародних стандартів. У США діє ряд стандартів, які тільки розробляються в згаданих організаціях і широко застосовуються при створенні СКС у всіх країнах. Організації ISO / CENELEC використовують

розробки ANSI / TIA / EIA як щабля для руху вперед. При цьому вони виправляють недоліки американських стандартів. В огляді наведені відмінності американських стандартів і відзначені виправлені недоліки.

Групи стандартів СКС

Організації стандартизації діють на міжнародному, регіональному й національному рівнях. Ініціатива розробки стандартів СКС належить США, які також лідирують у їхньому прийнятті. Ряд інших країн, наприклад, Канада, Німеччина, розробляють і використовують власні стандарти. Германія випереджає всіх у розробці й використанні нових категорій.

Стандарти Асоціації електронної й телекомунікаційної промисловості й Американського національного інституту стандартизації (ANSI) найбільше повно відбивають різні аспекти створення телекомунікаційної інфраструктури. Як видно з таблиці, наведеної нижче, міжнародні і європейські організації ще не опублікували свої варіанти стандартів, що діють у США з 1990 – 1995 року.

За змістом й областям застосування стандарти можна підрозділити на три групи – проектування, монтажу й експлуатації.

Стандарти проектування визначають середовище передачі, параметри роз'ємів, лінії й каналу, у тому числі гранично припустимі довжини, способи підключення провідників (послідовність), топологію й функціональні елементи СКС. Додатка доповнюють стандарти в суміжних областях і підрозділяються на нормативні (частина стандарту) і інформаційні (для відомості). До цієї групи можна віднести також документи, що визначають параметри заземлення, особливості СКС малих офісів і житлових будівель, централізованих систем і рекомендації з побудови відкритих офісів.

Стандарти монтажу визначають у широкому змісті телекомунікаційні аспекти проектування й будівництва (комплексу) будівель. Облік телекомунікаційної інфраструктури має на увазі наявність каналів для прокладки кабелів і приміщень для їхньої комутації й розміщення встаткування. У вузькому змісті під монтажем розуміють роботи з установки кабельних систем. Другий

підхід є більше дорогим. У дану групу включені також стандарти вимірів, оскільки на практиці якість монтажу СКС визначається за допомогою вимірів, які можуть завершувати процес створення систем.

Стандарти адміністрування визначають правила документування телекомунікаційної інфраструктури й створюються на базі стандартів проектування й монтажу.

Назви й час прийняття перерахованих стандартів наведено нижче.

Міжнародні стандарти

ISO / IEC 11801 Інформаційні технології – структуровані кабельні системи для приміщень замовника.

ISO / IEC 11801A1 / A2 Інформаційні технології – структуровані кабельні системи для приміщень замовника; Додатки 1/2.

Європейські стандарти

EN 50173:1995 Інформаційні технології – структуровані кабельні системи (1995 рік).

EN 50173 / A1:2000 Інформаційні технології – структуровані кабельні системи (2000 рік):

– 1 березня 2000 року – оголошення на національному рівні.

– 1 вересня 2000 року – публікація національного стандарту, ідентичного європейському.

– 1 вересня 2000 року – скасування діючих положень національних стандартів, що не відповідають європейському.

– 30 жовтня 2000 – коментарі національних комітетів стандартизації

Підготовлено до публікації друге видання EN 50173, що замінить EN 50173:1995 і EN 50173 / A1:2000.

Стандарти США

ANSI / TIA / EIA-568-A Телекомунікаційні стандарти кабельних систем комерційних будівель, (жовтень 1995).

Доповнення:

– ANSI / TIA / 568-A-1. Специфікації затримки й фазового зрушення для 100-омних 4-парних кабельних систем, 1997.

– ANSI / TIA / 568-A-2. Виправлення й доповнення до ANSI / TIA / EIA-568-A,1998.

– ANSI / TIA / 568-A-4. Виробнича методика вимірів NEXT сполучних кабелів і вимоги до кабелів незахищена кручена пари, 1999.

– ANSI / TIA / 568-A-5 Специфікації параметрів передачі 4-парних 100-омних кабельних систем категорії 5е, 1999.

EIA / TIA-569 Стандарти прокладки телекомунікаційних каналів комерційних будівель (жовтень 1990).

EIA / TIA-570 Стандарт телекомунікаційних кабельних систем житлових і малих комерційних будівель (червень 1991).

TIA / EIA-606 Стандарт адміністрування телекомунікаційної інфраструктури комерційних будівель (лютий 1993).

TIA / EIA-607 Вимоги по заземленню й електричним з'єднанням телекомунікаційних систем комерційних будівель (серпень 1994).

TIA / EIA TSB 72 Посібник із централізованих оптоволоконим кабельним системам (жовтень 1995).

TIA / EIA TSB 75 Додаткові вимоги побудови горизонтальних кабельних систем відкритих офісів (серпень 1996).

ISO / IEC 11801. Стандарт телекомунікаційної інфраструктури комерційних будівель

Стандарти визначають структуру й параметри слабкострумових кабельних систем, установлюваних в одному, декількох або комплексі будівель.

Універсальна телекомунікаційна інфраструктура будівель призначена для передачі сигналів всіх типів, включаючи мовні, інформаційні й відео. Системи сигналізації, які встановлюють у сучасних будинках, не висвітлюються в стандартах СКС (згадуються в ANSI / TIA / EIA-568-A. Вимоги по безпеці

(електричному, пожежному й іншому видам) і електромагнітної сумісності (ЕМС) визначаються іншими стандартами й нормативами. Положення базових стандартів СКС погодяться з нормами безпеки й ЕМС.

Стандарти забезпечують:

– користувачів – структурованої (добре організованої) кабельною системою, що не залежить від типу застосунків, і відкритий ринок – елементами для створення таких систем;

– користувачів – гнучкою схемою прокладки кабелів, що дозволяє легко й економічно виконувати модифікацію системи;

– будівельників-професіоналів (наприклад, архітекторів) інструкціями, що дозволяють проектувати й будувати кабельні системи ще до того, як стануть відомими конкретної вимоги користувачів, що забезпечує планування будівництва й ремонту;

– промисловість і організації стандартизації – кабельною системою, що забезпечує роботу наявного мережного встаткування й базу для розробки нових видів продукції.

Стандарти дозволяють створювати середовище передачі з елементів різних виробників завдяки взаємодії організацій стандартизації один з одним.

Стандарти США визначають два рівні вимог – обов'язковий і що рекомендується. Обов'язковий рівень виражається словом «повинен», що рекомендується – словами – «треба», «може», «бажано». Обов'язковий рівень задає мінімум характеристик і параметри сумісності. Рівень, що рекомендується, використовується для більше повної відповідності параметрів СКС вимогам застосунків і різних умов експлуатації. У тому випадку, якщо для одного параметра задаються два рівні, що рекомендується рівень задає більше високу якість систем і являє собою верхню планку при створенні нових СКС.

Міжнародні і європейські стандарти не визначають рівні вимог, однак використовують ті ж слова, що припускають їх. Обов'язкові й нормативи, що

рекомендуються, як правило, не розрізняють. У даному огляді рівні вимог точно позначені . Крім того, обов'язкові нормативи виділені жирним шрифтом.

1. Масштаб

Найважливіші принципи СКС – універсальність і довговічність. Вони дозволяють будівельниками створювати системи перш, ніж стануть відомі вимоги користувачів, і забезпечити термін служби телекомунікаційної інфраструктури будівель до 10 років і більше. Системи оптимізовані для будівель з офісною площею до 1,000,000 м2, числа користувачів 50 – 50,000 чоловік і відстаней між будинками до 3 км. Принципи побудови СКС рекомендується використовувати також для систем, число користувачів і розмір яких виходять за зазначені рамки.

2. Нормативні посилання

Після вступної частини, відбитої вище, у стандартах приводяться перелік стандартів, що доповнюють даний стандарт, що діють на момент прийняття стандартів.

3. Визначення й скорочення

Визначення й скорочення необхідні для точного розуміння категорій, без чого неможливо однозначне тлумачення положень стандартів.

Положення, викладені в стандартах, підлягають змінам, що відбивають прогрес мережних і кабельних технологій і термінального встаткування.

4. Відповідність

Кабельна система будується у відповідності з визначеними у стандартах вимогам і рекомендаціями.

5. Структура СКС

Під структурою СКС розуміють модель побудови системи з функціональних елементів і підсистем. Даний розділ визначає також інтерфейси крапки для підключення термінального встаткування до структурованої системи й самої СКС – до мережі загального користування. Групи функціональних елементів утворять підсистеми СКС.

5.1. Функціональні елементи СКС

Структурована кабельна система – середовище передачі електромагнітних сигналів – складається з елементів – кабелів і роз'ємів. Кабелі, оснащені роз'ємами й прокладені за певними правилами, утворять лінії й магістралі. Лінії, магістралі, точки підключення й комутації становлять функціональні елементи СКС.

В американському стандарті до функціональних елементів відносять два типи кабелів, три типи приміщень, елемент конструкції будівлі й документацію телекомунікаційної інфраструктури. Крім того, у даних групах стандартів використовується різна термінологія. Міжнародні / європейські стандарти підрозділяють СКС на вісім функціональних елементів, американський – на сім. Тільки два з них збігаються. У першому випадку функціональні елементи становлять середовище передачі, тобто властиво структуровану кабельну систему. Це дозволяє виділити підсистеми й провести точні границі між ними.

У другому до складу функціональних елементів не ввійшла магістраль комплексу й всі інтерфейси СКС і додані приміщення, елементи будівель і система документування. Це приведе до плутанини й змішування понять у технічній літературі, проспектах виробників і документації, створюваних по американській моделі – А.В.

5.2. Підсистеми СКС

Міжнародні / європейські стандарти підрозділяють СКС на три підсистеми: магістральна підсистема комплексу, магістральна підсистема будівлі, горизонтальна підсистема.

Розподільні пункти забезпечують можливість створення топології каналів типу «шина», «зірка» або «кільце».

5.2.1. Магістральна підсистема комплексу включає магістральні кабелі комплексу, механічне закінчення кабелів (роз'єми) у розподільчому пункті (РП) комплексу й РП будинки й комутаційні з'єднання в РП комплексу. Магістральні

кабелі комплексу також можуть з'єднувати між собою розподільні пункти будівель.

5.2.2. Магістральна підсистема будівлі включає магістральні кабелі будівлі, механічне закінчення кабелів (роз'єми) у РП будинки й РП поверху, а також комутаційні з'єднання в РП будинки. Магістральні кабелі будівлі не повинні мати крапок переходу, електропровідні кабелі не слід з'єднувати сплайсами.

5.2.3. Горизонтальна підсистема включає горизонтальні кабелі, механічне закінчення кабелів (роз'єми) у РП поверху, комутаційні з'єднання в РП поверху й телекомунікаційні роз'єми. У горизонтальних кабелях не допускається розривів. При необхідності допускається одна крапка переходу. Усі пари й волокна телекомунікаційного роз'єму повинні бути підключені. Телекомунікаційні роз'єми не є крапками адміністрування. Не допускається включення активних елементів і адаптерів до складу СКС.

Абонентські кабелі для підключення термінального встаткування не є стаціонарними й перебувають за рамками СКС. Однак, стандарти визначають параметри каналу, до складу якого входять абонентські й мережні кабелі.

5.3. Топологія СКС

Топологія СКС – «ієрархічна зірка», що допускає додаткові з'єднання розподільних пунктів одного рівня. Однак такі з'єднання не повинні заміняти магістралі основної топології. Число й тип підсистем залежить від розмірів комплексу або будинки й стратегії використання системи. Наприклад, у СКС один будинки досить одного РП будівлі й двох підсистем – горизонтальної й магістральної. З іншого боку, великий будинок можна розглядати як комплекс, що включає всі три підсистеми, і в тому числі, трохи РП будівлі.

5.4. Розміщення розподільних пунктів

Розподільні пункти розміщаються в телекомунікаційних приміщеннях і апаратних. Телекомунікаційні приміщення призначені для установки панелей і шаф, мережного й серверного встаткування, що обслуговують весь або частина

поверху. Апаратні виділяють для т елекомунікаційного встаткування, щообслуговує користувачів усього будівлі (наприклад, УАТМ, мультиплексори, сервери) і розміщення РП будівлі / комплексу. Панелі / шафи й устаткування РП поверху, сполучені із РП будівлі / комплексу, також можуть перебувати в приміщенні апаратної.

5.5. Інтерфейси СКС

Інтерфейси СКС це закінчення підсистем, що забезпечують підключення встаткування й кабелів зовнішніх служб методом підключення або комутації.

Для підключення до СКС досить одного мережного кабелю. У варіанті комутації використовують мережний і комутаційний кабель і додаткову панель.

Підключення до мережі загального користування здійснюється за допомогою інтерфейсу мережі загального користування. Місце розташування інтерфейсу мережі загального користування визначається національними, регіональними й місцевими правилами. Якщо інтерфейси мережі загального користування й СКС не з'єднані комутаційним кабелем або за допомогою встаткування, необхідно враховувати параметри проміжного кабелю.

5.6. Конфігурація

5.6.1. Розподільний пункт поверху

Як мінімум один РП поверху рекомендується на кожні 1000 квадратних метрів офісної площі. На кожному поверсі повинен бути, принаймні, один РП поверху. Якщо число робочих місць на поверсі невелико, його можна обслуговувати за допомогою розподільного пункту на суміжному поверсі.

5.6.2. Рекомендовані типи кабелів

У таблиці 3.1 дані рекомендації застосування різних типів середовища передачі в кожній з підсистем.

Таблиця 3.1 – Рекомендоване середовище передачі підсистем СКС

Підсистема	Тип середовища передачі	Застосунки
Горизонтальна підсистема	Симетричні кабелі	Мовні й інформаційні
	Оптоволоконі кабелі	Інформаційні
Магістральна підсистема будівлі	Симетричні кабелі	Мовні й інформаційні класів А й В
	Оптоволоконі кабелі	Інформаційні класів В й вище
Магістральна підсистема комплексу	Оптоволоконі кабелі	Для всіх застосунків
	Симетричні кабелі	Для застосунків класу А (наприклад, лінії УАТМ)

Дані рекомендації застаріли – інформаційні додатки класів А (до 0,1 МГц) і В (до 1,0 МГц) у локальних мережах практично не застосовуються. Вибір середовища передачі для магістралі будівлі залежить від також від довжини каналів. Якщо довжина магістральної лінії не перевищує 90 метрів, симетричні кабелі відповідної категорії покликані забезпечити роботу всіх діючих застосунків – А.В.

З іншого боку, більшість багатомодових кабелів непридатні для роботи Gigabit Ethernet при довжині лінії більше 220 метрів (у відповідності зі стандартами максимальна довжина ОВ ММ магістралі – 2000 метрів) – А.В.

5.6.3. Телекомунікаційні роз’єми (ТР)

Телекомунікаційні роз’єми розташовують на стіні, підлозі або в іншій крапці робочої області. При проектуванні СКС варто забезпечити зручність доступу до всіх роз’ємів. Висока щільність роз’ємів підвищує гнучкість системи й полегшує зміни телекомунікаційних ресурсів робочих місць. У багатьох країнах

на 10 м^2 використовуваній площі повинні установлюватися два телекомунікаційних роз'єми.

Допускається установка роз'ємів поодиноко або групами, однак кожне робоче місце **повинне** мати не менш двох роз'ємів.

На кожному робочому місці **повинен** бути передбачений, принаймні, одне роз'єми, установлений на симетричному кабелі 100 ом або 120 ом (перевага віддається кабелям 100 ом). Інші ТР **потрібно** встановлювати або на симетричним, або на оптоволоконом кабелі.

Симетричний кабель повинен мати дві або чотири пари; усе пари повинні бути змонтовані на роз'єми. Якщо передбачено менш чотирьох пар, це потрібно відбити в маркуванні. Застосунки збалансованої передачі можуть мати обмеження по затримці поширення сигналів по кожній з пар. Особливості специфікації ТР, що відповідають перерахованим вище типам кабелів, дані в розділі «Вимоги до роз'ємів».

Роз'єми повинні бути позначені постійним маркуванням, видної користувачеві. Варто звертати увагу на те, щоб реєструвалося первісне призначення пар, а також всі наступні зміни. Хвильові й інші адаптери, використовувані для узгодження різних передавальних середовищ, повинні перебувати із зовнішньої сторони роз'єму. Дозволяється міняти призначення пар за допомогою адаптерів.

У українськомовній літературі поняття «телекомунікаційне роз'єми» повсюдно підмінюють терміном «телекомунікаційна розетка». Роз'єми або закінчення кабелю, є частиною розетки, тобто складання роз'ємів, настановної й фіксуючої арматур. Розетка може поєднувати від одного до дванадцяти роз'ємів – А.В.

5.6.4. Телекомунікаційні приміщення й апаратні

Телекомунікаційне приміщення покликане забезпечувати наявність всіх засобів (простір, електроживлення, обігрів, вентиляція) для розташованих усередині нього пасивних елементів, активних пристроїв, а також інтерфейсів

мережі загального користування. Для кожного телекомунікаційного приміщення варто передбачити прямий доступ до магістралі будівлі.

Апаратна – простір у межах будівлі, де розміщається телекомунікаційне встаткування й можуть перебувати або бути відсутніми розподільні пункти. До апаратного висувають інші вимоги, чим до телекомунікаційних приміщень, оскільки встаткування, установлюване в них, є більше складним (наприклад, УАТМ або сервери). В апаратній може перебувати більше одного розподільного пункту. Якщо телекомунікаційне приміщення служить для розміщення двох і більше розподільних пунктів, його варто вважати апаратною.

Термін «телекомунікаційне приміщення» часто переводять як «телекомунікаційна шафа». Ці поняття не збігаються. Якщо використовується кілька шаф /стійок, неправильний переклад приводить до непорозумінь. Особливо серйозні помилки виникають при проектуванні системи заземлення й тлумаченні стандартів, що також використовують даний термін – А.В.

5.6.5. Пункт вводу в будинок

Пункти вводу в будинок обладнаються у випадку, коли зовнішні кабелі магістралі комплексу, приватних мереж і мережі загального користування (включаючи антену) уводять у будинок і здійснюють перехід на внутрішні кабелі. Місцеві правила можуть вимагати спеціального комутаційного устаткування для оснащення зовнішніх кабелів роз'ємами. Це встаткування дозволяє перейти від зовнішніх до внутрішніх кабелів.

5.7. Електромагнітна сумісність

Міжнародні стандарти електромагнітних випромінювань і стійкості (наприклад, CISPR 22) і місцеві правила **повинні бути** прийняті в увагу. Кабельна система вважається пасивної й не може бути протестована на відповідність вимогам EMC індивідуально. Активне встаткування повинне відповідати вимогам відповідних стандартів EMC із урахуванням використовуваного середовища передачі.

5.8. Заземлення

Елементи системи заземлення **повинні** відповідати вимогам відповідних норм і правил. Інструкції й вимоги виробників устаткування варто виконувати, якщо вони сумісні з електричними нормативами.

Важливо відзначити, що відповідальність за відповідність СКС вимогам електромагнітної сумісності делегована виробникам активного встаткування. Такий підхід не вирішує проблеми. Строго говорячи, пункти 5.7. Електромагнітна сумісність і 5.8. Заземлення не ставляться до конфігурації СКС і висвітлюються в розділі 10, спеціально присвяченому даним проблемам. Крім того, вони не містять норм і правил, а тільки посилання на інші стандарти – A.B.

1) Коли бажана більша гнучкість системи, варто використовувати чотирьохпарні кабелі.

2) Установка двопарних кабелів обмежує роботу застосунків класу D.

6. Підсистеми СКС

Дана глава визначає модель горизонтальної й магістральної підсистем, максимальну довжину, кращі й рекомендовані типи кабелів. Рекомендується відповідність цим вимогам для більшості встановлених систем.

Загальна довжина абонентських (A), комутаційних (B) і мережних кабелів (E), що утворять канал горизонтальної підсистеми, – до 10 метрів.

Довжина комутаційних кабелів у РП будинки (C) і РП комплексу (D) – не більше 20 метрів.

Довжина мережних кабелів у РП будинки (F) і РП комплексу (G) – не більше 30 метрів.

Дотримання зазначених довжин строго рекомендується, однак не є вимогою, оскільки абонентські й мережні кабелі перебувають за рамками міжнародного, європейського й американського стандартів.

Вимоги до елементів системи – кабелям і роз'ємим – визначається в розділах «Вимоги до кабелів» і «Вимоги до роз'ємів». Симетричні кабелі із хвильовим опором 100 і 120 ом і роз'єми для них підрозділяються по категоріях.

Параметри передачі категорій 3, 4 і 5 визначені в смузі частот 16, 20 і 100 Мгц відповідно.

Кабелі й роз'єми різних категорій можуть бути встановлені в межах підсистеми й / або кабельної лінії, але передавальні робітники характеристики лінії будуть визначатися категорією гіршого елемента.

Елементи з різним хвильовим опором не допускається встановлювати в одній лінії. Оптичні волокна з різними діаметрами серцевини не дозволяється з'єднувати в межах однієї кабельної лінії. Баг аторазова поява того самого провідника або провідників (шунтовані відводи), не може бути частиною кабельної системи.

6.1. Горизонтальна підсистема

6.1.1. Довжина кабелів.

Максимальна довжина горизонтального кабелю повинна становити 90 м, незалежно від типу середовища. Вона виміряється від роз'єми (панелі) у РП поверху до телекомунікаційного роз'єму на робочому місці. Максимальна механічна довжина абонентських, комутаційних (перемичок) і мережних кабелів – не більше 10 метрів.

Для відповідності вимогам застосунків настійно рекомендується використання абонентських і мережних кабелів, робочі характеристики яких відповідають або перевищують параметри комутаційних кабелів. Довжина комутаційних кабелів і перемичок у РП поверху не повинна перевищувати 5 м.

Показана модель горизонтальної підсистеми, що забезпечує узгодження параметрів кабелів («Вимоги до кабелів») і ліній («Специфікація ліній»). Для цього фіксований кабель горизонтальної лінії обмежений довжиною 90 метрів і гнучкий – довжиною 5 метрів (що еквівалент но сумарній електричній довжині 97,5 метрів), а лінія включає три роз'єми однакової категорії. Крапка переходу є резервною й відсутній у даній моделі. Якщо використовується крапка переходу, параметри лінії **повинні** відповідати моделі із двома роз'ємами й довжиною кабелю не більше 90 метрів.

Абонентський і мережний кабелі не входять до складу структурованої кабельної системи, однак дозволяють створити канал з параметрами, що задаються стандартами. Передбачається, що загальна електрична довжина мережного й абонентського кабелів еквівалентна 7,5 метрам (відповідно до умов «Вимоги до кабелів»). Різниця механічної й електричної довжини для гнучких кабелів обумовлена вимогами до загасання.

Відмінності ANSI / TIA / EIA-568-A

Довжина комутаційних кабелів (або перемичок) і мережних кабелів не повинна перевищувати 6 метрів. Передбачається, що довжина абонентського кабелю (від ТР до робочої станції) становить 3 метри, а загальна довжина сполучних кабелів обмежена 10 метрами.

Обмеження на рівні обов'язкової вимоги довжини комутаційних кабелів дозволяє встановити параметри горизонтальної підсистеми СКС. Для організації каналу діє рекомендація із сумарної довжини всіх гнучких кабелів – до 10 метрів. Гнучкі або сполучні кабелі відрізняються типом роз’ємів (штекерні , на відмі ну від гніздових у фіксованих кабелів) і конструкцією провідників – кожний провідник складається із семи мідних жил – А.В.

В американську модель лінії виявився включеним мережний кабель, що відповідно до положень того ж стандарту не входить до складу СКС. Це одне із протиріч, якого немає в міжнародних і європейських стандартах – А.В..

Модель оптоволоконих горизонтальних кабелів відрізняється можливою наявністю сплайсів на обох кінцях підсистеми й відсутністю комутаційних кабелів.

Деякі технології, зокрема моніторинг з'єднань СКС за допомогою системи LAN Sense, мають на увазі створення каналів з комутацією також і для оптоволоконих горизонтальних підсистем – А.В.

1) Специфікації комутаційних і інших гнучких кабелів дані в «Вимоги до гнучких симетричних кабелів 100, 120 і 150 ом»

6.1.2. Вибір типу кабелю.

Для використання в горизонтальній кабельній підсистемі рекомендуються кабелі двох типів:

Кращі: симетричний кабель 100 ом і багатомодове оптичне волокно 62,5/125 мкм.

Альтернативні: симетричний кабель 120 ом, симетричний кабель 150 ом, кабелі із багатомодовим оптичним волокном 50 / 125 мкм.

Параметри кабелів, роз'ємів наведені в «Вимоги до кабелів» і «Вимоги до роз'ємів». Для підключення декількох телекомунікаційних роз'ємів можливе застосування гібридного й композиційного кабелів. Якщо є екрановані або заземлені провідники, варто керуватися положеннями розділу «Практика екранування».

Відмінності ANSI / TIA / EIA-568-A

1. Відсутні симетричний кабель 120 ом і кабелі із багатомодовим оптичним волокном 50 / 125 мкм.

2. Як середовище передачі зізнається коаксіальний кабель 50 ом. Однак він не рекомендований для монтажу в знову встановлюваних СКС і повинен бути виключений з наступної редакції стандарту. Інші типи середовища передачі, також не включені в стандарт і доповнення, що допускаються до використання в якості, до мінімальної конфігурації, – екрановані кабелі 100 ом, багатопарні кабелі й коаксіальні кабелі 75 ом.

6.1.3. Конфігурація телекомунікаційних роз'ємів.

Два телекомунікаційних роз'єми, що забезпечують мінімальні ресурси робочої області відповідно до розділу «Структура СКС», можуть бути встановлені в такий спосіб:

а) один телекомунікаційний роз'єм повинен бути встановлений на симетричному кабелі категорії 3 або вище;

б) другий телекомунікаційний роз'єм повинен бути встановлений на симетричному кабелі категорії 5 (100 ом або 120 ом), на симетричному кабелі 150 ом або на багатомодовому оптоволоконому кабелі.

Вимоги по конфігурації ТР занижені з погляду сучасних вимог: кабелі категорії 3 практично не використовуються. Найбільше поширення одержали кабелі із хвильовим опором 100 ом, що забезпечують погоджене середовище передачі для переважної більшості зразків стандартного мережного встаткування – А.В.

6.2. Магістральна підсистема

6.2.1. Фізична топологія

У магістральній підсистемі **повинне** бути не більше двох рівнів комутації, що дозволяє обмежити деградацію сигналу в пасивних системах і спростити адміністрування. На шляху від РП поверху до РП комплексу повинен бути не більш ніж один розподільний пункт.

Єдиний розподільний пункт може забезпечити комутацію всієї магістральної підсистеми. Розподільні пункти магістральної кабельної системи можуть розташовуватися в телекомунікаційних приміщеннях або апаратних. У додатку D дані рекомендації зі створення логічної топології «кільце», «шина» і інших на основі фізичної топології «зірка».

Топологія «зірка» застосовна не тільки до кабелів, але й кабельним елементам передавального середовища, таким як індивідуальні волокна або пари Залежно від параметрів системи, кабельні елементи можуть перебувати в одному кабелі по всій довжині або тільки на частині довжини лінії. У магістральній підсистемі допускається використання гібридних і багатоелементних кабелів, що відповідають параметрам розділу 8Вимоги до кабелів.

6.2.2. Вибір типів кабелів

Стандарт визначає п'ять типів передавального середовища. У магістральній підсистемі можливе використання більше одного типу:

– багатомодове й одномодове оптичне волокно (перевага віддається багатомодовому волокну 62,5 / 125 мкм).

– симетричний кабель 100 0м, 120 0м або 150 0м (перевага віддається симетричному кабелю 100 0м). Відстані магістралі для всіх високошвидкісних застосунків, що використовують електропровідні кабелі повинні бути обмежені відповідно до розділу 6.1.1 Довжина кабелів.

Відмінності ANSI / TIA / EIA-568-A

1. Відсутні симетричний кабель 120 ом і багатомодові оптоволоконі кабелі 50 / 125 мкм.

2. Як середовище передачі зізнається коаксіальний кабель 50 ом. Однак він не рекомендований для монтажу в знову встановлюваних СКС і повинен бути виключений з наступної редакції стандарту. Інші типи середовища передачі, також не включені в стандарт і доповнення, що допускаються до використання в якості, до мінімальної конфігурації, – екрановані кабелі 100 ом, багатопарні кабелі й коаксіальні кабелі 75 ом.

6.2.3. Довжина кабелів магістралі

Максимальні відстані між розподільними пунктами повинні відповідати параметрам. У системах, розміри яких перевищують зазначені параметри, варто спроектувати додаткові РП, довжина магістралей яких не перевищує параметри даного розділу.

Обмеження довжини магістралі носять умовний характер. При використанні найпоширенішого багатомодового оптоволокна 62,5 / 125 зі смугою пропущення 160 МГц х км у вікні 850 нм канал довжиною 2000 метрів забезпечує роботу застосунків класу З (10 МГц) і нижче. Те ж волокно дозволяє передавати сигнали Fast Ethernet не більше ніж на 1300 метрів, а Gigabit Ethernet – 220 метрів. Інакше кажучи, при визначенні типу середовища й довжини каналів магістралей варто враховувати тип і вимоги протоколів – А.В.

Відстань між РП комплексу й РП поверху не повинне перевищувати 2000 м. Відстань між РП будівлі й РП поверху не повинне перевищувати 500 м.

Максимальна відстань в 2000 між РП комплексу й РП поверху може бути збільшене при використанні одномодового волоконо-оптичного кабелю. Відстань між РП комплексу й РП поверху, що перевищує 3 км у випадку застосування одномодового оптичного волокна, виходить за рамки справжнього стандарту. Довжина перемичок і комутаційних кабелів у РП комплексу й РП будівлі не повинна перевищувати 20 м. Значення довжин, що перевищують 20 м, віднімаються з максимально припустимої довжини магістрального кабелю.

Відмінності ANSI / TIA / EIA-568-A

Відстань між РП поверху й РП комплексу при використанні електропровідних ліній не повинне перевищувати 800 метрів.

Дане положення американського стандарту суперечить обмеженню сумарної довжини магістралі в 2000 метрів для багатомодового оптоволокна. Якщо в магістралі комплексу є електропровідні й оптоволоконі кабелі, буде діяти обмеження по меншій відстані. Відповідно до міжнародного / європейськими стандартамиами довжина каналу залежить від категорії середовища передачі й класу застосунків (наприклад, для кабелів категорії 5 і застосунків класу А припустима довжина каналу становить 3000 метрів) – А. В.

6.2.4. Зовнішні служби

Кабелі, по яких передаються сигнали зовнішніх мереж (наприклад, прийняті антеною) можуть входити в будинок у місцях, віддалених від розподільних пунктів. При визначенні максимальної довжини магістрального кабелю необхідно враховувати відстань між крапками вводу зовнішніх мереж і розподільним пунктом, до якого вони підключені. Місцеві нормативи й правила, що регулюють місце розташування інтерфейсів зовнішніх мереж, також впливають на їхнє видалення від розподільних пунктів. Довжину й параметри кабелів зовнішніх мереж варто документувати й надавати операторам послуг із запиту.

6.2.5. Підключення активного телекомунікаційного встаткування.

Передбачається, що довжина кабелів, що прямо з'єднують телекомунікаційне встаткування із РП комплексу й РП будівлі, не перевищує 30 м. Якщо використовуються кабелі більшої довжини, магістральні відстані повинні бути відповідно зменшені.

Відмінності ANSI / TIA / EIA-568-A

Стандарт містить додаткові рекомендації із планування магістралей. Як правило, практично неможливо або економічно недоцільно встановлювати магістраль на весь термін служби системи. Рекомендується передбачати один – три періоди тривалістю від трьох до десяти років. Для кожного з періодів проектується й установлюється максимальне число кабелів і роз'ємів у РП комплексу, будівель, поверхів і в крапках вводу.

3.2 Розробка структурної схеми

Адміністрування є важливим аспектом створення й експлуатації структурованої кабельної системи. Гнучкість СКС може бути повністю реалізована тільки при правильному адмініструванні. Адміністрування включає точне позначення й облік всіх елементів, що становлять кабельну систему, а також кабельних трас, телекомунікаційних і інших приміщень, у яких монтується система. Всі зміни, внесені в кабельну систему, варто вчасно реєструвати – це необхідно для збереження гнучкості. Настійно рекомендується проводити адміністрування з використанням комп'ютерних програм.

Сфера дії адміністрування

Вимоги по адмініструванню, описані в даному розділі, застосовні до структурованої кабельної системи, а також до трас і приміщень, у яких вона монтується. Настійно рекомендується застосовувати описані нижче принципи адміністрування до будь-якої кабельної системи й до активного встаткування.

Ідентифікатори

Кожний елемент структурованої кабельної системи, а також траси й приміщення, у яких вона монтується, повинні бути легко ідентифікуємі. Кожному кабелю, панелі й розніманню повинен мати унікальне позначення (наприклад, назва, колір, номер або рядок символів).

Для кожного телекомунікаційного роз'єму варто вказати наступну інформацію, що відбиває вибір і застосування встановленої системи:

а) ТР: IEC 603-7 Хвильовий опір, категорія й розташування пар у ТР.

б) ТР: оптоволокно. Дизайн волокна (діаметр серцевини й оболонки – А.В.).

Підходящі ідентифікатори повинні бути також привласнені трасам і приміщенням, у яких монтується кабельна система. Елементи, яким привласнюються ідентифікатори, повинні бути чітко маркіровані. Кабелі варто позначати з обох кінців.

Записи

Адміністрування СКС потрібно вести за допомогою записів. Результати тестування системи, якщо таке проводилося, варто зберігати. Не потрібно даним стандартом, але рекомендується вести облік підтримуваних застосунків. Це полегшує виявлення джерел проблем.

Документування

Для процесу адміністрування необхідний належний контроль над веденням записів (схеми кабельних маршрутів, розташування й позначення ТР, розташування й состав РП, результати тестування й схеми з'єднань). Важливо забезпечити реалізацію відповідних процедур своєчасного відновлення документації

Legrand Data Cabling Solutions LCS3

У цей час кабельні системи Legrand забезпечують високоякісний зв'язок більш ніж з 200 мільйонами пристроїв. Legrand є світовим лідером в області мереж зв'язку для передачі даних. Інвестиції в розробку й проектування

структурованих кабельних систем і рішень дозволили розширити пропозиція й досягти найвищого рівня продуктивності. Ці рішення ідеально підходять для сучасних мультимедійних мереж, технологій і застосунків. Комплексні глобальні рішення Legrand для передачі даних ідеально вирішують ключові проблеми цифрових мереж: продуктивність, масштабованість і ефективність.

Зростаючі обсяги обміну даними, що ростуть число мереж, потреба в більше високих швидкостях і щільності встаткування – все це робить необхідним створення більше надійних, безпечних і високопродуктивних електричних і цифрових інфраструктур будівель. LCS3, нова структурована кабельна лінійка Legrand, спеціально розроблена для задоволення цих потреб.

Він пропонує безліч досягнень із погляду продуктивності, масштабованості й ефективності. Нові роз'єми можуть працювати в самих критичних середовищах з мідними рішеннями до категорії 8. LCS3 також містить у собі значно розширену оптоволокону пропозицію, що забезпечує швидкість до 100 Гбіт/с. А також інновації з погляду ергономіки: наші нові структуровані кабельні рішення є модульними, простими в установці в корпусах і оптимізовані для технічного обслуговування.

Більше ефективний, масштабований і високопродуктивний, LCS3 задовольняє останнім вимогам локальних мереж і центрів обробки даних!

Структурована кабельна система (Structured Cabling System, SCS) – це набір комутаційних елементів (кабелів, роз'ємів, коннекторів, кроссових панелей і шаф), а також методика їхнього спільного використання, що дозволяє створювати регулярні, легко розширювані структури зв'язків в обчислювальних мережах.

Структурована кабельна система це система, за допомогою якої проектувальник мережі будує потрібну йому конфігурацію зі стандартних кабелів, з'єднаних стандартними роз'ємами й комутируються на стандартних кроссових панелях. При необхідності конфігурацію зв'язків можна легко

змінити – додати комп'ютер, сегмент, комутатор, вилучити непотрібне встаткування, а також поміняти з'єднання між комп'ютерами й концентраторами.

Структурована кабельна система планується й будується ієрархічно, з головною магістраллю й численними відгалуженнями від її.

Типова ієрархічна структура структурованої кабельної системи (рисунок 3.1) включає:

– горизонтальні кабельні підсистеми (у межах поверху);

– вертикальні кабельні підсистеми (усередині будівлі);

– магістральну кабельну підсистему ЦОД (у межах однієї території з декількома будинками);

– кабельну підсистему робочих місць;

– центральні комутаційні вузли будівель.

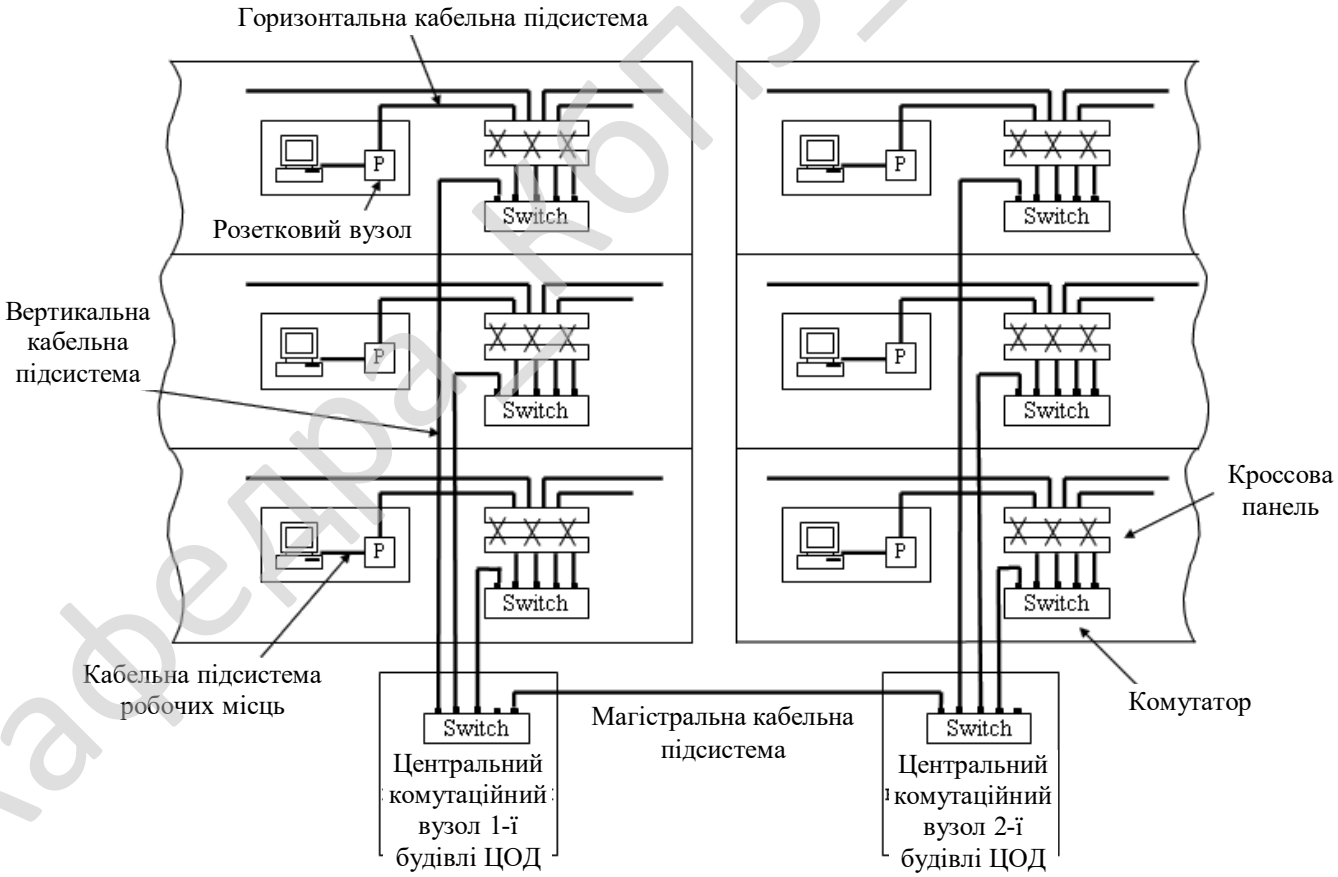

Рисунок 3.1 – Структурна схема системи

Горизонтальна підсистема

Горизонтальна кабельна система являє собою кабельне розведення, що йде від настінної розетки до місця підключення в комутаційній шафі. Ця ділянка включає наступні елементи:

– Адаптер (якщо необхідно) для перетворення інтерфейсу встаткування в модульний інтерфейс.

– Лінійні корди від комп'ютера до користувальницького інтерфейсу.

– Користувальницький інтерфейс до кабельної мережі.

– Кабелі від користувальницького інтерфейсу до комутаційної шафи.

– Неекранована кручена пари (UTP).

– Патч-кабелі й кроссове сполучне проведення, використовуваний у комутаційній шафі.

Вертикальна підсистема з'єднує кроссові шафи кожного поверху із центральної апаратної будівлі.

Підсистема ЦОД з'єднує кілька будівель з головною апаратною всього ЦОД. Ця частина кабельної системи звичайно називається магістраллю (backbone).

Комутаційний вузол

Комутаційний вузол містить необхідне встаткування для переходу між горизонтальними й вертикальними ділянками кабельної мережі й / або приєднання до якого-небудь активного встаткування (головній комп'ютерній системі, мережній апаратурі й т.д.). Інші назви комутаційного вузла включають комутаційну шафу, телекомунікаційну шафу, апаратну шафу, а також шафу для магістральної кабельної мережі.

Комутаційний вузол може бути:

– центральним – центр структурованої кабельної системи одного будівлі.

– поверховим – крапка переходу між вертикальною й горизонтальною кабельною системою.

Переваги використання структурованої кабельної системи:

– Системонезалежність.

– Легкість переміщення персоналу й устаткування без зміни проводки.

– Зручність росту й зміни структури системи.

– Реконфигуруємість.

– Модульний дизайн, що забезпечує гнучкість системи.

– Спрощується керування й обслуговування кабельного господарства.

– Легкість виявлення й локалізації несправностей.

– Підтримка широкого кола завдань.

– Зниження експлуатаційних витрат.

Загальні принципи побудови СКС передбачають:

– Універсальність СКС – сумісність із устаткуванням будь-яких виробників.

– Надмірність – забезпечення таких параметрів передачі сигналів і даних, які дозволять перейти до використання нових технологій або збільшити число користувачів СКС із мінімальними змінами в кабельній проводці.

– Модульність – можливість розвитку й зміни систем (модифікацій) з мінімальними матеріальними, трудовими й фінансовими витратами.

Кабельна система будується у відповідності міжнародним стандартам на прокладку кабелів по будинках EIA / TIA 568A – Commercial Building Telecommunication Wiring Standard і ANSI / EIA / TIA 569 – "Commercial Building Standard for Telecommunications Pathways and Spaces".

Вибір типу кабелю для горизонтальних підсистем

Більшість проектувальників починає розробку структурованої кабельної системи з горизонтальних підсистем, тому що саме до них підключаються кінцеві користувачі. При цьому вони можуть вибирати між екранованою кручений парою, неекранованою крученою парою, коаксіальним кабелем і волоконо-оптичним кабелем. Можливі використання й бездротові лінії зв'язку.

Горизонтальна підсистема характеризується дуже більшою кількістю відгалужень кабелю, тому що його потрібно провести до кожної користувальницької розетки, причому й у тих кімнатах, де поки комп'ютери в мережу не поєднуються. Тому до кабелю, використовуваному в горизонтальній проводці, пред'являються підвищені вимоги до зручності виконання відгалужень, а також зручності його прокладки в приміщеннях. На поверсі звичайно встановлюється кроссова панель, що дозволяє за допомогою коротких відрізків кабелю, оснащеного роз'ємами, провести перекомутацію з'єднань між користувальницьким устаткуванням і концентраторами / комутаторами. При виборі кабелю приймаються в увагу наступні характеристики:

– смуга пропущення,

– відстань,

– фізична захищеність,

– електромагнітна перешкодозахищеність,

– вартість.

Крім того, при виборі кабелю потрібно враховувати, яка кабельна система вже встановлена на підприємстві, а також які тенденції й перспективи існують на ринку в цей момент.

– Екранована кручена пари STP дозволяє передавати дані на більшу відстань і підтримувати більше вузлів, чим неекранована. Наявність екрана робить її більше дорогою й не дає можливості передавати голос. Екранована кручена пари використовується в основному в мережах, що базуються на продуктах IBM і Token Ring, і рідко підходить до іншому встаткуванню локальних мереж.

– Неекранована кручена пари UTP по характеристиках смуги пропущення й підтримуваних відстаней також підходить для створення горизонтальних підсистем. Але тому що вона може передавати дані й голос, вона використовується частіше.

– Коаксіальний кабель усе ще залишається одним з можливих варіантів кабелю для горизонтальних підсистем. Особливо у випадках, коли високий рівень електромагнітних перешкод не дозволяє використовувати кручену пару або ж невеликі розміри мережі не створюють більших проблем з експлуатацією кабельної системи.

– Товстий Ethernet володіє в порівнянні з тонким більшою смугою пропущення, він більше стійок до ушкоджень і передає дані на більші відстані, однак до нього складніше приєднатися й він менш гнучкий. З товстим Ethernet складніше працювати, і він мало підходить для горизонтальних підсистем. Однак його можна використовувати у вертикальній підсистемі як магістраль, якщо оптоволоконий кабель із якихось причин не підходить.

– Тонкий Ethernet – це кабель, що повинен був вирішити проблеми, пов'язані із застосуванням товстого Ethernet. До появи стандарту 10 Base-T тонкий Ethernet був основним кабелем для горизонтальних підсистем. Тонкий Ethernet простіше монтувати, чим товстий. Мережі на тонкому Ethernet можна швидко зібрати, тому що комп'ютери з'єднуються один з одним безпосередньо. Головний недолік тонкого Ethernet – складність його обслуговування. Кожний кінець кабелю повинен завершуватися термінатором 50 Ом. При відсутності термінатора або втраті їм своїх робочих властивостей (наприклад, через відсутність контакту) перестає працювати весь сегмент мережі, підключений до цього кабелю. Аналогічні наслідки має погане з'єднання будь-якої робочої станції (здійснюване через Т-коннектор). Несправності в мережах на тонкому Ethernet складно локалізувати. Часто доводиться від'єднувати Т-коннектор від мережного адаптера, тестувати кабельний сегмент і потім послідовно повторювати цю процедуру для всіх приєднаних вузлів. Тому вартість експлуатації мережі на тонкому Ethernet звичайно значно перевершує вартість експлуатації аналогічної мережі на кручений парі, хоча капітальні витрати на кабельну систему для тонкого звичайно нижче.

– Основні області застосування оптоволоконого кабелю – вертикальна підсистема й підсистеми ЦОД. Однак, якщо потрібна високий ступінь захищеності даних, висока пропускна здатність або стійкість до електромагнітних перешкод, волоконо-оптичний кабель може використовуватися й у горизонтальних підсистемах. З волоконо-оптичним кабелем працюють протоколи AppleTalk, Token Ring, а також нові протоколи 100 VG-AnyLAN, Fast Ethernet, ATM.

Переважним кабелем для горизонтальної підсистеми є неекранована кручена пари категорії 5. Її позиції ще більше зміцняться із прийняттям специфікації 802.3 ab для застосування на цьому виді кабелю технології Gigabit Ethernet.

Вибір типу кабелю для вертикальних підсистем

Кабель вертикальної (або магістральної) підсистеми, що з'єднує поверхи будівлі, повинен передавати дані на більші відстані й з більшою швидкістю у порівнянні з кабелем горизонтальної підсистеми. Найчастіше для вертикальних підсистем використовується оптоволоконий кабель.

Для вертикальної підсистеми вибір кабелю в цей час обмежується трьома варіантами.

– Оптоволокно – відмінні характеристики пропускної здатності, відстані й захисти даних; стійкість до електромагнітних перешкод; може передавати голос, відеозображення й дані. Але порівняно дорого, складно виконувати відгалуження.

– Товстий коаксіал – гарні характеристики пропускної здатності, відстані й захисти даних; може передавати дані, але з ним складно працювати.

– Широкополосний кабель, використовуваний у кабельному телебаченні, – гарні показники пропускної здатності й відстані; може передавати голос, відео й дані. Але дуже складно працювати й потрібні більші витрати під час експлуатації.

Застосування волоконо-оптичного кабелю у вертикальній підсистемі має ряд переваг:

– Передає дані на значно більші відстані без необхідності регенерації сигналу.

– Має сердечник меншого діаметра, тому може бути прокладений у більше вузьких місцях.

– Передані по ньому сигнали є світловими, а не електричними, тому оптоволоконий кабель не чутливий до електромагнітних і радіочастотних перешкод, на відміну від мідного коаксіального кабелю. Це робить оптоволоконий кабель ідеальним середовищем передачі даних для промислових мереж.

– Оптоволоконому кабелю не страшна блискавка, тому він гарний для зовнішньої прокладки.

– Забезпечує більше високий ступінь захисту від несанкціонованого доступу, тому що відгалуження набагато легше виявити, чим у випадку мідного кабелю (при відгалуженні різко зменшується інтенсивність світла).

Оптоволоконий кабель має й недоліки:

– Дорожче чим мідний кабель, дорожче обходиться і його прокладка.

– Оптоволоконий кабель менш міцний, чим коаксіальний.

– Інструменти, застосовувані при прокладці й тестуванні оптоволоконого кабелю, мають високу вартість і складні в роботі.

Вибір типу кабелю для підсистеми ЦОД

Як і для вертикальних підсистем, оптоволоконий кабель є найкращим вибором для підсистем декількох будівель, розташованих у радіусі декількох кілометрів. Для цих підсистем також підходить товстий коаксіальний кабель.

При виборі кабелю для ЦОД потрібно враховувати вплив середовища на кабель поза приміщенням. Для запобігання поразки блискавкою краще вибрати для зовнішньої проводки неметалічний оптоволоконий кабель. З багатьох причин зовнішній кабель виробляється в поліетиленовій захисній оболонці високої

щільності. При підземній прокладці кабель повинен мати спеціальну вологозахистну оболонку (від дощу й підземної вологи), а також металевий захисний шар від гризунів і вандалів. Вологозахистний кабель має прошарок з інертного газу між діелектриком, екраном і зовнішньою оболонкою. Кабель для зовнішньої прокладки не підходить для прокладки усередині будівель, тому що він виділяє при згорянні велика кількість диму.

3.3 Розробка функціональної схеми

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

– Статистика подій системи Legrand Cabling System 3 для ЦОД.

– Робота з ресурсами системи Legrand Cabling System 3 для ЦОД.

– Робота з сесіями системи Legrand Cabling System 3 для ЦОД.

– Функції для роботи з центрами обробки даних, побудованих з використанням системи Legrand Cabling System 3 для ЦОД.

– Моніторинг трафіку системи Legrand Cabling System 3 для ЦОД.

– Робота з файлами системи Legrand Cabling System 3 для ЦОД.

– Монітор з'єднань системи Legrand Cabling System 3 для ЦОД.

Розглянемо детальніше кожний з блоків.

Робота з ресурсами системи Legrand Cabling System 3 для ЦОД включає в себе:

– Визначення доступних ресурсів ЦОД.

– Закриття локального ресурсу ЦОД.

– Відкриття локального ресурсу ЦОД.

– Приховання й показ ресурсів ЦОД.

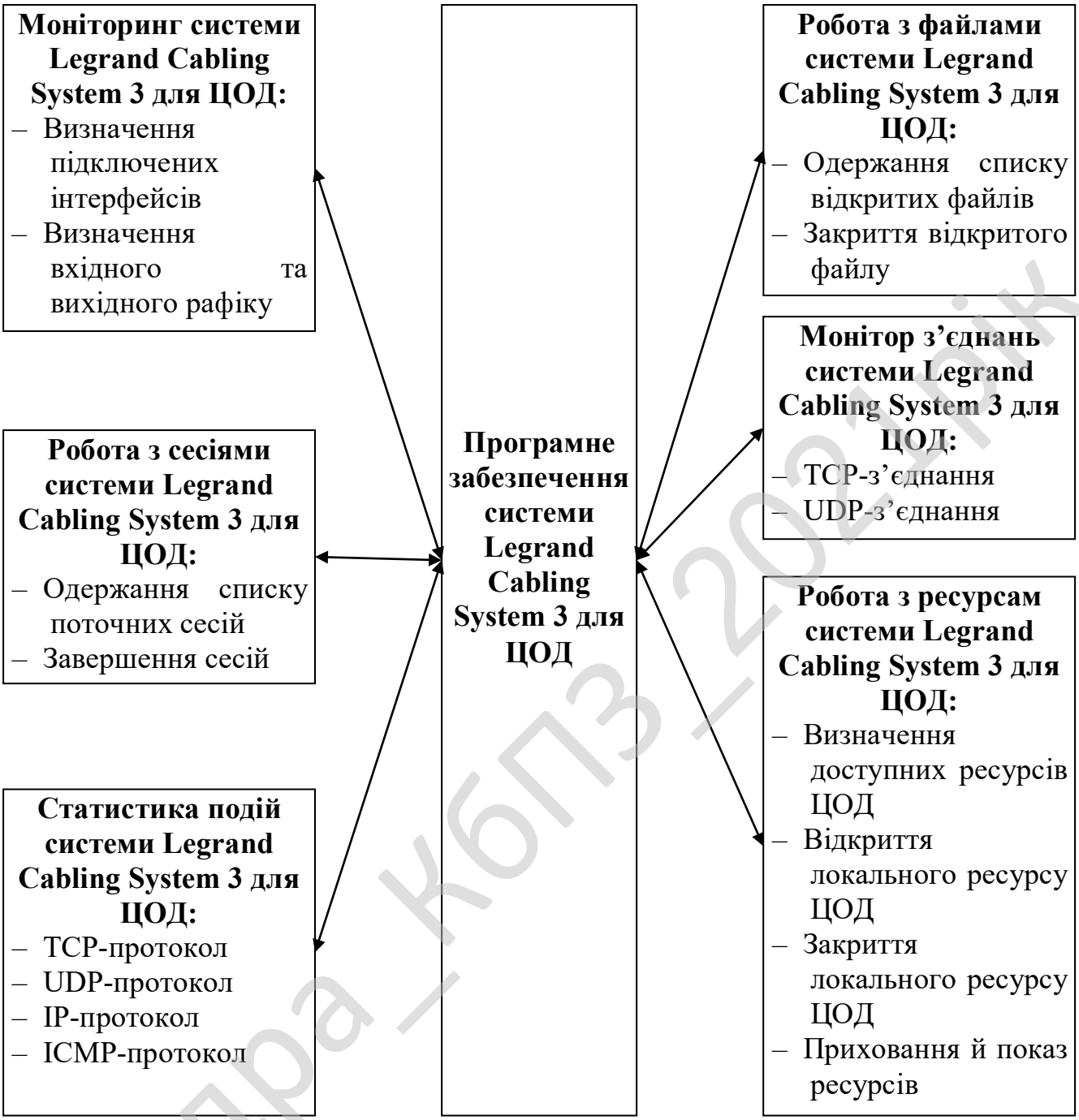

Рисунок 3.2 – Функціональна схема системи

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

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

Робота з сесіями системи Legrand Cabling System 3 для ЦОД вклю чає в себе:

– Одержання списку поточних сесій.

– Завершення сесій.

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

Моніторинг трафіку системи Legrand Cabling System 3 для ЦОД включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку.

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

Робота з файлами системи Legrand Cabling System 3 для ЦОД включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

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

Монітор з'єднань системи Legrand Cabling System 3 для ЦОД включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

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

Статистика подій системи Legrand Cabling System 3 для ЦОД включає в себе відстеження подій в наступних протоколах:

– TCP-протокол.

– UDP-протокол.

– IP-протокол.

– ICMP-протокол.

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

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

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

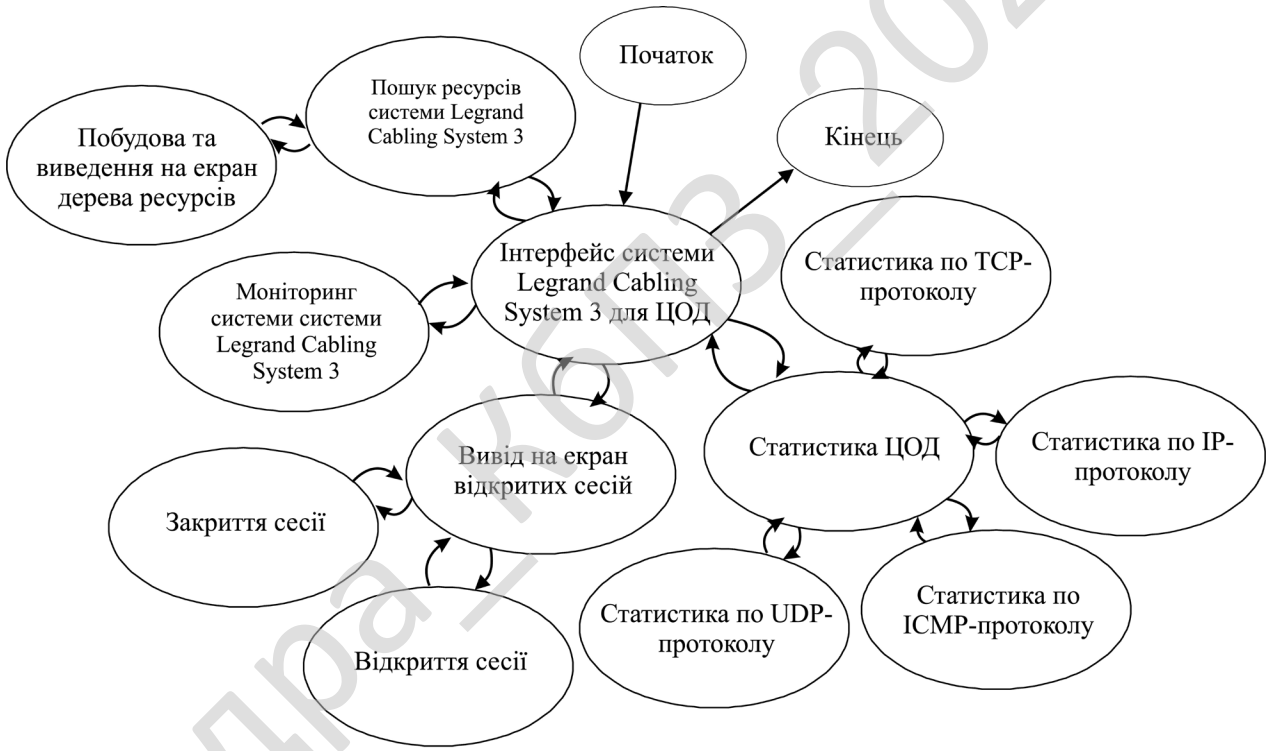

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми 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.

Розглянемо формат що використовується – **JSON** (JavaScript Object Notation, укр. запис об'єктів JavaScript, вимовляється джейсон) – це текстовий формат обміну даними між комп'ютерами.

JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних.

Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером.

При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

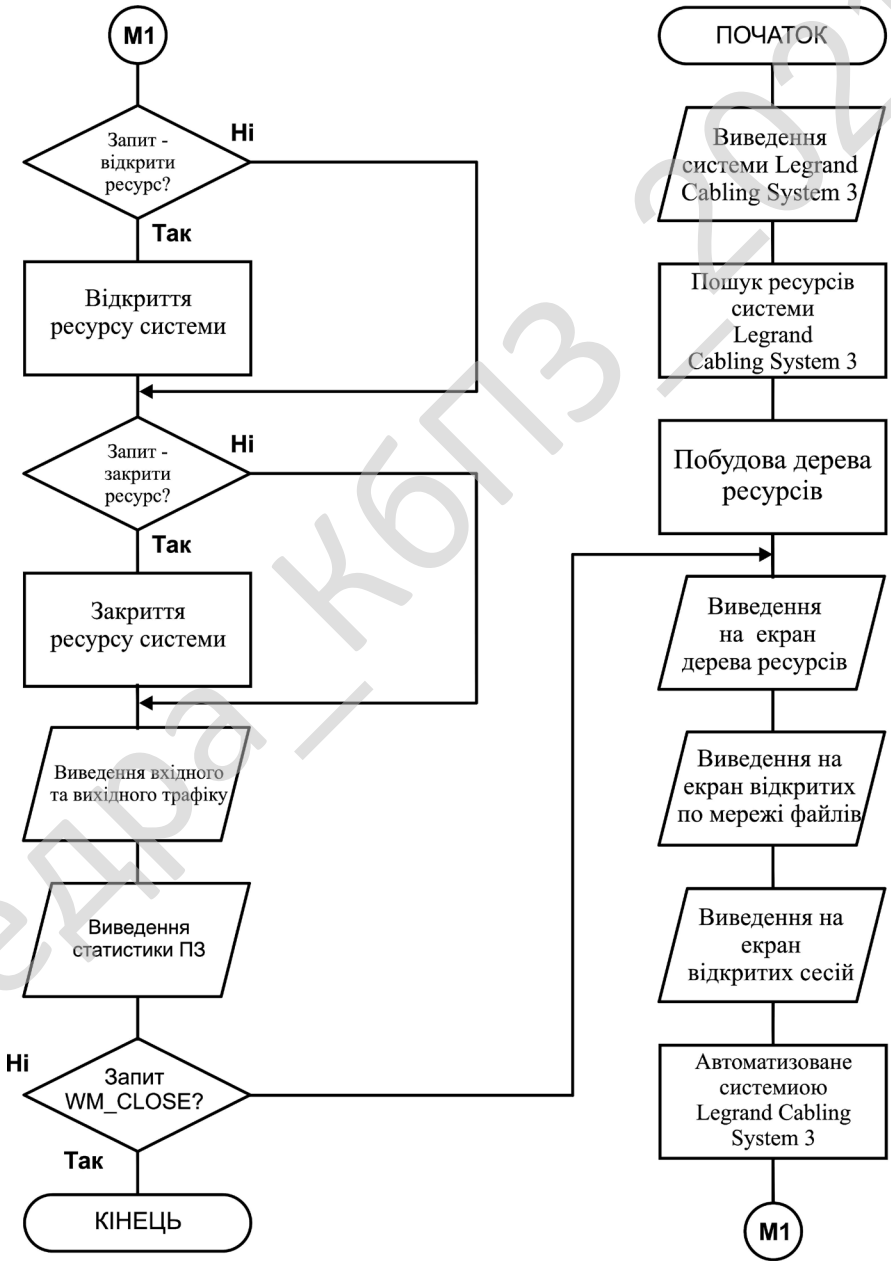

Рисунок 4.1 – Блок-схема основної програми

JSON з'явився через необхідність обміну даними з сервером у реальному часі без використання плагінів для браузерів, flash-додатків або Java-аплетів, які використовувались скрізь на початку 2000-х років.

Дуглас Крокфорд був тим, хто активно просував новий на той час формат. Він з колегами хотів створити технологію, яка використовувала б можливості звичайного браузера давала б веб-розробникам можливість створювати веб-додатки із постійним двостороннім зв'язком із веб-сервером.

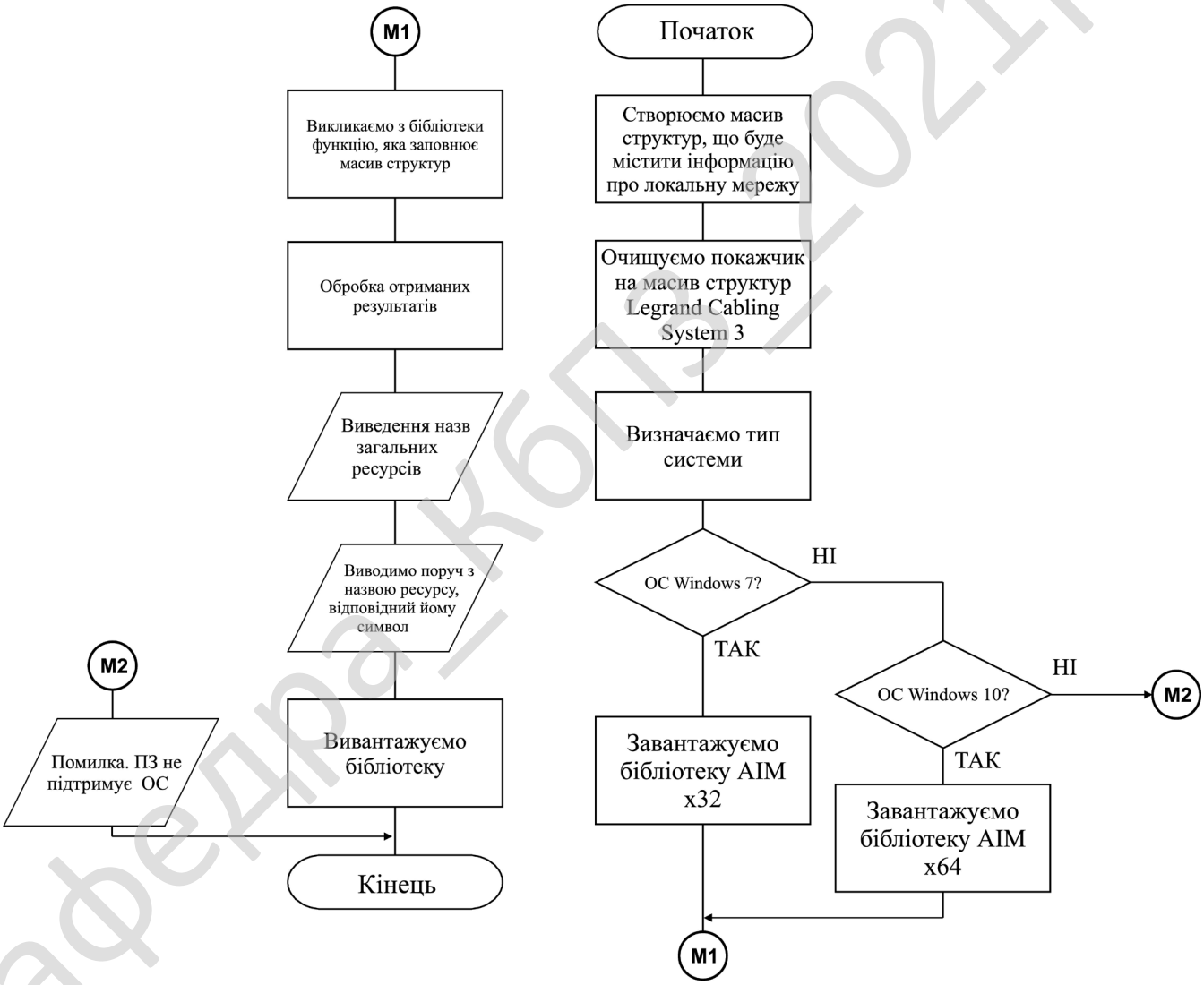

Рисунок 4.2 – Блок-схема роботи підпрограми

JSON вперше був використаний в проекті в Communities.com для Cartoon Network, він дозволяв обмінюватись повідомленнями і одночасно маніпулювати DHTML-елементами.

Веб-сайт JSON.org було запущено 2002 року. У грудні 2005-го року Yahoo! почав переводити деякі зі своїх веб-сервісів на роботу з JSON. Google взялася до роботи з технологією у своєму веб-протоколі GData у грудні 2006-го року.

Використання

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси).

Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

Приклад використання JSON (JavaScript)

```
const ajaxObj = JSON.parse(ajaxData)
console.log(`${ajaxObj.name}_${ajaxObj.rates[2]}`)
```

Синтаксис

JSON будується на двох структурах:

1. Набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом.

2. Впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

Це універсальні структури даних. Теоретично всі сучасні мови програмування підтримують їх у тій чи іншій формі.

Оскільки JSON використовується для обміну даними між різними мовами програмування, то є сенс будувати його на цих структурах.

У JSON використовуються такі їхні форми:

1. Об'єкт – це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

2. Масив – це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

3. Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

4. Рядок – це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски \. Символи представляються простим рядком. Тип Рядок (String) дуже схожий на String в мовах C і Java.

Число теж дуже схоже на C- або Java-число, за винятком того, що вісімкові та шістнадцяткові формати не використовуються. Пропуски можуть бути вставлені між будь-якими двома лексемами.

Наступний приклад показує JSON представлення об'єкта, що описує людину. У об'єкті є рядкові поля імені і прізвища, об'єкт, що описує адресу, і масив, що містить список телефонів.

```
{  "firstName": "Іван",
   "phoneNumbers": [
       "044 123-1234",
       "050 123-4567"   ]}
```

Використання JSON в AJAX

Наступний фрагмент коду JavaScript показує, як клієнт може використати XMLHttpRequest для запиту об'єктів в форматі JSON з серверу. (Серверна частина коду пропущена, вона просто повертає на запит URL рядок у JSON форматі).

Треба відзначити, що тут використання XMLHttpRequest не є кросс - браузерним (за деталями звертайтеся на сторінку XMLHttpRequest), і код зазнаватиме незначних модифікацій в різних версіях оглядачів Internet Explorer, Opera, Safari або Mozilla. Використання запиту XMLHttpRequest обмежено

правилом одного джерела (same origin policy): URL, що відповідає на запит, має посилатися на той же сайт, що обслуговує сторінку, що ініціювала запит.

Оглядачі можуть також використовувати тег <iframe> для асинхронного запиту JSON-даних в кросс-браузерному варіанті, або використати просте перенаправлення <form action="url_to_cgi_script" target="name_of_hidden_iframe">. Такий підхід був поширений до приходу популярного нині запиту XMLHttpRequest.

Динамічний тег <script> також можна використати для підвантаження JSON-даних. Ця техніка можлива, щоб обійти надто суворе правило одного джерела, але вона не є безпечною. Запит JSONRequest пропонується, як безпечніша альтернатива.

Питання безпеки

Хоча JSON призначений для передачі даних в серіалізованому вигляді, його синтаксис відповідає синтаксису JavaScript і це створює низку проблем безпеки. Часто для обробки даних, отриманих від зовнішнього джерела у форматі JSON, до них застосовується функція eval() без якої -небудь попередньої перевірки.

JavaScript eval()

Оскільки JSON представляється синтаксично правильним фрагментом коду JavaScript, природним способом розбору JSON-даних в JavaScript-програмі є використання вбудованої в JavaScript функції eval(), яка призначена для обчислення JavaScript-виразів. При цьому підході відпадає необхідність у використанні додаткових парсерів.

Техніка використання eval() робить систему вразливою, якщо джерело JSON-даних, що використовуються, не відноситься до надійних. Такими даними може виступати шкідливий JavaScript-код для атак за допомогою ін'єкції коду. За допомогою цієї вразливості можливо здійснювати крадіжку даних, підробку автентифікації. Проте, вразливість можна усунути за рахунок використання додаткових засобів перевірки даних на коректність. Наприклад, до виконання

eval() отримані від зовнішнього джерела дані можуть перевірятися за допомогою регулярних виразів. У RFC, що визначає JSON пропонується використовувати такий код для перевірки його відповідності формату JSON

```
const my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"(\\.|[^"\\])*"/g, '''))) &&
eval('(' + testedData + ')');
```

Як безпечніша альтернатива eval() була запропонована нова функція parseJSON(), здатна обробляти тільки JSON-дані.

Вбудований JSON

Останні версії веб-браузерів мають вбудовану підтримку JSON і здатні його обробляти без виклику функції eval(), що призводить до описаної проблеми. Обробка JSON у такому разі зазвичай здійснюється швидше. Так у червні 2009 року вбудовану підтримку JSON мали такі браузери: Mozil la Firefox 3.5+; SeaMonkey 2; Thunderbird 3; Microsoft Internet Explorer 8; Opera 10.5; Браузери, засновані на WebKit (наприклад, Google Chrome, Apple Safari).

Принаймні дві популярні бібліотеки JavaScript використовують вбудований JSON у разі його доступності: jQuery; Dojo.

Підробка крос-доменного запиту

Непродумане використання JSON робить сайти вразливими до підробки міжсайтових запитів (CSRF або XSRF). Оскільки тег <script> допускає використання джерела, що не належить до того ж домену, що і використовуваний ресурс, це дозволяє виконувати код даних, представлених у форматі JSON, в контексті довільної сторінки, що робить можливою компрометацію паролів або іншої конфіденційної інформації користувачів, що пройшли авторизацію на іншому сайті.

Це є проблемою тільки у разі вмісту в JSON -даних конфіденційної інформації, яка може бути компрометована третьою стороною і якщо сервер розраховує на політику одного джерела, блокуючи доступ до даних при виявленні зовнішнього запиту.

Це не є проблемою, якщо сервер визначає допустимість запиту, надаючи дані тільки у разі його коректності. HTTP cookie не можна використовувати для визначення цього. Виключне використання HTTP cookie використовується підробкою міжсайтових запитів.

Розширення, JSONP

JSONP або «JSON з підкладкою» є розширенням JSON, коли назва функції зворотного виклику вказується як вхідний аргумент. Спочатку ідея була запропонована в блозі MacPython в 2005 році, і в наш час використовується багатьма Web 2.0 застосунками, такими, як Dojo Toolkit Applications, Google Toolkit Applications і zanox Web Services.

Подальші розширення цього протоколу були запропоновані з урахуванням введення додаткових аргументів, як, наприклад, у разі JSONPP за підтримки S3DB вебсервісів.

Оскільки JSONP використовує скрипт-теги, виклики по суті відкриті світові. З цієї причини JSONP може бути недоречними для зберігання конфіденційних даних.

Включення скриптових тегів від віддалених сайтів дозволяє їм передати будь-який контент на сайті. Якщо віддалений сайт має вразливості, які дозволяють виконати ін'єкції JavaScript, то початковий сайт також може бути ними зачеплений.

Розширення, BSON

BSON – це бінарна форма представлення простих структур даних і асоціативних масивів (які називають об'єктами або документами). Назва «BSON» заснована на визначенні JSON і неофіційно означає «Binary JSON» (бінарний JSON).

JSON Reference

Стандарт JSON не описує посилання на інші об'єкти або частини, але існує чернетка стандарту IETF для посилань на об'єкти на основі JSON. Посилання дозволяють здійснювати трансклюзію – вставляти одні документи в інші.

JSON Reference – це JSON-об'єкт з ключем "$ref" (всі інші ключі ігноруються) і значенням стрічкового типу що містить URI, наприклад:

```
{ "$ref": "http://example.com/example.json#/foo/bar" }
```

Якщо URI містить ідентифікатор фрагмента (в прикладі вище "/foo/bar"), він інтерпретується як JSON Pointer.

Модуль dojox.json.ref в Dojo toolkit, забезпечує підтримку декількох форм JSON Reference.

Розглянемо протокол SNMP (Simple Network Management Protocol, простий протокол керування мережею) – це протокол керування мережами зв'язку на основі архітектури TCP/IP.

На основі концепції TMN в 1980–1990 р. різними органами стандартизації був вироблений ряд протоколів керування мережами передачі даних з різним спектром реалізації функцій TMN. До одного з типів таких протоколів керування належить Simple Network Management Protocol (SNMP).

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як сукупність мережних керуючих станцій й елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами. SNMP різних версій присвячений цілий ряд рекомендацій IETF (RFC).

Зазвичай при використанні SNMP присутні керовані та керуючі системи. До складу керованої системи входить компонент, який називається агентом, який відправляє звіти керуючій системі. По суті SNMP агенти передають управлінську інформацію на керуючі системи як змінні (такі як «вільна пам'ять», «ім'я системи», «кількість працюючих процесів» тощо).

Керуюча система може отримати достовірну інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту надсилати дані, використовуючи операцію протоколу TRAP або INFORM.

Управляючі системи можуть також відправляти конфігураційні оновлення або контролюючі запити, використовуючи операцію SET для безпосереднього управління системою. Операції конфігурування та управління використовуються тільки тоді, коли потрібні зміни у мережній інфраструктурі. Операції моніторингу зазвичай виконуються на регулярній основі.

Змінні, доступні через SNMP, організовані в ієрархії. Ці ієрархії та інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації' (Management Information Bases (MIBs)).

SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base).

Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний адресний простір імен, що містить унікальний ідентифікатор об'єкта (object identifier (OID)).

Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP. MIB'и використовують нотацію, визначену в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого приписані різними організаціями. На найвищому рівні MIB OID'и належать різним організаціям, що займаються стандартизацією, в той час як на нижчих рівнях OID'и виділяються асоційованими організаціями.

Ця модель забезпечує управління на всіх шарах мережної моделі OSI, адже MIB'и можуть бути визначені для будь-яких типів даних і операцій.

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт включає в себе один або більше екземплярів об'єкта (що ідентіфікуються за OID), які насправді є змінними.

Існує два типи керованих об'єктів: Скалярні об'єкти визначають єдиний екземпляр об'єкта; Табличні об'єкти визначають множинні, пов'язані екземпляри об'єктів які групуються в таблицях MIB.

Прикладом керованого об'єкта може бути atInput, який є скалярним об'єктом що містить єдиний екземпляр об'єкта, ціле число, яке показує загальну кількість вхідних пакетів AppleTalk на мережний інтерфейс маршрутизатора.

Ідентифікатор об'єкта (OID) унікально ідентифікує керований об'єкт в ієрархії MIB.

В телекомунікаціях і комп'ютерних мережах, ASN.1 є стандартною гнучкою нотацією для опису структур даних, що служать для кодування, передачі і декодування даних. ASN.1 являє собою набір правил для опису структури об'єктів, незалежних від специфічних для обладнання методик кодування, і формальну нотацію, яка дозволяє уникнути неоднозначностей.

ASN.1 це єдиний ISO та ITU-T стандарт, спочатку визначений у 1984 році як частина стандарту CCITT X.409: 1984. Пізніше, в 1988 році, завдяки його широкому застосуванню, він був виділений в окремий стандарт X.208. Значно переглянута версія 1995року описана в X.680.

Адаптована підмножина ASN.1 Структура Керуючої Інформації (SMI) описана в протоколі SNMP для визначення наборів пов'язаних MIB об'єктів, також званих MIB модулями.

SNMP працює на прикладному рівні TCP/IP (сьомий рівень моделі OSI). Агент SNMP отримує запити по UDP-порту 161.

Менеджер може посилати запити з будь-якого доступного порту джерела на порт агента. Відповідь агента буде відправлений назад на порт джерела на менеджері.

Менеджер отримує повідомлення (Traps і InformRequests) по порту 162. Агент може генерувати повідомлення з будь-якого доступного порту. При використанні TLS або DTLS запити виходять по порту 10161, а пастки відправляються на порт 10162.

У SNMPv1 зазначено п'ять основних протокольних одиниць обміну (protocol data units – PDU). Ще дві PDU, GetBulkRequest і InformRequest, були введені в SNMPv2 і перенесені в SNMPv3.

Нижче перераховані сім протокольних одиниць обміну SNMP:

1. GetRequest. Запит від менеджера до об'єкту для отримання значення змінної або списку змінних. Необхідні змінні вказуються в полі variable bindings (розділ поля values при цьому не використовується). Отримання значень зазначеної змінної повинно бути виконано агентом як Атомарна операція. Менеджеру буде повернений Response (відповідь) з поточними значеннями.

2. SetRequest. Запит від менеджера до об'єкту для зміни змінної або списку змінних. Зв'язані змінні вказуються в тілі запиту. Зміни всіх зазначених змінних повинні бути виконані агентом як атомарна операція. Менеджеру буде повернений Response з (поточними) новими значеннями змінних.

3. GetNextRequest. Запит від менеджера до об'єкту для виявлення доступних змінних і їх значень. Менеджеру буде повернений Response зі зв'язаними змінними для змінної, яка є наступною в базі MIB в лексіграфіческом порядку. Обхід всієї бази MIB агента може бути проведений ітераційним використанням GetNextRequest, починаючи з OID 0. Рядки таблиці можуть бути прочитані, якщо вказати в запиті OID-и колонок в пов'язаних змінних.

4. GetBulkRequest. Покращена версія GetNextRequest. Запит від менеджера до об'єкту для численних ітерацій GetNextRequest. Менеджеру буде повернений Response з декількома пов'язаними змінними, обійденими починаючи зі пов'язаної змінної (змінних) в запиті. Специфічні для PDU поля non-repeaters і max-repetitions використовуються для контролю за поведінкою відповіді. GetBulkRequest був введений в SNMPv2.

5. Response. Повертає зв'язані змінні і значення від агента менеджеру для GetRequest, SetRequest, GetNextRequest, GetBulkRequest і InformRequest. Повідомлення про помилки забезпечуються полями статусу помилки і індексу помилки.

Ця одиниця використовується як відповідь і на Get-, і на Set-запити, в SNMPv1 називається GetResponse.

6. Trap. Асинхронне повідомлення від агента – менеджеру. Включає в себе поточне значення sysUpTime, OID, що визначає тип trap (пастки), і необов'язкові зв'язані змінні. Адресація одержувача для пасток визначається за допомогою змінних trap-конфігурації в базі MIB. Формат trap -повідомлення був змінений в SNMPv2 і PDU перейменували в SNMPv2-Trap.

7. InformRequest. Асинхронне повідомлення від менеджера менеджеру або від агента менеджеру. Повідомлення від менеджера менеджеру були можливі вже в SNMPv1 (за допомогою Trap), але SNMP зазвичай працює на протоколі UDP, в якому доставка повідомлень не гарантована, і не повідомляється про втрачені пакетах. InformRequest виправляє це зворотним відправленням підтвердження про отримання. Одержувач відповідає Response-му, що повторює всю інформацію з InformRequest. Цей PDU був введений в SNMPv2.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного.

Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся лог іка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими

клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може

здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P-1)(Q-1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\mathrm{mod}\ \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \pmod N$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \pmod N$.

Крім того, він знаходить результат гешування прийнятого повідомлення *M* с допомогою такої ж геш-функції h(-): $m = h(M)$.

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (*M*, *S*) справжньою. Доведено, що тільки власник секретного ключа *D* може сформувати цифровий підпис *S* по документі *M*, а визначити секретне число *D* по відкритому числу *E* не легше, ніж розкласти модуль *N* на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису *S* буде позитивним тільки в тому випадку, якщо при обчисленні *S* був використаний секретний ключ *D*, що відповідає відкритому ключу *E*. Тому відкритий ключ Е іноді називають "ідентифікатором" того, хто підписав.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі система Legrand Cabling System 3 для ЦОД. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функціональних кнопок ПЗ; Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Розділу обрання групи; Розділу виведення результату роботи системи.

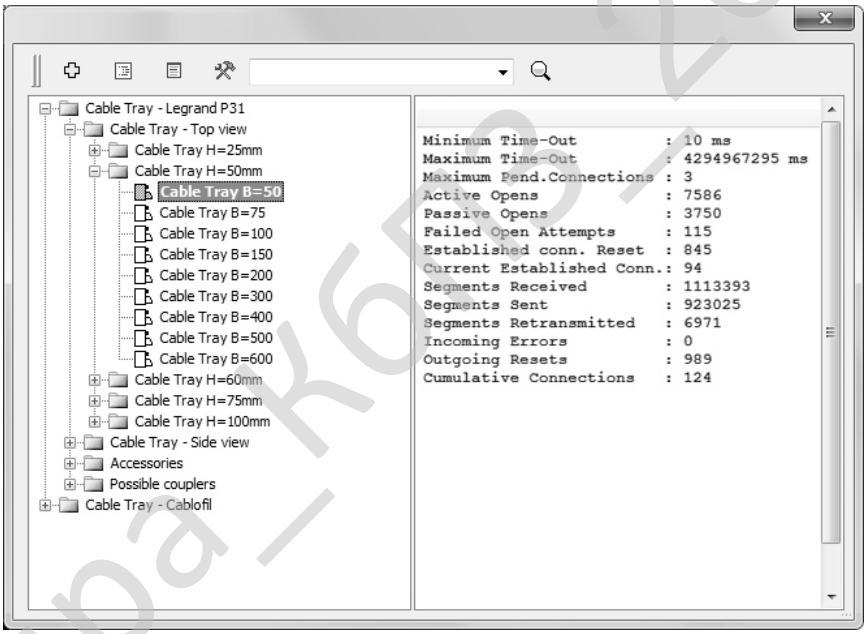

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких

помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

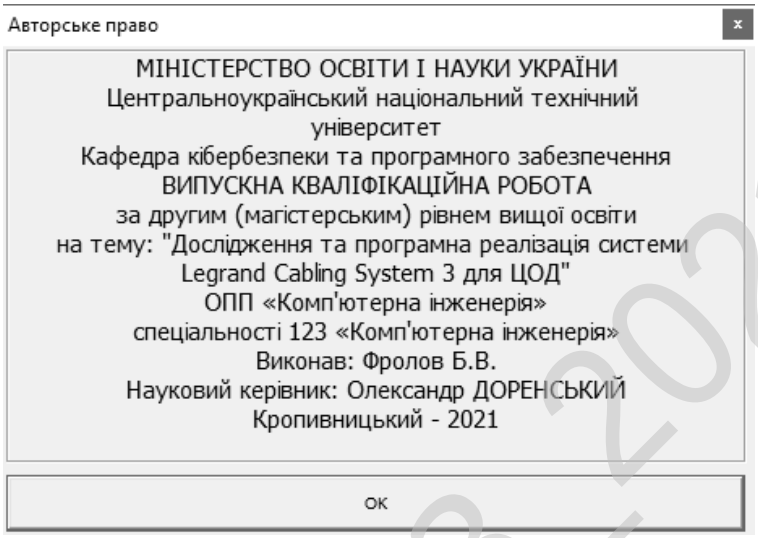

Рисунок 5.2 – Авторське право

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТ.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – commercial software. Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм.

Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям.

Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм.

Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником.

Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Legrand Cabling System 3 для ЦОД.

Метою розробки є дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД.

Об'єктом дослідження є процес Legrand Cabling System 3 для ЦОД.

Предметом дослідження є методи Legrand Cabling System 3 для ЦОД.

Методи дослідження базуються на методах теорії побудови комп'ютерних систем та мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод Legrand Cabling System 3 для ЦОД.

– Розроблено вітчизняний продукт Legrand Cabling System 3 для ЦОД , 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 24 днів (один місяць). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи Legrand Cabling System 3 для ЦОД.

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	250 (2 ост. цифри № зал*10^1)
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	В
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
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)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
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)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	1
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	1
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	1
32. Вартість ПЗ у розробника (НМА), грн.	–	260000
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	40
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.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

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.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}24 \cdot 1{,}25 \cdot 1{,}29) = 26 \text{ люд-міс.}$$

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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 26^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 100 = 236 \text{ люд/день.}$$

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	15	Д7
Робочий проект	236	Ф 7.1-7.4
Впровадження	15	Д13
Всього	285	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{285 \cdot 1}{24\text{-}3} = 13,6 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор – маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	70	175	3
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	1	285	5
Усього за рік:			$З_ч$	221

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{221 \cdot 1}{1,2} = 184,1 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 184,1/(24·8) = 1 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 2012 R2, 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	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	2	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		4	
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	18500	18500
Продакт-менеджер	0,5	16500	8250
Інженер-програміст	13,6	17000	231200
Інженер-електронщик	1	16000	16000
Інженер-системотехнік	0,25	16000	4000
Адміністратор мережі	0,5	16000	8000
Системний програміст	0,25	16000	4000
Дизайнер WEB	0,5	17000	8500
Інженер-верстальник	0,25	16000	4000
Бухгалтер-економіст	0,5	17500	8750
Всього за період розробки	$R_{сп}$ = 18,35	-	$Ф_{роб}$ = 311200

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.

$$З_{сд} = \frac{311200}{18,35 \cdot 24} = 706 \; грн.$$

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 12 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, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 12 \cdot 8 \cdot 20000 = 1920000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 192000 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.

$$І_{нв} = 12 \cdot 3500 = 42000 \text{ грн.}$$

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 за прайсом фірми Brain за 14.11.21 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11186
Системний блок		6490
Процесор	AMD Athlon 200GE (4M Cache, 3.2 GHz)	–
Системна плата	MSI A320M-A Pro (sAM4, AMD A320, PCI Ex16) 1 x HDMI, 1 x VGA, 1 x USB 3.0, 4 USB 2.0, 4 x USB 3.1, 1 x PS/2, Audio, 1 RJ45	–
Відеокарта	Вбудована AMD Radeon Vega 3	–
Жорсткий диск	SSD 120Gb + HDD 500 Gb WD 7200 64M SATAIII	–
Оперативна пам’ять	4Gb, DDR4, 2133 MHz, PC4-17000	–
DVD-привод	Super Multi LG SATA DVD±RW R+22x/ 22x, RW+8x/-6x, DL+16x/-12x, RAM 12x SecurDisc, black (GH22NS40RBB)	–
Корпус	Vinga Smart-400W FSP (Black/Silver panel ATX (90-PL861AF5C4-53CZ), Матеріал корпусу пластик, метал. Розташуванн блоку живлення – нижнє	–
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	–

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3200
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1496

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – 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	12	11186	13423,2	147655,2
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050

Продовження таблиці 7.7

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	169473,7

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	1920000	-	-
2. Передавальні пристрої	192000	-	-
Всього по групі	2112000	5	105600
Група 4			
3. Обчислювальна техніка	169474	-	-
Всього по групі	169474	50	84737
Нематеріальні активи			
4. Нематеріальні активи	260000	10	26000

Продовження таблиці 7.8

1	2	3	4
Група 5, 6			
5. Вимірювальні пристрої	5190	25	1297,5
6. Транспортні засоби	72500	20	14500
7. Господарський інвентар	42000	25	10500
Всього по групі	119690	-	26297,5
Разом	$К_р$ = 2661164		$А_р$ = 242634,5

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 2900 USD, що враховуючи прийнятий для розрахунку курс 25 складає 72500 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.

$$З_о = 706 \cdot 285/250 = 805 \text{ грн.}$$

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, %.

$$З_д = 805 \cdot 10 \cdot 0{,}01 = 81 \text{ грн.}$$

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

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

де: H_c – відрахування на соціальні потреби, %.

$$C_{оц} = 0{,}01 \cdot 22(805+81) = 195 \text{ грн.}$$

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

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

де: $H_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 805 \cdot 15 \cdot 0{,}01 = 121 \text{ грн.}$$

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

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

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам’ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 80 грн., визначаємо вартість паперу за період розробки $N_м$ = 1 міс:

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

$$З_{М1} = 80 \cdot 0{,}5 \cdot 1 = 40 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 13 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 13 грн./шт.

$$З_{М2} = 250 \cdot 13 + 13 = 3263 \text{ грн.}$$

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

$$З_{МЗ} = \sum Ц_{з}.,\quad (7.18)$$

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.

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

$$З_{М} = (40 + 3263 + 1702)/250 = 20 \text{ грн.}$$

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, %.

$$О_п = 805 \cdot 15 \cdot 0{,}01 = 121 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 250 прим.):

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

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

$$А_м = 242634 \cdot 1/(250 \cdot 12) = 81 \text{ грн.}$$

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

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

$$С_п = 805+81+195+121+20+121+81 = 1424 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 40%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п,\quad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 40 \cdot 1424 = 570 \text{ грн.}$$

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	$З_о$	805
2. Додаткова зарплата виконавців	$З_д$	81
3. Відрахування на соціальні потреби	$С_{оц}$	195
4. Загальногосподарські витрати	$Г_{осп}$	121
5. Витрати на матеріали	$З_М$	20
6. Освоєння нових операційних систем, мов програмування	$О_п$	121
7. Амортизація основних фондів	$А_м$	81
8. Повна собівартість програмного забезпечення	$С_п$	1424
9. Плановий прибуток	$П_р$	570
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	1994
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	399
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	2393

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	–	2393
Всього капітальних витрат	–	2393

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	12078	7247
2. Витрати на електроенергію	$З_{ел}$	7600	7220
3. Витрати на амортизацію	$З_{ам}$	0	598
Всього витрат за рік	$І$	19678	15065

Витрати на профілактичні роботи:

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

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

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 250 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 250 \cdot 36 \cdot 1{,}1 \cdot 1{,}22 = 12078\ грн,$$

до:

$$З_{р\ нов} = 150 \cdot 36 \cdot 1{,}1 \cdot 1{,}22 = 7247\ грн.$$

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

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

$$З_{ел\ баз} = 0{,}5 \cdot 8000 \cdot 1{,}9 = 7600 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}5 \cdot 0{,}95 \cdot 8000 \cdot 1{,}9 = 7220 \text{ грн.}$$

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	–	2393	–	598,25
Всього відрахувань	-	–	2393	–	598,25

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.

$$Ев = (1994 - 1424) \cdot 250 - (0,05 \cdot 2112000 + 0,5 \cdot 169474 +$$
$$+0,25 \cdot 47190 + 0,2 \cdot 72500 + 0,1 \cdot 260000) \cdot 1/12 = 122280 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{549164}{(1994 - 1424) \cdot 250 \cdot 12 / 1} = 0,32 \text{ року}.$$

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.

$$E_{сп} = (19678\text{-}15065)\text{-}0,25 \cdot 2393 = 4015 \text{ грн.}$$

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

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

$$T_{сп} = \frac{2393}{19678 - 15065} = 0,5 \text{ роки}.$$

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.	250
2. Повна собівартість розробленої програми	Грн.	1424
3. Ціна розробленої програми	Грн.	1994
4. Плановий прибуток від реалізації розробленої програми	Грн.	570
5. Рентабельність програмної продукції	%	40
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2661164
7. Загальний прибуток від реалізації програмної продукції	Грн.	142500
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	122280
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,32
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	2393
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4015
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,5

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

«З давніх давен людство приділяє прискіпливу увагу безпеці життя і охорони праці як її складової частини. Умови праці розглядали Арістотель (384-322 до н.е.) та Гіппократ (460-377 до н.е.)» [4].

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» [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 [2].

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 ic тотне 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.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними

терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	16
Довжина	6,4
Висота	3

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

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,8
Обсяг, V	$м^3$	не менше 20.0	20,5

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працюють 15 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годин у. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату

складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23,5	40-55	0,12
Тепла	23-25	50-70	0,1	23,5-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають нормам.

Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер BROTHER DCP-L2500DR, електродвигуни вентиляторів ЕОМ.

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

З 2019 року діють Державні будівельні норми україни “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою

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

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних 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 ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

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 Розрахункова частина

Початкові дані для розрахунку захисного штучного заземлення: опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*.

Для захисного штучного заземлення застосовуються ветикальні електроди з метелевого прутка діаметром 35 мм. (*D=35 мм.=0,035 м.)* довжиною *L*=2 *м.* та горизонтальний електрод – метелева полоса з перетином 45*5 мм. тип грунта – суглинок (питомий опір 100 Ом*м). Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*.

Глибина закладення горизонтального контура заземлення *t= 0,6 м*.

Умовна товщина верхнього шару грунта: *H=0,35 м*. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр. 13-16 [6], або аналогічної.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+2/2=1,6 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1,36*100=136 \text{ Ом*м.}$$

де

$\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [8];

$\rho_1 = 50$ *Ом*м.* – табличне значення питомого опору верхнього шару грунта [6];

$\rho_2 = 100$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта [8].

Опір розтіканню електричного струму одного електрода вертикального заземлювача [8]:

$$R_o = 0,366\frac{\rho}{L}(lg\frac{2L}{D} + \frac{1}{2}lg\frac{4T+L}{4T-L}) =$$

$$= 0,366\frac{136}{2}(lg\frac{2\cdot 2}{0,035} + \frac{1}{2}lg\frac{4\cdot 1,6+2}{4\cdot 1,6-2}) = 54,7 \text{ Ом.}$$

Відношення $A/L=3/2=1,5$.

Визначаємо коефіцієнт екранування вертикальньних електродів $Кев=0,8$ при попередній (орієтовній) кількості ветикальньних електродів, яке дорівнює 10 [8].

Визначаємо необхідну кількість вертикальних заземлювачів (без вхарування горизонтального електрода штучного заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_O /(Кев\ R_{ЗН})= 54,7 / (0,8*4)=17,1 \text{ шт.}$$

Враховуючи те, що горизонтальний електрод (з'єднуюча метелева полоса) буде зменшувати загальний опір розтіканню електричного струму штучного заземлювача, приймаємо орієнтовну кількість вертикальних заземлювачів 15 шт.

Визначаємо довжину з'єднуючої полоси [8]:

$$L_П = 1,05*A*N = 1,05*3*15 = 47,2 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси [8]:

$$R_П = 0,366(\rho_2 * K_П / L_П) lg(2(L_П{}^2)/(К*t)) =$$

$$= 0,366(100*5/47,2)*lg(2*47,2^2)/(0,045*0,6) = 19 \text{ Ом.}$$

де $K_П = 5$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони з'єднуючої полоси: [8].

Загальньний опір розтіканню електричного струму заземлювача [8]:

$$R = (R_О * R_П)/(R_О * \eta_П + N * R_П * Кев) =$$

$$= (54,7 * 19)/(54,7 * 0,6 + 15 * 19 * 0,8) = 3,98 \text{ Ом.}$$

де $\eta_П = 0,6$ – табличне значення коефіцієнта екранування з'єднуючої полоси [8].

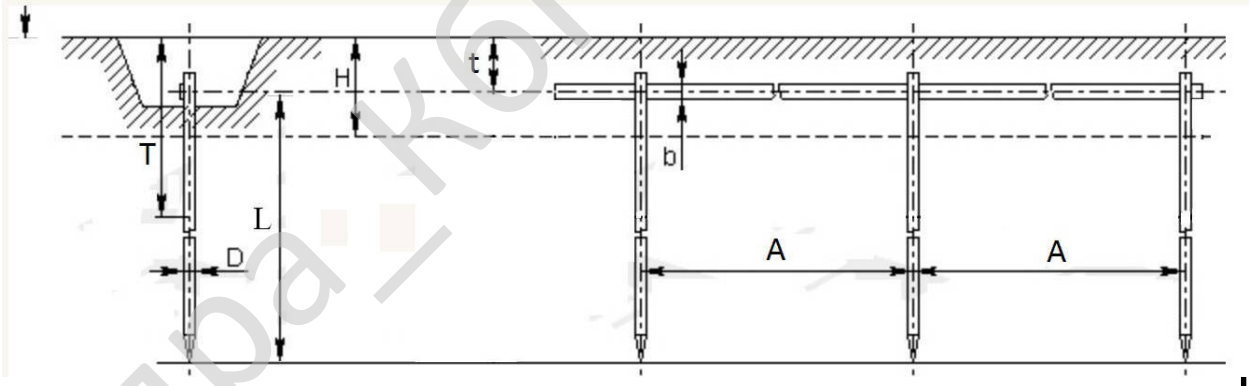

Рисунок 8.1 – Схема штучного заземлення

Умова $R \leq R_{ЗН}$ виконується ($3,98 \leq 4$).

Остаточно приймаємо кількість вертикальних заземлювачів N=15 шт.

8.6 Висновки до розділу

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи Legrand Cabling System 3 для ЦОД.

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 Legrand Cabling System 3 для ЦОД.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Legrand Cabling System 3 для ЦОД.

– Досліджена система Legrand Cabling System 3 для ЦОД.

– На основі отриманих результатів досліджень створена програмна реалізація системи Legrand Cabling System 3 для ЦОД.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Legrand Cabling System 3 для ЦОД.

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RSA.

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 4015 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,5 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фролов Б.В. Дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Microsoft Corporation. Межсетевое взаимодействие. Ресурсы Microsoft Windows 2000 Server. /Пер, с англ. – М.: Издательско-торговый дом "Русская Редакция*, 2002. – 736 с.: ил,

3. Microsoft Corporation. Разработка инфраструктуры сетевых служб Microsoft Windows 2000. Учебный курс MCSE. /Пер. с англ. – М: Издательско-торговый дом "Русская Редакция", 2001. – 992 стр.: ил.

4. Microsoft Corporation. Распределенные системы. Книга 1. Ресурсы Microsoft Windows 2000 Server. /Пер, с англ. – М.: Издательско-торговый дом "Русская Редакция", 2001. – 864 с.: ил.

5. Microsoft Corporation. Управление сетевой средой Microsoft Windows 2000. Учебный курс MCSA/MCSE. /Пер. с англ. – М.: Издательско-торговый дом "Русская Редакция". 2003. – 896 стр.: ил.

6. В.Г. Олифер, Н.А. Олифер. Компьютерные сети: Принципы, технологии, протоколы.

7. Селезнев Д.А. Построение локальной компьютерной сети масштаба малого предприятия на основе сетевой OS Linux. /МГИФИ, М., – 1999.

8. Терентьев А.М., Винокуров А.Е. Методы аудита локальных сетей в MS-DOS /Вопросы информационной безопасности узла Интернет в научных организациях. Сборник статей под ред. М.Д. Ильменского – М: ЦЭМИ РАН, 2001, с. 79 – 83.

9. Терентьев А.М. Методы и средства наблюдения загрузки локальных вычислительных сетей на примере ЦЭМИ РАН / Препринт #WP/2001/110 – М.: ЦЭМИ РАН, 2001. – 74 с.

10.Терентьев А.М. Задачи полноценного аудита корпоративных сетей. – «Концепции», N1(11), 2003, с.94-95.

11.Cisco Systems, Inc. Internetworking Technology Handbook, 4-th Edition. /Indianapolis: CiscoPress, September 2003 ISBN: 1587051192

12.Кочетова Н.А., Ляпичева Н.Г. Методы и средства защиты магистральных маршрутизаторов и серверов удаленного доступа производства Cisco Systems /Вопросы информационной безопасности узла Интернет в научных организациях. Сборник статей под ред. М.Д.Ильменского – М: ЦЭМИ РАН, 2001. – с.10-42 (Рус.)

13.Терентьев А.М. Информационная безопасность в крупных локальных сетях. – «Концепции», N1(9), 2002, с. 25-30.

14.Терентьев А.М. Построение и развитие системы сетевого мониторинга. / Развитие и использование средств сетевого мониторинга и аудита. Вып. 1. Сборник статей под ред. А.М.Терентьева – М.: ЦЭМИ РАН, 2004, с. 5-23.

15.Сторожук Д. О. Увеличение безопасности работы в локальной сети при использовании систем мониторинга / Гусева А. И., Сторожук Д. О. // Безопасность информационных технологий. – 2007, № 1. – с. 46-50.

16.Сторожук Д. О. Оптимизация по времени багатопоточной модели опроса компьютерной сети / Гусева А. И., Сторожук Д. О // Информационные технологии. – 2007, № 8. – с. 30-33.

17.Сторожук Д. О. Оптимизация по времени багатопоточной модели опроса сети / Гусева А. И., Сторожук Д. // XIY международный научно - технический семинар современные технологии в задачах управления, автоматики и обработки информации: сб. научных трудов – Алушта, 2005. – с. 49.

18.Сторожук Д. О. Оптимальное количество потоков в сети при мониторинге. сети / Сторожук Д. О. // XY международный научно-технический семинар современные технологии в задачах управления, автоматики и обработки информации: сб. научных трудов – Алушта, 2006. – с. 52.

19.Сторожук Д. О. Анализ аппаратного и программного обеспечения компьютера / Гусева А. И., Сторожук Д. О., Четвериков В. Н. // Научная сессия МИФИ– 2003: сб. научных трудов – М. МИФИ , 2003. – Том 2. – с. 99-100.

20.Сторожук Д. О. Применение информационных технологий для инвентаризации сетевого имущества на примере программного комплекса «ИнфраМенеджер»/ Гусева А. И., Сторожук Д. О., Четвериков В. Н. // Научная конференция МИФИ, М. МИФИ, 2005. – Том 2. – с. 131-132.

21.Сторожук Д.О. Использование системы моделирования GPSS World для сравнения различных реализаций системы мониторинга / Сторожук Д. О. // Научная сессия МИФИ-2007: сб. научных трудов – М. МИФИ , 2007. Том 2, стр. 106-107.

22.Сторожук Д.О. Оптимизация по времени багатопоточной модели опроса сети/ Сторожук Д. // Научная сессия МИФИ-2005: сб. научных трудов – М. МИФИ, 2005.– Том 2. – с. 133-134.

23.Сторожук Д.О. Основные задачи управления корпоративных сетей / Сторожук Д. О. // Научная сессия МИФИ-2007: сб. научных трудов – М. МИФИ , 2007.– Том 2. – с. 104-105.

24.Сторожук Д.О. Системные средства ОС для управления элементами сети и ресурсами / Сторожук Д. О. // Научная конференция МИФИ: сб. научных трудов – М. МИФИ, 2006. – Том 2. – с. 28-29.

25.Мохамад Гани Абу Таам Разработка математической GERT-модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / А.А.Смирнов, Мохамад Гани Абу Таам // Информационные системы в управлении, образовании, промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – 498 с.

26.Мохамад Гани Абу Таам Метод управления доступом в интеллектуальных узлах коммутации / Мохамад Гани Абу Таам, А.А.Смирнов // Информационные технологии и защита информации в

информационно-коммуникационных системах: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – 486 с.

27.Мохамад Гани Абу Таам Математическая GERT-модель технологии передачи метаданных в облачные антивирусные системы / В.В.Босько, А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Збірник наукових праць "Системи обробки інформації". – Випуск 1(117). – Х.: ХУПС – 2014. – С. 137-141.

28.Мохамад Гани Абу Таам Структурно-логическая GERT-модель технологии распространения компьютерных вирусов / А.А.Смирнов, И.А.Березюк, Мохамад Гани Абу Таам // Системи управління, навігації та зв'язку. – Випуск 1(29). – П.: ПНТУ. – 2014. – С. 120-125.

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

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

31.Мохамад Гани Абу Таам Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Гани Абу Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 4(17). – Харків: ХУПС. – 2014. – С.90-95.

32.Мохамад Гани Абу Таам Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Гани Абу

Таам, А.А. Смирнов, Н.С. Якименко, С.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 1(126). – Х.: ХУПС – 2015. – С. 150-153.

33.Мохамад Гани Абу Таам Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Системи озброєння і військова техніка. – Випуск 3(43) – Х.: ХУПС – 2015. – С. 100-107.

34.Мохамад Гани Абу Таам Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 3(19). – Х.: ХУПС. – 2015. – С. 134-141.

35.Mohamad Abou Taam Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

36.Мохамад Гани Абу Таам GERT-модель технологии передачи данных в облачные антивирусные системы / А.А. Смирнов, В.В. Босько, Мохамад Гани Абу Таам // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 18-19.

37.Мохамад Гани Абу Таам Математическое моделирование технологии передачи сигнатур в облачные антивирусные системы / Мохамад Гани Абу Таам, А.А. Смирнов // Збірник тез VI міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 260.

38.Мохамад Гани Абу Таам Анализ требований к качеству обслуживания в информационно-телекоммуникационных системах / А.А. Смирнов, Мохамад Гани Абу Таам // Збірник тез XVI міжнародного науково-практичного семінару

«Комбінаторні конфігурації та їх застосування». м. Кіровоград. 11-12 квітня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 124-126.

39. Мохамад Гані Абу Таам Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / Мохамад Гані Абу Таам, С.А. Смирнов // Збірник тез науково-практичної конференції «Інформаційні технології та комп'ютерна інженерія». м. Кіровоград. 4 грудня 2014 р. – Кіровоград: КНТУ. – 2014. – С. 168.

40. Мохамад Гани Абу Таам Исследование математических моделей технологии распространения компьютерных вирусов / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник наукових праць міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 25-28 лютого 2015 р. – Київ: Європейський університет. – 2015. – С. 90-91.

41. Мохамад Гани Абу Таам Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез всеукраїнської науково-практичної конференції «Інформаційна безпека держави, суспільства та особистості». м. Кіровоград. 16 квітня 2015. – Кіровоград: КНТУ. – 2015. – С. 50-52.

42. Мохамад Гани Абу Таам Разработка метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Гани Абу Таам, С.А. Смирнов // Збірник тез VII міжнародної науково-практичної конференції "Проблеми і перспективи розвитку ІТ-індустрії". м. Харків. 17-18 квітня 2015 р. – Харків: ХНЕУ. – 2015. – С. 14.

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

44. Мохамад Гани Абу Таам Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез II Міжнародної науково-практичної Інтернет-конференції «Інформаційна та економічна безпека» (INFECO-2015)». м. Харків. 21-22 травня 2015 р. – Харків: ХІБС УБС НБУ. – 2015. – С. 20-24.

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

46. Мохамад Гани Абу Таам Метод управления доступом к облачным телекоммуникационным ресурсам для обеспечения защиты данных / Мохамад Гани Абу Таам, А.А. Смирнов, С.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Комп'ютерні технології та інформаційна безпека». м. Кіровоград. 2-3 липня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 4-5.

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

48. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www. telecom61.ru›SharedFiles/Download.aspx? …pageid=106

49. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.:"Кудиц-Образ", 2003. – 528 с.

50. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. –2-е изд. – СПб.: Питер, 2007. – 958 с.

51. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

52. Свами М.Н., Тхуласираман К. Графы, сети и алгоритмы: пер. с англ. / М.Н. Свами, К. Тхуласираман; под ред. В.А. Горбатова. – М.: Мир, 1984. – 454 с.

53. Семенов С.Г. Анализ методов прогнозирования в телекоммуникационных сетях автоматизированных систем управления / С.Г.Семенов // Збірник наукових праць «Системи управління, навігації та зв'язку», – К.:ЦНДІ навігації і управління, – 2008.-Вип. 2(6) .- С.134-137

54. Семенов С.Г. Математическая модель процесса доставки информационных пакетов в компьютерной сети системы критического применения / С.Г.Семенов, И.В.Ильина // Науково-технічний журнал «Радіоелектронні і комп'ютерні системи» Х.:ХАІ, – 2008.-Вип. 1(28) – С.162-165

55. Семенов С.Г. Оптимизация трафика на основе сбалансированной загрузки информационно-телекоммуникационной сети // Системи обробки інформації. – Х.: ХВУ, 2004. – № 8(36). – С.206-210

56. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

57. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.:ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування..... 2
2 Підстава для розробки..... 2
3 Мета та призначення розробки..... 2
4 Джерела розробки..... 2
5 Технічні вимоги..... 2
5.1 Вміст проекту..... 2
5.2 Показники призначення..... 3
5.3 Вимоги до функціональних характеристик..... 3
5.4 Вимоги до архітектури..... 3
5.5 Вимоги до надійності..... 3
5.6 Умови експлуатації..... 4
5.7 Вимоги до складу та параметрів технічних засобів..... 4
5.8 Вимоги до інформаційної і програмної сумісності..... 4
5.8.1 Обладнання..... 4
5.8.2 Мова програмування..... 4
5.8.3 Вхідні дані..... 5
5.8.4 Вихідні дані..... 5
6 Вимоги до програмної документації..... 5
7 Економічні вимоги..... 5
8 Вимоги щодо охорони праці..... 5
9 Перелік документів, що розробляються..... 6
10 Етапи розробки..... 6
11 Порядок контролю та приймання..... 6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи Legrand Cabling System 3 для ЦОД.

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. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи Legrand Cabling System 3 для ЦОД.

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 Legrand Cabling System 3 для ЦОД;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.3.2.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 119 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 24.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Доренський О.П.

Дослідження та програмна реалізація

системи Legrand Cabling System 3 для ЦОД

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Control_LegrandCablingSystem .dpr основної програми

```
program Control_LegrandCablingSystem ;

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

{$R *.res}

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

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

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

//опис типів

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

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


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

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

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

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

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

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

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

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

type
  TMibIfTable = packed record
    dwNumEntries     : DWORD;
    Table            : TMibIfArray;
  end;
  PMibIfTable = ^TMibIfTable;
```

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

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

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

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

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

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

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

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

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

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
```

```
                        sReserved: SmallInt):DWORD; stdcall;

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

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

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

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

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

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

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

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

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

```
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів
//

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

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

////////////////////////////
//
//  Закриття загального ресурсу
//
```

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

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

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

//////////////////////////
//
//  Показу діалогу вибору директорії
//

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

```
  end else Result := ' ' ;
  Result := String(TempPath);
end;

////////////////////////////
//
//  Додавання загального ресурсу
//

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

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

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

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

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

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

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

```
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

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

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій
//

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

```
begin
  lvSessions.Items.Clear;

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

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
```

```
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

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

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів
//

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

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

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

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

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

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

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

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

```
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний- вихідний трафік
//

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

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

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

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

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

const
 RESOURCE_BUF_ENTRIES = 2000;

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

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
```

```
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

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

end;

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

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

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

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

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

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

end.
```

Файл IPHLPAPI.pas- обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

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

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

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

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

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

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

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

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , '
tokenring' ,
            ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' ' , ' PPP' ,
            ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

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

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

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

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

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

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

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

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

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

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати данні зразу. Стан >= DISCONNECTED
//
      // може передавати деякі данні. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                       або не з'єднується з картою.        //
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
```

```
      //                      не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                      з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                      в праці.                                    //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                        //
      //
//
      ////////////////////////

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

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

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

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

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          dwLastChange: DWORD;
          dwInOctets: DWORD;
          dwInUcastPkts: DWORD;
          dwInNUCastPkts: DWORD;
          dwInDiscards: DWORD;
          dwInErrors: DWORD;
          dwInUnknownProtos: DWORD;
```

```
          dwOutOctets: DWORD;
          dwOutUCastPkts: DWORD;
          dwOutNUCastPkts: DWORD;
          dwOutDiscards: DWORD;
          dwOutErrors: DWORD;
          dwOutQLen: DWORD;
          dwDescrLen: DWORD;
          bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
        end;

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

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

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

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

        PTMibTCPRow = ^TMibTCPRow;
        TMibTCPRow = packed record
          dwState: DWORD;
          dwLocalAddr: DWORD;
          dwLocalPort: DWORD;
          dwRemoteAddr: DWORD;
          dwRemotePort: DWORD;
        end;
        //
        PTMibTCPTable = ^TMibTCPTable;
        TMibTCPTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..0] of TMibTCPRow;
        end;
        //
        PTMibTCPStats = ^TMibTCPStats;
        TMibTCPStats = packed record
          dwRTOAlgorithm: DWORD;
          dwRTOMin: DWORD;
          dwRTOMax: DWORD;
          dwMaxConn: DWORD;
          dwActiveOpens: DWORD;
```

```
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

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

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

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

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
```

```
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

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

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

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

```
    dwAddrReps: DWORD;
  end;

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

//---------імпорт до IPHLPAPI.DLL------------------

var

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

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

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

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

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

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

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

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

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

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

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

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

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

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

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


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

    function LoadIpHlp: Boolean;

implementation

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

// відкрити DLL
```

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

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

        end.
```

Файл IPHelper.pas - функції роботи з IP-протоколом

```
unit IPHelper;


interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

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

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


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

```
    );


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

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


//-----перетворення різних перерахованих величин у рядки--------

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

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

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

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

type

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

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

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

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі данні---------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

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

implementation

var
  RecentIPs     : TStringList;

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

{ отримання наступного "токена" з рядка }
```

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

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

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

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

end;
```

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

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

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

//-----------------------------------
{ голона частина, фіксація мережний параметрів }

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

//-----------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // данні
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // данні
begin
```

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

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

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


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

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // данні
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // данні
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
```

```
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

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

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

```
end ;

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

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

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

```
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

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

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

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

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

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

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

```
  finally
      FreeMem( pBuf );
  end ;
end ;

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

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

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

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

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


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

```
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // данні
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

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

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

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

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

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

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

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

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

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

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

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

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

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

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації; }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
```

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

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

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

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

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

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

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

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

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

//-------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // данні
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

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

```
end;

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- монітор TCP/IP з'єднань

```
unit TCP_IP;

interface

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

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

var
  Form2: TForm2;

implementation

{$R *.dfm}

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

procedure TForm2.DOIpStuff;
begin

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

end;

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

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

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

end;

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

procedure TForm2.FormCreate(Sender: TObject);
begin

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

end.
```

Файл Stat.pas- статистика мережі

```
unit Stat;

interface

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

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

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

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

end;

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

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

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

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

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

end.
```

Файл About.pas- довідка

```
unit About;

interface

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

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

var
  Form1: TForm1;

implementation

{$R *.dfm}

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

end.
```