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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Марченко О.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____ 2023 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 2023 року

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

Марченку Олексію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску*

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

затверджені наказом вищого навчального закладу № 35-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Марченко О.В. Дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Метою розробки є дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Об'єктом дослідження є процес на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Предметом дослідження є методи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Методи дослідження базуються на методах теорії обчислювальних систем, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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

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

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

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерна інженерія, технологія самоконтролю, аналізу та звітності стану жорсткого диску

ABSTRACT

Marchenko O.V. Research and software implementation of the system based on the technology of self-monitoring, analysis and reporting of the state of the hard disk. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for a system based on the technology of self-monitoring, analysis and reporting of the state of the hard disk.

The purpose of the development is the research and software implementation of the system based on the technology of self-monitoring, analysis and reporting of the state of the hard disk.

The object of the study is a process based on the technology of self-monitoring, analysis and reporting of the state of the hard disk.

The subject of research are methods based on the technology of self-monitoring, analysis and reporting of the state of the hard disk.

Research methods are based on methods of the theory of computing systems, methods of mathematical statistics, and methods of software development.

The result of the work is the software implementation of the system based on the technology of self-monitoring, analysis and reporting of the state of the hard disk.

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

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

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

The program was developed in the Delphi 10 environment.

Keywords: computer engineering, technology of self-monitoring, analysis and reporting of hard disk status

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API	–	прикладний програмний інтерфейс
HDD	–	жорсткий диск
LBA	–	адресація секторів
MFC	–	Microsoft Foundation Class library
RO	–	тільки читання
S.M.A.R.T.	–	Self-Monitoring, Analysis and Reporting Technology
БМГ	–	блок магнітних головок
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Жорсткий диск є одним із найважливіших компонентів вашого ПК, оскільки він зберігає всі дані, створені, надані та збережені у вашій системі. Без пам'яті на жорсткому диску більшість функцій вашого ПК стають тимчасовими, оскільки ви не можете зберегти результат будь-якого завдання.

Це також один із найзавантаженіших пристроїв на вашому комп'ютері. Через таке суворе використання його потрібно регулярно перевіряти та обслуговувати, щоб запобігти будь-якій несправності.

Переобтяжені регулярними завданнями, ви не завжди маєте достатньо часу, щоб часто перевіряти стан жорсткого диска. Насправді більшість із нас не знають, як регулярно перевіряти стан жорсткого диска. Щоб вирішити цю проблему, багато програм перевірки стану жорсткого диска доступні в Інтернеті, якими ви можете скористатися.

S.M.A.R.T (від англ. self-monitoring, analysis and reporting technology – технологія самоконтролю, аналізу й звітності) – технологія оцінки стану НЖМД убудованою апаратурою самодіагностики, а також механізм пророкування часу виходу його з ладу.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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

– Огляд існуючих систем на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– Дослідження системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– Програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Об'єктом дослідження є процес на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Предметом дослідження є методи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Методи дослідження базуються на методах теорії обчислювальних систем, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– Розроблено вітчизняний продукт на базі технології самоконтролю, аналізу та звітності стану жорсткого диску, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску, 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

Жорсткий диск – це апаратний компонент, який використовується як у персональних комп'ютерах, так і на серверах для зберігання цифрових даних. Раптовий вихід з ладу жорсткого диска може призвести до остаточної втрати даних. Більшість жорстких дисків використовують технологію самоконтролю, аналізу та звітності (SMART), щоб відстежувати різні показники продуктивності та аналізувати власний стан. Однак не всі атрибути SMART ефективні для виявлення несправного жорсткого диска, тому адміністратори мережі та серверів повинні контролювати жорсткий диск, щоб забезпечити належну роботу сервера та підтримувати доступність мережі, уникаючи раптових непередбачених збоїв і помилок на жорстких дисках.

Жорсткі диски широко вважаються надійними компонентами, оскільки їхній середній час до виходу з ладу коливається від одного мільйона до 1,5 мільйона годин, що свідчить про низьку річну частоту відмов. Однак жорсткі диски є апаратним компонентом, який найчастіше змінюється у великомасштабній IT-інфраструктурі, і в більшості центрів обробки даних великий відсоток відомих збоїв спричинено несправними жорсткими дисками. Моніторинг жорсткого диска за допомогою комплексного монітора апаратного забезпечення для виявлення несправності є критичним для будь-якого бізнесу, щоб забезпечити доступність даних і уникнути незворотної втрати даних.

1.2 Область застосування

Більшість IT-мереж і центрів обробки даних використовують велику кількість жорстких дисків як пристрої для зберігання даних, і стає все складніше

контролювати та підтримувати надійність системи зберігання, оскільки кількість жорстких дисків зростає в геометричній прогресії. Нижче наведено деякі з проблем у моніторі жорсткого диска.

– Створення централізованої системи контролю та моніторингу для мереж багатьох постачальників у всьому світі.

– Проактивний монітор жорсткого диска для виявлення несправного обладнання до того, як воно виведе з ладу мережу.

– Аналіз основних сповіщень за типом сповіщень і визначення того, чи виправляються проблеми з жорстким диском, знизить частоту цих сповіщень.

– Можливість підтримувати оновлені пристрої, щоб забезпечити необмежене зростання мережевої інфраструктури.

– Надання вичерпних тенденцій дискового простору та звітів із використанням історичних даних, щоб IT-команда могла зробити усвідомлений вибір щодо змін у вимогах до жорсткого диска.

– Точне визначення причини проблем з продуктивністю жорсткого диска, щоб скоротити час для ремонту та мінімізувати час простою.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску, 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

Різні засоби перевірки стану жорсткого диска вважаються найкращими з усіх доступних на ринку. Однак лише деякі з них призначені для моніторингу стану жорсткого диска на ПК з Windows. Тому ми склали список найкращих програм на основі їхніх функцій, щоб допомогти вам прийняти зважене рішення.

1. Stellar Drive Monitor

Stellar Drive Monitor – це унікальний інструмент для перевірки стану жорсткого диска. Ви можете використовувати це програмне забезпечення для перевірки справності жорсткого диска/SSD/USB-накопичувача в Windows 11 і попередніх версіях. Програмне забезпечення на 100% безпечне та просте у використанні з чистим інтерфейсом користувача. Ось деякі відмінні риси Stellar Drive Monitor для Windows.

- Простий модуль стану диска для моніторингу температури, справності та продуктивності жорсткого диска.
- Функція стану SMART для детального звіту про критичні атрибути накопичувача з можливістю збереження звітів для довідки.
- Спеціальний модуль розділів диска для перевірки окремих розділів диска.
- Опція сканування диска для виявлення пошкоджених секторів на пристрої зберігання даних.
- Ефективно працює на внутрішніх і зовнішніх жорстких дисках PATA/SATA, SSD і жорстких дисках USB.
- Дозволяє клонувати диск. Ця функція особливо корисна для захисту

ваших цінних даних у складних ситуаціях, як-от збій диска.

– Якщо програмне забезпечення помічає будь-яку аномалію на диску, воно генерує швидкі попередження, щоб попередити користувача в режимі реального часу.

– Він надає користувачам прості для розуміння детальні звіти.

– Stellar Drive Monitor працює у фоновому режимі, не впливаючи на продуктивність системи, щоб контролювати стан жорсткого диска.

Недоліки:

– Він не підтримує моніторинг флеш-накопичувача USB, але функція сканування та клонування працює на флеш-накопичувачах USB, SD-картах тощо.

– Безкоштовна версія не дозволяє клонувати диск.

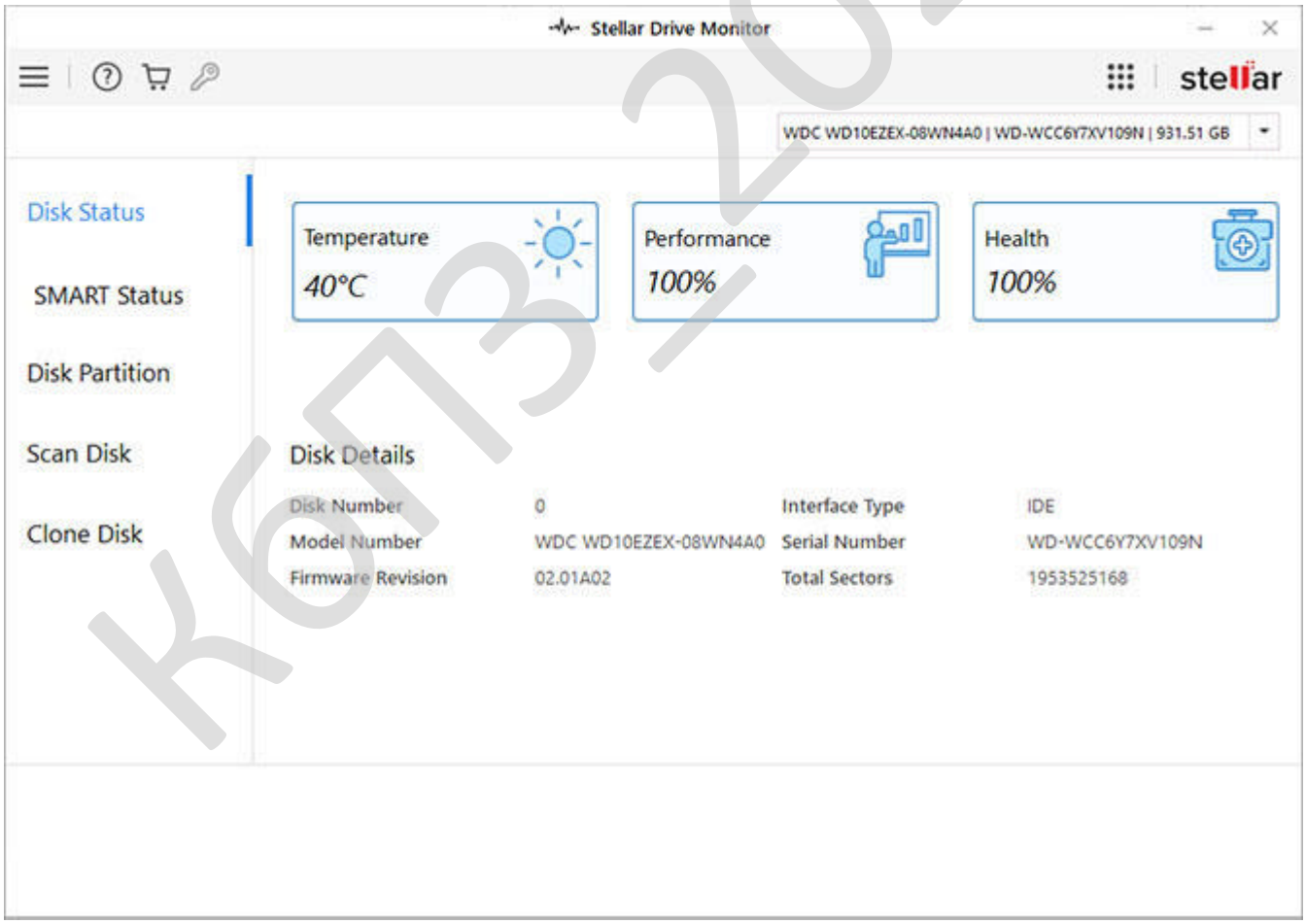


Рисунок 2.1 – Stellar Drive Monitor

2. Hard Disk Sentinel

Hard Disk Sentinel – це інструмент моніторингу жорстких дисків і SSD, сумісний із Windows 11 і попередніми версіями. Це ефективний інструмент для діагностики проблем з жорстким диском і моніторингу внутрішніх і зовнішніх дисків. Ось деякі важливі функції цього інструменту.

- Діагностує та відображає працездатність і продуктивність HDD і SSD.
- Надає текстовий опис проблем з жорстким диском і швидкі поради щодо виправлення простих помилок.
- Використовує швидкість передачі диска для аналізу продуктивності та робочого стану жорсткого диска.
- Оснащений декількома опціями сповіщень і звітів.
- Використовує систему оцінки здоров'я диска для надання вичерпних звітів про стан диска.
- Оцінює SMART-атрибути накопичувача, щоб попередити користувача у разі збою приводу.

Недоліки:

- Дуже дорого для інструмента Drive Monitoring.
- Розширені функції доступні лише у версії Pro.
- Функція сканування пошкоджених секторів недоступна.
- Не дозволяє клонувати диск у разі збою приводу.
- Складний інтерфейс користувача потребує досвіду використання програмного забезпечення.

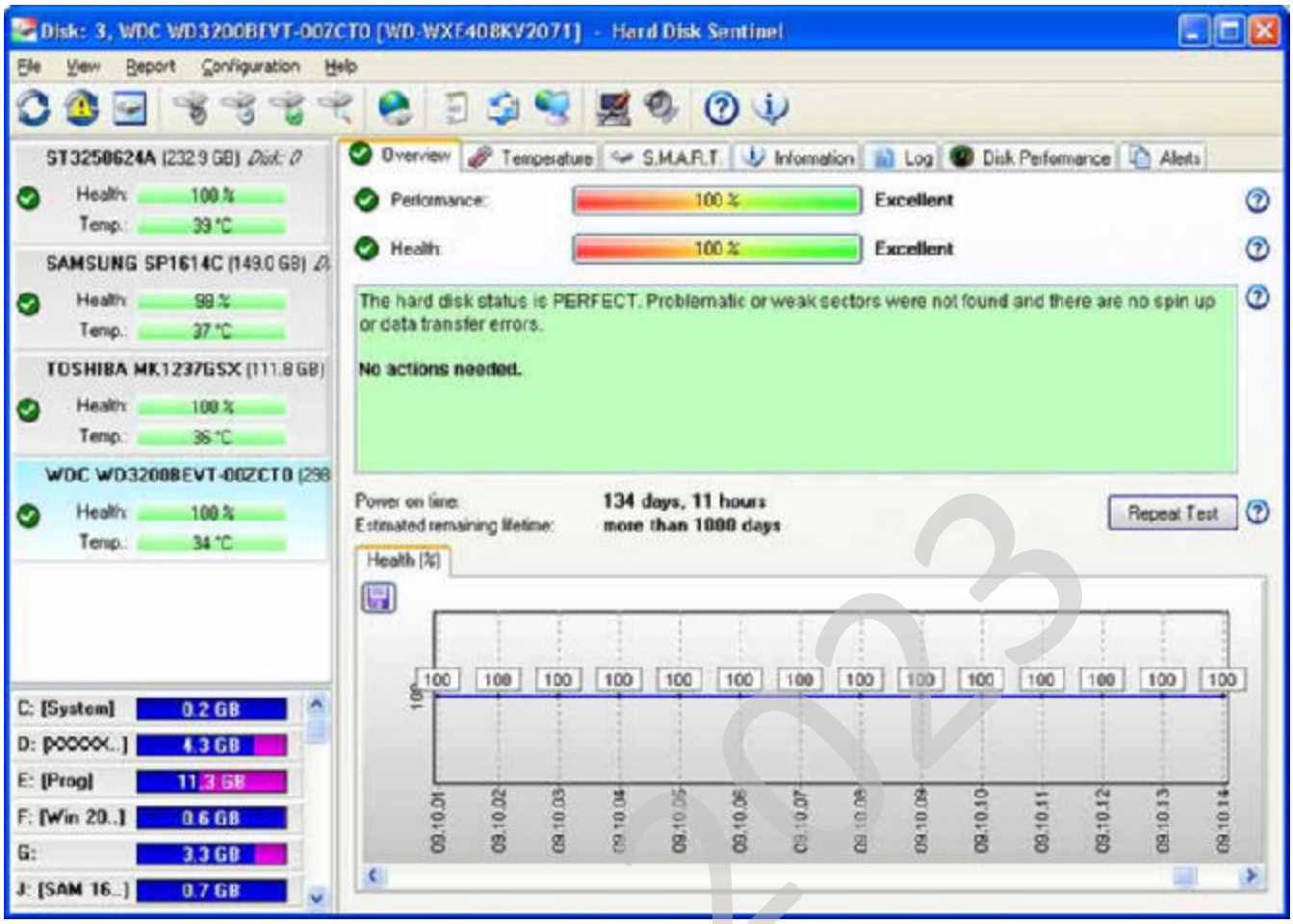


Рисунок 2.2 – Hard Disk Sentinel

3. HDDScan

HDDScan – це інструмент діагностики жорсткого диска з можливістю зміни деяких параметрів жорсткого диска, як-от AAM, APM тощо. Він корисний для проведення регулярних тестів працездатності жорсткого диска. Деякі інші функції програмного забезпечення обговорюються нижче.

- Перевіряє температуру та продуктивність накопичувача.
- Можливість проведення тестів читання/запису на диску.
- Підтримує SATA/ATA HDD, SSD та USB-накопичувачі.
- Читає та аналізує параметри SMART на ATA/SATA/USB/FireWire HDD.
- Повідомляє інформацію про помилку для жорсткого диска SCSI.
- Доступна як безкоштовна програма з можливістю друку згенерованих

звітів.

Недоліки:.

- Немає вбудованої функції для відновлення несправного диска.
- Немає автоматичних сповіщень або попереджень про помилки.
- Звіти про помилки можна зберегти лише у форматі MHT.
- Несумісний з Windows 11.

Рисунок 2.3 – HDDScan

4. Seagate SeaTools для Windows

SeaTools 5 для Windows – це комплексне програмне забезпечення для перевірки працездатності жорстких дисків/твердотільних накопичувачів із кількома варіантами тестування, як-от самоперевірка короткого диска, інформація про диск тощо. Програмне забезпечення розроблено одним із провідних виробників жорстких дисків Seagate. Розкажіть нам про додаткові

функції цього програмного забезпечення.

– Він поставляється з п’ятьма варіантами тестування для ретельної діагностики жорсткого диска.

– Ефективно працює на внутрішніх і зовнішніх жорстких дисках у Windows 11 і попередніх версіях ОС.

– Програмне забезпечення підтримує жорсткі диски SATA, PATA, USB, 1394, eSATA, SAS і SCSI.

– Легко тестує жорсткі диски незалежно від виробника.

– Надає інформацію про численні атрибути диска, такі як серійний номер, ємність, версія мікропрограми тощо.

– Прості функції та сумісність майже з будь-яким жорстким диском роблять його придатним для новачків.

Недоліки:

– Вам потрібно отримати SeaTools DOS для розширених функцій.

– Немає можливості перевірити диск на наявність помилок, таких як пошкоджені сектори, пошкодження тощо.

– Це лише програмне забезпечення для тестування без можливості захисту даних.

– Не працює у фоновому режимі. Щоб запустити цей інструмент, потрібно закрити всі функції.

5. PassMark DiskCheckup

DiskCheckup дозволяє користувачам відстежувати атрибути SMART своїх жорстких дисків, відображаючи поточні та порогові значення для кожного атрибута. Це спеціальне програмне забезпечення для перевірки стану жорсткого диска для Windows 11 і попередніх версій. Деякі інші характеристики цього програмного забезпечення:

– Відстежує зміни в атрибутах SMART і передбачає попередню ймовірність помилки та збою.

– DiskCheckup може виявити та визначити розмір прихованих областей

жорсткого диска, таких як Host Protected Area (HPA) і Device Configuration Overlay (DCO).

– Відображає інформацію про пристрій, наприклад серійний номер, номер моделі, швидкість обертання носія тощо.

– Можливість налаштувати сповіщення електронною поштою, коли будь-яке значення атрибута жорсткого диска перетинає порогове значення.

– Генерує прогнозовану дату виходу з ладу диска, використовуючи статистичні дані продуктивності диска.

Недоліки:

– Відсутність сумісності з деякими старими жорсткими дисками.

– Користувачі повідомили про помилки в програмному забезпеченні.

– Немає голосової підтримки.

– Складний інтерфейс користувача не підходить для новачків.

– Порівняно дорога ліцензія.

6. GSmartControl

GSmartControl – це програмне забезпечення для аналізу та контролю даних SMART на найновіших жорстких і твердотільних накопичувачах. Це не тільки дозволяє контролювати дані SMART, але й допомагає запускати спеціальні тести для перевірки стану жорсткого диска. Давайте перевіримо ще деякі функції цього інструменту моніторингу стану жорсткого диска.

– Автоматично виявляє будь-які проблеми на жорсткому диску та повідомляє про них.

– Функцію SMART можна ввімкнути/вимкнути.

– Вбудована функція для самотестування SMART.

– Відображає інформацію про жорсткий диск, таку як ємність, серійний номер тощо.

– Доступний як безкоштовне програмне забезпечення за ліцензією GNU General Public License.

– Підтримує всі основні ОС, включаючи Windows, Linux, macOS тощо.

Недоліки:

– Результати тесту надто вичерпні для нового користувача.

– Користувачі повідомили про проблеми з надійністю деяких операційних систем.

– Жодна функція виправлення не доступна у випадку збою диска.

7. Western Digital DashBoard

Western Digital (WD) Dashboard – це інструмент аналізу та моніторингу, який допоможе вам підтримувати максимальну продуктивність дисків WD у Windows. Програмне забезпечення також має вбудовану опцію стирання. Деякі інші функції цього інструменту перевірки стану жорсткого диска:

– Це дозволяє постійно оновлювати продуктивність і стан жорсткого диска.

– Програмне забезпечення надає детальну інформацію про жорсткий диск, включаючи серійний номер, номер моделі та ємність.

– Програмне забезпечення має параметри Quick Test і Extended Test для виконання перевірки диска за вимогою.

– Якщо ви хочете стерти дані на своєму диску, WD DashBoard дозволяє перезаписати наявні дані та стерти їх за допомогою параметра «Записати нулі».

– Є окремий розділ для перегляду результатів тесту.

Недоліки:

– Програма працює лише для пристроїв Western Digital.

– Немає можливості усунути проблеми, виявлені на жорсткому диску.

– Відображає неточні дані SMART під час багаторазового тестування.

– Немає автоматичного моніторингу. Щоб отримати повні результати діагностики, потрібно виконати тести.

8. Acronis Drive Monitor

Acronis Drive Monitor – це базовий монітор жорсткого диска. Однак те, що відрізняє його, так це надзвичайно простий інтерфейс користувача. Таким чином, програмне забезпечення ідеально підходить для тих, хто користується вперше та

шукає інструмент із базовими функціями моніторингу. Ось деякі інші функції цього інструменту.

– Він показує диски з їхнім станом справності, температурою та часом роботи на головному екрані.

– Програмне забезпечення має вкладку SMART для відображення атрибутів ваших дисків.

– Вкладка подій відображає всі помилки, які можуть існувати на вашому жорсткому диску.

– Він легко інтегрується з продуктами Acronis Home and Business.

– Інструмент має вбудований параметр резервного копіювання, доступ до якого здійснюється з лівої панелі.

– Зберігає записи про критичні події та помилки в системі та жорсткому диску.

Недоліки:

– Занадто простий із лише основними функціями, не підходить для комплексної перевірки стану жорсткого диска.

– Він надає лише інформацію про помилки на диску, але не створює докладних звітів.

– Хоча є резервний варіант, він не дуже ефективний для боротьби з помилками на жорсткому диску.

9. HD Tune

HD Tune – ще одне ефективне програмне забезпечення для перевірки працездатності жорсткого диска в списку. Він створює графічне представлення внутрішньої та зовнішньої інформації на жорсткому диску. Програмне забезпечення доступне лише у версії Pro, оскільки безкоштовна версія припинена. Перегляньте функції HD Tune, щоб зрозуміти, як працює цей інструмент.

– Він розміщує піктограму на панелі інструментів для відображення температури жорсткого диска.

– Дозволяє отримати доступ до даних атрибутів SMART у реальному часі.

– Ви можете миттєво перевірити справність і продуктивність усіх підключених дисків.

– Генерує попередження з рівнем загрози, щоб допомогти вам проаналізувати серйозність проблеми.

– Відображає детальну продуктивність блоків різних розмірів.

Недоліки:

– Висока вартість версії Pro, оскільки безкоштовна версія більше не доступна.

– Лише виявляє проблеми та попереджає про них без можливості їх усунення.

– Доступно лише для Windows 7 або попередніх версій, оскільки не було оновлень з 2008 року.

– Не вдається отримати звіти як текстовий файл, щоб зберегти їх для подальшого використання.

– Доступно лише для домашнього/особистого використання.

10. CrystalDiskInfo

CrystalDiskInfo – це програмне забезпечення для перевірки працездатності жорсткого диска з відкритим кодом і багатьма функціями. Він надає майже всі дані, пов’язані з вашим жорстким диском, і може бути використаний як надійний інструмент моніторингу диска. Давайте розглянемо ще деякі функції цього інструменту перевірки диска.

– У головному вікні програмного забезпечення відображається загальна інформація про диск, як-от стан працездатності, температура тощо.

– Функція стану SMART надає докладніші дані про ваш диск.

– Він включає в себе автоматичне акустичне керування (AAM) і розширену функцію керування живленням для керування продуктивністю накопичувача залежно від навантаження.

– За допомогою цього інструменту можна отримати графічне

представлення даних SMART.

– Є можливість отримати сповіщення на пошту.

Недоліки:

– Немає можливості врятувати дані у разі збою диска.

– Користувачі повідомили про проблеми з надійністю, оскільки програмне забезпечення є інструментом з відкритим кодом.

– Головний екран занадто переповнений через різні набори даних, які відображаються на одному екрані.

Висновок

З наведеним вище детальним списком інструментів перевірки працездатності жорсткого диска тепер ви можете вибрати найкращий відповідно до ваших потреб. Отримавши програмне забезпечення для моніторингу дисків, вам не потрібно турбуватися про перевірку справності жорсткого диска комп'ютера. Деяке програмне забезпечення, згадане вище, автоматично працює у фоновому режимі, щоб відстежувати продуктивність, температуру, працездатність та інші параметри вашої системи.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних

багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++

використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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

S.M.A.R.T

Технологія SMART дозволяє здійснювати:

– моніторинг параметрів стану;

– сканування поверхні;

– сканування поверхні з автоматичною заміною сумнівних секторів на надійні.

S.M.A.R.T дозволяє пророкувати вихід пристрою з ладу в результаті механічних несправностей, що становить близько 60 % причин, по яких вінчестери виходять із ладу. Пророчити наслідку стрибка напруги або ушкодження накопичувача в результаті удару SMART не здатна.

Накопичувачі не можуть самі повідомляти про свій стан за допомогою технології SMART, для цього існують спеціальні програми. Використання технології SMART неможливо без наявності наступних двох складових:

– ПЗ, убудованого в контролер накопичувача.

– Зовнішнього ПЗ, убудованого в хост.

Програми, що відображають стан SMART-атрибутів, працюють за наступним алгоритмом:

– Перевірка наявності підтримки технології SMART накопичувачем.

– Подається в накопичувач команду запиту SMART-таблиць.

– Одержання таблиці в буфер додатка.

– Розбираються табличні структури, витягаючи з них номера атрибутів і їхні числові значення.

– Зіставляються стандартизовані номери атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виготовлювача НЖМД).

– Виводяться числові значення в зручному для сприйняття виді (отут кожний програміст може робити по-своєму, наприклад, конвертувати HEX-Значення в десяткові).

– Витягають із таблиць прапори атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware накопичувача, наприклад, «життєво важливий» або «лічильник»).

– На підставі всіх таблиць, значень і прапорів виводяться загальний стан пристрою.

Живлення

Однією із причин помилок у роботі є живлення. До цього привів неухильний ріст технічних характеристик, насамперед щільності запису й часу доступу. Швидке позиціонування головок вимагає точно керованого струму в котушці, що відхиляє, і будь-які перепади напруги заважають процесу.

Диски 3.5 живляться від ліній 5 В (процесор і інші сигнальні ланцюги) і 12 В (шпіндельний двигун і привод головок), причому основні проблеми доставляє контур 12 В. Справа в тому, що ця лінія випробовує різкий сплеск навантаження при включенні диска, коли відбувається розкручування шпінделя й распарковка блоку магнітних головок. Стартовий струм на 4-15 секунд досягає 1.2-2.5 А, при сталому споживанні всього 0.4-0.9 А. Особливо ненажерливі в цьому плані диски Seagate Barracuda: так, у сімействі 7200.11 пікове споживання може досягати 3.0 А.

У типових БЖ лінія 12 В не має своєї незалежної системи стабілізації, і при росту навантаження напруга може «законно» знижуватися на 0.5-0.6 В (стандартом ATX допускаються відхилення ±5% від номіналу, що в цьому випадку становить діапазон 11.4-12.6 В). Додамо до цього спадання напруги в сполучних проводах і розніманнях, і одержимо на контактах ЖД до 11.3 В, при якому багато дисків уже не можуть нормально працювати. Останні моделі Seagate, наприклад, вимагають не менш 11.5 В.

Схеми, що стежать, щоб уникнути падіння головок на пластини, аварійно паркують БМГ і зупиняють шпіндель. Споживання по 12 В знижується, стабілізація в блоці живлення відновлюється, і при номінальних напругах диск виходить на новий цикл старту.

Зовні все це виглядає, як клацання усередині системного блоку з періодичністю 6-10 сек. Диск, природно, не зорієнтується: для цього він повинен вийти на номінальні оберти, провести рекалібровку й взяти паспорт зі службової зони. У підсумку весь комп'ютер непрацездатний.

Охолодження

Проблема нагрівання, і відповідно, відводу тепла – одна із самих гострих для сучасних НЖМД. Високооборотний шпіндель, швидкодіючий привод головок, і, нарешті, щільний потік даних при операціях читання й запису (до 100 Мбайт/с) вимагають значних витрат енергії. Типові накопичувачі середнього класу (нагадаємо, це форм-фактор 3.5″, швидкість обертання 7200 об/хв і інтерфейс PATA/SATA) споживають 4-9 Вт у режимі простою, і 8-18 Вт при активній роботі – пересиланню даних і пошуку. Стартова потужність при розкручуванні шпінделя значно вище (16-35 Вт), але такий режим короткочасний, до 10-15 сік, і на загальне нагрівання диска практично не впливає.

Вся ця потужність (з точністю до 1%) в остаточному підсумку виділяється у вигляді тепла, чим і пояснюється значне нагрівання НЖМД. Але ж він дуже шкідливий для механіки, і особливо для читаючих головок – ключового елемента всієї конструкції. Багатошарові тонкошарові магнітні резистори реагують як на магнітне поле, так і на температуру.

При тривалому перегріві головки деградують, їхня віддача (ступінь зміни опору залежно від намагніченості) зменшується, і зрештою мікропрограма при всіх математичних хитруваннях не може розпізнати, що саме записано на пластині – 0 або 1. Це стосується не тільки й не стільки користувальницьких даних: критично важливі для роботи сервомітки й модулі службової зони точно так само зчитуються усе гірше. Диск починає стукати, зорієнтується й у підсумку повністю виходить із ладу.

Тому виробники відміряють нинішнім НЖМД порівняно вузький діапазон робочих температур: нагрівання корпуса, обмірюваний у центрі кришки, не повинен перевищувати 60°, при температурі навколишнього середовища +5...55 °, рідше 0...60 °.

Нагрівання по SMART вище 45° украй небажаний: він ускладнює функціонування механіки ЖД (потрібні зайві рекалібровки), підвищує ймовірність помилок у даних, а головне – різко підсилює зношування головок читання. За деяким даними, кожні додаткові 5° прискорюють їхню деградацію вдвічі. Так що навіть нетривалий, але сильний перегрів (викликаний, наприклад, піковими навантаженнями, несправним вентилятором або просто жаркою погодою) ризикує відчутне скоротити життя диска, не говорячи про страшний – аварії.

Найгірший результат – заклинений шпіндель. Гідродинамічні підшипники сучасних накопичувачів, при всіх своїх перевагах (менший шум і нагрівання, здатність гасити вібрації) виявилися схильні до заклинювання в умовах підвищених температур. Видимо, погрішності у виготовленні перекреслюють теоретичні достоїнства конструкції. У деяких гарячих сімействах «клин» став нещастям.

Основним методом охолодження сучасних НЖМД залишається примусовий обдувши за допомогою вентилятора. Інші варіанти тепловідвода – пасивні радіатори, теплові трубки, рідинні системи – не одержали поширення, хоча ряд фірм (зокрема, Zalman і Scythe) у різний час пропонував подібні рішення. Вони були безшумні, довговічні, але відрізнялися громіздкістю й високою ціною, що визначило вузьку нішу на ринку (складання особливо тихих комп'ютерів)

Підбор кулера для дисків має свою специфіку. Насамперед, загальне тепловиділення НЖМД і особливо його щільність порівняно малі, тому досить легені вітерцю, щоб зняти перегрів. Також, що оптимальна температура диска під навантаженням становить 35-40^0 (приблизно на 10^0 вище навколишнього середовища) і що всі його поверхні варто прохолоджувати рівномірно.

У подібних умовах кращим вибором стане тихохідний великогабаритний вентилятор, що дме в торець кошика із НЖМД, але не дотичний її щоб уникнути вібрацій. Саме так улаштований обдувши кошика в сучасних якісних корпусах.

Вентилятор кріпиться до вирізу передньої панелі, а декоративна кришка постачена повітрязаборниками. Витяжка через задню панель, що часто зустрічається в корпусах середнього класу, також досить ефективна (звичайно, при належній герметизації інших місць).

Практика показала, що 120-мм вентилятор здатний прохолоджувати до п'яти НЖМД, так що потреби звичайних користувачів покриваються повністю. Для одного-двох дисків обдув навіть надлишковий, так що з метою зниження шуму можна зменшити швидкість обертання до 600-1000 про./хв.

Несправності апаратної частини

Ушкодження серворозмітки

У результаті ушкодження серворозмітки може стати не читаємою деяка область диску, причому спроби читання можуть супроводжуватися тривалими затримками й сильним стукотом головок. Як правило диском ще можна користуватися, працюючи в його справній частині, і навіть можна досягти повної функціональності, відкоригувавши списки збійних треків у службовій області диска, однак область ушкодженої серворозмітки виявляється безповоротно загубленої, неї відновити можна тільки за допомогою серворайтера.

Серворайтер – прилад для нанесення сервометок на млинці харда. Застосовується тільки на заводі, жодна ремонтна майстерня не може собі дозволити мати його – це дуже дорогий прилад (~100 тис $). Принцип дії серворайтера заснований на русі головок зовнішнім механічним пристроєм через технологічні вікна в гермоблоці. Для контролю за переміщенням використовується лазерний приціл і явище інтерференції світлових хвиль. Після закінчення серворозмітки технологічні вікна заклеюються плівкою й на гвинт установлюється плата електроніки.

Bad-блоки

Bad-блоки (биті сектори) – найпоширеніша проблема у всіх накопичувачів. Bad-блоки – це сектора жорсткого диска, які є збійними або що не читаються, ці сектори є механічно ушкодженими або магнітними здатностями, що втратили свої. Відбувається це, як правило через удари, струсів, падіння головки, або просто згодом від зношування.

Проявляються дані несправності в повільній роботі жорсткого диска, помилках при записі або читанні даних з HDD, і як наслідок незначна втрата даних. Можливі так само більше серйозні наслідки, такі як втрата логічних розділів, або неможливість запустити ОС.

Відновлення ушкоджених секторів у цьому випадку можливо тільки при низькорівневому форматуванні, наприклад програмою MHDD, але це можливо тільки при втраті своїх властивостей магнітною поверхнею згодом, але ніяк не при механічних ушкодженнях. Але й низькорівневе форматування може не дати ніякого результату, або дати його ненадовго.

Якщо низькорівневе форматування не допомогло, і биті сектори залишилися, то єдиним способом ремонту залишається – "викинути" ці сектори зі службової зони, для того щоб жорсткий диск не робив на них запис і читання. Але для цієї операції необхідний програмно апаратний комплекс PC3000, що виконає цю операцію на низькому рівні. Обсяг жорсткого диска при цьому не змінитися, "викинуті" сектори будуть замінені резервними.

Несправності програмної частини

Ушкодження службової інформації

Ушкодження службової інформації – досить розповсюджена несправність. Вона може приводити як до часткового, так і до повної втрати функціональності диска. Ремонт такого диска можливий, але для цього необхідно мати інформацію зі структури цієї області, а вона в основному є закритої, і відповідні інструменти, наприклад, комплекс PC-3000. Можливість відновити роботу накопичувача багато в чому залежить від характеру ушкоджень, так, наприклад, при втраті

транслятора й списку збійних секторів стає практично неможливо відновити інформацію

Втрата інформації в ПЗП плати контролера й інша інформація, необхідна для роботи

Втрата цих даних практично завжди веде до повної непрацездатності вінчестера. Відремонтувати диск можна, замінивши плату контролера або переписавши вміст ПЗП. Звичайно перезапис виробляється за допомогою програматора, іноді, якщо контролер зберігає здатність обробляти деякі службові команди – силами самого контролера. Варто помітити, що нерідко контролер містить трохи ПЗП довільної комбінації. Зустрічаються масочне неперезаписуєме ПЗП усередині контролера, перезаписуване, котре може перебувати як усередині мікросхеми контролера, так і окремо, а також окреме послідовне ПЗП. Для зчитування інформації також можна тимчасово поставити сумісну плату від робочого диска замість тій, яка має цю несправність.

3.2 Розробка структурної схеми

Стан працездатності оцінюється по декількох параметрах роботи накопичувача, які називаються атрибутами надійності – attributes. Кожний атрибут має свій номер – ID (ідентифікатор). Атрибутам надійності відповідають параметри роботи накопичувача, які можуть характеризувати його природне зношування й передаварійний стан.

Більшість S.M.A.R.T. HDD мають від 3 до 15 атрибутів надійності. Максимально можлива їхня кількість 30. Состав і кількість атрибутів надійності визначаються самими виробниками індивідуально для кожного типу HDD.

Значення атрибутів надійності можуть лежати в діапазоні від 1 до 253. Спочатку атрибути мають максимальні значення. У міру виробітку ресурсу вінчестера або у випадку виникнення передаварійного стану значення атрибутів надійності зменшуються. Отже, високе значення атрибутів говорить про низьку

ймовірність виходу накопичувача з ладу й, відповідно, низьке значення атрибутів – про низьку надійність накопичувача й високої ймовірності виходу його з ладу. Як правило, верхні границі атрибутів надійності мають значення 100 (IBM, Quantum, Fujitsu) або 253 (Samsung). Але є й виключення, так в HDD Western Digital моделей WDAC34000, WDAC33100, WDAC31600 перший атрибут надійності має максимальне значення 200, а інші 100.

Для кожного атрибута надійності розроблювачами HDD визначається граничне значення – threshold. Якщо хоча б одне зі значень атрибутів менше, ніж відповідне граничне значення, то зберігати дані на такому вінчестері стає небезпечно.

Значення thresholds залишаються постійними протягом всього життя накопичувача, а значення attributes зменшуються, наближаючись до граничного (thresholds).

Крім граничного значення для кожного атрибута визначений додатковий параметр pre-failure/advizory, так-же характеризує передаварійний стан накопичувача. Можливі три стани накопичувача, характеризуємі станом pre-failure/advizory і значенням атрибута надійності:

– pre-failure/advizory = 0, характеризує високий запас надійності накопичувача за умови, що значення атрибута надійності більше відповідного граничного значення;

– pre-failure/advizory = 1, характеризує низький запас надійності накопичувача за умови, що значення атрибута надійності менше відповідного граничного значення;

– pre-failure/advizory = 2, характеризує передаварійний стан накопичувача за умови, що значення атрибута надійності менше відповідного граничного значення.

Всі S.M.A.R.T. параметри – attributes, thresholds і pre-failure/advizory зберігаються в енергонезалежній пам'яті HDD.

На рисунку 3.1 зображена структурна схема розробленої системи моніторингу стану жорсткого диску з використанням технології SMART.

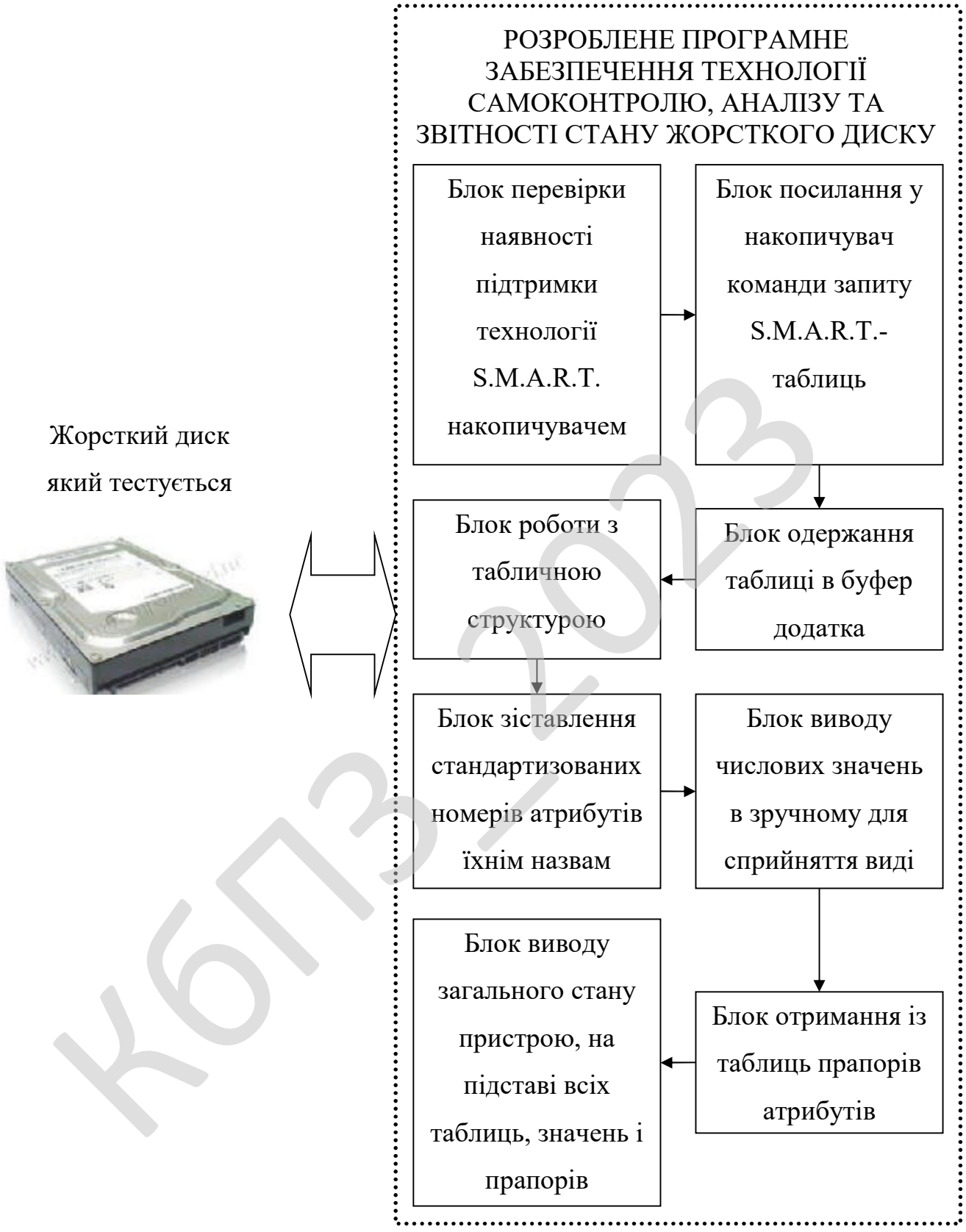


Рисунок 3.1 – Структурна схема системи

Структурна схема складається з наступних блоків:

– Блок перевірки наявності підтримки технології S.M.A.R.T. накопичувачем.

– Блок посилання у накопичувач команди запиту S.M.A.R.T.-таблиць.

– Блок одержання таблиці в буфер додатка.

– Блок роботи з табличною структурою, що витягає з них номера атрибутів і їхні числові значення.

– Блок зіставлення стандартизованих номерів атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Блок виводу числових значень в зручному для сприйняття вигляді.

– Блок отримання із таблиць прапорів атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware накопичувача, наприклад, «життєво важливий» або «лічильник»).

– Блок виводу загального стану пристрою, на підставі всіх таблиць, значень і прапорів.

3.3 Розробка функціональної схеми

Методи тестування

Існує два способи запуску тестів S.M.A.R.T.: автономний (off-line) або монопольний (captive). Результат тесту завжди зберігається накопичувачем у даних S.M.A.R.T. При автономному запуску накопичувач повідомляє про успішне завершення команди до її фактичного виконання й тільки після цього виконує тест. При цьому, по інтерфейсу прапор зайнятий (BSY) не виставляється й накопичувач у будь-який момент готовий приступитися до виконання чергової інтерфейсної команди, припиняючи роботу тесту. Фактично, тест виконується у фоновому режимі. При запуску тесту в монопольному режимі, по інтерфейсу виставляється прапор ЗАЙНЯТИЙ (BSY) і накопичувач починає безпосереднє виконання тесту в режимі реального часу. Будь-яка інтерфейсна команда під час

виконання цього тесту приведе до його переривання й зупинки, після чого накопичувач приступиться до обробки команди, що надійшла.

Автономне сканування поверхні (off-line read scanning)

Більшість накопичувачів забезпечують підтримку автономного сканування поверхні, що є однієї з функцій підпрограми автономного збору даних про стан накопичувача (off-line data collection). При виконанні цієї функції, накопичувач виконує повне сканування поверхні шляхом читання кожного сектора й заміщенням ненадійних секторів на запасні сектори з резервної області (spare area) для запобігання втрати користувальницьких даних.

Якщо під час виконання сканування накопичувач одержує команду по інтерфейсу, то процес сканування переривається й накопичувач приступає до обробки команди, що надійшла. При цьому гарантується максимальний час реагування на команду, що надійшла, – до 2 секунд.

Журнали помилок (S.M.A.R.T. error log)

У більшості сучасних накопичувачів реалізована функція журналювання, помилок або інших подій, що з'являються в плині роботи накопичувача. В основному, накопичувачі надають інформацію про п’ять останніх помилок. При цьому зберігаються останні 5 команд, що надійшли в накопичувач, що передують виникненню цієї помилки, і інша необхідна інформація. Накопичувач може також підтримувати додаткові журнали. Їхня структура, розмір і призначення встановлюються фірмою-виробником. При відновленні мікропрограми накопичувача, всі журнали накопичувача очищаються, а загальна кількість помилок установлюється в значення 0.

У журналах зберігається час по внутрішніх годинниках накопичувача, тобто або загальний відпрацьований час на даний момент, або час від моменту останнього включення накопичувача.

Extended Comprehensive Error Log

Тип: Розширений комплексний журнал помилок [S.M.A.R.T. Error Logging].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Примітка: підтримується 28/ 48-бітна адресація секторів.

Призначення даного журналу аналогічно журналу Comprehensive Error Log і містить у собі копію його записів, однак цей журнал має іншу структуру, що дозволяє реалізувати підтримку як 28-бітної, так і 48-бітної адресації секторів. Максимальна кількість помилок, що зберігаються – 327,680.

Self-test Log

Тип: Журнал результатів самоконтролю [S.M.A.R.T. self-test].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить інформацію про результати виконання команд внутрішньої самодіагностики накопичувача. Журнал може зберігати до 21 запису. При перевищенні цієї кількості, журнал починає заповнюватися заново, перезаписуючи 1-й запис 22-м, 2-й – 23-м і так далі. У кожному записі журналу зберігається регістр із номером тесту, код статусу виконання тесту, час на момент запуску/переривання тесту, номер поточної контрольної точки (або точки останова) тесту, а також LBA-адресу сектора, на якому відбулося переривання/скасування тесту.

Extended Self-test Log

Тип: Розширений журнал результатів самоконтролю [S.M.A.R.T. self-test].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Примітка: підтримується 28/ 48-бітна адресація секторів.

Призначення даного журналу аналогічно журналу Self-test Log і містить у собі копію його записів, однак цей журнал має іншу структуру, що дозволяє реалізувати підтримку як 28-бітної, так і 48-бітної адресації секторів. Максимальна кількість записів – 1,179,648.

Streaming Performance Log

Тип: Журнал параметрів продуктивності потоків [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Даний журнал містить інформацію про переданий накопичувачу параметрів командами керування режимом Automatic Acoustic Management і Typical Host Interface Sector Time (докладніше – див. ATA/ ATAPI-6 rev 1e). У журналі зберігається набір параметрів, по яких виробляється налаштування накопичувача й переклад у нього в режим, коли всі операції читання/запису можливі тільки спеціальними командами й передача даних відбувається у вигляді безперервного потоку, для якого гарантовані й ураховуються всі часові інтервали (на обробку команди, читання й передачу даних; мінімальні/максимальні затримки, час доступу, позиціонування й т.п.). Докладніше про призначення даного виду журналів можна довідатися з опису технології Audio/Video (AV) Streaming Feature.

Write Stream Error Log

Тип: Журнал помилок потокового запису [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить інформацію про виниклі помилки запису в період роботи накопичувача в потоковому режимі (streaming mode). У цьому журналі зберігається загальна кількість подібних помилок, номер останньої помилки, попереднє і поточне значення регістрів стану й помилки, кількість і LBA-номер сектора, на якому дана помилка була зафіксована. Після читання даного журналу, накопичувач скидає лічильник загальної кількості помилок і очищає журнал. Вміст журналу зберігається тільки під час роботи й очищається в момент наступного включення/вимикання накопичувача або при надходженні сигналу

апаратного скидання (hardware reset). Максимальна кількість помилок, що зберігаються – 31.

Read Stream Error Log

Тип: Журнал помилок потокового читання [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить інформацію про виниклі помилки читання в період роботи накопичувача в потоковому режимі (streaming mode). У цьому журналі зберігається загальна кількість подібних помилок, номер останньої помилки, попереднє і поточне значення регістрів стану й помилки; кількість і LBA-номер сектора, на якому дана помилка була зафіксована. Після читання даного журналу, накопичувач скидає лічильник загальної кількості помилок і очищає журнал. Вміст журналу зберігається тільки під час роботи й очищається в момент наступного включення/вимикання накопичувача або при надходженні сигналу апаратного скидання (hardware reset). Максимальна кількість помилок, що зберігаються – 31.

Delayed LBA Sector Log

Тип: Vendor Specified [General Purpose Logging].

Вид доступу: тільки читання (RO).

Розмір: установлюється виробником (VS).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить LBA-адреси всіх секторів, які були переміщені зі свого нормального фізичного розташування, а також адреси границь недоступної послідовності секторів. У такий спосіб ведеться журнал всіх дефектних або нестабільних секторів. Максимальний розмір журналу встановлюється виробником. Нове фізичне розташування, метод і час доступу до заміщених секторів також установлюється виробником і не документується. Запис у даний журнал може бути додана в будь-який момент часу, за умови активності

(живлення) самого накопичувача. Для процесу відновлення журналу встановлюється найвищий пріоритет і виконання всіх інших команд припиняється. При цьому видалити існуючий запис із журналу не можливо. Вміст журналу зберігається при циклах включення/вимикання накопичувача й при надходженні сигналу апаратного скидання (hardware reset).

ECC Uncorrectable Sector Log

Тип: Журнал непоправних помилок ECC [S.M.A.R.T. Recovering].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить список LBA-адрес секторів, на яких була зафіксована й зігнорована некоректуєма помилка ECC при виконанні операції READ CONTINUOUS (див. AV feature). При цьому, виконання процедури автоматичного перепризначення збійного сектора (ADR – Automatic Defects Reassigment) накопичувачем заблоковано. Журнал може містити до 126 записів. Даний журнал доступний для читання тільки при дозволеній операції READ CONTINUOUS. У противному випадку накопичувач поверне код помилки ERR->ABRT, перерве виконання команди або поверне порожній журнал. Після успішного читання журналу, у самому накопичувачі він буде очищений.

Reassigned Sector Log

[under construction]

Drive Activity Log

[under construction]

Drive Time Buffer Log

[under construction]

Host Vendor Specific Log

Тип: Користувальницькі журнали.

Вид доступу: читання/запис (R/W).

Розмір: максимум 31 журнал по 16 секторів (253,952 байт).

Примітка: зміст і формат журналу – кожне, на розсуд користувача.

Цей вид журналу може бути використаний для зберігання довільних користувальницьких даних. Для запису цього журналу використовується команда WRITE S.M.A.R.T. LOG. Якщо даний журнал жодного разу не був записаний, то при читанні накопичувач поверне порожній журнал, заповнений нулями.

Device Vendor Specific Log

Тип: Технічні журнали виробника.

Вид доступу: не визначений, на розсуд виробника (VS).

Розмір: максимум 31 журнал по 16 секторів (253,952 байт).

Примітка: зміст, формат і розміри журналу – на розсуд виробника.

Цей вид журналу призначений для внутрішнього використання фірмовими утилітами виробника, для зберігання результатів роботи убудованих підпрограм аналізу й діагностики стану накопичувача й т.п. Можливість читання/запису цього виду журналу встановлюється виробником і не документується. Нові накопичувачі Seagate (моделі Ux і Barracuda ATA) підтримують і навіть реально використовують ще три види журналів S.M.A.R.T., однак їхнє призначення й опис поки не відомі.

Вбудовані функції самоконтролю (self-test)

Практично з моменту появи стандарту S.M.A.R.T. II, у більшості накопичувачів з'явилася нова функція – внутрішня діагностика й самоконтроль, для поглибленого контролю стану механіки накопичувача, поверхні дисків і т.п. Для запуску цієї функції, у набір команд S.M.A.R.T. була введена нова команда – S.M.A.R.T. EXECUTE OFF-LINE IMMEDIATE. Результат роботи зберігається або в спеціалізованих атрибутах, або окремим параметром серед інших даних в атрибутах. Якщо накопичувач підтримує журнали S.M.A.R.T., то результат виконання тестів зберігається також у журналі Self-test Log. Після виконання тесту, накопичувач в обов'язковому порядку обновляє показання у всіх атрибутах і інших параметрах. Якщо під час виконання внутрішнього тесту накопичувач

одержить по інтерфейсу нову команду, то виконання тесту переривається й накопичувач приступає до обробки команди, що надійшла.

Log Directory

Тип: Каталог журналів S.M.A.R.T.

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримка мультисекторних журналів.

Даний журнал являє собою свого роду каталог, у якому зазначені адреси всіх підтримуваних журналів S.M.A.R.T. і їхній розмір у секторах. Максимальна кількість журналів – 255.

Summary Error Log

Тип: Сумарний журнал помилок.

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить інформацію про загальну кількість помилок, зафіксованих накопичувачем з моменту першого включення (або відновлення мікропрограми) і докладні записи про останні 5 помилок. Для кожної з 5 зафіксованих помилок зберігаються останні 5 команд, що надійшли в накопичувач. У цьому журналі зберігаються всі помилки UNC, IDNF, помилки сервосистеми, запису/читання й т.д. При цьому, для кожної команди зберігається значення всіх регістрів, час і поточний стан накопичувача на момент подачі самої команди. Помилки, викликані подачею непідтримуваних команд або командами з помилковими параметрами не фіксуються в журналі. Якщо накопичувач підтримує Comprehensive Error Log, то журнал Summary Error Log дублює останні п'ять записів з журналу Comprehensive Error Log.

Comprehensive Error Log

Тип: Комплексний журнал помилок [S.M.A.R.T. Error Logging].

Вид доступу: тільки читання (RO).

Розмір: 1..51 сектор (максимум 26,112 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить докладну інформацію про загальну кількість помилок, зафіксованих накопичувачем з моменту першого включення (або відновлення мікропрограми) і докладні записи про останні помилки. Максимальна кількість помилок, що зберігаються – 255. Для кожної зафіксованої помилки зберігаються останні 5 команд, що надійшли в накопичувач. У цьому журналі зберігаються всі помилки UNC, IDNF, помилки сервосистеми, запису/читання й т.д. При цьому, для кожної команди зберігається значення всіх регістрів, час і поточний стан накопичувача на момент подачі самої команди. Помилки, викликані подачею непідтримуваних команд або командами з помилковими параметрами не фіксуються в журналі.

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Так, як у схемі є використання атрибутів S.M.A.R.T. розглянемо їх більш детально.

Атрибути

Атрибути S.M.A.R.T. – особливі характеристики, які використовуються при аналізі стану й запасу продуктивності накопичувача. Атрибути вибираються виробником накопичувача, ґрунтуючись на здатності цих атрибутів пророкувати погіршення робочих характеристик накопичувача або визначити його дефектність. Кожний виробник має свій характерний набір атрибутів і може вільно вносити зміни в цей набір у відповідності зі своїми власними вимогами й без повідомлення про це фірм-продавців і кінцевих користувачів.

Типи атрибутів

Кожний атрибут може мати деякий набір прапорів, що визначають його функціональні особливості. Нижче приводяться всі шість основних типів і їхні короткі описи:

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується

працездатність накопичувача й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності накопичувача за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов S.M.A.R.T.

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті накопичувача й при виконанні тестів S.M.A.R.T.).

Значення атрибутів

Значення атрибутів (value) використовуються для подання відносної надійності окремого експлуатаційного або еталонного атрибута. Припустиме значення атрибута лежить у діапазоні від 1 до 255. Високе значення атрибута говорить про те, що результат аналізу даної робочої характеристики вказує на низьку ймовірність її погіршення або виходу накопичувача з ладу. Відповідно, низьке значення атрибута говорить про те, що результат аналізу даної робочої характеристики вказує на високу ймовірність її погіршення або виходу накопичувача з ладу.

Граничні значення атрибутів

Кожний атрибут має власне граничне значення (threshold), що використовується для порівняння зі значенням атрибута (value) і вказує на погіршення робочих характеристик або дефектність накопичувача. Числове значення граничного атрибута визначається виробником накопичувача через

конструкційні особливості накопичувача й аналіз результатів випробувань на надійність. Граничне значення кожного атрибута вказує на нижню припустиму границю значення атрибута, аж до якої зберігається позитивний статус надійності. Граничні значення встановлюються в заводських умовах виробником накопичувача й, у більшості випадків, можуть бути змінені тільки після перемикання накопичувача в технологічний (factory mode). Припустиме граничне значення атрибута може перебуває в діапазоні від 1 до 255.

Якщо значення одного або більше атрибутів, що мають тип pre-failure (в HDD Speed відзначаються символом "*"), менше або дорівнює відповідного граничного значення, то це свідчить про майбутнє погіршення робочих характеристик і/або повному виході накопичувача з ладу.

Короткий опис основних атрибутів

Даний перелік атрибутів є найбільш повним з доступних на сьогоднішній момент у інтернеті або інших джерелах. Призначення атрибутів і спосіб інтерпретації їхніх значень виявлені або досвідченим шляхом, або отримані від служб технічної підтримки компаній-виробників накопичувачів.

Короткий опис відомих атрибутів:

– * – (використовується в програмі HDD Speed) – даний покажчик показує, що відповідний атрибут S.M.A.R.T. є критичним для нормального функціонування накопичувача. Погіршення значень таких атрибутів з найбільшою ймовірністю приводить до виходу накопичувача з ладу. У нових материнських платах BIOS мають убудовану функцію контролю стану накопичувача саме по цих атрибутах.

– Raw Read Error Rate – Частота появи помилок при читанні даних з диска. – Даний параметр показує частоту появи помилок при операціях читання з поверхні диска з вини апаратної частини накопичувача.

– Throughput Performance – Середня продуктивність (пропускна здатність) диска. – Зменшення значення value цього атрибута з великою ймовірністю вказує на проблеми в накопичувачі.

– Spin Up Time – Час розкручування шпинделя. – Середній час розкручування шпинделя диска від 0 RPM до робочої швидкості. Можливо, у поле raw value утримується час у міллісекундах/секундах.

– Start/Stop Count – Кількість циклів запуск/останов шпинделя. – Поле raw value зберігає загальна кількість включень/вимикань диска.

– Reallocated Sectors Count – Кількість перепризначених секторів. – Коли жорсткий диск зустрічає помилку читання/запису/верифікації він намагається перемістити дані з нього в спеціальну резервну область (spare area) і, у випадку успіху, позначає сектор як "перепризначений". Також, цей процес називають remapping, а перепризначений сектор – remap. Завдяки цій можливості, на сучасних жорстких дисках дуже рідко видні [при тестуванні поверхні] так звані bad block. Однак, при великій кількості ремапів, на графіку читання з поверхні будуть помітні "провали" – різке падіння швидкості читання (до 10% і більше). Поле raw value містить загальна кількість перепризначених секторів.

– Read Channel Margin – Запас каналу читання. – Призначення цього атрибута не документовано й у сучасних накопичувачах він не використовується.

– Seek Error Rate – Частота появи помилок позиціонування блоку магнітних головок (БМГ). – У випадку збоїти в механічній системі позиціонування, ушкодження сервометок (servo), сильного термічного розширення дисків і т.п. виникають помилки позиціонування. Чим їх більше, тим гірше стан механіки й/або поверхні жорсткого диска.

– Seek Time Performance – Середня продуктивність операцій позиціонування БМГ. – Даний параметр показує середню швидкість позиціонування привода БМГ на зазначений сектор. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Power-On Hours – Кількість відпрацьованих годин у включеному стані. – Поле raw value цього атрибута показує кількість годин (мінут, секунд – залежно від виробника), відпрацьованих жорстким диском. Зниження значення (value) атрибута до критичного рівня (threshold) указує на виробіток диском

ресурсу (MTBF – Mean Time Between Failures). На практиці, навіть падіння цього атрибута до нульового значення не завжди вказує на реальне вичерпування ресурсу й накопичувач може продовжувати нормально функціонувати.

– Spin Retry Count – Кількість повторів спроб старту шпинделя диска. – Даний атрибут фіксує загальну кількість спроб розкручування шпинделя і його виходу на робочу швидкість, за умови, що перша спроба була невдалою. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Recalibration Retries – Кількість повторів спроб рекалібровки накопичувача. – Даний атрибут фіксує загальну кількість спроб скидання стану накопичувача й установки головок на нульову доріжку, за умови, що перша спроба була невдалою. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Device Power Cycle Count – Кількість повних циклів запуску/останова жорсткого диска.

– Soft Read Error Rate – Частота появи "програмних" помилок при читанні даних з диска. – Даний параметр показує частоту появи помилок при операціях читання з поверхні диска з вини програмного забезпечення, а не апаратної частини накопичувача.

– Emergency Re-track

– ECC On-The-Fly Count

– Load/Unload Cycle Count – Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення. Докладніше – опис технології Head Load/Unload Technology.

– Temperature – Температура. – Даний параметр відбиває в поле raw value показання убудованого температурного сенсора в градусах Цельсія.

– Reallocation Event Count – Кількість операцій перепризначення (ремаппінгу). – Поле raw value цього атрибута показує загальну кількість спроб перепризначення збійних секторів у резервну область, початих накопичувачем. При цьому, ураховуються як успішні, так і невдалі операції.

– Current Pending Sector Count – Поточна кількість нестабільних секторів. – Поле raw value цього атрибута показує загальну кількість секторів, які накопичувач у цей момент уважає претендентами на перепризначення в резервну область (remap). Якщо надалі якийсь із цих секторів буде прочитаний успішно, то він виключається зі списку претендентів. Якщо ж читання сектора буде супроводжуватися помилками, то накопичувач спробує відновити дані й перенести їх у резервну область, а сам сектор позначити як перепризначений (remapped). Постійно ненульове значення raw value цього атрибута говорить про низьку якість (окремої зони) поверхні диска.

– Uncorrectable Sector Count – Кількість нескоректованих помилок. – Атрибут показує загальну кількість помилок, що виникли при читанні/запису сектора і які не вдалося скорегувати. Ріст значення в поле raw value цього атрибута вказує на явні дефекти поверхні й/або проблеми в роботі механіки накопичувача.

– UltraDMA CRC Error Count – Загальна кількість помилок CRC у режимі UltraDMA. – Поле raw value містить кількість помилок, що виникли в режимі передачі даних UltraDMA у контрольній сумі (ICRC – Interface CRC). Практика, зібрана статистика й вивчення журналів помилок S.M.A.R.T. показують: у більшості випадків помилки CRC виникають при сильному завищенні частоти PCI (більше номінальних 33.6 MHz), сильно перекрученому кабелі, а також – з вини драйверів ОС, які не дотримують вимог до передачі/прийому даних у режимах UltraDMA.

– Write Error Rate (Multi Zone Error Rate) – Частота появи помилок при записі даних. – Показує загальну кількість помилок, виявлених під час запису сектора. Чим більше значення в поле raw value (і нижче значення value), тим гірше стан поверхні диска й/або механіки привода.

– Disk Shift – Зрушення пакета дисків щодо осі шпинделя. – Актуальне значення атрибута втримується в поле raw value. Одиниці виміру – не відомі. Подробиці в описі технології G-Force Protection. Зрушення пакета дисків можливий у результаті сильного ударного навантаження на накопичувач у результаті його падіння або з інших причин.

– G-Sense Error Rate – Частота появи помилок у результаті ударних навантажень. – Даний атрибут зберігає показання ударочуттєвого сенсора – загальна кількість помилок, що виникли в результаті отриманих накопичувачем зовнішніх ударних навантажень (при падінні, неправильній установці, і т.п.). Докладніше в описі технології G-Force Protection.

– Loaded Hours – Навантаження на привод БМГ, викликана загальним наробітком годин накопичувачем. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load/Unload Retry Count – Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п. Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load Friction – Навантаження на привод БМГ, викликане тертям у механічних частинах накопичувача. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load/Unload Cycle Count – Загальна кількість циклів навантаження на привод БМГ. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load-in Time – Загальний час навантаження на привод БМГ. – Можливо, даний атрибут показує загальний час роботи накопичувача під навантаженням, за умови, що головки перебувають у робочому стані (поза парковочною зоною).

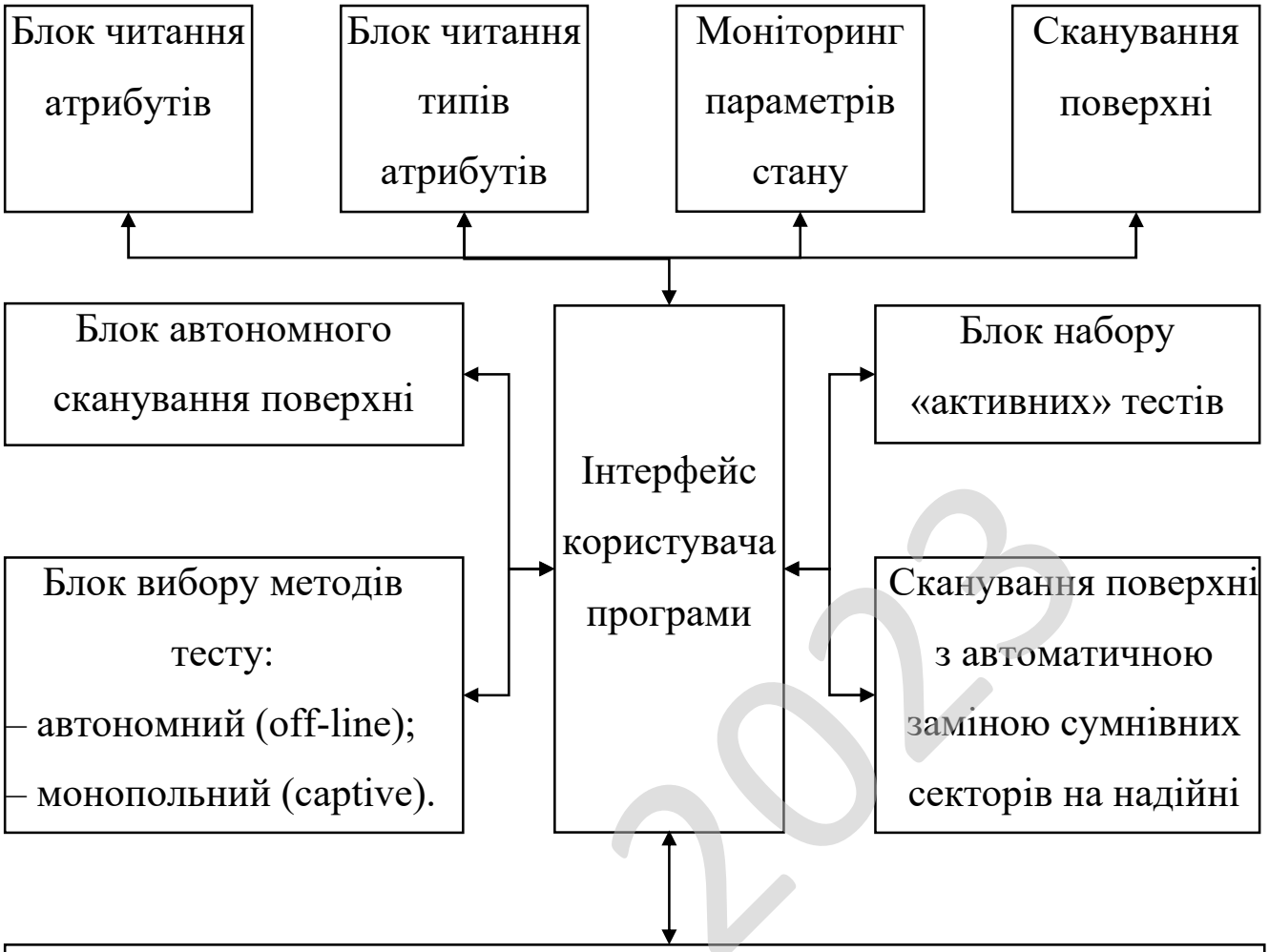
– Torque Amplification Count – Кількість зусиль обертаючого моменту привода.

– Power-Off Retract Count – Кількість зафіксованих повторів включення/вимикання живлення накопичувача.

– GMR Head Amplitude – Амплітуда тремтіння головок (GMR-head) у робочому стані.

– Head Flying Hours

– Read Error Retry Rate



Блок читання журналу помилок:

- Сумарний журнал помилок.
- Журнал непоправних помилок.
- Розширений журнал результатів самоконтролю.
- Журнал помилок потокового запису.
- Розширений комплексний журнал помилок.
- Технічні журнали виробника.
- Каталог журналів S.M.A.R.T.
- Комплексний журнал помилок.
- Журнал результатів самоконтролю.
- Журнал параметрів продуктивності потоків.
- Журнал помилок потокового читання.
- Користувальницькі журнали.

Рисунок 3.2 – Функціональна схема системи

Функціональна схема складається з наступних блоків:

1. Блок читання типів атрибутів:

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті накопичувача й при виконанні тестів S.M.A.R.T.).

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується працездатність накопичувача й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності накопичувача за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов S.M.A.R.T.

2. Блок автономного сканування поверхні.

3. Блок читання атрибутів:

– Кількість відпрацьованих годин у включеному стані.

– Кількість повторів спроб старту шпинделя диска.

– Кількість повторів спроб рекалібровки накопичувача.

– Кількість повних циклів запуску/останова жорсткого диска.

– Частота появи "програмних" помилок при читанні даних з диска.

– Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення.

– Температура.

– Кількість операцій перепризначення (ремаппінгу).

– Поточна кількість нестабільних секторів.

– Кількість нескоректованих помилок.

– Загальна кількість помилок CRC у режимі UltraDMA.

– Частота появи помилок при записі даних.

– Зрушення пакета дисків щодо осі шпинделя.

– Частота появи помилок у результаті ударних навантажень.

– Навантаження на привод БМГ, викликана загальним наробітком годин накопичувачем.

– Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п.

– Навантаження на привод БМГ, викликане тертям у механічних частинах накопичувача.

– Загальна кількість циклів навантаження на привод БМГ.

– Загальний час навантаження на привод БМГ.

– Кількість зусиль обертаючого моменту привода.

– Кількість зафіксованих повторів включення/вимикання живлення накопичувача.

– Амплітуда тремтіння головок (GMR-head) у робочому стані.

– Частота появи помилок при читанні даних з диска.

– Середня продуктивність (пропускна здатність) диска.

– Час розкручування шпинделя.

– Кількість циклів запуск/останов шпинделя.

– Кількість перепризначених секторів.

– Запас каналу читання.

– Частота появи помилок позиціонування БМГ.

– Середня продуктивність операцій позиціонування БМГ.

4. Блок вбудованих функцій самоконтролю.

5. Блок читання журналу помилок:

– Журнал параметрів продуктивності потоків.

– Журнал помилок потокового запису.

– Журнал помилок потокового читання.

– Журнал непоправних помилок.

– Користувальницькі журнали.

– Технічні журнали виробника.

– Каталог журналів S.M.A.R.T.

– Сумарний журнал помилок.

– Комплексний журнал помилок.

– Розширений комплексний журнал помилок.

– Журнал результатів самоконтролю.

– Розширений журнал результатів самоконтролю.

6. Блок вибору методів тесту:

– автономний (off-line);

– монопольний (captive).

7. Блок набору «активних» тестів.

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

3.4 Розробка діаграми процесів

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

– Головне вікно ПЗ.

– Бібліотека самоконтролю, аналізу та звітності стану жорсткого диску.

– Функції отримання звітності стану жорсткого диску.

– Обробник помилок ПЗ.

– Сканування наявності ресурсів для отримання звітності.

– Отримання таблиць прапорів атрибутів жорсткого диску.

– Отримання таблиці.

– Формування звітності стану жорсткого диску.

– Аналіз загального стану жорсткого диску.

– Зіставлення номерів атрибутів їхнім назвам.

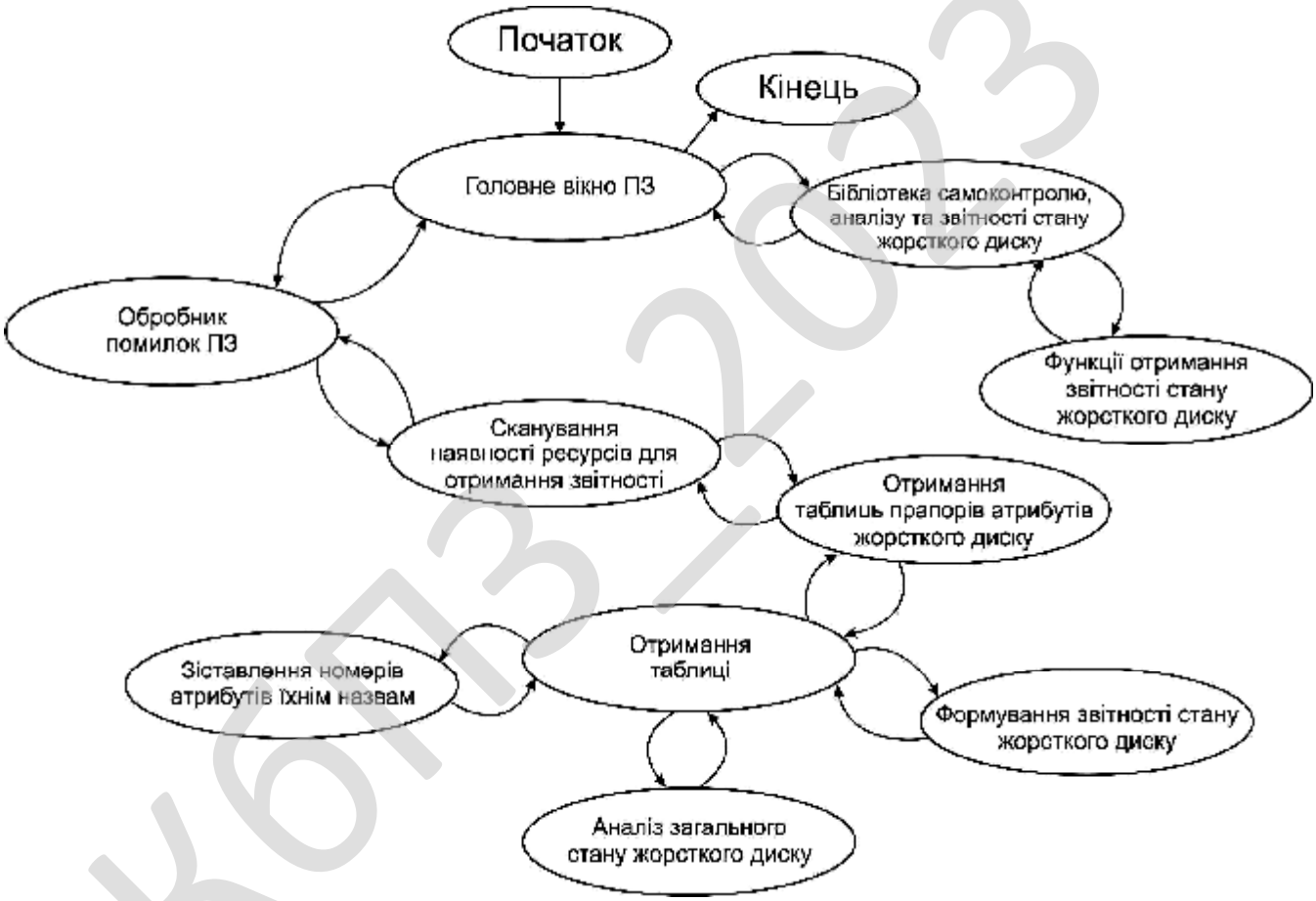


Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. 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.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Крім цього було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які

підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

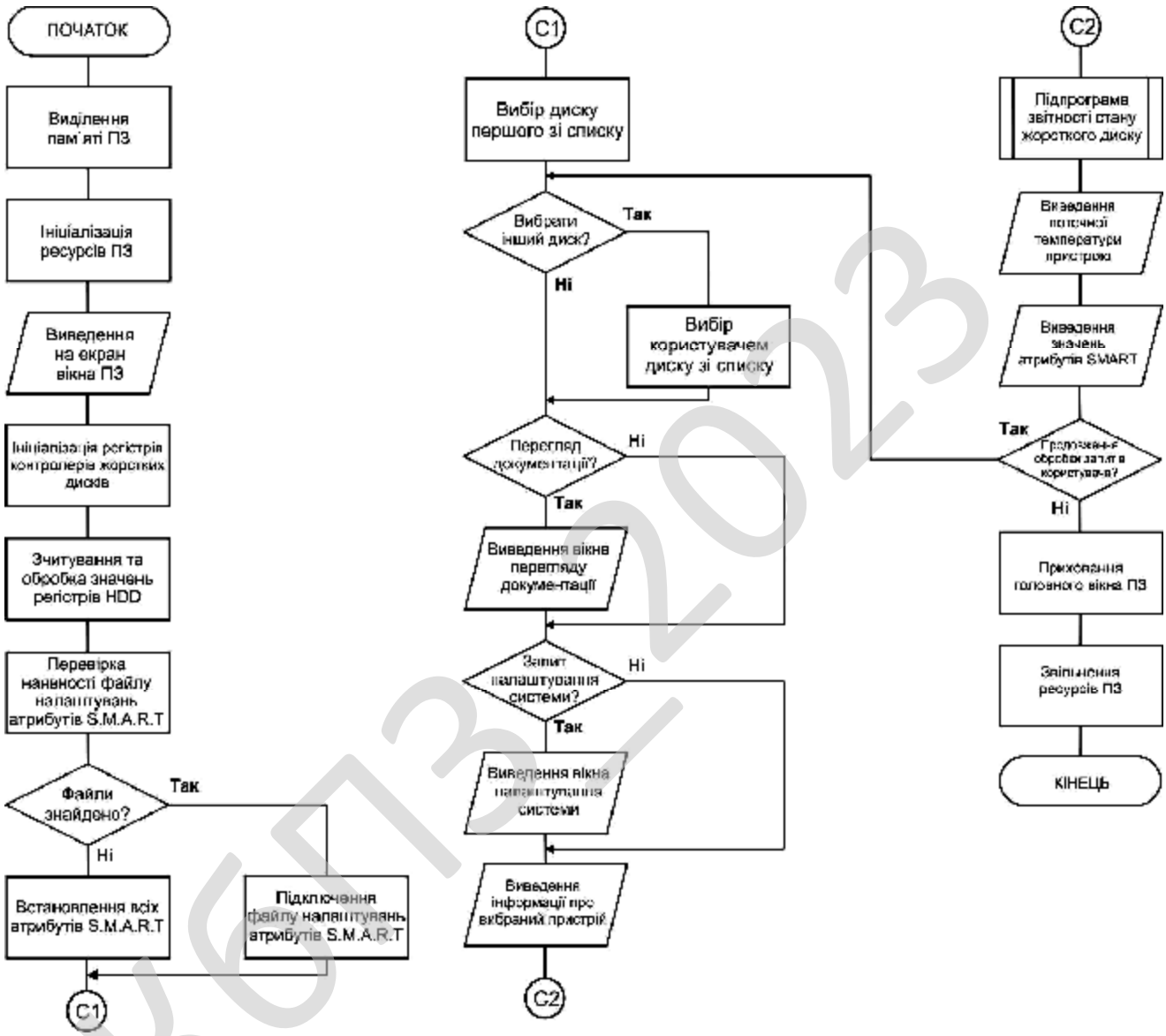


Рисунок 4.1 – Блок схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

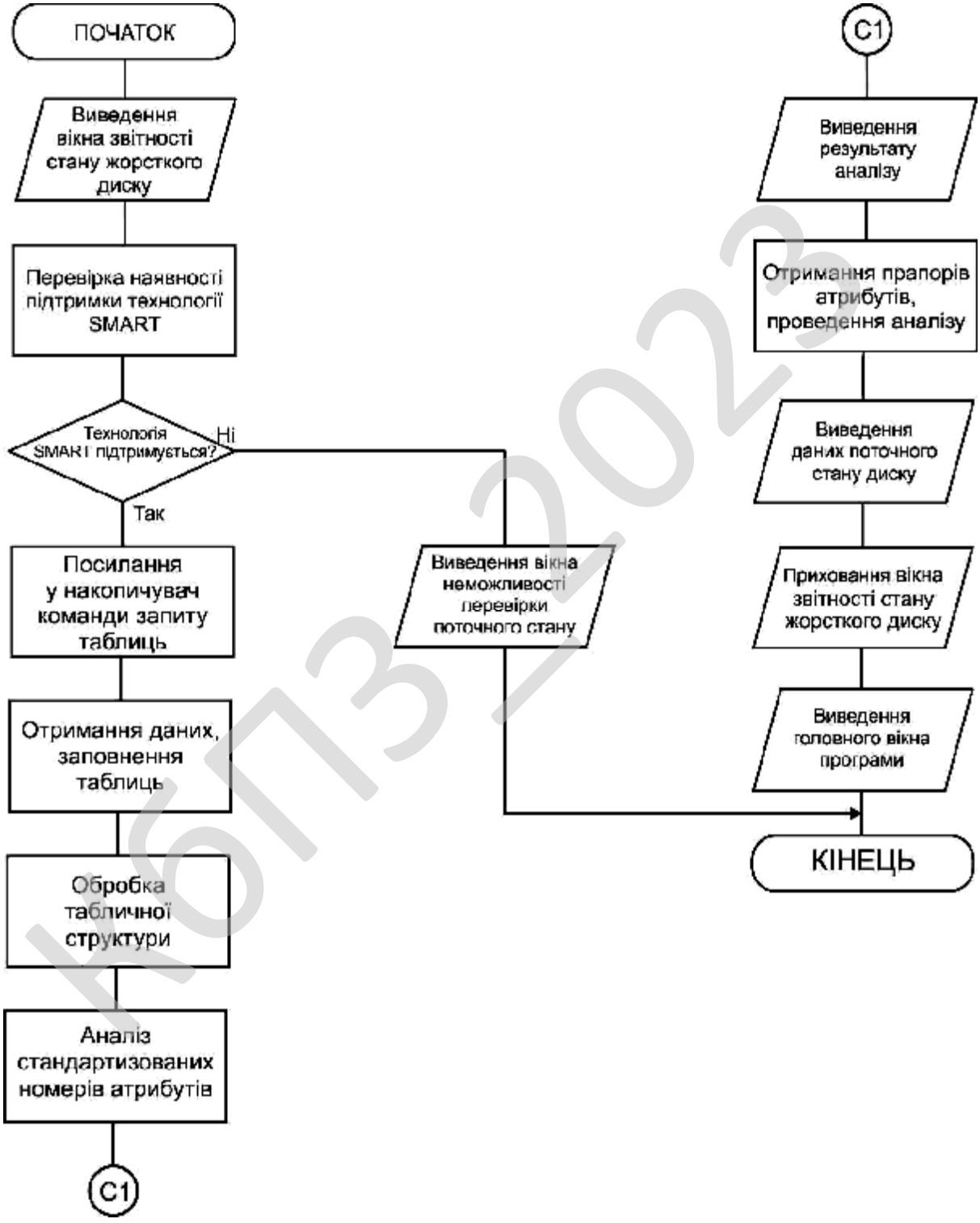


Рисунок 4.2 – Блок схема підпрограм

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо розроблений код ПЗ. Код, який визначає тип диска і намагається активувати SMART:

```
for i: = 0 to 3 do
// Кількість і тип пристроїв визначається параметром bIDEDeviceMap
// Структури TGetVersionOutParams
begin
  // Якщо пристрій з номером "i" – IDE, передаємо йому команди.
  if VersionParams.bIDEDeviceMap shr i and 1 = 1 then
  begin
    // Ігноруємо ATAPI – пристрої.
```

```
      if VersionParams.bIDEDeviceMap shr i and $ 10 = 0 then
      begin
        ZeroMemory (@ scip, sizeof (scip));
// Обнуляємо TSendCmdInParams
        ZeroMemory (@ OutCmd, sizeof (OutCmd));
// Обнуляємо TSendCmdOutParams
        // Намагаємося активувати SMART.
        if DoEnableSMART (hSMARTIOCTL, @ scip, @ OutCmd, i) then
          ShowMessage ('Команда запуску SMART виконана, диск:'
            + Inttostr (i))
        else ShowMessage ('Команда запуску SMART не виконана, диск:'
          + Inttostr (i));
      end;
    end;
end;
```

Розглянемо реалізацію читання атрибутів SMART Як уже згадувалося, щоб провести аналіз стану привода, необхідно знати поточні та порогові значення атрибутів. Створимо два типи для читання цих значень.

Перший – для читання значень атрибутів:

```
type
  TDriveAttribute = packed record
    bAttrID: BYTE; // Ідентифікатор атрибуту
    wStatusFlags: WORD; // Прапори стану
    bAttrValue: BYTE; // Поточне нормалізоване значення
    bWorstValue: BYTE; // Найгірше значення
    bRawValue: array [0 .. 5] of BYTE;
// Поточне ненормалізованого значення
    bReserved: BYTE; // Зарезервовано
end;
  DRIVEATTRIBUTE = TDriveAttribute;
  PDriveAttribute = ^ TDriveAttribute;
```

Параметр wStatusFlags може приймати наступні значення або їх комбінації:

```
const
  // Життєво важливий
  PRE_FAILURE_WARRANTY = $ 01;
  // Колекція реального часу
  ON_LINE_COLLECTION = $ 02;
  // Атрибут, що відображає продуктивність диска
```

```
  PERFORMANCE_ATTRIBUTE = $ 04;
  // Атрибут, що відображає частоту появи помилок
  ERROR_RATE_ATTRIBUTE = $ 08;
  // Лічильник подій
  EVENT_COUNT_ATTRIBUTE = $ 10;
  // Самозберігається атрибут
  SELF_PRESERVING_ATTRIBUTE = $ 20;
```

Другий тип призначений для читання порогових значень:

```
type
  TAttrThreshold = packed record
    bAttrID: BYTE; // Ідентифікатор атрибуту
    bWarrantyThreshold: BYTE; // Граничне значення
    bReserved: array [0 .. 9] of BYTE; // Зарезервовано
end;
  ATTRTHRESHOLD = TAttrThreshold;
  PAttrThreshold = ^ TAttrThreshold;
```

Функція читання значень атрибутів виглядає наступним чином:

```
function DoReadAttributesCmd (hSMARTIOCTL: THandle;
                              pSCIP: PSENDCMDINPARAMS;
                              pSCOP: PSENDCMDOUTPARAMS;
                              bDriveNum: BYTE): BOOL;
var
  cbBytesReturned: DWORD;
begin
  // Константа = 512
  pSCIP.cBufferSize: = READ_ATTRIBUTE_BUFFER_SIZE;
  // Константа = $ D0
  pSCIP.irDriveRegs.bFeaturesReg: = SMART_READ_ATTRIBUTE_VALUES;
  pSCIP.irDriveRegs.bSectorCountReg: = 1;
  pSCIP.irDriveRegs.bSectorNumberReg: = 1;
  pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
  pSCIP.irDriveRegs.bCylHighReg: = SMART_CYL_HI;
  // Обчислюємо номер накопичувача.
  pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
  pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_SMART_FUNCTION;
  pSCIP.bDriveNumber: = bDriveNum;
  result: = DeviceIoControl (hSMARTIOCTL, DFP_RECEIVE_DRIVE_DATA,
    pSCIP, sizeof (SENDCMDINPARAMS) - 1, pSCOP, sizeof (SENDCMDOUTPARAMS)
    + READ_ATTRIBUTE_BUFFER_SIZE - 1, cbBytesReturned, nil);
end;
```

Функція для читання порогових значень DoReadThresholdsCmd виглядає аналогічно, параметр bFeaturesReg = $ D1.

Дескриптор для роботи з функціями SMART.

```
DeviceIoControl.function OpenSMART (DrvNum: Byte): THandle;
  var
    hSMARTIOCTL: THandle;
begin
  // Якщо у нас Windows сімейства NT
  if OSVersionInfo.dwPlatformId = VER_PLATFORM_WIN32_NT then
      hSMARTIOCTL: = CreateFile (PChar ('\ \. \ PhysicalDrive' + inttostr
(DrvNum)),
        GENERIC_READ or GENERIC_WRITE, FILE_SHARE_READ or FILE_SHARE_WRITE,
        nil, OPEN_EXISTING, 0, 0);
  else // Якщо у нас Windows сімейства 9x
  begin
    hSMARTIOCTL: = CreateFile ('\ \. \ SMARTVSD', 0, 0, nil, CREATE_NEW, 0,
0);
    if hSMARTIOCTL = INVALID_HANDLE_VALUE then
      ShowMessage ('Неможливо відкрити SMARTVSD, код помилки:'
        + Inttostr (GetLastError) + '-' + SysErrorMessage (GetLastError))
  end;
  result: = hSMARTIOCTL;
end;
```

В якості параметра даної функції виступає номер фізичного диска. Максимальна кількість IDE-дисків – 4. Цей параметр ігнорується, якщо програма запущена в операційній системі Win9x.

В операційних системах Windows 95 OSR, 98, 98SE, Me за роботу зі SMART відповідає драйвер віртуального пристрою SMARTVSD.VXD, який знаходиться в папці ... \ WINDOWS \ SYSTEM \ IOSUBSYS.

В операційних системах лінійки NT робота з пристроями (як фізичними, так і віртуальними) побудована іншим чином, тому при роботі з SMART в цих операційних системах необхідно відкривати дескриптор доступу до фізичного диску.

Змінна OSVersionInfo є глобальною, і її тип визначений як TOSVersionInfo. Вона заповнюється до виклику функції OpenSMART.

Отримавши дескриптор SMART, необхідно визначити версію SMART IOCTL. За це відповідає функція:

```
function GetVersionSMART (hSMARTIOCTL: THandle): TGetVersionOutParams;
var
  VersionParams: TGetVersionOutParams;
  cbBytesReturned: DWORD;
begin
  ZeroMemory (@ VersionParams, sizeof (TGetVersionOutParams));
    if not DeviceIoControl (hSMARTIOCTL, DFP_GET_VERSION, nil, 0,
      @ VersionParams, sizeof (VersionParams), cbBytesReturned, nil)
    then
      ShowMessage (SysErrorMessage (GetLastError));
  Result: = VersionParams;
end;
```

В якості параметра передається дескриптор SMART Функція повертає структуру типу:

```
 TGetVersionOutParams.type
  TGetVersionOutParams = packed
record
    bVersion: BYTE; // Бінарна версія драйвера.
    bRevision: BYTE; // Бінарна підверсія драйвера.
    bReserved: BYTE; // Не використовується.
    bIDEDeviceMap: BYTE; // Бітовий масив IDE - пристроїв.
    fCapabilities: DWORD; // Бітова маска можливостей драйвера.
    // Зарезервовано для майбутнього використання.
    dwReserved: array [0 .. 3] of DWORD;
end;
GETVERSIONOUTPARAMS = TGetVersionOutParams;
PGetVersionOutParams = ^ TGetVersionOutParams;
```

Константа DFP_GET_VERSION = $ 00074080 є командою отримання версії SMART IOCTL. Наступний крок – знайти IDE-диски і спробувати активувати на них SMART Для цього треба знати такі структури.

```
 type
  TIDERegs = packed record
    // Використовується для визначення "підкоманди" SMART
    bFeaturesReg: BYTE;
    // Регістр кількості секторів IDE
    bSectorCountReg: BYTE;
    // Регістр номера сектора IDE
```

```
    bSectorNumberReg: BYTE;
    // Молодший розряд номера циліндра IDE
    bCylLowReg: BYTE;
    // Старший розряд номера циліндра IDE
    bCylHighReg: BYTE;
    // Регістр диска / головки IDE
    bDriveHeadReg: BYTE;
    // Фактична команда IDE
    bCommandReg: BYTE;
    // Зарезервовано для майбутнього використання. Повинне бути 0.
    bReserved: BYTE;
end;
  IDEREGS = TIDERegs;
  PIDERegs = ^ TIDERegs;
```

Тип TIDERegs описує регістри IDE-диска. Допустимі значення параметра.

```
 bCommandReg: const
 // Повертає ID сектора для ATAPI.
 IDE_ATAPI_ID = $ A1;
 // Повертає ID сектора для ATA.
 IDE_ID_FUNCTION = $ EC;
 // Виконує команду SMART. Вимагає правильних значень для параметрів
 // BFeaturesReg, bCylLowReg, bCylHighReg.
 IDE_EXECUTE_SMART_FUNCTION = $ B0;
```

Параметри bCylLowReg і bCylHighReg повинні бути обов'язково дорівнюють $ 4F (SMART_CYL_LOW) і $ C2 (SMART_CYL_HI) відповідно.

```
type
  TSendCmdInParams = packed record
    // Розмір буфера в байтах.
    cBufferSize: DWORD;
    // Структура зі значеннями регістрів диска.
    irDriveRegs: TIDERegs;
    // Фізичний номер диска для виконання команд.
    bDriveNumber: BYTE;
    // Зарезервовано для майбутнього розширення.
    bReserved: array [0 .. 2] of Byte;
    // Зарезервовано для майбутнього використання.
    dwReserved: array [0 .. 3] of DWORD;
    // Вхідний буфер.
    bBuffer: array [0 .. 0] of Byte;
end;
  SENDCMDINPARAMS = TSendCmdInParams;
```

```
  PSendCmdInParams = ^ TSendCmdInParams;
```

Тип TSendCmdInParams містить вхідні параметри для функції, яка посилає команди диску.

```
type
  TDriverStatus = packed record
    // Код помилки драйвера.
    bDriverError: Byte;
    // Зміст регістра помилки. Правильно, тільки коли
    // BDriverError = SMART_IDE_ERROR (1).
    bIDEStatus: Byte;
    // Зарезервовано для майбутнього розширення.
    bReserved: array [0 .. 1] of Byte;
    // Зарезервовано для майбутнього розширення.
    dwReserved: array [0 .. 1] of DWORD;
end;
  DRIVERSTATUS = TDriverStatus;
  PDriverStatus = ^ TDriverStatus;
```

Тип TDriverStatus призначений для відстеження помилок драйвера. Якщо параметр bDriverError містить значення, відмінне від нуля, значить, відбулася помилка.

```
type
  TSendCmdOutParams = packed record
    // Розмір bBuffer в байтах
    cBufferSize: DWORD;
    // Структура стану драйвера.
    DriverStatus: TDriverStatus;
    // Буфер довільної довжини для збереження даних, прочитаних з диска.
    bBuffer: array [0 .. 0] of BYTE;
end;
  SENDCMDOUTPARAMS = TSendCmdOutParams;
  PSendCmdOutParams = ^ TSendCmdOutParams;
```

Тип TSendCmdOutParams призначений для деяких команд, які повертають через нього дані. Тепер власне функція активації SMART:

```
function DoEnableSMART (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
  pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
  lpcbBytesReturned: DWORD;
begin
  pSCIP.cBufferSize: = 0;
```

```
  // Активувати S.M.A.R.T.
  pSCIP.irDriveRegs.bFeaturesReg: = SMART_ENABLE_SMART_OPERATIONS ($ D8);
  pSCIP.irDriveRegs.bSectorCountReg: = 1;
  pSCIP.irDriveRegs.bSectorNumberReg: = 1;
  pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
  pSCIP.irDriveRegs.bCylHighReg: = SMART_CYL_HI;
  // Обчислюємо номер накопичувача.
  pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
  // Виконати функцію S.M.A.R.T.
  pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_SMART_FUNCTION;
  pSCIP.bDriveNumber: = bDriveNum;
  result: = DeviceIoControl (hSMARTIOCTL, DFP_SEND_DRIVE_COMMAND, pSCIP,
    sizeof (SENDCMDINPARAMS) - 1, pSCOP, sizeof (SENDCMDOUTPARAMS) - 1,
    lpcbBytesReturned, nil);
end;
```

Слід сказати пару слів про переданих параметрах. В якості параметрів pSCIP і pSCOP передаються обнуління структури TSendCmdInParams і TSendCmdOutParams, відповідно.

Параметр bDriveNum – це номер диска в межах від 0 до 3. Після заповнення необхідних параметрів структури PSENDCMDOUTPARAMS виконуємо функцію DeviceIoControl з керуючим кодом DFP_SEND_DRIVE_COMMAND ($ 0007C084).

Якщо функція виконана успішно, результат який повертається – TRUE. Приклад читання поточних і порогових значень атрибутів диска «i» наведено нижче:

```
var
  // Два буфера для отримання даних
  AttrOutCmd, ThreshOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) - 1)
  +(READ_ATTRIBUTE_BUFFER_SIZE - 1)] of BYTE;
  bSuccess: bool;
begin
  ZeroMemory (@ AttrOutCmd, sizeof (AttrOutCmd));
  ZeroMemory (@ ThreshOutCmd, sizeof (ThreshOutCmd));
  bSuccess: = DoReadAttributesCmd (hSMARTIOCTL, @ scip,
    PSENDCMDOUTPARAMS (@ AttrOutCmd), i);
  if bSuccess = false then ShowMessage (
    'Помилка при виконанні команди читання атрибутів SMART на диску: '
    + Inttostr (i))
```

```
  // Команда читання атрибутів виконана успішно.
  // Намагаємося прочитати порогові значення атрибутів.
  else if not DoReadThresholdsCmd (hSMARTIOCTL, @ scip,
    PSENDCMDOUTPARAMS (@ ThreshOutCmd), i)
  then
    ShowMessage ('Помилка при виконанні команди читання порогових значень'
      + 'Атрибутів S.M.A.R.T. на диску: '+ inttostr (i));
  if bSuccess <> false then
    // Виводимо інформацію про атрибути та їх порогових значеннях
    DoPrintData (@ PSENDCMDOUTPARAMS (@ AttrOutCmd). BBuffer,
      @ PSENDCMDOUTPARAMS (@ ThreshOutCmd). BBuffer);
end;

procedure TForm1.DoPrintData (pAttrBuffer: PCHAR; pThrsBuffer: PCHAR);
var
  i: integer;
  pDA: PDRIVEATTRIBUTE;
  pAT: PATTRTHRESHOLD;
begin
  Label8.Caption: = 'Версія структури атрибутів:'
    + Inttostr (WORD (pAttrBuffer [0]));
  Label9.Caption: = 'Версія структури порогових значень атрибутів:'
    + Inttostr (WORD (pThrsBuffer [0]));
  pDA: = PDRIVEATTRIBUTE (@ pAttrBuffer [2]);
  pAT: = PATTRTHRESHOLD (@ pThrsBuffer [2]);
  for I: = 0 to 29 do
  begin
    // Виводимо інформацію:
    // Ідентифікатор атрибуту
StringGrid1.Rows [i + 1]. Strings [0]: = inttostr (pDA.bAttrID);
    // Його назва
StringGrid1.Rows [i + 1]. Strings [1]: = pAttrNames [pDA.bAttrID];
    // Поточне значення
StringGrid1.Rows [i + 1]. Strings [2]: = inttostr (pDA.bAttrValue);
    // Граничне значення
StringGrid1.Rows [i + 1]. Strings [3]: = inttostr (pAT.bWarrantyThreshold);
    // Найгірше значення
StringGrid1.Rows [i + 1]. Strings [4]: = inttostr (pDA.bWorstValue);
    inc (pDA);
    inc (pAT);
  end;
end;
```

У даній процедурі змінна pAttrNames – це масив строкових значень, що містять назву атрибуту у відповідності зі своїм порядковим номером у масиві.

Атрибут під назвою Temperature (Температура). Його ідентифікатор – 194 або 231. Як ясно випливає з його назви, він показує температуру вінчестера, яка вимірюється в градусах Цельсія. Щоб її обчислити, необхідно скористатися нижченаведеними кодом:

```
if (pDA.bAttrID = 194) or (pDA.bAttrID = 231) then
  Label7.Caption: = 'Температура:'
    + Inttostr ((84 - (pDA.bAttrValue - 1) div 3)) + # 176 + 'C'
```

Більшість сучасних жорстких дисків підтримують технологію SMART – Self-Monitoring, Analysis and Reporting Technology (Технологія самодіагностики, аналізу і звіту), завдяки якій можливо передбачити появу збоїв в роботі жорсткого диска, і дозволити користувачеві своєчасно зробити резервну копію диска або ж повністю його замінити.

Існує безліч програм, що дають можливість стежити за станом вінчестера за допомогою технології SMART, проте більшість із них – платні.

Я у магістерському проекті розробив ПЗ використовуючи вбудовані засоби ОС Windows. При розробці я використовував документ «Small Form Factor Committee. Specification for Self-Monitoring, Analysis and Reporting Technology», затверджений такими компаніями, як Compaq Computer Corporation, Hitachi Ltd., IBM Storage Products Company, Maxtor Corporation, Quantum Corporation, Seagate Technology, Toshiba Corporation і Western Digital Corporation. Більшість положень цього документа актуально і по сей день.

Слід також зазначити, що на сьогоднішній день стандарт на технологію SMART не затверджено. Однак у стандарті ATA, починаючи з версії 3, описаний обов'язковий мінімум для технології SMART, і якщо ваш жорсткий диск відповідає ATA (3-8), то він буде підтримувати дану технологію у відповідності з цим стандартом.

Аналіз стану диска проводиться за допомогою вивчення атрибутів SMART Максимальна кількість атрибутів на одному диску залежить від виробника і не перевищує 30. Атрибути можуть приймати значення в діапазоні від 1 до 253.

Для кожного атрибута існує граничне значення, ґрунтуючись на якому можна судити про близькість моменту виходу приводу з ладу.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Serpent – симетричний блочний алгоритм шифрування, розроблений Россом Андерсоном, Елі Біхамом та Ларсом Кнудсеном. Алгоритм був одним з фіналістів 2-го етапу конкурсу AES. Як і інші алгоритми, які брали участь у конкурсі AES, Serpent має розмір блоку 128 біт і можливі довжини ключа 128, 192 або 256 біт. Алгоритм являє собою 32-раундовий шифр на основі SP-мережі, і працює з блоком з чотирьох 32-бітових слів. Serpent був розроблений так, що всі операції можуть бути виконані паралельно, використовуючи 32-а 1-бітних «потоків».

При розробці Serpent використовувався консервативніший підхід до безпеки, ніж у інших фіналістів AES, проектувальники шифру вважали, що 16 раундів достатньо, щоб протистояти відомим видам криптоаналізу, але збільшили число раундів до 32, щоб алгоритм міг краще протистояти ще не відомим методам криптоаналізу.

Шифр Serpent не запатентований і є громадським надбанням.

Алгоритм створювався під гаслом «криптографічний алгоритм 21 століття» для участі в конкурсі AES. При створенні нового алгоритму Serpent його автори дотримувалися консервативних поглядів на проектування, що підтверджується первісним рішенням про використання таблиць підстановки з відомого багато років раніше алгоритму шифрування DES, який протягом довгого часу вивчався провідними фахівцями в області криптографії та захисту

інформації і чиї властивості і особливості були добре відомі науковому світу. Одночасно з цим до нового алгоритму міг бути застосований вичерпний аналіз, вже розроблений для DES. Не використовувалися нові, неперевірені і невипробувані технології при створенні шифру, який у разі прийняття був би використаний для захисту величезних масивів фінансових транзакцій та урядової інформації. Основною вимогою до учасників конкурсу AES було те, що алгоритм-претендент повинен бути швидшим, ніж 3DES, і надавати як мінімум такий же рівень безпеки: він повинен мати блок даних довжиною 128 біт і ключ завдовжки 256 біт. 16-раундовий Serpent був би таким же надійним, як 3DES, але в два рази швидшим. Однак, автори вирішили, що для більшої надійності варто збільшити кількість раундів до 32. Це зробило шифр таким же швидким, як DES, і набагато надійнішим, ніж 3DES.

Структура алгоритму

Алгоритм Serpent є SP-мережею, у котрій весь блок даних довжиною 128 біт на кожному раунді розбивається на 4 слова довжиною 32 біти. Всі значення, що використовуються при шифруванні є бітовими потоками. Бітові індекси змінюють значення від 0 до 31 для 32-бітових слів, від 0 до 127 – для 128-бітових блоків та від 0 до 255 для 256-бітових ключів тощо. Для внутрішніх обчислень всі біти величин представлені в прямому порядку (little-endian).

Serpent шифрує відкритий текст P довжиною 128 біт в шифротекст C довжиною таких же 128 біт за 32 раунд за допомогою 33 подключів {\displaystyle K_{0},...,K_{32}\,}{\displaystyle K_{0},...,K_{32}\,} довжиною 128 біт. Довжина використовуваного ключа може приймати різні значення, але для конкретики зафіксуємо їх довжину в 128, 192 або 256 біт. Короткі ключі довжиною менше 256 біт доповнюються до повної довжини в 256 біт.

Шифрування складається з наступних основних кроків:

– Початкова перестановка.

– 32 раунд, кожен з яких складається з операції змішування з 128-бітовим ключем (побітове логічне виключаюче «або»), таблична заміна (S-box) і лінійне

перетворення. В останньому раунді лінійне перетворення замінюється додатковим накладанням ключа.

– Кінцева перестановка.

Початкова і кінцева перестановки не мають будь-якої криптографічного значущості. Вони використовуються для спрощення оптимізованої реалізації алгоритму і підвищення обчислювальної ефективності.

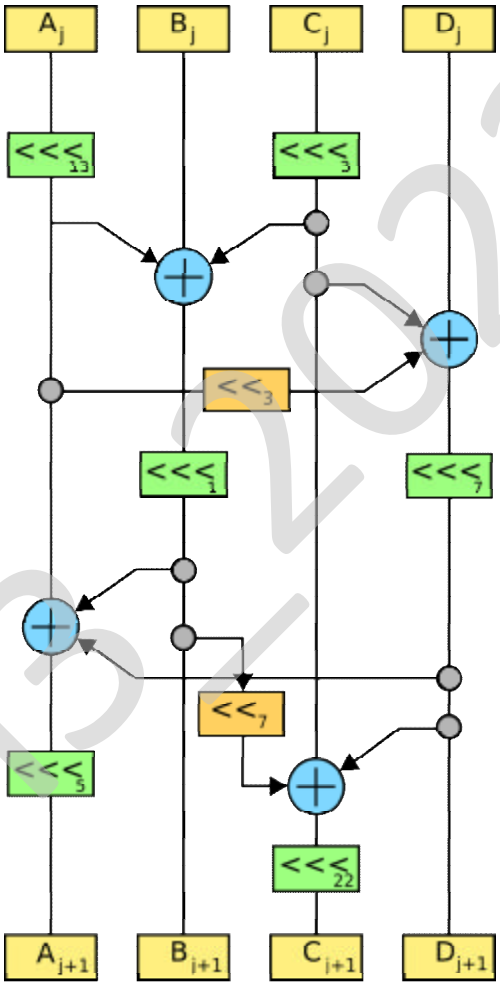


Рисунок 4.3 – Структура алгоритму Serpent

Розширення ключа

Як і інші алгоритми, що брали участь в конкурсі AES, Serpent має можливі довжини ключа 128, 192 або 256 біт. «Неповний» ключ довжиною менше 256 біт

доповнюється за наступним правилом: додається одиничний біт справа, за ним слід стільки нульових бітів, щоб довжина ключа стала дорівнює 256 бітам.

Початкова перестановка IP

Дана перестановка IP задається таблицею, де вказується позиція, на яку перейде відповідний біт (наприклад, біт 1 перейде на 32 позицію):

S-бокси (таблиці замін)

В алгоритмі Serpent таблиці замін є 4-бітовими перестановками з властивостями стійкості до диференціального криптоаналізу, до лінійного криптоаналізу і такою властивістю, що порядок вихідних біт, як функції вхідних повинен бути максимальний, тобто бути рівним 3.

Таблиця підстановки генерується з відомих і добре вивчених таблиць для алгоритму DES в ітераційному процесі, поки не будуть отримані бажані диференціальні й лінійні властивості. Таким чином, створюється 8 таблиць підстановки.

Лінійне перетворення LT

Лінійне перетворення LTзадається таблицею, де біти перераховані від 0 до 127 (наприклад, вихідний 2 біт утворений 2, 9, 15, 30, 76, 84, 126 битами, складеними за модулем 2) . В кожному рядку описується 4 вихідних біти, які разом складають вхідні дані на одну таблицю замін в наступному раунді. Варто зазначити, що даний набір являє собою таблицю IP(LT(FP(x))), де LT і є те лінійне перетворення.

Таблиця зворотного лінійного перетворення, яке використовується при розшифровці ILT.

Кінцева перестановка FP

Дана перестановка є зворотною до початкової, тобто $FP=IP^{-1}$ і задається наступною таблицею.

Ефективна реалізація алгоритму

Бажання авторів зробити алгоритм саме таким, яким він є стає зрозумілим при розгляді його ефективної низькорівневої реалізації.

Serpent був створений таким чином, щоб всі операції в процесі шифрування і розшифрування одного блоку могли бути виконані паралельно в 32 потоках. До того ж низькорівневий опис алгоритму набагато простішій, ніж стандартний опис. Ніяких початкових і кінцевих перестановок не потрібно.

Шифрування складається з 32 раундів. Відкритий текст є першими проміжними даними $B_0 = P$. Потім виконується 32 раунди, кожен і-й раунд складається з:

– Змішування з ключем. Проводиться побітове виключаюче «або» проміжних даних B_i з ключем довжиною 128 біт.

– Застосування таблиць підстановки. Вхідні дані довжиною 128 біт поділяються на 4 слова по 32 біта. Таблиця підстановки, реалізована послідовністю логічних операцій (як якщо це було б реалізовано апаратно), застосовується до цих 4 словам. В результаті виходить 4 вихідних слова. Таким чином, центральний процесор виконує підстановку по 32 копій таблиці одночасно.

– Лінійне перетворення. 32-бітові слова перетворюються заданим порядком.

Першою причиною вибору такого лінійного перетворення є максимізація лавинного ефекту. Такі таблиці підстановки мають властивість, що зміна кожного вхідного біта призведе до зміни 2 вихідних бітів. Таким чином, кожен вхідний біт відкритого тексту вже через 3 раунди впливає на всі вихідні біти. Аналогічно кожен біт ключа впливає на результат шифрування.

Друга причина полягає в простоті перетворення. Воно може бути реалізоване на будь-якому сучасному процесорі з мінімальними витратами.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

1. Меню. З самого верху йде меню розробленого ПЗ. Воно повністю повторює функціонал всієї програми.

2. Обрання HDD.

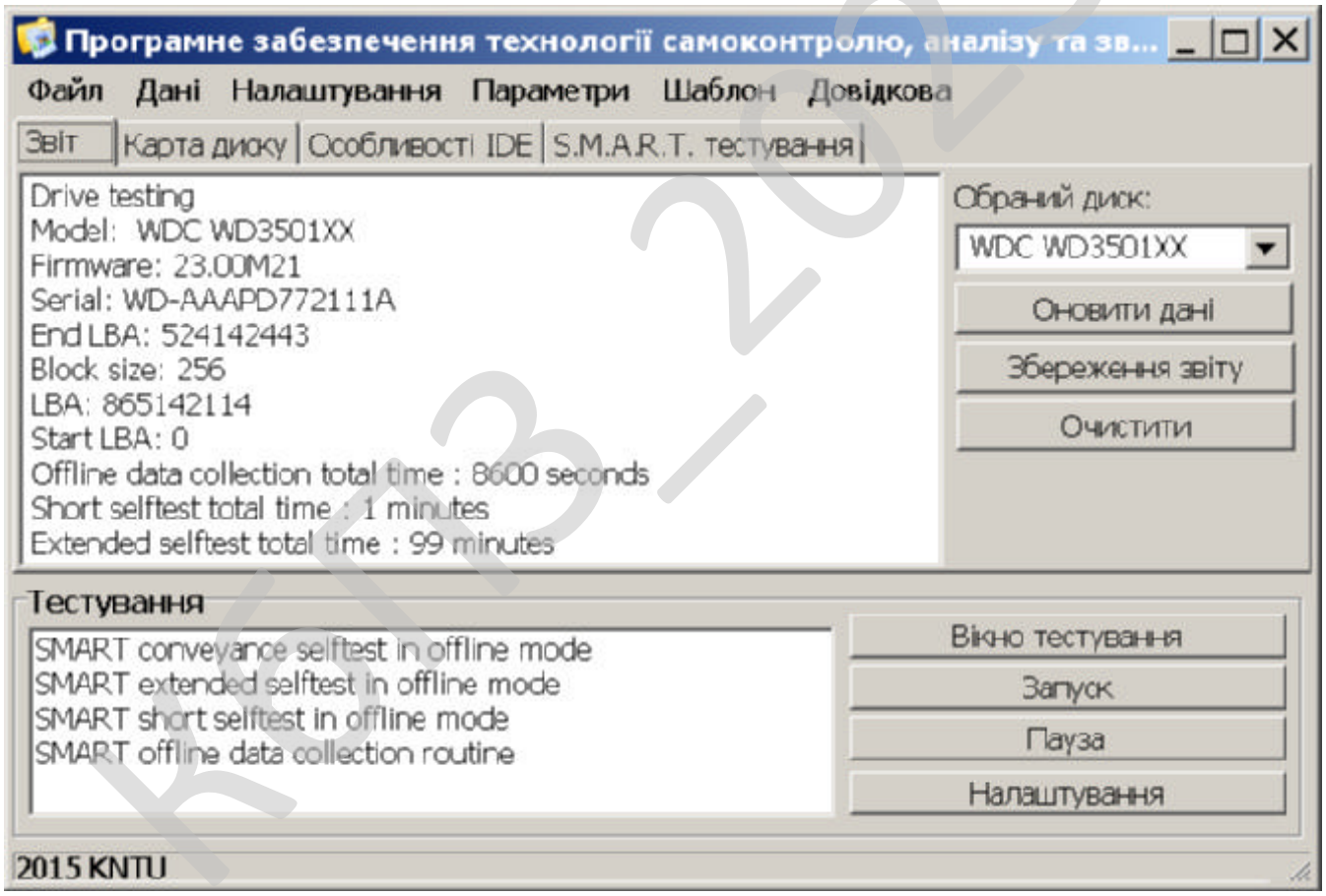


Рисунок 5.1 – Головне вікно ПЗ

3. Тестування. Нижче обрання диску знаходиться розділ «Тестування», який відповідає за перевірку диска. Тест обирається у меню. З правого боку

знаходяться кнопки які відповідають за функціонал тестування:

– Вікно тестування.

– Запуск.

– Пауза.

– Налаштування.

4. Поточний стан тестування. Нижче тестування знаходиться графічний індикатор. Він відображає поточний стан тестування (якщо таке відбувається).

5. Звіт, карта диску, особливості IDE, SMART тестування. Нижче поточного стану знаходяться вкладки які відображають поточний стан диску та його налаштування.

Система призначена для реалізації технології самоконтролю, аналізу та звітності стану жорсткого диску. S.M.A.R.T робить спостереження за основними характеристиками накопичувача, кожна з яких одержує оцінку. Характеристики можна розбити на дві групи:

– параметри, що відбивають процес природного старіння жорсткого диска (число обертів шпінделя, число переміщень головок, кількість циклів включення-вимикання);

– поточні параметри накопичувача (висота головок над поверхнею диска, число перепризначених секторів, час пошуку доріжки й кількість помилок пошуку).

Розроблена програма має дуже простий і інтуїтивно 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.

Обрано умови розповсюдження – Freeware.

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

На рисунку 5.2 зображена форма авторського права.

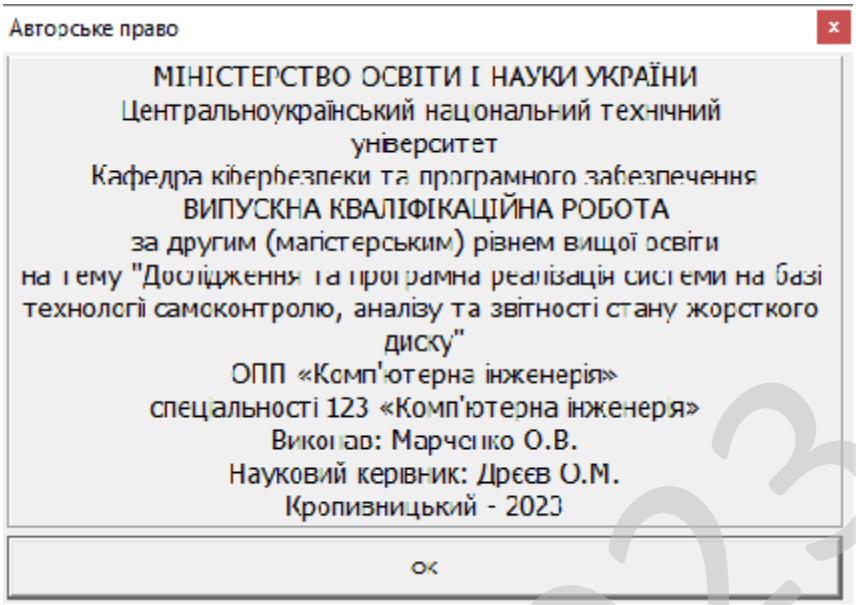


Рисунок 5.2 – Авторське право

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 на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Метою розробки є дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Об'єктом дослідження є процес на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Предметом дослідження є методи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Методи дослідження базуються на методах теорії обчислювальних систем, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– Розроблено вітчизняний продукт на базі технології самоконтролю, аналізу та звітності стану жорсткого диску, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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	150
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	15000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	60
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}02\ 7.$$

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

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:

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

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	90	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	3	360	6
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м. п.	2,5	350	875	14,58
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	56,08

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф^{с}_{др} = \frac{56 \cdot 3}{1,2} = 140 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 140/(60 \cdot 8) = 0{,}3 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,5	0,25
	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	0,5	
Всього		2	

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	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	0,5	0,2
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	

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	15496	46488
Продакт-менеджер	0,25	15000	11250
Інженер-програміст	3,8	15000	171000
Інженер-електронщик	0,3	15000	13500
Адміністратор мережі	0,25	15000	11250
Дизайнер WEB	0,2	15000	9000
Інженер-верстальник	0,2	15000	9000
Бухгалтер-економіст	0,2	15000	9000
Всього за період розробки	$R_{сп}$ = 6,2	-	$Ф_{роб}$ = 280488

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.

$$З_{сд} = \frac{280488}{6,2 \cdot 60} = 754 \; грн.$$

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, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 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 13500 грн. на одне робоче місце. Тобто:

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

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

$$I_{нв} = 8 \cdot 13500 = 108000 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.6.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 29.10.23. (джерело https://compbest.com.ua.)

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10747
Системний блок	HP ProDesk 600 G2 Tower	7347
Процесор	6th gen Intel® Core™ i3 i3-6100 (3M Cache, 2 (4) ядра по 3.7 GHz)	-
Системна плата	Intel® Q150, PCI Express x16 – 1, PCI Express x1 – 3, DDR4-SDRAM, 4x USB 2.0, 6x USB 3.0, 4x Audio Ports, RJ-45 (LAN), VGA, 2x DP, COM-Port, 2x PS/2.	-
Відеокарта	Вбудована Intel® HD Graphics 530	-
Жорсткий диск	HDD: 500 Gb 7200 Serial ATA	-
Оперативна пам’ять	8GB (2133 MHz) DDR4-SDRAM (2 x 4 GB)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	ATX Middle Tower Pro, 3GTLA-489, PSU 450W(FSP Brand: ATX-350PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3400
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.

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	8	10747	8597,6	94573,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	113125,1

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

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	113125	-	-
Всього по групі	113125	40	45250
Нематеріальні активи			
4. Нематеріальні активи	15000	10	1500
Група 5, 6			
5. Вимірювальні пристрої	5190	25	1297,5
6. Транспортні засоби	0	20	0,0
7. Господарський інвентар	108000	25	27000
Всього по групі	113190	-	28297,5
Разом	$К_р$ = 1649315		$А_р$ = 145447,5

Примітка: вартість транспортного засобу приймаємо рівним нулю.

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.

$$З_о = 754 \cdot 209/150 = 1050 \text{ грн.}$$

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, %.

$$З_д = 1050 \cdot 10 \cdot 0{,}01 = 105 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(1050+105) = 254 \text{ грн.}$$

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, %.

$$Г_{осп} = 1050 \cdot 15 \cdot 0{,}01 = 158 \text{ грн.}$$

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

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

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

Згідно виданих норм приймаємо 0,6 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 200$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

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

$$З_{М1} = 200 \cdot 0,6 \cdot 3 = 360 \text{ грн.}$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 32,5 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 32,5 грн./шт.

$$З_{М2} = 32,5 \cdot 10 = 325 \text{ грн.}$$

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.

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

$$З_М = (360 + 325 + 1702)/150 = 16 \text{ грн.}$$

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, %.

$$О_п = 1050 \cdot 15 \cdot 0,01 = 158 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е = 150$ прим.):

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

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

$$A_м = 145448 \cdot 3/(150 \cdot 12) = 242 \text{ грн.}$$

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	$З_о$	1050
2. Додаткова зарплата виконавців	$З_д$	105
3. Відрахування на соціальні потреби	$С_{оц}$	254
4. Загальногосподарські витрати	$Г_{осп}$	158
5. Витрати на матеріали	$З_М$	16
6. Освоєння нових операційних систем, мов програмування	$О_п$	158
7. Амортизація основних фондів	$А_м$	242
8. Повна собівартість програмного забезпечення	$С_п$	1983
9. Плановий прибуток	$П_р$	1189,8
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	3172,8
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	634,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3807,4

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

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

$$C_п = 1050+105+254+158+16+158+242 = 1983 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 60%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 60 \cdot 1983 = 1189{,}8 \text{ грн.}$$

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. Вартість програмного рішення взятого для порівняння складе 4000 грн. 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	4000	3807
Всього капітальних витрат	4000	3807

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	61195	51533
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	1000	952
Всього витрат за рік	$І$	62195	52485

Витрати на обслуговування системи:

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

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

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

Після купівлі нового програмного забезпечення кількість годин для обслуговування системи зменшилось з 380 годин на рік до 320 годин на рік, тому витрати складуть:

$$З_{р\,баз} = 380 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 61195 \text{ грн},$$

до:

$$З_{р\,нов} = 320 \cdot 120 \cdot 1{,}1 \cdot 1{,}22 = 51533 \text{ грн}.$$

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

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

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	4000	3807	1000	951,75
Всього відрахувань	-	4000	3807	1000	951,75

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:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum E_р K_р, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.

$$E_в = (3172,8 - 1983) \cdot 150 - (0,05 \cdot 1408000 + 0,4 \cdot 113125 + 0,25 \cdot 113190 + 0,1 \cdot 15000) \frac{3}{12} = 142108 \text{ грн.}$$

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.

$$T_в = \frac{1649315}{(3172,8 - 1983) \cdot 150 \cdot 12 / 3} = 2,3 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	150
2. Повна собівартість розробленої програми	Грн.	1983
3. Ціна розробленої програми	Грн.	3172,8
4. Плановий прибуток від реалізації розробленої програми	Грн.	1189,8
5. Рентабельність програмної продукції	%	60
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1649315
7. Загальний прибуток від реалізації програмної продукції	Грн.	178470
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	142108
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,3
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3807
11.Величина економічного ефекту у користувача програмної продукції	Грн.	9758
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (62195+0{,}25\cdot4000)-(52485+0{,}25\cdot3807) = 9758 \text{ грн.}$$

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

$$T_{сп} = \frac{\Delta K}{E_{сп}}, \tag{7.28}$$

$$T_{сп} = \frac{4000-3807}{9758} < 0{,}1 \; року .$$

Як бачимо з розрахунків запропонований варіант є більш економічно доцільним ніж варіант вибраний для порівняння, який на даний момент є лідером продаж на ринку.

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

Техніка безпеки – це система правил і заходів, які допомагають запобігти травмам, хворобам і аваріям на робочому місці або в повсякденному житті. Знання техніки безпеки дуже важливе, бо воно рятує життя і здоров'я людей.

Законом України "Про охорону праці" [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», 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 істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

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

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

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;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– 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	6
Довжина	7
Висота	2,9

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	7
Об'єм, V	$м^3$	не менше 20.0	20,3

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

У зазначеному приміщенні працюють 6 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

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

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

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

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

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

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 Prinics PicKit M1 Smartphone Photo Printer White, електродвигуни вентиляторів ЕОМ.

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

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

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк., Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

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

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

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

8.5 Розрахункова частина

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

Для захисного штучного заземлення застосовуються ветикальні електроди з метелевого прутка діаметром 30 мм. (*D=30 мм.=0,03 м.)* довжиною *L*=2,5 *м.* та горизонтальний електрод – метелева полоса з перетином 40*4 мм. тип грунта – глина (питомий опір 40 Ом*м). Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*.

Глибина закладення горизонтального контура заземлення *t= 0,8 м*.

Умовна товщина верхнього шару грунта: *H=0,4 м*. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

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

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,8+2,5/2=2,05 \text{ м.}$$

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

$$\rho = \psi \rho_2 = 1,36*40=54,5 \text{ Ом*м.}$$

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

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

$\rho_2 = 40$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта [8].

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

$$R_o = 0,366\frac{\rho}{L}\left(lg\frac{2L}{D}+\frac{1}{2}lg\frac{4T+L}{4T-L}\right)= 0,366\frac{54,5}{2,5}\left(lg\frac{2\cdot 2,5}{0,03}+\frac{1}{2}lg\frac{4\cdot 2,05+2,5}{4\cdot 2,05-2,5}\right)= 20,1 \text{ Ом.}$$

Відношення $A/L=3/2,5=1,2$.

Визначаємо коефіцієнт екранування ветикальньних електродів $Кев=0,8$ при попередній (орієтовній) кількості ветикальньних електродів, яке дорівнює 4 [8].

Визначаємо необхідну кількість вертикальних заземлювачів (без вхарування горизонтального заземлювача), при $R_{3H} = 4$ *Ом* :

$$N=R_O /(Кев\ R_{3H})= 20,1 / (0,8*4)=5,87 \approx 6 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси [8]:

$$L_П =1,05*A*N= 1,05*3*6=18,8\approx 19 \text{ м.}$$

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

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

$$=0,366(40*5/16)*[lg(2*16*16)/(0,04*0,8)]=16,5 \text{ Ом.}$$

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

Загальний опір розтіканню електричного струму заземлювача [8]:

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

$$=(20,1* 16,5)/(20,1* 0,75+ 6* 16,5* 0,8)=3,32\ Ом.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [8].

Умова $R \le R_{ЗН}$ виконується ($3,32 \le 4$).

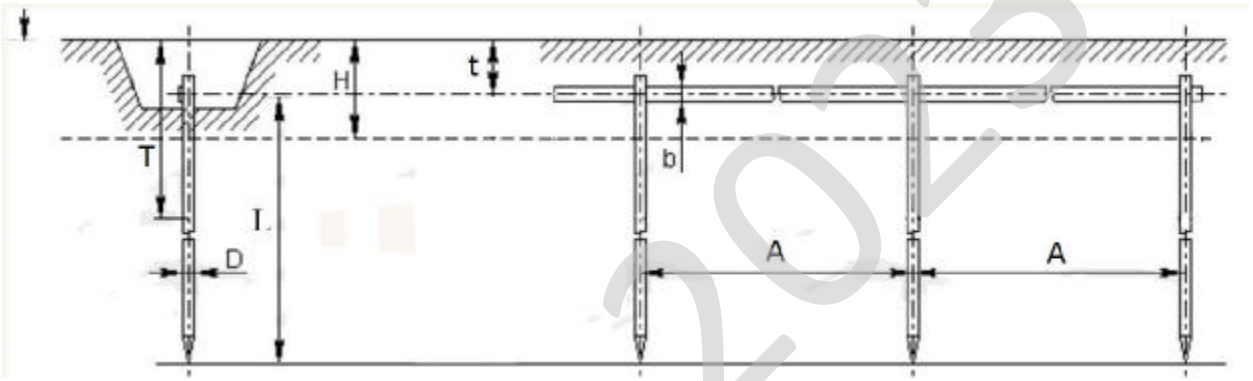


Рисунок 8.1 – Схема штучного заземлення

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.

Список використаних джерел інформації

1. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

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

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

5. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

6. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ 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

7. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

8. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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 на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– Досліджена система на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

– На основі отриманих результатів досліджень створена програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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

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

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

Програма реалізована на мові високого рівня Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Serpent.

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 9758 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Марченко О.В. Дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9
3. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.
4. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.
5. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.
6. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.
7. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.
8. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.
9. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.
10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises».

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

30. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

31. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

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

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

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

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

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

37. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

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

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

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

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

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

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

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

45. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску.

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 на базі технології самоконтролю, аналізу та звітності стану жорсткого диску;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий розрахунок захисного штучного заземлення.

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 – 117 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 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Дрєєв О.М.

Дослідження та програмна реалізація системи на базі технології самоконтролю, аналізу та звітності стану жорсткого диску

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 55

Літера: РП

Кропивницький – 2023 року

ФАЙЛ ПРОЕКТУ РОЗРОБЛЕНОГО ПЗ PROGRAM PROJECT_SMART.DPR

```
program Project_SMART; // Назва ПЗ

uses // Підключення  бібліотек
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
  Unit2 in 'Unit2.pas' {Form2};
  Unit3 in 'Unit3.pas' {Form3};
  Unit4 in 'Unit4.pas' {Form4};
  Unit5 in 'Unit5.pas' {Form5};

{$R *.res} // Ресурси

Begin // Початок основного процесу
  Application.Initialize; // Ініціалізація ПЗ
  Application.Title := 'Дані S.M.A.R.T. підсистеми';
  Application.CreateForm(TForm1, Form1);  // Підключення форм
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.CreateForm(TForm5, Form5);
  Application.Run; // Запус процесу
end. // Кінець роботи ПЗ
```

ФАЙЛ МОДУЛЮ UNIT5.PAS

```
unit Unit5; // Об’ява модулю

interface

uses
  Sysutils, Windows, Messages, Classes, ShellAPI, nb30, Registry, StrUtils;


const
        MaxByte: Byte = 255;
        MaxShortInt: ShortInt = 127;
        MaxWord: Word = 65535;
        MaxTriplet: LongInt  = $FFFFFF;
        MaxLongInt: LongInt  = $7FFFFFFF;    // 2147483647
        MaxInteger = $7FFFFFFF;
        MaxLongWord: LongWord = $FFFFFFFF;   // 4294967295
        MaxLongWrd = $FFFFFFFF;
        MaxInt64: int64 = $7FFFFFFFFFFFFFFF;
        MaxReal: Real = 1.7e38;
        MaxSingle: Single = 3.4e38;
        MaxDouble: Double = 1.7e308;
        MaxExtended: Extended = 1.1e4932;
        MinByte: Byte = 0;
        MinShortInt: ShortInt = -128;
        MinInt: Integer = -32768;
        MinWord: Word = 0;
        MinLongInt = $80000000;
        MinReal: Real = 2.9e-39;
        MinSingle: Single = 1.5e-45;
        MinDouble: Double = 5.0e-324;
        MinExtended: Extended = 3.4e-4932;

const

function IsWin95: boolean;
function IsWinNT: boolean;
function IsWin2K: boolean;
function IsWinXP: boolean;
function IsWinXPE: boolean;
function IsWin2K3: boolean;
function IsWinVista: boolean;
function IsWin2K8: boolean;
function GetOSVersion: string;
procedure GetOSInfo;
function IsSpace (Ch: Char): Boolean;
function IsDigit (Ch: Char): Boolean;
function IsLetterOrDigit (Ch: Char): Boolean;
function IsPathSep (Ch: Char): Boolean;
function IsDigitsDec (info: string; decimal: boolean) : boolean;
function IsDigits (info: string) : boolean;

procedure ConvHexStr (instr: string; var outstr: string);
procedure ByteSwaps(DataPtr : Pointer;NoBytes : integer);
function ConIntHex (value: cardinal): string;
function StripQuotes (filename: string): string;
function StripNewLines (const S: string): string;
function IndexFiles (searchfile: string; mask: integer;
                  var FileList: TStringList; var totsize: cardinal): integer;
function DeleteOldFiles (fname: string): integer;
function GetEnvirVar (name: string): string;

function StripChars (AString, AChars: String): String;
function UpAndLower (const S: String): String;
function StripChar (const AString: String; const AChar: Char): String;
function StripSpaces (const AString: String): String;
function StripCommas (const AString: String): String;
function StripNulls (const AString: String): String;
function StripAllCntls (const AString: String): String;
```

```
function StripCharsAnsi (AString, AChars: AnsiString): AnsiString;
function UpAndLowerAnsi (const S: AnsiString): AnsiString;
function StripCharAnsi (const AString: AnsiString; const AChar: AnsiChar):
AnsiString;
function StripSpacesAnsi (const AString: AnsiString): AnsiString;
function StripCommasAnsi (const AString: AnsiString): AnsiString;
function StripNullsAnsi (const AString: AnsiString): AnsiString;
function StripAllCntlsAnsi (const AString: AnsiString): AnsiString;

procedure StringTranCh (var S: String; FrCh, ToCh: Char);
procedure StringTranChAnsi (var S: AnsiString; FrCh, ToCh: AnsiChar);
procedure StringCtrlSafe (var S: AnsiString);
procedure StringCtrlRest (var S: AnsiString);
function StrCtrlSafe (const S: AnsiString): AnsiString;
function StrCtrlRest (const S: AnsiString): AnsiString;
procedure StringFileTran (var S: String);
function StringRemCntls (var S: String): boolean;
function StringRemCntlsEx (var S: String): boolean;
procedure DosToUnixPath (var S: String);
procedure UnixToDosPath (var S: String);
function UnxToDosPath (const S: String): String;
function DosToUnxPath (const S: String): String;
function StrFileTran (const S: String): String;
procedure StringFileTranEx (var S: String);
function StrFileTranEx (const S: String): String;

// фалові перетворення
function FileTimeToInt64 (const FileTime: TFileTime): Int64;
function Int64ToFileTime (const FileTime: Int64): TFileTime;
function FileTimeToDateTime(const FileTime: TFileTime): TDateTime;
function DateTimeToFileTime(DateTime: TDateTime): TFileTime;
function FileTimeToSecs2K (const FileTime: TFileTime): integer;
function CheckFileOpen(const FName: String): integer;
function TruncateFile(const FName: String; NewSize: int64): int64;
function GetSizeFile (filename: string): LongInt;
function GetSize64File (filename: string): Int64;
function GetFUAgeSizeFile (filename: string; var FileTime: TFileTime;
                                             var FSize: Int64): boolean;
function GetUAgeSizeFile (filename: string; var FileDT: TDateTime;
                                             var FSize: Int64): boolean;
function GetFAgeSizeFile (filename: string; var FileTime: TFileTime;
                                             var FSize: Int64): boolean;
function GetAgeSizeFile (filename: string; var FileDT: TDateTime;
                                             var FSize: Int64): boolean;
function TrimSpRight(const S: string): string;
function ExtractNameOnly (FileName: string): string;

function GetExceptMess (ExceptObject: TObject): string;

{ Конвертація String into a LongInt }
function Str2LInt (const S: String): LongInt;

{ Конвертація String into a Word }
function Str2Word (const S: String): Word;

{ Конвертація String into a Byte }
function Str2Byte (const S: String): Byte;

{ Конвертація String into a ShortInt }
function Str2SInt (const S: String): ShortInt;

{ Конвертація String into an Integer }
function Str2Int (const S: String): Integer;

{ Конвертація LongInt into a String of length N with
        zeros Padding to the Left }
function Int2StrZ (const L: LongInt; const Len: Byte): String;
```

```
{ Конвертація LongInt into a String of length N with
        NumPadCh Padding to the Left }
function LInt2Str (const L: LongInt; const Len: Byte): String;

{ Конвертація LongInt into a String of length N with
        NumPadCh Padding to the Left }
function Byte2Str (const L: LongInt; const Len: Byte): String;

{ Конвертація LongInt into a String of length N with
        NumPadCh Padding to the Left }
function LInt2ZStr (const L: LongInt; const Len: Byte): String;

{ Конвертація LongInt into a String of length N with
        NumPadCh Padding to the Left, with blanks returned
       if Value is 0 }
function LInt2ZBStr (const L: LongInt; const Len: Byte): String;

{ Конвертація LongInt into a Comma'ed String of length Len,
        with NumPadCh Padding to the Left }
function LInt2CStr (const L : LongInt; const Len : Byte): string;

{ Конвертація LongInt into an exact String, No Padding }
function LInt2EStr (const L: LongInt): String;

{ Конвертація LongInt into an exact String, No Padding,
        with null returned if Value is 0 }
function LInt2ZBEStr (const L: LongInt): String;

{ Конвертація LongInt into a Comma'ed String without Padding }
function LInt2CEStr (const L : LongInt): string;

{ Конвертація Int64 to a comma'ed string, no padding }
function Int642CEStr (const L : Int64): string;

{ Повертає рядок, що складається of N occurrences of Ch. }
function FillStr (const Ch : Char; const N : Integer): string;

{ Повертає рядок, що складається of N blank spaces (i.e. #32) }
function BlankStr (const N : Integer): string;

{ Повертає рядок, що складається of N occurrences of '-'. }
function DashStr (const N : Integer): String;

{ Повертає рядок, що складається of N occurrences of '='. }
function DDashStr (const N : Integer): string;

{ Повертає рядок, що складається of N occurrences of 'Д' (196). }
function LineStr (const N : Integer): string;

{ Повертає рядок, що складається of N occurrences of 'Н' (205). }
function DLineStr (const N : Integer): string;

{ Повертає рядок, що складається of N occurrences of '*'. }
function StarStr (const N : Integer): string;

{ Повертає рядок, що складається of N occurrences of '#'. }
function HashStr (const N : Integer): string;

function PadRightStr (const S : string; const Len : Integer): string;

function DateTimeToAStr(const DateTime: TDateTime): string; // always alpha
month and numeric hh:mm:ss
function DateToAStr(const DateTime: TDateTime): string; // always alpha month
function TimeToNStr(const DateTime: TDateTime): string; // always numeric
hh:mm:ss
function TimeToZStr(const DateTime: TDateTime): string; // always numeric
hh:mm:ss:zzz
function timeHour(T: TDateTime): Integer;
function timeMin(T: TDateTime): Integer;
```

```
function timeSec(T: TDateTime): Integer;
function timeToInt(T: TDateTime): Integer;  // seconds
function HoursToTime (hours: integer): TDateTime;
function MinsToTime (mins: integer): TDateTime;
function SecsToTime (secs: integer): TDateTime;
function TimerToStr (duration: TDateTime): string;
function PackedISO2Date (info: string): TDateTime;
function PackedISO2UKStr (info: string): string;
function DTtoISODT (D: TDateTime): string;
function AlphaDTtoISODT (sdate, stime: string): string;
function ISODTtoPacked (ISO: string): string;
function PackedDTtoISODT (info: string): string;
function GetUTCTime: TDateTime;
function QuoteNull (S: string): string;

function strLastCh(const S: String): Char;
procedure strStripLast(var S: String);
function strAddSlash(const S: String): String;
function strDelSlash(const S: String): String;
function ExtractUNIXPath(const FileName: string): string;
function ExtractUNIXName(const FileName: string): string;
function GetYN (const value: boolean): char;

function CharPos (TheChar: AnsiChar; const Str: AnsiString): Integer;
function DownCase( ch : AnsiChar ) : AnsiChar;
function ConvHexQuads (S: string): string;

function NowPC : TDateTime;
function GetPerfCountsPerSec: int64;
function PerfCountCurrent: int64;
function PerfCountToMilli (LI: int64): integer;
function PerfCountGetMilli (startLI: int64): integer;
function PerfCountGetMillStr (startLI: int64): string;
function PerfCountToSecs (LI: int64): integer;
function PerfCountGetSecs (startLI: int64): integer;

function InetParseDate(const DateStr: string): TDateTime;
function URLEncode(const psSrc: AnsiString): AnsiString;
function URLDecode(const AStr: AnsiString): AnsiString;

function FormatLastError: string;
function Int2Kbytes (value: integer): string;
function Int2Mbytes (value: int64): string;
function IntToKbyte (Value: Int64): String;

procedure EmptyRecycleBin (fname: string);
procedure TrimWorkingSetMemory;
procedure FreeAndNilEx(var Obj);
function IsProgAdmin: Boolean;
function GetTickCountX: longword;
function DiffTicks (const StartTick, EndTick: longword): longword;
function ElapsedTicks (const StartTick: longword): longword;
function ElapsedMsecs (const StartTick: longword): longword;
function ElapsedSecs (const StartTick: longword): integer;
function ElapsedMins (const StartTick: longword): integer;
function WaitingSecs (const EndTick: longword): integer;
function GetTrgMSecs (const MilliSecs: integer): longword;

type
  TOSVERSIONINFOEXW = record
    dwOSVersionInfoSize: DWORD;
    dwMajorVersion: DWORD;
    dwMinorVersion: DWORD;
    dwBuildNumber: DWORD;
    dwPlatformId: DWORD;
    szCSDVersion: array[0..127] of WideChar
    wServicePackMajor: WORD;
    wServicePackMinor: WORD;
    wSuiteMask: WORD;
```

```
    wProductType: BYTE;
    wReserved: BYTE;
  end;

// handle for DLL
var
    SensapiModule: THandle;

type
  TOSVersion = (OSW9x, OSNT4, OSW2K, OSWXP, OSVista);
var
  MagRasOSVersion: TOSVersion;

var
IsDestinationReachable: function (lpszDestination: PWideChar;
var QocInfo: TQocInfo): bool; stdcall;

IsNetworkAlive: function (var Flags: DWORD): bool; stdcall;

function SHGetSpecialFolderLocation (handle: HWND; nFolderL: integer;
                                                    LPITEMIDLIST: pointer): bool
stdcall;
function SHGetPathFromIDList (LPCITEMIDLIST: pointer;
                                                    pszPath: PWideChar): bool
stdcall;  // unicode

function SHEmptyRecycleBin (Wnd:HWnd; pszRootPath:PWideChar;
Flags:DWORD):Integer; stdcall;   // unicode

function SHGetPathFromIDList; external shell32 name 'SHGetPathFromIDListW';
// unicode

function SHEmptyRecycleBin; external shell32 name 'SHEmptyRecycleBinW';


implementation


function TrimAnsi(const S: AnsiString): Ansistring;
var
  I, L: Integer;
begin
  L := Length(S);
  I := 1;
  while (I <= L) and (S[I] <= ' ') do Inc(I);
  if I > L then Result := '' else
  begin
    while S[L] <= ' ' do Dec(L);
    Result := Copy(S, I, L - I + 1);
  end;
end;

function TrimLeftAnsi(const S: AnsiString): AnsiString;
var
  I, L: Integer;
begin
  L := Length(S);
  I := 1;
  while (I <= L) and (S[I] <= ' ') do Inc(I);
  Result := Copy(S, I, Maxint);
end;

function TrimRightAnsi(const S: Ansistring): AnsiString;
var
  I: Integer;
begin
  I := Length(S);
  while (I > 0) and (S[I] <= ' ') do Dec(I);
  Result := Copy(S, 1, I);
```

```
end;

function LowerCaseAnsi(const S: AnsiString): AnsiString;
var
    Ch : AnsiChar;
    L, I  : Integer;
    Source, Dest: PAnsiChar;
begin
    L := Length(S);
    if L = 0  then
        Result := ''
    else begin
        SetLength(Result, L);
        Source := Pointer(S);
        Dest := Pointer(Result);
        for I := 1 to L do begin
            Ch := Source^;
            if Ch in ['A'..'Z'] then Inc(Ch, 32);
            Dest^ := Ch;
            Inc(Source);
            Inc(Dest);
        end;
    end;
end;

function UpperCaseAnsi(const S: AnsiString): AnsiString;
var
    Ch : AnsiChar;
    L, I : Integer;
    Source, Dest: PAnsiChar;
begin
    L := Length(S);
    if L = 0  then
        Result := ''
    else begin
        SetLength(Result, L);
        Source := Pointer(S);
        Dest := Pointer(Result);
        for I := 1 to L do begin
            Ch := Source^;
            if Ch in ['a'..'z'] then Dec(Ch, 32);
            Dest^ := Ch;
            Inc(Source);
            Inc(Dest);
        end;
    end;
end;

function CompareTextAnsi(const S1, S2: AnsiString): Integer;
var
    L1, L2, I : Integer;
    MinLen : Integer;
    Ch1, Ch2 : AnsiChar;
    P1, P2 : PAnsiChar;
begin
    L1 := Length(S1);
    L2 := Length(S2);
    if L1 > L2 then
        MinLen := L2
    else
        MinLen := L1;
    P1 := Pointer(S1);
    P2 := Pointer(S2);
    for I := 1 to MinLen do
    begin
        Ch1 := P1[I];
        Ch2 := P2[I];
        if (Ch1 <> Ch2) then
        begin
```

```
            if (Ch1 > Ch2) then
            begin
                if Ch1 in ['a'..'z'] then
                    Dec(Byte(Ch1), 32);
            end
            else begin
                if Ch2 in ['a'..'z'] then
                    Dec(Byte(Ch2), 32);
            end;
        end;
        if (Ch1 <> Ch2) then
        begin
            Result := Byte(Ch1) - Byte(Ch2);
            Exit;
        end;
    end;
    Result := L1 - L2;
end;

function IntToStrAnsi(N : Integer) : AnsiString;
var
    I : Integer;
    Buf : array [0..11] of AnsiChar;
    Sign : Boolean;
begin
    if N >= 0 then
        Sign := FALSE
    else begin
        Sign := TRUE;
        if N = Low(Integer) then
        begin
            Result := '-2147483648';
            Exit;
        end
        else
            N := Abs(N);
    end;
    I := Length(Buf);
    repeat
        Dec(I);
        Buf[I] := AnsiChar(N mod 10 + $30);
        N := N div 10;
    until N = 0;
    if Sign then begin
        Dec(I);
        Buf[I] := '-';
    end;
    SetLength(Result, Length(Buf) - I);
    Move(Buf[I], Pointer(Result)^, Length(Buf) - I);
end;

function IntToHexAnsi(N : Integer; Digits: Byte) : AnsiString;
var
    Buf : array [0..7] of Byte;
    V : Cardinal;
    I : Integer;
begin
    V := Cardinal(N);
    I := Length(Buf);
    if Digits > I then Digits := I;
    repeat
        Dec(I);
        Buf[I] := V mod 16;
        if Buf[I] < 10 then
            Inc(Buf[I], $30)
        else
            Inc(Buf[I], $37);
        V := V div 16;
    until V = 0;
```

```
    while Digits > Length(Buf) - I do begin
       Dec(I);
       Buf[I] := $30;
    end;
    SetLength(Result, Length(Buf) - I);
    Move(Buf[I], Pointer(Result)^, Length(Buf) - I);
end;

function PosAnsi(const Substr, S: AnsiString): Integer;
var
  P: PAnsiChar;
begin
  Result := 0;
  P := AnsiStrPos(PAnsiChar(S), PAnsiChar(SubStr));
  if P <> nil then
    Result := Integer(P) - Integer(PAnsiChar(S)) + 1;
end;

function StrLenWide(const Str: PWideChar): Cardinal;
asm
  cmp word ptr [eax], 0
  je @ZeroLength
  mov edx, eax
  neg edx
@ScanLoop:
  mov cx, [eax]
  add eax, 2
  test cx, cx
  jnz @ScanLoop
  lea eax, [eax + edx - 2]
  shr eax, 1
  ret
@ZeroLength:
  xor eax, eax
end;

function StrLCopyWide(Dest: PWideChar; const Source: PWideChar; MaxLen:
Cardinal): PWideChar;
var
  Len: Cardinal;
begin
  Result := Dest;
  Len := StrLenWide(Source);
  if Len > MaxLen then
    Len := MaxLen;
  Move(Source^, Dest^, Len * SizeOf(WideChar));
  Dest[Len] := #0;
end;

function StrPLCopyWide(Dest: PWideChar; const Source: String; MaxLen: Cardinal):
PWideChar;
var
  W: WideString;
begin
  W := Source;
  Result := StrLCopyWide(Dest, PWideChar(W), MaxLen);
end;

function FixedToPasStr (fixstr: PAnsiChar; fixsize: integer): AnsiString;
var
    temp: AnsiString;
begin
    SetLength (temp, fixsize);
    Move (fixstr^, PAnsiChar (temp)^, fixsize);
    result := temp;
end;


function GetDevNamePort (fixstr: PAnsiChar; fixsize: integer;
```

```
                                        var devport: AnsiString): AnsiString;
var
    I: integer;
    temp: AnsiString;
begin
    devport := '';
    result := '';
    temp := TrimRightAnsi (FixedToPasStr (fixstr, fixsize));
    if Length (temp) = 0 then exit;
    I := CharPos (#0, temp);
    if I > 1 then
    begin
        temp [I] := '{';
        devport := LowerCaseAnsi (TrimAnsi (Copy (temp, I + 1, 99)));
        result := TrimAnsi (Copy (temp, 1, I - 1));
    end
    else
        result := temp;
end;

function FixedToPasStrW (fixstr: PWideChar; fixlen: integer): WideString;
begin
    SetLength (Result, fixlen);
    Move (fixstr^, PWideChar (result)^, fixlen * 2);
 end;

function GetDevNamePortW (fixstr: PWideChar; fixlen: integer;
                                         var devport: WideString): WideString;
var
    I: integer;
    temp: WideString;
begin
    devport := '';
    result := '';
    temp := TrimRight (FixedToPasStrW (fixstr, fixlen));
    if Length (temp) = 0 then exit;
    I := Pos (#0, temp);
    for I := 1 to Length (temp) do
    begin
        if temp [I] = #0 then break;
    end;
    if (I > 1) and (I < Length (temp)) then
    begin
        temp [I] := '{';
        devport := LowerCase (Trim (Copy (temp, I + 1, 99)));
        result := Trim (Copy (temp, 1, I - 1));
    end
    else
        result := temp;
end;

function GetWinDir: String;
var
    Path: array [0..MAX_PATH] of WideChar;   // Unicode
    NLen: DWORD;
begin
    Path [0] := #0;
    NLen := GetWindowsDirectoryW (Path, Length (Path)); // Unicode
    SetString (Result, Path, NLen);
end;

function GetShellPath (location: integer): string;
var
    PIDL: Pointer;
    Path: array [0..MAX_PATH] of WideChar;   // Unicode
begin
    Result := '';
    Path [0] := #0;
    SHGetSpecialFolderLocation (HInstance, location, @PIDL);
```

```
      if SHGetPathFromIDList (PIDL, Path) then Result := Path;
end;

function GetUsersName: string;
var
    Buffer: array[0..255] of WideChar;
    NLen: DWORD;
begin
    Buffer [0] := #0;
    result := '';
    NLen := Length (Buffer);
    if GetUserNameW (Buffer, NLen) then SetString (Result, Buffer, NLen);
end;

function GetCompName: string;
var
    Buffer: array[0..255] of WideChar;
    NLen: DWORD;
begin
    Buffer [0] := #0;
    result := '';
    NLen := Length (Buffer);
    if GetComputerNameW (Buffer, NLen) then SetString (Result, Buffer, NLen);
end;

function TStamptoDT (stamp: DWORD): TDateTime;
begin
    result := (stamp / SecsPerDay) + 25569;
end;

function TDTtoStamp (D: TDateTime): DWORD;
begin
    result := 0;
    if D < 25569 then exit;
    D := D - 25569;
    if D > 21900 then exit;
    result := Trunc (D * SecsPerDay);
end;

function ExcludeTrailingBackslash(const S: string): string;
begin
  Result := S;
  if IsPathDelimiter(Result, Length(Result)) then
    SetLength(Result, Length(Result)-1);
end;

function DirectoryExists(const Name: string): Boolean;
var
    Code: DWORD;
begin
    Code := GetFileAttributes (PChar(Name));
    Result := (Code <> $FFFFFFFF) and (FILE_ATTRIBUTE_DIRECTORY and Code <> 0);
end;

function ExcludeTrailingPathDelimiter(const S: string): string;
begin
  Result := S;
  if IsPathDelimiter(Result, Length(Result)) then
    SetLength(Result, Length(Result)-1);
end;

function ForceDirs (Dir: string): Boolean;
begin
    Result := True;
    if Length(Dir) = 0 then
    begin
        Result := false;
        exit;
    end;
```

```
end;

function IsWin95: boolean;
begin
    if OsInfo.dwPlatformId = 0 then GetOSInfo;
    result := false;
    if OsInfo.dwPlatformId = VER_PLATFORM_WIN32_WINDOWS then result := true;
end;

function IsWinNT: boolean;
begin
    if OsInfo.dwPlatformId = 0 then GetOSInfo;
    result := false;
    if OsInfo.dwPlatformId = VER_PLATFORM_WIN32_NT then result := true;
end;

function IsWin2K: boolean;
begin
    result := false;
    if IsWinNT and (OsInfo.dwMajorVersion >= 5) then result := true;
end;

function IsWinXP: boolean;
begin
    result := false;
    if IsWin2K and (OsInfo.dwMinorVersion > 0) then result := true;
end;

function IsWinXPE: boolean;
begin
    result := false;
    if IsWinXP and (((OsInfo.wSuiteMask AND VER_SUITE_EMBEDDEDNT) <> 0) or
      ((OsInfo.wSuiteMask AND VER_SUITE_EMBEDDED_RESTRICTED) <> 0)) then
      result := true;
end;

function IsWin2K3: boolean;
begin
    result := false;
    if IsWin2K and (OsInfo.dwMinorVersion >= 2) then result := true;
end;

function IsWinVista: boolean;
begin
    result := false;
    if IsWinNT and (OsInfo.dwMajorVersion = 6) and
                  (OsInfo.wProductType <= VER_NT_WORKSTATION) then result:=
true;
end;

function IsWin2K8: boolean;
begin
    result := false;
    if IsWinNT and (OsInfo.dwMajorVersion = 6) and
    (OsInfo.wProductType > VER_NT_WORKSTATION) then result := true;
end;

procedure GetOSInfo;
begin
    FillChar (OsInfo, sizeof (TOSVERSIONINFOEXW), 0);
    OsInfo.dwOSVersionInfoSize := sizeof (TOSVERSIONINFOEXW);
    if GetVersionExW2 (OsInfo) then exit;
    OsInfo.dwOSVersionInfoSize := sizeof (TOSVERSIONINFOW);
    GetVersionExW2 (OsInfo);
end;

function IsSpace (Ch: Char): Boolean;
begin
    Result := (Ch = ' ') or (Ch = Char($09));
```

```
end;

function IsLetterOrDigit (Ch: Char): Boolean;
begin
    Result := ((Ch >= 'a') and (Ch <= 'z')) or
              ((Ch >= 'A') and (Ch <= 'Z')) or
              ((Ch >= '0') and (Ch <= '9'));
end;

function IsDigit(Ch : Char): Boolean;
begin
    Result := (ch >= '0') and (ch <= '9');
end;

function IsPathSep (Ch: Char): Boolean;
begin
    Result := (Ch = '.') or (Ch = '\') or (Ch = ':');
end;

function IsDigitsDec (info: string; decimal: boolean) : boolean;
var
    count, len: integer;
    onedotflag: boolean;
begin
    result := false;
    onedotflag := false;
    info := trim (info);
    len := length (info);
    if len = 0 then exit;
    for count := 1 to len do
    begin
        if NOT IsDigit (info [count]) then
        begin
            if (count <> 1) then
            begin
                if NOT decimal then exit;
                if info [count] <> DecimalSeparator then exit;
                if onedotflag then exit;
                onedotflag := true;
            end
            else
            begin
                if (info [1] = '-') or (info [1] = '+') then
                begin
                    if (len = 1) then exit;
                end
                else
                    exit;
            end;
        end;
    end;
    result := true;
end;

function IsDigits (info: string) : boolean;
begin
    result := IsDigitsDec (info, false);
end;

procedure ByteSwaps(DataPtr : Pointer;NoBytes : integer);
var
    i : integer;
    dp : PAnsiChar;
    tmp : AnsiChar;
begin
    if (NoBytes > 1) then
    begin
        Dec(NoBytes);
        dp := PAnsiChar(DataPtr);
```

```
        for i := NoBytes downto (NoBytes div 2 + 1) do
        begin
            tmp := PAnsiChar(Integer(dp)+i)^;
            PAnsiChar(Integer(dp)+i)^ := PAnsiChar(Integer(dp)+NoBytes-i)^;
            PAnsiChar(Integer(dp)+NoBytes-i)^ := tmp;
        end;
    end;
end;

procedure ConvHexStr (instr: string; var outstr: string);
var
    flen, inx, nr1, nr2, outpos: integer;
begin
    flen := Length (instr);
    if flen = 0 then exit;
    SetLength (outstr, flen * 2);
    outpos := 1;
    for inx := 1 To flen do
    begin
        nr1 := ord (instr [inx]);
        nr2 := nr1 SHR 4;
        If (nr2 > 9) then nr2 := nr2 + 7;
        outstr [outpos] := Chr (nr2 + 48);
        inc (outpos);
        nr2 := nr1 and 15;
        If (nr2 > 9) then nr2 := nr2 + 7;   // handle ascii characters
        outstr [outpos] := Chr(nr2 + 48);
        inc (outpos);
    end;
End;

function ConIntHex (value: cardinal): string;
var
    reshex: string;
    serbin: string [6];
begin
    Move (value, serbin [1], 4);
    ByteSwaps (@serbin [1], 4);
    serbin [0] := chr(4);
    ConvHexStr (serbin, reshex);
    result := reshex;
end;


function StripQuotes (filename: string): string;
var
    delim: char;
    flen: integer;
begin
    result := filename;
    flen := length (filename);
    if flen < 2 then exit;
    delim := filename [1];
    if ((delim = SQUOTE) or (delim = DQUOTE)) then
    begin
        if (filename [flen] = delim) then
        begin
            if flen > 2 then
                result := copy (filename, 2, flen - 2)
            else
                result := '';
        end;
    end;
    if (delim = '<') then
    begin
        if (filename [flen] = '>') then
        begin
            if flen > 2 then
                result := copy (filename, 2, flen - 2)
```

```
            else
                result := '';
        end;
    end;
end;

function StripNewLines (const S: string): string;
var
    I: Integer;
begin
    result := S;
    if Length (result) = 0 then exit;
    for I := 1 to Length (result) do
    begin
        if (result [I] = CR) or (result [I] = LF) or  // Unicode
                                    (result [I] = TAB) then result [I] := Space;
    end;
end;

function IndexFiles (searchfile: string; mask: integer;
                    var FileList: TStringList; var totsize: cardinal): integer;
var
    SearchRec: TSearchRec;
    SearchResult: integer;
begin
    totsize := 0;
    result := 0;
    if NOT Assigned (FileList) then exit;
    try
        FileList.Clear;
        SearchResult := SysUtils.FindFirst (searchfile, mask, SearchRec);
        while SearchResult = 0 do
        begin
            if ((SearchRec.Attr and mask) = SearchRec.Attr) then
            begin
                if (SearchRec.Name <> '.') and
                            (SearchRec.Name <> '..') then
                begin
                    FileList.Add (SearchRec.Name);
                    inc (totsize, SearchRec.Size);
                end;
            end;
            SearchResult := SysUtils.FindNext (SearchRec);
        end;
        SysUtils.FindClose (SearchRec);
        FileList.Sort;
        result := FileList.Count;
    except
        SysUtils.FindClose (SearchRec);
        result := 0;
    end;
end;

function DeleteOldFiles (fname: string): integer;
var
    flist: TStringList;
    I: integer;
    totsize: cardinal;
begin
    result := 0;
    flist := TStringList.Create;
    try
        if IndexFiles (fname, faNormArch, flist, totsize) = 0 then exit;
        for I := 0 to Pred (flist.Count) do
        begin

        if SysUtils.DeleteFile (ExtractFilePath (fname) + flist [I]) then
                                    inc (result);
        end;
```

```
    finally
        flist.Free;
    end;
end;

function GetEnvirVar (name: string): string;
var
    Buffer: array[0..1023] of WideChar;
    len: integer;
    WideName: WideString;  // Unicode
begin
    WideName := name;
    len := GetEnvironmentVariableW (PWideChar (WideName),
    Buffer, Length (Buffer));
    SetString (Result, Buffer, len);
end;

function StripChars (AString, AChars: String): String;
var
    K: integer;
begin
    if Length (AChars) <> 0 then
    begin
        while Length (AString) <> 0 do
        begin
            K := Pos (AChars, AString);
            if K = 0 then break;
            Delete (AString, K, Length (AChars));
        end;
    end;
    result := AString;
end;

function StripCharsAnsi (AString, AChars: AnsiString): AnsiString;
var
    K: integer;
begin
    if Length (AChars) <> 0 then
    begin
        while Length (AString) <> 0 do
        begin
            K := PosAnsi (AChars, AString);
            if K = 0 then break;
            Delete (AString, K, Length (AChars));
        end;
    end;
    result := AString;
end;

function StripChar (const AString: String; const AChar: Char): String;
var
  Ch: Char;
  L, M: Integer;
  Source, Dest: PChar;
begin
  L := Length (AString);
  SetLength (Result, L);
  Source := Pointer (AString);
  Dest := Pointer (Result);
  M := 0;
  while L <> 0 do
  begin
    Ch := Source^;
    if AChar = #255 then
    begin
        if (Ch >= space) then
        begin
            Dest^ := Ch;
            Inc (Dest);
```

```
            Inc (M);
        end;
    end
    else
    begin
        if (Ch <> AChar) then
        begin
            Dest^ := Ch;
            Inc(Dest);
            Inc(M);
        end;
    end;
    Inc(Source);
    Dec(L);
  end;
  SetLength(Result, M);
end;

function StripCharAnsi (const AString: AnsiString; const AChar: AnsiChar):
AnsiString;
var
  Ch: AnsiChar;
  L, M: Integer;
  Source, Dest: PAnsiChar;
begin
  L := Length (AString);
  SetLength (Result, L);
  Source := Pointer (AString);
  Dest := Pointer (Result);
  M := 0;
  while L <> 0 do
  begin
    Ch := Source^;
    if AChar = #255 then
    begin
        if (Ch >= space) then
        begin
            Dest^ := Ch;
            Inc (Dest);
            Inc (M);
        end;
    end
    else
    begin
        if (Ch <> AChar) then
        begin
            Dest^ := Ch;
            Inc(Dest);
            Inc(M);
        end;
    end;
    Inc(Source);
    Dec(L);
  end;
  SetLength(Result, M);
end;

function StripSpaces (const AString: String): String;
begin
    result := StripChar (AString, space);
end;




function StripSpacesAnsi (const AString: AnsiString): AnsiString;
begin
    result := StripCharAnsi (AString, space);
```

```
end;

function StripCommas (const AString: String): String;
begin
    result := StripChar (AString, comma);
end;

function StripCommasAnsi (const AString: AnsiString): AnsiString;
begin
    result := StripCharAnsi (AString, comma);
end;

function StripNulls (const AString: String): String;
begin
    result := StripChar (AString, nulll);
end;

function StripNullsAnsi (const AString: AnsiString): AnsiString;
begin
    result := StripCharAnsi (AString, nulll);
end;

function StripAllCntls (const AString: String): String;
begin
    result := StripChar (AString, #255);
end;

function StripAllCntlsAnsi (const AString: AnsiString): AnsiString;
begin
    result := StripCharAnsi (AString, #255);
end;

procedure StringTranCh (var S: String; FrCh, ToCh: Char);  // Unicode
var
    L: Integer;
    Source: PChar;
begin
    UniqueString (S);
    L := Length (S);
    Source := Pointer (S);
    while L <> 0 do
    begin
        if (Source^ = FrCh) then Source^ := ToCh;
        Inc (Source);
        Dec (L);
    end;
end;

function StrCtrlSafe (const S: AnsiString): AnsiString;
begin
    result := S;
    StringCtrlSafe (result);
end;

function StrCtrlRest (const S: AnsiString): AnsiString;
begin
    result := S;
    StringCtrlRest (result);
end;

function StrFileTran (const S: String): String;
begin
    result := S;
    StringFileTran (result);
end;

function StrFileTranEx (const S: String): String;
begin
    result := S;
```

```
    StringFileTranEx (result);
end;

procedure UnixToDosPath (var S: String);
begin
    StringTranCh (S, '/','\');
end;

function UnxToDosPath (const S: String): String;
begin
    result := S;
    UnixToDosPath (result);
end;

procedure DosToUnixPath (var S: String);
begin
    StringTranCh (S, '\', '/');
end;

function DosToUnxPath (const S: String): String;
begin
    result := S;
    DosToUnixPath (result);
end;

function StringRemCntls (var S:String): boolean;
var
    L: Integer;
    Source: PChar;
begin
    result := false;
    UniqueString (S);
    L := Length (S);
    Source := Pointer (S);
    while L <> 0 do
    begin
        if (Source^ < space) then
        begin
            Source^ := space;
            result := true;
        end;
        Inc (Source);
        Dec (L);
    end;
end;

function StringRemCntlsEx (var S: String): boolean;
var
    L: Integer;
    Source: PChar;
begin
    result := false;
    UniqueString (S);
    L := Length (S);
    Source := Pointer (S);
    while L <> 0 do
    begin
        if (Source^ < space) then
        begin
            if (Source^ <> CR) and (Source^ <> LF) then
            begin
                Source^ := space;
                result := true;
            end;
        end;
        Inc (Source);
        Dec (L);
    end;
end;
```

```
Function PosPrev (const Find : AnsiString; const S : AnsiString;
const LastPos: Integer = 0) : Integer;
var I, J : Integer;
  Begin
    if Find = '' then
      begin
        Result := 0;
        exit;
      end;

    if LastPos = 0 then
      J := Length (S) - Length (Find) + 1 else
      J := LastPos - 1;
    For I := J downto 1 do
      if Match (Find, S, I) then
        begin
          Result := I;
          exit;
        end;

    Result := 0;
  End;

procedure StrArrayDelete (var S: StringArray; index: integer);
var
    I, tot: integer;
begin
    tot := Length (S);
    if (tot = 0) or (index >= tot) then exit;
    dec (tot);
    if tot > 0 then
    begin
        for I := index to Pred (tot) do S [I] := S [Succ(I)];
    end;
    SetLength (S, tot);
end;

procedure StrArrayInsert (var S: StringArray; index: integer; T: string);
var
    I, tot: integer;
begin
    tot := Length (S);
    SetLength (S, Succ(tot));
    if index > tot then index := tot;
    if (index < tot) and (tot <> 0) then
    begin
        for I := tot downto Succ (index) do S [I] := S [Pred(I)];
    end;
    S [index] := T;
end;

procedure StrArrayFromList (T: TStringList; var S: StringArray);
var
    I, tot: integer;
begin
    tot := T.Count;
    SetLength (S, tot);
    if tot = 0 then exit;
    for I := 0 to Pred (tot) do S [I] := T [I];
end;

procedure StrArrayToList (S: StringArray; var T: TStringList);
var
    I, tot: integer;
begin
    tot := Length (S);
    T.Clear;
    if tot = 0 then exit;
```

```
    for I := 0 to pred (tot) do T.Add (S [I]);
end;

function StrArrayPosOf (L: string; S: StringArray): integer;
var
    I, tot: integer;
begin
    tot := Length (S);
    result := -1;
    if tot = 0 then exit;
    for I := 0 to pred (tot) do
    begin
        if L = S [I] then
        begin
            result := I;
            exit;
        end;
    end;
end;

function StrArrayPosOfEx (const L: string; S: StringArray; T: integer): integer;
var
    I: integer;
begin
    if T > Length (S) then T := Length (S);
    result := -1;
    if T = 0 then exit;
    for I := 0 to pred (T) do
    begin
        if L = S [I] then
        begin
            result := I;
            exit;
        end;
    end;
end;

procedure StrArrayToMultiSZ (S: StringArray; var Buffer: PAnsiChar);
var
    I, tot, size: integer;
    P: PAnsiChar;
begin
    tot := Length (S);
    size := 2;
    if tot > 0 then
    begin
        for I := 0 to Pred (tot) do inc (size, Length (S [I]) + 1);
    end;
    GetMem (Buffer, size);
    P := Buffer;
    if tot > 0 then
    begin
        for I := 0 to Pred (tot) do
        begin
            LstrcpyA (P, PAnsiChar (S [I]));
            inc (P, LstrlenA (P) + 1);
        end;
    end;
    P^ := #0;
    inc (P);
    P^ := #0;
end;

procedure StrArrayToMultiSZ (S: StringArray; var Buffer: PWideChar);
var
    I, tot, size: integer;
    P: PWideChar;
    W: WideString;
begin
```

```
    tot := Length (S);
    size := 2;
    if tot > 0 then
    begin
        for I := 0 to Pred (tot) do inc (size, Length (S [I]) + 1);
    end;
    GetMem (Buffer, size);
    P := Buffer;
    if tot > 0 then
    begin
        for I := 0 to Pred (tot) do
        begin
            W := S [I];
            LstrcpyW (P, PWideChar (W));
            inc (P, LstrlenW (P) + 1);
        end;
    end;
    P^ := #0;
    inc (P);
    P^ := #0;
end;

procedure StrArrayFromMultiSZ (Buffer: PAnsiChar; Len: integer; var S:
StringArray);
var
    I, tot: integer;
    P: PAnsiChar;
begin
    tot := 0;
    if Len > 0 then
    begin
        P := Buffer;
        for I := 1 to Len do
        begin
            if P^ = #0 then inc (tot);  // count strings
            inc (P);
        end;
    end;
    SetLength (S, tot);
    if tot = 0 then exit;
    P := Buffer;
    tot := 0;
    while P^ <> #0 do
    begin
        S [tot] := P;
        inc (tot);
        inc (P, lstrlenA (P) + 1);
    end;
    SetLength (S, tot);
end;

procedure StrArrayFromMultiSZ (Buffer: PWideChar; Len: integer; var S:
StringArray);
var
    I, tot: integer;
    P: PWideChar;
begin
    tot := 0;
    if Len > 0 then
    begin
        P := Buffer;
        for I := 1 to Len do
        begin
            if P^ = #0 then inc (tot);  // count strings
            inc (P);
        end;
    end;
    SetLength (S, tot);   // might include end nulls
    if tot = 0 then exit;
```

```
    P := Buffer;
    tot := 0;
    while P^ <> #0 do
    begin
        S [tot] := P;
        inc (tot);
        inc (P, lstrlenW (P) + 1);
    end;
    SetLength (S, tot);
end;

function FileTimeToInt64 (const FileTime: TFileTime): Int64;
begin
    Move (FileTime, result, SizeOf (result));
end;

function Int64ToFileTime (const FileTime: Int64): TFileTime;
begin
    Move (FileTime, result, SizeOf (result));
end;

function FileTimeToDateTime(const FileTime: TFileTime): TDateTime;
begin
    Result := FileTimeToInt64 (FileTime) / FileTimeStep;
    Result := Result + FileTimeBase;
end;

function CheckFileOpen(const FName: String): integer;
var
    H: Integer;
begin
    result := -1;   // file not found
    if NOT FileExists (FName) then exit;
    H := FileOpen(FName, fmOpenReadWrite);
    result := 1;   // file open
    if H < 0 then exit;
    FileClose(H);
    result := 0;   // file found but closed
end;

function TruncateFile(const FName: String; NewSize: int64): int64;
var
    H: Integer;
begin
    result := -1;   // file not found
    if NOT FileExists (FName) then exit;
    H := FileOpen(FName, fmOpenReadWrite);
    if H < 0 then exit;
    result := FileSeek (H, Int64 (0), soFromEnd);   // size of file
    if NewSize < result then
    begin
        result := FileSeek (H, NewSize, soFromBeginning);
        if result >= 0 then SetEndOfFile (H);
    end;
    FileClose(H);
end;

function FileTimeToSecs2K (const FileTime: TFileTime): integer;
begin
    result := (FileTimeToInt64 (FileTime) - FileTime2000) div FileTimeSecond;
end;

function DateTimeToFileTime(DateTime: TDateTime): TFileTime;
var
  E: Extended;
begin
  E := (DateTime - FileTimeBase) * FileTimeStep;
  result := Int64ToFileTime (Round(E));
end;
```

```
function GetFUAgeSizeFile (filename: string; var FileTime: TFileTime;
                                             var FSize: Int64): boolean;
var
   SResult: integer;
   SearchRec: TSearchRec;
   TempSize: TULargeInteger;
begin
   Result := false;
SResult := SysUtils.FindFirst(filename, faAnyFile, SearchRec);
   if SResult = 0 then
   begin
        TempSize.LowPart := SearchRec.FindData.nFileSizeLow;
        TempSize.HighPart := SearchRec.FindData.nFileSizeHigh;
        FSize := TempSize.QuadPart;
        FileTime := SearchRec.FindData.ftLastWriteTime;
        result := true;
   end;
   SysUtils.FindClose(SearchRec);
end;

function GetFAgeSizeFile (filename: string; var FileTime: TFileTime;
                                             var FSize: Int64): boolean;
var
   UTCFileTime: TFileTime;
begin
   Result := GetFUAgeSizeFile (filename, UTCFileTime, FSize);
   if Result then FileTimeToLocalFileTime (UTCFileTime, FileTime);
end;

function GetUAgeSizeFile (filename: string; var FileDT: TDateTime;
                                             var FSize: Int64): boolean;
var
   UTCFileTime: TFileTime;
begin
   Result := GetFUAgeSizeFile (filename, UTCFileTime, FSize);
   if Result then FileDT := FileTimeToDateTime (UTCFileTime);
end;

function TrimSpRight(const S: string): string;
var
  I: Integer;
begin
  I := Length(S);
  while (I > 0) and (S[I] = ' ') do Dec(I);
  Result := Copy(S, 1, I);
end;

function ExtractNameOnly(FileName: string): string;
var
  I: Integer;
begin
  FileName := ExtractFileName (FileName);  // remove path
  I := Length(FileName);
  while (I > 0) and not (IsPathSep (FileName[I])) do Dec(I);  // Unicode
  if (I = 0) or (FileName[I] <> '.') then I := MaxInt;
  Result := Copy(FileName, 1, I - 1);
end;

function GetExceptMess (ExceptObject: TObject): string;
var
  MsgPtr: PChar;
  MsgEnd: PChar;
  MsgLen: Integer;
  MessEnd: String;
begin
  MsgPtr := '';
  MsgEnd := '';
  if ExceptObject is Exception then
```

```
  begin
    MsgPtr := PChar(Exception(ExceptObject).Message);
    MsgLen := StrLen(MsgPtr);
    if (MsgLen <> 0) and (MsgPtr[MsgLen - 1] <> '.') then MsgEnd := '.';
  end;
  result := Trim (MsgPtr);
  MessEnd := Trim (MsgEnd);
  if Length (MessEnd) > 5 then result := result + ' - ' + MessEnd;
end;

function AscToInt64Ansi (value: AnsiString): Int64;
var
    E: Integer;
begin
    Val (value, result, E);
end;

function Str2LInt (const S: String): LongInt;
begin
    result := AscToInt (Trim (S));
end;

function Str2Byte (const S: String): Byte;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxByte then
        Result := MaxByte
    else if L < MinByte then
        Result := MinByte
    else
        Result := L;
end;

function Str2SInt (const S: String): ShortInt;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxShortInt then
        Result := MaxShortInt
    else if L < MinShortInt then
        Result := MinShortInt
    else
        Result := L;
end;

function Str2Int (const S: String): Integer;
begin
    result := Str2LInt (S);
end;


function Str2Word (const S: String): Word;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxWord then
        Result := MaxWord
    else if L < MinWord then
        Result := MinWord
    else
        Result := L;
end;

function AddThouSeps (const S: string): string;
var
```

```
    LS, L2, I, N: Integer;
    Temp : string;
begin
    result := S;
    LS := Length (S);
    N := 1;
    if LS > 1 then
    begin
        if S [1] = '-' then
        begin
            N := 2;
            LS := LS - 1;
        end;
    end;
    if LS <= 3 then exit;
    L2 := (LS - 1) div 3;
    Temp := '';
    for I := 1 to L2 do
Temp := ThousandSeparator + Copy (S, LS - 3 * I + 1, 3) + Temp;
    Result := Copy (S, N, (LS - 1) mod 3 + 1) + Temp;
    if N > 1 then Result := '-' + Result;
end;

function IntToCStr (const N: integer): string;
begin
    result := AddThouSeps (IntToStr (N));
end;

function Int64ToCStr (const N: int64): string;
begin
    result := AddThouSeps (IntToStr (N));
end;

function AddThouSepsAnsi (const S: AnsiString): AnsiString;
var
    LS, L2, I, N: Integer;
    Temp : AnsiString;
begin
    result := S;
    LS := Length (S);
    N := 1;
    if LS > 1 then
    begin
        if S [1] = '-' then
        begin
            N := 2;
            LS := LS - 1;
        end;
    end;
    if LS <= 3 then exit;
    L2 := (LS - 1) div 3;
    Temp := '';
    for I := 1 to L2 do
Temp := ThousandSeparator + Copy (S, LS - 3 * I + 1, 3) + Temp;
    Result := Copy (S, N, (LS - 1) mod 3 + 1) + Temp;
    if N > 1 then Result := '-' + Result;
end;

function IntToCStrAnsi (const N: integer): AnsiString;
begin
    result := AddThouSepsAnsi (IntToStrAnsi (N));
end;

function Int64ToCStrAnsi (const N: int64): AnsiString;
begin
    result := AddThouSepsAnsi (IntToStrAnsi (N));
end;

function LInt2Str (const L: LongInt; const Len: Byte): String;
```

```
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2EStr (const L: LongInt): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
end;

function LInt2ZBEStr (const L: LongInt): String;
begin
    if L = 0 then
        Result := ''
    else
        try
            Result := IntToStr (L);
        except
            Result := '';
    end;
end;

function FillStr (const Ch : Char; const N : Integer): string;
var
  I: integer;
begin
    SetLength (Result, N);
    for I := 1 to N do Result [I] := Char (Ch);
end;

function BlankStr (const N : Integer): string;
begin
    Result := FillStr (' ', N);
end;

function PadRightStr (const S : string; const Len : Integer): string;
var
    N: Integer;
begin
    N := Length (S);
    if N < Len then
        Result := S + BlankStr (Len - N)
    else
        Result := S;
end;

function PadLeftStr (const S : string; const Len : Integer): string;
var
    N: Integer;
begin
    N := Length (S);
    if N < Len then
        Result := BlankStr (Len - N) + S
    else
        Result := S;
end;

function PadChLeftStr (const S : string; const Ch : Char; const Len : Integer):
string;
var
    N: Integer;
```

```
begin
    N := Length (S);
    if N < Len then
        Result := FillStr (Ch, Len - N) + S
    else
        Result := S;
end;

function Int2StrZ (const L: LongInt; const Len: Byte): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), '0', Len);
end;

function Byte2Str (const L: LongInt; const Len: Byte): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2ZBStr (const L: LongInt; const Len: Byte): String;
begin
    Result := LInt2ZBEStr (L);
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2CStr (const L : LongInt; const Len : Byte): string;
begin
    Result := LInt2CEStr (L);
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2CEStr (const L : LongInt): string;
begin
    try
        Result := AddThouSeps (IntToStr (L));
    except
        Result := '';
    end;
end;

function Int642CEStr (const L : Int64): string;
begin
    try
        Result := AddThouSeps (IntToStr (L));
    except
        Result := '';
    end;
end;

function Packed2Date (info: string): TDateTime;
var
    yy, mm, dd: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 8 then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 5, 2));
    dd := Str2Word (copy (info, 7, 2));
```

```
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
    if length (info) < 15 then exit;
    if info [9] <> '-' then exit;
    timeDT := Packed2Time (copy (info, 10, 10));
    if timeDT < 0 then exit;
    result := result + timeDT;
end;

function PackedISO2Date (info: string): TDateTime;
var
    yy, mm, dd: word;
    hh, nn, ss: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 10 then exit;
    if info [5] <> '-' then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 6, 2));
    dd := Str2Word (copy (info, 9, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
    if length (info) <> 19 then exit;
    if info [14] <> ':' then exit;
    hh := Str2Word (copy (info, 12, 2));
    nn := Str2Word (copy (info, 15, 2));
    ss := Str2Word (copy (info, 18, 2));
    if NOT TryEncodeTime (hh, nn, ss, 0, timeDT) then exit;
    result := result + timeDT;
end;

function PackedISO2UKStr (info: string): string;
begin
    result := '';
    info := trim (info);
    if length (info) < 10 then exit;
    if info [5] <> '-' then exit;
    result := copy (info, 9, 2) + '/' +copy(info, 6, 2)+'/'+copy(info, 1, 4);
    if length (info) <> 19 then exit;
    if info [14] <> ':' then exit;
    result := result + ' ' + copy (info, 12, 2) + ':' + copy (info, 15, 2);
    if copy (info, 18, 2) <> '00' then
    result := result + ':' + copy (info, 18, 2);
end;

function Packed2Secs (info: string): integer;
var
    len: integer;
begin
    result := 0;
    info := trim (info);
    len := length (info);
    if len < 4 then exit;
    while length (info) < 8 do info := '0' + info;
    if info [6] <> TimeSeparator then exit;
    result := AscToInt (copy (info, 1, 2)) * 60;
    result := (result + AscToInt (copy (info, 4, 2))) * 60;
    result := result + AscToInt (copy (info, 7, 2));
end;

function ConvLongDate (info: string): TDateTime;
```

```
var
    yy, mm, dd: word;
begin
    result := 0;
    info := trim (info);
    if length (info) <> 10 then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 6, 2));
    dd := Str2Word (copy (info, 9, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
end;

function ConvUSADate (info: string): TDateTime;
var
    yy, mm, dd: word;
begin
    result := 0;
    info := trim (info);
    if length (info) <> 10 then exit;
    yy := Str2Word (copy (info, 7, 4));
    mm := Str2Word (copy (info, 1, 2));
    dd := Str2Word (copy (info, 4, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then result := 0;
end;

function ConvUKDate (info: string): TDateTime;
var
    yy, mm, dd: word;
    hh, nn, ss: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 10 then exit;
    if info [3] <> '/' then exit;
    yy := Str2Word (copy (info, 7, 4));
    mm := Str2Word (copy (info, 4, 2));
    dd := Str2Word (copy (info, 1, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := 0;
        exit;
    end;
    if length (info) < 16 then exit;
    if info [14] <> ':' then exit;
    hh := Str2Word (copy (info, 12, 2));
    nn := Str2Word (copy (info, 15, 2));
    ss := 0;
    if length (info) >= 19 then ss := Str2Word (copy (info, 18, 2));
    if NOT TryEncodeTime (hh, nn, ss, 0, timeDT) then exit;
    result := result + timeDT;
end;

function TestTrgTick (const TrgTick: longword): boolean;
var
    curtick: longword;
begin
    result := false;
    if TrgTick = TriggerDisabled then exit;
disabled
    if TrgTick = TriggerImmediate then
    begin
        result := true;
        exit;
    end;
```

```
    curtick := GetTickCountX;
    if curtick <= MaxInteger then
    begin
        if curtick >= TrgTick then result := true;
        exit;
    end;
    if TrgTick <= MaxInteger then exit;
    if curtick >= TrgTick then result := true;
end;

procedure FreeAndNilEx(var Obj);
var
  Temp: TObject;
begin
  if Pointer(Obj) = Nil then exit;
  Temp := TObject(Obj);
  Pointer(Obj) := nil;
  Temp.Free;
end;

end.
```

ФАЙЛ МОДУЛЮ UNIT1.PAS

```
unit Unit1; // об'ява модулю

interface

uses
  Windows, SMART, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids;

type
  TForm1 = class (TForm)
    StringGrid1: TStringGrid;
    ComboBox1: TComboBox;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox2: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label7: TLabel;
    GroupBox3: TGroupBox;
    Label9: TLabel;
    Label8: TLabel;
    procedure DoPrintData (pAttrBuffer: PCHAR; pThrsBuffer: PCHAR);
    procedure FormCreate (Sender: TObject);
    procedure FormClose (Sender: TObject; var Action: TCloseAction);
    procedure ComboBox1Change (Sender: TObject);
  private
    {Private declarations}
  public
    {Public declarations}
  end;

var
  Form1: TForm1;
  pAttrNames: TStringList;
  CurrentHandle: THandle;
  OSVersionInfo: TOSVersionInfo;

// Визначення глобальних буферів

AttrOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) +
(READ_ATTRIBUTE_BUFFER_SIZE-1)] of BYTE;
ThreshOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) +
(READ_THRESHOLD_BUFFER_SIZE-1)] of BYTE;
IdOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) + (IDENTIFY_BUFFER_SIZE-1)]
of BYTE;

implementation

{$ R *. Dfm}

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду IDENTIFY
// * BDriveNum = 0-3
// * BIDCmd = IDE_ID_FUNCTION або IDE_ATAPI_ID
// ********************************************** ***************************

function DoIDENTIFY (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
pSCOP: PSENDCMDOUTPARAMS; bIDCmd: BYTE; bDriveNum: BYTE): BOOL;
var
lpcbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = IDENTIFY_BUFFER_SIZE;
```

```
pSCIP.irDriveRegs.bFeaturesReg: = 0;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = 0;
pSCIP.irDriveRegs.bCylHighReg: = 0;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = bIDCmd;
// Команда може ідентифікувати IDE або ATAPI.
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду SMART_ENABLE_SMART_OPERATIONS
// * BDriveNum = 0-3
// ********************************************** ***************************

function DoEnableSMART (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS; pSCOP:
PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
lpcbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = 0;
pSCIP.irDriveRegs.bFeaturesReg: = SMART_ENABLE_SMART_OPERATIONS;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = SMART_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_SMART_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду SMART_READ_ATTRIBUTE_VALUES
// * BDriveNum = 0-3
// ********************************************** ***************************

function DoReadAttributesCmd (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
cbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = READ_ATTRIBUTE_BUFFER_SIZE;
pSCIP.irDriveRegs.bFeaturesReg: = SMART_READ_ATTRIBUTE_VALUES;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = SMART_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_SMART_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду SMART_READ_ATTRIBUTE_THRESHOLDS
// * BDriveNum = 0-3

// ********************************************** ***************************
function DoReadThresholdsCmd (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
```

```
var
cbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = READ_THRESHOLD_BUFFER_SIZE;
pSCIP.irDriveRegs.bFeaturesReg: = SMART_READ_ATTRIBUTE_THRESHOLDS;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = SMART_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_SMART_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: = DeviceIoControl (hSMARTIOCTL, DFP_RECEIVE_DRIVE_DATA, pSCIP,
sizeof (SENDCMDINPARAMS) -1, pSCOP, sizeof (SENDCMDOUTPARAMS) +
READ_THRESHOLD_BUFFER_SIZE-1, cbBytesReturned, nil);
end;

// ********************************************** ***************************
// * DoPrintData
// *
// * FUNCTION: Відображає атрибути та порогові значення SMART
// ********************************************** ***************************

procedure TForm1.DoPrintData (pAttrBuffer: PCHAR; pThrsBuffer: PCHAR);
var
i: integer;
pDA: PDRIVEATTRIBUTE;
pAT: PATTRTHRESHOLD;
begin
Label8.Caption: = 'Версія структури атрибутів:' + inttostr (WORD (pAttrBuffer
[0]));
Label9.Caption: = 'Версія структури порогових значень атрибутів:' + inttostr
(WORD (pThrsBuffer [0]));
// Виводимо інформацію: ідентифікатор і назву
// атрибута, його поточне і граничне значення

pDA: = PDRIVEATTRIBUTE (@pAttrBuffer [2]);
pAT: = PATTRTHRESHOLD (@pThrsBuffer [2]);
for i: = 0 to NUM_ATTRIBUTE_STRUCTS-1 do
begin
StringGrid1.Rows [i +1]. Strings [0]: = inttostr (pDA.bAttrID);
if (pDA.bAttrID = 194) or (pDA.bAttrID = 231) then Label7.Caption: =
'Температура:' + inttostr ((84 - (pDA.bAttrValue-1) div 3)) + # 176 + 'C';
StringGrid1.Rows [i +1]. Strings [1]: = pAttrNames [pDA.bAttrID];
StringGrid1.Rows [i +1]. Strings [2]: = inttostr (pDA.bAttrValue);
StringGrid1.Rows [i +1]. Strings [3]: = inttostr (pAT.bWarrantyThreshold);
StringGrid1.Rows [i +1]. Strings [4]: = inttostr (pDA.bWorstValue);
inc (pDA);
inc (pAT);
end;
end;

// Міняємо WORD-масив на BYTE-масив
procedure ChangeByteOrder (szString: PCHAR; uscStrSize: USHORT);
var
i: USHORT;
temp: CHAR;
begin
i: = 0;
while i <uscStrSize do
begin
temp: = szString [i];
szString [i]: = szString [i +1];
szString [i +1]: = temp;
i: = i + 2;
end;
end;
```

```
// ------------------------------------------------ ---------------------
// Відкриваємо дескриптор (хендл) SMART для операцій за допомогою
DeviceIoControl.
// ------------------------------------------------ ---------------------

function OpenSMART (DrvNum: Byte): THandle;
var
hSMARTIOCTL: THandle;
begin
if OSVersionInfo.dwPlatformId = VER_PLATFORM_WIN32_NT then
begin
hSMARTIOCTL: = CreateFile (PChar ('\ \. \ PhysicalDrive' + inttostr (DrvNum)),
GENERIC_READ or GENERIC_WRITE, FILE_SHARE_READ or FILE_SHARE_WRITE, nil,
OPEN_EXISTING, 0,0);
end
else // Windows 95 OSR2, Windows 98
begin
hSMARTIOCTL: = CreateFile ('\ \. \ SMARTVSD', 0,0, nil, CREATE_NEW, 0,0);
if hSMARTIOCTL = INVALID_HANDLE_VALUE then
ShowMessage ('Неможливо відкрити SMARTVSD, код помилки:' + inttostr
(GetLastError) + '-' + SysErrorMessage (GetLastError))
end;
result: = hSMARTIOCTL;
end;

// ********************************************** ************************
// * DisplayIdInfo
// *
// * Відображає вміст ID буфера
// ********************************************** ************************
procedure DisplayIdInfo (pids: PIDSECTOR; bIDCmd: BYTE; bDriveNum: BYTE);
begin
if bIDCmd = IDE_ID_FUNCTION then
begin
Form1.Label5.Caption: = 'Диск' + inttostr (bDriveNum) + 'є IDE пристроєм, який
підтримує SMART';
Form1.Label6.Caption: = 'Циліндрів:' + inttostr (pids.wNumCyls) + '; голівок:' +
inttostr (pids.wNumHeads) + '; Секторів на доріжку:' + inttostr
(pids.wSectorsPerTrack);
end
else Form1.Label5.Caption: = 'Диск' + inttostr (bDriveNum) + 'є ATAPI
пристроєм.';
end;

function GetVersionSMART (hSMARTIOCTL: THandle): TGetVersionOutParams;
var
VersionParams: TGetVersionOutParams;
cbBytesReturned: DWORD;
begin
ZeroMemory (@VersionParams, sizeof (TGetVersionOutParams));
if not then ShowMessage (SysErrorMessage (GetLastError));
result: = VersionParams;
end;

procedure TForm1.FormCreate (Sender: TObject);
var
hSMARTIOCTL: THandle;
i: integer;
VersionParams: TGetVersionOutParams;
bIDCmd, bDfpDriveMap: BYTE;
// Команда ідентифікації IDE або ATAPI
scip: TSendCmdInParams;
OutCmd: TSendCmdOutParams;
bSuccess: bool;
pids: PIDSECTOR;
begin
StringGrid1.ColWidths [0]: = 30;
StringGrid1.Cols [0]. Strings [0]: = 'ID';
StringGrid1.ColWidths [1]: = 260;
```

```
StringGrid1.Cols [1]. Strings [0]: = 'Назва атрибуту';
StringGrid1.ColWidths [2]: = 130;
StringGrid1.Cols [2]. Strings [0]: = 'Поточне значення';
StringGrid1.ColWidths [3]: = 130;
StringGrid1.Cols [3]. Strings [0]: = 'Граничне значення';
StringGrid1.ColWidths [4]: = 130;
StringGrid1.Cols [4]. Strings [0]: = 'Найгірше значення';
pAttrNames: = TStringList.Create;
pAttrNames.LoadFromFile (ExtractFilePath (Application.ExeName) +
'attributes.txt');

OSVersionInfo.dwOSVersionInfoSize: = SizeOf (OSVersionInfo);
GetVersionEx (OSVersionInfo);
bDfpDriveMap: = 0;
CurrentHandle: = OpenSMART (0);
// Отримуємо версію і т.п. SMART IOCTL
VersionParams: = GetVersionSMART (CurrentHandle);

case VersionParams.fCapabilities of
CAP_IDE_ID_FUNCTION: Label3.Caption: = 'Підтримується команда ATA ID';
CAP_IDE_ATAPI_ID: Label3.Caption: = 'Підтримується команда ATAPI ID';
CAP_IDE_EXECUTE_SMART_FUNCTION: Label3.Caption: =
'Підтримуються команди SMART';
CAP_IDE_ID_FUNCTION or CAP_IDE_ATAPI_ID or CAP_IDE_EXECUTE_SMART_FUNCTION:
Label3.Caption: =
'Підтримуються команди SMART, ATA ID і ATAPI ID';
CAP_IDE_ID_FUNCTION or CAP_IDE_EXECUTE_SMART_FUNCTION: Label3.Caption: =
'Підтримуються команди SMART і ATA ID ';
CAP_IDE_ATAPI_ID or CAP_IDE_EXECUTE_SMART_FUNCTION: Label3.Caption: =
'Підтримуються команди SMART і ATAPI ID ';
CAP_IDE_ID_FUNCTION or CAP_IDE_ATAPI_ID: Label3.Caption: =
'Підтримуються команди ATA ID і ATAPI ID';
end;
for i: = 0 to MAX_IDE_DRIVES-1 do
begin
hSMARTIOCTL: = OpenSMART (i);
// Якщо пристрій з номером "i" - IDE, передаємо йому команди.
if VersionParams.bIDEDeviceMap shr i and 1 = 1 then
begin
// Ігноруємо ATAPI-пристрої.
if VersionParams.bIDEDeviceMap shr i and $ 10 = 0 then
begin
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@OutCmd, sizeof (OutCmd));
// Намагаємося активувати SMART.
if DoEnableSMART (hSMARTIOCTL, @scip, @OutCmd, i) then
begin
Label4.Caption: = Label4.Caption + '' + inttostr (i) + ';';
// Позначаємо диск, як накопичувач з активним SMART
bDfpDriveMap: = bDfpDriveMap or (1 shl i);

// Тепер отримуємо ID сектора для всіх пристроїв IDE в системі.
// якщо пристрій - ATAPI, використовуємо команду IDE_ATAPI_ID,
// в іншому випадку використовуємо команду IDE_ID_FUNCTION.
if VersionParams.bIDEDeviceMap shr i and $ 10 = 1 then bIDCmd: = IDE_ATAPI_ID
else bIDCmd: = IDE_ID_FUNCTION;
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@IdOutCmd, sizeof (IdOutCmd));
if DoIDENTIFY (hSMARTIOCTL, @scip, PSENDCMDOUTPARAMS (@IdOutCmd), bIDCmd, i)
then
begin
pids: = @PSENDCMDOUTPARAMS (@IdOutCmd). bBuffer;
if bIDCmd = IDE_ID_FUNCTION then ComboBox1.Items.Add (inttostr (i));
ChangeByteOrder (pids.sModelNumber, sizeof (pids.sModelNumber));
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sModelNumber;
ChangeByteOrder (pids.sFirmwareRev, sizeof (pids.sFirmwareRev));
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sFirmwareRev;
```

```
ChangeByteOrder (pids.sSerialNumber, sizeof (pids.sSerialNumber));
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sSerialNumber;
end
else ShowMessage ('Команда Identify не виконана на диску:' + inttostr (i) + # 10
# 13 + 'DriverStatus: bDriverError =' + inttostr (PSENDCMDOUTPARAMS (@IdOutCmd).
DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (PSENDCMDOUTPARAMS
(@IdOutCmd). DriverStatus.bIDEStatus));
end
else ShowMessage ('Команда запуску SMART не виконана, диск:' + inttostr (i) + #
10 # 13 + 'DriverStatus: bDriverError =' + inttostr
(OutCmd.DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (OutCmd.
DriverStatus.bIDEStatus));
end;
end;
end;
// Перебираємо всі можливі IDE-приводи і посилаємо
// команди тим, які підтримують SMART.
for i: = 0 to MAX_IDE_DRIVES-1 do
begin
if bDfpDriveMap shr i and 1 = 1 then
begin
ZeroMemory (@AttrOutCmd, sizeof (AttrOutCmd));
ZeroMemory (@ThreshOutCmd, sizeof (ThreshOutCmd));
bSuccess: = DoReadAttributesCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@AttrOutCmd), i);
if bSuccess = false then ShowMessage ('Помилка при виконанні команди читання
атрибутів SMART на диску:' + inttostr (i) + # 10 # 13 + 'DriverStatus:
bDriverError =' + inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd).
DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (PSENDCMDOUTPARAMS
(@AttrOutCmd). DriverStatus.bIDEStatus))
// Команда читання атрибутів виконана успішно.
// Намагаємося прочитати порогові значення атрибутів.
else
if not DoReadThresholdsCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@ThreshOutCmd), i) then ShowMessage ('Помилка при виконанні команди читання
порогових значень атрибутів SMART на диску:' + inttostr (i) + # 10 # 13 +
'DriverStatus: bDriverError = '+ inttostr (PSENDCMDOUTPARAMS (@ThreshOutCmd).
DriverStatus.bDriverError) +', bIDEStatus = '+ inttostr (PSENDCMDOUTPARAMS
(@ThreshOutCmd). DriverStatus.bIDEStatus));
// Якщо функції DoReadAttributesCmd і DoReadThresholdsCmd виконані успішно,
// Процедура DoPrintData виводить обидва
// Значення атрибутів. Якщо DoReadThresholdsCmd
// Не підтримується, виводяться тільки поточні значення атрибутів
if bSuccess <> false then DoPrintData (@PSENDCMDOUTPARAMS (@AttrOutCmd).
bBuffer, @PSENDCMDOUTPARAMS (@ThreshOutCmd). bBuffer);
end;
end;
DisplayIdInfo (@PSENDCMDOUTPARAMS (@IdOutCmd). BBuffer, IDE_ID_FUNCTION, 0);
ComboBox1.ItemIndex: = 0;
end;

procedure TForm1.FormClose (Sender: TObject; var Action: TCloseAction);
begin
pAttrNames.Free;
// Закриваємо дескриптор (хендл) SMART.
CloseHandle (CurrentHandle);
end;

procedure TForm1.ComboBox1Change (Sender: TObject);
var
Num: string [1];
scip: TSendCmdInParams;
bSuccess: bool;
begin
Num: = ComboBox1.Items.Strings [ComboBox1.ItemIndex];
CurrentHandle: = OpenSMART (strtoint (Num));
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@IdOutCmd, sizeof (IdOutCmd));
```

```
DoIDENTIFY (CurrentHandle, @scip, PSENDCMDOUTPARAMS (@IdOutCmd),
IDE_ID_FUNCTION, strtoint (Num));
ZeroMemory (@AttrOutCmd, sizeof (AttrOutCmd));
ZeroMemory (@ThreshOutCmd, sizeof (ThreshOutCmd));
bSuccess: = DoReadAttributesCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@AttrOutCmd), strtoint (Num));
if bSuccess = false then ShowMessage ('Помилка при виконанні команди читання
атрибутів SMART на диску:' + Num + # 10 # 13 + 'DriverStatus: bDriverError =' +
inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd). DriverStatus.bDriverError) + ',
bIDEStatus = '+ inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd).
DriverStatus.bIDEStatus))
// Команда читання атрибутів виконана успішно.
// Намагаємося прочитати порогові значення атрибутів.
else
if not DoReadThresholdsCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@ThreshOutCmd), strtoint (Num)) then ShowMessage ('Помилка при виконанні
команди читання порогових значень атрибутів SMART на диску:' + Num + # 10 # 13 +
'DriverStatus: bDriverError =' + inttostr (PSENDCMDOUTPARAMS (@ThreshOutCmd).
DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (PSENDCMDOUTPARAMS
(@ThreshOutCmd). DriverStatus.bIDEStatus));
// Якщо функції DoReadAttributesCmd і DoReadThresholdsCmd виконані успішно,
// процедура DoPrintData виводить обидва значення атрибутів. Якщо
DoReadThresholdsCmd
// не підтримується, виводяться тільки поточні значення атрибутів
if bSuccess <> false then DoPrintData (@PSENDCMDOUTPARAMS (@AttrOutCmd).
bBuffer, @PSENDCMDOUTPARAMS (@ThreshOutCmd). bBuffer);
DisplayIdInfo (@PSENDCMDOUTPARAMS (@IdOutCmd). BBuffer, IDE_ID_FUNCTION,
strtoint (Num));
end;

// Об'єм жорсткого диска
function TForm1.GetTotalSpaceMb (Root: string): string;
var
  free, total: int64;
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = total div 1000000000;
   Mb: = (total mod 1000000000) div 1000000;
   Kb: = (total mod 1000000) div 1000;
   b: = total mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Обсяг вільного місця жорсткого диска
function TForm1.GetFreeSpaceMb (Root: string): string;
var
  free, total: int64;
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = free div 1000000000;
   Mb: = (free mod 1000000000) div 1000000;
   Kb: = (free mod 1000000) div 1000;
   b: = free mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Обсяг зайнятого місця жорсткого диска
function TForm1.GetFullSpaceMb (Root: string): string;
var
  free, total: int64;
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = (total - free) div 1000000000;
```

```
   Mb: = ((total - free) mod 1000000000) div 1000000;
   Kb: = ((total - free) mod 1000000) div 1000;
   b: = (total - free) mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Як визначити тип файлової системи на вказаному диску
function TForm1.GetHDDFileSystem (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = FSName;
end;

// Мітка тому на вказаному диску
function TForm1.GetHDDLabel (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = VolumeName;
end;

// Зміна мітки томи на вказаному диску
// Перший аргумент - те, мітку якого потрібно змінити
// Другий аргумент - нова мітка
// SetVolumeLabel (PChar ('c: \'), PChar ('Win98'));

// Серійний номер тому
function TForm1.GetHDDSerialNumber (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = Format ('% .8 x', [SerialNum]);
end;

// Загальна кількість кластерів на вказаному диску
function GetTotalNumberClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
```

```
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = TotalNumberClusters;
end;

// Кількість вільних кластерів на вказаному диску
function GetNumberOfFreeClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = NumberOfFreeClusters;
end;

// Кількість зайнятих кластерів на вказаному диску
function GetNumberOfFullClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = TotalNumberClusters - NumberOfFreeClusters;
end;

end.
```

ФАЙЛ МОДУЛЮ UNIT2.PAS

```
unit Unit2; // об'ява модулю

interface

uses windows;

type
   USHORT = Word;

const
MAX_IDE_DRIVES = 4;
// Максимальне число дисків, що приймають
// Primary / secondary, master / slave топологію

READ_ATTRIBUTE_BUFFER_SIZE = 512;
IDENTIFY_BUFFER_SIZE = 512;
READ_THRESHOLD_BUFFER_SIZE = 512;

// IOCTL команди
DFP_GET_VERSION = $ 00074080;
DFP_SEND_DRIVE_COMMAND = $ 0007C084;
DFP_RECEIVE_DRIVE_DATA = $ 0007C088;

// ------------------------------------------------- ------------
// GETVERSIONOUTPARAMS містить дані, що повертаються
// Функцією Get Driver Version.
// ------------------------------------------------- ------------
type
   TGetVersionOutParams = packed record
      bVersion: BYTE; // Бінарна версія драйвера.
      bRevision: BYTE; // Бінарна підверсія драйвера.
      bReserved: BYTE; // Не використовується.
      bIDEDeviceMap: BYTE; // Бітовий масив IDE-пристроїв.
      fCapabilities: DWORD; // Бітова маска можливостей драйвера.
      dwReserved: array [0 .. 3] of DWORD; // Зарезервовано для майбутнього
використання.
   end;
   GETVERSIONOUTPARAMS = TGetVersionOutParams;
   PGetVersionOutParams = ^ TGetVersionOutParams;

// Біти, що повертаються в fCapabilities структури GETVERSIONOUTPARAMS
const
CAP_IDE_ID_FUNCTION = 1; // Підтримується команда ATA ID
CAP_IDE_ATAPI_ID = 2; // Підтримується команда ATAPI ID
CAP_IDE_EXECUTE_SMART_FUNCTION = 4; // Підтримуються команди SMART
// ------------------------------------------- --------------------
// Регістри IDE
// ------------------------------------------- --------------------
type
   TIDERegs = packed record
      bFeaturesReg: BYTE; // Використовується для визначення SMART "під
команди".
      bSectorCountReg: BYTE; // Регістр кількості секторів IDE
      bSectorNumberReg: BYTE; // Регістр номера сектора IDE
      bCylLowReg: BYTE; // Молодший розряд номера циліндра IDE
      bCylHighReg: BYTE; // Старший розряд номера циліндра IDE
      bDriveHeadReg: BYTE; // Регістр диска / головки IDE
      bCommandReg: BYTE; // Фактична команда IDE
      bReserved: BYTE; // Зарезервовано для
                                       // Майбутнього використання. Повинне бути
0.
   end;
   IDEREGS = TIDERegs;
   PIDERegs = ^ TIDERegs;

// Допустимі значення для параметра bCommandReg структури IDEREGS.
```

```
const
IDE_ATAPI_ID = $ A1; // Повертає ID сектора для ATAPI.
IDE_ID_FUNCTION = $ EC; // Повертає ID сектора для ATA.
IDE_EXECUTE_SMART_FUNCTION = $ B0;
 // Виконує команду SMART.
// Вимагає правильних значень для параметрів bFeaturesReg
// BCylLowReg, і bCylHighReg.
// Значення регістра циліндра потрібні при використанні команди SMART
SMART_CYL_LOW = $ 4F;
SMART_CYL_HI = $ C2;
// ------------------------------------------------ --------------------
// SENDCMDINPARAMS містить вхідні параметри для функції Send Command to Drive.
// ------------------------------------------------ --------------------
type
   TSendCmdInParams = packed record
      cBufferSize: DWORD; // Розмір буфера в байтах.
      irDriveRegs: TIDERegs; // Структура зі значеннями регістрів диска.
      bDriveNumber: BYTE; // Фізичний номер диска
                                             // команд (0,1,2,3).
      bReserved: array [0 .. 2] of Byte; // Зарезервовано для майбутнього
                                                          // розширення.
      dwReserved: array [0 .. 3] of DWORD; // Зарезервовано для майбутнього
                                                                 //
використання.
      bBuffer: array [0 .. 0] of Byte; // Вхідний буфер.
   end;
   SENDCMDINPARAMS = TSendCmdInParams;
   PSendCmdInParams = ^ TSendCmdInParams;

// ------------------------------------------------ --------------------
// Стан повертане драйвером
// ------------------------------------------------ --------------------

type
   TDriverStatus = packed record
      bDriverError: Byte; // Код помилки драйвера.
      bIDEStatus: Byte; // Зміст регістра помилки.
                        // Правильно, тільки коли bDriverError = SMART_IDE_ERROR.
      bReserved: array [0 .. 1] of Byte;
// Зарезервовано для майбутнього розширення.
      dwReserved: array [0 .. 1] of DWORD;
// Зарезервовано для майбутнього розширення.
   end;
   DRIVERSTATUS = TDriverStatus;
   PDriverStatus = ^ TDriverStatus;

// Значення bDriverError
const
SMART_NO_ERROR = 0; // Без помилок
SMART_IDE_ERROR = 1; // Помилка контролера IDE
SMART_INVALID_FLAG = 2; // Неправильний прапор команди
SMART_INVALID_COMMAND = 3; // Неправильний байт команди
SMART_INVALID_BUFFER = 4; // Неправильний буфер (нульовий, невірна адреса і
т.д.)
SMART_INVALID_DRIVE = 5; // Неправильний номер привода
SMART_INVALID_IOCTL = 6; // Неправильна IOCTL-команда
SMART_ERROR_NO_MEM = 7; // Неможливо захопити буфер користувача
SMART_INVALID_REGISTER = 8; // Деякі регістри IDE неправильні
SMART_NOT_SUPPORTED = 9; // Неприпустимий набір прапорів команди.
SMART_NO_IDE_DEVICE = 10; // Команда, послана пристрою не існує, хоча номер
диска правильний.
// Значення з 11 по 255 зарезервовані

// ------------------------------------------------ --------------------
// Структура, яка повертається SMART IOCTL для декількох команд
// ------------------------------------------------ --------------------
type
   TSendCmdOutParams = packed record
      cBufferSize: DWORD; // Розмір bBuffer в байтах
```

```
      DriverStatus: TDriverStatus; // Структура стану драйвера.
      bBuffer: array [0 .. 0] of BYTE; // Буфер, довільної довжини,
                                                    // для збереження
даних, прочитаних з диска.
   end;
   SENDCMDOUTPARAMS = TSendCmdOutParams;
   PSendCmdOutParams = ^ TSendCmdOutParams;
// ------------------------------------------------ --------------------
// Константи для параметра bFeaturesReg структури TIDERegs для "під-команнд"
SMART
// ------------------------------------------------ --------------------
const
SMART_READ_ATTRIBUTE_VALUES = $ D0;
SMART_READ_ATTRIBUTE_THRESHOLDS = $ D1;
SMART_ENABLE_DISABLE_ATTRIBUTE_AUTOSAVE = $ D2;
SMART_SAVE_ATTRIBUTE_VALUES = $ D3;
SMART_EXECUTE_OFFLINE_IMMEDIATE = $ D4
SMART_ENABLE_SMART_OPERATIONS = $ D8;
SMART_DISABLE_SMART_OPERATIONS = $ D9;
SMART_RETURN_SMART_STATUS = $ DA;
// ------------------------------------------------ --------------------
// Наступний тип визначає структуру атрибутів диска
// ------------------------------------------------------------ --------------------
type
   TDriveAttribute = packed record
      bAttrID: BYTE; // Ідентифікатор атрибуту
      wStatusFlags: WORD; // Прапори стану
      bAttrValue: BYTE; // Поточне нормалізоване значення
      bWorstValue: BYTE; // Найгірше значення
      bRawValue: array [0 .. 5] of BYTE; // ненормалізованного значення
      bReserved: BYTE; // Зарезервовано
   end;
   DRIVEATTRIBUTE = TDriveAttribute;
   PDriveAttribute = ^ TDriveAttribute;

// ------------------------------------------------ --------------------
// Значення параметра wStatusFlags структури TDriveAttribute
// ------------------------------------------------ --------------------
const
PRE_FAILURE_WARRANTY = $ 01; // Життєво-важливий
ON_LINE_COLLECTION = $ 02; //
PERFORMANCE_ATTRIBUTE = $ 04; // Атрибут, що відображає продуктивність диска
ERROR_RATE_ATTRIBUTE = $ 08; / / Атрибут, що відображає частоту появи помилок
EVENT_COUNT_ATTRIBUTE = $ 10; // Лічильник подій
SELF_PRESERVING_ATTRIBUTE = $ 20; // самозберігається атрибут

// ------------------------------------------------ --------------------
// Наступна структура визначає структуру гарантійного порогу
// ------------------------------------------------ --------------------
type
   TAttrThreshold = packed record
      bAttrID: BYTE; // Ідентифікатор атрибуту
      bWarrantyThreshold: BYTE; // Граничне значення
      bReserved: array [0 .. 9] of BYTE; // Зарезервовано
   end;
   ATTRTHRESHOLD = TAttrThreshold;
   PAttrThreshold = ^ TAttrThreshold;
// ------------------------------------------------ --------------------
// Наступна структура визначає частину буфера IDENTIFY:
// ------------------------------------------------ --------------------
   TIdSector = packed record
      wGenConfig: USHORT;
      wNumCyls: USHORT;
      wReserved: USHORT;
      wNumHeads: USHORT;
      wBytesPerTrack: USHORT;
      wBytesPerSector: USHORT;
      wSectorsPerTrack: USHORT;
      wVendorUnique: array [0 .. 2] of USHORT;
```

```
      sSerialNumber: array [0 .. 19] of CHAR;
      wBufferType: USHORT;
      wBufferSize: USHORT;
      wECCSize: USHORT;
      sFirmwareRev: array [0 .. 7] of CHAR;
      sModelNumber: array [0 .. 39] of CHAR;
      wMoreVendorUnique: USHORT;
      wDoubleWordIO: USHORT;
      wCapabilities: USHORT;
      wReserved1: USHORT;
      wPIOTiming: USHORT;
      wDMATiming: USHORT;
      wBS: USHORT;
      wNumCurrentCyls: USHORT;
      wNumCurrentHeads: USHORT;
      wNumCurrentSectorsPerTrack: USHORT;
      ulCurrentSectorCapacity: ULONG;
      wMultSectorStuff: USHORT;
      ulTotalAddressableSectors: ULONG;
      wSingleWordDMA: USHORT;
      wMultiWordDMA: USHORT;
      bReserved: array [0 .. 127] of BYTE;
   end;
   PIdSector = ^ TIdSector;
   IDSECTOR = TIdSector;
   _IDSECTOR = TIdSector;

const
NUM_ATTRIBUTE_STRUCTS = 30;

implementation

end.
```

ФАЙЛ МОДУЛЮ UNIT3.PAS

```
unit Unit3; // Об'ява модулю

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, magfmtdisk, FileCtrl, ComCtrls, magsubs1;

type
  TMainF2 = class(TForm)
    Log: TMemo;
    Panel1: TPanel;
    doChkDsk: TButton;
    doExit: TButton;
    DriveBox: TDriveComboBox;
    OptCorrectErrors: TCheckBox;
    OptVerbose: TCheckBox;
    OptCheckDirty: TCheckBox;
    OptScanDrive: TCheckBox;
    OptQuickFmt: TCheckBox;
    doAbort: TButton;
    doFrmtDsk: TButton;
    ProgressBar: TProgressBar;
    FileSystem: TComboBox;
    Label1: TLabel;
    VolumeLabel: TEdit;
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure doChkDskClick(Sender: TObject);
    procedure doExitClick(Sender: TObject);
    procedure doAbortClick(Sender: TObject);
    procedure doFrmtDskClick(Sender: TObject);
  private
    { Private declarations }
    procedure LogInfo (Info: String);
    procedure ProgressEvent (Percent: integer; var Cancel: boolean);
    procedure InfoEvent (Info: string; var Cancel: boolean);
  public
    { Public declarations }
  end;

var
  MainForm: TMainF2;
  MagFmtChkDsk: TMagFmtChkDsk;
  CancelFlag: boolean;

implementation

{$R *.dfm}

procedure TMainF2.FormDestroy(Sender: TObject);
begin
    FreeAndNil (MagFmtChkDsk);
end;

procedure TMainF2.FormCreate(Sender: TObject);
begin
    MagFmtChkDsk := TMagFmtChkDsk.Create (self);
    MagFmtChkDsk.onProgressEvent := ProgressEvent;
    MagFmtChkDsk.onInfoEvent := InfoEvent;
    if NOT IsProgAdmin then
end;

procedure TMainF2.LogInfo (Info: String);
begin
    Log.Lines.Add (Info);
```

```
end;

procedure TMainF2.ProgressEvent (Percent: integer; var Cancel: boolean);
begin
    ProgressBar.Position := Percent;
    Application.ProcessMessages;
    Cancel := CancelFlag;
end;

procedure TMainF2.InfoEvent (Info: string; var Cancel: boolean);
begin
    LogInfo (Info);
    Application.ProcessMessages;
    Cancel := CancelFlag;
end;

procedure TMainF2.doChkDskClick(Sender: TObject);
begin
    CancelFlag := false;
    doChkDsk.Enabled := false;
    doFrmtDsk.Enabled := false;
    try
        try
            ProgressBar.Position := 0;
if NOT MagFmtChkDsk.CheckDisk (DriveBox.Drive + ':\', OptCorrectErrors.Checked,
                OptVerbose.Checked, OptCheckDirty.Checked,
                OptScanDrive.Checked) then
LogInfo ('Check Disk Failed');
            ProgressBar.Position := 0;
            if MagFmtChkDsk.FileSysProblem then
LogInfo ('!!! Check Disk Found Problems with ' +
Uppercase (DriveBox.Drive) + ':');
        except
            on E:Exception do LogInfo ('Error: ' + E.Message);
        end;
    finally
        doChkDsk.Enabled := true;
        doFrmtDsk.Enabled := true;
        LogInfo ('');
    end;
end;

procedure TMainF2.doFrmtDskClick(Sender: TObject);
var
    MediaType: TMediaType;
begin
    if Trim (VolumeLabel.Text) = '' then
    begin
        exit;
    end;
    CancelFlag := false;
    doChkDsk.Enabled := false;
    doFrmtDsk.Enabled := false;
    MediaType := mtHardDisk;
    if UpperCase (DriveBox.Drive) < 'C' then MediaType := mtFloppy;
    try
        try
            ProgressBar.Position := 0;

            if NOT MagFmtChkDsk.DATADisk (DriveBox.Drive + ':\', MediaType,
                   TFileSystem (FileSystem.ItemIndex), Trim (VolumeLabel.Text),
                                                        OptQuickFmt.Checked, 0)
then
            ProgressBar.Position := 0;
        except
            on E:Exception do LogInfo ('Error: ' + E.Message);
        end;
    finally
        doChkDsk.Enabled := true;
```

```
        doFrmtDsk.Enabled := true;
        LogInfo ('');
    end;
end;

procedure TMainF2.doExitClick(Sender: TObject);
begin
    CancelFlag := true;
    Close;
end;

procedure TMainF2.doAbortClick(Sender: TObject);
begin
    CancelFlag := true;
end;


end.
```

ФАЙЛ МОДУЛЮ UNIT4.PAS

```
unit Unit4; // Об’ява модулю

interface

uses
  Windows, Messages, SysUtils, Classes;

const
    fmifs = 'fmifs.dll';
    WM_GETOBJ = WM_USER + 701;

// прапори
    FMIFS_HARDDISK = $0C;
    FMIFS_FLOPPY   = $08;

type
    TextOutput = record
          Lines:  DWORD;
          Output: PAnsiChar;  // unicode
    end;
   PTextOutput = ^TextOutput;

// Callback функція
    TCallBackCommand = (
        PROGRESS,
        DONEWITHSTRUCTURE,
      UNKNOWN2,
          UNKNOWN3,
      UNKNOWN4,
          UNKNOWN5,
      INSUFFICIENTRIGHTS,
          FSNOTSUPPORTED,
      VOLUMEINUSE,
      UNKNOWN9,
          UNKNOWNA,
      DONE,
          UNKNOWNC,
      UNKNOWND,
      OUTPUT,
        STRUCTUREPROGRESS,
        CLUSTERSIZETOOSMALL,
        UNKNOWN11,
        UNKNOWN12,
        UNKNOWN13,
        UNKNOWN14,
        UNKNOWN15,
        UNKNOWN16,
        UNKNOWN17,
        UNKNOWN18,
        PROGRESS2,
        UNKNOWN1A);
var

// Chkdsk процедура

Chkdsk: procedure (
    DriveRoot: PWCHAR;
    DATA: PWChar;
    CorrectErrors: BOOL;
      Verbose: BOOL;
      CheckOnlyIfDirty: BOOL;
      ScanDrive: BOOL;
      Unused2: DWORD;
      Unused3: DWORD;
      Callback: Pointer); stdcall;
```

```
DATAEx: procedure (
    DriveRoot: PWCHAR;
      MediaFlag: DWORD;
      DATA: PWCHAR;
      DiskLabel: PWCHAR;
      QuickDATA: BOOL;
      ClusterSize: DWORD;
      Callback: Pointer); stdcall;

// Enable/Disable функція

EnableVolumeCompession: function (
    DriveRoot: PWCHAR;
      Enable: BOOL): BOOLEAN; stdcall;

type

  TMediaType = (mtHardDisk, mtFloppy);
  TFileSystem = (fsNTFS, fsFAT, fsFAT32);
  TProgressEvent = Procedure (Percent: integer; var Cancel: boolean) of object;
  TInfoEvent = Procedure (Info: string; var Cancel: boolean) of object;

  TMagFmtChkDsk = class(TComponent)
  private
    { Private declarations }
    fProgressEvent: TProgressEvent;
    fInfoEvent: TInfoEvent;
    fDoneOK: boolean;
    fFileSysProblem: boolean;
    fFreeSpaceAlloc: boolean;
    fFirstErrorLine: string;
  protected
    { Protected declarations }
    function CheckDriveExists (const WDrive: WideString;
                               CheckInUse: boolean; var WDATA: WideString):
boolean;
    function doProgressEvent (const Percent: integer): boolean;
    function doInfoEvent (const Info: string): boolean;
    procedure WMGETOBJ (var msg: TMessage); message WM_GETOBJ;
  public
    { Public declarations }
    function LoadFmifs: boolean;
    function DATADisk (const DrvRoot: string; MediaType: TMediaType;
FileSystem: TFileSystem;
        const
DiskLabel: string; QuickDATA: boolean; ClusterSize: integer): boolean;
    function CheckDisk (const DrvRoot: string; CorrectErrors, Verbose,
                        CheckOnlyIfDirty, ScanDrive: boolean): boolean;
    function VolumeCompression (const DrvRoot: string; Enable: boolean):
boolean;
  published
    { Published declarations }
    property FileSysProblem: boolean          read fFileSysProblem;
    property FreeSpaceAlloc: boolean          read fFreeSpaceAlloc;
    property FirstErrorLine: string           read fFirstErrorLine;
    property onProgressEvent: TProgressEvent  read fProgressEvent write
fProgressEvent;
    property onInfoEvent:     TInfoEvent      read fInfoEvent     write
fInfoEvent;

  end;

  FmtChkException = class(Exception);

var
  MagFmifsib: THandle = 0;
  MagFmifs_Loaded: Boolean = false;   // DLL functions завантажено
  MagFmtObj: TObject;
```

```
implementation

procedure Register;
begin
  RegisterComponents('Samples', [TMagFmtChkDsk]);
end;

// об'ява

function DATACallback (Command: TCallBackCommand; SubAction: DWORD;
                                                    ActionInfo: Pointer): Boolean;
stdcall;
var
    flag: pboolean;
    percent: pinteger;
    toutput: PTextOutput;
    Obj: TObject;
    cancelflag: boolean;
    info: string;
    xlatbuf : AnsiString;
    progper, slen: integer;
begin
    result := true;
    cancelflag := false;
    Obj := TObject (SendMessage (HInstance, WM_GETOBJ, 0, 0));
    Obj := MagFmtObj;
    progper := -1;
    info := '';
    if NOT Assigned (TMagFmtChkDsk (Obj)) then exit;
    case Command of
        Progress:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Output:
            begin
                toutput := ActionInfo;
                slen := StrLen (toutput^.Output);
                SetLength (xlatbuf, slen);
                info := Trim (String (xlatBuf));
            end;
    if progper >= 0  then cancelflag :=
            TMagFmtChkDsk (Obj).doProgressEvent (progper);
    if info <> '' then cancelflag := TMagFmtChkDsk (Obj).doInfoEvent (info);
    result := NOT cancelflag;
end;

function ChkDskCallback (Command: TCallBackCommand; SubAction: DWORD;
                                                    ActionInfo: Pointer): Boolean;
stdcall;
var
    flag: pboolean;
    percent: pinteger;
    toutput: PTextOutput;
    Obj: TObject;
    info: string;
    progper, slen: integer;
    cancelflag: boolean;
    xlatbuf : AnsiString;
begin
    result := true;
    cancelflag := false;
    progper := -1;
    info := '';
    Obj := TObject (SendMessage (HInstance, WM_GETOBJ, 0, 0));
    Obj := MagFmtObj;
    if NOT Assigned (TMagFmtChkDsk (Obj)) then exit;
```

```
    case Command of
        Progress:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Progress2:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Output:
            begin
                toutput := ActionInfo;
                slen := StrLen (toutput^.Output);
                SetLength (xlatbuf, slen);
                OemToCharBuffA (PAnsiChar (toutput^.Output), PAnsiChar
                (xlatBuf), slen);
                info := Trim (String (xlatBuf));
                if (Pos ('found problems', info) > 0) or
                     (Pos ('Correcting errors', info) > 0) or
                       (Pos ('Errors found', info) > 0) or
                         (Pos ('(fix) option', info) > 0) then
                begin
                     TMagFmtChkDsk (Obj).fFileSysProblem := true;
                     if TMagFmtChkDsk (Obj).fFirstErrorLine = '' then
                                  TMagFmtChkDsk (Obj).fFirstErrorLine:=info;
                end;
                if (Pos ('free space marked as allocated', info) > 0) then
                begin
                     TMagFmtChkDsk (Obj).fFreeSpaceAlloc := true;
                     if TMagFmtChkDsk (Obj).fFirstErrorLine = '' then
                                  TMagFmtChkDsk(Obj).fFirstErrorLine:=info;
                end;
            end;
        Done:
            begin
                flag := ActionInfo;
                TMagFmtChkDsk (Obj).fDoneOK := flag^;
                if flag^ then
                  info := 'Check Disk: Finished OK'
                else
                    info := 'Check Disk: Unable to Finish';
            end;
          FSNotSupported: info := 'Check Disk: FS Not Supported';
      VolumeInUse: info := 'Check Disk: Volume In-Use';
        InsufficientRights: info := 'Check Disk: Insufficient Rights';
        else
            info := 'Check Disk Callback: ' + IntToStr (Ord (Command));
    end;
    if progper >= 0  then cancelflag:=
TMagFmtChkDsk (Obj).doProgressEvent (progper);
    if info <> '' then cancelflag := TMagFmtChkDsk (Obj).doInfoEvent (info);
    result:=NOT cancelflag;
end;

procedure TMagFmtChkDsk.WMGETOBJ (var msg: TMessage);
begin
    msg.Result := Integer (TMagFmtChkDsk);
end;

function TMagFmtChkDsk.doProgressEvent (const Percent: integer): boolean;
begin
    result := false;
    if Assigned (fProgressEvent) then fProgressEvent (Percent, result);
end;

function TMagFmtChkDsk.doInfoEvent (const Info: string): boolean;
begin
```

```
    result := false;
    if Assigned (fInfoEvent) then fInfoEvent (Info, result);
end;


function TMagFmtChkDsk.CheckDriveExists (const WDrive: WideString;
                           CheckInUse: boolean; var WDATA: WideString): boolean;
var
    FileSysName  : Array[0..MAX_PATH] of WChar;
    VolumeName   : Array[0..MAX_PATH] of WChar;
    maxcomlen, flags: longword;
    handle: THandle;
    voldev: WideString;
begin

    if NOT GetVolumeInformationW (PWChar (WDrive), VolumeName,
SizeOf(VolumeName) div 2,
                Nil, maxcomlen, flags, FileSysName, SizeOf(FileSysName) div 2)
then
    begin
        raise FmtChkException.Create('Drive Not Found: ' + WDrive);
        exit;
    end;
    WFormat := FileSysName;
    doInfoEvent (WDrive + ' Volume Label: ' + VolumeName + ', File System: ' +
FileSysName);

    // спроба отримання доступу до тому
    if CheckInUse then
    begin
        voldev := '\\.\' + WDrive [1] + ':';
        handle := CreateFileW (PWChar (voldev), Generic_Write, 0, nil,
Open_Existing, 0, 0);
        if handle = INVALID_HANDLE_VALUE then
        begin
            raise FmtChkException.Create('Drive In Use: ' + WDrive);
            exit;
        end;
        CloseHandle (handle);
    end;
    result := true;
end;

function TMagFmtChkDsk.DATADisk (const DrvRoot: string; MediaType: TMediaType;
                          FileSystem: TFileSystem; const DiskLabel: string;
                             QuickDATA: boolean; ClusterSize: integer):
boolean;
var
    wdrive, wformat, wfilesystem, wdisklabel: widestring;
    mediaflags, newsize: DWORD;

begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    wdisklabel := Uppercase (DiskLabel);
    if MediaType = mtHardDisk then
        mediaflags := FMIFS_HARDDISK
    else if MediaType = mtFloppy then
        mediaflags := FMIFS_FLOPPY
    else
        exit;
    if FileSystem = fsFAT then
        wfilesystem := 'FAT'
    else if FileSystem = fsFAT32 then
        wfilesystem := 'FAT32'
    else if FileSystem = fsNTFS then
        wfilesystem := 'NTFS'
    else
```

```
        exit;
    newsize := 0;
    if ((ClusterSize = 512) or (ClusterSize = 1024) or (ClusterSize = 2048) or
        (ClusterSize = 4096) or (ClusterSize = 8192) or (ClusterSize = 16384) or
            (ClusterSize = 32768) or (ClusterSize = 65536)) then newsize :=
ClusterSize;
    fDoneOK := false;
    if DiskSize (Ord (WDrive [1]) - 64) > 100 then
    begin
        doInfoEvent (WDrive + ' Checking Existing Drive Format');
        if NOT CheckDriveExists (wdrive, true, wformat) then exit;
        if wformat <> wfilesystem then QuickFormat := false;
    end
    else
    begin
        if (Length (WDrive) < 2) or (WDrive [2] <> ':') then
        begin
raise FmtChkException.Create('Invalid Drive Specification: ' + WDrive);
            exit;
        end;
                QuickDATA := false;
    end;
    MagFmtObj := Self;
    fFirstErrorLine := '';
    DATAEx (PWchar (wdrive), mediaflags, PWchar (wfilesystem), PWchar
(wdisklabel), QuickDATA, newsize, @DATACallback);
    result := fDoneOK;
    if NOT result then exit;
    doInfoEvent (WDrive + ' Checking New Drive DATA');
    if NOT CheckDriveExists (wdrive, false, wformat) then exit;
    doInfoEvent (WDrive + ' New Volume Space: ' + IntToStr (DiskFree (Ord
(WDrive [1]) - 64)));
end;

function TMagFmtChkDsk.CheckDisk (const DrvRoot: string; CorrectErrors, Verbose,
                                  CheckOnlyIfDirty, ScanDrive: boolean):
boolean;
var
    wdrive, wDATA: widestring;
begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    if NOT CheckDriveExists (wdrive, CorrectErrors, wDATA) then exit;
    MagFmtObj := Self;
    fDoneOK := false;
    fFileSysProblem := false;
    fFreeSpaceAlloc := false;
    fFirstErrorLine := '';
    Chkdsk (PWchar (wdrive), PWchar (wDATA), CorrectErrors, Verbose,
                             CheckOnlyIfDirty, ScanDrive, 0, 0,
@ChkDskCallback);
    if fFileSysProblem then
        result := true  // припинення при помилці
    else
        result := fDoneOK;
end;

function TMagFmtChkDsk.VolumeCompression (const DrvRoot: string; Enable:
boolean): boolean;
var
    wdrive, wDATA: widestring;
begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    if NOT CheckDriveExists (wdrive, true, wDATA) then exit;
    result := EnableVolumeCompession (PWchar (wdrive), Enable);
end;
```

```
function TMagFmtChkDsk.LoadFmifs: boolean;
begin
    result := Assigned (Chkdsk);
    if MagFmifs_Loaded then exit;
    result := false;
    if Win32Platform <> VER_PLATFORM_WIN32_NT then exit;

// завантаження бібліотеки
    result := false;
    MagFmifs_Loaded := True;
    MagFmifsib := LoadLibrary (fmifs);
    if MagFmifsib = 0 then exit;

// встановлення адрес функції у DLL
    Chkdsk := GetProcAddress (MagFmifsib, 'Chkdsk');
    DATAEx := GetProcAddress (MagFmifsib, 'DATAEx');
    EnableVolumeCompession := GetProcAddress (MagFmifsib,
                                              'EnableVolumeCompession');
    result := Assigned (Chkdsk);
end;

Initialization
    MagFmifsib := 0;
    MagFmifs_Loaded := false;
finalization
    if MagFmifs_Loaded then FreeLibrary (MagFmifsib);
end.
```