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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Курбанов Б.
«____» _____ 2024 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Курбанову Бахтіяру
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Курбанов Б.
(прізвище та ініціали)

АНОТАЦІЯ

Курбанов Б. Програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу стану жорсткого диску з використанням технології SMART.

Метою розробки є програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART.

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 RAD Studio Delphi 10.

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

ABSTRACT

Kurbanov B. Hard disk status monitoring system software using SMART technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

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

The purpose of the development is the software of the hard disk status monitoring system using SMART technology.

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 RAD Studio Delphi 10 environment.

Keywords: computer engineering, monitoring, 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 програмне забезпечення 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.

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

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

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).

1.2 Область застосування

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). З математичної точки зору один атрибут ідеальний, якщо наступна нерівність ІСТИННА:

$$A - f(r) >= C \quad (1),$$

де: A – теоретичне максимальне значення, «найкраще» можливе значення атрибута; f – функція для обчислення декремента на основі необроблених (r) значень. Зазвичай це лінійна функція, тому в більшості випадків r множиться на константу B; C – пороговий рівень постачальника, нижче цього рівня атрибут вважається проблемним.

Цей метод має деякі недоліки. Значення A, B, C (або функція f) не визначені точно (вони можуть відрізнятися від моделі до моделі навіть на двох жорстких дисках одного виробника). Іншим недоліком є те, що атрибути оцінюються незалежно, зв'язок між ними ігнорується.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Жорсткий диск або SSD є важливою частиною вашого ПК, де встановлюється операційна система та зберігаються всі ваші дані. Без нього ви не змогли б зберегти свою роботу. Оскільки він постійно використовується, жорсткий диск є одним із найбільш завантажених компонентів вашого комп'ютера, тому його потрібно регулярно перевіряти та обслуговувати, щоб запобігти проблемам.

З нашим напруженим графіком легко забути регулярно перевіряти стан жорсткого диска. Крім того, багато людей не знають, як робити ці перевірки. На щастя, в Інтернеті є багато інструментів перевірки стану жорсткого диска. Але який вибрати? Не хвилюйтеся, ми склали список із 8 найкращих програм для перевірки стану жорсткого диска в Windows. Перегляньте особливості кожної програми, щоб знайти найкращий варіант.

Інструмент тестування HDD №1. Безкоштовна версія DiskGenius

DiskGenius Free Edition – це універсальний інструмент, який пропонує широкий спектр функціональних можливостей, окрім просто перевірки працездатності жорстких дисків, SSD та SD-карт. Він також містить такі функції, як відновлення даних, керування розділами диска, клонування диска, міграція Windows, видалення жорсткого диска, що робить його комплексним рішенням для керування пристроями зберігання даних.

Переваги:

– Підтримка перевірки та відновлення пошкоджених секторів.

– Підтримка перевірки даних SMART.

– Комплексний інструмент із багатьма функціями.

– Зручний інтерфейс, у якому легко орієнтуватися.

– Доступна безкоштовна версія з багатьма корисними функціями.

Недоліки:

– Для деяких додаткових функцій потрібна платна версія.

– Версія для Mac недоступна.

Як використовувати засіб:

1. Перевірте стан жорсткого диска, перевіривши інформацію SMART.

Крок 1. Після запуску DiskGenius Free Edition виберіть жорсткий диск або SSD, для якого ви бажаєте перевірити справність, а потім клацніть **Диск** – **Переглянути інформацію SMART**.

Крок 2. Ви можете переглянути детальну інформацію, щоб дізнатися про стан працездатності, як-от стан працездатності, годину ввімкнення, термін служби, що залишився, критичне попередження тощо.

2. Перевірка та відновлення пошкоджених секторів

Крок 1. У лівій частині інтерфейсу DiskGenius клацніть жорсткий диск, який потрібно перевірити, а потім клацніть **Диск** – **Перевірити або відновити пошкоджені сектори**.

Крок 2. Натисніть **«Почати перевірку»**, і програма почне сканувати диск, щоб перевірити наявність пошкоджених секторів.

Стан справності диска представлено кольоровими блоками, наприклад, червоний блок означає пошкоджені сектори.

Інструмент тестування HDD №2. Інструмент перевірки помилок Windows

Інструмент перевірки помилок Windows – це графічний інтерфейс для CHKDSK, що полегшує користувачам перевірку та виправлення помилок жорсткого диска без використання командного рядка. Це вбудована функція Windows, яка допомагає виявляти та виправляти проблеми з жорстким диском.

Переваги:

– Простий у використанні завдяки графічному інтерфейсу.

– Не потрібно встановлювати програмне забезпечення.

– Може виявляти та виправляти помилки файлової системи.

Недоліки:

– Обмежується базовою перевіркою та виправленням помилок.

– Не надає детальних показників здоров’я.

Як використовувати інструмент для перевірки працездатності HDD:

Крок 1. Відкрийте Провідник файлів і клацніть правою кнопкою миші диск, який потрібно перевірити.

Крок 2. У контекстному меню виберіть «Властивості».

Крок 3. Перейдіть на вкладку «Інструменти» та натисніть «Перевірити» в розділі Перевірка помилок. Дотримуйтеся вказівок, щоб сканувати та відновити диск.

Інструмент тестування HDD №3. CHKDSK

CHKDSK – це вбудована утиліта Windows, яка перевіряє файлову систему та метадані файлової системи тому на наявність логічних і фізичних помилок. Він може виправляти помилки файлової системи та відновлювати дані з пошкоджених секторів, що робить його цінним інструментом для підтримки працездатності жорсткого диска.

Переваги:

– Не потрібно встановлювати програмне забезпечення; він вбудований у Windows.

– Може виправляти помилки файлової системи та відновлювати дані з пошкоджених секторів.

– Можна запустити з командного рядка.

Недоліки:

– Інструмент командного рядка може налякати деяких користувачів.

– Не надає детальних показників здоров’я, як інші інструменти.

Як використовувати CHKDSK для перевірки справності жорсткого диска:

Крок 1. Відкрийте командний рядок як адміністратор.

Крок 2. Введіть chkdsk E: /r (замініть E: буквою диска, яку потрібно перевірити).

Крок 3. Натисніть Enter і дотримуйтеся вказівок на екрані.

Інструмент тестування HDD №4. CrystalDiskInfo

CrystalDiskInfo – це безкоштовний і легкий інструмент, який надає повну інформацію про стан вашого жорсткого диска, включаючи детальні атрибути SMART (технологія самоконтролю, аналізу та звітності).

Переваги:

– Легкий і простий в установці.

– Надає детальні дані SMART і моніторинг температури.

– Підтримує різні типи накопичувачів, включаючи SSD і HDD.

Недоліки:

– Деяким користувачам інтерфейс може здатися технічним.

– Обмежується моніторингом здоров'я; не пропонує функції ремонту.

Як використовувати інструмент для перевірки даних SMART для HDD/SSD:

Крок 1. Завантажте та встановіть CrystalDiskInfo з офіційного сайту.

Крок 2. Відкрийте програму, і вона виявить і відобразить стан справності ваших дисків.

Крок 3. Перегляньте дані SMART і показники температури, щоб оцінити стан накопичувача.

Тестовий інструмент HDD №5. HD Tune Pro

HD Tune Pro – добре відома утиліта для моніторингу стану жорсткого диска, порівняльного аналізу продуктивності та пошуку помилок. Він надає детальну інформацію про продуктивність і стан справності жорсткого диска, що робить його популярним вибором як серед звичайних, так і досвідчених

користувачів.

Переваги:

– Простий у використанні завдяки чистому та інтуїтивно зрозумілому інтерфейсу.

– Пропонує функції порівняльного аналізу для перевірки продуктивності драйву.

Недоліки:

– Обмежена безкоштовна версія; більшість функцій доступні лише в платній версії.

Як використовувати засіб:

Крок 1. Завантажте та встановіть HD Tune Pro з офіційного сайту.

Крок 2. Запустіть програму та перейдіть на вкладку «Здоров'я».

Крок 3. У спадному меню виберіть диск, який потрібно перевірити.

Тестовий інструмент HDD №6. HDDScan

HDDScan – це безкоштовна утиліта, яка виконує діагностику жорсткого диска, включаючи моніторинг працездатності, перевірку помилок і перевірку продуктивності. Він підходить як для домашніх користувачів, так і для професіоналів, яким потрібен надійний інструмент для діагностики жорсткого диска.

Переваги:

– Безкоштовний і простий у використанні.

– Пропонує різноманітні діагностичні тести, включаючи поверхневі тести та моніторинг SMART.

– Підтримує різні пристрої зберігання даних, включаючи SSD, жорсткі диски та USB-накопичувачі.

Недоліки:

– Інтерфейс може бути не дуже інтуїтивно зрозумілим для новачків.

Як користуватися програмним забезпеченням:

Крок 1. Скачайте та встановіть HDDScan з офіційного сайту.

Крок 2. Запустіть програму та виберіть зі спадного меню диск, який ви хочете перевірити.

Крок 3. Виберіть потрібний тест (наприклад, тест SMART, тест поверхні) зі списку варіантів.

Крок 4. Розпочніть тест і перегляньте результати після завершення процесу.

Інструмент тестування HDD №7. AIDA64 Екстрім

AIDA64 Extreme – це комплексний інструмент для діагностики та порівняльного аналізу системи, який серед багатьох функцій включає моніторинг стану жорсткого диска. Він призначений для користувачів, яким потрібна детальна інформація про апаратне забезпечення та продуктивність системи.

Переваги:

– Надає поглиблену інформацію про систему та діагностику.

– Підтримує широкий спектр апаратних компонентів.

Недоліки:

– В першу чергу розроблено для досвідчених користувачів.

Як користуватися програмою:

Крок 1. Завантажте та встановіть AIDA64 Extreme з офіційного сайту.

Крок 2. Відкрийте програму та перейдіть до розділу «Сховище» в головному меню.

Крок 3. Виберіть «SMART», щоб переглянути детальну інформацію про стан ваших жорстких дисків.

Тестовий інструмент HDD №8. BIOS

BIOS (Basic Input/Output System) може надати основну інформацію про стан вашого жорсткого диска, особливо якщо він підтримує технологію SMART (технологія самоконтролю, аналізу та звітування). Доступ до BIOS – це простий спосіб виконати початкову перевірку працездатності жорсткого диска без необхідності завантажувати Windows.

Переваги:

– Не потрібно встановлювати програмне забезпечення.

– Доступний практично з будь-якого комп'ютера.

Недоліки:

– Інтерфейс і навігація відрізняються від бренду.

Як перевірити працездатність HDD в BIOS:

Крок 1. Перезавантажте комп’ютер і увійдіть до налаштування BIOS, натиснувши певну клавішу під час запуску (зазвичай F2, Del, Esc або F10).

Крок 2. Перейдіть у меню BIOS, щоб знайти розділ, у якому відображається інформація про жорсткий диск. Цей розділ часто позначається як «Hardware Monitor», «System Health» або «SMART Status».

Крок 3. Перегляньте стан SMART або інші індикатори працездатності вашого жорсткого диска.

Поширені запитання щодо перевірки працездатності жорсткого диска

1. Що таке справність HDD?

Справність жорсткого диска означає загальний стан і функціональність жорсткого диска, включаючи його продуктивність, частоту помилок і фізичний стан. Моніторинг стану жорсткого диска допомагає виявити потенційні проблеми, які можуть призвести до втрати даних або збою диска.

2. Як перевірити жорсткий диск на наявність проблем?

Ви можете перевірити жорсткий диск на наявність проблем за допомогою різних програмних засобів, таких як DiskGenius Free Edition, HD Tune Pro, CrystalDiskInfo, AIDA64 Extreme, HDDScan, CHKDSK або інструмент перевірки помилок Windows. Ці інструменти можуть виконувати сканування для виявлення помилок, моніторингу даних SMART і оцінки справності накопичувача.

3. Як перевірити, чи справний жорсткий диск?

Щоб перевірити, чи справний жорсткий диск, скористайтеся таким програмним забезпеченням, як DiskGenius Free Edition, для моніторингу даних SMART або перевірки пошкоджених секторів. Програмне забезпечення надає

інформацію про стан накопичувача, температуру та частоту помилок. Якщо диск не має значних проблем, він, імовірно, у хорошому стані.

4. Як провести стрес-тест жорсткого диска?

Для стрес-тестування жорсткого диска використовуйте такі інструменти, як HD Tune Pro або DiskGenius, які пропонують функції порівняльного аналізу та стрес-тестування. Ці інструменти виконуватимуть інтенсивні операції читання/запису на диску, щоб перевірити його продуктивність і стабільність під навантаженням. Відстежуйте температуру накопичувача та частоту помилок під час тестування, щоб переконатися, що він справляється з навантаженням без проблем.

Висновок

Підтримка працездатності вашого жорсткого диска має вирішальне значення для загальної продуктивності та довговічності вашого комп'ютера. Регулярна перевірка справності вашого жорсткого диска може допомогти вам завчасно виявити можливі проблеми та запобігти втраті даних. Інструменти, згадані вище, надають різноманітні функції, які допоможуть контролювати та підтримувати працездатність жорсткого диска. Використовуючи ці інструменти, ви можете переконатися, що ваш жорсткий диск залишається в хорошому стані та уникнути несподіваних збоїв.

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 першим (бакалаврським) 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 першим (бакалаврським) 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

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

#1 Неправильні пороги

Більшість проблем із SMART (відсутність передбачення помилок) спричинені 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, тому що 1000 таких пошкоджених секторів займає «всього» 512000 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 години), 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.

#2 Неправильний метод оцінки

Більшість програм використовують описаний вище метод постачальника для обчислення та відображення працездатності 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.

#3 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.

#4 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).

#5 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.

#6 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 жорсткого диска не мають 100% правильних драйверів для Vista. Це також може запобігти виявленню детальної інформації про жорсткий диск і передбаченню збоїв у новій операційній системі.

#7 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.

3.2 Розробка структурної схеми

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 є надзвичайно рідкісним, за винятком випадків падіння диска, високої потужності (зміщення) або 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.

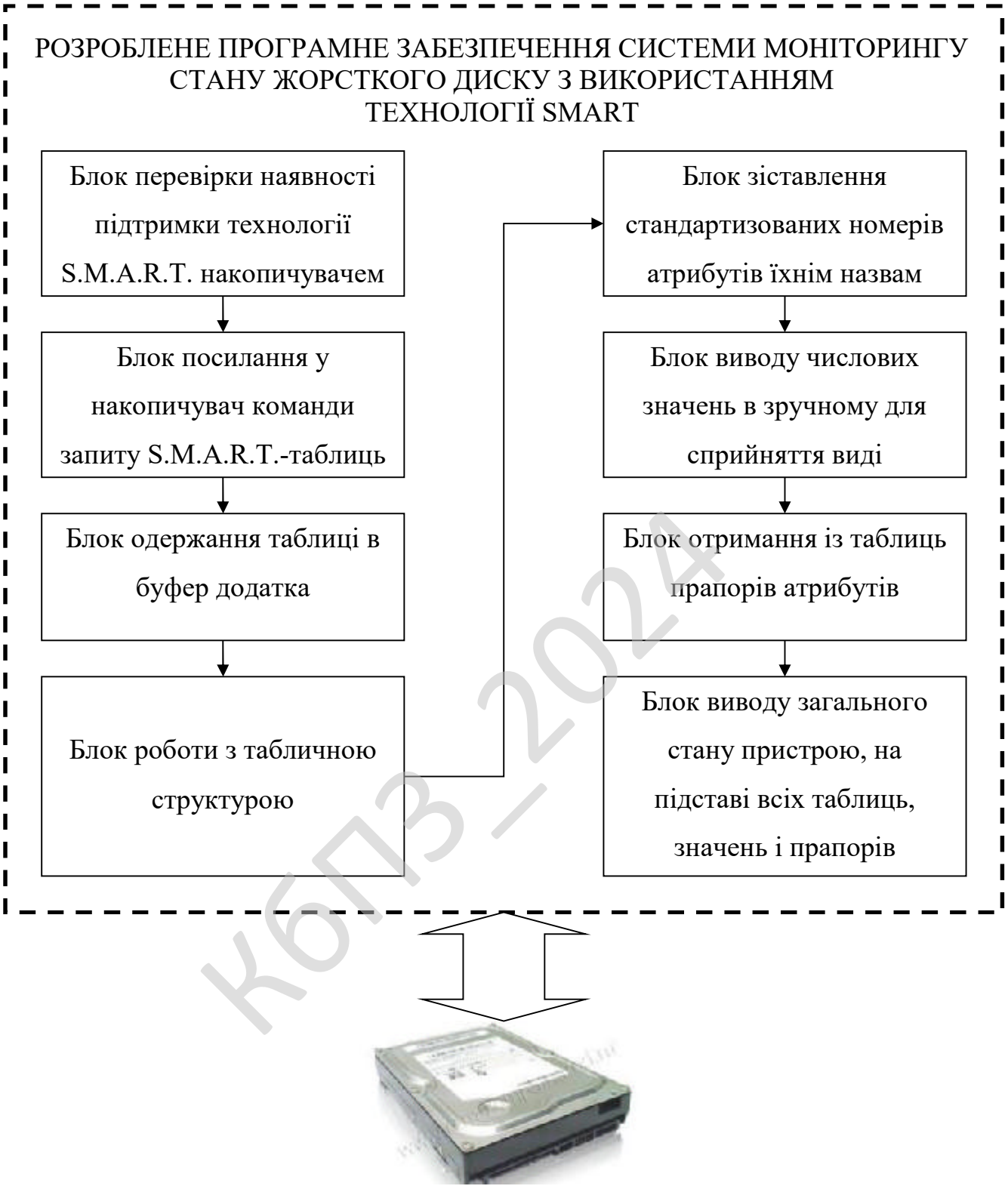

Рисунок 3.1 – Структурна схема системи

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.

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.

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

Короткий опис відомих атрибутів:

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

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

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

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

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

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

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

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

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

– Power-On Hours – Кількість відпрацьованих годин у включеному стані. – Поле raw value цього атрибута показує кількість годин (мінут, секунд – залежно від виробника), відпрацьованих жорстким диском. Зниження значення (value) атрибута до критичного рівня (threshold) указує на виробіток диском ресурсу (MTBF – Mean Time Between Failures). На практиці, навіть падіння цього атрибута до нульового значення не завжди вказує на реальне вичерпування ресурсу й накопичувач може продовжувати нормально функціонувати.

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

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

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

– Soft Read Error Rate – Частота появи "програмних" помилок при читанні даних з диска. – Даний параметр показує частоту появи помилок при операціях

читання з поверхні диска з вини програмного забезпечення, а не апаратної частини накопичувача.

– Emergency Re-track

– ECC On-The-Fly Count

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

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

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

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

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

– UltraDMA CRC Error Count – Загальна кількість помилок CRC у режимі UltraDMA. – Поле raw value містить кількість помилок, що виникли в режимі

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

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

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

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

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

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

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

головки перебували в робочому положенні.

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

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

– Torque Amplification Count – Кількість зусиль обертаючого моменту привода.

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

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

– Head Flying Hours

– Read Error Retry Rate

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.

– 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.

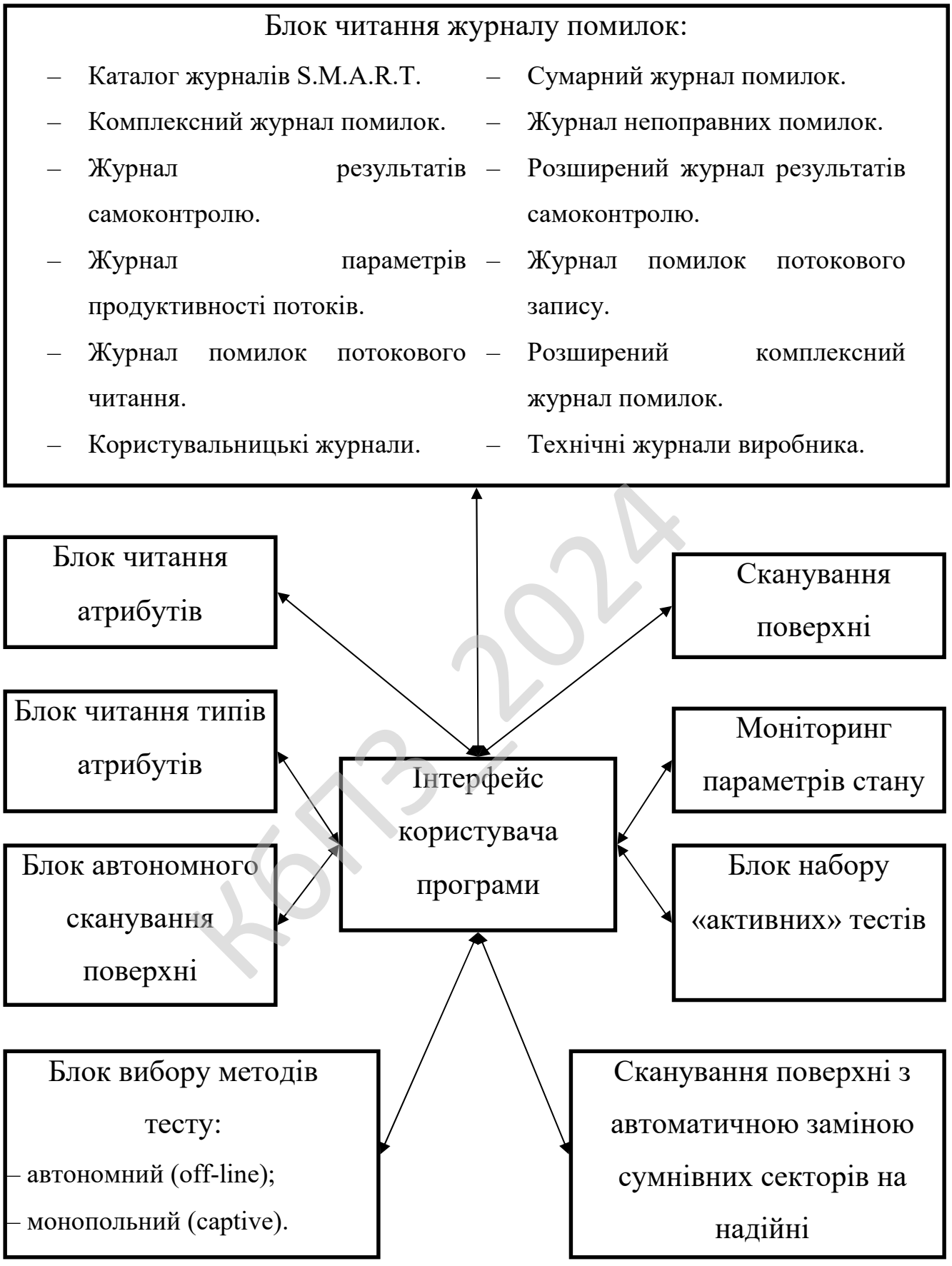

Рисунок 3.2 – Функціональна схема системи

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 дипломного 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.

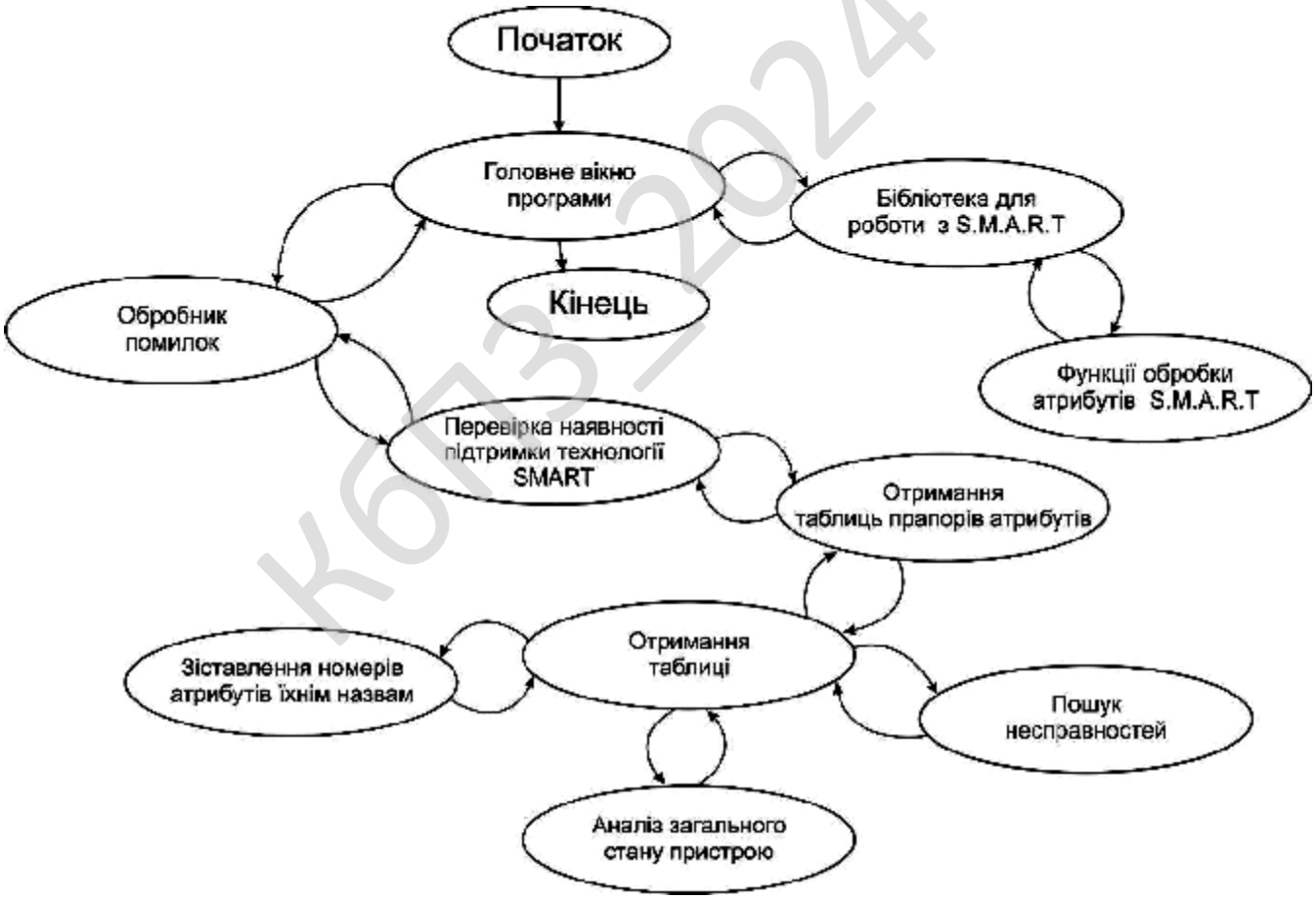

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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.

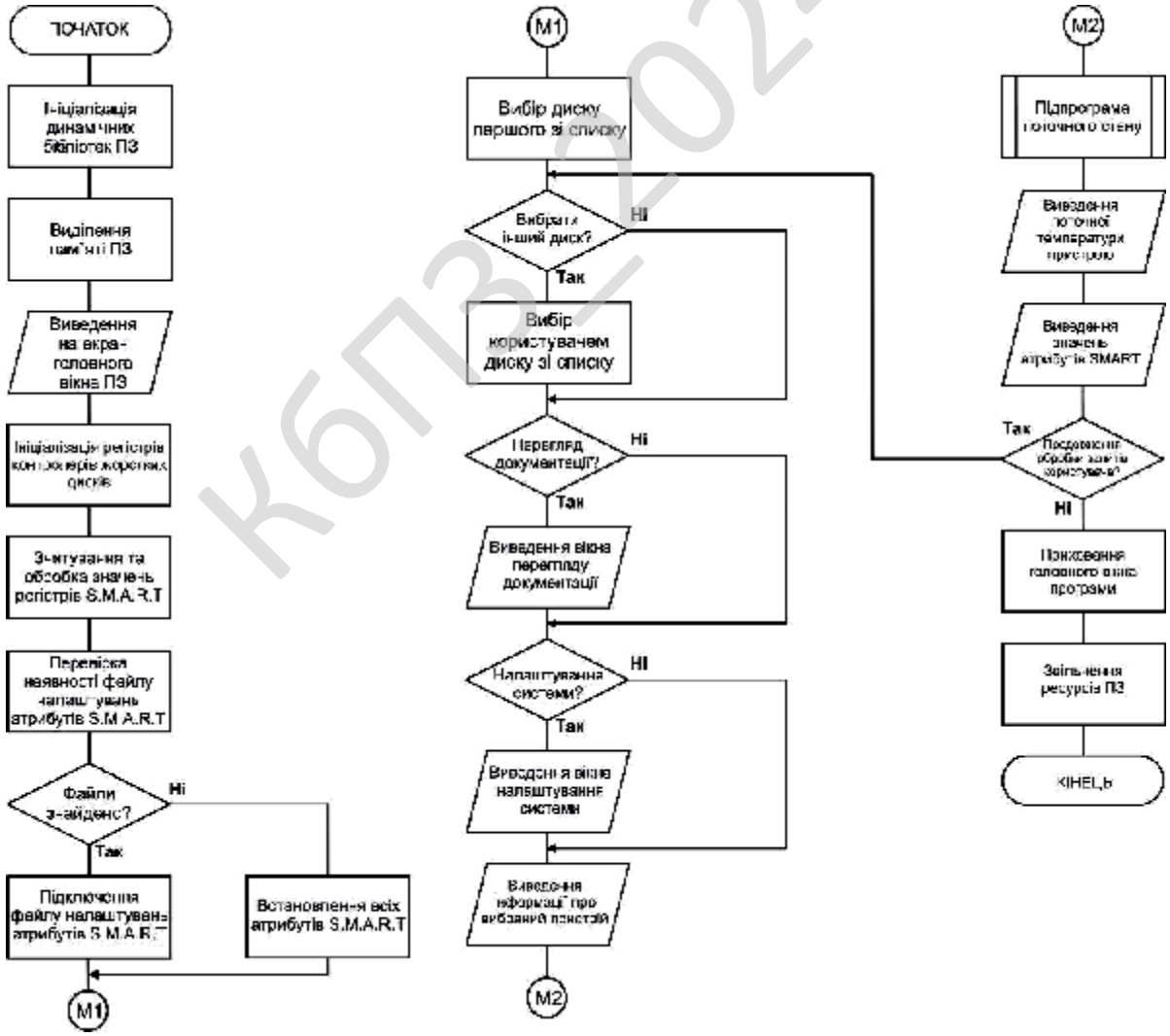

Рисунок 4.1 – Блок-схема основної програми

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.

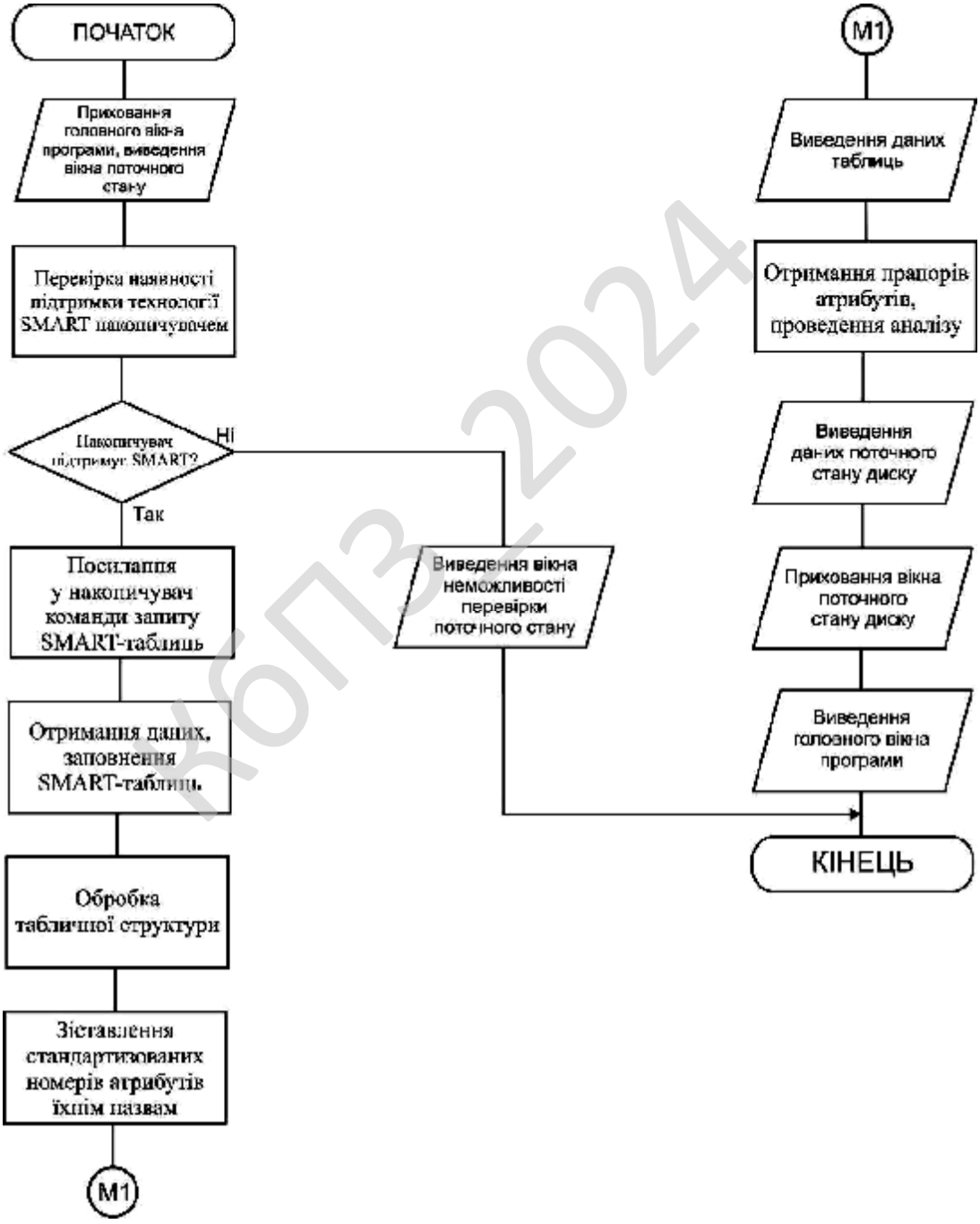

Рисунок 4.2 – Блок-схема роботи підпрограми

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 дипломному 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

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256

біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст $m_1 ... m_{64}$, ключ $K = k_1 ... k_{128}$.

ВИХІД: зашифрований текст $c_1 ... C_{64}$.

1. (розгорнення ключа) становить 16 пару підключів $\{Km_i, Kr_i\}$ отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- $(m_1 ... m_{64})$. (Розділяє текст на ліву й праву 32-бітні половини $L_0 = m_1 ... m_{32}$ і $R_0 = m_{33} ... m_{64}$).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: $L_i = R_{i-1}$; $R_i = L_{i-1} \wedge F(R_{i-1}, Km_i, Kr_i)$, де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. $c_1 ... c_{64}$ <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) - (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] - S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i - R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1,..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене дипломне 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.

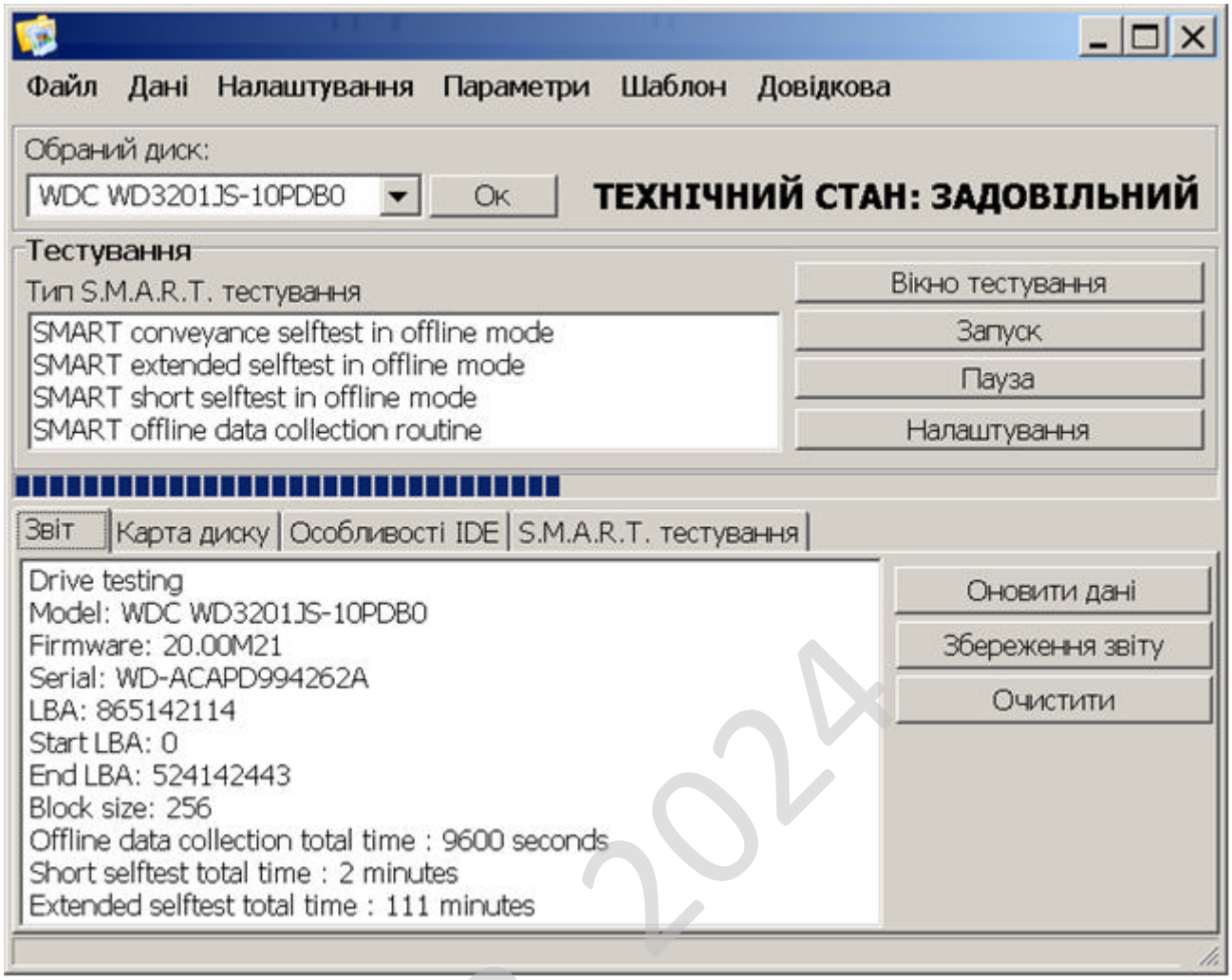

Рисунок 5.1 – Головне вікно ПЗ

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.

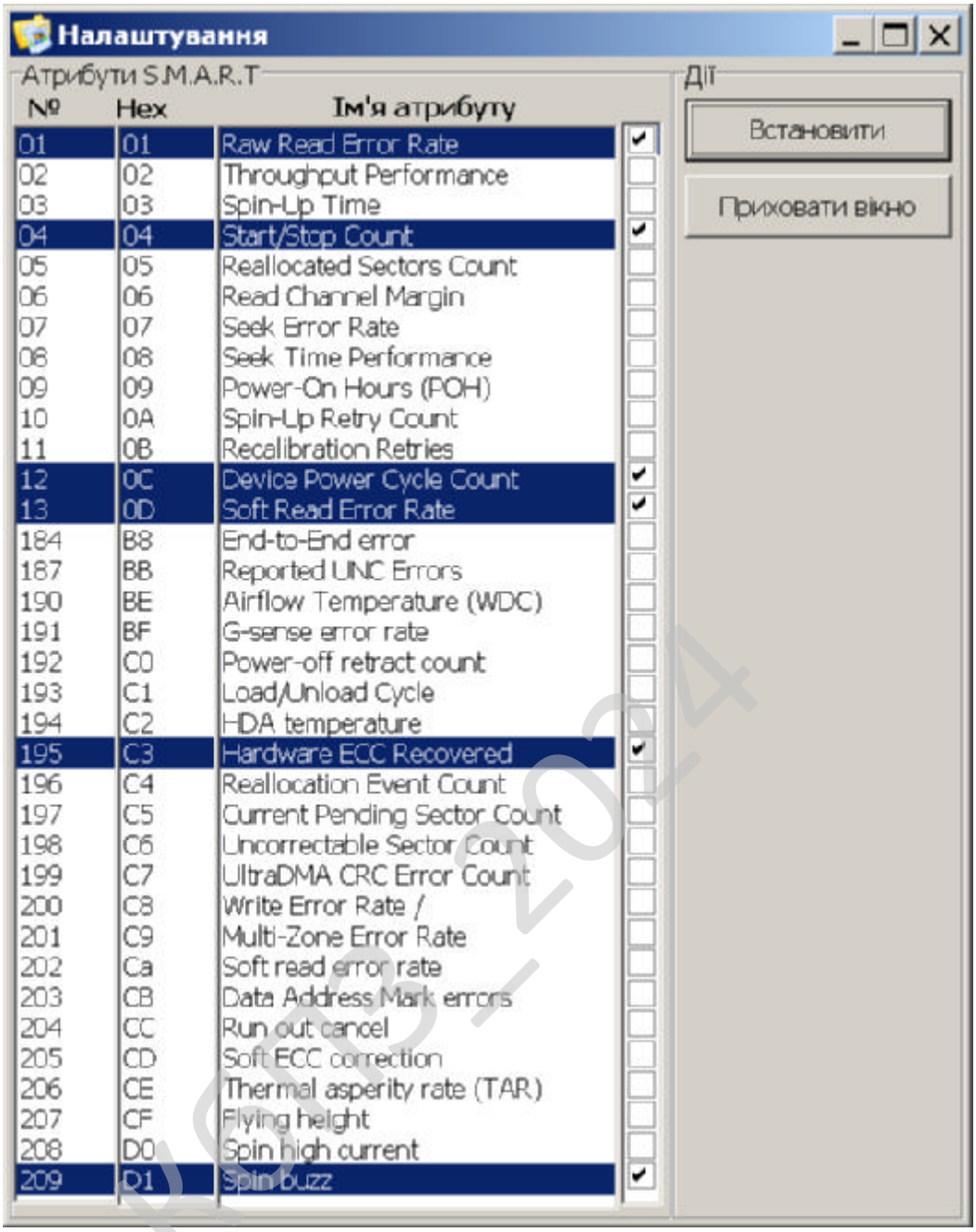

Рисунок 5.2 – Вікно налаштування атрибутів S.M.A.R.T.

На рисунку 5.3 зображена форма авторського права.

Обраний тип ліцензії – умовно безкоштовне розповсюдження.

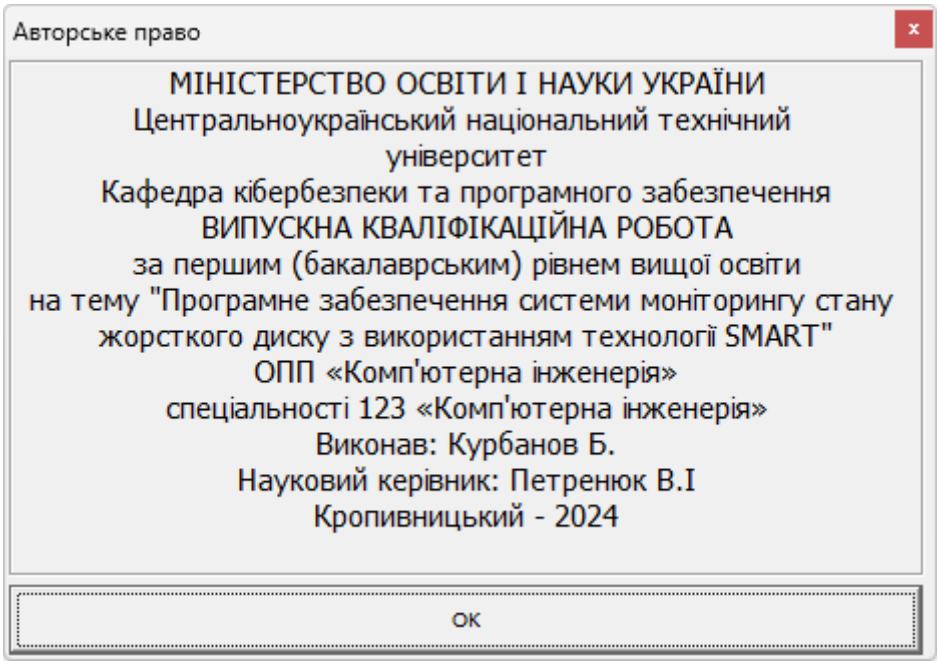

Рисунок 5.3 – Авторське право

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) 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.

Рішення 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 першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу стану жорсткого диску з використанням технології SMART.

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи моніторингу стану жорсткого диску з використанням технології SMART. 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 CAST-128.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

3. 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.

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

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

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

7. 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.

8. 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.

9. 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.

10. 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.

11. 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

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

13. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

16. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

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

18. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

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

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

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

22. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

23. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

24. 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.

25. 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.

26. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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).

35. 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.

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

37. 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.

38. 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.

39. 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

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи моніторингу стану жорсткого диску з використанням технології SMART.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи моніторингу стану жорсткого диску з використанням технології SMART.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– 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 забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 75 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 10.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Петренюк В.І.

Програмне забезпечення 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 – 2024 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.
```