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 моніторингу стану жорсткого диску з використанням технології SMART”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Касяненко І.С.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 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

Касяненку Іллі Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART*

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

затверджені наказом вищого навчального закладу № 32-13 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 моніторингу стану жорсткого диску з використанням технології SMART*

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

Касяненко І.С. Дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART. 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 моніторингу стану жорсткого диску з використанням технології SMART.

Метою розробки є дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

Об'єктом дослідження є процес моніторингу стану жорсткого диску з використанням технології SMART.

Предметом дослідження є методи моніторингу стану жорсткого диску з використанням технології SMART.

Методи дослідження базуються на методах схемотехніки, теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

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

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

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

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, жорсткий диск, SMART

ABSTRACT

Kasianenko I.S. Research and software implementation of the hard disk status monitoring system using SMART technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the hard disk condition monitoring system using SMART technology.

The goal of the development is the research and software implementation of the hard disk condition monitoring system using SMART technology.

The object of the study is the process of monitoring the state of the hard disk using SMART technology.

The subject of the study is methods of monitoring the state of the hard disk using SMART technology.

Research methods are based on circuit engineering methods, reliability theory, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the hard disk status monitoring system using SMART technology.

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

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

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 Sydney environment.

Keywords: computer engineering, hard disk, SMART

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. SMART (Self-Monitoring Analysis and Reporting Technology) розроблено IBM. Він створений для контролю стану диска за допомогою різних методів і пристроїв (датчиків). Один жорсткий диск ATA може мати до 30 таких вимірюваних значень, які називаються атрибутами . Деякі з них прямо чи опосередковано впливають на стан жорсткого диска, а інші надають статистичну інформацію.

Сьогодні всі сучасні жорсткі диски IDE/Serial ATA/SCSI мають функцію SMART. Насправді це не стандарт, тому значення атрибутів можуть відрізнятися від виробника до виробника. У цій статті ми обговорюємо лише жорсткі диски ATA (IDE та Serial ATA). Жорсткі диски SCSI працюють по-іншому: дані прогнозування збоїв є стандартними, і існують суворі правила щодо датчиків і алгоритмів. Наприклад, різниця між реальною температурою та результатом, виміряним датчиком, має бути менше +/- 3 градусів Цельсія.

Багато атрибутів використовуються всіма виробниками, і вони використовуються однаково (або майже однаково). Ось чому, наприклад, можна визначити температуру та загальну потужність багатьох жорстких дисків. Нові програми здатні виявляти, обробляти та відображати цю інформацію.

Згідно зі специфікаціями SMART, у разі виявлення проблеми (передбачуваної несправності) жорсткий диск повинен працювати принаймні 24 години для виконання резервного копіювання даних. Але в багатьох випадках цього часу недостатньо, тому важливо розпізнати проблеми та підготуватися, поки не пізно.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

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

– Огляд існуючих систем моніторингу стану жорсткого диску з використанням технології SMART.

– Дослідження системи моніторингу стану жорсткого диску з використанням технології SMART.

– Програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

Об'єктом дослідження є процес моніторингу стану жорсткого диску з використанням технології SMART.

Предметом дослідження є методи моніторингу стану жорсткого диску з використанням технології SMART.

Методи дослідження базуються на методах схемотехніки, теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу стану жорсткого диску з використанням технології SMART.

– Розроблено вітчизняний продукт моніторингу стану жорсткого диску з використанням технології SMART, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу стану жорсткого диску з використанням технології SMART.

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 моніторингу стану жорсткого диску з використанням технології SMART, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації моніторингу стану жорсткого диску з використанням технології S.M.A.R.T.

Поточний стан жорсткого диска постійно перевіряється багатьма датчиками. Потім виміряні значення обробляються певними алгоритмами, а відповідні атрибути змінюються відповідно до результатів.

Один атрибут SMART має такі поля:

– **Ідентифікатор** (байт): значення атрибута. Багато атрибутів мають стандартні значення (наприклад, 5 = кількість перерозподілених секторів, 194 = температура тощо). Більшість програм надають назву та текстовий опис атрибутів.

– **Дані** (6 байт): у цьому полі зберігаються необроблені виміряні значення, надані датчиком або лічильником. Потім ці дані обробляються за алгоритмом, розробленим виробником жорсткого диска. Іноді різні частини (наприклад, молодші, середні, старші 16 бітів) цього значення містять різний тип інформації.

– **Поріг** (байт): граничне значення (відмов) для атрибута.

– **Значення** (байт): поточний відносний "справність" атрибута. Це число розраховується алгоритмом, використовуючи необроблені дані (див. вище). На новому жорсткому диску це число є високим (теоретичний максимум, наприклад 100, 200 або 253) і воно зменшується протягом терміну служби диска.

– **Найгірше** (байт): найгірше (найменше) значення, будь-коли знайдене за попередній термін служби жорсткого диска.

– **Прапорці статусу** : вказують на основне призначення атрибута. Атрибут може бути, наприклад, критичним (може передбачити збій) або статистичним (не впливає безпосередньо на стан).

Програмне забезпечення може відображати більше інформації на основі цих полів (наприклад, статус атрибута, який може бути «ОК» або «Завжди ОК» тощо) і може надати допомогу в оцінці або керуванні атрибутами.

Атрибут **правильний, якщо значення більше або дорівнює граничному значенню** . Якщо для критичного атрибута це не відповідає дійсності, передбачається збій, жорсткий диск вважається несправним і його слід негайно замінити (атрибут визначає проблему). Виробники/постачальники замінюють жорсткий диск на умовах гарантії. Функція SMART у сучасних BIOS материнських плат попереджає користувача про цей момент перед завантаженням операційної системи. Якщо Порогове значення дорівнює 0 для будь-якого атрибута, цей атрибут *не* може передбачити помилку (оскільки значення не може бути менше 0).

1.2 Область застосування

Областю застосування є система перевірки на помилки, надійності та відказостійкості жорсткого диску.

Описана вище модель має багато слабких місць. Через ці проблеми в більшості випадків передбачення несправностей взагалі не працює. Основні проблеми:

Неправильні пороги

Більшість проблем із SMART (відсутність передбачення помилок) викликані неправильно вибраними пороговими значеннями. Через це атрибути жорсткого диска **не мають шансів** досягти порогових значень - зазвичай вони виходять з ладу (стають марними), не досягнувши цієї точки. У таких випадках SMART дійсно не передбачає збою.

На практиці ми можемо знайти нереалістичні порогові значення. Наприклад, на більшості жорстких дисків потрібні тисячі пошкоджених (неможливих для читання та запису) секторів (відповідно до розміру резервної

області), перш ніж SMART покаже проблему. Здається, це не є великою проблемою, оскільки 1000 таких пошкоджених секторів займає «всього» 512 000 байт даних (і це не означає втрати ємності через використання резервної області), але може бути важливим те, як народжуються ці пошкоджені сектори, де вони розташовані на поверхні та яка швидкість збільшення пошкодженого сектора.

У більшості випадків проблеми можна виявити задовго до того, як значення атрибута досягне порогового значення. Наприклад, проблема з головкою, яка може зробити багато тисяч секторів непридатними (поганими), може призвести до того, що більші частини поверхні диска стануть нечитабельними, перешкоджаючи відновленню даних із цієї області диска. Крім того, для аналізу такої проблемної області та збереження даних у резервну область може знадобитися багато часу (навіть годин), і можливо, що операція не завершиться без помилок. Під час цього процесу операційна система зазвичай перестає відповідати, тому проблемний жорсткий диск може спричинити повну нестабільність системи.

Ми також можемо обговорити неправильно обрані порогові значення. Деякі виробники жорстких дисків можуть визначити 60-70 років або навіть більше для загального терміну служби жорсткого диска, якщо перевірити відповідний атрибут. Це дійсно цікаво - тому що виробники зазвичай визначають розрахований термін служби в **5 років** в інструкціях до продукції. Крім того, SMART не попереджатиме про закінчення терміну служби, оскільки цей атрибут зазвичай не є критичним.

Крім того, **порогове значення дорівнює 0 для багатьох критичних атрибутів** . Оскільки значення не можна зменшити нижче 0, ці атрибути ніколи не вказуватимуть на будь-яку ознаку помилки – навіть якщо вони «хочуть» це зробити. Тож **SMART ніколи не попередить** .

Іноді дуже важливі атрибути не позначені як «критичні». Це означає, що програми моніторингу жорсткого диска та функція BIOS SMART взагалі не перевіряють ці атрибути.

Неправильний метод оцінки

Більшість програм використовують описаний вище метод постачальника для обчислення та відображення справності диска. Результатом є те, що більшість жорстких дисків виглядають набагато краще, ніж їх реальний стан. Виробники жорстких дисків можуть вибрати порогові значення або алгоритми, щоб показувати свої жорсткі диски краще, ніж інші жорсткі диски іншого виробника. Це може ввести програми та користувачів в оману.

Розробники програмного забезпечення просто використовують метод оцінки, який залежить від виробника, і вони нічого не роблять для визначення справжнього стану працездатності дисків. Через це можливо, що користувач використовує програму моніторингу жорсткого диска, але жорсткий диск вийде з ладу, перш ніж виявить будь-які ознаки проблем або навіть погіршення стану. Такі додатки можуть відображати 10-20 років або більше як очікуваний термін служби, що залишився, що є принаймні сумнівним.

Вага атрибутів

Різні атрибути по-різному впливають на стан диска. Деякі атрибути (наприклад, 10 - кількість повторних спроб) є дуже критичними. Невелика зміна в цьому атрибуті може вказувати на серйозну проблему, наприклад, поганий двигун або підшипник, але, можливо, слабке джерело живлення також може спричинити цю проблему.

Для таких атрибутів виробники часто використовують високі порогові значення, тому їх можна відносно легко досягти. Але через вибір порогового значення та функції f, описаної вище в нерівності (1), деякі проблеми можна повністю проігнорувати. Тому користувачі не помітять жодних змін критичних атрибутів.

Інша проблема полягає в тому, що зв'язок між атрибутами часто ігнорується. Цілком можливо, що два або більше значень атрибутів майже досягають порогових значень, але невдача не передбачена, оскільки жодне значення не досягло порогового рівня.

Відсутність зворотного зв'язку

Без використання належного програмного забезпечення, здатного зчитувати інформацію SMART, користувач **не помічає жодних проблем** із жорстким диском, просто коли стає надто пізно. Якщо кількість пошкоджених секторів зростає повільно (жорсткий диск знаходить нові проблемні сектори, перевіряє їх і перерозподіляє), користувач може нічого не помітити, особливо якщо працює лише заставка. Але під час процедури повторного розподілу операційна система, здається, зависла (не відповідає), і користувачі можуть скинути або вимкнути комп'ютер у цей час. Така втрата живлення не дуже допомагає жорсткому диску в процесі відновлення (його буде перезапущено пізніше).

Температура, проблеми з датчиком

Без використання програмного забезпечення користувач також може не помітити високої температури жорсткого диска. І процесор, і новіші карти VGA мають захист (аварійне вимкнення) від високих температур, але жорсткі диски не мають такого захисту. Що ще гірше, **жорсткі диски набагато чутливіші до високих температур,** ніж будь-які інші компоненти в корпусі комп'ютера. Тому більшість виробників обмежують максимальну робочу температуру 50-55 градусами Цельсія.

Більшість BIOS підтримує перевірку напруги джерела живлення, швидкості вентилятора, температури процесора тощо. Але перевірити температуру жорсткого диска з BIOS неможливо. Функція BIOS SMART не попереджає, якщо температура жорсткого диска занадто висока. Тому можливо, що жорсткий диск працює в дуже гарячому середовищі.

Але важливо знати, що багато датчиків температури жорсткого диска не надто точні (іноді різниця між виявленою та реальною температурою може становити 8-10 градусів за Цельсієм і навіть більше). Рекомендується використовувати зовнішній пристрій (наприклад, інфрачервоний термометр) для вимірювання температури жорсткого диска та налаштування різниці між

виміряними та відображеними значеннями (калібрування). Таким чином, програмне забезпечення потім відображає правильне (скориговане) значення температури (якщо ця функція підтримується).

Також рекомендується перевіряти температуру, коли жорсткий диск простоює і коли він працює протягом тривалого часу.

Неправильні драйвери

Ми можемо знайти багато неправильних драйверів для контролерів жорсткого диска. Використовуючи такі драйвери, один або кілька жорстких дисків не надають інформацію SMART, підключену до таких контролерів (або материнських плат). Зазвичай це не залежить від використовуваного програмного забезпечення, оскільки програми зазвичай використовують той самий метод для доступу до жорсткого диска та виявлення інформації про нього. Можливо, що два жорсткі диски надають 100% однакову інформацію (зазвичай відомості про перший або ОСНОВНИЙ ГОЛОВНИЙ жорсткий диск). Програмне забезпечення може відфільтрувати це та відобразити реальну (але часткову) інформацію, але рекомендується перевірити правильність деталей (наприклад, серійний номер жорсткого диска не відображається 2 або більше разів).

Зазвичай драйвери підтримують лише обмежений діапазон команд жорсткого диска. Тому деякі функції працюють не у всіх випадках (наприклад, управління акустикою), навіть якщо диск це підтримує.

Рекомендується перевірити, чи виробник оновив, виправив пакети драйверів або оновлення мікропрограми. Вони **можуть** покращити ситуацію. Якщо контролер має драйвери RAID і не RAID, важливо використовувати правильні драйвери (не RAID, якщо не використовується масив RAID). Використання іншого пакета може обмежити деякі функції, і зазвичай не відображатимуться температура та стан працездатності диска(ів).

Багато материнських плат або контролерів жорстких дисків не мають 100% правильних драйверів для Vista. Це також може запобігти виявленню

детальної інформації про жорсткий диск і передбаченню збоїв у новій операційній системі.

Неправильне обладнання або неправильні дані

Це розширення попередньої задачі №6. Деякі контролери жорстких дисків або материнські плати взагалі не забезпечують порогові значення SMART або всі порогові значення дорівнюють 0. Жорсткі диски, підключені до таких контролерів, не демонструватимуть жодних ознак несправності, оскільки значення атрибутів не можуть опускатися нижче 0. Програми також можуть показати стан жорсткого диска "відмінний", оскільки значення далекі від порогових значень.

Також можливо, що інформація, надана контролером жорсткого диска, неповна. Це не впливає на стан передбачення збою жорсткого диска, але деяка інформація, виявлена та відображена, може бути неправильною. На щастя, це **не** впливає на температуру та стан жорсткого диска. Новіші програми перевіряють підпис ATA та значення контрольної суми (описано на сторінці 116 «AT Attachment — 8 ATA/ATAPI Command Set») і відображають попередження, якщо ці значення неправильні.

Пошкоджені сектори?

Користувачі часто запитують про те, що таке «погані сектори», як вони народжуються і що вони можуть зробити, щоб їх виправити. Користувачі збентежені, оскільки перевірка поверхні диска за допомогою програмного забезпечення (наприклад, за допомогою Windows Scandisk) не повідомляє про проблеми чи пошкоджені сектори.

SMART постійно аналізує поверхню диска під час нормальної роботи. Якщо він знаходить проблемну область (один або кілька секторів, де дані важко читати або записувати), він намагається прочитати дані та скопіювати їх у резервну область. Тоді вихідне розташування (внутрішньо) позначається як погане, і всі подальші операції читання/запису, що вказують на вихідне розташування, потім перенаправляються до резервної області.

Після завершення операції вихідна (несправна) область більше **не доступна програмному забезпеченню.** Навіть повторна інсталяція чи багато повних операцій форматування не викличуть проблем, оскільки вихідна пошкоджена область більше не використовується. Тому програмне забезпечення (наприклад, Scandisk Windows) не знайде проблемних секторів. Лише **апаратна** функція стирання безпеки матиме доступ до цієї області (також очистить ці сектори).

Ось чому, наприклад, команда DOS "форматувати" ніколи не покаже пошкоджені сектори на більшості сучасних жорстких дисків через SMART (за винятком випадків, коли резервна область заповнена, але знайти такий жорсткий диск справді важко).

Перерозподіл секторів може бути завершено з деякими помилками або без них (жорсткі диски зараз працюють набагато краще порівняно зі старими моделями). Але процедура перерозподілу може спричинити нестабільність системи, якщо вона займає занадто багато часу.

Користувач не повинен помічати нічого про кроки, описані вище - просто коли кількість пошкоджених секторів достатньо висока (досягнуто порогового значення), і тоді SMART передбачає можливий збій.

Рішення

Збій жорсткого диска без будь-яких ознак перед катастрофою трапляється надзвичайно рідко, за винятком випадків падіння диска, високої потужності (зміщення) чи стихійного лиха. Але ці ситуації неможливо передбачити SMART, звичайно. Зазвичай народжуються деякі пошкоджені сектори, їх кількість повільно збільшується (можливо, можуть пройти тижні без жодних ознак нових проблем). В інших випадках висока температура та/або кілька, але дійсно критичних проблем можуть спричинити смерть диска.

Також дуже часто комбінований ефект двох або більше атрибутів вказує на різні проблеми. Наприклад, якщо двигун жорсткого диска не може легко розкручуватися (це потребує кількох повторних спроб) або диск обертається

надто повільно, це може свідчити про можливу проблему з двигуном або підшипником. Такі проблеми містять сліди у відповідних атрибутах SMART. Тож усі (навіть дуже незначні) зміни можна виявити.

Важливо виявити ці ознаки задовго до того, як вони можуть призвести до збою. Рекомендується повністю відкинути всю модель, описану вище, і ігнорувати неправильно вибрані (або відсутні) порогові значення та оцінювати лише необроблені **виміряні дані, щоб виявити реальну кількість різних проблем** із жорсткими дисками. Бажано також перевірити зв'язок між різними атрибутами. Таким чином ми матимемо правильну картину про реальний стан і зможемо підготуватися та навіть уникнути втрати даних.

Також рекомендується вибрати спосіб оцінки стану жорсткого диска залежно від реального використання та «навантаження» жорсткого диска. Наприклад, у випадку з сервером, ноутбуком або жорстким диском з критичною інформацією, найменша проблема може бути небезпечною, тому будь-яка проблема (навіть невелика) повинна бути помічена.

Деякі програми можуть пропонувати такі різні методи оцінки для різних видів використання жорстких дисків і вони можуть надавати текстовий опис поточної ситуації та поради щодо покращення стану. Це чудова функція, якщо програмне забезпечення може створювати пасивні тривоги (надсилати електронну пошту, відтворювати звук або вимикати комп’ютер), але може бути краще, якщо програма здатна активно запобігати втраті даних, наприклад, виконуючи операцію автоматичного резервного копіювання якщо виявлено нову проблему.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART, 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

Програма «Victoria»

Victoria HDD/SSD – найкраща безкоштовна програма для діагностики, дослідження, тестування та дрібного ремонту жорстких дисків, SSD-накопичувачів, карт пам'яті, а також будь-яких інших накопичувачів в операційній системі Windows.

– Victoria з англійським, українським та іспанським перекладом.

– Технологія SMART HDD/SSD українською мовою.

– Підтримка накопичувачів з інтерфейсом M.2/U.2 NVMe.

– Підтримка накопичувачів з інтерфейсом SAS/SCSI.

Програма Victoria 5.xx отримала інтерфейс, який відповідає світовим стандартам.

Які перспективи її подальшого розвитку це відкриває? Це насамперед необмежене місце для розширень функціоналу, переклад інтерфейсу будь-якими мовами світу, кросплатформеність. І як наслідок – розширення аудиторії користувачів, яким програма реально допомагає у вирішенні багатьох завдань.

За останній рік «Вікторія» була значно покращена, адаптована до сучасних умов, набула безліч нових функцій. Але ще більше з'явилося нових ідей, які потрібно реалізувати.

Тестування стану поверхні будь-яких накопичувачів

У програму вбудований потужний сканер поверхні HDD, який дозволяє продіагностувати накопичувач на наявність збійних ділянок, дефектів, що плавають, і помилок інтерфейсу. Victoria здатна протестувати більшість дисків на

граничній швидкості, незалежно від їхнього типу. Особливість сканера – застосування спеціальних алгоритмів автоматичного налаштування таймаутів та розміру блоку, що дозволяє ефективно перевіряти як повільні, так і швидкі диски.

Швидке тестування поверхні. Будь-який обсяг – за 3 хвилини

Графік будується по 188 точках, рівномірно розподіленим по об'єму накопичувача, що тестується. Результат візуально та за числами ідентичний повному 4-годинному скануванню, крім знаходження дефектів. Він – для дослідження свідомо справних накопичувачів з метою вимірювання основних параметрів: швидкості на початку і наприкінці, часу доступу, поведінки при різних розмірах блоку, а також для порівняння різних пристроїв за технічними характеристиками. Графіки можна зберігати у файли та завантажувати назад у програму.

Підтримка зовнішніх накопичувачів з інтерфейсом USB

Зроблено отримання паспорта та SMART-параметрів HDD та SSD через USB, що дозволяє дізнатися все про вінчестери, приховані в USB-коробках. Підтримується 90% моделей USB-SATA мостів, і цей показник збільшуватиметься.

Додано керування кешем через USB, SMART-тести через USB. Працює також Automatic Acoustic Management (AAM) на USB-накопичувачах: програмне управління рівнем акустичного шуму за рахунок зміни швидкості переміщення головок.

Покращений паспорт накопичувача

Паспорт HDD – це інформація, що характеризує сімейство HDD/SSD та його індивідуальні особливості. До нього входить назва фірми-виробника, назва моделі, серійний номер, версія мікрокоду, логічні параметри (геометрія), параметри інтерфейсу та багато іншого.

Victoria здатна показати паспорт накопичувача, прихованого в USB-футлярі, і таким чином визначити тип встановленого всередині накопичувача.

Покращений SMART-монітор з підтримкою USB HDD та SSD

Технологія моніторингу та передбачення відмов (Self Monitoring, Analysis and Reporting Technology, скорочено SMART) є у складі сучасних HDD та SSD накопичувачів.

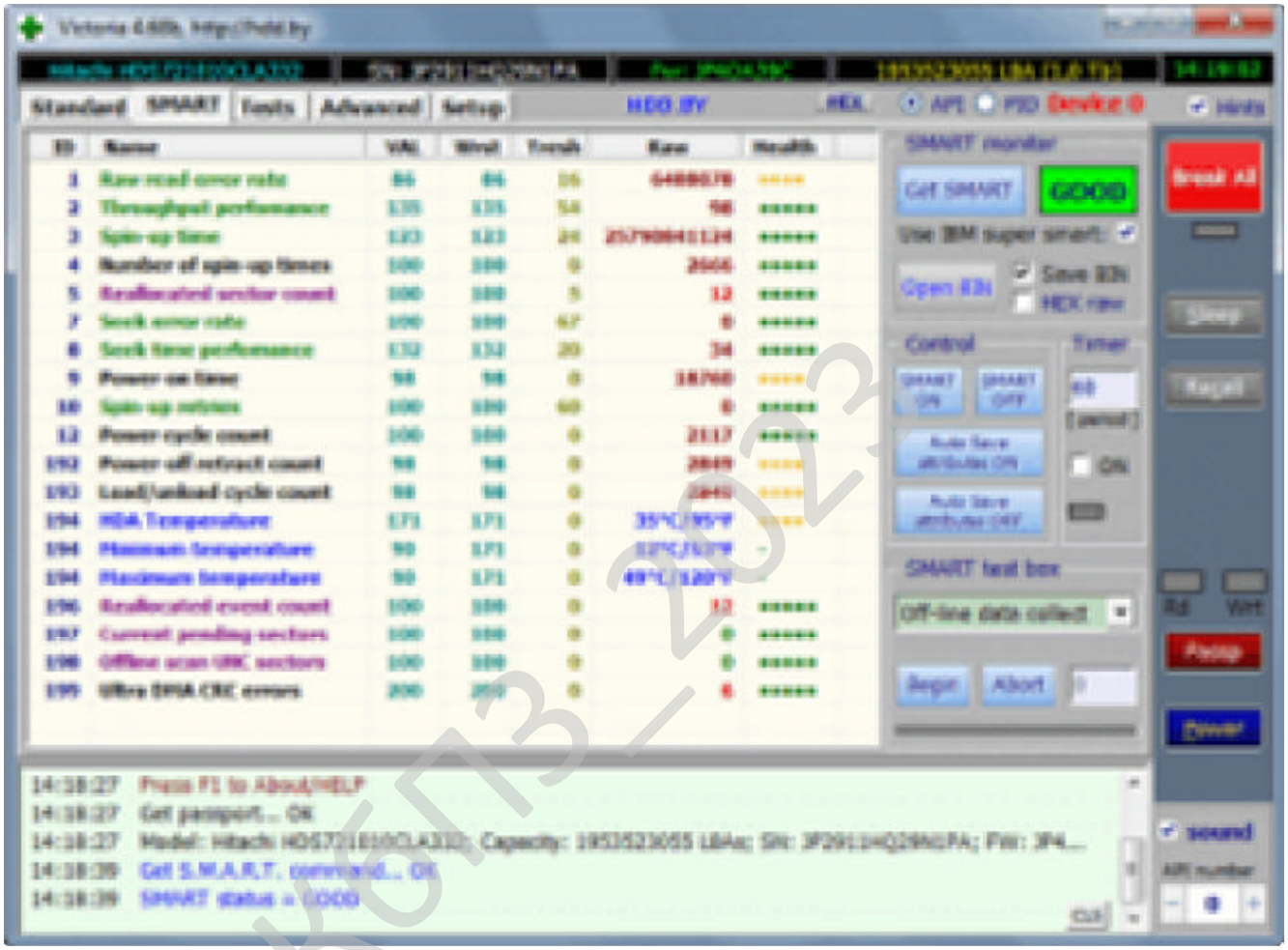


Рисунок 2.1 – SMART-монітор з підтримкою USB HDD та SSD

Victoria аналізує SMART-дані, і виводить стан кожного атрибуту, що наочно показує, наскільки добре вінчестер себе почуває. Перехід графіка до червоної зони свідчить, що накопичувач вичерпав свій ресурс. У 2018 році до програми додано нові атрибути, характерні для SSD-дисків.

Перегляд, аналіз, читання та запис внутрішніх SMART-журналів HDD та SSD NEW

Накопичувачі інформації у процесі свого життєвого циклу ведуть внутрішні журнали помилок, у тому числі можна дізнатися багато цікавого. Однак ці журнали практично недоступні користувачеві через малу кількість програм, які вміють адекватно інтерпретувати їх.

Програма Victoria здатна отримати доступ до SMART-журналів та показати їх вміст у зручній формі. Також можна ознайомитися з результатом вбудованих SMART-тестів, які можна запустити в цій же програмі, а потім переглянути результат у журналі.

Зміна заводської конфігурації HDD NEW

Багато накопичувачів HDD підтримують функцію **Device configuration overlay** (**DCO**), за допомогою якої можна вимикати непотрібні функції, включати раніше вимкнені або змінювати доступний об'єм. Таким чином, можна підвищити інформаційну безпеку або адаптувати вінчестер до нестандартних умов.

Програма Victoria здатна отримати доступ до енергонезалежної пам'яті DCO, відобразити її вміст у зручній формі, відредагувати на свій розсуд і знову записати в накопичувач.

Режим роботи з накопичувачем через порти контролера

Крім роботи через API Windows, програма має режим PIO: звернення до тестованого IDE/SATA накопичувача безпосередньо через порти, в обхід Windows та BIOS. При цьому Victoria сама повністю реалізує фізичний протокол роботи з обладнанням, що дозволяє отримати доступ до частково несправних пристроїв, коли драйвера безсилі.

PIO-режим призначений для використання у професійних цілях, і потребує спеціальних знань. Також даний режим застосовується для особливо точних вимірювань характеристик пристроїв і за наявності несправностей накопичувача, що тестується.

Побудова графіків при повному посекторному скануванні

Графічний метод дуже наочний, і доповнює основний режим.

У програмі Victoria застосований середній спосіб побудови графіка. У справного HDD графік являє собою лінію, що плавно спадає вниз, на якій зазвичай видно сходинки.

У SSD ідеальний графік – пряма лінія. Наявність провалів швидкості свідчить про знос мікросхем пам'яті або особливості роботи контролера SSD.

Встановлення паролів на накопичувачі з інтерфейсом USB

Всі сучасні HDD і SSD підтримують так звані функції безпеки, серед яких – можливість встановлення паролів для захисту доступу до даних. Однак, розміщення накопичувача в USB-контейнері зазвичай робить ці функції недоступними для більшості пристроїв.

Victoria, працюючи з накопичувачем через USB-міст, має набагато менше обмежень, ніж утиліти від виробників HDD.

Стирання інформації без можливості її відновлення

Victoria має можливість очищення носія від інформації на максимальній швидкості для цього накопичувача.

HDD Scan

Основні можливості програми:

– Запуск і зупинка шпиндельного двигуна.

– Керування шумовими характеристиками вінчестера.

– Перевірка поверхні диска в трьох режимах: Verify, Read, Erase.

– Перегляд інформації S.M.A.R.T.

Вибір вінчестера

У меню, що розкривається, що перебуває в розділі Source Disk, можна вибрати вінчестер, з яким буде працювати програма. Унизу вказується основна інформація про обраний вінчестер: модель (Model), версія мікропрограми (Firmware), серійний номер (Serial) і кількість доступних фізичних секторів (LBA).

Перегляд S.M.A.R.T.

Для перегляду параметрів S.M.A.R.T. необхідно нажати кнопку S.M.A.R.T., що перебуває у верхній частині вікна програми.

Опис стовпців таблиці:

– Treshold – граничне значення атрибута. Використовується для порівняння зі значенням параметра (Value).

– RAW – значення атрибута в шістнадцятковій системі вирахування (за винятком температури (Temperature), значення якої програма вказує в градусах Цельсія).

– Worst – найнижче значення атрибута за увесь час.

– Value – значення атрибута. Може перебувати в діапазоні від 1 до 255 (чим число вище – тим краще).

– Description – найменування атрибута.

– Attribute – номер атрибута.

Колір індикатора означає стан атрибута:

– Червоний – сильне відхилення атрибута від норми.

– Жовтий – відхилення атрибута від норми.

– Зелений – атрибут у нормі.

При цьому отримана інформація автоматично зберігається у файл S.M.A.R.T.txt, що перебуває в каталозі із програмою.

Інтерфейс

Інтерфейс програми виконаний повністю англійською мовою. Проте, програма дуже проста у використанні навіть для починаючих користувачів.

Перевірка поверхні диска

Параметри перевірки поверхні перебувають у розділі Process:

– Block size – кількість секторів в одному блоці. За замовчуванням – 256 секторів (не рекомендується змінювати).

– Start і Stop – запуск і зупинка перевірки.

– Command – вибір типу перевірки (Verify – верифікація, Read – повноцінне читання секторів, Erase – запис в усі сектори нулів). Вибір параметра Erase приведе до повної втрати всіх даних на диску!

– Start LBA і End LBA – сектор, з якого починається перевірка й сектор, на якому зупиняється перевірка. Якщо потрібно перевірити весь диск, даний параметр змінювати не треба.

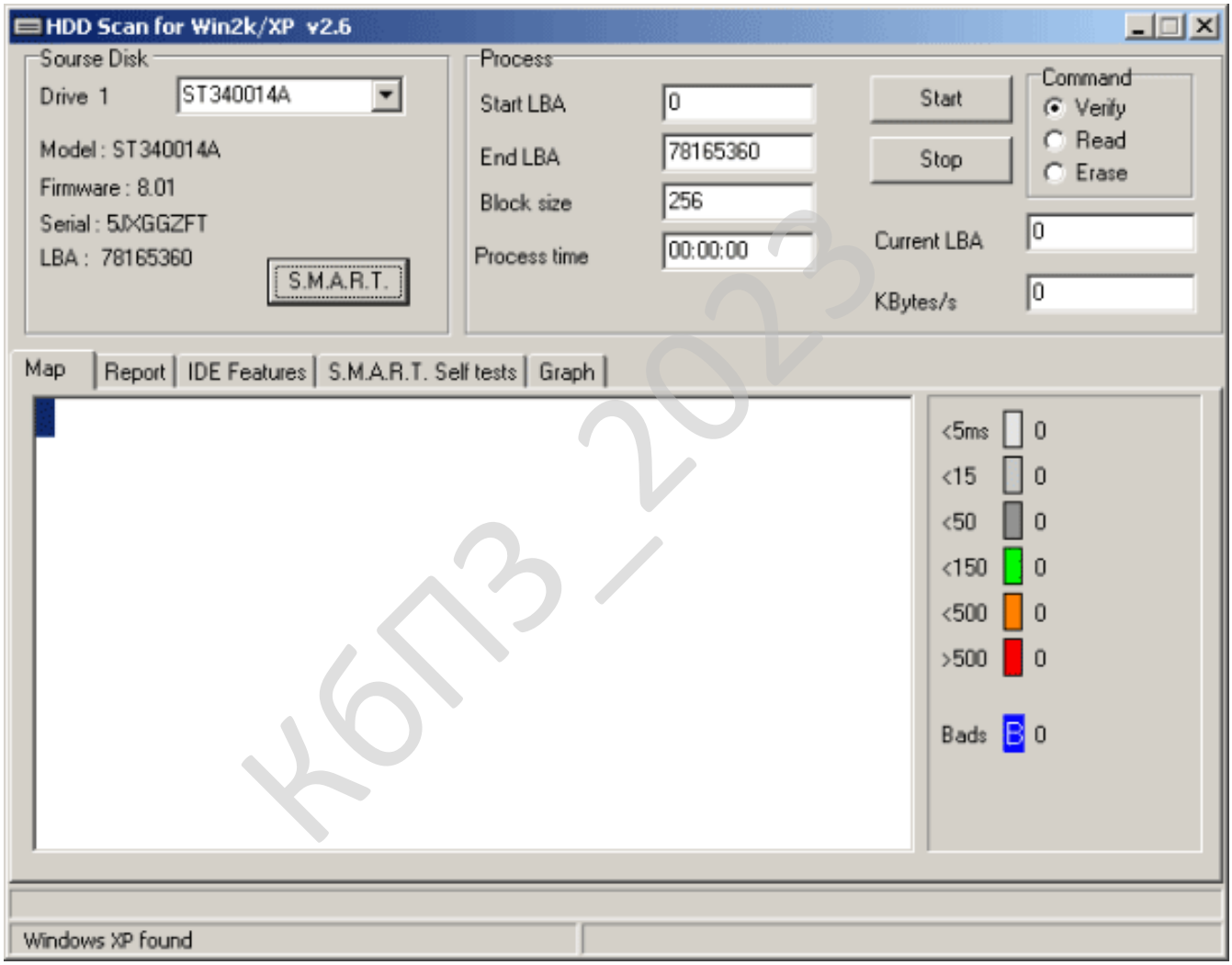


Рисунок 2.2 – Інтерфейс користувача HDDScan

Під час перевірки диска вказується минулий час (Process time), поточний сектор (Current LBA) і швидкість читання (Kbytes/s).

У вкладці Map розташовується карта диска. Ліворуч перебуває покажчик, у якому позначені кольори прямокутників на карті, залежно від часу доступу. Синій прямокутник (Bads) показує bad-блоки (ушкоджені сектори). Праворуч від кольорових прямокутників зазначена кількість секторів з таким часом доступу, знайдених при перевірці диска.

У вкладці Report розташовується текстовий звіт про перевірку. У ньому вказується інформація про сектори згодом доступу більше 50 мс (зелені, жовтогарячі й червоні прямокутники), а також про bad-блоки.

На вкладці Graph розташовується графік швидкості читання диска. По осі X розташовуються номери секторів, по осі Y – швидкість читання.

Керування шумовими характеристиками

На вкладці IDE Features, у лівому верхньому розділі, можна змінювати рівень шуму, видаваного вінчестером. Робити це можна тільки в тому випадку, якщо така можливість дозволена виробником (відображається зелений індикатор і напис Enabled ліворуч від нього). Варто врахувати, що зниження шуму спричиняє зниження швидкості роботи вінчестера.

Запуск і зупинка шпиндельного двигуна

На вкладці IDE Features, у розділі Spindle Start/Stop можна запускати й зупиняти шпиндельний двигун вінчестера. Не рекомендується зупиняти шпиндельний двигун на системному диску.

SIGuardian

Існує велика кількість програм, що контролюють S.M.A.R.T., це може бути спеціально спрямована програма (Drive Health, SIGuardian), або програма, що містить контроль параметрів S.M.A.R.T. як додаткову функцію. Програма надає можливість стежити за практично всіма атрибутами S.M.A.R.T., має приємний інтерфейс, і має велику кількість налаштувань.

Загальні відомості про диски

У лівій половині зазначені: технічні характеристики, такі як обсяг диска, кількість циліндрів, головок і т.п.; режим роботи диска в даний момент (PIO,

multiword DMA, UDMA); підтримувані режими роботи диска (тільки в Розширеному режимі).

У правій половині показується логотип фірми-виробника жорсткого диска й нижче – загальна інформація про диск: модель диска, серійний номер диска, дата/ревізія прошивання мікропрограми.

Загальні відомості S.M.A.R.T.

Закладка "S.M.A.R.T." показує загальну інформацію про стан диска на основі S.M.A.R.T. атрибутів або S.M.A.R.T. – інформацію:

– Найближчу прогнозовану дату T.E.C. (ThresholdExceedCondition) – тобто дату, коли за прогнозами SIGuardian один з S.M.A.R.T. атрибутів досягне граничного (критичного) значення.

– Дату початку моніторингу S.M.A.R.T. – тобто дату, коли ви почали контроль за станом диска за допомогою SIGuardian. Найчастіше, це дата першого запуску SIGuardian.

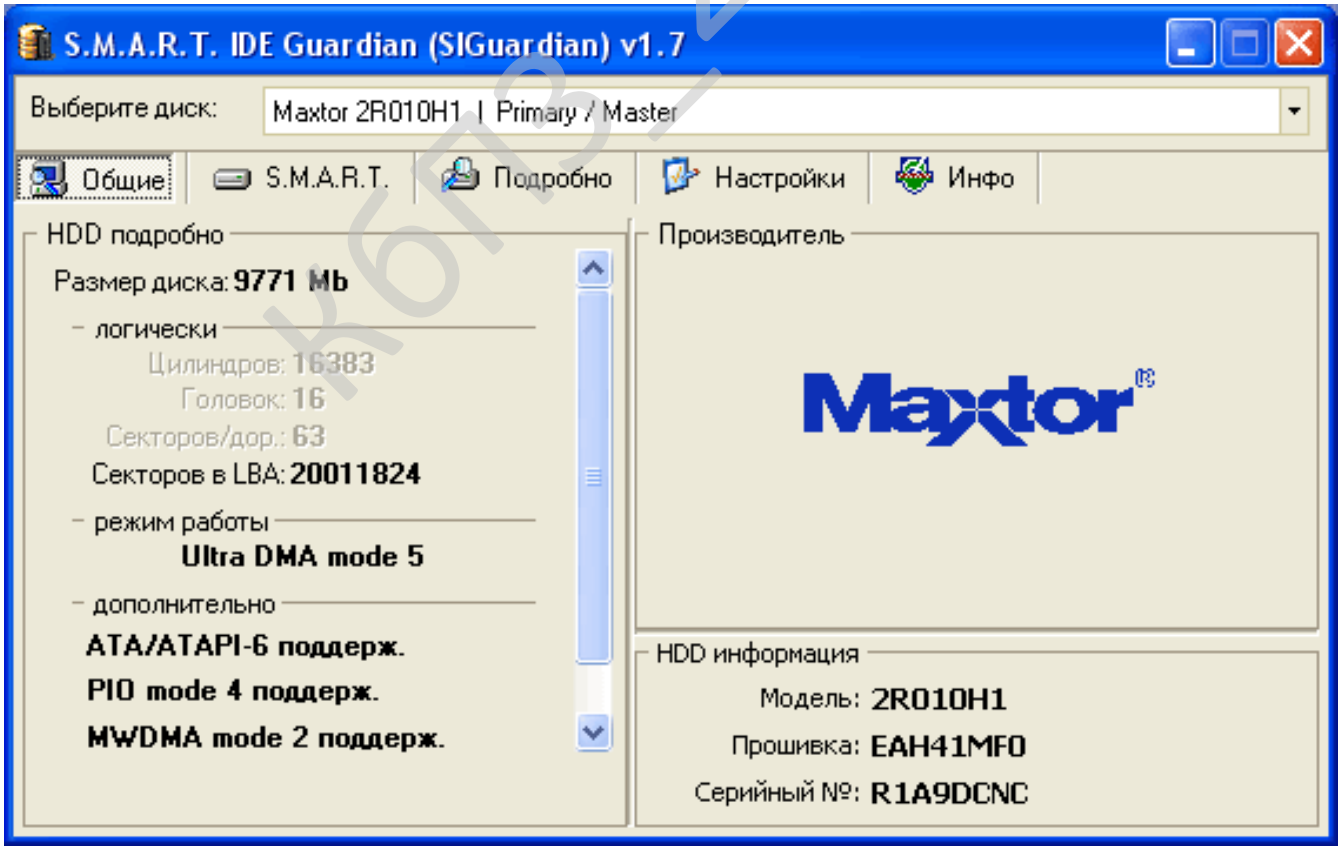


Рисунок 2.3 – Інтерфейс користувача SIGuardian

S.M.A.R.T. докладно

Закладка "Докладно" призначена для відображення повної інформації про S.M.A.R.T.-атрибутах диска. Вона показує:

– Raw – "чисте" значення атрибута – просто числове значення атрибута в чистому, неопрацьованому виді.

– Worst – гірше значення атрибута – саме гірше (мінімальне) значення, що даний атрибут приймав за увесь час життя жорсткого диска. Може використовуватися чисто в ознайомлювальних цілях.

– T.E.C. – Threshold Exceeds Condition – передбачувана дата, коли даний атрибут досягне граничного значення, інакше кажучи, дата можливого виходу з ладу диска. Прогноз цієї дати робиться на основі показника "швидкості падіння атрибута", тому не дивуйтеся сильним коливанням дати відразу після зміни атрибутів S.M.A.R.T.

– Threshold – граничне (критичне) значення атрибута – значення, величину якого виробник жорсткого диска вважає критичної й при досягненні якого цілком імовірний вихід диска з ладу.

– Value – значення атрибута – поточне значення даного атрибута S.M.A.R.T.

– 1/month – швидкість падіння атрибута – на скільки пунктів на місяць упало значення атрибута. Цей коефіцієнт обчислюється автоматично при будь-якій зміні атрибутів S.M.A.R.T. для кожного атрибута окремо. Обчислення виробляється щодня, тому ставитесь нормально до коливань цього показника, особливо відразу після зміни атрибута.

– Attribute name – Графічне відображення значення атрибута. При наведенні покажчика миші на нього показується у вікні спливаючої підказки більше докладний текстовий опис змісту цього атрибута.

Налаштування

Закладка "Налаштування" призначена для самостійного налаштування користувачем параметрів SIGuardian для роботи на комп'ютері. Якщо ви не

вважаєте себе досвідченим користувачем, рекомендуємо скористатися "Майстром налаштування" – він допоможе вам вибрати найбільш підходящі параметри роботи.

Основні й найбільш важливі налаштування:

– Hibernate on overheat temperature – якщо температура HDD перевищує встановлене значення, комп'ютер переходить у режим hibernate.

– WiseControl – інформація тільки про значні зміни (погіршеннях) параметрів S.M.A.R.T.

– При завантаженні перевірка й вихід – відзначте цей режим, якщо ви хочете щоб SIGuardian перевіряв стан S.M.A.R.T. тільки при завантаженні операційної системи.

– Звіти на e-mail – уведіть тут адресу електронної пошти, на який SIGuardian повинен посилати повідомлення. Ви не повинні бачити ніяких повідомлень при роботі в цьому випадку.

– Опитування S.M.A.R.T. – установите тут період опитування S.M.A.R.T. при роботі SIGuardian у фоні.

– Режим роботи – Звичайний або Розширений – Звичайний режим – основної для користувачів. У цьому режимі SIGuardian показує значення атрибута, граничне значення й T.E.C., швидкість падіння атрибута. На закладці "Загальне" Ви не побачите інформації про підтримуваний диском режимах роботи (передачі даних). У розширеному режимі додатково показують Гірше й Чисте (Raw) значення атрибута й повну інформацію про диск на закладці "Загальне".

– Включити контроль S.M.A.R.T. – при вимиканні цього режиму SIGuardian не буде перевіряти цей диск (або всі диски) на значення атрибутів S.M.A.R.T.

– Загальні налаштування для всіх дисків – SIGuardian буде використовувати загальні налаштування для всіх дисків у комп'ютері. Вони включають: контроль S.M.A.R.T., період опитування S.M.A.R.T. і адреса

електронної пошти для повідомлень. Ви можете встановити загальні або індивідуальні для кожного диска параметри.

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 моніторингу стану жорсткого диску з використанням технології SMART.

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. – це технологія внутрішньої оцінки стану диска, і механізм пророкування можливого виходу з ладу жорсткого диска. Важливо відзначити те, що технологія в принципі не вирішує виникаючих проблем, вона здатна лише попередити про вже виниклу проблему або про ту, що може бути в найближчому часі.

При цьому потрібно також сказати, що технологія не в змозі пророчити абсолютно всі можливі проблеми й це логічно: вихід електроніки в результаті стрибка напруги, псування головок і поверхні в результаті удару й т.п. ніяка технологія пророчити не в силах. Передбачувані лише ті проблеми, які пов'язані з поступовим погіршенням яких-небудь характеристик, рівномірною деградацією яких або компонент.

S.M.A.R.T. являє собою набір міні-підпрограм, які є частиною мікрокоду накопичувача й визначають підтримувані діагностичні функції. Найпоширеніші серед них:

– набір атрибутів, що відбивають стан окремих параметрів накопичувача (до 30);

– внутрішні тести накопичувача (self– test);

– журнали S.M.A.R.T. (помилок, загального стану, дефектних секторів і т.п.).

У даний момент не існує офіційної документації або стандарту на технологію S.M.A.R.T. У зв'язку із цим, виробники не публікують повні характеристики й підтримувані функції S.M.A.R.T. у своїх накопичувачах. Обов'язковий мінімум описаний в останньому стандарті ATA/ATAPI-6.

Розвиток технології S.M.A.R.T.

Історія технології S.M.A.R.T. не так уже й багата подробицями:

– S.M.A.R.T. I передбачав моніторинг основних життєво важливих параметрів і запускався тільки після команди по інтерфейсу.

– в S.M.A.R.T. II з'явилася можливість фонової перевірки поверхні, що виконувалася накопичувачем автоматично під час "холостого ходу"; з'явилася функція журналювання помилок.

– в S.M.A.R.T. III уперше з'явилася не тільки функція виявлення дефектів поверхні, але й можливість їхнього відновлення "прозоро" для користувача й багато інших нововведень.

Відомо, що першими розробили основи й запропонували цю технологію спільно Western Digital, Seagate і Quantum. Після цього їх уже підтримали такі компанії як IBM, Maxtor і Samsung. Hitachi взяла участь у розвитку технології S.M.A.R.T. уже на стадії розробки S.M.A.R.T. II, першими запропонувавши методику повної самодіагностики накопичувача (extended self-test).

У цей час виробники жорстких дисків готуються прийняти до використання новий варіант технології S.M.A.R.T. – "1024 S.M.A.R.T.", характерною рисою якого буде помітно більший розмір журналів, повсюдне використання мультисекторних журналів, більше точні алгоритми аналізу показань убудованих у накопичувач сенсорів (термодатчики, сенсори ударів, і т.п.) і багато чого іншого. От кілька нових функцій:

– введення алгоритму аналізу температурного режиму накопичувача;

– введення обмеження по мінімальній і максимальній температурі в робочому стані;

– введення лічильника загальної кількості записаних секторів протягом життєвого циклу накопичувача;

– введення лічильника запусків внутрішніх алгоритмів відновлення (recovery counters).

Головним же плюсом можна вважати введення нових атрибутів, які дозволять контролювати стан і робочі характеристики по кожній з головок читання/запису:

– відносна стійкість (стабільність "польоту") головки;

– виправлення помилок читання (з "схованими" повторними спробами);

– автоматичний перерозподіл дефектних ділянок поверхні при операціях запису;

– лічильник-накопичувач G-List для обліку кількості прийнятих ударних навантажень;

– лічильник-накопичувач S-List для обліку загальної кількості "програмних" помилок.

Програми, що відображають стан S.M.A.R.T.-атрибутів, працюють за наступним алгоритмом:

– Перевіряють наявність підтримки технології S.M.A.R.T. накопичувачем.

– Подають у накопичувач команду запиту S.M.A.R.T.-таблиць.

– Одержують таблиці в буфер додатка.

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

– Зіставляють стандартизовані номери атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Виводять числові значення в зручному для сприйняття виді.

– 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 виводять загальний стан пристрою.

3.2 Розробка структурної схеми

Перш ніж перейти до структурної схеми розглянемо детально як працює технологія S.M.A.R.T.

Технологія S.M.A.R.T. – Self-Monitoring, Analysis and Reporting Technology (від англ. "Технологія Самодіагностики, Аналізу й Звіту") – була розроблена для підвищення надійності й схоронності даних на жорстких дисках. У більшості випадків, S.M.A.R.T.-сумісні пристрої дозволяють пророчити появу найбільш імовірних помилок і, тим самим, дають користувачеві можливість вчасно зробити резервну копію даних і/або повністю замінити накопичувач до виходу його з ладу.

Різновиди тестів S.M.A.R.T.

Офіційно документовані три види внутрішніх тестів, однак ще існує набір так званих "активних" тестів, функціональні особливості яких різні в різних виробників і для широкої публіки не документовані.

Таблиця 3.1 – Різновиди тестів

№	Назва тесту	off-line	captive
1	Off-line collection	+	-
2	Short Self-test	+	+
3	Extended Self-test	+	+
4	Drive Activity test #1..#4	+	+

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 останіх 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.

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.

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).

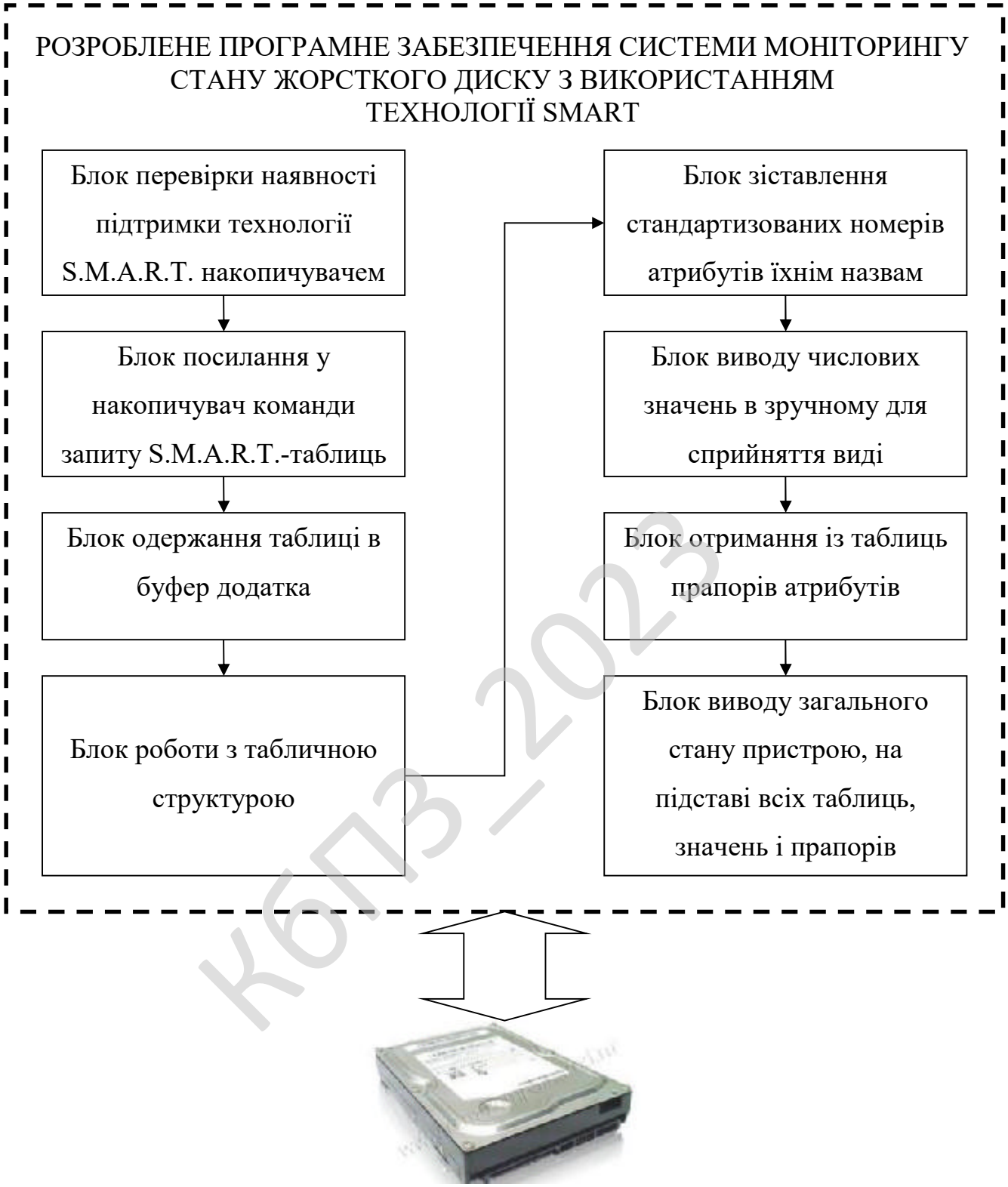
Розмір: максимум 31 журнал по 16 секторів (253,952 байт).

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.



Жорсткий диск який тестується

Рисунок 3.1 – Структурна схема системи

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.

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

Структурна схема складається з наступних блоків:

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

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

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

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

– Блок виводу числових значень в зручному для сприйняття вигляді.

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

– Блок одержання таблиці в буфер додатка.

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

3.3 Розробка функціональної схеми

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.

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.

Таблиця 3.2 – Значення атрибутів

ID	Назва атрибута
0	= атрибут не використовується
1	Raw Read Error Rate
2	Throughput Performance
3	Spin Up Time
4	Start/Stop Count
5	Reallocated Sector Count
6	Read Channel Margin
7	Seek Error Rate
8	Seek Time Performance
9	Power-On Hours Count
10	Spin Retry Count
11	Recalibration Retries
12	Device Power Cycle Count
13	Soft Read Error Rate
??	Emergency Re-track (Hitachi)
??	ECC On-The-Fly Count (Hitachi)
96	? (Maxtor)
97	? (Maxtor)
98	? (Maxtor)
99	? (Maxtor)
100	? (Maxtor)
101	? (Maxtor)
191	G-Sense Error Rate
192	Power-Off Retract Cycle
193	Load/Unload Cycle Count
194	Temperature
195	? (Quantum AS, Seagate, Maxtor)
196	Reallocation Events Count

Продовження таблиці 3.2

ID	Назва атрибута
197	Current Pending Sector Count
198	Uncorrectable Sector Count
199	UltraDMA CRC Error Rate
200	Write Error Rate (в WD – MultiZone Error Rate)
201	TA Counter Detected
202	TA Counter Increased
203	? (Maxtor)
204	? (Maxtor)
205	? (Maxtor)
206	? (Maxtor)
207	? (Maxtor)
208	? (Maxtor)
209	? (Maxtor)
220	Disk Shift
221	G-Sense Error Rate (в Hitachi – Shock Sense Error Rate)
222	Loaded Hours
223	Load/Unload Retry Count
224	Load Friction
225	Load/Unload Cycle Count
226	Load-in Time
227	Torque Amplification Count
228	Power-Off Retract Count
229	? (IBM DTTA, thanx to Vladislav Shaklein)
230	GMR Head Amplitude
231	Temperature
240	Head Flying Hours (Hitachi)
250	Read Error Retry Rate

– 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

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.).

Функціональна схема складається з наступних блоків:

1. 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.

2. 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.

3. Блок автономного сканування поверхні.

4. 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.

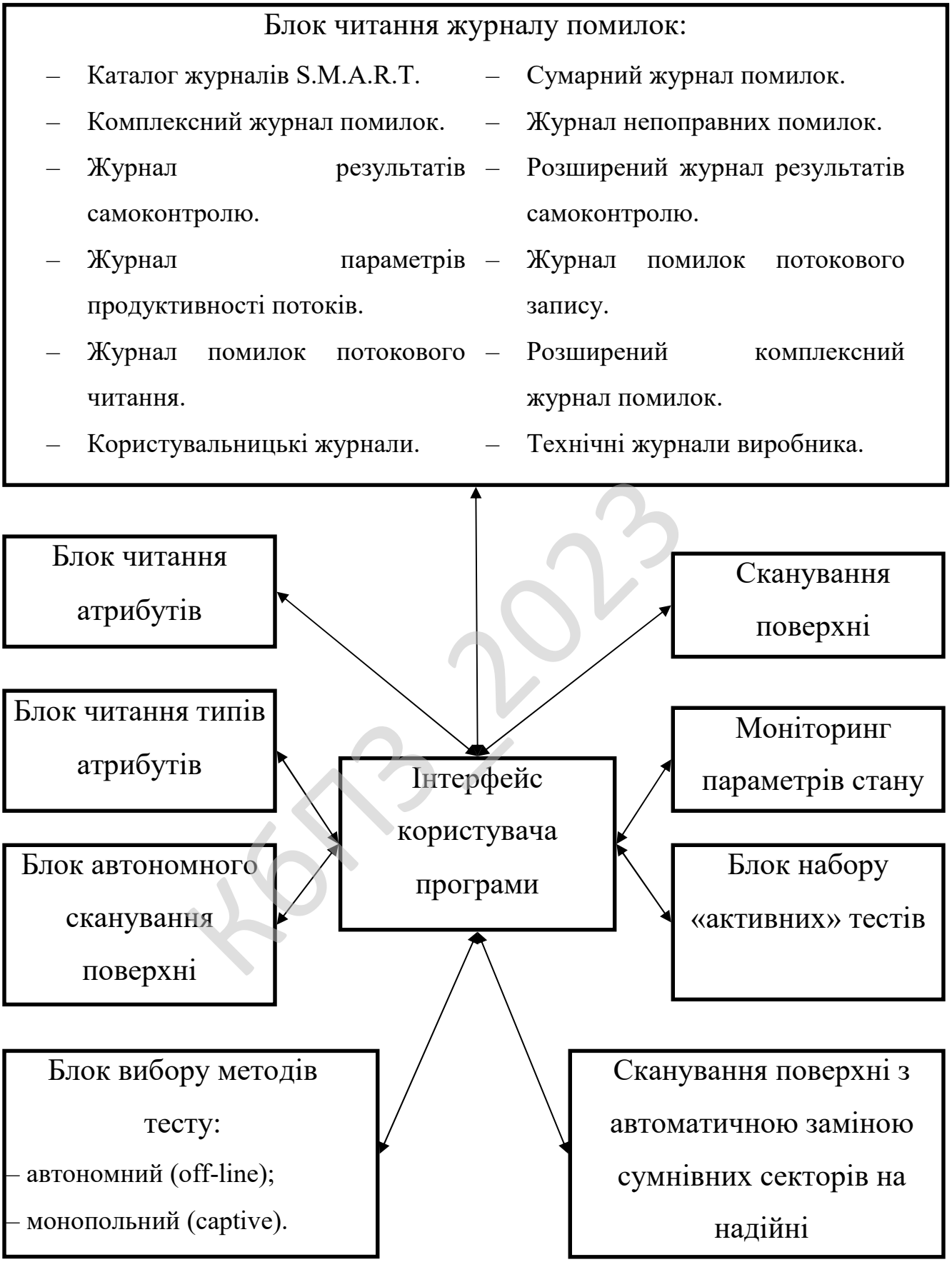


Рисунок 3.2 – Функціональна схема системи

– Час розкручування шпинделя.

– Кількість циклів запуск/останов шпинделя.

– Кількість перепризначених секторів.

– Запас каналу читання.

– Частота появи помилок позиціонування БМГ.

– Середня продуктивність операцій позиціонування БМГ.

5. Блок вбудованих функцій 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 взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. Після початку роботи ПЗ ми потопляємо до головного вікна програми з якого можна використовуючи бібліотеку для роботи зі S.M.A.R.T використовувати функції обробки атрибутів S.M.A.R.T.

Через обробник помилок та перевірку наявності підтримки технології SMART отримати таблиці прапорів атрибутів та провести: зіставлення номерів атрибутів їхнім назвам; аналіз загального стану пристрою; провести пошук несправностей.

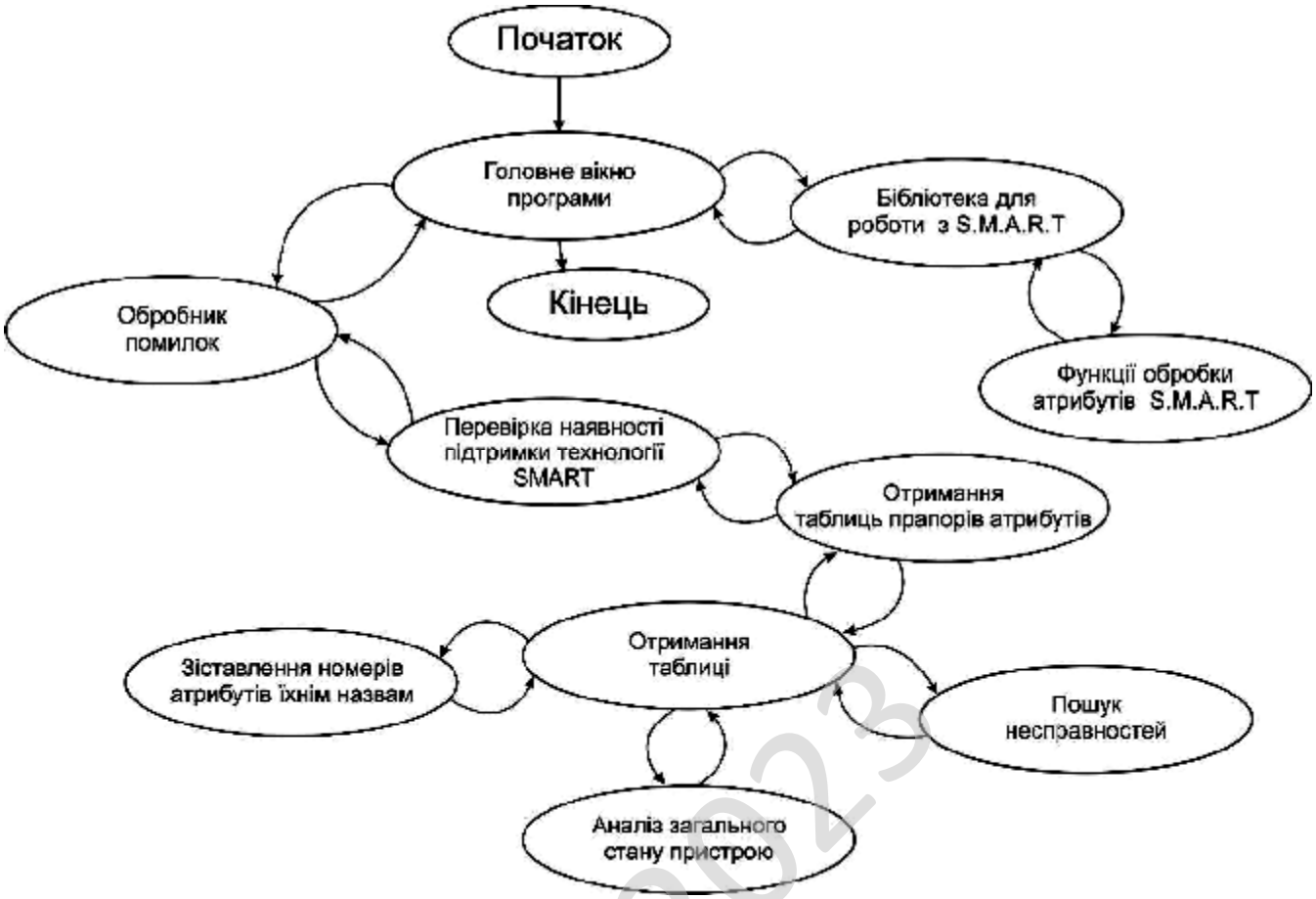


Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Розглянемо її роботу яка складається з виконання наступних кроків. Спершу відбувається ініціалізація динамічних бібліотек ПЗ, виділення пам`яті ПЗ та виведення на екран головного вікна ПЗ з ініціалізацією регістрів контролерів жорстких дисків.

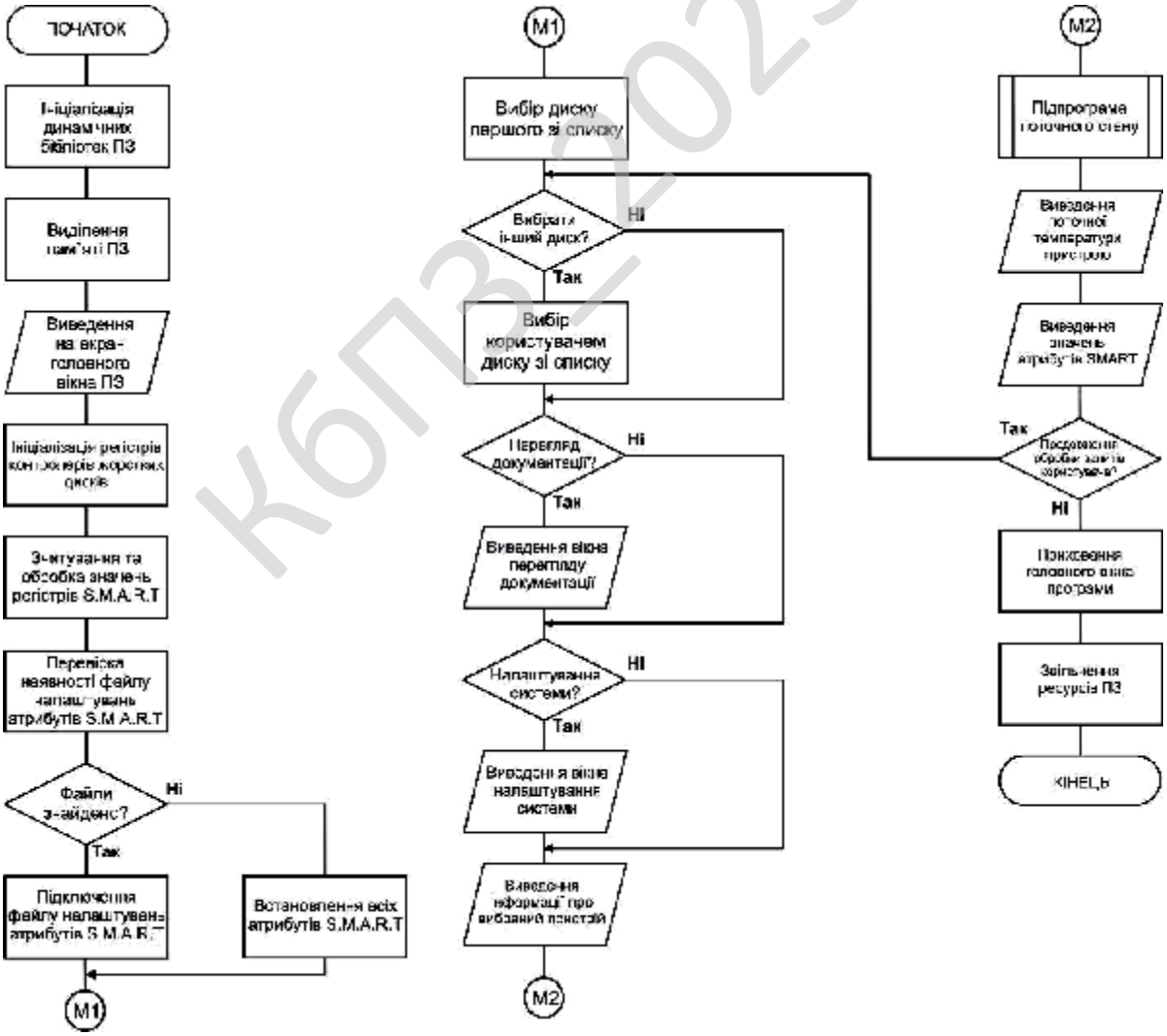


Рисунок 4.1 – Блок-схема основної програми

Далі проводиться зчитування, обробка значень регістрів S.M.A.R.T, перевірка наявності файлу налаштувань атрибутів S.M.A.R.T та при наявності проводиться підключення файлу налаштувань атрибутів S.M.A.R.T якщо ні – встановлення всіх атрибутів S.M.A.R.T. Після цих дій проходе вибір диску першого зі списку та запити на вибір іншого диску, перегляд документації та налаштування системи з виконанням цих дій при необхідності.

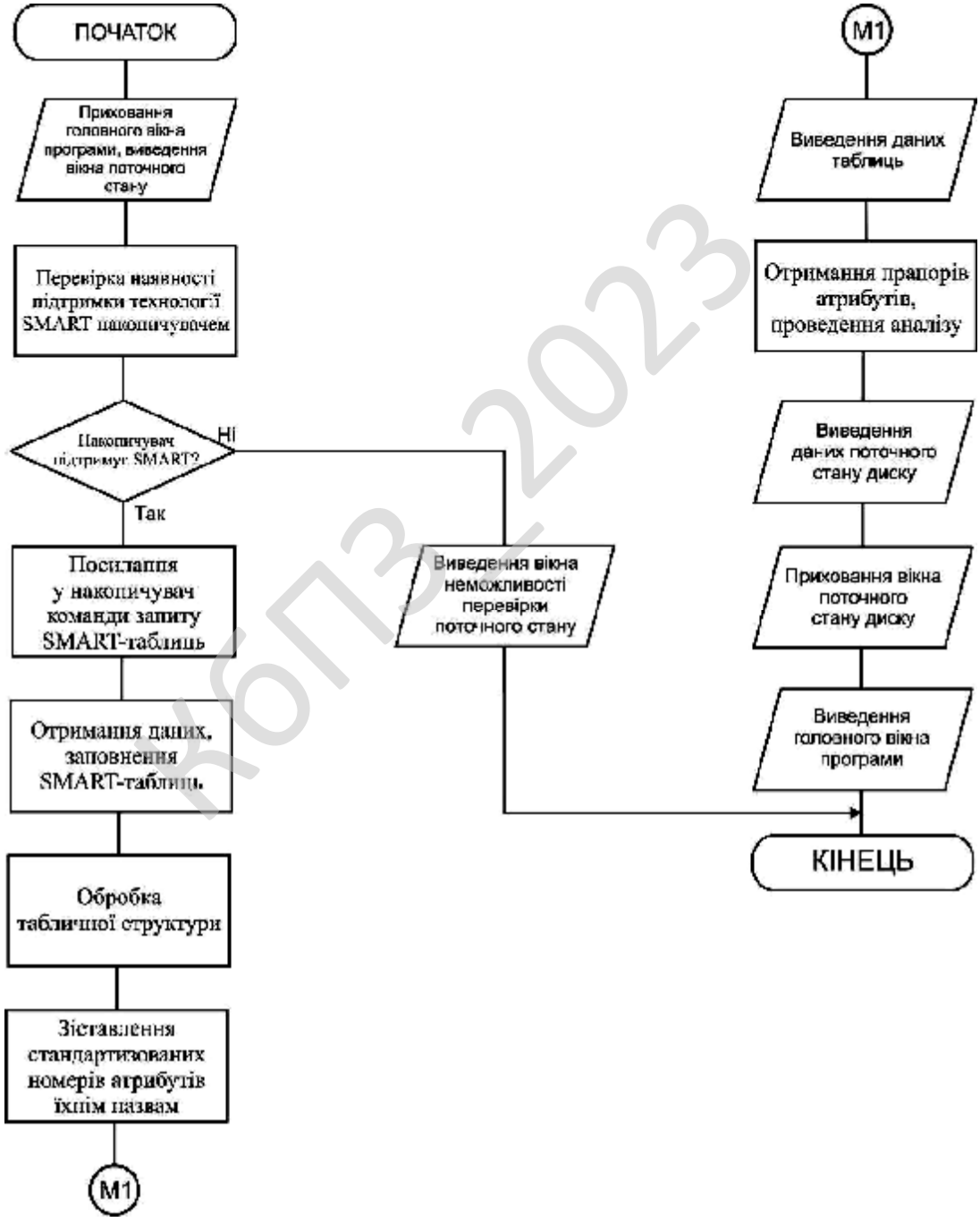


Рисунок 4.2 – Блок-схема роботи підпрограми

Далі проходе виведення інформації про вибраний пристрій та виконання підпрограми поточного стану (рис. 4.2) з подальшим виведенням поточної температури пристрою та значень атрибутів SMART. При завершенні роботи програми проводиться приховання головного вікна програми та звільнення ресурсів ПЗ.

Опис розробленого ПЗ. 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. Наприклад, Hard Drive Inspector 1.6 коштує $ 29,95; Active SMART 2.4 – $ 24,95; SiGuardian 1.6 – $ 14.

Я в магістерському проекті розробив ПЗ використовуючи вбудовані засоби ОС Windows і за допомогою мови Object Pascal.

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.

Розглянемо розроблений код. Для початку необхідно створити 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.

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;
```

Розглянь 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.

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'
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними

установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME. У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

SEED являє собою мережу Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем. Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»). Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини. Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля. В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім поєднується в помірковано комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів.

SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio).

Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене магістерське ПЗ, головне вікно ПЗ зображується на рисунку 5.1. Як можна побачити вікно складається з наступних частин:

1. Меню. З самого верху йде меню розробленого ПЗ. Воно повністю повторює функціонал всієї програми. В ньому присутні наступні розділи:

– Файл.

– Дані.

– Налаштування.

– Параметри.

– Шаблон.

– Довідка.

2. Обрання HDD. Після меню йде розділ обрання жорсткого диску. Після обрання в автоматичному режимі перевіряється наявність технології SMART. Якщо вона присутня проходе тестування.

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.

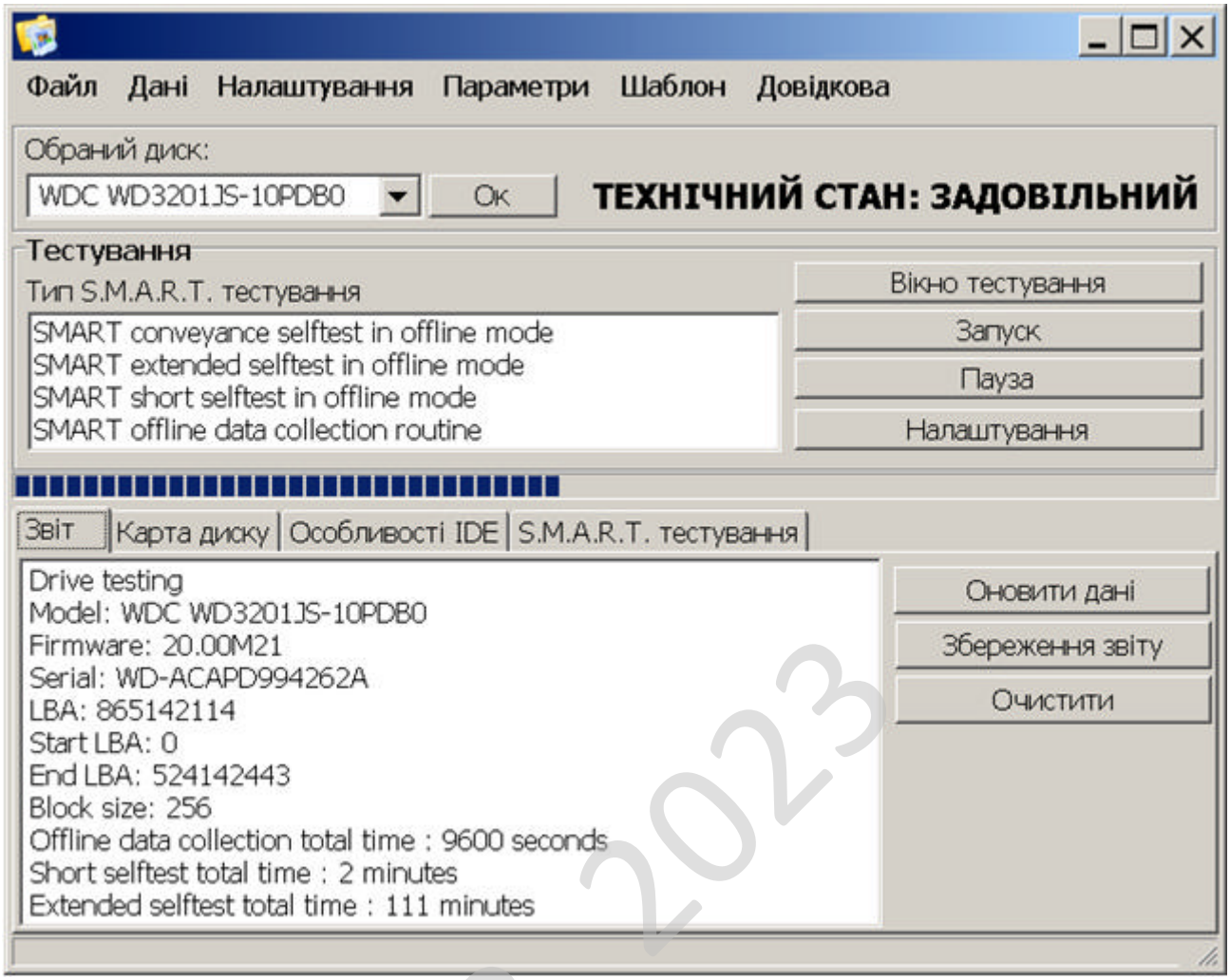


Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено вікно налаштування атрибутів S.M.A.R.T. Вікно дозволяє керувати всіма параметрами жорсткого диску. Це дозволяє повністю відображати його поточний стан та передбачити можливі помилки у майбутньому, із застосуванням різних методів.

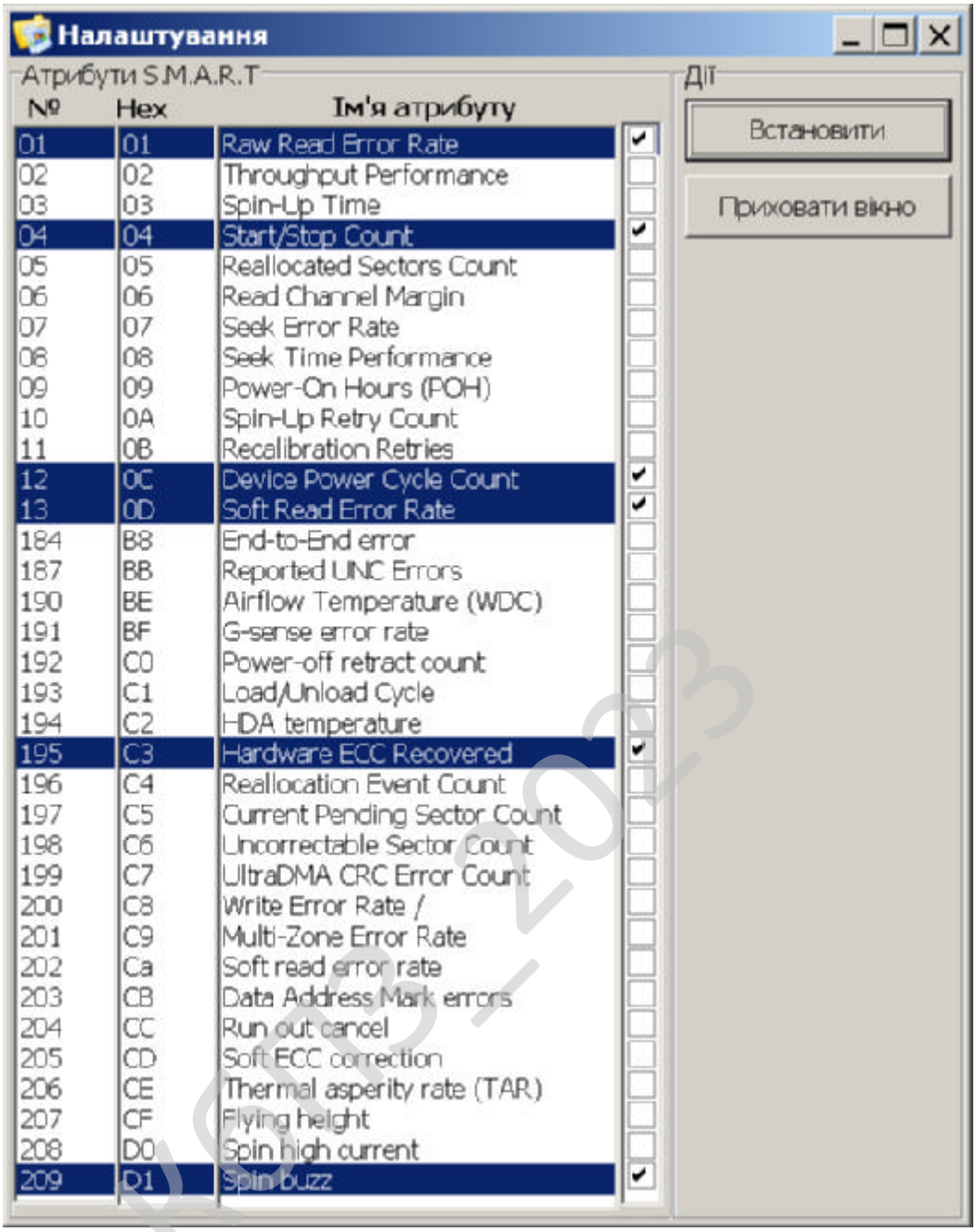


Рисунок 5.2 – Вікно налаштування атрибутів S.M.A.R.T.

На рисунку 5.3 зображена форма авторського права.

Обраний тип ліцензії – умовно безкоштовне розповсюдження.

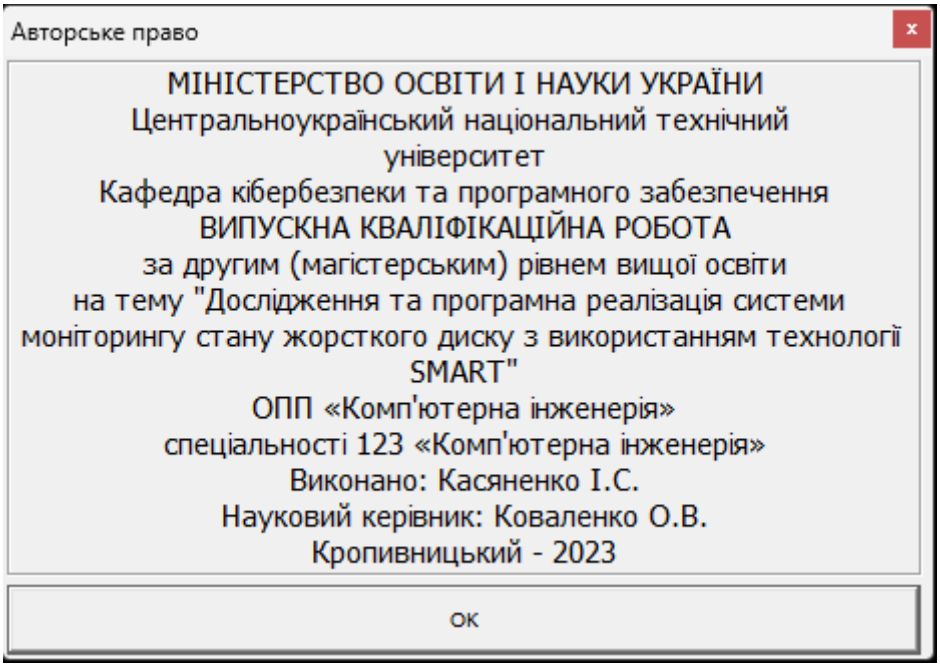


Рисунок 5.3 – Авторське право

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу стану жорсткого диску з використанням технології SMART.

Метою розробки є дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

Об'єктом дослідження є процес моніторингу стану жорсткого диску з використанням технології SMART.

Предметом дослідження є методи моніторингу стану жорсткого диску з використанням технології SMART.

Методи дослідження базуються на методах схемотехніки, теорії надійності, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу стану жорсткого диску з використанням технології SMART.

– Розроблено вітчизняний продукт моніторингу стану жорсткого диску з використанням технології SMART, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

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	22
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)	–	3
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.	–	22000
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	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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

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

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

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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 70 = 117 \text{ люд/день.}$$

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	117	Ф 7.1-7.4
Впровадження	13	Д13
Всього	158	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{158 \cdot 1}{60 - 5} = 2{,}8 \text{ ставки.}$$

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	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п	2,5	280	700	11,67
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	53,33

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф^{с}_{др} = \frac{53 \cdot 3}{1,2} = 132,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$ = 132,5/(60·8) = 0,27 ставки.

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2012 R2, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	18132	54396
Продакт-менеджер	0,25	14000	10500
Інженер-програміст	2,8	18000	151200
Інженер-електронщик	0,27	14000	11340
Інженер-системотехнік	0,25	14000	10500
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 6,32	-	$Ф_{роб}$ = 301086

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.

$$З_{сд} = \frac{301086}{6{,}32 \cdot 60} = 794 \text{ грн.}$$

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

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

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

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8м^2$. З урахуванням цього:

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

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

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

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу hotline за 26.10.23 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11186
Системний блок		6490
Процесор	Intel Core i7-4770 (4 (8) ядра по 3.4 - 3. GHz), 8 MB Smart Cache	-
Системна плата	Intel Q85 Express Chipset, 4x USB 3.0, 6 USB 2.0, 4x Audio, 1x VGA, 2x DisplayPor 1x COM-порт, 1x LAN (RJ-45), 2x PS/2	-
Відеокарта	nVidia GeForce GT 730, 2 GB GDDR3, 64-bi	-
Жорсткий диск	SSD диск Samsung 870 Evo-Series 250GI 2.5" SATA III V-NAND 3bit MLC (TLC (MZ-77E250BW)	-
Оперативна пам’ять	DDR3 4GB 1333 MHz CL9 SAMSUNG 1,5V (M378B5273CH0-CH9) два модулі	-
DVD-привод	Super Multi LG SATA DVD±RW R+22x/ 22x, RW+8x/-6x, DL+16x/-12x, RAM 12x SecurDisc, black (GH22NS40RBB)	-
Корпус	HP ProDesk 600 G1 Tower, 320W	-
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	240
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

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

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни. Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	11186	8948,8	98436,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	8948,8	98436,8
Копіюв. апарат	1	5965	540	5940
Всього	–	–	–	214803,6

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	214804	-	-
Всього по групі	214804	50	107402
Група 5, 6			
4. Вимірювальні пристрої	5190	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	28000	25	-
Всього по групі	33190	-	8297,5
Нематеріальні активи			
7. Нематеріальні активи	22000	10	2200
Разом	$К_р$ = 1677994		$А_р$ = 188299,5

Примітка: вартість автомобіля приймаємо рівною нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 794 \cdot 158/22 = 5700 \text{ грн.}$$

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

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

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

$$З_д = 5700 \cdot 10 \cdot 0{,}01 = 570 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(5700+570) = 1379 \text{ грн.}$$

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, %.

$$Г_{осп} = 5700 \cdot 15 \cdot 0{,}01 = 855 \text{ грн.}$$

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 прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,75 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0,75 = 150 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 27,6 грн./шт., DVD-R box – 32,15 грн./шт.

$$З_{М2} = 10 \cdot 27,6 = 276 \text{ грн.}$$

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.

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

$$З_М = (150 + 276 + 1702)/22 = 97 \text{ грн.}$$

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, %.

$$О_п = 5700 \cdot 15 \cdot 0,01 = 855 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 22 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.

$$A_м = 188300 \cdot 3/(22 \cdot 12) = 2140 \text{ грн.}$$

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

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

$$C_п = 5700+570+1379+855+97+855+2140 = 11596 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 11596 = 5798 \text{ грн.}$$

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	$З_о$	5700
2. Додаткова зарплата виконавців	$З_д$	570
3. Відрахування на соціальні потреби	$С_{оц}$	1379
4. Загальногосподарські витрати	$Г_{осп}$	855
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	855

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	2140
8. Повна собівартість програмного забезпечення	$С_п$	11596
9. Плановий прибуток	$П_р$	5798
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	17394
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3478,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	20872,8

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	20873
Всього капітальних витрат	–	20873

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	80520	28182
2. Витрати на електроенергію	$З_{ел}$	-	-
3. Витрати на амортизацію	$З_{ам}$	-	5218
Всього витрат за рік	$І$	80520	33400

Витрати на обслуговування:

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

де: $Т_р$ – кількість годин, що витрачається на обслуговування на рік, *год.* (приймаємо 1000 год.);

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

$$З_{р\ баз} = 1000 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 80520\ грн,$$

до:

$$З_{р\ нов} = 350 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 28182\ грн.$$

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

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

$$З_{ел\ баз} = З_{ел\ нов}$$

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	–	20873	–	5218,25
Всього відрахувань	-	–	20873	–	5218,25

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (17394 - 11596) \cdot 22 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 214804 + +0{,}25 \cdot 33190 + 0{,}1 \cdot 22000) \frac{\cdot 3}{12} = 80481 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - С_п) \cdot N_в}, \quad (7.26)$$

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

$$T_в = \frac{1677994}{(17394-11596) \cdot 21 \cdot 12/3} = 3{,}2 \text{ років.}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (80520\text{-}33400)\text{-}0{,}25 \cdot 20873 = 41902 \text{ грн.}$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	22
2. Повна собівартість розробленої програми	Грн.	11596
3. Ціна розробленої програми	Грн.	17394
4. Плановий прибуток від реалізації одної програми	Грн.	5798
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1677994
7. Загальний прибуток від реалізації програмної продукції	Грн.	127556
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	80481
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	3,2
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	20873
11.Величина економічного ефекту у користувача програмної продукції	Грн.	41902
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,44

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

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

$$T_{сп} = \frac{20873}{80520 - 33400} = 0{,}44 \; року.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – це: система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності;

Охорона праці є складовою частиною безпеки життєдіяльності [3,4].

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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.

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

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний покладено на професійні спілки. За безпекою контроль праці здійснюють державні й відомчі спеціалізовані інспекції.

У Законодавстві про працю міститься вимоги і норми з виробничої санітарії, техніки безпеки та норми, що регулюють робочий час, час відпочинку, звільнення та переведення на іншу роботу, а також норми праці щодо жінок, молоді, гігієнічні норми і правила, тощо.

8.2 Аналіз умов праці

Фірма дотримується всіх правил з охорони праці і слідкує за їх дотриманням.

При виконанні робіт на комп'ютерах працівникам необхідно дотримуватись вимог загальної інструкції з охорони праці.

До роботи на комп'ютерах допускаються особи, які пройшли: медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний

інструктаж з охорони праці на робочому місці. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Основним обладнанням робочого місця є монітор, системний блок, миша та клавіатура.

Робочі місця розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м та між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Монітор розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Елементи робочого місця розміщуються так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Джерела освітлення розташовані з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, також використовують антиблікові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Використовуються скляні поляризаційні фільтри вони забезпечують найкращу якість зображення. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Працівники мають пам’ятки, що раціональною робочою позою вважається положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктєвого суглоба коливається в межах 70 – 90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15 – 20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних принтерах, збільшується вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Для попередження травм усе електричне обладнання заземлене. Приступаючи до роботи працівникам необхідно перевірити справність обладнання. В разі виявлення порушень їм треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.3 Розробка заходів з охорони праці

Працівники повинні дотримуватися статті 18 Закону України ”Про охорону праці” згідно цій статті працівники зобов'язані:

− знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

− співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу;

− дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства.

Також повинні виконуватися вимоги безпеки перед початком роботи, працівникам потрібно:

− увімкнути систему кондиціювання в приміщенні;

− перевірити надійність встановлення апаратури на робочому столі.

Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;

− перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;

− відрегулювати освітленість робочого місця;

− відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

− приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з’єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

− ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

− відрегулювати яскравість монітора, мінімальний розмір світної точки, фокусування, контрастність.

Працівники повинні дотримуватися рекомендацій:

− яскравість монітору – не менше 100Kg/M2;

− відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;

− мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких неполадок роботу не розпочинати, повідомити про це керівника.

Працівникам потрібно дотримуватися вимог безпеки під час виконання роботи:

− необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

− для забезпечення несприятливого впливу на користувача пристроїв типу ”миша” належить забезпечувати вільну велику поверхню столу для переміщення ”миші” і зручного упору ліктьового суглоба;

− не дозволяються сторонні розмови, подразнюючі шуми;

− періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Працівники не повинні порушувати правил з охорони праці та їм забороняється:

− самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, 1 раз на півроку повинні відкривати процесор і вилучати пилососом пил і бруд, що накопичилися;

− класти будь-яку предмети на апаратуру комп'ютера;

− закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з’єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою роздільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При зминанні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Працівника потрібно дотримуватися вимоги безпеки після закінчення роботи:

− закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;

− вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою запобігання попаданню в неї пилу;

− прибрати робоче місце;

− ретельно вимити руки теплою водою з милом;

− вимкнути кондиціонер, освітлення і загальне електроживлення;

− пройти в спеціально обладнаному приміщенні сеанс психофізіологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

8.4 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові

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

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

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

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

Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин

електроустаткування або порушення заземлення треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.5 Розрахункова частина

В приміщенні (де відсутні джерела виділення шкідливих речовин) працює одна людина. Робота пов'язана з використанням ПЕОМ. Розміри приміщення: А = 4 м, В = 3,5 м, Н = 2,8 м, устаткування займає 15% об'єму. Визначити найменшу необхідну кількість повітря для вентиляції.

Для приміщень, в яких відсутні виділення шкідливих речовин у повітрі, розрахунок вентиляції здійснюється залежно від кількості працюючих.

Необхідна кількість повітря ($м^3$ /год.), яка забезпечує відповідність параметрів повітря робочої зони нормованим значенням, визначається за наступною формулою:

$$L = L' \cdot N, \quad (8.1)$$

де L' – нормативна кількість повітря на одного працюючого, яка залежить від питомого об'єму приміщення, $м^3$ /(год.-люд.);

N – кількість працюючих.

Питомий об'єм приміщення $V_п$, ($м^3$ /люд.), визначається за формулою:

$$V_п = V / N, \quad (8.2)$$

де V – об'єм приміщення, $м^3$.

Визначаємо вільний об'єм приміщення

$V = А \cdot В \cdot Н \cdot 0,85 = 4 \cdot 3,5 \cdot 2,8 \cdot 0,85 = 33,3$ $м^3$.

Питомий вільний об'єм складає

$V' = V / N = 33,2 / 1 = 33,2$ $м^3$ /люд. < 20 $м^3$/люд.

Нормована кількість повітря на одну людину при $V' < 20$ $м^3$ /люд. становить 30 $м^3$ /(год. • люд.).

Висновки до розділу

У даному розділі магістерської роботи проведено аналіз умов працівника робота якого пов'язана з комп'ютерною технікою. Проведено аналіз основних санітарно – гігієнічних показників в заданому приміщені , де працівник зайнятий постійною роботою за комп'ютером.. Створені умови повинні забезпечувати комфортну роботу. На підставі вивченої літератури з даної проблеми, були зазначені оптимальні параметри мікроклімату, освітлення, допустимі рівні шуму та іонізуючого випромінювання при роботі з ПЕОМ, а також розраховано найменшу необхідну кількість повітря для вентиляції.

Дотримання умов, що визначають оптимальну організацію робочих місць працівників, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить як в кількісному, так і в якісному відносинах продуктивність їх праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи моніторингу стану жорсткого диску з використанням технології SMART.

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 моніторингу стану жорсткого диску з використанням технології SMART.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу стану жорсткого диску з використанням технології SMART.

– Досліджена система моніторингу стану жорсткого диску з використанням технології SMART.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу стану жорсткого диску з використанням технології SMART.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4 Sydney. 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 SEED.

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 41902 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,44 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Касяненко І.С. Дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2 Adam Freeman. Pro Go The Complete Guide to Programming Reliable and Efficient Software Using Golang. Apress Media. 2022. 1078 p.

3 Fernando Doglio. Skills of a Successful Software Engineer. Manning. 2022. 182 c.

4 M. Holmes He. Creating Apps with React Native. Apress Media. 2022. 445 p.

5 Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p

6 Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.

7 JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.

8 Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.

9 Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.

10 Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

11 Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

12 Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

13 Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

14 Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

15 Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

16 Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

17 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.

18 Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

19 Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

20 Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

21 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.

22 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.

23 Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

24 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.

25 Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

26 Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

27 Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

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

29 Смірнов О.А., 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.

30 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).

31 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.

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

33 O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

34 Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

35 Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

36 Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

37 Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

38 Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

39 Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

40 Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). - Полтава: ПолтНТУ. - 2016. - С. 98-103.

41 Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). - Харків: ХУПС. - 2016. - С.96-100.

42 Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових

праць "Системи обробки інформації". - Випуск 8 (145). - Х.: ХУПС - 2016. - С. 77-80.

43 Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". - Випуск 6 (143). - Х.: ХУПС - 2016. - С. 216-220.

44 Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). - Харків: ХУПС. - 2016. - С. 128-133.

45 Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). - Харків: ХУПС. - 2016. - С. 150-158.

46 Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 40-42.

47 Смірнов О.А., Коваленко А.С., Коваленко О.В., Доренський О.П. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи. Системи озброєння і військова техніка. – Випуск 2(46) – Х.: ХУПС – 2016. – С. 103-107.

48 Smirnov A.A., Kovalenko A.V. Kovalenko A.S. Dorensky A.P. Information model and its element for displaying information on technical condition of objects of integrated information system. International Journal of Computational Engineering Research (IJCER). – Volume 6, Issue 1. – India. Delhi. – 2016. – P. 21-27.

49 Смірнов О.А., Євсеєв С.П., Король О.Г., Коваленко О.В., Коваленко А.С., Смірнов С.А. Архітектура мікропроцесорів та компонентів ЕОМ. Навчальний посібник – Кіровоград: Вид. Лисенко В.Ф., 2015. – 550 с.

50 Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення. Навчальний посібник. За ред. О.А. Смірнова. – Кіровоград: КНТУ 2013. – 409с.

51 Смірнов О.А., Осадчій С.І., Мелешко Є.В., Іванов С.Г., Павленко М.А., Усачов О.М. Основи технічної експлуатації АСУ. Навчальний посібник. – Кіровоград: КНТУ 2013. – 322с.

52 Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник. – Кіровоград: КНТУ 2013. – 257с.

53 Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи моніторингу стану жорсткого диску з використанням технології SMART.

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. № 32-13 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 моніторингу стану жорсткого диску з використанням технології SMART.

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 моніторингу стану жорсткого диску з використанням технології SMART;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4 Sydney.

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 повинна бути розглянута пожежна безпека.

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 – 118 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 21.12.2023 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

_____ Коваленко О.В.

Дослідження та програмна реалізація системи моніторингу стану жорсткого диску з використанням технології SMART

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 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 функцій 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.
```

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.
```