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 флеш-масивів Pure Storage для зберігання даних Big Data”

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

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Володимир ПЕТРЕНЮК
«____» _____ 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

Теніченку Єгору Костянтиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(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 флеш-масивів Pure Storage для зберігання даних Big Data*

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

Теніченко Є.К. Дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data. 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 флеш-масивів Pure Storage для зберігання даних Big Data.

Метою розробки є дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

Об'єктом дослідження є процес флеш-масивів Pure Storage для зберігання даних Big Data.

Предметом дослідження є методи флеш-масивів Pure Storage для зберігання даних Big Data.

Методи дослідження базуються на методах теорії Big Data, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

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 RAD Studio Delphi 10.3.

Ключові слова: комп'ютерна інженерія, флеш-масиви, Pure Storage, Big Data

ABSTRACT

Tenichenko Ye.K. Research and software implementation of the Pure Storage flash array system for Big Data storage. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualifying work for the second (master's) level of higher education, software has been developed that is designed for the Pure Storage flash array system for storing Big Data.

The purpose of the development is to research and software implementation of the Pure Storage flash array system for storing Big Data.

The object of the study is the process of Pure Storage flash arrays for storing Big Data.

The subject of research is the methods of Pure Storage flash arrays for storing Big Data.

Research methods are based on the methods of Big Data theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of the Pure Storage flash array system for storing Big Data.

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

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

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

The program is developed in the environment of RAD Studio Delphi 10.3.

Keywords: computer engineering, flash arrays, Pure Storage, Big Data

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
БЗ	–	база знань
БРС	–	блок розрахункових співвідношень
ВКПК	–	вибір кількості пріоритетних користувачів
ІМА	–	інтелектуальний мережний адміністратор
КОМ	–	корпоративні обчислювальні мережі
ЛОМ	–	локальні обчислювальні мережі
МА	–	мережне адміністрування
ПЗ	–	програмне забезпечення
СЗД	–	системи зберігання даних
СУБД	–	система управління базою даних
ЦОД	–	центр обробки даних
CMIP	–	протокол загальної керуючої інформації
DHCP	–	протокол динамічної конфігурації вузла
FMP	–	HPOpenView Fault Management Platform
MIB	–	база даних інформації керування
OEMF	–	HPOpenView Element Management Framework
OER	–	Optimized Edge Routing
OV	–	HPOpenView
Pf	–	Performance Routing
RMON	–	протокол моніторингу комп’ютерних мереж
SNMP	–	простий протокол керування мережею
WINS	–	служба зіставлення netbios-імен комп'ютерів з ip-адресами вузлів

ВСТУП

Актуальність теми. Флеш-масиви стають основним носієм даних: продажі жорстких дисків падають, а флеш-накопичувачів ростуть. В 2020 році за даними IDC поставки гібридних масивів перевершили продажу дискових систем, а цього року, як очікується, їх обійдуть і «чисті» флеш-масиви (All-Flash Array, AFA). Серед законодавців мод на цьому ринку – компанія Pure Storage, що от уже протягом останніх п'яти років Gartner відносить до числа лідерів у цьому ключовому сегменті ринку систем зберігання даних.

Компанія Pure Storage, як заявляється, – самий швидко зростаючий єдиноріг в історії: рубіж в 1 млрд доларів компанія переборола через 8 років після свого створення. Ще більш швидкими темпами росту міг би похвастатися інший стартап в області флеш-масивів, ізраїльська компанія XtremeIO – їй удалося досягти обсягу продажів своєї продукції в 1 млрд доларів усього за 6 років, але на той момент компанія вже три роки як входила до складу EMC.

Pure Storage продовжує нарощувати продажі, причому навіть швидше, ніж раніше. За підсумками II кварталу 2021-го фінансового року її оберт виріс на 37% у порівнянні з аналогічним кварталом попереднього року. Тільки за останній рік число клієнтів збільшилося майже на 40% і перевищило 5000 компаній. У Європі продажу масивів щороку практично подвоюються (середньорічний ріст 98%). Як очікується, за результатами 2021 фінансового року продажу досягнуть 1, 35-1,38 млрд доларів.

Завдяки високому коефіцієнту стиску даних (3, 5-3,8:1) дані обсягом у кілька сотень терабайт удалося розмістити в системі висотою 5U. Це дозволяє відмовитися від необхідності розширення існуючого ЦОД.

Окремо потрібно відмітити те, що в Pure Storage технічна підтримка надається в тому числі й українською мовою, причому в компанії немає підтримки першого рівня – звернення відразу направляється інженерові. Це

виявляється можливо завдяки високій надійності масивів. У середньому по мирі в 98% масивів 100-процентний показник безперервної експлуатації. Це досягається в тому числі за рахунок відновлення не тільки програмного, але й апаратного забезпечення, включаючи контролери, без зупинок. Поки всі поставлені в Україну системи працюють без простоїв (uptime 100%).

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

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

– Огляд існуючих систем флеш-масивів Pure Storage для зберігання даних Big Data.

– Дослідження системи флеш-масивів Pure Storage для зберігання даних Big Data.

– Програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

Об'єктом дослідження є процес флеш-масивів Pure Storage для зберігання даних Big Data.

Предметом дослідження є методи флеш-масивів Pure Storage для зберігання даних Big Data.

Методи дослідження базуються на методах теорії Big Data, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод флеш-масивів Pure Storage для зберігання даних Big Data.

– Розроблено вітчизняний продукт флеш-масивів Pure Storage для зберігання даних Big Data, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі флеш-масивів Pure Storage для зберігання даних Big Data.

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 флеш-масивів Pure Storage для зберігання даних Big Data, 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

Крім технічних інновацій рішення Pure Storage відрізняє революційна модель їхньої підтримки, що одержала назву Evergreen. Як затверджується, придбавши один раз масив, Pure Storage клієнт зможе його використовувати й через 20 років, при цьому він буде адекватну продуктивність і ємність. Замовникам більше немає необхідності регулярно мігрувати дані зі старих систем на нові. Наші технології будуються навколо даних, а не навколо контейнерів, на яких вони зберігаються. Такий підхід дозволяє скоротити вартість володіння на 61%.

Сервіс Evergreen передбачає два види підписки: Evergreen Silver і Evergreen Gold. «Золотий» пакет припускає можливість відновлення не тільки програмного, але й апаратного забезпечення – заміну контролерів з появою рішень наступного покоління (у середньому раз у три роки). Вартість трирічної підтримки срібного рівня становить менш 20% від вартості системи, а золотого не перевищує 40%. В Україні клієнти поки воліють більше дешеві сервісні контракти (на них доводиться більше 80%), оскільки розрахунок вартості володіння обмежується переважно 3-5 років.

Підписка Evergreen Gold виявляється вигідніше при розрахунку ТСО за 10-літній період. У більшості вендорів підтримок обмежується 5 роками, оскільки за цей час масиви застарівають. Відповідно, у розрахунок вартості володіння за 10 років необхідно включати покупку нової СЗД. У випадку масивів Pure Storage їхній інтелект (контролери) буде оновлений три рази, у результаті наприкінці цього строку замовник буде мати набагато більше продуктивний масив, ніж споконвічно, а сумарно витратить менше.

1.2 Область застосування

Областю застосування є системи зберігання даних для Big Data. Великі дані (Big Data) – позначення структурованих і неструктурованих даних величезних обсягів і значного різноманіття, ефективно оброблюваних горизонтально масштабованими програмними інструментами, що з'явилися наприкінці 2000-х років і альтернативних традиційним системам керування базами даних і рішенням класу Business Intelligence.

Як бачимо, у цьому визначенні присутні такі невизначені терміни, як «величезних», «значного», «ефективно» і «альтернативних». Навіть сама назва досить суб'єктивно. Наприклад, 4 Терабайта (ємність сучасного зовнішнього жорсткого диска для ноутбука) – це вже Big Data або ще ні? До цього визначення Вікіпедія додає наступне: «у широкому змісті про "Big Data" говорять як про соціально-економічний феномен, пов'язанийу з появою технологічних можливостей аналізувати величезні масиви даних, у деяких проблемних областях – весь світовий обсяг даних, і які виходять із цього трансформаційних наслідків».

Аналітики компанії IBS «весь світовий обсяг даних» оцінили такими величинами:

- 2003 р. – 5 ексабайтів даних (1 ЕБ = 1 млрд гігабайтів);
- 2008 р. – 0,18 зеттабайта (1 ЗБ = 1024 ексабайта);
- 2015 р. – більше 6,5 зеттабайтів;
- 2021 р. – 40–44 зеттабайта (прогноз);
- 2025 р. – цей обсяг виросте ще в 10 разів.

У доповіді також відзначається, що більшу частину даних генерувати будуть не звичайні споживачі, а підприємства (згадаємо Промисловий інтернет речей).

Можна користуватися й більше простим визначенням, що цілком відповідає устояній думці журналістів і маркетологів.

Big Data – це сукупність технологій, які покликані робити три операції:

1. Обробляти більші в порівнянні з «стандартними» сценаріями обсяги даних.

2. Уміти працювати зі швидко даними, що надходять, у дуже великих обсягах. Тобто даних не просто багато, а їх постійно стає усе більше й більше.

3. Уміти працювати зі структурованими й слабко структурованими даними паралельно й у різних аспектах.

Вважається, що ці «уміння» дозволяють виявити сховані закономірності, що вислизають від обмеженого людського сприйняття. Це дає безпрецедентні можливості оптимізації багатьох сфер нашого життя: державного керування, медицини, телекомунікацій, фінансів, транспорту, виробництва й так далі. Не дивно, що журналісти й маркетологи настільки часто використовували словосполучення Big Data, що багато експертів вважають цей термін дискредитованим і пропонують від нього відмовитися.

Більше того, у жовтні 2020 року компанія Gartner виключила Big Data із числа популярних трендів. Своє рішення аналітики компанії пояснили тим, що до складу поняття «Big Data» входить велика кількість технологій, уже активно застосовуваним на підприємствах, вони частково відносяться до інших популярних сфер і тенденцій і стали повсякденним робочим інструментом.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data, 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

Big Data – тема, що активно обговорюється технологічними компаніями. Деякі з них встигли розчаруватися в Big Data, інші – навпроти, максимально використовують їх для бізнесу.

Ключові характеристики

Big Data, на сьогоднішній момент, є одним із ключових драйверів розвитку інформаційних технологій. Це напрямок, відносно новий для українського бізнесу, одержав широке поширення в західних країнах. Зв'язано це з тим, що в епоху і нформаційних технологій, особливо після буму соціальних мереж, по кожному користувачі інтернету стало накопичуватися значну кількість інформації, що в остаточному підсумку дало розвиток напрямку Big Data.

Термін «Big Data» викликає безліч спорів, багато хто думає, що він означає лише обсяг накопиченої інформації, але не варто забувати й про технічну сторону, дане напрямок містить у собі технології зберігання, обчислення, а також сервісні послуги.

Слід зазначити, що до даної сфери відноситься обробка саме великого обсягу інформації, що важко обробляти традиційними способами.

Сфера Big Data характеризується наступними ознаками:

Volume – обсяг, накопичена база даних являє собою великий обсяг інформації, що трудомістко обробляти й зберігати традиційними способами, для них потрібні новий підхід і вдосконалені інструменти.

Velocity – швидкість, дана ознака вказує як на швидкість, що збільшується, нагромадження даних (90% інформації було зібрано за останні 2

роки), так і на швидкість обробки даних, останнім часом стали більше затребувані технології обробки даних у реальному часі.

Variety – різноманіття, тобто можливість одночасної обробки структурованої й неструктурованої різноформатної інформації. Головна відмінність структурованої інформації – це те, що вона може бути класифікована. Прикладом такої інформації може служити інформація про клієнтські транзакції.

Неструктурована інформація містить у собі відео, аудіо файли, вільний текст, інформацію, що надходить із соціальних мереж. На сьогоднішній день 80% інформації входить у групу неструктурованої. Дана інформація має потребу в комплексному аналізі, щоб зробити її корисної для подальшої обробки.

Veracity – вірогідність даних, все більше значення користувачі стали надавати значимість вірогідності наявних даних. Так, в інтернет-компаній є проблема по поділу дій, проведених роботом і людиною на сайті компанії, що приводить в остаточному підсумку до утруднення аналізу даних.

Value – цінність накопиченої інформації. Big Data повинні бути корисні компанії й приносити певну цінність для неї. Приміром, допомагати в удосконаленні бізнес-процесів, складанні звітності або оптимізації витрат.

При дотриманні зазначених вище 5 умов, накопичені обсяги даних можна відносити до числа великих.

Сфери застосування Big Data

Сфера використання технологій Big Data велика. Так, за допомогою Big Data можна довідатися про переваги клієнтів, про ефективність маркетингових кампаній або провести аналіз ризиків. Нижче представлені результати опитування IBM Institute, про напрямки використання Big Data у компаніях.

Більшість компаній використовують Big Data в сфері клієнтського сервісу, друге по популярності напрямок – операційна ефективність, у сфері керування ризиками Big Data менш поширені на сучасний момент.

Слід також зазначити, що Big Data є однією із самих швидкоростучих сфер інформаційних технологій, відповідно до статистики, загальний обсяг одержуваних і збережених даних подвоюється кожні 1,2 роки.

За період з 2017 по 2020 рік кількість даних, щомісяця переданих мобільними мережами, виросло на 81%. По оцінках Cisco, в 2020 році обсяг мобільного трафіку склав 2,5 ексабайта (одиниця виміру кількості інформації, рівна 10^18 стандартним байтам) на місяць, а вже в 2020 році він буде дорівнює 24,3 ексабайтам.

Таким чином, Big Data – це вже устояна сфера технологій, навіть незважаючи на відносно молодій її вік, що одержав поширення в багатьох сферах бізнесу й граючу немаловажну роль у розвитку компаній.

Технології Big Data

Технології, використовувані для збору й обробки Big Data, можна розділити на 3 групи:

– Програмне забезпечення.

– Устаткування.

– Сервісні послуги.

До найпоширеніших підходів обробки даних (ПЗ) відносяться:

SQL – мова структурованих запитів, що дозволяє працювати з базами даних. За допомогою SQL можна створювати й модифікувати дані, а управлінням масиву даних займається відповідна система керування базами даних.

NoSQL – термін розшифровується як Not Only SQL (не тільки SQL). Містить у собі ряд підходів, спрямованих на реалізацію бази даних, що мають відмінності від моделей, використовуваних у традиційних, реляційних СУБД. Їх зручно використовувати при постійно мінливій структурі даних. Наприклад, для збору й зберігання інформації в соціальних мережах.

MapReduce – модель розподілу обчислень. Використовується для паралельних обчислень над дуже більшими наборами даних (петабайти і більше).

У програмному інтерфейсі не дані передаються на обробку програмі, а програма – даним. У такий спосіб запит являє собою окрему програму. Принцип роботи полягає в послідовній обробці даних двома методами Map і Reduce. Map вибирає попередні дані, Reduce агрегує їх.

Hadoop – використовується для реалізації пошукових і контекстних механізмів високонавантажених сайтів – Facebook, eBay, Amazon і ін. Відмінною рисою є те, що система захищена від виходу з ладу кожн ого з вузлів кластера, так як кожний блок має, як мінімум, одну копію даних на іншому вузлі.

SAP HANA – високопродуктивна NewSQL платформа для зберігання й обробки даних. Забезпечує високу швидкість обробки запитів. Ще однією відміною ознакою є те, що SAP HANA спрощує системний ландшафт, зменшуючи витрати на підтримку аналітичних систем.

До технологічного устаткування відносять:

– сервери;

– інфраструктурне устаткування.

Сервери містять у собі сховища даних

До інфраструктурного устаткування відносять засоби прискорення платформ, джерела безперебійного живлення, комплекти серверних консолей і ін.

Сервісні послуги

Сервісні послуги містять у собі послуги з побудови архітектури системи бази даних, облаштованості й оптимізації інфраструктури й забезпеченню безпеки зберігання даних.

Програмне забезпечення, устаткування, а також сервісні послуги разом утворять комплексні платформи для зберігання й аналізу даних. Такі компанії, як Microsoft, HP, EMC пропонують послуги з розробки, розгортанню рішень Big Data і керування ними.

Застосування в галузях

Big Data одержали широке поширення в багатьох галузях бізнесу. Їх використовують в охороні здоров'я, телекомунікаціях, торгівлі, логістиці, у фінансових компаніях, а також у державному керуванні.

Нижче представлено кілька прикладів застосування Big Data у деяких з галузей.

Роздрібна торгівля

У базах даних роздрібних магазинів може бути накопичена безліч інформації про клієнтів, систему керування запасами, поставками товарної продукції. Дана інформація може бути корисна у всіх сферах діяльності магазинів.

Так, за допомогою накопиченої інформації можна управляти поставками товару, його зберіганням і продажем. На підставі накопиченої інформації можна прогнозувати попит і поставки товару. Також система обробки й аналізу даних може вирішити й інші проблеми ритейлера, наприклад, оптимізувати витрати або підготувати звітність.

Фінансові послуги

Big Data дають можливість проаналізувати кредитоспроможність позичальника, також вони корисні для кредитного скорингу й андерайтингу. Впровадження технологій Big Data дозволить скоротити час розгляду кредитних заявок. За допомогою Big Data можна проаналізувати операції конкретного клієнта й запропонувати підходящі саме йому банківські послуги.

Телеком

У телекомунікаційній галузі широке поширення Big Data одержали в стільникових операторів.

Оператори стільникового зв'язка нарівні з фінансовими організаціями мають одні із самих об'ємних баз даних, що дозволяє їм проводити найбільш глибокий аналіз накопиченої інформації.

Головною метою аналізу даних є втримання існуючих клієнтів і залучення нових. Для цього компанії проводять сегментацію клієнтів, аналізують їх трафіки, визначають соціальну приналежність абонента.

Крім використання Big Data у маркетингових цілях, технології застосовуються для запобігання шахрайських фінансових операцій.

Гірничодобувна й нафтова промисловості

Big Data використовуються як при видобутку корисних копалин, так і при їхній переробці й збуті. Підприємства можуть на підставі інформації, що надійшла, робити виводи про ефективність розробки родовища, відслідковувати графік капітального ремонту й стану устаткування, прогнозувати попит на продукцію й ціни.

За даними опитування Tech Pro Research, найбільше поширення Big Data одержали в телекомунікаційній галузі, а також в інжинірингу, IT, у фінансових і державних підприємствах. За результатами даного опитування, менш популярні Big Data в утворенні й охороні здоров'я.

Приклади використання Big Data у компаніях

На сьогоднішній день Big Data активно впроваджуються в закордонних компаніях. Такі компанії, як Nasdaq, Facebook, Google, IBM, VISA, Master Card, Bank of America, HSBC, AT&T, Coca Cola, Starbucks і Netflix уже використовують ресурси Big Data.

Сфери застосування обробленої інформації різноманітні й варіюються залежно від галузі й завдань, які необхідно виконати.

Далі будуть представлені приклади застосування технологій Big Data на практиці.

HSBC використовує технології Big Data для протидії шахрайських операцій із пластиковими картами. За допомогою Big Data компанія збільшила ефективність служби безпеки в 3 рази, розпізнавання шахрайських інцидентів – в 10 разів. Економічний ефект від впровадження даних технологій перевищив 10 млн дол. США.

Антифрод **VISA** дозволяє в автоматичному режимі обчислити операції шахрайського характеру, система на даний момент допомагає запобігти шахрайським платежам на суму 2 млрд дол. США щорічно.

Суперкомп'ютер Watson компанії **IBM** аналізує в реальному часі потік даних по грошових транзакціях. За даними IBM, Watson на 15% збільшив

кількість виявлених шахрайських операцій, на 50% скоротив помилкові спрацьовування системи й на 60% збільшив суму коштів, захищених від транзакцій такого характеру.

Procter & Gamble за допомогою Big Data проектують нові продукти й роблять глобальні маркетингові кампанії. P&G створив спеціалізовані офіси Business Spheres, де можна переглядати інформацію в реальному часі.

Таким чином, у менеджменту компанії з'явилася можливість миттєво перевіряти гіпотези й проводити експерименти. P&G вважають, що Big Data допомагають у прогнозуванні діяльності компанії.

Рітейлер офісного приладдя **OfficeMax** за допомогою технологій Big Data аналізують поводження клієнтів. Аналіз Big Data дозволив збільшити B2B виторг на 13%, зменшити витрати на 400 000 доларів США в рік.

На думку **Caterpillar**, її дист риб'ютори щорічно упускають від 9 до 18 млрд дол. США прибутку тільки через те, що не впроваджують технології обробки Big Data. Big Data дозволили б клієнтам більш ефективно управляти парком машин, за рахунок аналізу інформації, що надходить із датчиків, установлених на машинах.

На сьогоднішній день уже є можливість аналізувати стан ключових вузлів, їхнього ступеня зношування, управляти витратами на паливо й технічне обслуговування.

Luxottica group є виробником спортивних окулярів, таким марок, як Ray-Ban, Persol і Oakley. Технології Big Data компанія застосовує для аналізу поводження потенційних клієнтів і «розумного» смс-маркетингу. У результаті Big Data Luxottica group виділила більше 100 мільйонів найцінніших клієнтів і підвищила ефективність маркетингової кампанії на 10%.

За допомогою Yandex Data Factory розроблювачі гри **World of Tanks** аналізують поводження гравців. Технології Big Data дозволили проаналізувати поводження 100 тисяч гравців World of Tanks з використанням більше 100 параметрів (інформація про покупки, ігри, досвід і ін.). У результаті аналізу був

отриманий прогноз відтоку користувачів. Дана інформація дозволяє зменшити відхід користувачів і працювати з учасниками гри адресно. Розроблена модель виявилася на 20-30% ефективніше стандартних інструментів аналізу ігрової індустрії.

Міністерство праці Німеччини використовує Big Data в роботі, пов'язаній з аналізом вступників заявок на видачу допомога з безробіття. Так, проаналізувавши інформацію, стало зрозуміло, що 20% допомог виплачувалося незаслужено. За допомогою Big Data міністерство праці скоротило витрати на 10 млрд євро.

Дитяча лікарня Торонто впровадила проект Project Artemis. Це інформаційна система, що збирає й аналізує дані по дітях у реальному часі. Система щомиті відслідковує 1260 показників стану кожної дитини. Project Artemis дозволяє прогнозувати нестабільний стан дитини й почати профілактику захворювань у дітей.

Огляд світового ринку Big Data

Поточний стан світового ринку

В 2020 р. Big Data, на думку Data Collective, стали одними і з пріоритетних напрямків інвестування в сфері венчурної індустрії. Зв'язане це з тим, що розробки з даного напрямку почали приносити значні результати для їхніх користувачів. За минулий рік кількість компаній з реалізованими проектами в сфері керування Big Data збільшилося на 125%, обсяг ринку виріс на 45% у порівнянні з 2018 роком.

Більшу частину виторгу ринку Big Data, на думку Wikibon, в 2020 році склали сервісні послуги, їхня частка була дорівнює 40% у загальному розмірі виручки. Згідно даним Wikibon, застосунки й аналітика становить 36% виторгу Big Data в 2020 році принесли застосунки й аналітика Big Data, 17% – обчислювальне устаткування й 15% – технології зберігання даних. Найменше виторгу було згенеровано NoSQL технологіями, інфраструктурним устаткуванням і забезпеченням мережею компаній (корпоративні мережі).

Найбільшою популярністю користуються такі технології Big Data, як in-memory платформи компаній SAP, HANA, Oracle і ін. Результати опитування T-Systems показали, що їх вибрали 30% опитаних компаній. Другими по популярності стали NoSQL платформи (18% користувачів), також компанії використовували аналітичні платформи компаній Splunk і Dell, їх вибрало 15% компаній. Найменш корисними для рішення проблем Big Data, за результатами опитування виявилися продукти Hadoop/MapReduce.

За даними опитування Accenture, у більш ніж 50% компаніях, що використовує технології Big Data, витрати на Big Data становлять від 21% до 30%.

Згідно з наступним аналізом Accenture, 76% компаній, вважають, що дані витрати збільшаться в 2015 році, а 24% компаній не змінять свого бюджету на технології Big Data. Це говорить про те, що в даних компаніях Big Data стали вже устояним напрямком IT, що став невід'ємною частиною розвитку компанії.

Результати опитування Economist Intelligence Unit survey підтверджують позитивний ефект від впровадження Big Data. 46% компаній заявляють, що за допомогою технологій Big Data вони поліпшили клієнтський сервіс більш, ніж на 10%, 33% компаній оптимізували запаси й поліпшили продуктивність основних активів, 32% компаній поліпшили процеси планування.

Big Data в різних країнах світу

На сьогоднішній день технології Big Data найчастіше впроваджуються в компаніях США, але вже зараз і інші країни світу почали проявляти інтерес. В 2020 році, за даними IDC, на країни Європи, Близького Сходу, Азії (за винятком Японії) і Африки довелося 45% ринку ПЗ, послуг і устаткування в сфері Big Data.

Також, відповідно до опитування CIO, компанії із країн Азіатсько-Тихоокеанського регіону швидкими темпами освоюють нові рішення в області аналізу Big Data, безпечного зберігання й хмарних технологій. Латинська Америка перебуває на другому місці по кількості інвестицій у розвиток технологій Big Data, випереджаючи країни Європи й США.

Далі буде представлений опис і прогнози розвитку ринку Big Data декількох країн.

Китай

Обсяг інформації Китаю становить 909 ексабайт, що дорівнює 10% загального обсягу інформації у світі, до 2021 року обсяг інформації досягне 8060 ексабайт, збільшиться й частка інформації в загальносвітовій статистиці, через 5 років вона буде дорівнює 18%. Потенційний ріст Big Data Китаю має одну із самих швидкоростучих динамік.

Бразилія

Бразилія за підсумками 2020 року нагромадила інформації на 212 ексабайт, що становить 3% від загальносвітового обсягу. До 2021 року обсяг інформації виросте до 1600 ексабайт, що складе 4% інформації всього світу.

Індія

За даними EMC, обсяг накопичених даних Індії за підсумками 2020 року становить 326 ексабайт, що становить 5% від загального обсягу інформації. ДО 2021 року обсяг інформації виросте до 2800 ексабайт, що складе 6% інформації всього світу.

Японія

Обсяг накопичених даних Японії за підсумками 2020 року становить 495 ексабайт, що становить 8% від загального обсягу інформації. ДО 2021 року обсяг інформації виросте до 2200 ексабайт, але зменшиться частка ринку Японії й складе 5% про загальний обсяг інформації всього світу.

Таким чином, обсяг ринку Японії зменшиться на більш, ніж 30%.

Германія

За даними EMC, обсяг накопичених даних у Німеччині за підсумками 2020 року становить 230 ексабайт, що становить 4% від загального обсягу інформації у світі. ДО 2021 року обсяг інформації виросте до 1100 ексабайт і складе 2%.

На ринку Німеччини більшу частку виторгу, за прогнозами Experton Group, буде генерувати сегмент сервісних послуг, частка яких в 2021 році складе 54%, а в 2020 році збільшиться до 59%, частки програмного забезпечення й устаткування, навпаки, зменшаться.

У цілому, обсяг ринку виріс із 1,345 млрд євро в 2015 році до 3,198 млрд євро в 2020 році, середній темп росту склав 24%.

Таким чином, на підставі аналітики CIO і EMC, можна зробити вивід про те, що країни, що розвиваються, світу в найближчі роки стануть ринками активного розвитку технологій Big Data.

Основні тенденції ринку

На думку IDG Enterprise, в 2021 витрати компаній на сферу Big Data складуть у середньому 7,4 млн дол. США на компанію, великі компанії мають намір витратити приблизно 13,8 млн дол. США, малі й середні – 1,6 млн дол. США.

Найбільше буде інвестовано в такі області, як аналіз і візуалізація даних і їхній збір.

Відповідно до поточних тенденцій і попиту на ринку, інвестиції в 2021 році будуть використані на поліпшення якості даних, удосконалювання планування й прогнозування, а також на збільшення швидкості обробки даних.

Компаніями фінансового сектора, за даними Bain Company's Insights Analysis, будуть зроблені значні інвестиції, так в 2021 році планується витратити 6,4 млрд дол. США на технології Big Data, середній темп росту інвестицій складе 22% до 2021 року. Інтернет-компанії планують витратити 2,8 млрд дол. США, середній темп росту збільшення витрат на Big Data складе 26%.

За прогнозами IDC тенденції розвитку ринку виглядають у такий спосіб:

– У наступні 5 років витрати на хмарні рішення в сфері технологій Big Data будуть рости в 3 рази швидше, ніж витрати н а локальні рішення. Стануть затребуваними гібридні платформи для зберігання даних.

– Ріст застосунків з використанням складної прогнозної аналітики, включаючи машинне навчання, прискориться в 2021 році, ринок таких застосунків буде рости на 65% швидше, ніж застосунки, що не використовують прогнозну аналітику.

– Медіа аналітика потроїться в 2021 році й стане ключовим драйвером росту ринку технологій Big Data.

– Прискориться тенденція впровадження рішень для аналізу постійного потоку інформації, що застосовна для інтернету речей.

– ДО 2018 року 50% користувачів будуть взаємодіяти із сервісами, заснованими на когнітивному обчисленні.

Драйвери й обмежники ринку

Експерти IDC, виділили 3 драйвери ринку Big Data 2021 року:

– Масові поглинання клієнтської бази компаній, що пропонують мобільні застосунки й інші дати-платформи.

– Розвиток хмарної інфраструктури.

– Зміни в законах про конфіденційність даних.

– Крім цього також варто виділити:

– Підвищений інтерес на обробку медіа-матеріалів, що відносяться раніше до неструктурованої інформації.

– Ріст популярності навчальних курсів у сфері Big Data.

– Інвестиції у візуалізацію даних і активне storytelling аналітиками даних.

– Постійні інвестиції в Big Data веб-гігантами, такими як Google, Amazon, Facebook і ін.

Серед обмежників ринку Big Data виділяють:

– Усе ще висока вартість впровадження технологій Big Data.

– Необхідність забезпечення захисту даних і їхньої конфіденційності.

– Недостача кваліфікованих кадрів.

– Недовіра компаній до даних технологій.

– Недостатній обсяг накопиченої інформації.

– Підтримка бази даних вимагає постійного фінансування, що створює додатковий бар'єр на впровадження Big Data.

– Складність інтеграції з існуючими системами.

– Обмежене число постачальників даних.

Відповідно до опитування Accenture, питання безпеки даних є зараз головним бар'єром на шляху впровадження технологій Big Data, більше 51% респондентів підтвердили, що турбуються за забезпечення захисту даних і їхньої конфіденційності. 47% компаній повідомили, про неможливість впровадження Big Data у зв'язку з обмеженим бюджето м, 41% компаній як проблема вказали недостачу кваліфікованих кадрів.

Wikibon прогнозує, що обсяг ринку Big Data виросте в 2021 році до 38,4 млрд дол. США й збільшиться в порівнянні з попереднім роком на 36%. У найближчі роки буде спостерігатися спад темпів росту до 10% в 2021 році. З урахуванням даних прогнозів, обсяг ринку в 2021 році буде дорівнює 68,7 млрд дол. США.

Більшу частину ринку буде займати технології зі сфери поліпшення клієнтського сервісу. Крапковий маркетинг буде на другому місці по пріоритетності в компаній аж до 2020 року, в 2021 році, за прогнозом Heavy Reading, він поступиться місцем рішенням по поліпшенню операційної ефективності.

Найвищий темп росту також буде в сегменту «поліпшення клієнтського сервісу», приріст – 49% щорічно.

Переважну частку ринку, як видно з діаграми, займають професійні послуги, найвищий темп ріст буде в застосунків з аналітикою, їхня частка виросте з нинішніх 12% до 18% в 2021 році й обсяг даного сегмента буде дорівнює 12,3 млрд дол. США, частка обчислювального устаткування, навпаки, упаде з 20% до 14% і складе порядку 9,3 млрд дол. США в 2021 році, ринок хмарних технологій буде поступово збільшуватися й в 2021 році досягне 6,3 млрд дол. США, частка ринку рішень для зберігання даних, навпаки, зменшиться з

15% в 2020 році до 13% в 2021 році й у грошовому вираженні буде дорівнює 8,9 млрд дол. США.

Згідно із прогнозом Bain & Company's Insights Analysis, розподіл ринку Big Data по галузях в 2021 році буде виглядати в такий спосіб:

– Фінансова галузь буде здійснювати витрати на Big Data у розмірі 6,4 млрд дол. США із середнім темпом росту 22% у рік.

– Інтернет-компанії витратять 2,8 млрд дол. США й середній темп росту витрат складе 26% за наступні 5 років.

– Витрати держсектору будуть розмірні витратам інтернет-компаній, але темп росту буде нижче – 22%.

– Сектор телекомунікацій буде рости із середнім темпом росту 40% і досягне 1,2 млрд дол. США в 2021 році.

Енергетичні компанії будуть інвестувати в дані технології порівняно невелику суму – 800 млн дол. США, але темп росту буде од ним з найвищих – 54% щорічно.

Таким чином, більшу частку ринку Big Data в 2021 році займуть компанії фінансової галузі, а самим швидкоростучим сектором буде енергетика.

Дотримуючись прогнозів аналітиків, загальний обсяг ринку в найближчі роки буде збільшуватися. Ріст ринку буде забезпечений за рахунок впровадження технологій Big Data у країнах світу, що розвиваються.

Прогнозований обсяг ринку буде залежати від того, що як розвиваються країни сприймуть технології Big Data, чи буде вони також популярні як у розвинених країнах. В 2020 році країни, що розвиваються, світу займали 40% від обсягу накопиченої інформації. За прогнозом EMC, нинішня структура ринку, з перевагою розвинених країн, зміниться вже в 2021 році. Відповідно до аналітики EMC, в 2021 році частка країн, що розвиваються, буде більше 60%.

На думку Cisco і EMC, країни світу, що розвиваються, будуть досить активно працювати з Big Data, багато в чому це буде пов'язане з доступністю технологій і нагромадженням достатнього обсягу інформації до рівня Big Data.

Аналіз українського ринку

Поточний стан українського ринку

Відповідно до результатів дослідження CNews Analytics і Oracle, рівень зрілості українського ринку Big Data за останній рік підвищився. Респонденти, що представляють 108 великих підприємств із різних галузей, продемонстрували більш високий ступінь поінформованості про ці технології, а також сформоване розуміння потенціалу подібних рішень для свого бізнесу.

За станом на 2020 рік, за даними IDC, в Україні накопичено 155 ексабайт інформації, що становить усього лише 1,8% світових даних. Обсяг інформації до 2021 року досягне 980 ексабайт і займе 2,2%. Таким чином, середній темп росту обсягу інформації складе 36% у рік.

Компанія IDC оцінює ринок Україні в 340 млн дол. США, з них 100 млн дол. США – рішення SAP, приблизно 240 млн дол. США – аналогічні рішення Oracle, IBM, SAS, Microsoft і ін.

Темп росту українського ринку Big Data становить не менш, ніж 50% у рік.

Прогнозується збереження позитивної динаміки в цьому секторі українського ринку IT, навіть в умовах загальної стагнації економіки. Це пов'язане з тим, що бізнес як і раніше пред'являє попит на рішення, що дозволяють підвищити ефективність роботи, а також оптимізацію витрат, поліпшення точності прогнозування й мінімізувати можливі ризики компанії.

Основними провайдерами послуг у сфері Big Data на українському ринку є:

- Sap.
- Oracle.
- IBM.
- EMC.
- Microsoft.
- IBS.

– Cloudera.

– Hortonworks.

– Teradata.

Огляд ринку по галузях і досвід застосування Big Data у компаніях

В Україні лише 10% компаній почали використовувати технології Big Data, коли у світі частка таких компаній становить порядку 30%. Готовність до проектів Big Data росте в багатьох галузях економіки Україні – свідчить звіт CNews Analytics і Oracle. Більше третини опитаних компаній (37%) приступили до роботи з технологіями Big Data, серед яких 20% уже використовують такі рішення, а 17% починають експериментувати з ними. Друга третина респондентів у даний момент розглядають таку можливість.

В Україні більшою популярністю технології Big Data користуються в банківській сфері й телекомі, але вони також затребувані в сфері видобувної промисловості, енергетиці, рітейлі, у логістичних компаніях і держсекторі.

Далі будуть розглянуті приклади застосування Big Data в українських реаліях.

Телеком

Телеком-оператори мають одні із самих об'ємних баз даних, що дозволяє їм проводити найбільш глибокий аналіз накопиченої інформації.

Однієї зі сфер застосування технології Big Data є керування лояльністю абонентів.

Головною метою аналізу даних є втримання існуючих клієнтів і залучення нових. Для цього компанії проводять сегментацію клієнтів, аналізують їх трафіки, визначають соціальну приналежність абонента. Крім використання інформації в маркетингових цілях, у телеком-технології застосовуються для запобігання шахрайських фінансових операцій.

Банки

Значну частку користувачів Big Data займають фахівці з фінансової галузі. Одним з успішних досвідів був проведений у Приват-банку, де інформаційну

базу стали використовувати для аналізу клієнтів, банк почав пропонувати спеціалізовані кредитні пропозиції, внески й інші послуги. За рік використання даних технологій роздрібний кредитний портфель компанії виріс на 55%.

В Альфа-банку аналізують інформацію із соціальних мереж, обробляють заявки на одержання кредиту, аналізують поводження користувачів сайту компанії.

Ощадбанк також приступив до обробки масиву даних з метою сегментації клієнтів, запобігання шахрайських дій, перехресних продажів і керування ризиками. Надалі планується вдосконалити сервіс і аналізувати дії клієнтів у режимі реального часу.

Роздрібна торгівля

В Україні технології Big Data були впроваджені компаніями як онлайн, так і офлайн торгівлі. На сьогоднішній день, за даними CNews Analytics, Big Data використовують 20% рітейлерів. 75% фахівців роздрібної торгівлі вважають Big Data необхідними для розвитку конкурентоспроможної стратегії просування компанії. По статистиці Hadoop після впровадження технології Big Data прибуток у торговельних організаціях виростає на 7-10%.

Нафтогазова галузь

У даній галузі сфера застосування Big Data досить широка. Технології Big Data можуть бути застосовані при видобутку корисних копалин з надр. З їхньою допомогою можна аналізувати сам процес видобутку й найбільш ефективні способи його добування, відслідковувати процес буравлення, аналіз якості сировини, а також обробку й збут кінцевої продукції.

Державні органи

У таких країнах, як Німеччина, Австралія, Іспанія, Японія, Бразилія й Пакистан технології Big Data використовуються для рішення питань національного масштабу. Дані технології допомагають органам державної влади більш ефективно надавати послуги населенню, робити адресну соціальну підтримку.

Потенціал реалізації проектів з використанням Big Data великий, даної технології могли б допомогти в поліпшенні якості сервісів, і, як наслідок, рівня життя населення.

Логістика й транспорт

Big Data також можуть бути використані транспортними компаніями. За допомогою технологій Big Data можна відслідковувати парк автомобілів, ураховувати витрати на паливо, проводити моніторинг заявок клієнтів.

Далі буде наведений список технологій використовуваних великими українськими компаніями, із вказівкою функціонала, що використовується на даних підприємствах.

Основні драйвери й обмежники ринку

Драйверами розвитку технологій Big Data на українському ринку є:

– Підвищений інтерес із боку користувачів до можливостей Big Data, як до способу збільшення конкурентоспроможності компанії.

– Розвиток методів обробки медіа-файлів на загальносвітовому рівні.

– Перенос серверів, що обробляють персональну інформацію на територію Україні, відповідно до прийнятого закону про зберігання й обробку персональних даних.

– Здійснення галузевого плану по імпортозаміщенню програмного забезпечення. Даний план містить у собі державну підтримку вітчизняних виробників ПЗ, а також надання преференцій вітчизняної ІТ-продукції при здійсненні закупівель за державний рахунок.

– У новій економічній ситуації, коли курс долара виріс практично в 3 рази, буде спостерігатися тренд по все більшому використанню послуг українських провайдерів хмарних послуг, ніж закордонних.

– Створення технопарків, що сприяють розвитку ринку інформаційних технологій, у тому числі ринку Big Data.

– Державна програма по впровадженню грид-систем, основою яким служать технології Big Data.

Основними бар'єрами для розвитку Big Data на українському ринку є:

– Забезпечення безпеки й конфіденційності даних.

– Недостача кваліфікованих кадрів.

– Недостатність накопичених інформаційних ресурсів до рівня Big Data у більшості українських компаній.

– Складності впровадження нових технологій в устояні інформаційні системи компаній.

– Висока вартість технологій Big Data, що приводить до обмеженого кола підприємств, що мають можливість впровадити дані технології.

– Політична й економічна невизначеність, яка привела до відтоку капіталу й заморозку інвестиційних проектів на території Україні.

– Ріст цін на імпортну продукцію й сплеск інфляції, на думку IDC, гальмують розвиток усього ринку IT.

Прогноз українського ринку

За станом на сьогоднішній день, український ринок Big Data не настільки популярний як у розвинених країнах. Більшість українських компаній виявляють цікавість до нього, але скористатися їхніми можливостями не вирішуються.

Приклади великих компаній, які вже покористувалися від використання технологій Big Data, розширюють усвідомлення можливостей даних технологій.

В аналітиків також досить оптимістичні прогнози щодо українського ринку. IDC уважає, що частка українського ринку за наступні 5 років збільшиться, у відмінності від ринку Німеччини і Японії.

До 2021 року обсяг Big Data Україні виросте з нинішніх 1,8% до 2,2% від загальносвітового обсягу даних. Кількість інформації виросте, за даними EMC, з нинішніх 155 ексабайт до 980 ексабайт в 2021 році.

У даний момент в Україні триває нагромадження обсягу інформації до рівня Big Data.

Відповідно до опитування CNews Analytics, 44% опитаних компаній працюють із даними не більше 100 терабайт* і лише 13% працюють із обсягами вище 500 терабайт.

Проте український ринок, дотримуючись світових тенденцій, буде збільшуватися. За станом на 2020 рік обсяг ринку компанія IDC оцінює в 340 млн дол. США.

Темп росту ринку за попередні роки становив 50% у рік, якщо він залишиться на колишньому рівні, те вже в 2018 році обсяг ринку досягне 1,7 млрд дол. США. Частка українського ринку у світовому складе близько 3%, збільшившись із нинішніх 1,2%.

До найбільш сприйнятливих галузей до використання Big Data в Україні відносяться:

– Рітейл і банки, для них насамперед важливий аналіз клієнтської бази, оцінка ефекту маркетингових кампаній.

– Телеком – сегментація клієнтської бази й монетизація трафіку.

– Держсектор – ведення звітності, аналіз заявок від населення й ін.

– Нафтові компанії – моніторинг робіт і планування видобутку й збуту.

– Енергетичні компанії – створення інтелектуальних електроенергетичних систем, оперативний моніторинг і прогнозування.

У розвинених країнах Big Data одержала широке поширення в сферах охорони здоров'я, страхуванні, металургії, інтернет-компаніях і на виробничих підприємствах, швидше за все в найближчому майбутньому українські компанії з даних сфер також оцінять ефект впровадження Big Data і будуть пристосовувати дані технології у своїх галузях.

В Україні також, як і у світі, у найближчому майбутньому буде спостерігатися тренд на візуалізацію даних, аналіз медіа файлів і розвитку інтернету речей.

Незважаючи на загальну стагнацію економіки, у найближчі роки аналітики прогнозують подальший ріст ринку Big Data, у першу чергу це

пов'язане з тим, що використання технологій Big Data дає конкурентну перевагу її користувачам у частині підвищення операційної ефективності бізнесу, залучення додаткового потоку клієнтів, мінімізації ризиків і впровадження технологій прогнозування даних.

Таким чином, можна укласти, що сегмент Big Data в Україні перебуває в стадії формування, але попит на дані технології з кожним роком збільшується.

Основні результати аналізу ринку

Світовий ринок

За підсумками 2020 року ринок Big Data характеризується наступними параметрами:

– обсяг ринку склав 28,5 млрд дол. США, збільшившись на 45% у порівнянні з попереднім роком;

– більшу частину виторгу ринку Big Data склали сервісні послуги, їхня частка була дорівнює 40% у загальному розмірі виручки;

– 36% виторгу принесли застосунки й аналітика Big Data, 17% – обчислювальне устаткування й 15% – технології зберігання даних;

– найбільшою популярністю для рішення проблем Big Data користуються in-memory платформи таких компаній, як SAP, HANA і Oracle.

– на 125% збільшилася кількість компаній з реалізованими проектами в сфері керування Big Data;

Прогноз ринку на наступні роки виглядає в такий спосіб:

– в 2021 році обсяг ринку досягне 38,4 млрд дол. США, в 2021 році – 68,7 млрд дол. США;

– середній темп росту буде дорівнює 16% щорічно;

– середні витрати компанії на технології Big Data складуть 13,8 млн дол. США для великих компаній і 1,6 млн дол. США для малого й середнього бізнесу;

– технології будуть мати найбільшу поширеність у сферах клієнтського сервісу й крапкового маркетингу;

– в 2021 році зміниться загальносвітова структура ринку убік переваги компаній-користувачів із країн, що розвиваються.

Український ринок

Український ринок Big Data перебуває в стадії формування, результати 2020 року виглядають у такий спосіб:

– обсяг ринку досяг 340 млн дол. США;

– середній темп росту ринку в попередні роки склав 50% щорічно;

– загальний обсяг накопиченої інформації склав 155 ексабайт;

– 10% українських компаній почали використовувати технології Big Data;

– більшою популярністю технології Big Data користувалися в банківській сфері, телекомі, інтернет-компаніях і ритейлі.

Прогноз українського ринку на найближчі роки виглядає в такий спосіб:

– обсяг ринку Україні в 2021 році досягне 500 млн дол. США, а в 2021 році – 1,7 млрд дол. США;

– частка українського ринку у світовому складе близько 3% в 2018 році;

– кількість накопичених даних в 2021 році складе 980 ексабайт;

– обсяг даних виросте до 2,2% від загальносвітового обсягу даних в 2021 році;

– найбільшу популярність придбають технології візуалізації даних, аналізу медіа файлів і інтернету речей.

За результатами аналізу можна зробити вивід про те, що ринок Big Data усе ще перебуває на ранніх стадіях розвитку, і в найближчому будущем ми будемо спостерігати його ріст і розширення можливостей даних технологій.

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 Windo ws 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 застосунок 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++. Уведення 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 флеш-масивів Pure Storage для зберігання даних Big Data.

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

Визначальними характеристиками для Big Data є, крім їхнього фізичного обсягу, і інші, що підкреслюють складність завдання обробки й аналізу цих даних. Набір ознак VVV (volume, velocity, variety – фізичний обсяг, швидкість приросту даних і необхідності їхньої швидкої обробки, можливість одночасно обробляти дані різних типів) був вироблений компанією Meta Group в 2001 році з метою вказати на рівну значимість керування даними по всім трьох аспектах.

Надалі з'явилися інтерпретації із чотирма V (додавалася veracity – вірогідність), п'ятьома V (viability – життєздатність і value – цінність), родина V (variability – мінливість і visualization – візуалізація). Але компанія IDC, наприклад, інтерпретує саме четверте V як value (цінність), підкреслюючи економічну доцільність обробки більших обсягів даних у відповідних умовах.[5]

Виходячи з вищенаведених визначень, основні принципи роботи з Big Data такі:

1. **Горизонтальна масштабованість**. Це – базовий принцип обробки Big Data. Як уже говорилося, Big Data з кожним днем стає усе більше. Відповідно, необхідно збільшувати кількість обчислювальних вузлів, по яких розподіляються ці дані, причому обробка повинна відбуватися без погіршення продуктивності.

2. **Відказостійкість**. Цей принцип випливає з попередні. Оскільки обчислювальних вузлів у кластері може бути багато (іноді десятки тисяч) і їхня кількість, не виключена, буде збільшуватися, зростає й імовірність виходу машин з ладу. Методи роботи з Big Data повинні враховувати можливість таких ситуацій і передбачати превентивні міри.

3. **Локальність даних**. Так як дані розподілені по великій кількості обчислювальних вузлів, то, якщо вони фізично перебувають на одному сервері, а обробляються на іншому, витрати на передачу даних можуть стати невиправдано великими. Тому обробку даних бажано проводити на тій же машині, на якій вони зберігаються.

Ці принципи відрізняються від тих, які характерні для традиційних, централізованих, вертикальних моделей зберігання добре структурованих даних. Відповідно, для роботи з Big Data розробляють нові підходи й технології.

Технології й тенденції роботи з Big Data

Споконвічно в сукупність підходів і технологій включалися засоби масово-паралельної обробки невиразно структурованих даних, такі як СУБД NoSQL, алгоритми MapReduce і засобу проекту Hadoop. Надалі до технологій Big Data стали відносити й інші рішення, що забезпечують подібні по характеристиках можливості по обробці надвеликих масивів даних, а також деякі апаратні засоби.

– **MapReduce** – модель розподілених паралельних обчислень у комп'ютерних кластерах, представлена компанією Google. Відповідно до цієї моделі застосунок розділяється на велику кількість однакових елементарних завдань, виконуваних на вузлах кластера й потім зводяться природно в кінцевий результат.

– **NoSQL** (від англ. Not Only SQL, не тільки SQL) – загальний термін для різних нереляційних баз даних і сховищ, не позначає яку-небудь одну конкретну технологію або продукт. Звичайні реляційні бази даних добре підходять для досить швидких і однотипних запитів, а на складні й гнучко побудованих запитах, характерних для Big Data, навантаження перевищує розумні межі й використання СУБД стають неефективним.

– **Hadoop** – вільно розповсюджуваний набір утиліт, бібліотек і фреймворк для розробки й виконання розподілених програм, що працюють на

кластерах із сотень і тисяч вузлів. Вважається однією з основних технологій Big Data.

– **R** – мова програмування для статистичної обробки даних і роботи із графікою. Широко використовується для аналізу даних і фактично став стандартом для статистичних програм.

– **Апаратні рішення**. Корпорації Teradata, EMC і інші пропонують апаратно-програмні комплекси, призначені для обробки Big Data. Ці комплекси поставляються як готові до установки телекомунікаційні шафи, що містять кластер серверів і керуюче програмне забезпечення для масово-паралельної обробки. Сюди також іноді відносять апаратні рішення для аналітичної обробки в оперативній пам'яті, зокрема, апаратно-програмні комплекси Hana компанії SAP і комплекс Exalytics компанії Oracle, незважаючи на те, що така обробка споконвічно не є масово-паралельної, а обсяги оперативної пам'яті одного вузла обмежуються декількома терабайтами.

Консалтингова компанія McKinsey, крім розглянутих більшістю аналітиків технологій NoSQL, MapReduce, Hadoop, R, включає в контекст застосовності для обробки Big Data також технології Business Intelligence і реляційні системи керування базами даних з підтримкою мови SQL.

Методи й техніки аналізу Big Data

Міжнародна консалтингова компанія McKinsey, що спеціалізується на рішенні завдань, зв'язаних зі стратегічним керуванням, виділяє 11 методів і технік аналізу, застосовних до Big Data.

– **Методи класу Data Mining** (видобуток даних, інтелектуальний аналіз даних, глибинний аналіз даних) – сукупність методів виявлення в даних раніше невідомих, нетривіальних, практично корисних знань, необхідних для прийняття рішень. До таких методів, зокрема, відносяться навчання асоціативним правилам (association rule learning), класифікація (розбивка на категорії), кластерний аналіз, регресійний аналіз, виявлення й аналіз відхилень і ін.

– **Краудсорсинг** – класифікація й збагачення даних силами широкого, невизначеного кола осіб, що виконують цю роботу без вступу в трудові відносини

– **Змішання й інтеграція даних** (data fusion and integration) – набір технік, що дозволяють інтегрувати різнорідні дані з різноманітних джерел з метою проведення глибинного аналізу (наприклад, цифрова обробка сигналів, обробка природної мови, включаючи тональний аналіз, і ін.)

– **Машинне навчання**, включаючи навчання із учителем і без учителя – використання моделей, побудованих на базі статистичного аналізу або машинного навчання для одержання комплексних прогнозів на основі базових моделей

– **Штучні нейронні мережі**, мережний аналіз, оптимізація, у тому числі генетичні алгоритми (genetic algorithm – евристичні алгоритми пошуку, використовувані для рішення завдань оптимізації й моделювання шляхом випадкового підбора, комбінування й варіації шуканих параметрів з використанням механізмів, аналогічних природному добору в природі)

– **Розпізнавання образів**.

– Прогнозна аналітика.

– Імітаційне моделювання (simulation) – метод, що дозволяє будувати моделі, що описують процеси так, як вони проходили б у дійсності. Імітаційне моделювання можна розглядати як різновид експериментальних випробувань

– **Просторовий аналіз** (spatial analysis) – клас методів, що використовують топологічну, геометричну й географічну інформацію, що витягається з даних

– **Статистичний аналіз** – аналіз тимчасових рядів, A/ B -Тестування (A/B testing, split testing – метод маркетингового дослідження; при його використанні контрольна група елементів рівняється з набором тестових груп, у яких один або кілька показників були змінені, для того щоб з'ясувати, які зі змін поліпшують цільовий показник)

– **Візуалізація аналітичних даних** – подання інформації у вигляді рисунків, діаграм, з використанням інтерактивних можливостей і анімації як для одержання результатів, так і для використання в якості вихідних даних для подальшого аналізу. Дуже важливий етап аналізу Big Data, що дозволяє представити найважливіші результати аналізу в найбільш зручному для сприйняття виді.

Big Data в промисловості

Відповідно до звіту компанії McKinsey «Global Institute, Big data: The next frontier for innovation, competition, and productivity», дані стали таким же важливим фактором виробництва, як трудові ресурси й виробничі активи. За рахунок використання більших дані компанії можуть одержувати відчутні конкурентні переваги. Технології Big Data можуть бути корисними при рішенні наступних завдань:

- прогнозування ринкової ситуації;
- маркетинг і оптимізація продажів;
- удосконалювання продукції;
- прийняття управлінських рішень;
- підвищення продуктивності праці;
- ефективна логістика;
- моніторинг стану основних фондів.

На виробничих підприємствах Big Data генеруються також внаслідок впровадження технологій Промислового інтернету речей. У ході цього процесу основні вузли й деталі верстатів і машин забезпечуються датчиками, виконавчими пристроями, контролерами й, іноді, недорогими процесорами, здатними робити граничні (мрячні) обчислення. У ході виробничого процесу здійснюється постійний збір даних і, можливо, їхня попередня обробка (наприклад, фільтрація). Аналітичні платформи обробляють ці масиви інформації в режимі реального часу, представляють результати в найбільш зручному для сприйняття виді й зберігають для подальшого використання. На

основі аналізу отриманих даних робляться виводи про стан устаткування, ефективності його роботи, якості продукції, що випускається, необхідності внесення змін у технологічні процеси й т.д.

Завдяки моніторингу інформації в режимі реального часу персонал підприємства може:

– скорочувати кількість простоїв;

– підвищувати продуктивність устаткування;

– зменшувати витрати на експлуатацію устаткування;

– запобігати нещасні випадки.

Останній пункт особливо важливий. Наприклад, оператори, що працюють на підприємствах нафтохімічної промисловості, одержують у середньому близько 1500 аварійних повідомлень у день, тобто більше одного повідомлення в мінуту. Це приводить до підвищеної утоми операторів, яким доводиться постійно приймати миттєві рішення про те, як реагувати на той або інший сигнал. Але аналітична платформа може відфільтрувати другорядну інформацію, і тоді оператори одержують можливість зосередитися в першу чергу на критичних ситуаціях. Це дозволяє їм більш ефективно виявляти й запобігати аварії й, можливо, нещасні випадки. У результаті підвищуються рівні надійності виробництва, промислової безпеки, готовності технологічного устаткування, відповідності нормативним вимогам.

Крім того, за результатами аналізу Big Data можна розраховувати строки окупності устаткування, перспективи зміни технологічних режимів, скорочення або перерозподіли обслуговуючого персоналу – тобто приймати стратегічні рішення щодо подальшого розвитку підприємства.

3.2 Розробка структурної схеми

Сучасне Big Data рішення складається з декількох блоків, що вимагають спільної роботи команд із різними компетенціями й інтеграції набору Open-source і пропрієтарних програмних компонентів:

1. Технічне рішення по збору, зберіганню й обробці більших обсягів даних, позначене на схемі як Big Data Tools. Таке рішення, як правило, будується на основі стека Hadoop, так як він представляє гарний баланс між вартістю, надійністю й функціональністю.

2. Просунутий аналіз даних з використанням методів науки про даний (Data Science) і алгоритмів машинного навчання

3. Візуалізація Big Data, а також створення інтерактивних звітів для керівництва компанії, співробітників і клієнтів (Business Intelligence). При цьому використовувана аналітична платформа повинна бути сумісна зі стеком Hadoop

Інтеграція платформи системи флеш-масивів Pure Storage для зберігання даних зі стеком Big Data

Big Data рішення будуються на базі open-source технологічного стека Hadoop. Для рішення завдань візуалізації, моделювання й інтерактивного аналізу даних у такі рішення інтегрується аналітична платформа системи флеш-масивів Pure Storage для зберігання даних. Також можлива інтеграція із уже існуючими системами Big Data.

Складання рішення Hadoop вимагають глибокої експертизи. Фахівці мають необхідні знання для побудови оптимальної архітектури аналітичного Big Data рішення й використовують наступні технології, що забезпечують роботу в реальному масштабі часу в умовах підприємства:

1. Багаторівнева сегментація даних. Наприклад, самий часто затребуваний, але щодо невеликий обсяг даних зберігається в базі даних у пам’яті, платформи системи флеш-масивів Pure Storage для зберігання даних, а повний обсяг даних – в розподіленому файловому сховищі.

2. Кешовані на різних рівнях системи.

3. λ-архітектура для забезпечення відновлення всіх рівнів даних у реальному часі.

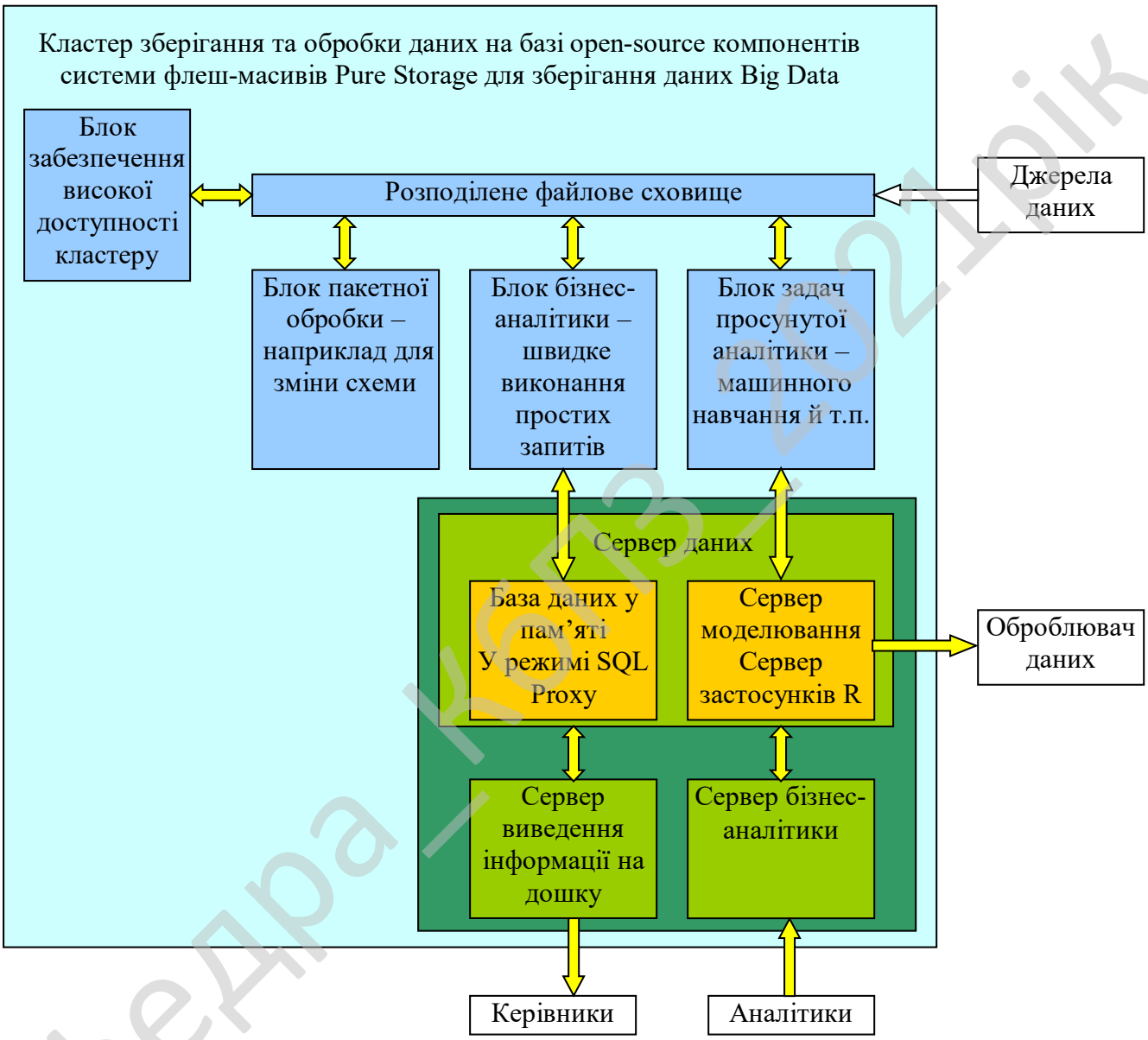

Рисунок 3.1 – Структурна схема системи

Просунутий аналіз даних і моделювання з використанням методів Data Science

Інструменти Data Science – це комп'ютерні методи й алгоритми, що дозволяють застосувати розділи математичної статистики, теорії ймовірностей, чисельних методів оптимізації дискретного аналізу для виділення знань із даних.

– Метричні методи класифікації й регресії.

– Логічні методи класифікації.

– Критерії вибору моделей і методи відбору ознак.

– Градієнтні методи навчання.

– Метод опорних векторів.

– Багатомірна лінійна регресія.

– Нелінійна регресія.

– Прогнозування тимчасових рядів.

– Байєсовська теорія класифікації.

– Логістична регресія. Поділ суміші розподілів.

– Кластеризація.

– Нейронні мережі.

– Лінійні композиції, бустинг.

– Евристичні, стохастичні, нелінійні композиції.

– Ранжирування.

– Пошук асоціативних правил.

– Завдання із частковим навчанням.

– Колаборативна фільтрація.

– Тематичне моделювання.

– Навчання з підкріпленням.

За допомогою методів Data Science можна оптимізувати виробничі процеси без значних капітальних витрат. Важливою особливістю проектів Data Science є дослідницький характер, до проведення серйозного аудита даних неможливо дати точний висновок про досяжність тих або інших бізнес-цілей. Для

рішення цієї проблеми системи флеш-масивів Pure Storage для зберігання даних пропонує підхід, що дозволяє максимально знизити ризики клієнта, що сформований у відповідності з наступними принципами.

3.3 Розробка функціональної схеми

Скористайтеся міццю й швидкістю системи флеш-масивів Pure Storage для зберігання даних Big Data для рішення самих складних бізнес-завдань. Системи зберігання забезпечують продуктивність сучасних твердотільних накопичувачів, а також простоту, універсальність і економічність для будь-яких цілей, будь те консолідація навантажень загального призначення, розгортання нової інфраструктури віртуальних робочих столів, великомасштабні OLTP-системи або приватна хмара на базі флеш-технологій. Навіщо іти на компроміс, здобуваючи менш комплексні рішення? Ці масиви мають всі необхідні функції, щоб ви змогли швидше приступитися до роботи із флеш-технологіями й раніше одержати результати.

Комплексна система на базі флеш-технологій допомагає вам досягти неможливого й вивести свій бізнес на новий рівень ефективності й процвітання. Орієнтована на продуктивність конфігурація "активний/активний" підтримує високе число операцій вводу-виводу в секунду в процесі горизонтального й вертикального масштабування й гарантує, що всі інвестиції в систему зберігання даних прямо підуть на прискорення роботи застосунків.

Динамічні інтелектуальні функції

Системи зберігання даних на базі флеш-технологій спеціально створені для адаптації до постійних змін і росту. Вони стабільно підтримують навантаження в умовах непередбаченого розвитку середовища. Більше не потрібно відмовлятися від бізнес-можливостей через неадаптуєму інфраструктуру зберігання даних. Тепер, як ніколи раніше, ви зможете експериментувати з новими ідеями й міняти стратегії по ходу справи.

Ефективність і вигода

Інноваційні функції автоматичної оптимізації, включаючи інтелектуальні способи дедуплікації й стиску, допомагають скоротити витрати. Повсюдно застосовувані "тонкі методи" і інтуїтивні засоби керування дозволяють використовувати ресурси по максимуму, а сама головне – вся розширена функції ПЗ включені в системах флеш-масивів Pure Storage для зберігання даних Big Data за замовчуванням. Не потрібно купувати й підтримувати ніякі додаткові ліцензії!

Об'єднання продуктивності і ємності до 10 масивів

Убудований "гіпервізор сховища" дозволяє плавно переносити навантаження між масивами, вносячи зміни в систему зберігання без порушення роботи серверів. Перевірена на підприємствах операційна система флеш-масивів Pure Storage для зберігання даних Big Data автоматизує завдання керування, що займають багато часу й піддані помилкам, включаючи зіставлення LUN із серверами й виділення ресурсів RAID. Підвищте швидкість розгортання й адаптуємость конфігурацій, щоб успішно конкурувати на сучасних динамічних ринках.

Просте перенесення даних

Перенесення даних усередині федерації виконується швидко й прозоро для керуючих систем. Потрібно перенести тім в інший масив? Функція динамічного переносу допоможе легко це зробити, не перериваючи виконання навантажень і зберігаючи зв'язок між моментальними копіями й даними реплікації протягом усього переносу.

Мастери балансування навантаження

Вхідна в продукт утиліта дає поточні рекомендації з балансування навантаження, пропонуючи найкращі варіанти первісного розміщення даних і потім згодом рекомендуючи нові можливості оптимізації.

Постійно доступна система зберігання даних

Live Volume також підтримує безперервне виконання навантажень у часонепередбувальних простоїв і катастрофічних збоїв. Автоматичне

перемикання при відмові між синхронізованими томами в локальних і віддалених масивах без переривання роботи допомагає захистити найважливіші бізнес-процеси.

Цінність масивів на флеш-дисках – усе включено!

Тільки в системах флеш-масивів Pure Storage для зберігання даних Big Data всі розширені функції включені за замовчуванням, що забезпечує максимальну цінність і перспективність серед всіх продуктів лінійки системи флеш-масивів Pure Storage для зберігання даних Big Data. Всі функції ліцензовані для масиву повної ємності. Ви одержуєте наступне:

– Інтелектуальні методи дедуплікації й стиску.

– Динамічний перенос усередині федерації.

– Live Volume з автоматичним перемиканням при відмові.

– Технологія Data Progression.

– Розподіл даних по рівнях RAID.

– Динамічні контролери.

– Тонкі моментальні копії.

– Thin Clones ("Тонке" клонування).

– Реплікація (одновузлова, багатовузлова, "один до багатьох").

– Remote Instant Replay (синхронна/асинхронна реплікація в IP-мережах).

– Підтримка мереж (SAN) і мережних систем (NAS) зберігання даних (блокове й файлове зберігання з використанням одного пула через опціональний NAS-пристрій FS8600 з горизонтальним масштабуванням).

– Мережі з підтримкою декількох протоколів.

– Власні засоби відновлення застосунків.

– Веб-інтерфейси на основі HTML – Unisphere for системи флеш-масивів Pure Storage для зберігання даних Big Data, Unisphere Central.

– Хмарна аналітика й моніторинг – CloudIQ.

– Інтерфейс клієнтського застосунку.

– Підтримка дисків із самошифруванням.

– Розрахунки між підрозділами.

– Маркування множини мереж VLAN.

– Якість обслуговування.

– Підтримка віртуальних томів VVOL.

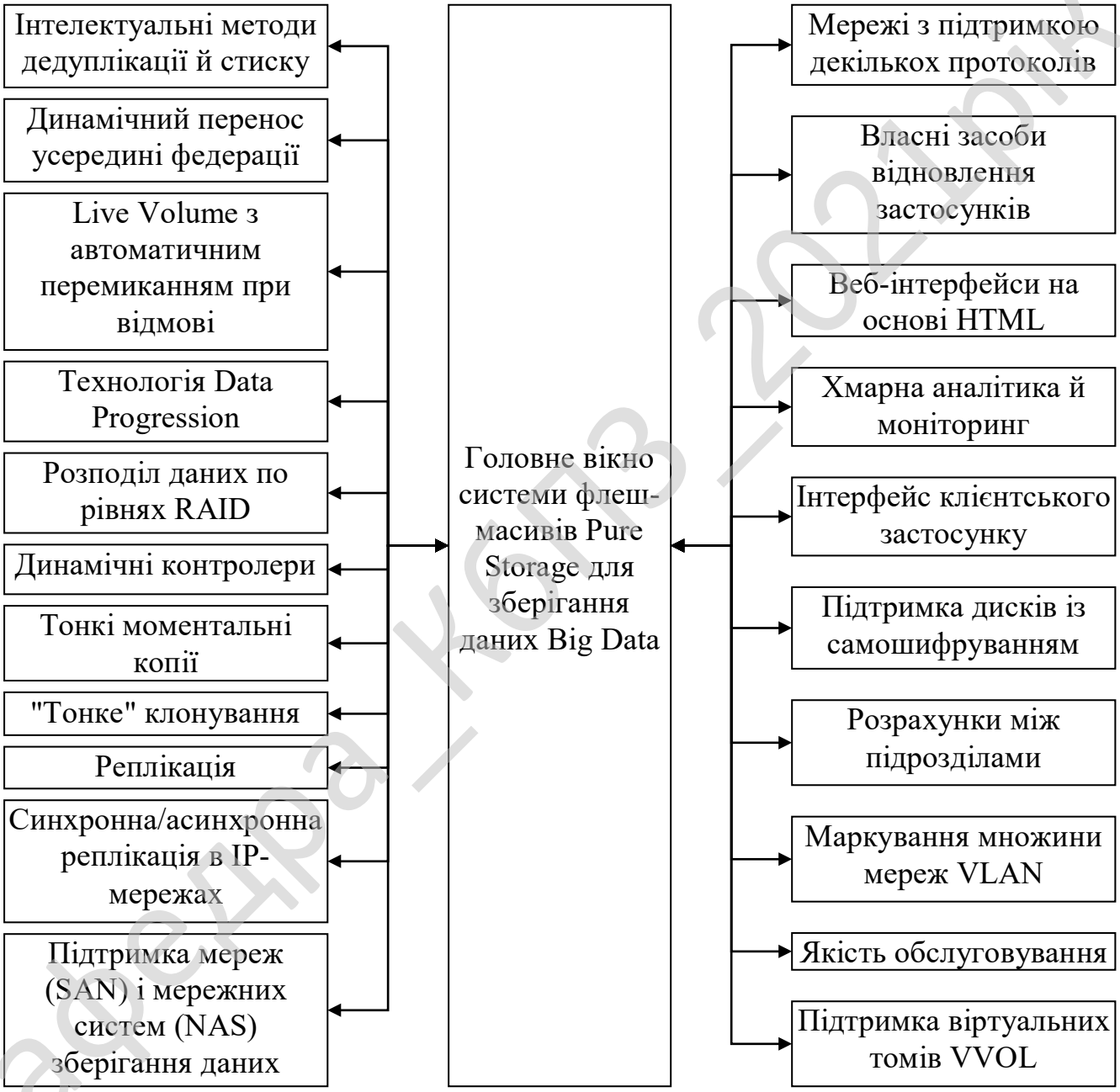

Рисунок 3.2 – Функціональна схема системи

Надійне рішення корпоративного класу

І звичайно, якість системи флеш-масивів Pure Storage для зберігання даних Big Data підкріплено репутацією лідера ринку масивів на флеш-дисках і лідера в секторі систем зберігання в цілому. Продукти системи флеш-масивів Pure Storage для зберігання даних Big Data повсюдно використовуються в центрах обробки даних по усьому світі. Ви можете скористатися перевагами настільки великого досвіду в технологіях і бізнесі!

Можливість збільшення ємності

Підтримувані шасі розширення:

– системи флеш-масивів Pure Storage для зберігання даних Big Data SC420F: 24 відсіку для накопичувачів 2,5" (SAS 12 Гбіт/с).

Максимальна кількість накопичувачів**:**

– SC7020F: 606 накопичувачів на кожний масив; більше у федеративних системах.

– SC5020F: 222 на кожний масив (з опціональними полками розширення); більше у федеративних системах.

Максимальна неформатована ємність:

– SC7020F: 4 Пбайт на кожний масив; більше у федеративних системах.

– SC5020F: 2,16 Пбайт на кожний масив; більше у федеративних системах.

Максимальна ємність без стиску даних (мережна система зберігання даних):

– 4 Пбайт у єдиному просторі імен (потрібно NAS-пристрій FS8600).

– До 20 Пбайт у глобальному просторі імен (потрібно FS8600 з декількома масивами).

Носії для зберігання даних:

– Твердотільні накопичувачі: Write Intensive і Read Intensive (можна використовувати спільно в одній системі), твердотільні накопичувачі із самошифруванням.

Оптимізація даних

– **Метод автоматичного багаторівневого розміщення даних** Заснований на політиках перенесення даних залежно від їхнього використання в реальному часі; набудовується размер, що, сторінки від 512 Кбайт до 4 Мбайт.

– **Структура автоматичного багаторівневого розміщення.** До 2 рівнів твердотільних накопичувачів (Write Intensive і Read Intensive)

– **Настроювання багаторівневого розміщення.** Користувальницькі профілі, можливість "закріплення" томів на будь-якому рівні

– **Підтримка RAID.** RAID 0, 1, 5, 6, RAID 1 0 і RAID 10 DM (подвійне дзеркалювання); можливі будь-які комбінації рівнів RAID в одному масиві

– **Розподіл даних по рівнях RAID.** Автоматичне надання ресурсів і динамічний перерозподіл даних по декількох рівнях RAID на одному рівні; не потрібне попереднє виділення груп RAID

– **Динамічне виділення ресурсів.** За замовчуванням активно на всіх томах, працює з повною продуктивністю для всіх функцій

– **Тонкі моментальні копії.** Зберігаються тільки зміни записів, моментальні копії автоматично переносяться в більше економічну підсистему зберігання даних

– **Інтелектуальні методи дедуплікації й стиску**. Можливість вибору для кожного тому. Також доступний варіант тільки зі стиском

Загальні відомості про корпус

Якщо не зазначене інше, технічні характеристики відносяться до обох моделей масивів SC7020F і SC5020F.

Формат корпуса

"Усе в одному" (дубльовані контролери, внутрішні відсіки дисководів, мережні порти й порти розширення).

Розмір стійки

3U.

Контролери

Два на корпус із можливістю гарячого перемикання (конфігурація " активний-активний").

Число процесорів на контролер

– SC7020F: два процесори Intel® Xeon® E 5-2628 v3, 2,5 Ггц, 8 ядер.

– SC5020F: один процесор Intel® Xeon® E 5-2630 v3, 2,4 Ггц, 8 ядер.

Ємність внутрішнього сховища

30 відсіків для накопичувачів 2,5".

Системна пам'ять

– SC7020F: 256 Гбайт на кожний масив (128 Гбайт на кожний контролер).

– SC5020F: 128 Гбайт на кожний масив (64 Гбайт на кожний контролер).

Операційна система

Dell Storage Center OS (SCOS) 7.2 або більше пізньої версії.

Мережа й розширення підсистеми вводу-виводу

Мережні протоколи зовнішнього інтерфейсу

FC, iSCSI (підтримка одночасно декількох протоколів).

Максимальне число портів FC 32 Гбіт/с

– SC7020F: 24 на кожний масив (SFP+).

– SC5020F: 8 на кожний масив (SFP+).

Максимальне число портів FC 8/16 Гбіт/с

– SC7020F: 24 на кожний масив (SFP+).

– SC5020F: 8 на кожний масив (SFP+).

Максимальне число портів iSCSI 100 Гбіт/с

– SC7020F: 16 QSFP28 на кожний масив.

– SC5020F: 8 QSFP28 на кожний масив.

Максимальне число портів iSCSI 25 Гбіт/с

– SC7020F: 16 SFP28 на кожний масив.

– SC5020F: 8 SFP28 на кожний масив.

Максимальне число портів iSCSI 10 Гбіт/с

– SC7020F: 32 SFP+ (10 Гбіт/с) або Base-T на кожний масив.

– SC5020F: 16 SFP+ (10 Гбіт/с) або Base-T на кожний масив.

Порти керування

2 на кожний масив (BASE-T 1 Гбіт/с).

Внутрішні протоколи розширення

SAS 12 Гбіт/с.

Максимальне число внутрішніх портів розширення

– SC7020: 24 на кожний масив.

– SC5020: 8 на кожний масив.

– До 16 корпусів розширення на кожний масив.

Функціональність

Конфігурації масиву

Тільки повні флеш-масиви.

Формат зберігання даних

Блоковий (SAN) і/або файловий (NAS) із загального пула з опціональной системою FS8600.

Максимальна кількість хостів SAN: 500.

Максимальна кількість портів ініціатора: 1 000.

Максимальна розмір LUN: 500 Тбайт.

Максимальна кількість LUN: 2 000.

Максимальне число моментальних копій:

– SC7020F: 16 384.

– SC5020F: 8 192.

Максимально операцій вводу-виводу в секунду:

– SC7020F: 1 200 000.

– SC5020F: 1 025 000.

Максимально операцій вводу-виводу в секунду (із затримкою менш 1 мс):

– SC7020F: 1 050 000.

– SC5020F: 818 000.

Максимально операцій вводу-виводу в секунду (80% читання, 20% запис):

– SC7020F: 346 000**.**

– SC5020F: 330 000.

Максимальна пропускна здатність (читання):

– SC7020F: 29 000 Мбайт/с.

– SC5020F: 19 000 Мбайт/с.

Максимальна пропускна здатність (запис):

– SC7020F: 14 000 Мбайт/с.

– SC5020F: 9 500 Мбайт/с.

Мобільність і перенесення даних

Реплікація

– Реплікація з іншими масивами серії SC.

– Синхронна або асинхронна реплікація по протоколах FC або iSCSI.

– Взаємозв'язок мети/джерела може мати тип "один до багатьох" або "багато до одного".

– Підтримує всі служби даних системи флеш-масивів Pure Storage для зберігання даних Big Data на томах джерела й мети.

– Зміна типу реплікації й топологій на вимогу.

– Підтримка крос-платформної реплікації з масивами серії PS/EqualLogic (у будь-якому напрямку).

Мобільність томів

Вхідна до складу базового продукту функція Live Migrate забезпечує прозорий для вузла перенесення даних між масивами.

Федеративні системи з декількома масивами

– Вхідна до складу базового продукту функція Live Migrate забезпечує прозорий для вузла перенесення даних між масивами.

– Моментальні копії зберігаються під час переносу.

Thin Import ("Тонкий" імпорт)

Ефективно використовує простір перенесення даних з масивів серій PS (EqualLogic) і MD3 без порушення роботи.

Thin Clones ("Тонке" клонування)

– Клонування окремих томів з нульовою дедуплікацією даних.

– Клони підтримують незалежні моментальні копії й реплікацію.

– Оптимально підходять для інфраструктури віртуальних робочих столів, тестування й розробки, а також інших застосунків, яким потрібні дискретні екземпляри загальних даних.

– Більше ефективний спосіб, чим дедуплікація для копій баз даних.

Захист даних, аварійне відновлення, безпека

Безперервність бізнесу

– Компонент Live Volume забезпечує двостороннє автоматичне перемикання й відновлення при відмові.

– Безперервна робота, відновлення після збоїв, запобігання збоїв.

– Компонент Live Volume Managed Replication підтримує варіанти реплікації з додатковим третім вузлом.

– Нульові значення цільового часу відновлення/цільової крапки відновлення з набудовується соглашением, що, про рівень обслуговування при перемиканні вузла при відмові для кожного тому.

– Підтримка технології кластера VMware Metro Stretch.

– VMware Site Recovery Manager.

Розподілене резервування

Не потрібно виділення дисків гарячого резерву, всі диски доступні для читання й запису.

Тонкі моментальні копії

Зберігаються тільки зміни записів, моментальні знімки автоматично переносяться на більше економічну підсистему зберігання даних.

Погодженість із додатками

– Replay Manager: погоджені з додатками моментальні копії в середовищах Microsoft або VMware.

– AppSync: погоджені з додатками моментальні копії в середовищах Microsoft, VMware і Oracle**.**

Шифрування неактивних даних

– Підтримка дисків із самошифруванням.

– Повне шифрування дисків (FDE) за стандартом AES-256.

– Накопичувачі сертифіковані у відповідності зі стандартом FIPS 140-2 рівні 2.

– Для FIPS 140-2 рівнів 1, 2 і 3 доступні варіанти із сервером керування ключами (KMS).

Підтримка зовнішнього диспетчера ключів

– SafeNet KeySecure k460, SafeNet KeySecure k250, SafeNet KeySecure k150v компанії Gemalto.

– Thales EMS 200.

Умови експлуатації

Температура при експлуатації: від 10 до 35 °C (від 50 до 95 °F).

Температура при зберіганні: від –40 до 65 °C (від –40 до 149 °F).

Вологість при експлуатації (без конденсації): від 10 до 80% з максимальною крапкою роси 29 °C (84,2 °F).

Вологість при зберіганні (без конденсації): від 5 до 95% з максимальною крапкою роси 33 °C (91 °F).

Тип рознімання живлення: NEMA 5-15/CS22.2, n°42.

Живлення

Живлення/потужність:

– SC7020F: два джерела живлення потужністю 1 485 Вт із можливістю гарячого перемикання.

– SC5020F: два джерела живлення з можливістю гарячого перемикання (варіанти потужністю 1 378 або 1 485 Вт).

Максимальна вихідна потужність:

– SC7020F: 1 485 Вт.

– SC5020F: 1 378 Вт (варіант БП 1 378 Вт) або 1 485 Вт (варіант БП 1 485 Вт).

Максимальна вхідна потужність:

– SC7020F: 1 688 Вт.

– SC5020F: 1 584 Вт (варіант БП 1 378 Вт) або 1 688 Вт (варіант БП 1 485 Вт).

Максимальна вхідний струм:

– SC7020F: 8,8 А.

– SC5020F: 16 А (варіант БП 1 378 Вт) або 8,8 А (варіант БП 1 485 Вт).

Максимальна пусковий струм:

55 А в плину 10 мс або менш.

Номінальний робочий діапазон вхідної напруги:

– SC7020F: 200-240 У змінного струму.

– SC5020F: 100-240 У змінного струму (варіант БП 1 378 Вт) або 200-240В змінного струму (варіант БП 1 485 Вт).

Номінальна вхідна частота: 50/60 Гц

Максимальна тепловиділення/тепловіддача:

– SC7020F: 5 760 БТЕ/год.

– SC5020F: 5 770 БТЕ/год (варіант БП 1 378 Вт) або 5 760 БТЕ/год (варіант БП 1 485 Вт).

Керування

Інтерфейси керування

– Інтерфейс на основі браузера (HTML 5).

– Unisphere for системи флеш-масивів Pure Storage для зберігання даних Big Data (менеджер компонентів).

– Unisphere Central (керування декількома масивами).

– CloudIQ (хмарна аналітика й моніторинг).

– Клієнтський застосунок.

– Міжплатформене керування декількома масивами, вузлами й об'єктами.

Об'єднання у федерацію

Створення великих систем з декількома масивами за допомогою уніфікованого керування із плавним переносом навантаження між масивами за допомогою функції Live Migrate. Додавання масивів будь-якого типу без переривання роботи, ефективне використання їхньої сукупної ємності й продуктивності. Утиліта Volume Advisor контролює кілька масивів, пропонуючи варіанти оптимального розміщення даних і балансування навантаження. Переміщення томів не впливає на моментальні копії або захист даних реплікації.

Підтримка сценаріїв

– Microsoft PowerShell API.

– RESTful API.

Підтримка серверних ОС

– Microsoft® Windows Server®, Oracle® Solaris, HP®-UX, Oracle Linux, IBM® AIX®, Novell® NetWare, SLES, Apple, HPTru64, VMware®, Citrix® XenServer®, RedHat®.

Інтеграція сторонніх застосунків

VMware, Microsoft, IBM, OpenStack, Symantec, Commvault, Foglight, Docker

Сумісність із масивами серії PS

– Двостороння реплікація.

– Керування щоденними завданнями через єдиний інтерфейс.

– Thin Import ("Тонкий" імпорт): ефективно використовує простір перенесення даних без порушення роботи з масивів серії PS.

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.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз'яснення який саме тип блок-схем використовується. Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок - схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об'єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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.

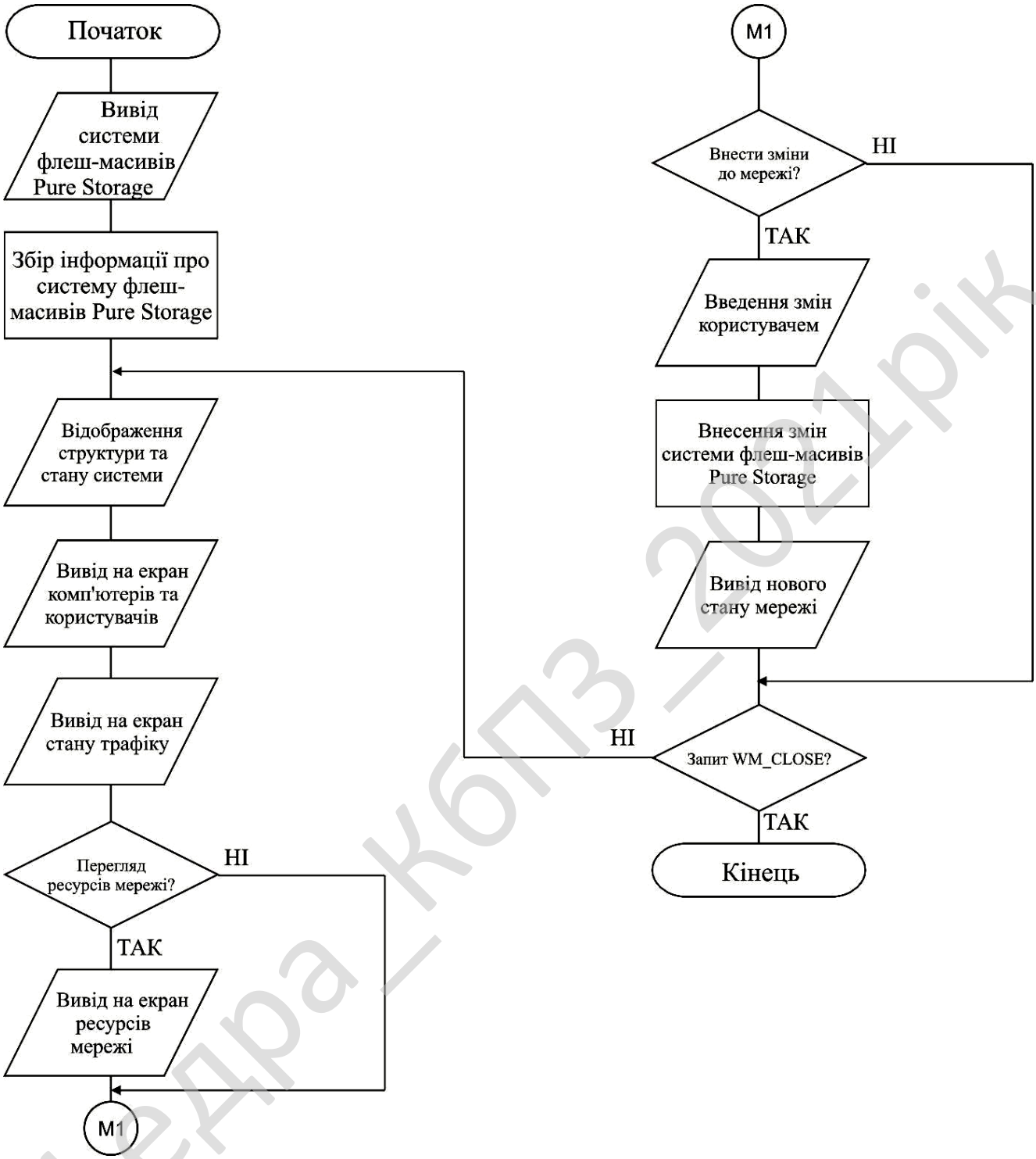

Рисунок 4.1 – Блок-схема основної програми

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;

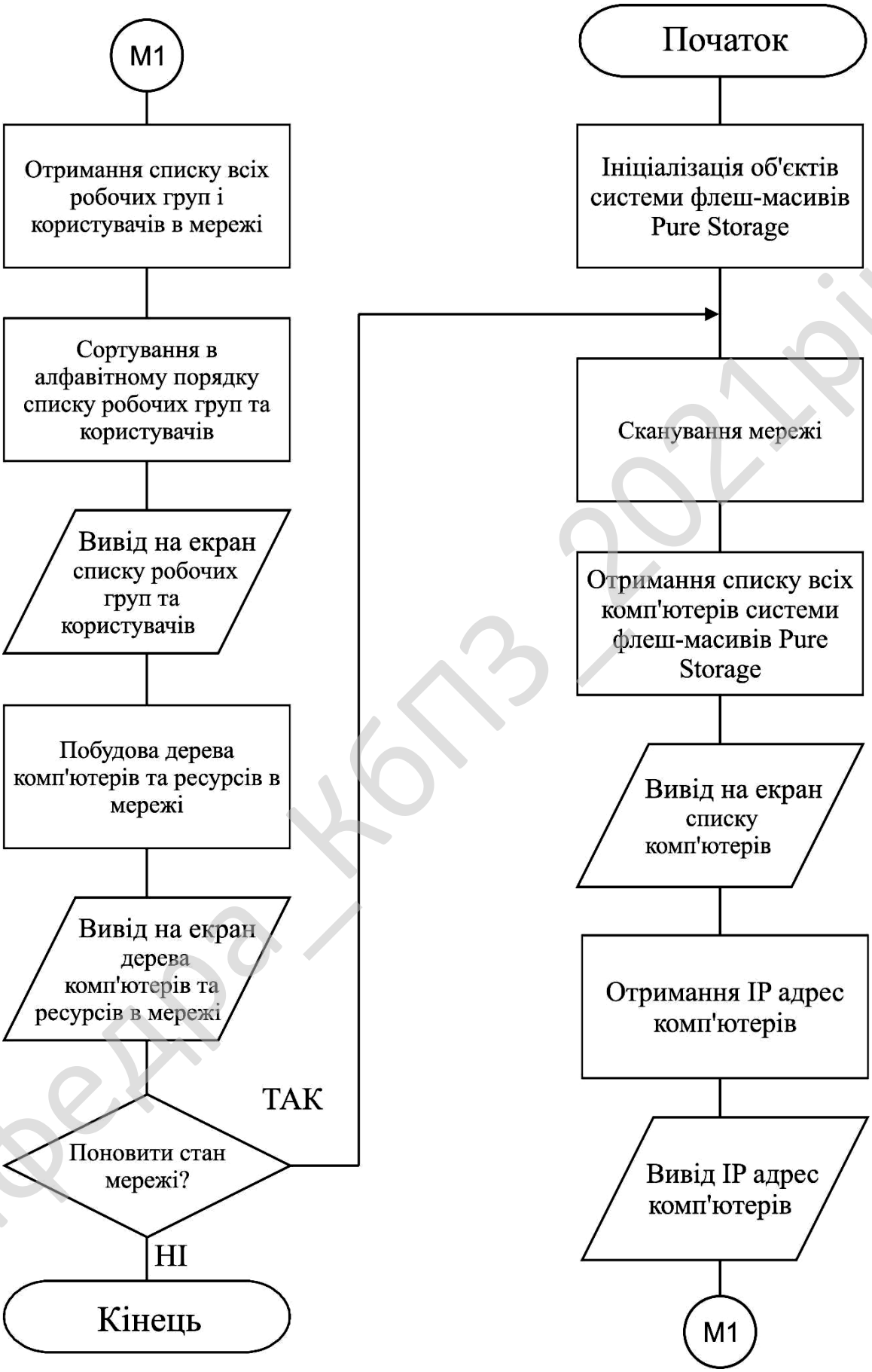

Рисунок 4.2 – Блок-схема роботи підпрограми

– 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 логіка 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).

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики XP зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Технологія екстремального програмування була розроблена Кентом Беком, Уардом Каннінгхемом та Роном Джеффріесом під час роботи над Chrysler Comprehensive Compensation System (C3). У 1996 Кент Бек став лідером проекту і почав вдосконалювати методи розробки, що застосовувалися в роботі над проектом. Свій метод він виклав у книзі «Extreme Programming Explained», котру було видано у жовтні 1999. Після купівлі Крайслера компанією Даймлер–Бенц проект C3 було скасовано у лютому 2000.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В XP ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які

перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

– Спроба примирити гуманність і продуктивність.

– Механізм для соціальної зміни.

– Шлях до удосконалення.

– Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою за заплановану (традиційна особливість для програмного забезпечення, що проектується).

XP використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

XP намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн.

Написання коду. Прихильники XP заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми XP. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).

1.1. Розробка через тестування (Test driven development).

1.2 Гра в планування (Planning game).

1.3. Замовник завжди поруч (Whole team, Onsite customer).

1.4 Парне програмування (Pair programming).

2. Безперервний, а не пакетний процес.

2.1 Безперервна інтеграція (Continuous Integration).

2.2 Рефакторинг (Design Improvement, Refactor).

2.3 Часті невеликі релізи (Small Releases).

3. Розуміння, що поділяється всіма учасниками.

3.1 Простота (Simple design).

3.2 Метафора системи (System metaphor).

3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).

3.4 Стандарт кодування (Coding standard or Coding conventions).

4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка

робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Microsoft SQL Server – комерційна система керування базами даних, що розповсюджується корпорацією Microsoft.

Мова, що використовується для запитів – Transact-SQL, створена спільно Microsoft та Sybase.

Transact-SQL є реалізацією стандарту ANSI/ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

Походження

Базовий код MS SQL Server (до версії 7.0) ґрунтувався на коді Sybase SQL Server. Це дозволило Microsoft вийти на ринок баз даних для підприємств, де конкурували Oracle, IBM, і, пізніше, сама Sybase. Microsoft, Sybase і Ashton-Tate спочатку об'єдналися для створення і випуску на ринок першої версії програми, що отримала назву SQL Server 1.0 для OS/2 (близько 1989 року), яка фактично була еквівалентом Sybase SQL Server 3.0 для Unix, VMS та ін. Microsoft SQL Server 4.2 був випущений у 1992 році та входив до складу операційної системи Microsoft OS/2 версії 1.3.

Офіційний реліз Microsoft SQL Server версії 4.21 для ОС Windows NT відбувся одночасно з релізом самої Windows NT (версії 3.1). Microsoft SQL Server 6.0 був першою версією SQL Server, створеною виключно для архітектури NT і без участі в процесі розробки Sybase.

До того часу, як вийшла на ринок ОС Windows NT, Sybase і Microsoft розійшлися та створювали вже власні моделі цього програмного продукту. Microsoft намагалася отримати виняткові права на всі версії SQL Server для Windows. Пізніше Sybase змінила назву свого продукту на Adaptive Server Enterprise щоб уникнути плутанини з Microsoft SQL Server. До 1994 року Microsoft отримала від Sybase три повідомлення про авторські права як натяк на походження Microsoft SQL Server.

Після розділення компанії зробили декілька самостійних релізів програм. SQL Server 7.0 був першим сервером баз даних зі справжнім графічним інтерфейсом адміністрування. Для усунення претензій з боку Sybase у порушенні авторських прав, весь успадкований код в сьомій версії був переписаний. Це

забезпечило також й успіх SQL Server 2000, який був першою редакцією, орієнтованою на архітектуру IA-64.

Протягом подальших шести років корпорація Microsoft працювала над вдосконаленням вже існуючої версії SQL Server 2000 доки не збудувала зручнішу систему Microsoft SQL Server 2005. Були вдосконалені продуктивність, клієнтські інструменти інтегрованого середовища розробки, а також у декількох додаткових системах, що встановлюються разом із SQL Server 2005. Змінено: інструментарій процесів керування сховищами даних (SQL Server Integration Services або SSIS), сервер звітів, сервер OLAP та інтелекутального аналізу даних (Analysis Services), а також декілька технологій повідомлень, особливо Service Broker та Notification Services.

SQL Server 2012+

SQL Server 2012 включає низку вдосконалень для роботи з критичними бізнес-застосунками і бізнес-аналітикою як в традиційних дата-центрах, так і в приватних, публічних і гібридних хмарних середовищах. Серед нових можливостей SQL Server 2012 виділяються SQL Server AlwaysOn (рішення підтримки високого рівня доступності даних та аварійного відновлення), xVelocity (технологія збільшення продуктивності сховищ даних та програм бізнес-аналітики), нові рішення в області візуалізації PowerPivot і PowerView для створення звітів і аналітичних програм з Excel і SharePoint, покращені інструменти для інтеграції даних і управління ними, включаючи SQL Server Data Quality Services і Master Data Services, нова семантична модель бізнес-аналітики та інструмент для адміністраторів баз даних і розробників застосунків SQL Server Data Tools.

Також Microsoft зробила значні інвестиції в області Big Data, а саме в інтеграцію SQL Server і популярних інструментів для бізнес-аналітики з неструктурованою інформацією.

Microsoft робить SQL Server доступним у різноманітних варіантах, які різняться наборами властивостей в залежності від цілей кінцевого користувача. Це такі редакції як:

SQL Server Compact Edition (SQL CE) – Компактний механізм бази даних. Завдяки малому обсягу (2 Мб для DLL) має зменшені властивості у порівнянні з іншими варіантами. Розмір бази даних обмежений 4 Гб і не може використовуватися як служба Windows.

SQL Server Express Edition – Раніше відомий під назвою MSDE (Microsoft SQL Server Desktop Engine), Microsoft SQL Server Express є вільно по ширюваною версією SQL Server. Дана версія має деякі технічні обмеження, також відсутні графічні інструменти адміністрування. Такі обмеження роблять її непридатною для розгортання великих баз даних. В основному вона використовується у застосунках, при проектуванні, або для самостійного вивчення. Розмір бази даних обмежений 4 Гб, розмір пам'яті, що може бути адресованою–1 Гб, підтримує лише один процесор.

SQL Server Workgroup Edition – SQL Server Workgroup Edition включає функціональність ядра бази даних, але не включає додаткові сервіси.

SQL Server Standard Edition – SQL Server Standard edition включає механізми ядра бази даних, а також автономні сервіси. Відрізняється від варіанту Enterprise Edition тим, що підтримує менше активних вузлів та не включає деякі функції збільшення продуктивності.

SQL Server Enterprise Edition – SQL Server Enterprise Edition –це повнофункціональна версія SQL Server

SQL Server Developer Edition – SQL Server Developer Edition включає ті самі функції, що й SQL Server Enterprise Editio n, але містить обмеження щодо використання його лише для розробки та тестування. Його ліцензія не дозволяє використання як виробничого сервера.

Функціональність

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком).

Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

Microsoft SQL Server також підтримує Open Database Connectivity (ODBC)– інтерфейс взаємодії застосунків з СУБД. Версія SQL Server 2005 надає можливість підключення користувачів через веб-сервер-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кроссплатформенно з'єднуватися з SQL Server. Microsoft також випустила сертифікований драйвер JDBC, що дозволяє застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

SQL Server підтримує дзеркалювання та кластеризацію баз даних. Кластер серверу SQL–це сукупність однаково конфігурованих серверів; така схема допомагає розподілити робоче навантаження між декількома серверами. Усі сервери мають одне віртуальне ім'я, а дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. Також у разі відмови або збою на одному з серверів кластеру доступне автоматичне перенесення навантаження на інший сервер.

SQL Server підтримує надлишкове дублювання даних за трьома сценаріями:

1. Знімок: Виконується «знімок» бази даних, який сервер відправляє одержувачам.

2. Історія змін: Всі зміни бази даних безперервно передаються користувачам.

3. Синхронізація з іншими серверами: Бази даних декількох серверів синхронізуються між собою. Зміни усіх баз даних відбуваються незалежно на кожному сервері, а під час синхронізації відбувається звірка даних. Дублювання такого типу передбачає можливість вирішення протиріч між базами даних.

SQL Server 2005 має вбудовану підтримку .NET Framework. Завдяки цьому, процедури бази даних, що зберігаються, можуть бути написані на будь-якій мові платформи .NET з використанням повного набору бібліотек, доступних для .NET Framework. На відміну від інших процесів, .NET Framework виділяє додаткову пам'ять і будує засоби керування SQL Server, не використовуючи вбудовані засоби Windows. Це підвищує продуктивність порівняно із загальними алгоритмами Windows, оскільки алгоритми розподілу ресурсів спеціально налагоджені для використання у структурах SQL Server.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Lucifer. Алгоритм Lucifer являє собою мережу перестановок і підстановок, його основні блоки нагадують блоки алгоритму DES. В DES результат функції f складається операцією XOR із входом попереднього раунду, утворюючи вхід наступного раунду. В S-блоках алгоритму Lucifer 4-бітові входи й виходи, вхід S -блоків являє собою перетасований вихід S-блоків попереднього раунду, входом S-блоків першого раунду служить відкритий текст.

Для вибору використовуваного S-блоку із двох можливих використовується біт ключа. (Lucifer реалізує все це в єдиному T-блоці з 9 бітами на вході й 8 бітами на виході).

На відміну від алгоритму DES, половини блоку між раундами не переставляються, та й саме поняття половини блоку в алгоритмі Lucifer не

використовується. У цього алгоритму 16 раундів, 128-бітові блоки й більше проста, чим в DES, схема розгорнення ключа.

Блок тексту розглядається як ненегативне ціле число, або як кілька незалежних ненегативних цілих чисел.

Довжина блоку завжди вибирається рівною ступеню двійки. У алгоритмі Lucifer використовуються наступні типи операцій:

– Таблична підстановка, при якій група біт відображається в іншу групу біт. Це так звані S-box.

– Переміщення, за допомогою якого біти повідомлення переупорядковуються.

– Операція додавання по модулю 2, позначувана XOR або ⊕.

– Операція додавання по модулю 2^{32} або по модулю 2^{16}.

– Циклічне зрушення на деяке число біт.

Ці операції циклічно повторюються в алгоритмі, створюючи так звані раунди. Входом кожного раунду є вихід попереднього раунду й ключ, що отриманий по певному алгоритму із ключа шифрування K.

Ключ раунду називається підключем. Алгоритм шифрування може бути представлений у такий спосіб:

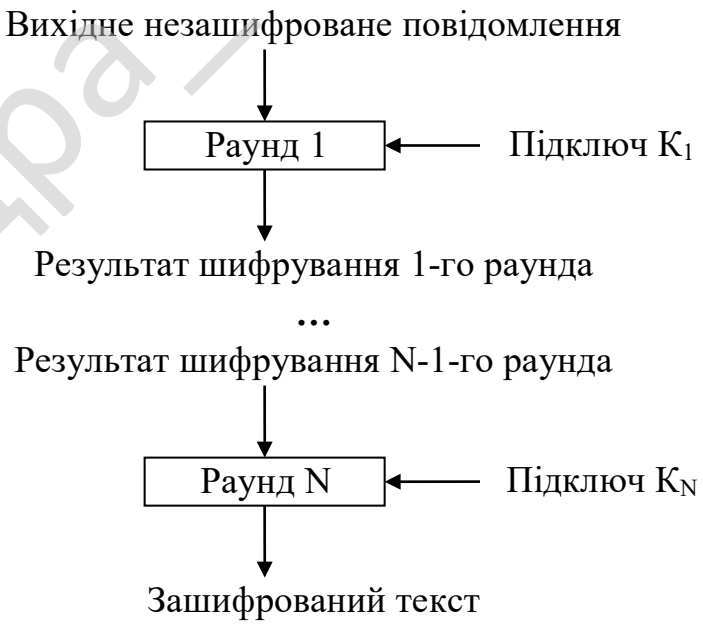

Рисунок 4.3 – Структура алгоритму алгоритмі Lucifer

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі система флеш-масивів Pure Storage для зберігання даних Big Data. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші; Верхнього меню; Розділу виведення результату роботи системи; Розділу обрання групи; Функції представлені у графічному вигляді (іконки); Функціональних кнопок ПЗ.

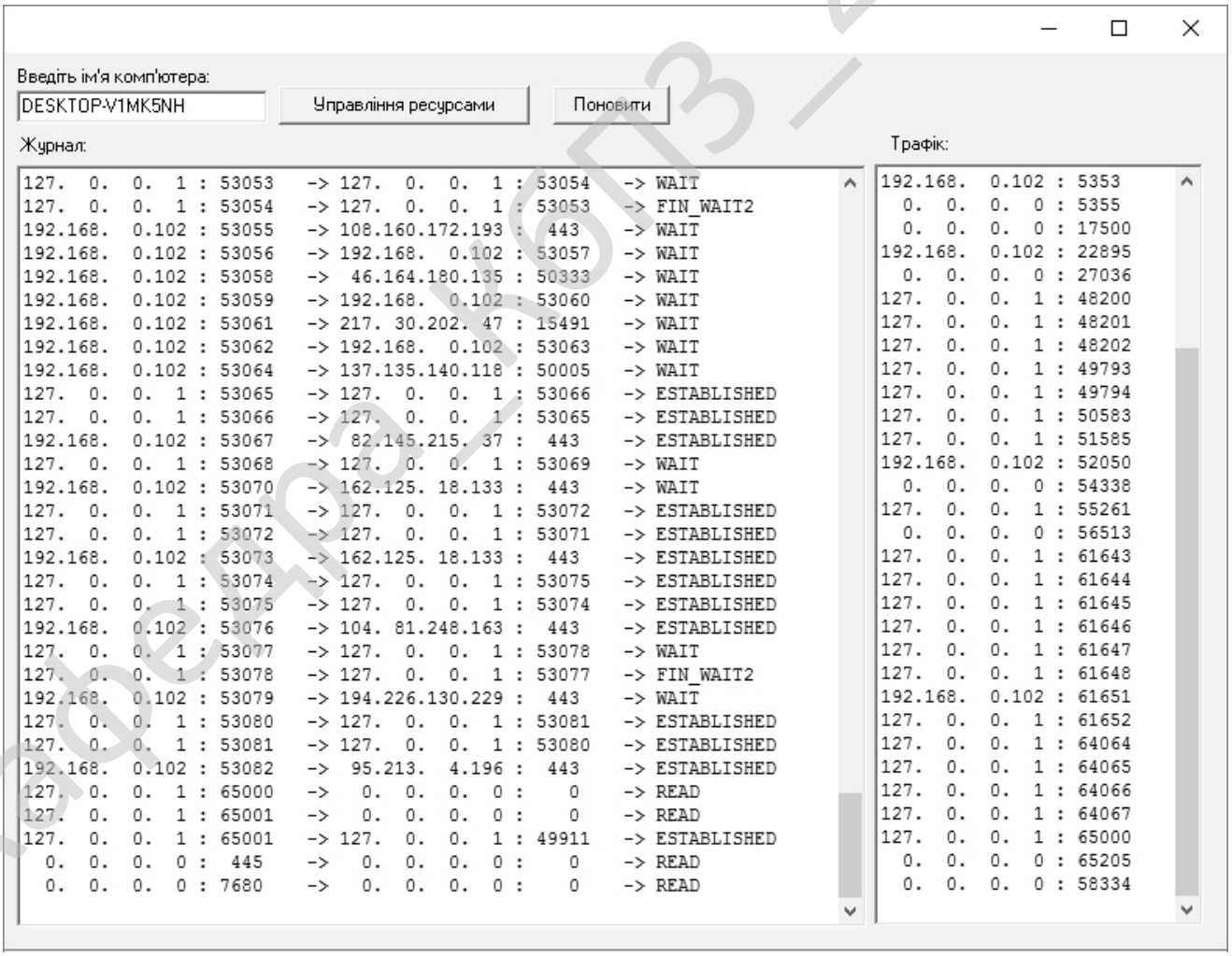

Рисунок 5.1 – Головне вікно ПЗ

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.

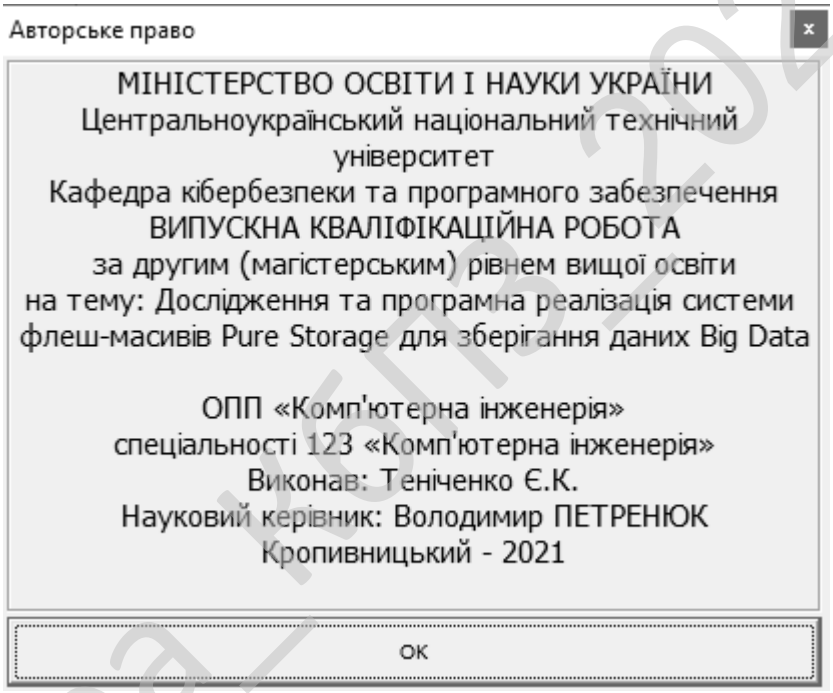

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих

процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

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

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 ITc).

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

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

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 – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

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

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи флеш-масивів Pure Storage для зберігання даних Big Data.

Метою розробки є дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

Об'єктом дослідження є процес флеш-масивів Pure Storage для зберігання даних Big Data.

Предметом дослідження є методи флеш-масивів Pure Storage для зберігання даних Big Data.

Методи дослідження базуються на методах теорії Big Data, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод флеш-масивів Pure Storage для зберігання даних Big Data.

– Розроблено вітчизняний продукт флеш-масивів Pure Storage для зберігання даних Big Data, 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 48 днів (два місяці). В магістерській роботі була досліджена та розроблена програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

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	20 (2 ост. цифри № зал)
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	20000 (2 ост. цифр № зал*10^4)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	37
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

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

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

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

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

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

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{227 \cdot 2}{1,2} = 378 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 378/(48 \cdot 8) = 1 \text{ ставка.}$$

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 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13000	39000
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	1,0	10500	31500
Інженер-системотехнік	0,25	10500	7875
Адміністратор мережі	0,5	10500	15750
Системний програміст	0,25	10500	7875
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 284325

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.

$$З_{сд} = \frac{284325}{8,25 \cdot 48} = 718 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

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

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 16.10.21 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	AMD Athlon II X4 740 3.2GHz/4MB	1750
Системна плата	Материнська плата MB ASUS F1A55-M L sFM1 mATX (A55 FCH (Hudson D2), sFM1 2xDDR3-2250(OC), VC – AMD Radeon HI 6000	1200
Відеокарта	Gigabyte GeForce 210 1024Mb 64bit	750
Жорсткий диск	Seagate Barracuda 7200.12 500GB 7200rpr 16MB ST500DM002 3.5 SATA III	1200
Оперативна пам’ять	Kingston DDR3-1600 4096MB PC3-12800	900
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silve body material – 0.6mm), 80mm fan (rear 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911
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	220

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)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат (МФУ)	1	5965	596,5	6561,5
Всього	–	–	–	199177

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Нематеріальні активи			
4. Нематеріальні активи	20000	10	2000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	5000
Разом	$К_р$ = 1807208		$А_р$ = 176988,5

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

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 718 \cdot 209/20 = 7503 \text{ грн.}$$

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, %.

$$З_д = 7503 \cdot 10 \cdot 0{,}01 = 750{,}3 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 37(7503+750{,}3) = 3054 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

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

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

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

Згідно виданих викладачем норм приймаємо одну пачку паперу на два місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 103 грн., визначаємо вартість паперу за період розробки $N_м$ = *2* міс:

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

$$З_{М1} = 103 \cdot 2 \cdot 0{,}5 = 105 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 5 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 22 грн./шт.

$$З_{М2} = 20 \cdot 5 + 12 = 112 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (105 + 112 + 1702)/20 = 96 \text{ грн.}$$

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, %.

$$О_п = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 20 прим.):

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

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

$$A_м = 176989 \cdot 2/(20 \cdot 12) = 1475 \text{ грн.}$$

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

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

$$C_п = 7305+730,5+3054+1125+96+1125+1475 = 14910,5 \text{ грн.}$$

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	$З_о$	7305
2. Додаткова зарплата виконавців	$З_д$	730,5
3. Відрахування на соціальні потреби	$C_{оц}$	3054
4. Загальногосподарські витрати	$Г_{осп}$	1125
5. Витрати на матеріали	$З_М$	96
6. Освоєння нових операційних систем, мов програмування	$О_п$	1125
7. Амортизація основних фондів	$A_м$	1475
8. Повна собівартість програмного забезпечення	$C_п$	14910,5
9. Плановий прибуток	$П_р$	7455,25
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	22365,75
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4473,15
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	26838,9

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 14910{,}5 = 7455{,}25 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	26839
Всього капітальних витрат	–	26839

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	$З_р$	25316	6329
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	6710
Всього витрат за рік	$І$	25316	13039

Витрати на профілактичні роботи:

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

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

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилось з 400 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 400 \cdot 6 \cdot 1{,}1 \cdot 1{,}37 \cdot 7 = 25316\ грн,$$

до:

$$З_{р\,нов} = 100 \cdot 6 \cdot 1{,}1 \cdot 1{,}37 \cdot 7 = 6329\ грн.$$

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 ($Ц_{ел}$):

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

Визначити різницю споживання електроенергії при впровадженні систем не має можливості, тому витрати на електроенергію в розрахунку приймаємо рівними нулю.

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	–	26839	–	6709,75
Всього відрахувань	-	–	26839	–	6709,75

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.

$$Ев = (22365 - 14910) \cdot 20 - (0,05 \cdot 1408000 + 0,5 \cdot 199177 + $$
$$+0,2 \cdot 143000 + 0,25 \cdot 37031 + 0,1 \cdot 20000) \cdot 2/12 = 104286 \text{ грн.}$$

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.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	14910
3. Ціна розробленої програми	Грн.	22365
4. Плановий прибуток від реалізації розробленої програми	Грн.	7455
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1807208
7. Загальний прибуток від реалізації програмної продукції	Грн.	149100
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	104286
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	26839
11.Величина економічного ефекту у користувача програмної продукції	Грн.	5567
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	2

$$T_в = \frac{399208}{(22365 - 14910) \cdot 20 \cdot 12 / 2} = 0{,}5 \ років .$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (25316\text{-}13039)\text{-}0{,}25 \cdot 26839 = 5567 \text{ грн.}$$

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

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

$$T_{сп} = \frac{26839}{25316 - 13039} = 2 \text{ роки}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж іс тотне навантаженням [1]. 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, та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом [1]. Так як р обота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

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

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	7,4
Довжина	8,2
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	6,7
Обсяг, V	$м^3$	не менше 20.0	20,2

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

У зазначеному приміщенні працє 9 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним

вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00 -1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

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

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

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	41-58	0,1
Тепла	23-25	50-70	0,1	23-24	56-70	0,11

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.

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер *Epson L120*, електродвигуни вентиляторів ЕОМ.

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

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р. можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи

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

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних 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).

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 данні для розрахунку захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Заземленню підлягає виробниче обладнання організації. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – супісок. Умовна товщина верхнього шару грунта: *H=1,2 м*. Для захисного заземлення: застосовуються ветикальні електроди – пруток довжиною *L=2,4 м.* Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*. Діаметр ветикальньного електрода (прутка) *D=80 мм*, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: *80 х 6 мм*. (*b=80 мм*.). Опір заземлювача, який нормується: $R_{ЗН}$ *= 4 Ом*. Глибина закладення контура заземлення *t= 0,7 м*.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 [3]. Додаткову перевірку отриманих результатів можна робити за допомогою он-лайн сервісу URL: https://kalk.pro/electricity/earthing/.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі [3]:

$$T=t+L/2=0,7+2,4/2=1,9 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв} = \psi \rho_1 \rho_2 L / [\rho_1 \psi(L-H+t) + \rho_2 \psi(H-t)] = 171,1 \text{ Ом}$$

де $\psi = 1,1$ табличне значення коефіцієнта сезонності у багатошаровому грунті;

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

$\rho_2 = 200$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта (джерело: https://ukrtechnobez.com.ua).

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[(ln)2L/D)+0,5ln[(4T+L)/(4T-L)] = 50,1 \text{ Ом.}$$

Опір розтіканню електричного струму заземлювача, який нормується, якщо $\rho_{екв} > 100$ *(у нас* $\rho_{екв} = 171,1 > 100$*):*

$$R = R_{ЗН} * \rho_{екв}/100 = 4*194,5/100 = 6,8 \text{ Ом.}$$

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_П = 0,366(\rho_{екв}\ \psi / L_П \cdot \eta_П) \lg(2 / L_П * L_П / bt) = 8,65 \text{ Ом.}$$

де $\eta_П = 0,27$ – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення *A/L*; довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру:

$$L_П = A*n = 3*14 = 42 \text{ м.;}$$

де n – ітераційна кількість вертикальних заземлювачів.

Загальниц опір штучного заземлюючого пристрою розтіканню електричного струму на землю:

$$R_B = R_П R / (R_П - R) = 3,8 \text{ Ом.}$$

Кількість вертикальних заземлювачів [3]:

$$n = R_O / R_B * \eta_B = 14 \text{ шт.}$$

де $\eta_B = 0{,}47$ – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L.

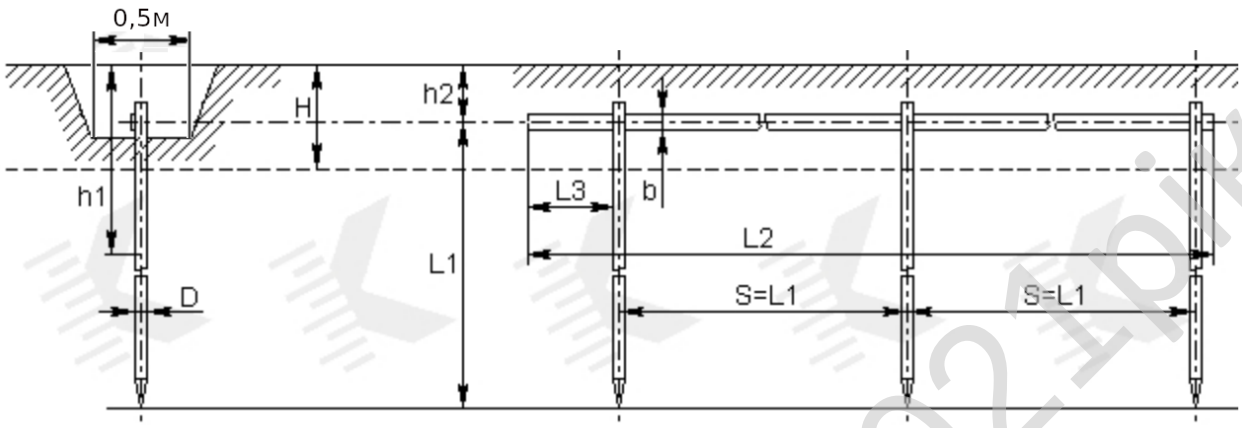

Рисунок 8.1 – Штучний заземлювач

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 флеш-масивів Pure Storage для зберігання даних Big Data.

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 флеш-масивів Pure Storage для зберігання даних Big Data.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем флеш-масивів Pure Storage для зберігання даних Big Data.

– Досліджена система флеш-масивів Pure Storage для зберігання даних Big Data.

– На основі отриманих результатів досліджень створена програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання флеш-масивів Pure Storage для зберігання даних Big Data.

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 RAD Studio Delphi 10.3. 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 Lucifer.

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 5567 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Теніченко Є.К. Дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.
2. Хаусли Т. Системы передачи и телеобработки данных: пер. с англ. / Т. Хаусли; под ред. Ю.М. Мартынова. – М.: Радио и связь, 1994. – 452 с.
3. Королев А.В. Адаптивная маршрутизация в корпоративных сетях / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2003. – 224 с.
4. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет / Евгений Андреевич Кучерявый. – СПб.: Наука и техника, 2004. – 336 с.
5. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.
6. Кучук Г.А. Управление ресурсами инфотелекоммуникаций / Г.А. Кучук, Р.П. Гахов, А.А. Пашнев. – М.: Физматлит, 2006. – 220 с.
7. Назаров А.Н. АТМ: Технология высокоскоростных сетей / А.Н. Назаров, М.В. Симонов. – М.: Эко-Трендз, 1997. – 232 с.
8. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров АТМ сетей / Алексей Николаевич Назаров. – М.: Горячая линия – Телеком, 2002. – 256 с.
9. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society– 2004, P. 1232-1236.
10. Дымарский Я.С. Управление сетями связи: Принципы, протоколы, прикладные задачи / Я.С. Дымарский, Н.П. Крутякова, Г.Г. Яновский. – М.: ИТЦ «Мобильные коммуникации», 2003. – 384 с.
11. Кириллов И.Г. Математическая модель адаптивной маршрутизации

цифровой информации о воздушной обстановке в телекоммуникационной сети перспективной АСУ авиации и ПВО / И.Г. Кириллов, С.Г. Семенов // Системи обробки інформації. – Х.: ХУ ПС, 2007. – № 5(63). – С.50-53

12. Королев А.В. Управление сетевыми ресурсами / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2004. – 272 с.

13. Круглов В.В., Дли М.И. Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода / В.В. Круглов, М.И. Дли – М.: Физматлит, 2002. – 193 с.

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

15. Кириллов И.Г. Комбинированный метод поиска множества путей передачи информации в телекоммуникационных сетях / И.Г. Кирилов, С.Г. Семенов // Збірник наукових праць ХУ ПС. – Х.: ХУ ПС, 2005. – №5(5) – С.90-94

16. Козелков С.В. Оптимальний вибір топології віртуальної підмережі транзакції телекомунікаційної системи / С.В. Козелков, М.Ю. Кузнєцова, С.Г. Семенов // Збірник наукових праць Об'єднаного науково-дослідного інституту Збройних Сил. – Х.: ОНДІ ЗС, 2006. – Вип. 2(4) – С.211 – 216.

17. Семенов С.Г. Математическая модель процесса доставки информационных пакетов в компьютерной сети системы критического применения / С.Г. Семенов, И.В. Ильина // Радіоелектронні і комп'ютерні системи. – 2008. №1. – С. 162-165.

18. Andrew B. Kahng, Stefanus Mantik, and Dirk Stroobandt. Toward Accurate Models of Achievable Routing // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2001. – №20(5), P. 648-659.

19. Basabi Chakraborty. Simultaneous Search for Multiple Routes using Genetic Algorithm / IEEE lntemational Conference on Computational oltelligencs for Measurement System and Applications Boston. MA, USA, 14-16, July 2004, P. 77-80/

20. Dan Pei, Dan Massey, Lixia Zhang. Detection of Invalid Routing

Announcements in RIP Protocol // IEEE GLOBECOM – 2003, P. 1450-1455.

21. Jun Wang, Klara Nahrstedt. Hop-by-Hop Routing Algorithms For Premium-class Traffic In DiffServ Networks // Proceedings of IEEE INFOCOM.– 2002, P. 705-714.

22. Nianjun Zhou, Huaming Wu, and Alhussein A. Abouzeid, Member. The Impact of Traffic Patterns on the Overhead of Reactive Routing Protocols // IEEE Journal on selected areas in communications. – 2005. – №23(3), P. 547-560.

23. Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The Stable Paths Problem and Interdomain Routing // IEEE/ACM Transactions on Networking.– 2002. – №10(2), P. 232-243.

24. Toyofumi Takenaka, Satosi Kato and Hidetosi Okamoto. Adaptive Load Balancing Content Address Hashing Routing For Reverse Proxy Servers // IEEE Communications Society. – 2004, P. 1522-1526.

25. Vutukury S., Garcia-Luna-Aceves J. J. MDVA: A Distance-Vector Multipath Routing Protocol // Proc. IEEE INFOCOM.– Anchorage, 2001. – P. 557-564.

26. Vutukury S., Garcia-Luna-Aceves J.J. MPATH: a loop-free multipath routing algorithm // Elsevier Journal of Microprocessors and Microsystems.– 2001. - 24(6). – P. 319-327.

27. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробиенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

28. Столлингс В. Современные компьютерные сети / Вильям Столлингс.– СПб.: Питер, 2003. – 778 с.

29. Сэломон Д. Сжатие данных, изображений и звука / Д. Сэломон.– М.: Техносфера, 2004. – 368 с.

30. Ершов В.А. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов – М.: Изд. МГТУ им. Н.Э. Баумана, 2003. – 432 с.

31. Dai Boong Lee and Hwangjun Song. Dynamic Class Selecting Mechanism for Guaranteed Service with Minimum Cost over Relative Differentiated-Services

Networks / IEEE International Conference on Multimedia and Expo (ICME) – 2004, P. 237-240.

32. Donna Ghosh, Venkatesh Sarangan, and Raj Acharya. Quality-of-Service Routing in IP Networks // IEEE Transactions on Multimedia. – 2001. – 3(2), P. 200-208.

33. Вишневский В.М. Широкополосные беспроводные сети передачи информации / В.М. Вишневский, А.И. Ляхов, С.Л. Портной, И.В. Шахнович. – М.: Техносфера, 2005. – 591 с.

34. Бодянский Е.В. Нейро-фаззи сети Петри в задачах моделирования сложных систем. Монография (научное издание) / Е.В. Бодянский., Е.И. Кучеренко, А.И. Михалев – Дніпропетровськ: Системні технології, 2005. –311 с.

35. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский. – М.: Техносфера, 2003. – 512 с.

36. Конахович Г.Ф. Сети передачи пакетных данных / Г.Ф. Конахович, В.М.Чуприн. – К.:МК-Пресс, 2006. – 272 с.

37. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

38. Уолрэнд Дж. Телекоммуникационные и компьютерные сети / Дж. Уолрэнд. – М.: Постмаркет, 2001. – 480 с.

39. Ma Anwar. Integrating knowledge-base and Dijkstra's algorithm for finding best alternate route dynamically // Proceedings IEEE INMIC – 2003, P. 428-433.

40. Mohamed G. Gouda, Member, IEEE, and Marco Schneider. Maximizable Routing Metrics // IEEE/ACM Transactions on networking. – 2003. – №11(4), P. 663-675.

41. Joo Young Hwang, Jun Song Kim, Sang Seok Lim, and Kyu Ho Park. A Fast Path Planning by Path Graph Optimization // IEEE Transactions on systems, man, and cybernetics-part a: systems and humans. – 2003. – №1, P. 121-128.

42. Jui-Fa Chen, Wei-Chuan Lin. A Message Interchange Protocol based on Routing Information Protocol in a Virtual World / Proceedings of the 19th International

Conference on Advanced Information Networking and Applications (AINA'05) – 2005, P. 201-208.

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

44. Босько В.В. Разработка методики оценки середнего времени обслуживания информационных пакетов в телекоммуникационной сети / Смирнов А.А., Босько В.В., Мелешко Е.В. // Системи управління, навігації та зв'язку. – Київ: ДП «Центральний науково-дослідний інститут навігації і управління», 2009. – Вип. 2(10). – С.162-165.

45. Босько В.В. Разработка математической модели процесса динамического управления маршрутизацией данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Системи управління, навігації та зв'язку. – К.: ДП «ЦНДІ навігації і управління», 2009. – Вип. 3(11). – С.208-210.

46. Босько В.В. Разработка метода определения оптимального множества путей передачи информации в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Збірник наукових праць. – Х.: ХУПС, 2009. – Вип. 3(21). – С.102-108.

47. В.В. Босько. Разработка метода прогнозирования поведения информационного потока в телекоммуникационной сети // Збірник наукових праць. Харківського університету Повітряних Сил: – Х.:ХУ ПС, – 2010.-Вип. 3 (25) .- С.126-130.

48. Босько В.В. Исследование особенностей построения современных телекоммуникационных систем и сетей / Смирнов А.А., Босько В.В. // Проблемы информатики и моделирования. Материалы восьмой международной научно-технической конференции 26-28 ноября. – Х.: НТУ "ХПИ", 2008. – С.54.

49. Босько В.В. Исследование влияния времени мониторинга телекоммуникационной сети на точность прогнозирования поведения трафика / Смирнов А.А., Босько В.В. // Перша міжнародна науково-практична конференція

“Проблеми й перспективи розвитку ІТ-індустрії” м. Харків , 18 -19 листопада 2009р. – С.198-200.

50. Босько В.В. Анализ математических моделей телекоммуникационной сети / Смирнов А.А., Босько В.В. // Проблемы информатики и моделирования. Материалы девятой международной научно-технической конференции 26-28 ноября. – Х.: НТУ “ХПИ”, 2009. – С.52-53.

51. Босько В.В. Метод повышения оперативности передачи данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Збірник тез доповідей науково-практичної конференції “Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку” 24-25 березня. – Х.: АВВ МВС України, 2010. – С.54.

52. Босько В.В. Динамическое управление процессом маршрутизации данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Тези доповідей Міжнародної науково-практичної конференції м. Вінниця, Україна 19-21 травня 2010 року. – Вінниця: ВНТУ, 2010. – С.231-232.

53. Можаєв О.О. Часова прозорість мережі, як характеристика, що визначає виконання необхідної якості обслуговування / О.О. Можаєв, О.Д. Анохіна, С.Ю. Гайдаров, С.Г. Семенов // Системи обробки інформації. – Х.: ХВУ, 2004. – Вип. 11(39). – С.133-139.

54. Матип Эссунга Лазар Методика выбора количества приоритетных пользователей в корпоративных вычислительных сетях реального времени «Системы управления и информационные технологии», Воронеж 2008, №4.2(30). 261-264с

55. Матип Эссунга Лазар, к вопросу исследования эффективности подсистемы клиент-сервер средствами системы массового обслуживания и теории очередей Труды Новомосковского института Российского химико-технологического университета им. Д.и, Менделеева Новомосковск, 2004 г. выпуск № 3 (14) 240-243с

56. Матип Эссунга Лазар, Абросимов Л.И Задачи управления

производительностью корпоративных вычислительных сетей XIII Международной научно-технической конференции студентов и аспирантов радиоэлектроника, электротехника и энергетика 1-2 марта 2007 г. тезисы докладов Изд-во МЭИ, том 1. 412-413с,

57. Матип Эссунга Лазар, Выбор количества приоритетных пользователей для управления производительностью КОМ на второй ежегодной международной конференции, организации сети Управления, "В IWAN '07", (Цюрих, Швейцария, 01-05 октября 2007) p. 312-314; Matip Essounga Lazare Users priority choice for management of performance of corporate computer network In Second annual International Working Conference on Active, Management Networking, «In IWAN'07», (Zurich, Switzerland, 01-05 October 2007) p. 312-314

58. Зеркалов Д. В. Охорона праці в Галузі: З агальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0020.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Теніченко Є.К.			Дослідження та програмна реалізація системи флеш-масивів Pure Storage для зберігання даних Big Data	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Петренюк В.І.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи флеш-масивів Pure Storage для зберігання даних Big Data.

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 флеш-масивів Pure Storage для зберігання даних Big Data.

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 флеш-масивів Pure Storage для зберігання даних Big Data;

– 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 забезпечення.

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 RAD Studio Delphi 10.3.

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 повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 128 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 20.12.2021 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

_____ Петренюк В.І.

Дослідження та програмна реалізація

системи флеш-масивів Pure Storage для зберігання даних Big Data

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Main.pas основної програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Dim, Networks, StdCtrls, ComCtrls, ImgList, ShellAPI, ShlObj, IpHlpApi,
IPtraff,
  ExtCtrls, About, Buttons;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    ClickMe: TButton;
    TreeView1: TTreeView;
    ImageList1: TImageList;
    ListView1: TListView;
    Memo2: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Memo3: TMemo;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    MemoTR: TMemo;
    Timer1: TTimer;
    Label7: TLabel;
    Button2: TButton;
    Button3: TButton;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    MemoX: TMemo;
    Label8: TLabel;
    procedure ClickMeClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);

  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

function GetShellFolder(CompName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=CompName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo3.Lines.Clear;
 Screen.Cursor:=crHourGlass;
 try
  if not EnumSharedResources(Edit1.Text, Memo3.Lines)
   then raise Exception.Create(‘ Невірне ім"я комп"ютера’ );
 finally
  Screen.Cursor:=crDefault;
 end;
end;



procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
 WorkgroupNode, ComputerNode: TTreeNode;
begin
 Screen.Cursor:=crHourGlass;
 try
  // Ініціалізація об’ єктів та сканування мережі системи флеш-масивів Pure
Storage для зберігання даних Big Data
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп’ ютерів в мережі системи флеш-масивів Pure
Storage для зберігання даних Big Data
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);

   // Отримання списку всіх робочих груп і комп’ ютерів в мережі системи флеш-
масивів Pure Storage для зберігання даних Big Data, відсортованих в алфавітному
порядку
   List:=TStringList.Create;
   ListView1.Items.Clear;
   try
    N.ListNetwork(List);
    for i:=0 to List.Count - 1 do begin
     ListViewItem:=ListView1.Items.Add;
     ListViewItem.Caption:=List[i];
     ListViewItem.ImageIndex:=Integer(List.Objects[i]);
    end;
   finally
    List.Free;
   end;
```

```
   // Побудова дерева робочих груп і комп’ ютерів в мережі системи флеш-масивів
Pure Storage для зберігання даних Big Data
   TreeView1.Items.Clear;
   for i:=0 to N.Count - 1 do begin
    WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
    WorkgroupNode.ImageIndex:=1;
    WorkgroupNode.SelectedIndex:=1;
    for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
     ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode, (N.Objects[i] as
TStrings).Strings[j]);
     ComputerNode.ImageIndex:=0;
    end;
   end;

  // Отримання IP адрес комп’ ютерів
  GetIPAddresses(N, Memo2.Lines);

  finally
   N.Free;
  end;
  TreeView1.FullExpand;

 finally
  Screen.Cursor:=crDefault;
 end;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:=GetComputerName;

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end;

end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin

   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;

end;

procedure TForm1.DOIpStuff;
begin

 Get_TCPTable( MemoX.Lines );
 Get_UDPTable( MemoTR.Lines );

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Form1.Close;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
```

```
begin
Form1.Close;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
Form2.Show;
end;


end.
```

Файл NetConst.pas – ініціалізація констант

```
unit NetConst;

interface

resourcestring
  SShellLinkReadError = ' Помилка  читання' ;
  SShellLinkWriteError = ' Помилка запису' ;
  SShellLinkLoadError = ' Не можу завантажити %s' ;
  SShellLinkSaveError = ' Не можу зберегти %s' ;
  SShellLinkCreateError = ' Не можу ініціалізувати інтерфейс' ;
  SDynArrayIndexError = ' У масиві %s немає елемента з таким індексом (%d)' ;
  SDynArrayCountError = ' Перевищена довжина масиву (%d)' ;
  SSharedMemoryError = ' Не можу створити файл' ;
  SCannotInitTimer = ' Не можу ініціалізувати таймер' ;
  SPrinterIndexError = ' Принтер не доступний (%d)' ;
  SIndicesOutOfRange = ' Недопустимий індекс матриці [%d: %d]' ;
  SRowIndexOutOfRange = ' Недопустиме занчення рядка матриці (%d)' ;
  SColIndexOutOfRange = ' Недопустиме занчення стовбчика матриці (%d)' ;
  SNoAdminRights = ' У Вас немає прав адміністратора' ;

  SFileError = ' Помилка %s файл %s%s' ;
  SFileReading = ' читання' ;
  SFileWriting = ' запис' ;
  SFileError002 = '  - файл не знайдено' ;
  SFileError003 = '  - шлях не знайдено' ;
  SFileError004 = '  - не можу відкрити файл' ;
  SFileError005 = '  - немає доступу' ;
  SFileError014 = '  - не достатньо пам"яті' ;
  SFileError015 = '  - не можу знайти драйвер' ;
  SFileError017 = '  - не можу перемістити файл' ;
  SFileError019 = '  - носій захищений від запису' ;
  SFileError020 = '  - не можу знайти пристрій' ;
  SFileError021 = '  - пристрій не відкривається для читання' ;
  SFileError022 = '  - пристрій не може розпізнати команду' ;
  SFileError025 = '  - вказана область не знайдена' ;
  SFileError026 = '  - пристрій недоступний' ;
  SFileError027 = '  - сектор не знайдено' ;
  SFileError029 = '  - помилка запису на пристрій' ;
  SFileError030 = '  - помилка читання з пристрою' ;
  SFileError032 = '  - файл використовується іншою програмою' ;
  SFileError036 = '  - занадто багато відкритих файлів' ;
  SFileError038 = '  - досягнутий кінець файлу' ;
  SFileError039 = '  - диск переповнений' ;
  SFileError050 = '  - запит не підтримується' ;
  SFileError051 = '  - віддалений комп' ютер недоступний' ;
  SFileError052 = '  - у мережі системи флеш-масивів Pure Storage для зберігання
даних Big Data знайдені ідентичні імена' ;
  SFileError053 = '  - мережний шлях не знайдено' ;
  SFileError054 = '  - мережа системи флеш-масивів Pure Storage для зберігання
даних Big Data зайнята' ;
  SFileError055 = '  - ресурс мережі системи флеш-масивів Pure Storage для
зберігання даних Big Data або пристрій недоступний' ;
  SFileError057 = '  - апаратна помилка в мережному адаптері' ;
  SFileError058 = '  - сервер не в змозі виконувати дану дію' ;
  SFileError059 = '  - помилка у мережі системи флеш-масивів Pure Storage для
зберігання даних Big Data' ;
  SFileError064 = '  - недоступне мережне ім"я' ;
  SFileError065 = '  - немає доступу до мережі системи флеш-масивів Pure Storage
для зберігання даних Big Data' ;
  SFileError066 = '  - невірно вказатий тип мережного ресурсу' ;
  SFileError067 = '  - не знайдене вказане мережне ім"я' ;
  SFileError070 = '  - відключейний сервер мережі системи флеш-масивів Pure
Storage для зберігання даних Big Data' ;
  SFileError082 = '  - не можу створити файл чи каталог' ;
  SFileError112 = '  - не достатньо вільного місця на диску' ;
```

```
  SFileError123 = '  - в імені файлу вказано недопустимий символ' ;
  SFileError161 = '  - неправильно вказано шдях' ;
  SFileError183 = '  - файл не існує' ;

  SCannotSetSize = ' Не можу змінити розмір файлу' ;

  SUnableToCompress = ' Не можу заархівувати дані' ;
  SUnableToDecompress = ' Не можу розархівувати дані' ;

  SCannotFindNetwork = ' Не можу знайти мережу системи флеш-масивів Pure Storage
для зберігання даних Big Data' ;

implementation

end.
```

Файл Networks.pas – робота з мережею системи флеш-масивів Pure Storage для зберігання даних Big Data

```
unit Networks;

interface

uses Windows, ShellAPI, ShlObj, ActiveX, ComObj, Dim, Classes, SysUtils,
WinSock;

type
  ComputerFound = class (Exception);
  ECannotFindNetwork = class (Exception);

  TStringObject = class (TObject)
  private
    FValue: TString;
    FTag: Integer;
    FData: Pointer;
    FRefObj: TObject;
    procedure SetValue(const Value: TString);
    procedure SetData(const Value: Pointer);
    procedure SetRefObj(const Value: TObject);
    procedure SetTag(const Value: Integer);
  public
    property Value: TString read FValue write SetValue;
    property RefObj: TObject read FRefObj write SetRefObj;
    property Tag: Integer read FTag write SetTag;
    property Data: Pointer read FData write SetData;
  end;

  TStringObjectArray = class (TDynamicArray)
  private
    function GetObject(Index: Integer): TStringObject;
    function GetData(Index: Integer): Pointer;
    function GetRefObj(Index: Integer): TObject;
    function GetTag(Index: Integer): Integer;
    function GetValue(Index: Integer): TString;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetRefObj(Index: Integer; const Value: TObject);
    procedure SetTag(Index: Integer; const Value: Integer);
    procedure SetValue(Index: Integer; const Value: TString);

    function FreeObject(Index: Integer; var Obj: TStringObject): Integer;

    procedure FreeItem(Index: Integer);
    procedure CreateItem(Index: Integer);

  protected
    procedure SetCount(const NewCount: Cardinal); override;
  public
    function Add: Integer; override;
    procedure Insert(Index: Integer); override;
    procedure Delete(Index: Integer); override;
    function AddItem(const Item): Integer; override;
    procedure InsertItem(Index: Integer; const Item); override;
    procedure DeleteItem(Index: Integer; out Item); override;
    property Value[Index: Integer]: TString read GetValue write SetValue;
default;
    property RefObj[Index: Integer]: TObject read GetRefObj write SetRefObj;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;
    property Data[Index: Integer]: Pointer read GetData write SetData;
    constructor Create;
    destructor Destroy; override;
  end;

  TStringObjectList = class (TStrings)
```

```
  private
    FArray: TStringObjectArray;
    function GetData(Index: Integer): Pointer;
    function GetTag(Index: Integer): Integer;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetTag(Index: Integer; const Value: Integer);
  protected
    function Get(Index: Integer): string; override;
    function GetCount: Integer; override;
    function GetObject(Index: Integer): TObject; override;
    procedure Put(Index: Integer; const S: string); override;
    procedure PutObject(Index: Integer; AObject: TObject); override;

  public
    property Data[Index: Integer]: Pointer read GetData write SetData;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Exchange(Index1, Index2: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;


    constructor Create;
    destructor Destroy; override;
  end;

  {TNetworkWorkgroup - клас, який створює список всіх комп' ютерів в робочій
групі. Цей клас є нащадком класу TStrings і повністю сумісний з іншим нащадками
цього класу.  Об' єкти цього класу записуються у властивості об' єктів класу
TNetworkNeiborhood. Клас TNetworkNeiborhood містить об' єкти і властивості
робочих груп. }

  TNetworkWorkgroup = class (TStringObjectList);


  {TNetworkNeiborhood - клас, який створює список всіх робочих груп в
копоратвній мережі}
  TNetworkNeighborhood = class (TStringObjectList)
  private
    function CreatePIDL(Size: Integer): PItemIDList;
    procedure DisposePIDL(ID: PItemIDList);
    function NextPIDL(IDList: PItemIDList): PItemIDList;
    function GetPIDLSize(IDList: PItemIDList): Integer;
    function CopyPIDL(IDList: PItemIDList): PItemIDList;
    procedure StripLastID(IDList: PItemIDList);
    function GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
    class function GetDisplayName(ShellFolder: IShellFolder; PIDL: PItemIDList):
TString;
    function OriginFolder:  IShellFolder;
    function OriginFolderNT: IShellFolder;
    class function EnumObjects(ShellFolder: IShellFolder): IEnumIDList;
    class procedure ParseFolder(Folder: IShellFolder; Items: TStringObjectList;
StorePIDLs: Boolean = False);
    class procedure ParseFolderEx(Folder: IShellFolder; Items: TStrings);

    function FreeRefObj(Index: Integer; var Obj: TStringObject): Integer;
    function GetWorkgroup(Name: TString): TNetworkWorkgroup;

  public
    { Процедура Оновлення шукає всі доступні робочі групи в мережі системи флеш-
масивів Pure Storage для зберігання даних Big Data }

    procedure Refresh;

    { містить списки всіх комп' ютерів в  мережі системи флеш-масивів Pure
Storage для зберігання даних Big Data}
```

```
    property Workgroup[Name: TString]: TNetworkWorkgroup read GetWorkgroup;

   { Функція FindComputer шукає комп’ ютер зі вказаним ім’ ям і повертає ім’ я
робочої групи де його знайдено, або порожню строку
     якщо комп’ ютера з таким іменем у мережі системи флеш-масивів Pure Storage
для зберігання даних Big Data немає }

    function FindComputer(Name: TString): TString;

    { Процедура ListComputers копіює список всіх комп’ ютерів в мережі системи
флеш-масивів Pure Storage для зберігання даних Big Data
      у об’ єкті TStrings}
    procedure ListComputers(Strings: TStrings);

    { Процедура ListNetwork копіює відсортований в алфавітному порядку список
всіх робочих груп і комп’ ютерів в мережі системи флеш-масивів Pure Storage для
зберігання даних Big Data.
     Робочі групи мають ‘ TObject(1)’  у відповідному елементі властивості об’
єктів, а комп’ ютери - ‘ nil’ }

    procedure ListNetwork(Strings: TStrings);

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;
    constructor Create;
  end;

{ Функція GetIPAddress отримує IP адресу комп’ ютера або сервера.
  Параметр NetworkName конкретизує ім’ я комп’ ютера або  сервера.
  Ця функція повертає адреси IP у форматі XXX.XXX.XXX.XXX у разі успішного
виконання, ‘ Error’  - коли неможливо ініціалізувати, ‘ Unknown’  – коли
параметр NetworkName посилається на неіснуючий комп’ ютер або на комп’ ютер  без
встановленого протоколу TCP/IP }

  function GetIPAddress(NetworkName: TString): TString;


{ GetIPAddresses отримує IP адреси всіх доступних комп’ ютерів мережі системи
флеш-масивів Pure Storage для зберігання даних Big Data }

procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);



{ Функція EnumSharedResources перераховує загальні ресурси мережі системи флеш-
масивів Pure Storage для зберігання даних Big Data. Параметр ComputerName
конкретизує
  ім’ я комп’ ютера. }

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;


implementation

uses NetConst;

{ TStringObject }

procedure TStringObject.SetData(const Value: Pointer);
begin
  FData := Value;
end;

procedure TStringObject.SetRefObj(const Value: TObject);
begin
  FRefObj := Value;
end;
```

```
procedure TStringObject.SetTag(const Value: Integer);
begin
  FTag := Value;
end;

procedure TStringObject.SetValue(const Value: TString);
begin
  FValue := Value;
end;

{ TStringObjectArray }

function TStringObjectArray.Add: Integer;
begin
 Result:=inherited Add;
 CreateItem(Result);
end;

function TStringObjectArray.AddItem(const Item): Integer;
begin
 Result:=Add;
end;

constructor TStringObjectArray.Create;
begin
 inherited Create(0, SizeOf(TStringObject));
end;

procedure TStringObjectArray.CreateItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 P^:=TStringObject.Create;
end;

procedure TStringObjectArray.Delete(Index: Integer);
begin
 FreeItem(Index);
 inherited;
end;

procedure TStringObjectArray.DeleteItem(Index: Integer; out Item);
begin
 Delete(Index);
end;

destructor TStringObjectArray.Destroy;
begin
 ForEach(Integer(Self), @TStringObjectArray.FreeObject);
 inherited;
end;

procedure TStringObjectArray.FreeItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 FreeAndNil(P^);
end;

function TStringObjectArray.FreeObject(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj);
 Result:=0;
end;
```

```
function TStringObjectArray.GetData(Index: Integer): Pointer;
begin
 Result:=GetObject(Index).Data;
end;

function TStringObjectArray.GetObject(Index: Integer): TStringObject;
begin
 GetItem(Index, Result);
end;

function TStringObjectArray.GetRefObj(Index: Integer): TObject;
begin
 Result:=GetObject(Index).RefObj;
end;

function TStringObjectArray.GetTag(Index: Integer): Integer;
begin
 Result:=GetObject(Index).Tag;
end;

function TStringObjectArray.GetValue(Index: Integer): TString;
begin
 Result:=GetObject(Index).Value;
end;

procedure TStringObjectArray.Insert(Index: Integer);
begin
 inherited;
 CreateItem(Index);
end;

procedure TStringObjectArray.InsertItem(Index: Integer; const Item);
begin
 Insert(Index);
end;

procedure TStringObjectArray.SetCount(const NewCount: Cardinal);
var
 i, OldCount: Integer;
begin
 OldCount:=Count;
 if NewCount > Count then begin
  inherited SetCount(NewCount);
  for i:=OldCount to NewCount - 1 do CreateItem(i);
 end else if NewCount < Count then begin
  for i:=NewCount to OldCount - 1 do FreeItem(i);
  inherited SetCount(NewCount);
 end;
end;

procedure TStringObjectArray.SetData(Index: Integer; const Value: Pointer);
begin
 GetObject(Index).Data:=Value;
end;

procedure TStringObjectArray.SetRefObj(Index: Integer;
  const Value: TObject);
begin
 GetObject(Index).RefObj:=Value;
end;

procedure TStringObjectArray.SetTag(Index: Integer; const Value: Integer);
begin
 GetObject(Index).Tag:=Value;
end;

procedure TStringObjectArray.SetValue(Index: Integer; const Value: TString);
begin
 GetObject(Index).Value:=Value;
```

```
end;

{ TStringObjectList }

function TStringObjectList.Add(const S: string): Integer;
begin
 Result:=FArray.Add;
 FArray.Value[Result]:=S;
end;

procedure TStringObjectList.Clear;
begin
 FArray.Count:=0;
end;

constructor TStringObjectList.Create;
begin
 inherited Create;
 FArray:=TStringObjectArray.Create;
end;

procedure TStringObjectList.Delete(Index: Integer);
begin
 FArray.Delete(Index);
end;

destructor TStringObjectList.Destroy;
begin
 FArray.Free;
 inherited;
end;

procedure TStringObjectList.Exchange(Index1, Index2: Integer);
begin
 FArray.Swap(Index1, Index2);
end;

function TStringObjectList.Get(Index: Integer): string;
begin
 Result:=FArray.Value[Index];
end;

function TStringObjectList.GetCount: Integer;
begin
 Result:=FArray.Count;
end;

function TStringObjectList.GetData(Index: Integer): Pointer;
begin
 Result:=FArray.Data[Index];
end;

function TStringObjectList.GetObject(Index: Integer): TObject;
begin
 Result:=FArray.RefObj[Index];
end;

function TStringObjectList.GetTag(Index: Integer): Integer;
begin
 Result:=FArray.Tag[Index];
end;

procedure TStringObjectList.Insert(Index: Integer; const S: string);
begin
 FArray.Insert(Index);
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.Put(Index: Integer; const S: string);
```

```
begin
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.PutObject(Index: Integer; AObject: TObject);
begin
 FArray.RefObj[Index]:=AObject;
end;

procedure TStringObjectList.SetData(Index: Integer; const Value: Pointer);
begin
 FArray.Data[Index]:=Value;
end;

procedure TStringObjectList.SetTag(Index: Integer; const Value: Integer);
begin
 FArray.Tag[Index]:=Value;
end;

{ TNetworkNeighborhood }

function TNetworkNeighborhood.Add(const S: string): Integer;
begin
 Result:=inherited Add(S);
 Objects[Result]:=TNetworkWorkgroup.Create;
end;

procedure TNetworkNeighborhood.Clear;
begin
 FArray.ForEach(Integer(Self), @TNetworkNeighborhood.FreeRefObj);
 inherited;
end;

function TNetworkNeighborhood.CopyPIDL(IDList: PItemIDList): PItemIDList;
var
 Size: Integer;
begin
 Size := GetPIDLSize(IDList);
 Result := CreatePIDL(Size);
 if Assigned(Result) then CopyMemory(Result, IDList, Size);
end;

constructor TNetworkNeighborhood.Create;
begin
 inherited Create;
 Refresh;
end;

function TNetworkNeighborhood.CreatePIDL(Size: Integer): PItemIDList;
var
 Malloc: IMalloc;
 HR: HResult;
begin
 Result := nil;
 HR := SHGetMalloc(Malloc);
 if Failed(HR) then  Exit;
 try
  Result := Malloc.Alloc(Size);
  if Assigned(Result) then FillChar(Result^, Size, 0);
 finally
 end;
end;

procedure TNetworkNeighborhood.Delete(Index: Integer);
begin
end;

procedure TNetworkNeighborhood.DisposePIDL(ID: PItemIDList);
var
```

```
 Malloc: IMalloc;
begin
 if ID = nil then Exit;
 OLECheck(SHGetMalloc(Malloc));
 Malloc.Free(ID);
end;

class function TNetworkNeighborhood.EnumObjects(
  ShellFolder: IShellFolder): IEnumIDList;
const
 Flags = SHCONTF_FOLDERS or SHCONTF_NONFOLDERS or SHCONTF_INCLUDEHIDDEN;
begin
 ShellFolder.EnumObjects(0, Flags, Result);
end;

function TNetworkNeighborhood.FindComputer(Name: TString): TString;
var
 i, j: Integer;
 List: TNetworkWorkgroup;
 S: TString;
begin
 Result:=' ' ;
 try
  for i:=0 to Count - 1 do begin
   List:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to List.Count - 1 do begin
    S:=List[j];
    CleanUp(S);
    if EqualText(Name, S) then begin
     Result:=Strings[i];
     raise ComputerFound.Create(' ' );
    end;
   end;
  end;
 except
  if not (ExceptObject is ComputerFound) then raise;
 end;
end;

function TNetworkNeighborhood.FreeRefObj(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj.FRefObj);
 Result:=0;
end;

class function TNetworkNeighborhood.GetDisplayName(ShellFolder: IShellFolder;
  PIDL: PItemIDList): TString;
var
 StrRet: TStrRet;
 P: PChar;
begin
 Result := ' ' ;
 ShellFolder.GetDisplayNameOf(PIDL, SHGDN_NORMAL, StrRet);
 case StrRet.uType of
  STRRET_CSTR: SetString(Result, StrRet.cStr, lStrLen(StrRet.cStr));
  STRRET_OFFSET: begin
   P := @PIDL.mkid.abID[StrRet.uOffset - SizeOf(PIDL.mkid.cb)];
   SetString(Result, P, PIDL.mkid.cb - StrRet.uOffset);
  end;
  STRRET_WSTR: Result := StrRet.pOleStr;
 end;
 CleanUp(Result, True);
end;

function TNetworkNeighborhood.GetPIDLSize(IDList: PItemIDList): Integer;
begin
 Result := 0;
 if Assigned(IDList) then begin
```

```
  Result := SizeOf(IDList^.mkid.cb);
  while IDList^.mkid.cb <> 0 do begin
   Result := Result + IDList^.mkid.cb;
   IDList := NextPIDL(IDList);
  end;
 end;
end;

function TNetworkNeighborhood.GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
var
 Temp: PItemIDList;
begin
 Temp := CopyPIDL(PIDL);
 if Assigned(Temp) then StripLastID(Temp);
 if Temp.mkid.cb <> 0 then Result:=Temp else Result:=nil;
end;

function TNetworkNeighborhood.GetWorkgroup(Name: TString): TNetworkWorkgroup;
var
 Index: Integer;
begin
 Index:=IndexOf(Name);
 if Index<>-1 then Result:=Objects[Index] as TNetworkWorkgroup else Result:=nil;
end;

procedure TNetworkNeighborhood.Insert(Index: Integer; const S: string);
begin
end;

procedure TNetworkNeighborhood.ListComputers(Strings: TStrings);
var
 i, j: integer;
 L: TNetworkWorkgroup;
 S: TString;
begin
 Strings.BeginUpdate;
 try
  Strings.Clear;
  for i:=0 to Count - 1 do begin
   L:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to L.Count - 1 do begin
    S:=L[j];
    CleanUp(S);
    Strings.Add(S);
   end;
  end;
 finally
  Strings.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.ListNetwork(Strings: TStrings);
var
 List: TStringList;
 i: Integer;
begin
 List:=TStringList.Create;
 try
  List.AddStrings(Self);
  for i:=0 to List.Count - 1 do List.Objects[i]:=TObject(1);
  for i:=0 to Count - 1 do begin
   List.AddStrings(Objects[i] as TStrings);
  end;
  for i:=Count to List.Count - 1 do List.Objects[i]:=nil;
  List.Sort;
  Strings.Assign(List);
 finally
  List.Free;
 end;
```

```
end;

function TNetworkNeighborhood.NextPIDL(IDList: PItemIDList): PItemIDList;
begin
 Result := IDList;
 Inc(PChar(Result), IDList^.mkid.cb);
end;

function TNetworkNeighborhood.OriginFolder: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString;
 P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 P:=StringToOleStr(S);
 Flags:=0;
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 try
   Network:=GetPrevPIDL(Workgroup);
  try
   Desktop.BindToObject(Network, nil, IShellFolder, Pointer(Result));
  finally
   DisposePIDL(Network);
  end;
 finally
  DisposePIDL(Workgroup);
 end;
end;

function TNetworkNeighborhood.OriginFolderNT: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString; W: WideString; P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
 NetShell: IShellFolder;
 Enum: IEnumIDList;
 ID: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 W:=S; P:=PWideChar(W);
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 Network:=GetPrevPIDL(Workgroup);
 Desktop.BindToObject(Network, nil, IShellFolder, NetShell);
 Enum:=EnumObjects(NetShell);
 Enum.Next(1, ID, Flags);
 NetShell.BindToObject(ID, nil, IShellFolder, Pointer(Result));
 DisposePIDL(Network);
 DisposePIDL(Workgroup);
end;

class procedure TNetworkNeighborhood.ParseFolder(Folder: IShellFolder;
  Items: TStringObjectList; StorePIDLs: Boolean);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
 Index: Integer;
begin
```

```
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Index:=Items.Add(S);
   if StorePIDLs then Items.Data[Index]:=ID;
  end;
 finally
  Items.EndUpdate;
 end;
end;

class procedure TNetworkNeighborhood.ParseFolderEx(Folder: IShellFolder;
  Items: TStrings);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Items.Add(S);
  end;
 finally
  Items.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.Refresh;
var
 Network: IShellFolder;
 Workgroup: IShellFolder;
 i: Integer;
begin
 try
  if WinNT and (not Win2K) then Network:=OriginFolderNT else
   Network:=OriginFolder;
  ParseFolder(Network, Self, True);
  for i:=0 to Count - 1 do begin
   Network.BindToObject(PItemIDList(Data[i]), nil, IShellFolder, Workgroup);
   ParseFolder(Workgroup, Objects[i] as TStringObjectList, False);
   Workgroup:=nil;
  end;
 except
  raise ECannotFindNetwork.Create(SCannotFindNetwork);
 end;
end;

procedure TNetworkNeighborhood.StripLastID(IDList: PItemIDList);
var
 MarkerID: PItemIDList;
begin
 MarkerID := IDList;
 if Assigned(IDList) then begin
  while IDList.mkid.cb <> 0 do begin
   MarkerID := IDList;
   IDList := NextPIDL(IDList);
  end;
  MarkerID.mkid.cb := 0;
 end;
end;
```

```
procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);
var
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
 i: Integer;
 TmpList: TStringList;
 S: TString;
begin
{ List.BeginUpdate;
 try}
  List.Clear;
  Error:=WSAStartup(MakeWord(1, 1), Data);
  if Error = 0 then begin
   TmpList:=TStringList.Create;
   try
    Network.ListComputers(TmpList);
    for i:=0 to TmpList.Count - 1 do begin
     HostEntry:=gethostbyname(PChar(TmpList[i]));
     Error:=GetLastError;
     if Error <> 0 then S:=‘ Unknown’  else begin
      Address:=PInAddr(HostEntry^.h_addr_list^)^;
      S:=inet_ntoa(Address);
     end;
     List.Add(Format(‘ %s [%s]’ , [TmpList[i], S]));
    end;
   finally
    TmpList.Free;
   end;
  end else begin
   List.Add(‘ Error’ );
  end;
{ finally
  List.EndUpdate;
 end;}
end;


function GetShellFolder(ComputerName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=ComputerName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;
var
 ShellFolder: IShellFolder;
begin
 ShellFolder:=GetShellFolder(ComputerName);
 Result:=Assigned(ShellFolder);
 if Result then TNetworkNeighborhood.ParseFolderEx(ShellFolder, List);
end;
```

```
function GetIPAddress(NetworkName: TString): TString;
var
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
begin
 Error:=WSAStartup(MakeWord(1, 1), Data);
 if Error = 0 then begin
  HostEntry:=gethostbyname(PChar(NetworkName));
  Error:=GetLastError();
  if Error = 0 then begin
   Address:=PInAddr(HostEntry^.h_addr_list^)^;
   Result:=inet_ntoa(Address);
  end else begin
   Result:=‘ Unknown’ ;
  end;
 end else begin
  Result:=‘ Error’ ;
 end;
 WSACleanup();
end;

end.
```

Файл IPtraff.pas – відслідковування та контроль трафіку мережі системи флеш-масивів Pure Storage для зберігання даних Big Data

```
unit IPtraff;

interface

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

const
  NULL_IP       = '   0.  0.  0.  0' ;


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


const

  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Пінгування    }
    ( 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'  ),     { Протокол File Transfer Protocol -
дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { Протокол File Transfer Protocol -
управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Протокол Simple Mail Transfer
Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Протокол Time Protocol }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { WHO IS сервіс  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP клієнт }
    ( 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'  ),     { SUN віддалений виклик функцій }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Локальний сервіce             }
    ( 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   '  )
    );
```

```
  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_ERROR' , ' 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'
  );


  ARPEntryType  : array[1..4] of string = ( ' Інший' , ' Несправний' ,
    ' Динамічний' , ' Статичний'
    );
  TCPConnState  :
    array[1..12] of string =
 ( ' CLOSED' , ' READ' , ' SYN_SENT' ,
    ' SYN_RCVD' , ' ESTABLISHED' , ' FIN_WAIT1' ,
    ' FIN_WAIT2' , ' WAIT' , ' CLOSING' ,
    ' LAST_ACK' , ' WAIT' , ' DELETE_TCB'    );

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

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

  IPForwProtos  : array[1..18] of string =
    ( ' OTHER' , ' 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 ;
    LeaseExpires: LongInt;
  end ;

  TAdaptorRows = array of TAdaptorInfo ;



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 );

// утіліти 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;


{ перетворення точена у рядок, потім рядок знищується }
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;


{ пертворення MAC-адреси до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    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  крапкову десяткову IP-адресу  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  номер порту в мережний байт типу  DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;
```

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


{ перетворення  відомого номера порту у 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;


{ фіксуємо параметри мережі системи флеш-масивів Pure Storage для зберігання
даних Big Data }

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( ' HOSTNAME          : '  + HostName );
        List.Add( ' DOMAIN            : '  + DomainName );
        List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP SCOPE        : '  + ScopeID );
        List.Add( ' ROUTING ENABLED   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   ENABLED   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     ENABLED   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS SERVER ADDR   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         //
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   //
begin
    InfoSize := 0 ;   //
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  //
    if result <> ERROR_BUFFER_OVERFLOW then exit ; //
    GetMem (FixedInfo, InfoSize) ;                    //
```

```
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   //
    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 ;  //
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  //
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    //
    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;


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

// включаємо байти вводу/виводу для кожного адаптеру

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 ; //
  TableSize := 0;
   //  перший виклик: беремо необхідний розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  //
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищуємо буфер, з W98 не потрібно

   // беремо показчик на таблицю
      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( ' Даних 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) ;  // перетворюємо
Unicode у string
             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),  //
counters are 32-bit
             sIfName, sDescr] )  //  , додаємо введення/вивід
             );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // звільняємо 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);  // очищуємо буфер, з W98 не
потрібно
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------------------------------------------------
{ інформація про мережні 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 масок для 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) ;  // more memory
            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 ;  jpt
  //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( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' -'  + Description ); // jpt : не корисне
            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 : not useful
            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 ;

//---------------------------------------------------------------------------
{ зчитуємо кількість раундів, та час звертання 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; //
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;

//---------------------------------------------------------------------------
{ ARP-таблиця – список відношень між віддаленими IP та віддаленими MAC-адресами.

}
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 );   //
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //.
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x - %12s - %16s - %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  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 розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  //
```

```
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яти, який потрібно та здійснюємо виклик 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 наступний 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 виведення         : '  + IntToStr( dwRTOMin )
+ '  ms'  );
      List.Add( ' Максимальний час виведення         : '  + 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( ' Відновлений зв'язок   : '  + IntToStr( dwEstabResets ) );
      List.Add( ' Поточний зв'язок .: '  + IntToStr( dwCurrEstab ) );
      List.Add( ' Отримано сегментів        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлено сегментів            : '  + IntToStr( dwOutSegs )
);
      List.Add( ' Ретрансльовано сегментів   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилки входження          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виведення          : '  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні 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 розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо потрібний розмір пам'яті, 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 наступний запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  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 );  //
  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( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we must restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
{ беремо дані з таблиці маршрутизатора Cisco}
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 ;   //  , враховуємо погані значення
          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( ' Даних немає.'  );
  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( ' Розблоковане пересилання      : '  + ' Так'  )
```

```
      else
        List.add( ' Розблоковане пересилання      : '  + ' Hi'  );
      List.add( ' Вбудований TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримані датаграми       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку     (в)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Адреса помилкова    (в)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
//
      List.add( ' Невідомий протокол (в)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Датаграма відвергнута     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Запит виведення            : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрутизація в маршрутизаторі Cisco відвергнута     : ' +
inttostr( dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (з)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Час виведення перебору      : '  + inttostr( dwReasmTimeOut )
);
      List.add( ' Запити перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Перебори здійснено : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Фрагментація мережі успішна: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграми фрагментовано    : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' маршрут в таблиці маршрутизації Cisco : '  + inttostr(
dwNumRoutes ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      //
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( ' Datagrams (в)    : '  + inttostr( dwInDatagrams ) );
      List.add( ' Datagrams (з)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' No Ports          : '  + inttostr( dwNoPorts ) );
      List.add( ' Errors    (в)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP Listen Ports  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     //
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( ' Отримано повідослень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилки ICMP              : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищено        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми параметрів      : '  + IntToStr( dwParmProbs ) );
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перенаправлення           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу   : '  + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адреси : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор маски адреси   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення відправлено        : '  + IntToStr( dwMsgs )
);
      ICMPOut.Add( 'Помилки ICMP            : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування    : '  + IntToStr( dwDestUnreachs ) );
      ICMPOut.Add( 'Час перевищено        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми параметрів      : '  + IntToStr( dwParmProbs ) );
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перенаправлення           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо повтор         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит маски адреси: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор маски адреси   : '  + 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 About.pas - довідка

```
unit About;

interface

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

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```