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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи діагностування помилок жорсткого диску”

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

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 123 *"Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

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

Білозору Дмитру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи діагностування помилок жорсткого диску*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи діагностування помилок жорсткого диску*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Білозор Д.О. Дослідження та програмна реалізація системи діагностування помилок жорсткого диску. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи діагностування помилок жорсткого диску.

Метою розробки є дослідження та програмна реалізація системи діагностування помилок жорсткого диску.

Об'єктом дослідження є процес діагностування помилок жорсткого диску.

Предметом дослідження є методи діагностування помилок жорсткого диску.

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

Результат роботи – програмна реалізація системи діагностування помилок жорсткого диску.

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

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

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

Програму розроблено в середовищі Visual C++.

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

ABSTRACT

Bilozor D.O. Research and software implementation of hard disk error diagnosis system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the second (master's) level of higher education, software is developed, which is intended for the hard disk error diagnosis system.

The purpose of the development is the research and software implementation of the hard disk error diagnosis system.

The object of the study is the process of diagnosing hard disk errors.

The subject of the study is methods of diagnosing hard disk errors.

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

The result of the work is the software implementation of the hard disk error diagnosis system.

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

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

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

The program was developed in the Visual C++ environment.

Keywords: computer engineering, diagnostics, hard disk

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. Сучасні жорсткі диски мають велику кількість сучасних технологій збереження даних цілісними на носіях інформації. Однією з них є «S.M.A.R.T.» (Self-Monitoring, Analysis and Reporting Technology), що вбудована в контролер жорсткого диска. S.M.A.R.T. дивиться за станом поверхні вінчестера, виявляє, локалізує і усуває збійні області й помилки.

Цю технологію, на сучасному рівні розвитку, використовують всі виробники вінчестерів. Вона дозволила помістити всю технічну інформацію про збійні області всередину жорсткого диска (вони існують завжди, навіть у вінчестерах від самих надійних виробників). Тому, використовуючи спеціалізоване ПЗ, можна "вилікувати" жорсткий диск (прочитати не тільки ці необхідні дані, відновити інформацію, але й іноді виправити помилки).

Сучасний вінчестер дозволяє усунути збійні блоки шляхом заміни цієї області на резервну. У цьому випадку погана область жорсткого диска міститься в «bad-list» і більше не використовується для зберігання даних, а з резерву виділяється аналогічна за обсягом. Як правило, виробники жорстких дисків допускають заміну 100 ділянок, що, у принципі, повинне вистачити для життя вінчестера, якщо до нього не застосовувати фізичного впливу. Подібну операцію дозволяють робити спеціальні програми перевірки від виробників, однак, потрібно пам'ятати, що не кожну проблему вони можуть вирішити.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи діагностування помилок жорсткого диску.

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

– Огляд існуючих систем діагностування помилок жорсткого диску.

– Дослідження системи діагностування помилок жорсткого диску.

– Програмна реалізація системи діагностування помилок жорсткого диску.

Об'єктом дослідження є процес діагностування помилок жорсткого диску.

Предметом дослідження є методи діагностування помилок жорсткого диску.

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

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

– Удосконалено метод діагностування помилок жорсткого диску.

– Розроблено вітчизняний продукт діагностування помилок жорсткого диску, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі діагностування помилок жорсткого диску.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи діагностування помилок жорсткого диску, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

При всіх достоїнствах сучасних IDE накопичувачів на жорстких дисках, критичним моментом у процесі експлуатації є рівень і ресурс роботи вінчестера. В останні роки було чимало сумних прецедентів, коли тенденція швидкого виходу з ладу відзначалася в цілих лінійок у модельних рядах HDD. У кожному разі, у плані надійності HDD сильно уразливий у принципі, тому що це складні електронно-механічні пристрої. Механіка, як відомо, має властивість поступово зношуватися. Накопичувачі які знаходяться в експлуатації по кілька років вимагають до себе підвищеної уваги тільки через природне зношування. Тому проблема своєчасної діагностики, перевірка стану вінчестера хвилює користувачів, і, насамперед тих, хто довіряє своїм накопичувачам важливі дані.

Розробляєма, у результаті виконання магістерського проектування, система дозволяє діагностування помилок жорсткого диску, за рахунок застосуванн технології S.M.A.R.T.

S.M.A.R.T. (**S**elf **M**onitoring **A**nalysis and **R**eporting **T**echnology) – технологія оцінки стану жорсткого диска й пророкування виходу його з ладу. Вона була розроблена виробниками жорстких дисків для забезпечення більш високого ступеня надійності зберігання інформації. Надійності жорсткого диска завжди надається величезне значення, і справа аж ніяк не в його вартості, а в цінності тої інформації, що при поломці накопичувача губиться назавжди. Суть технології S.M.A.R.T. полягає в тому, що жорсткий диск самостійно відслідковує свій стан, а спеціальна програма, що здійснює моніторинг параметрів S.M.A.R.T. диска, здатна заздалегідь попередити користувача про передаварийній стан пристрою. Після одержання попередження користувачі одержують можливість почати дії, необхідні для забезпечення безпеки своїх даних.

1.2 Область застосування

Областю застосування системи є забезпечення нормальної роботи накопичувачів інформації. Нормальна робота неможлива при наявності збоїв. Умовно збої можна розділити на передбачувані й непередбачені. Непередбачені збої відбуваються швидко, наприклад, стрибок напруги, що ушкоджує електронну схему, псування магнітних головок і поверхні магнітного диска в результаті удару. Підвищення якості, удосконалювання конструкції, технології й виробництва зменшують імовірність непередбачених аварій. Передбачувані збої характеризуються поступовою деградацією того або іншого параметра. Багато механічних збоїв звичайно розцінюються як передбачувані. З розвитком технології S.M.A.R.T. все більше число збоїв стає передбачуваним і попадає в поле її відповідальності, підвищуючи ймовірність пророкування збоїти. При роботі накопичувача S.M.A.R.T. відслідковує всі виникаючі помилки й підозрілі явища, які знаходять висвітлення у відповідних атрибутах.

Кількість атрибутів у різних моделей і виробників відрізняється. Кожний атрибут має значення від 0 до 253. Його величина – надійність конкретного атрибута щодо деякого його еталонного значення, обумовленого виробником. Високе значення говорить про відсутність змін даного параметра. Низьке значення говорить про швидку деградацію або про можливий швидкий збій, тобто чим вище значення атрибута, тим краще. Для кожного атрибута виробником визначається мінімальне можливе значення, при якому гарантується безвідмовна робота накопичувача. При значенні атрибута нижче цієї величини дуже ймовірний збій у роботі або повна відмова. Атрибути бувають критично важливими й некритичними. Вихід критично важливого параметра за межі порога фактично означає вихід з ладу, вихід за межі припустимих значень некритично важливого параметра свідчить про наявність проблеми, при якій жорсткий диск зберігає свою працездатність, хоча й з деяким погіршенням продуктивності.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо дослідження сучасних програмних рішень щодо діагностування стану жорсткого диску. Компанії, що займаються виробництвом жорстких дисків, пропонують діагностичні утиліти власної розробки, які призначені для роботи з їхніми накопичувачами. Не дивно, що саме тестові утиліти від самих виробників HDD вселяють користувачам найбільшу довіру. Звичайно ці програми адаптовані для діагностики накопичувачів виробництва тільки певної компанії, але є й універсальні засоби, не всі з них однаково функціональні. Деякі утиліти із цього ПЗ можуть не тільки протестувати стан вінчестера, але й виправити незначні помилки й дефекти, виконувати деякі інші маніпуляції з дисками, такі як форматування, очищення, збереження й відновлення завантажувальних секторів, спроба реанімації ушкоджених секторів.

Здебільшого тестові утиліти від виробників HDD виконують свої дії під керуванням DOS-подібних операційних систем, що запускаються із завантажувальних дискет. Це цілком виправдане рішення для роботи з жорстким диском, адже завантаження з дискети робить роботу діагностичної програми можливої навіть у випадку краху ОС або FAT на вінчестері, а те й поломки самого накопичувача. При цьому передбачається можливість створення завантажувальної діагностичної дискети з установчого файлу із середовища Windows, з її потім і буде вироблятися вся подальша діагностика жорсткого диска.

Крім, властиво, перевірки вінчестерів на придатність більшість виробників HDD включають у свої утиліти засобу для низькорівневого форматування

жорстких дисків, обнуління вмісту секторів або стирання даних (Erase). Відмінність низькорівневого форматування від звичайного із середовища операційної системи (як правило, програма format) полягає в тому, що при низькорівневому форматі на диску знищується вся інформація, у тому числі службова, а не тільки розмітка. Теж саме може називатися обнулінням секторів або стиранням всіх даних. Таким чином, можна гарантованно знищити будь-які дані, однак головне достоїнство низькорівневого форматування або обнуління полягає в іншому. За допомогою цих засобів іноді можна позбутися від секторів, які операційна система бачить як ушкоджені. Мова йде про так звані софт-помилки. На відміну від фізичних ушкоджених секторів, причиною софт-помилки не є ушкодження поверхні диска. Софт-помилки виникають якщо в службову область сектора з якихось причин заноситься некоректна інформація. Досить обнулити весь сектор з його службовим записом, щоб дефект такого роду був усунутий. Однак при цьому вся інформація користувача в цих секторах буде теж загублена.

Так само варто відзначити, що практично всі виробники HDD перед запуском своїх утиліт рекомендують створити резервну копію важливої інформації й зберегти її в надійному місці, на іншому носії. Мало що може трапитися під час перевірки, тим більше, що накопичувач підозрюється в несправності. Та й не можна виключити можливості некоректних дій з боку користувача, скажемо, що помилково задали команду здатну привести до знищення даних. Раз уже творці діагностичних утиліт попереджають, що інформація потенційно може постраждати, то й ми, зі своєї сторони, повинні рекомендувати дотримуватися певних правил обережності.

Western Digital

Western Digital для своїх жорстких дисків пропонує набори утиліт для роботи як під Windows, так і запускаються із завантажувальної дискети з DOS-подібної операційної системи. На сайті WD доступний цілий ряд програмних засобів для підтримки власних вінчестерів. Не дуже давно там з'явилася утиліта

Data LifeGuard Diagnostics for Windows, що інсталюється й працює під Windows 10/11.

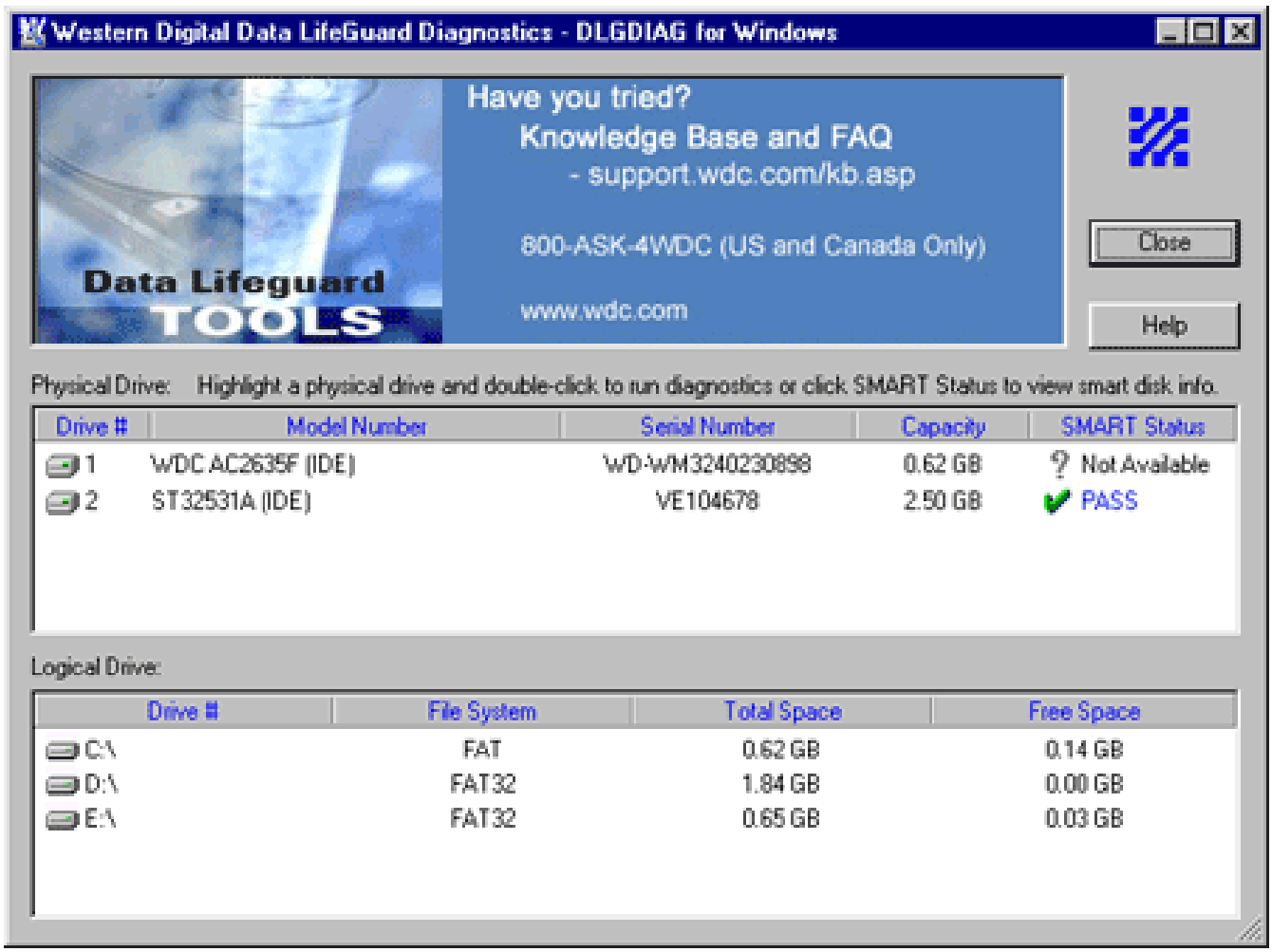

Рисунок 2.1 – Інтерфейс користувача Data LifeGuard Diagnostics for Windows

Утиліта може проводити прискорену (Quick Test) і розширену (Extended Test) діагностику дисків, досить двічі кликнути потрібний привод зі списку у вікні програми, а потім вибрати потрібний режим.

Режим «Quick Test» проходить досить швидко, краще скористатися розширеним режимом, тут ґрунтовно перевіряється поверхня.

Примітно, що підтримуються не тільки вінчестери WD, але й диски інших виробників. Так само за допомогою цієї програми можна видалити всю

інформацію з диска, запуском у режимі занулення – «Write Zeros», що в ряді випадків може позбавити накопичувач від софт-лих. Правда, реалізувати тотальне очищення можна тільки у випадку установки другого, не системного вінчестера. Номінально підтримується відображення атрибутів S.M.A.R.T., як для своїх, так і для чужих вінчестерів, якщо така можливість підтримується з боку накопичувача.

Для роботи із завантажувальної дискети WD пропонує комплект утиліт Data Lifeguard Tools. Цей комплект має набагато більші можливості, чим утиліта, що працює в середовищі Windows, за винятком хіба що відсутності можливості відображення атрибутів S.M.A.R.T. Завантажувальна дискета створюється з установчого файлу в середовищі Windows. Зараз на сайтах WD доступні Data Lifeguard Tools у версії v.10, а також більше стара v.2.8. Можливості обох наборів аналогічні, різниця лише в інтерфейсі так у тім, що v.10 може створювати завантажувальний диск із середовища Windows 10/11, а v.2.8 тільки з 9X/Me. Інше розходження вдалося виявити вже на практиці. Так при установці в системі привода WD Caviar 3.2Gb у комплекті v.10 чому сь не запускався розділ діагностики. А коли в систему був доданий другий, ще більш старий диск WD, то сама дискета v.10 початку підвисати. У випадку ж з v.2.8 ніяких проблем не виникло. Так що доступність більше старої версії ніяк не можна вважати зайвою.

При завантаженні Data Lifeguard Tools v.10 пропонується вибрати один із двох розділів: «Install Drive» або «Data Lifeguard Utilities», причому за замовчуванням спочатку пропонується «Install Drive».

У розділі «Install Drive» діагностичних засобів немає, тут зібрані інструменти, які допоможуть установити на комп'ютер новий вінчестер або переустановити накопичувач заново. В «Install Drive» можна зробити очищення поверхні диска, розбити його на логічні розділи, відформатувати, скопіювати вміст на інший вінчестер, створити резервну копію завантажувального сектора. Тут програма відразу ж запускається в покроковому режимі майстра, причому можна працювати з мишею. У вікні майстра можна вибрати будь-яку дію, щоб

перейти в наступне вікно відповідного інструмента, а при виборі «Advanced Options» надається перелік ще із трьох інструментів.

«EZ-BIOS Setup» – це компонент інсталюємий на жорсткий диск, він вступає в роботу перед завантаженням операційної системи, має сенс використовувати, якщо BIOS материнської плати некоректно розпізнає накопичувач або ж сама ОС не може працювати з дисками великого обсягу. «EZ-BIOS Setup» може заважати роботі деяких дискових утиліт від інших розроблювачів, тому без особливої потреби цю частину не варто використовувати. За допомогою «Backup/Restore Track Zero» можна створити резервну копію завантажувального сектора (Track 0) диска, і зберегти неї на окремому носії, а потім у випадку яких або проблем дуже швидко відновити boot-сектор.

Щоб вийти з режиму майстра, досить закрити його вікно, тоді ставати доступно меню верхнього рядка, звідки можна вибрати будь-який інструмент за своїм розсудом.

Таким чином, можна переглянути інформацію про встановлений HDD, рухаючись по шляху «View/Drive Information».

При виході з розділу «Install Drive» (File/Exit) програма автоматично переходить у розділ «Data Lifeguard Utilities».

У випадку запуску Data Lifeguard Tools v.2.8 ми відразу ж попадаємо в розділ утиліт, де інструменти й інтерфейс аналогічні v.10, там же під окремим пунктом присутні й засоби аналогічні розглянутому вище розділу «Install Drive» з v.10, і навіть деякі додаткові можливості.

В обох версіях присутня «Ultra ATA Management» для відображення й можливої корекції DMA режимів роботи накопичувача в системі, що може знадобитися для деяких морально застарілих материнських плат. «BIOS Check» перевірить сумісність вінчестера з можливостями BIOS'a. Подальший розгляд Data Lifeguard Tools доцільно провести на прикладі v.2.8.

Всі що ми бачили вище в розділі «Install Drive» v.10, тут (v.2.8) перебуває в розділі «EZ-Install», тому нема рації зупинятися окремо на тих же можливостях.

Правда тепер програма працює в типовому інтерфейсі DOS'a. Можливо, за рахунок відсутності громіздкого графічного інтерфейсу майстри в v.2.8 на дискеті вдалося розмістити ще ряд додаткових засобів.

Сюди ставиться докладна довідкова інформація (EZ-Install/Advanced Options/View Drive Jumper), де дані установки джамперів не тільки для вінчестерів WD, а практично для всіх виробників, навіть тих, які дуже рідко зустрічалися на ринку. Така інформація може придатися власникові будь-якого вінчестера.

Діагностичні засоби Data Lifeguard Tools зібрані в розділі «Diagnostic», у всіх версіях програми інтерфейс цього розділу однотипний.

Після вибору одного із установлених у системі HDD можна зробити над ним ряд тестів або заповнити зміст всіх секторів нулями, очистивши тим самим диск, з можливим відновленням так званих софт-бедів, якщо такі виявляться. У режимі «Quick Test» перевіряється тільки коректність підключення HDD до комп'ютера, проходить він дуже швидко. У режимі «Extended Test» запускається перевірка диска шляхом процедури посекторного читання всієї поверхні. Наприкінці розширеного тесту видається результат, якщо код становить 0000, те це значить, що помилок не виявлено.

Недавно Western Digital випустила Data Lifeguard Tools v.11. Нова версія дискової утиліти заслуговує окремого розгляду, тому що сильно відрізняється від всіх попередніх версій Data Lifeguard Tools. Справа в тому, що загальним між версіями 11 і 10 Data Lifeguard Tools залишилася, напевно, лише назва. Data Lifeguard Tools v.11 зовсім відрізняється від v.10 як по дизайну й організації інтерфейсу, так і по змісту. Причому, вийшла Data Lifeguard Tools v.11 окремо для операційних систем Windows і для завантаження з дискети працюючої під DOS.

Відразу ж варто відзначити, що, незважаючи на багато позитивних

моментів, загалом, функціональність Data Lifeguard Tools v.11 стала менше ніж версій 2,8 і 10. Мало того, функціональність Data Lifeguard Tools v.11 for Windows ще менша в порівнянні з її DOS-аналогом, при тому, що обсяг Windows версії більш ніж у два рази перевершує розміри DOS утиліти. Тому в огляді досить розглянути Data Lifeguard Tools v.11 for DOS, а точніше, що працює із завантажувальної дискети з DOS-подібною операційною системою. Тому що інтерфейси утиліт v.11 для Windows і DOS дуже схожі.

Отже, Data Lifeguard Tools v.11 for DOS, як звичайно, з установчого файлу з Windows створює завантажувальну дискету, з якої може завантажуватися програма під керуванням DOS-подібної операційної системи. Інтерфейс програми дуже зручний і до того ж став насичений стильною графікою, чому, напевно, можуть позаздрити навіть більшість Windows-програм.

У роботі повністю задіяна миша. Відразу ж треба відзначити, що з Data Lifeguard Tools v.11, у порівнянні з попередніми версіями, зник розділ діагностики, так само тут ні «Ultra ATA Management» і «BIOS Check», але додалася пара нових функцій.

У розділі «View Installation Tutorial» тепер дається щось типу посібника з підключення й установки нового вінчестера на комп'ютер. Розділ «Set Up Your Hard Drive» призначений для підготовки вінчестера до роботи. Тут у режимі майстра можна очистити жорсткий диск від старої інформації, створити на ньому розділи й відформатувати логічні диски. «Maintenance Options» цікавий, насамперед, тим, що тут можна створити/відновити резервну копію завантажувального запису MBR жорсткого диска.

У розділі утиліт залишилася можливість копіювати інформацію з розділу одного HDD на інший. Новим стала функція «Sit Hard Drive Size», що дозволяє користувачеві вручну встановити доступний обсяг накопичувача, менше номінального рівня.

Сучасні HDD дозволяють програмним шляхом обмежувати доступний на диску обсяг, що буде ідентифікуватися BIOS материнської плати й ОС, звичайно

обрізається кінцева ділянка диска. Природно, обсяг накопичувача можна потім повернути назад. Штучно обмежити розмір нижче номінального може знадобитися у випадку, якщо BIOS або операційна система не приймають більших дисків.

У розділі «Hard Drive Information» можна переглянути як загальну технічну інформацію про HDD і його розділи, так і комбінації установок джамперів для вінчестерів більшості виробників.

Причому перегляд установки джамперів в Data Lifeguard Tools v.11 for DOS реалізований надзвичайно наочно. Потрібно зі списку вибрати виробника й модель, відзначити мишкою бажаний режим роботи накопичувача, і відразу на рисунку автоматично відобразиться розташування й комбінація джамперів для заданого режиму.

Говорячи про Data Lifeguard Tools v.11 for Windows досить тільки відзначити, чого в ній немає в порівнянні з версією працюючої із завантажувальної DOS-дискети. Так, щоб скористатися інструментами «Set Up Your Hard Drive» в Windows у системі знадобитися два вінчестери, тому що не можна розбивати й форматувати системний диск, у той час як при роботі із завантажувальної дискети такої проблеми не виникає, адже ОС повністю перебуває на флоппі-диску.

Немає в Windows-версії й можливостей для штучної установки розміру накопичувача, створення копії й відновлення завантажувального запису MBR. Установки джамперів тепер відображаються тільки для приводів WD.

IBM-Hitachi

IBM пропонує для діагностики своїх жорстких дисків і вінчестерів під маркою Hitachi досить функціональну утиліту Drive Fitness Test. З установчого файлу в середовищі Windows або Linux створюється завантажувальна дискета, що може застосовуватися для роботи з IDE і SCSI накопичувачами. Визнає Drive Fitness Test не тільки диски IBM або Hitachi, але не відмовляється від діагностики вінчестерів від будь-якого іншого виробника. Однак специфічні інструменти,

якось засобу для відновлення збійних секторів, доступні тільки для фірмових HDD.

При завантаженні Drive Fitness Test з дискети, першою справою програма попросить вибрати із двох пунктів: запуск програми з підтримкою SCSI і ATA або ж тільки ATA (IDE) накопичувачів. Якщо SCSI вінчестерів у вас ні, то краще вибрати другий пункт, так програма запуститися набагато швидше. Потім відбувається детектування підключених у системі HDD. Користувачеві пропонується підтвердити список виявлених накопичувачів, після чого ми попадаємо в головне вікно програми.

Головне вікно Drive Fitness Test являє собою класичний DOS'овский інтерфейс із інструментальним меню у верхньому рядку й меню вибору HDD посередині екрана. При цьому підтримується робота з мишею. У меню накопичувачів доступні кнопки запуску швидкого (Quick Test) або розширеного (Advanced Test) тестів. За замовчуванням пропонується провести спочатку швидкий тест для виділеного в списку вінчестера.

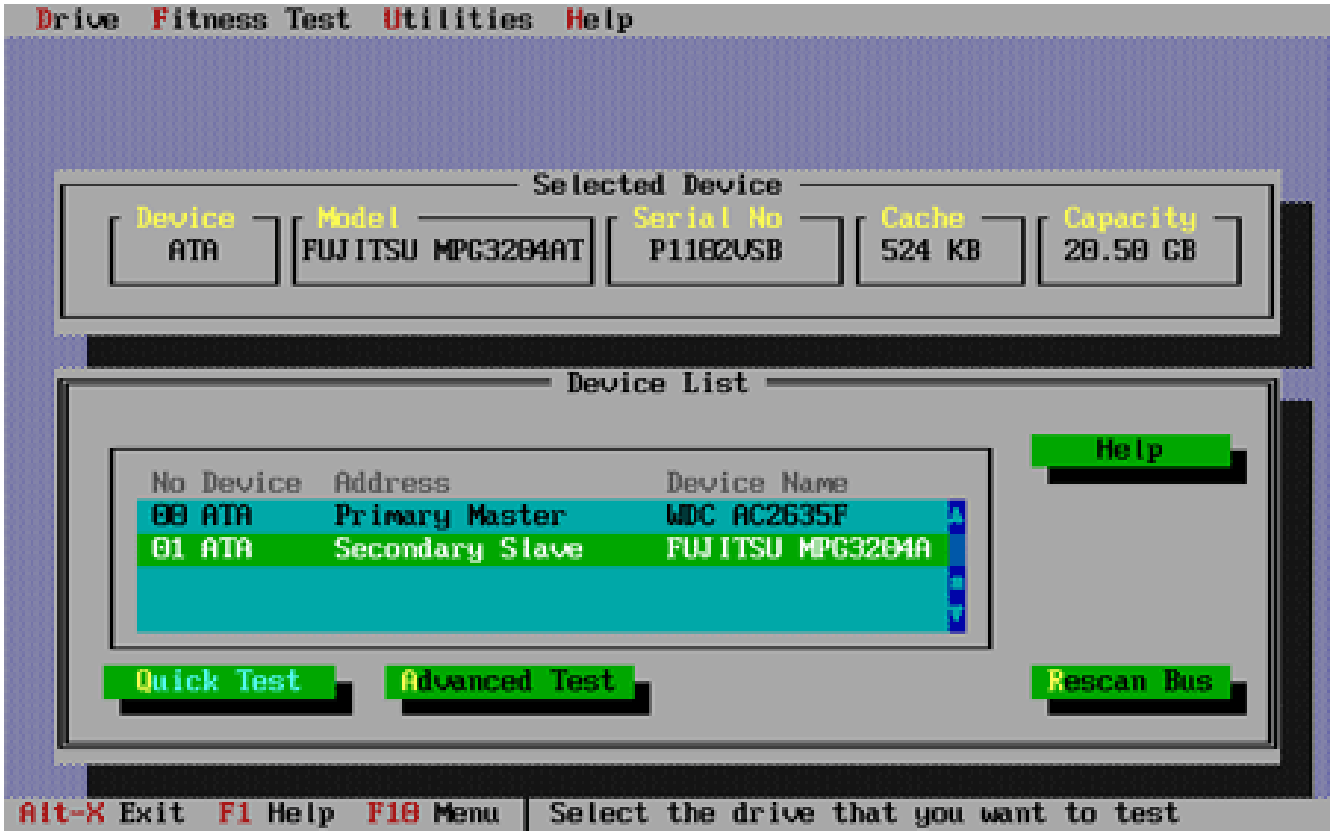

Рисунок 2.2 – Інтерфейс користувача Drive Fitness Test

Під час проходження Quick Test перевіряється загальна функціональність накопичувача, стан S.M.A.R.T., перевіряється система позиціонування й читання головок, сканується невелика ділянка поверхні спочатку диска. Advanced Test повторює всі початкові етапи швидкого тесту, але при цьому посекторно сканується вся поверхня жорсткого диска. Розширений тест займає набагато більше часу, але тільки він може бути гарантією нормального стану накопичувача, що перевіряється. Наприкінці виконання тестів програма видає результат з 00», то це значить, що помилок не(кодом помилки: якщо у звіті коштує код «0 виявлений. У противному випадку Drive Fitness Test може видавати велика кількість кодів помилок, їхнього значення можна довідатися в розділі Help/Contents/DFT error codes.

Набагато цікавіше виглядають інструменти доступні через меню з верхнього рядка. Тут у меню «Fitness Test» доданий ще один тест «Exerciser», емулюючий інтенсивний режим роботи вінчестера, при цьому є можливість задавати велику кількість циклів повторення тесту.

З розділу «Utilities/Drive Info» можна одержати більш докладну інформацію про накопичувачі. У складі програми є засоби для стирання завантажувального сектора (Erase Boot Sector), обнуління поверхні всього накопичувача (Erase Disk), засоби для відновлення ушкоджених секторів (Corrupted Sector Repair), однак всі вони працюють тільки з вінчестерами IBM або Hitachi, на відміну від діагностичних тестів, спроба запуску на HDD інших виробників виявиться безуспішною.

Досить корисними, що є присутнім тільки в утиліті від IBM-Hitachi, можуть виявитися можливості «Corrupted Sector Repair», скориставшись цим інструментом, можна спробувати відновити ушкоджені сектори, не прибігаючи до очищення всього диска. При цьому вся інформація, що була у відновлюваних секторах, звичайно ж, буде загублена. У випадку успішного відновлення ці сектори будуть заповнені нулями, або ж замінені поверхнею з резервної області, і, знову ж, виявляться порожніми.

Під пунктом «ATA Functions» ховається доступ до стану системи самодіагностики S.M.A.R.T. накопичувача. Атрибути S.M.A.R.T. не виводяться, відображається тільки статус. Так само тут можна включити або відключити систему внутрішньої самодіагностики, але тільки на жорстких дисках, які підтримують включення/відключення S.M.A.R.T., та й то, навряд чи виробники вінчестерів реалізує її повне відключення усередині накопичувача, краще S.M.A.R.T. залишати включеним.

Seagate

На сайті Seagate доступний набір з декількох діагностичних утиліт. Сама остання розробка SeaTools Enterprise працює з SCSI дисками Seagate, є версії для Windows 10/11 і Linux. Програма вимагає інсталяції й потім, працюючи в системі, може виводити інформацію про SCSI накопичувач і контролер, проводити ряд тестів на цілісність структури даних і стан фізичної частини, а так само проводити форматування дисків. IDE вінчестери SeaTools Enterprise не бачить взагалі. Інша утиліта від Seagate – SeaTools Desktop працює із завантажувальної дискети й призначена для діагностики як IDE так і SCSI вінчестерів. Завантажувальна дискета SeaTools Desktop створюється в середовищі Windows. Потім комп'ютер завантажується із цієї дискети. Треба відзначити, що виглядає SeaTools Desktop досить оригінально. Хоч це й DOS-програма, але її користувальницький інтерфейс буяє графікою. Кожний режим роботи програми супроводжується ілюстрацій власної картинки й окремим вікном з коротким описом що відбувається. У заголовному вікні SeaTools Desktop в області ліворуч зібраний список всіх доступних тестів і засобів для роботи з дисками, праворуч виводиться коротке пояснення для виділеного пункту. SeaTools Desktop може проводити діагностику не тільки HDD Seagate, але й вінчестерів інших виробників.

У режимі «Generic Diagnostic» SeaTools Desktop може проводити швидкий і розширений тести для будь-якого накопичувача, а не тільки для вінчестерів Seagate. При запуску «Generic Diagnostic» спочатку з'явитися підменю, де буде

запропонований вибір швидкої (Quick) або повної (Full) діагностики. Після вибору типу діагностики з'явитися наступне підменю в якому потрібно вибрати один з підключених HDD, до речі, тут є пункт вибору для діагностики відразу всіх установлених у системі приводів жорстких дисків. Під час швидкого режиму діагностики перевіряється коректність підключення, стан S.M.A.R.T., тестуються технічні параметри, проводиться частковий тест читання поверхні диска.

Рисунок 2.3 – Інтерфейс користувача SeaTools Desktop

Якщо все нормально, вінчестеру дається «зеленої світло» – наприкінці з'являється короткий звіт із зображенням світлофора із зеленим світлом. У режимі «Full Diagnostic» проводиться той же короткий функціональний тест, перевірка S.M.A.R.T., а так само повне посекторне сканування поверхні

жорсткого диска, тут же перевіряється логічна структура розділів на диску накопичувача.

Зате тести «SeaTools Shot Diagnostic» і «SeaTool Extend Diagnostic» призначені виняткова для дисків Seagate, видимо, з урахуванням специфіки цих HDD. Тут уже в підменю вибору накопичувачів будуть запропоновані тільки встановлені в системі моделі вінчестерів Seagate. Під час «SeaTools Shot Diagnostic» проводиться ряд стандартних тестів, а так само прискорена перевірка на предмет читання випадкових ділянок поверхні, все це займає кілька мінут за часом. При запуску «SeaTool Extend Diagnostic», крім того, проводиться повне сканування поверхні диска. Наприкінці тесту видається короткий звіт про стан HDD.

Програма створює й зберігає на дискеті файли звіту, окремо для «Generic Diagnostic» і «SeaTools Diagnostic» звідки можна переглянути інформацію про результати тестування. Звіт «View Generic Diagnostic Report File» відображає основну інформацію про параметри HDD минулу діагностику, дані про логічні розділи, результати завершення тестів, інформації про технічні показники отриманих досвідченим шляхом, якось швидкості передачі даних, тут немає. У звіті «View SeaTools Diagnostic File», крім стандартної інформації про HDD Seagate, даються показники отримані досвідченим шляхом під час тестів: швидкості передачі даних у різних режимах роботи. Інструментів для примусового форматування дисків або якихось інших маніпуляцій з даними SeaTools Desktop у свій состав не включає. Для виконання маніпуляцій над дисками, таких як: очищення, розбивка на логічні розділи й робота з ними, форматування, Seagate рекомендує використовувати окрему утиліту Seagate Disk Manager.

Maxtor-Quantum

Maxtor для перевірки власних дисків і вінчестерів поглиненої їм Quantum пропонує утиліту Powermax. Знову ж, створюється завантажувальна дискета за допомогою запуску установчого файлу в Windows. Програма поводиться досить

демократично, крім накопичувачів від своїх власних вендорів, не відмовляється перевіряти диски й від усякого іншого виробника, чесно визначаючи його модель і параметри. При запуску утиліти, після того як користувач побачить всі попередження й підтвердить свою згоду використовувати програму, з'явитися спочатку список виявлених пристроїв, а потім і меню вибору накопичувачів.

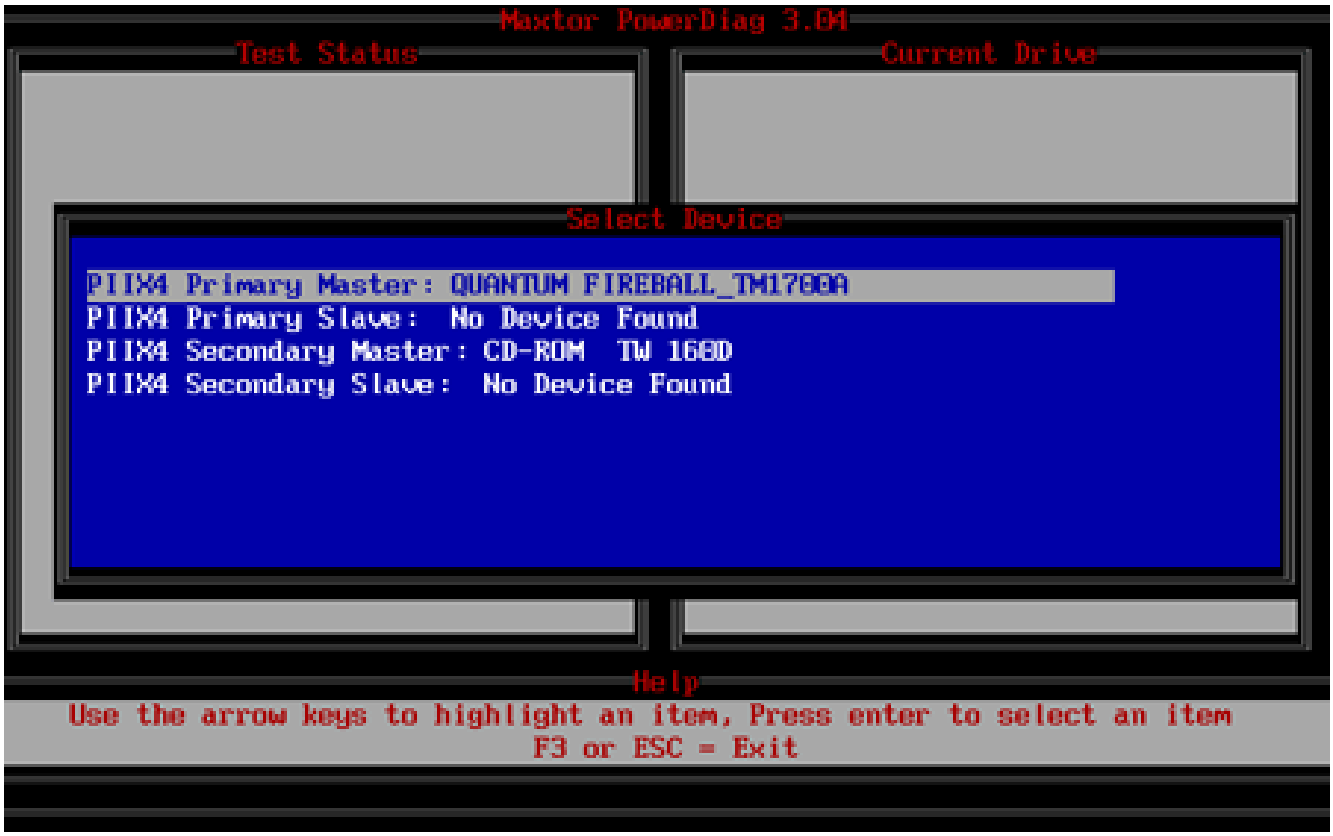

Рисунок 2.4 – Інтерфейс користувача Powermax

Вибравши потрібний HDD, ми попадаємо в головне вікно з меню вибору дій. Тут доступно сім позицій, серед яких чотири тести .

Настановний тест: перевіряє надійність підключення кабелю, коректність установки джамперів, можливість підтримки з боку BIOS більших дисків і аналізує атрибути S.M.A.R.T., щоправда, без виводу на екран самих чисел атрибутів.

За допомогою швидкого (Basic Quick Test) або розширеного (Advanced Test) тестів можна перевірити функціональність накопичувача, звичайно, краще віддати перевагу розширеному. Під час розширеного тесту сканується вся поверхня накопичувача, під час цього процесу в лівій області вікна програми відображається тільки перерахування секторів. На крайній випадок є ще й «Burn In Test», самий інтенсивний з тестів, повністю перевіряється вся поверхня дисків на читання, але тут можна задати до 60-ти циклів повторення цього тесту. Тільки вдало завершивши «Burn In Test» або «Advanced Test», на думку Maxtor, можна вважати накопичувач вільним від помилок. Серед додаткових інструментальних можливостей у цьому наборі є тільки інструмент для низькорівневого форматування жорсткого диска (Write Disk Pack). Зрозуміло, при форматуванні буде знищена вся інформація. щоб запустити процес форматування потрібно на вимогу програми дати свою згоду, увівши із клавіатури слово «YES», що на екрані не відображається. Причому, Maxtor рекомендує відключити інші жорсткі диски, при запуску процесу форматування. Якщо спроба низькорівневого форматування за допомогою Powermax виявляється невдалою через існуючи на диску дефекти, то утиліта рекомендує запустити «Advanced Test», і це іноді допомагає. Важко сказати, чи реалізується виправлення помилок засобами самої Powermax, або ж під час навантаження розширеного тесту дефектні ділянки поверхні відновлюються внутрішньою мікропрограмою S.M.A.R.T. шляхом заміни з резервної області, але результат може бути позитивним і дефекти на поверхні диска будуть виправлені.

Samsung

Пропонована Samsung утиліта Shdiag під Windows не працює, власних завантажувальних дискет теж не створює. Їй потрібний DOS, може влаштувати й командний режим після перезавантаження Windows 9X. Визнає одні тільки накопичувачів від Samsung. На сайті виробника перебуває опис цієї програми ілюстроване чималим числом скриншотів. Хоча, по суті, ілюструвати там особливо-то нема чого. Ніяких варіантів користувачеві не пропонується, крім як

завжди погоджуватися на запропонований програмою наступний крок, або відмовлятися й припиняти діагностику.

При запуску Shdiag, як звичайно, спочатку з'являється меню вибору HDD. Утиліта правильно відображає в списку назва моделі будь-якого накопичувача, але починає роботу тільки при виборі HDD від Samsung.

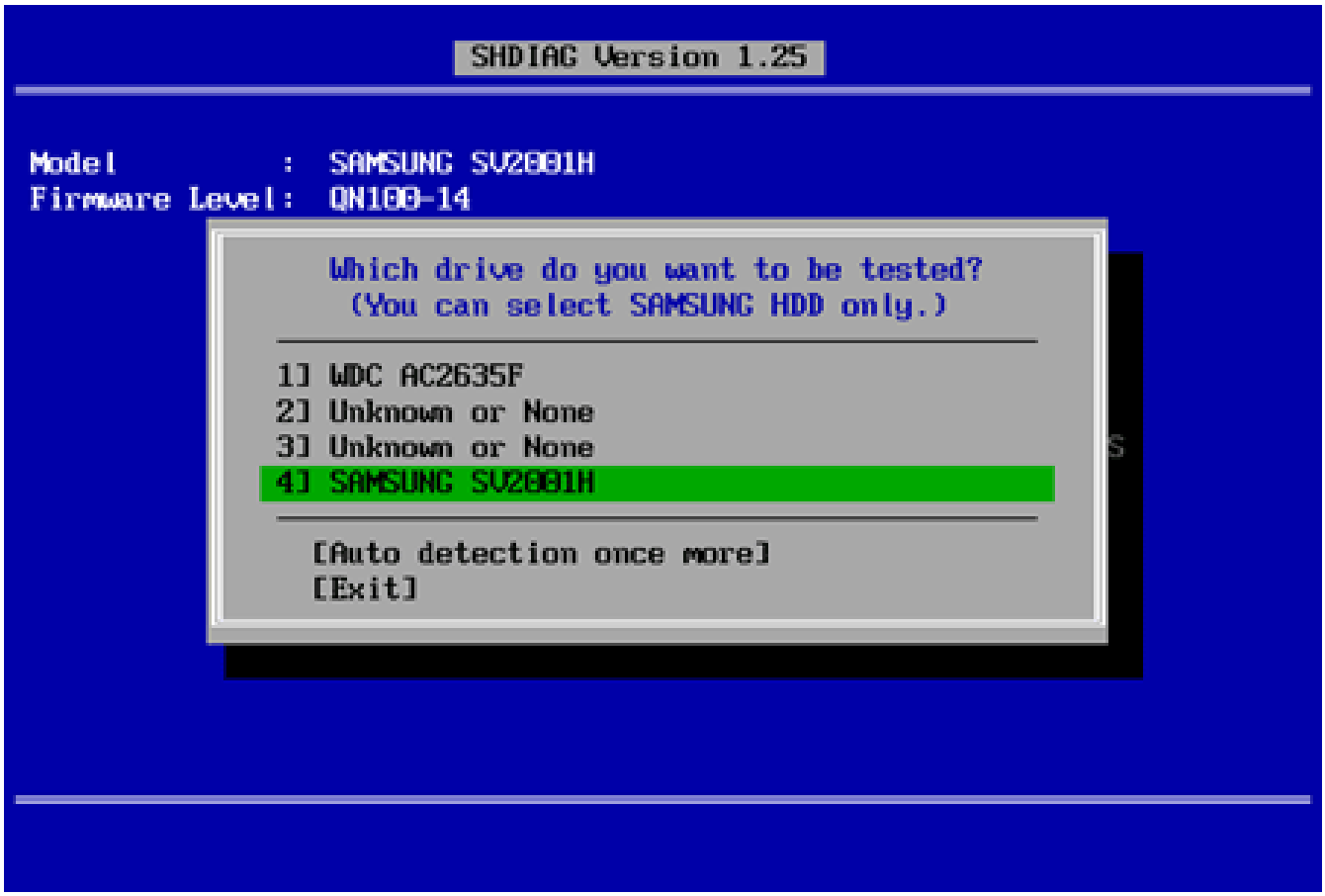

Рисунок 2.5 – Інтерфейс користувача Shdiag

Діагностика починається відразу ж після уведення обраної моделі вінчестера Samsung. Тестування Shdiag можна розділити на два етапи. Спочатку утиліта проводить серію коротких тестів для швидкої перевірки стану накопичувача. Сюди входить перевірка основних параметрів, стану S.M.A.R.T., а так само часткова перевірка поверхні. Результати всіх перевірок відображаються в списку. Після досить швидкого першого етапу діагностики, користувачеві

пропонується приступитися до другого етапу – скануванню всієї поверхні диска, що вже займає набагато більше часу. У процесі сканування посекторно перевіряється вся поверхня, індикатором протікання цього процесу є жовта сегментна смужка в нижній частині екрана. Наприкінці сканування видається результат про відсутність або наявність на диску помилок.

Якщо на диску були виявлені помилки, то єдине що може запропонувати Shdiag із власних засобів, так це зробити низькорівневе форматування. Однак способу примусового запуску процесу форматування за бажанням користувача не передбачено. Низькорівневе форматування доступно тільки тоді, якщо в процесі діагностики утилітою виявлені помилки, і його виконання автоматично пропонується самою програмою: користувач може або погодитися й втратити всі дані, або відмовитися, за своїм розсудом.

Fujitsu

Раніше Fujitsu активно просувала EIDE вінчестери для настільних систем. У користувачів на руках залишилася чимала кількість цих накопичувачів. Для роботи зі своїми EIDE і SCSI накопичувачами Fujitsu пропонує ряд утиліт. У діагностичних цілях рекомендується використовувати Fujitsu ATA Diagnostic Tool (FJDT) або SCSI Diagnostics (SDIAG). Працює FJDT у середовищі DOS, сама завантажувальних дискет не створює. Її можна запустити з будь-який іншої завантажувальної DOS-дискети або навіть перезавантажившись в Windows 9X у командному режимі. При запуску FJDT спочатку ініціалізує установлені в системі накопичувачі, при цьому для вінчестерів виробництва не Fujitsu навіть не відображається абревіатура моделі. Ясно, що вибрати для перевірки можна тільки HDD Fujitsu. При введенні обраного накопичувача Fujitsu утиліта пропонує запустити швидкий тест (Quick Test) або ..вийти із програми. Під час швидкого тесту перевіряється ряд технічних робочих параметрів, стан S.M.A.R.T. (без відображення атрибутів), проводиться часткова перевірка поверхні – сканується початкова й кінцева ділянка диска.

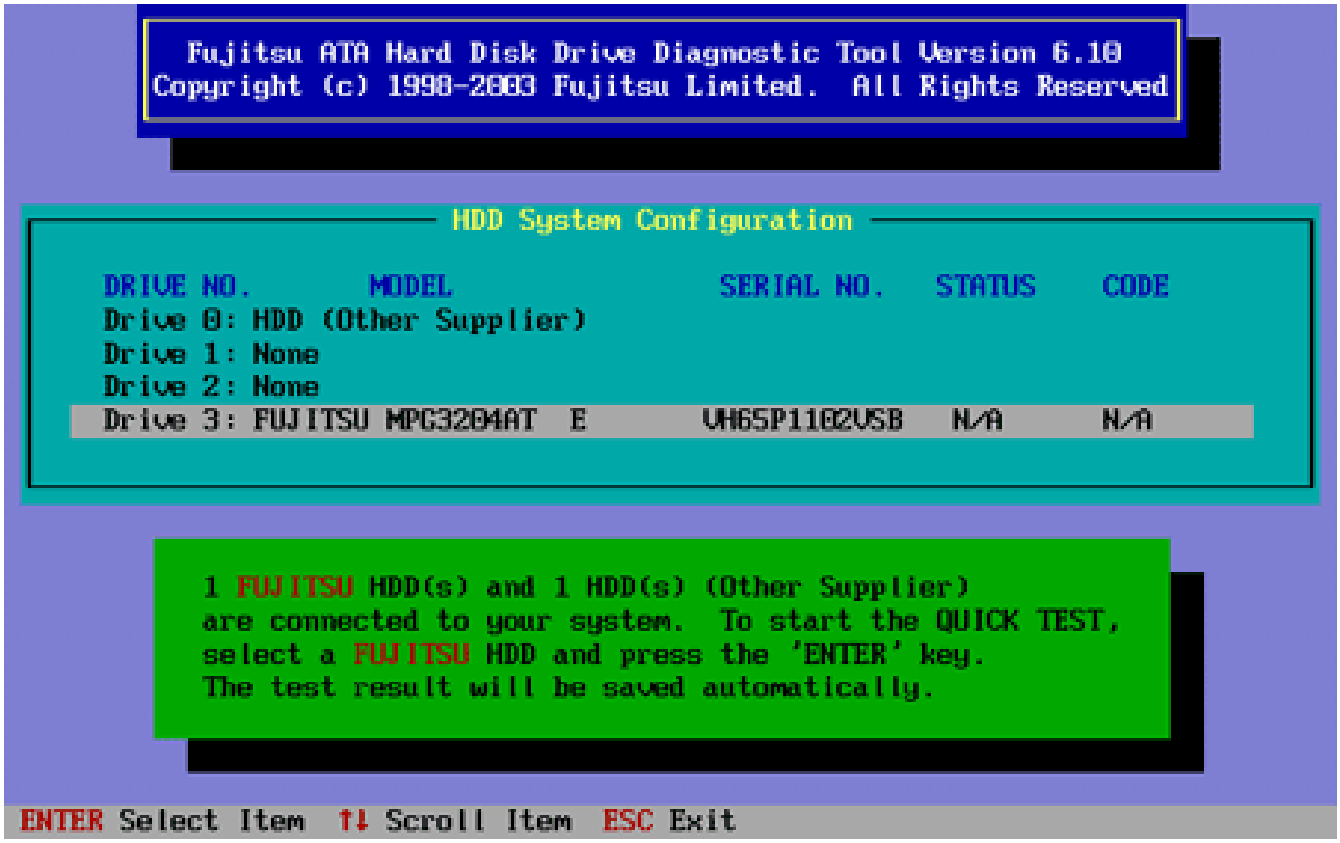

Рисунок 2.6 – Інтерфейс користувача Fujitsu ATA Diagnostic Tool

Тільки після проходження швидкого тесту доступний наступний крок – розширений тест (Comprehensive) і, знову ж, ніяких інших варіантів. Під час розширеної діагностики виконується ряд перевірок і сканується вже вся робоча поверхня накопичувача, це і є заключним етапом про прийняття програмою рішення про придатність накопичувача, тому що нічого іншого FJDT робити вже не запропонує. Ситуація з SDIAG аналогічна, тільки там передбачено три послідовних тести. Для обнуління змісту всього жорсткого диска, що рівносильно низькорівневому форматуванню, Fujitsu пропонує окрему утиліту – Fujitsu Erase Utility. Потрібно взяти до уваги, що на сайтах Fujitsu у різних країнах програми можуть мати трохи відмінну назву. Так на деяких сайтах доступна програма за назвою FJ-IDE Drive Initializer Utility, – спробуйте по її назві догадатися, що це теж засіб для тотального очищення вмісту накопичувача – так званий «ерайзер». Так що описи краще все-таки читати. Комусь може придатися утиліта [Ultra-

ATA] Change Mode Utility, призначена для примусового включення UDMA режимів.

Як стало видно з нашого дослідження, всі провідні виробники накопичувачів на жорстких дисках піклуються про програмне діагностичне забезпечення для своєї продукції. Інша розмова, наскільки функціонально це ПЗ? Зразком можуть служити набори утиліт Data Lifeguard Tools різних версій від Western Digital, та й взагалі та кількість утиліт, що пропонує ця компанія. Гарне враження залишає Drive Fitness Test від IBM, що поряд із завданнями діагностики, дає можливість вибіркового відновлення ушкоджених секторів без втрати інформації на іншій поверхні накопичувача. При усьому цьому, жодна з діагностичних утиліт від виробників HDD не може представляти тести читання/запису поверхні дисків у графічному виді. Але ж найчастіше саме графіки можуть дати найбільш об'єктивне подання про дійсний стан дисків, навіть якщо вся їхня поверхня номінально читається. Так само виробники HDD чомусь не приділяють належну увагу системі самодіагностики накопичувачів S.M.A.R.T., що є присутнім на всіх без винятку сучасних вінчестерах. Звичайно, діагностичне ПЗ перевіряє S.M.A.R.T. на наявність помилок, але от самі підконтрольні атрибути, а їх може бути більше двох десятків, більшістю утиліт користувачеві не представляються. Але ж вивчення поточних атрибутів S.M.A.R.T. може дати важливі відомості про стан накопичувача, навіть якщо там ще немає критичних для його стану значень.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст

повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього використовуються генератори застосунків. Програміст відповідає на питання генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би

дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об'єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний

розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи діагностування помилок жорсткого диску.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

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

Являє собою набір міні-підпрограм, які є частиною мікрокоду накопичувача й визначають підтримувані діагностичні функції. Найпоширеніші серед них:

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

– внутрішні тести накопичувача (self-test);

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

У даний момент не існує офіційної документації або стандарту на технологію S.M.A.R.T. у своїх накопичувачах. Обов'язковий мінімум описаний в останньому стандарті ATA/ ATAPI-6.

У зв'язку із цим, виробники не публікують повні характеристики й підтримувані функції

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

S.M.A.R.T. I передбачав моніторинг основних життєво важливих параметрів і запускався тільки після команди по інтерфейсі в S.M.A.R.T. II

з'явилася можливість фонової перевірки поверхні, що виконувалася накопичувачем автоматично під час "холостого ходу"; з'явилася функція журналювання помилок в S.M.A.R.T. III уперше з'явилася не тільки функція виявлення дефектів поверхні, але й можливість їхнього відновлення "прозоро" для користувача й багато інших нововведень

Історія S.M.A.R.T. починається в 1992 році, коли інженери корпорації IBM розробили технологію, що відслідковує декілька критично важливих параметрів накопичувача, а також алгоритм, здатний пророчити вихід диска з ладу на підставі зібраних даних. Запропонована технологія одержала назву “Predictive Failure Analysis” (PFA), що можна перевести як “Аналіз Можливих Відмов”.

Ідею IBM підхопила Compaq, що трохи пізніше створила свою технологію – IntelliSafe. До розробок Compaq підключилися такі ветерани індустрії накопичувачів, як Seagate, Quantum і Conner. Створена альянсом технологія також відслідковувала ряд робочих характеристик диска, порівнювала їх із припустимим значенням і видавала попередження у випадку небезпеки.

Поява на ринку жорстких дисків з можливістю самодіагностики знаменувало величезний крок уперед якщо й не в підвищенні надійності вінчестерів, те хоча б у зменшенні ризику втрати інформації при їхньому використанні. Практика використання першого покоління технологій діагностики виявилася успішної, але показала необхідність подальшого розвитку.

Саме “пробні кулі” – технології PFA і IntelliSafe лягли в основу майбутньої технології S.M.A.R.T., розробка якої почалася під крилом об'єднання всіх найбільших виробників жорстких дисків. На цей момент часу технологія S.M.A.R.T. пройшла вже три етапи у своєму розвитку. Настільки активний розвиток пішов на користь технології – якщо перші версії дозволяли прогнозувати можливі збої з імовірністю близько 20%, то версія S.M.A.R.T. III дозволяє попередити про прийдешній збій уже в 60% випадків.

Сучасний етап розвитку технології S.M.A.R.T. ґрунтується на наступних принципах:

– Відстеження значень важливих параметрів накопичувача.

– Переміщення збійних секторів.

– Протоколювання роботи диска за допомогою записів у спеціальних журналах.

– Виконання внутрішніх тестів, що дозволяють оцінити стан накопичувача.

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

От кілька нових функцій, що взялли участь у розвитку технології:

– введення алгоритму аналізу температурного режиму накопичувача;

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

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

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

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

– відносна стійкість (стабільність "польоту") головки;

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

– автоматичний перерозподіл дефектних ділянок поверхні при операціях запису;

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

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

S.M.A.R.T. – це набір програм, що вшитих у мікрокод вінчестера. Кожна фірма-виробник дисків веде свої розробки, звідси й розмаїтість параметрів для різних дисків. Однак існують загальні параметри:

1. Атрибути, що відбивають загальний стан диска (приблизно 30).

2. Внутрішні тести (self-tests).

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

Повний обов'язковий перелік S.M.A.R.T атрибутів описаний у стандарті ATA/ATAPI-6.

Атрибути S.M.A.R.T.

Атрибути S.M.A.R.T. – особливі характеристики, які використовуються при аналізі стану й запасу продуктивності накопичувача. Вони вибираються виробником, ґрунтуючись на їхній здатності пророкувати погіршення робочих характеристик накопичувача або визначити його дефектність.

Значення атрибутів (value) використовуються для подання відносної надійності окремого експлуатаційного або еталонного атрибута. Припустиме значення атрибута лежить у діапазоні від 1 до 255. Його високе значення 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.

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 через конструкційні особливості накопичувача й аналіз результатів випробувань на надійність. Граничне значення кожного атрибута вказує на його нижню припустиму границю, до якої накопичувач нормально функціонує.

Нижче наведене короткий опис основних атрибутів:

– 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.

– Start/Stop Count – Кількість циклів запуску/останову шпинделя. Зберігає загальну кількість включень/вимикань диска.

– Reallocated Sectors Count – Кількість перепризначених секторів. Коли жорсткий диск зустрічає помилку читання/запису/верифікації, він намагається перемістити дані 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).

– Seek Error Rate – Частота появи помилок позиціонування МГ (магнітної головки). У випадку збою в механічній системі позиціонування, ушкодження сервоміток (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 МГ. Даний параметр показує середню швидкість позиціонування привода МГ на зазначений сектор. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Power-On Hours – Кількість відпрацьованих годин у включеному стані. Значення value цього атрибута показує кількість годин (мінут, секунд – залежно від виробника), відпрацьованих жорстким диском. Зниження значення атрибута до критичного рівня (threshold) вказує на виробіток ресурсу диском. На практиці, навіть падіння цього атрибута до нульового значення не завжди вказує на реальне вичерпання 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 рекалібрування 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.

– Load/Unload Cycle Count – Кількість циклів виводу МГ у спеціальну парковочну зону/у робоче положення.

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

– Reallocation Event Count – Кількість операцій перепризначення (ремаппінгу). Показує загальна 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. Показує загальна кількість секторів, які накопичувач у цей момент вважає 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).

– Uncorrectable Sector Count – Кількість нескоректованих помилок. Атрибут показує загальну кількість помилок, що виникли при читанні/запису сектора, 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, містить кількість помилок, що виникли в режимі передачі даних UltraDMA у контрольній сумі (ICRC – Interface CRC). У більшості випадків помилки CRC виникають при сильному завищенні частоти PCI (більше номінальних 33.3 MHz), сильно перекрученому кабелі, а також – з вини драйверів ОС, які не дотримують вимог до передачі/прийому даних у режимах UltraDMA.

– Write Error Rate – Частота появи помилок при записі даних. Показує загальна кількість помилок, виявлених під час запису сектора. Чим нижче значення value, тим гірше стан поверхні диска й/або механіки привода.

– Disk Shift – Зрушення пакета дисків щодо осі шпинделя.

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

Тут наведені атрибути, за допомогою яких можна визначити надійність функціонування диска. Інші ж не представляють практичної важливості.

Перелічимо основні критичні атрибути, на які варто звернути увагу при оцінці стану жорсткого диска. Всі програми оцінки “здоров'я” накопичувача у своїй роботі опираються саме на них:

– Raw Read Error Rate (Відсоток Помилок Низькорівневого Читання) – частота появи помилок читання, обумовлених апаратною частиною диска.

– Spin Up Time (Час Розкручування) – час розкручування пакета дисків зі стану спокою до робочої швидкості.

– Spin Up Retry Count (Лічильник Повторів Розкручування) – число повторних спроб розкручування дисків, у випадку, якщо перша спроба була невдалою.

– Seek Error Rate (Відсоток Помилок Пошуку) – частота помилок при позиціонуванні блоку головок.

– Reallocated Sector Count (Лічильник Переміщених Секторів) – кількість переміщених збійних секторів.

Нижче дається неповний список некритичних атрибутів, які, проте, безпосередньо впливають на показники роботи накопичувача:

– Start/Stop Count (Лічильник Запусків/Зупинок) – повне число запусків/остановів шпинделя.

– Power On Hours (Відпрацьований час) – число годин, проведених у включеному стані.

– Drive Power Cycle Count (Лічильник Включень Накопичувача) – кількість повних циклів включення/вимикання диска.

– Temperatue (Температура) – температура диска за показниками вбудованого термодатчика (у градусах Цельсія).

Інтерпретація значень атрибутів

Як же інтерпретувати значення атрибутів для вашого диска? Як зробити вивід про те, що накопичувач справний, або, навпаки – незабаром вийде з ладу?

Методика оцінки така. Насамперед необхідно звернути увагу на критично важливі атрибути, такі як кількість переміщених секторів (Reallocated Sector Count) або частота появи помилок (Raw Read Error Rate).

За допомогою програми для роботи з інформацією S.M.A.R.T., зберіть дані про значення атрибутів і пороги. Як правило, програми подібного роду виводять інформацію не тільки в числовому виді, але й у тій або іншій формі, що дозволяє візуально визначити якість показника, наприклад у вигляді кольорової смуги, де гарним показникам відповідає зелений колір, а поганим – червоний.

Розглянемо, як інтерпретувати числові показники. Допустимо, ми оцінюємо атрибут Reallocated Sector Count – кількість переміщених секторів. Припустимо, що поточне значення атрибута дорівнює 85, а граничним є значення 60. Що це означає? В ідеалі, значення атрибута повинне рівнятися 100 – але навіть нові диски мають менші значення, тобто частина секторів уже переміщена в резервні області диска. У нашому випадку безпечний діапазон зміни атрибута становить 40 пунктів, від 60 до 100, тому ми можемо умовно розбити його на 100 розподілів, відповідно до процентної шкали. Поточне значення атрибута на 25 пунктів більше граничного й на 15 менше ідеального, тобто здоров'я диска по цьому показнику у відсотковому відношенні становить 60% відповідно до простої пропорції: 40 ~ 100%, 25 ~ x%, x = (25 * 100) / 40 = 60%.

Саме значення 60% говорить про те, що кількість переміщених секторів перебуває в робочих межах. Однак для вірної діагностики набагато важливіша динаміка показника – його зміна в часі. Кількість переміщених секторів може залишатися практично на тому самому рівні досить довго, можливо – роками, а може дуже різко піти долілиць, дібравшись до критичної оцінки всього за два тижні, наприклад, при погіршенні умов експлуатації накопичувача.

Саме тому необхідно постійно відслідковувати як значення показників, так і динаміку їхньої зміни. Сучасні програми моніторингу стану S.M.A.R.T. реалізують різні математичні алгоритми, здатні пророчити ймовірність збоїти або відмови й дату настання цієї події, з огляду на всі доступні атрибути.

Автономне сканування поверхні (off-line read scanning)

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

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

Вбудовані функції самоконтролю (self-test)

Практично з моменту появи стандарту S.M.A.R.T. II, у більшості накопичувачів з'явилася нова функція – внутрішня діагностика й самоконтроль, для поглибленого контролю стану механіки накопичувача, поверхні дисків і т.п. Для запуску цієї функції, у набір команд S.M.A.R.T. була уведена нова команда – S.M.A.R.T. EXECUTE OFF-LINE IMMEDIATE. Результат роботи зберігається або в спеціалізованих атрибутах, або окремим параметром серед інших даних в атрибутах. Після виконання тесту, накопичувач в обов'язковому порядку обновляє показання у всіх атрибутах і інших параметрах. Якщо під час виконання внутрішнього тесту накопичувач одержить по інтерфейсі нову команду, то виконання тесту переривається й накопичувач приступає до обробки команди, що надійшла.

Методи тестування

Існує два способи запуску тестів S.M.A.R.T.: автономний (off-line) або монопольний (captive). Результат тесту завжди зберігається накопичувачем у даних S.M.A.R.T.

При автономному запуску накопичувач повідомляє про успішне завершення команди до її фактичного виконання й тільки після цього виконує тест. При цьому, по інтерфейсу прапор "зайнятий" (busy) не виставляється й накопичувач у будь-який момент готовий приступитися до виконання чергової інтерфейсної команди, припиняючи роботу тесту. Фактично, тест виконується у фоновому режимі.

При запуску тесту в монопольному режимі, по інтерфейсу виставляється прапор "зайнятий" (busy) і накопичувач починає безпосереднє виконання тесту в режимі реального часу. Будь-яка інтерфейсна команда під час виконання цього тесту приведе до його переривання й зупинки, після чого накопичувач приступиться до обробки команди, що надійшла.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленої системи діагностування помилок жорсткого диску з використанням технології S.M.A.R.T.

Структурна схема складається з наступних блоків:

– Жорсткий диск який діагностується на наявність помилок.

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

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

– Блок одержання таблиці в буфер додатка.

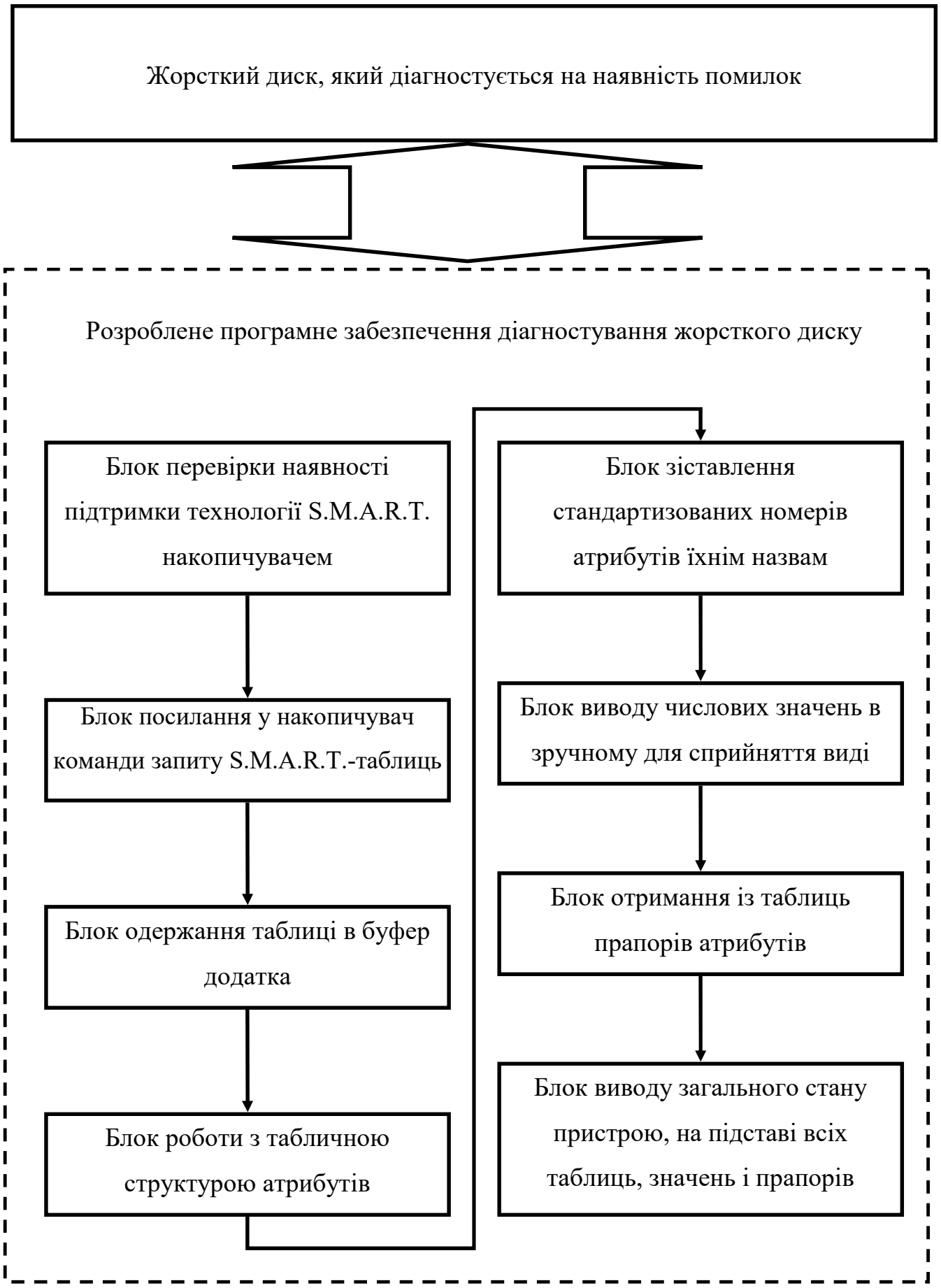

Рисунок 3.1 – Структурна схема системи

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

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

– Блок виводу числових значень в зручному для сприйняття виді.

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

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

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

1. Блок читання типів атрибутів:

– 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.).

2. Блок читання атрибутів:

– 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.

– 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.

3. Блок автономного сканування поверхні.

4. Блок читання журналу помилок:

– Каталог журналів S.M.A.R.T.

– Сумарний журнал помилок.

– Комплексний журнал помилок.

– Розширений комплексний журнал помилок.

– Журнал результатів самоконтролю.

– Розширений журнал результатів самоконтролю.

– Журнал параметрів продуктивності потоків.

– Журнал помилок потокового запису.

– Журнал помилок потокового читання.

– Журнал непоправних помилок.

– Користувальницькі журнали.

– Технічні журнали виробника.

Рисунок 3.2 – Функціональна схема системи

5. Блок вбудованих функцій самоконтролю.

6. Блок вибору методів тесту:

– автономний (off-line);

– монопольний (captive).

7. Блок набору «активних» тестів.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. З нього видно, що першим процесом, який запускається у системі є процес виведення списку дисків встановлених у системі.

Цей процес взаємодіє з процесом вибору диску. Процес вибору диску, у свою чергу, взаємодіє з наступними процесами:

– Процес виведення загальної інформації про диск.

– Процес запуску діагностики диску.

Останній процес взаємодіє з процесом перевірки наявності підтримки технології SMART.

Якщо така технологія підтримується виробником жорсткого диску, тоді запускається процес запиту SMART-таблиці.

Ця таблиця записується у буфер програми. Після запису таблиці запускається процес одержання із таблиці прапорів атрибутів.

Цей процес взаємодіє з наступними процесами:

– Процес порівняння одержаних значень атрибута із допустимими.

– Процес пошуку несправностей.

– Процес аналізу загального стану пристрою.

Останній процес взаємодіє з процесом виведення результатів діагностики.

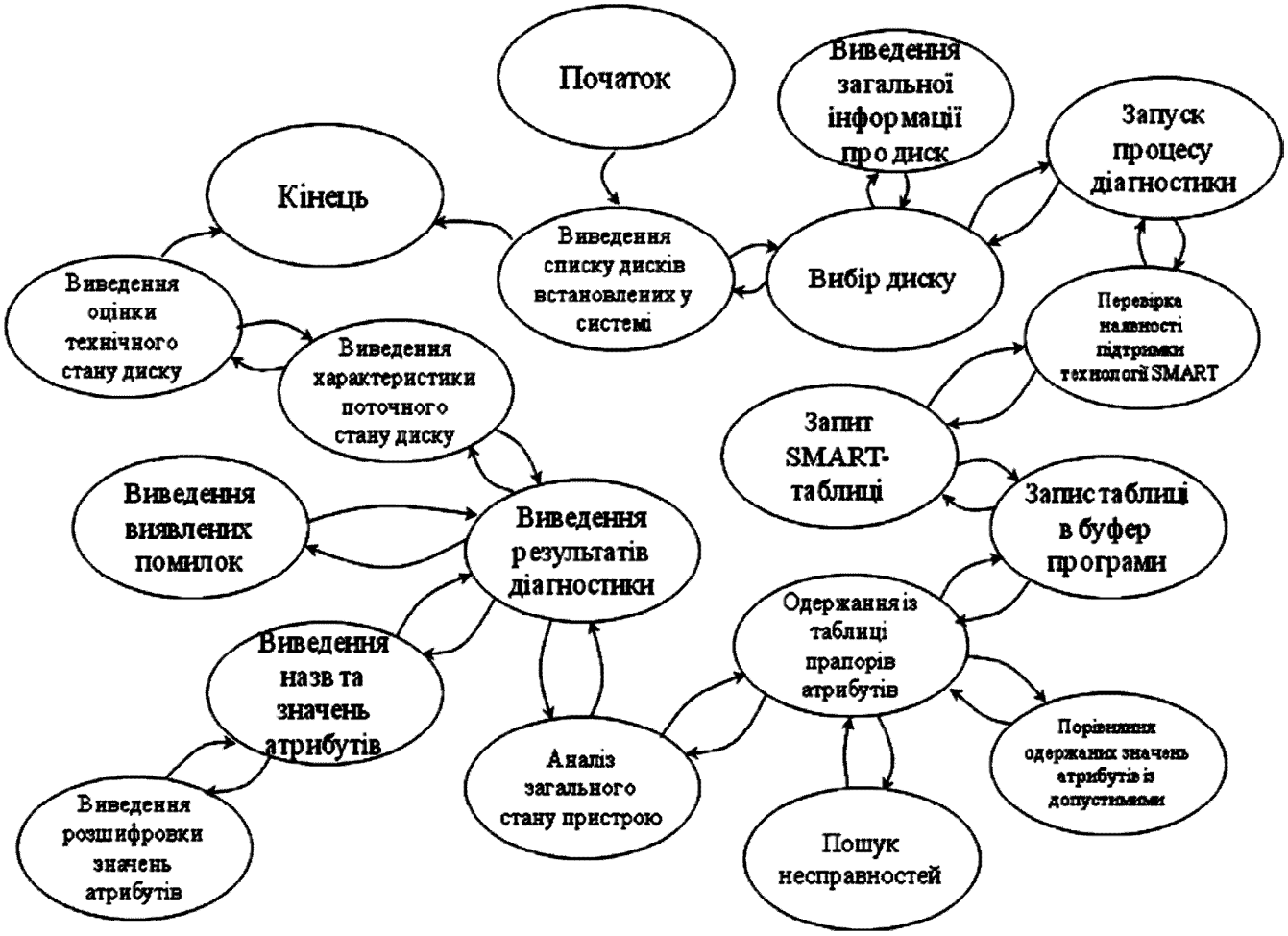

Рисунок 3.3 – Діаграма процесів системи

Процес виведення результатів діагностики взаємодіє з наступними процесами:

– Процес виведення назв та значень атрибутів, якій взаємодіє з процесом виведення розшифровки значень атрибутів.

– Процес виведення виявлених помилок.

– Процес виведення характеристики поточного стану диску.

Останній процес взаємодіє з процесом виведення оцінки технічного стану диску, який є завершальним у системі.

Розглянувши діаграму взаємодії процесів, перейдемо до розгляду блок-схем алгоритмів основної програми та підпрограм.

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. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається ініціалізація регістрів контролерів жорстких дисків.

Наступним етапом є зчитування значень регістрів.

Після цього ці зчитані значення регістрів оброблюються.

Завершальною ітерацією цього етапу є вивід повідомлення про список дисків встановлених у системі.

Після цього користувач обирає диск, який необхідно перевірити на помилки.

Виводиться загальна інформація про обраний диск.

Якщо потрібно продіагностувати обраний диск на наявність помилок, то запускається підпрограма діагностування диску, після виконання якої виводяться наступні дані:

– Поточний стан пристрою.

– Назви та значення атрибутів.

– Розшифровка значень атрибутів.

– Виявлені помилки.

– Характеристика та оцінка технічного стану диску.

Після усіх вище перерахованих операцій користувач обирає працювати йому далі з програмою, або ні.

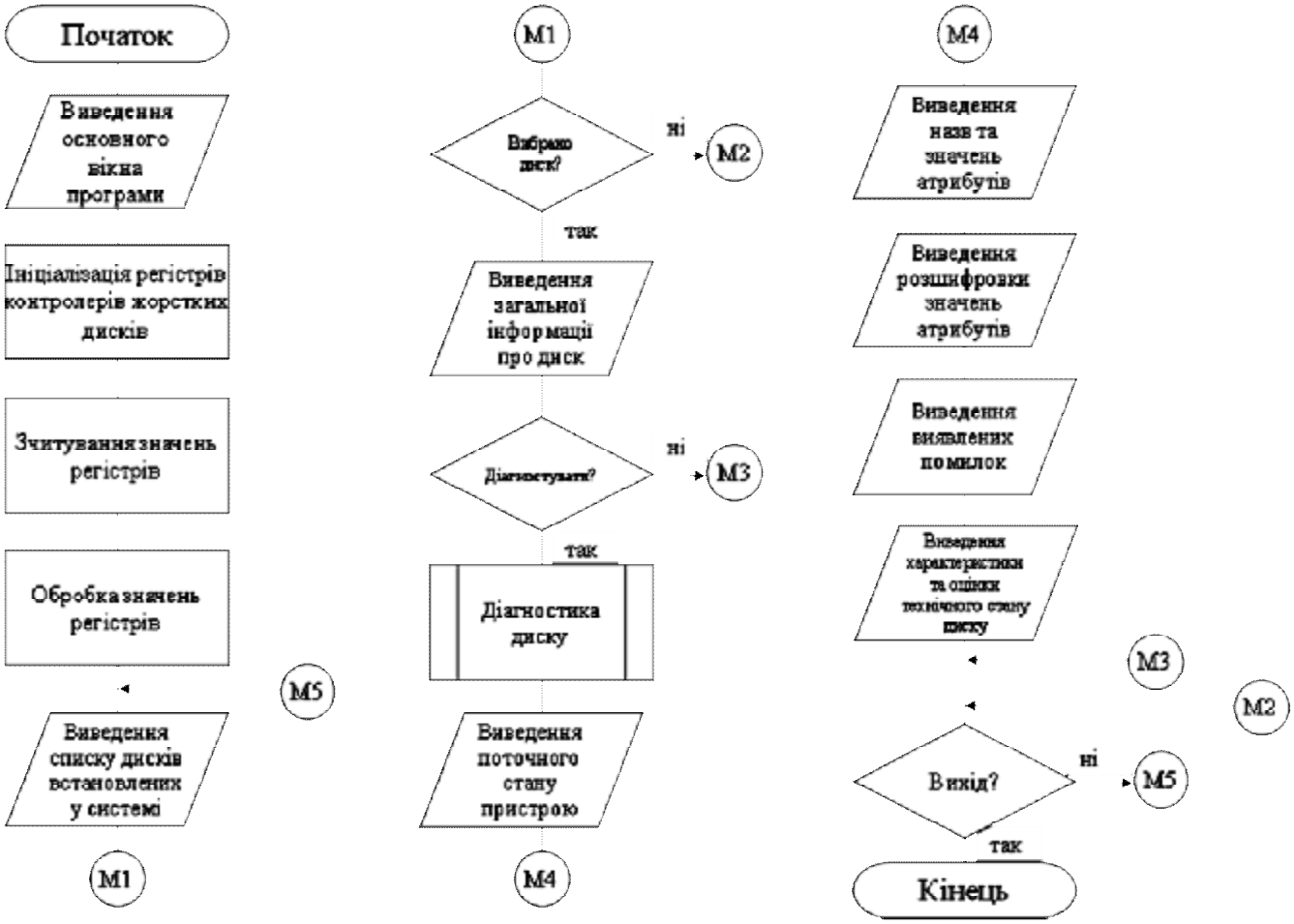

Рисунок 4.1 – Блок-схема роботи основної програми

На рисунку 4.2 зображено блок-схему підпрограми діагностики жорсткого диску.

Вона працює наступним чином.

Спершу відбувається перевірка наявності підтримки технології SMART накопичувачем.

Якщо диск підтримує технологію SMART, тоді відбувається поетапне виконання наступних кроків:

– Запит SMART таблиці.

– Запис таблиці у буфер програми.

– Робота з табличною структурою.

– Зіставлення стандартизованих номерів атрибутів їхнім назвам.

– Отримання таблиць номерів атрибутів.

– Порівняння одержаних значень атрибутів з допустимими.

– Пошук несправностей.

– Аналіз загального стану пристрою.

Після цього підпрограма закінчує свою роботу.

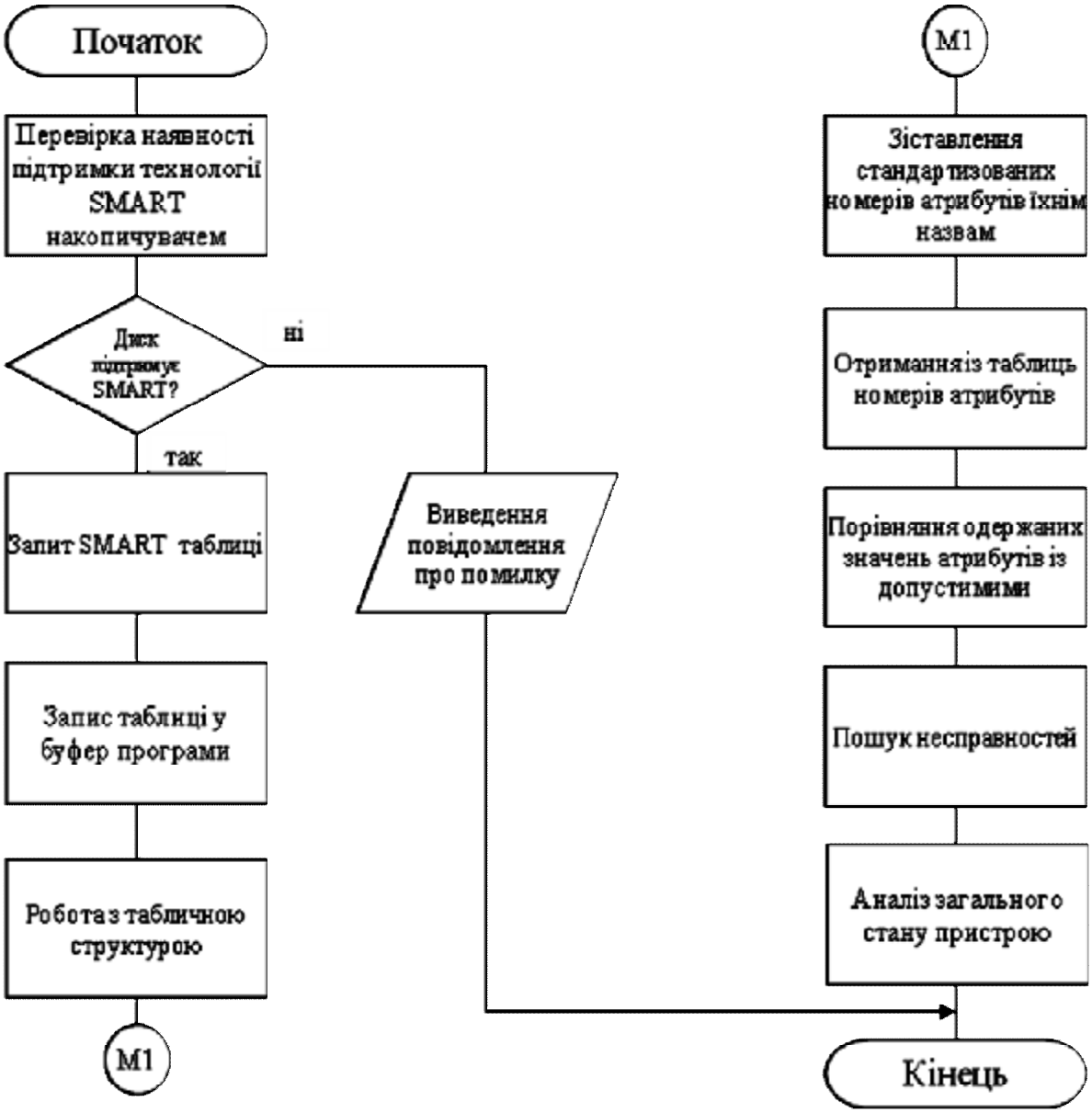

Рисунок 4.2 – Блок-схема роботи підпрограми діагностики жорсткого диску

Наведемо частину коду, яка реалізує деякі функції розробленої, у ході виконання магістерського проектування, системи.

```
// Визначення ATA
CTestdisk::CTestdisk()
{
     BOOL bosVersionInfoEx;
     ZeroMemory(&m_Os, sizeof(OSVERSIONINFOEX));
     m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
     if(!(bosVersionInfoEx = GetVersionEx((OSVERSIONINFO *)&m_Os)))
     {
          m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
          GetVersionEx((OSVERSIONINFO *)&m_Os);
     }
     m_FlagAtaPassThrough = FALSE;
     if(m_Os.dwMajorVersion >= 6 || (m_Os.dwMajorVersion == 5 &&
m_Os.dwMinorVersion == 2))
     {
          m_FlagAtaPassThrough = TRUE;
     }
     else if(m_Os.dwMajorVersion == 5 && m_Os.dwMinorVersion == 1)
     {
          CString cstr;
          cstr = m_Os.szCSDVersion;
          cstr.Replace(_T("Service Pack "), _T(""));
          if(_tstoi(cstr) >= 2)
          {
               m_FlagAtaPassThrough = TRUE;
          }
     }
}
//Перевірка атрибутів SMART
DWORD CTestdisk::CheckSmartAttributeUpdate(DWORD index, SMART_ATTRIBUTE* pre,
SMART_ATTRIBUTE* cur)
{
     if(memcmp(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE) == 0)
     {
          return SMART_STATUS_NO_CHANGE;
     }
     else
     {
          for(int i = 0; i < MAX_ATTRIBUTE; i++)
```

```
          {
               switch(cur[i].Id)
               {
               case 0x09: // Кількість відпрацьованих годин у включеному стані
                    {
                         DWORD preRawValue = MAKELONG(
                              MAKEWORD(pre[i].RawValue[0], pre[i].RawValue[1]),
                              MAKEWORD(pre[i].RawValue[2], pre[i].RawValue[3])
                              );
                         DWORD curRawValue = MAKELONG(
                              MAKEWORD(cur[i].RawValue[0], cur[i].RawValue[1]),
                              MAKEWORD(cur[i].RawValue[2], cur[i].RawValue[3])
                              );
                         if(GetPowerOnHours(preRawValue,
vars[index].DetectedTimeUnitType)
                         != GetPowerOnHours(curRawValue,
vars[index].DetectedTimeUnitType))
                         {
                              memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                              return SMART_STATUS_MAJOR_CHANGE;
                         }
                         if(GetPowerOnHours(preRawValue,
vars[index].MeasuredTimeUnitType)
                         != GetPowerOnHours(curRawValue,
vars[index].MeasuredTimeUnitType))
                         {
                              memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                              return SMART_STATUS_MAJOR_CHANGE;
                         }
                    }
                    break;
               case 0x0C: // Кількість зафіксованих повторів включення/вимикання
живлення накопичувача
                    {
                         DWORD preRawValue = MAKELONG(
                              MAKEWORD(pre[i].RawValue[0], pre[i].RawValue[1]),
                              MAKEWORD(pre[i].RawValue[2], pre[i].RawValue[3])
                              );
                         DWORD curRawValue = MAKELONG(
                              MAKEWORD(cur[i].RawValue[0], cur[i].RawValue[1]),
```

```
                              MAKEWORD(cur[i].RawValue[2], cur[i].RawValue[3])
                              );
                         if(preRawValue != curRawValue)
                         {
                              memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                              return SMART_STATUS_MAJOR_CHANGE;
                         }
                    }
                    break;
               case 0xC2: // Температура
                    if(pre[i].RawValue[0] != cur[i].RawValue[0]
                    || pre[i].CurrentValue != cur[i].CurrentValue)
                    {
                         memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE);
                         return SMART_STATUS_MAJOR_CHANGE;
                    }
                    break;
               default:
                    break;
               }
          }
          return SMART_STATUS_MINOR_CHANGE;
     }
}
     // \\\\.\\PhysicalDrive%d
     for(int i = 0; i < MAX_SEARCH_PHYSICAL_DRIVE; i++)
     {
          BOOL bRet;
          HANDLE    hIoCtrl;
          DWORD     dwReturned;
          DISK_GEOMETRY dg;
          hIoCtrl = GetIoCtrlHandle(i);
          if(hIoCtrl == INVALID_HANDLE_VALUE)
          {
               continue;
          }
          bRet = ::DeviceIoControl(hIoCtrl, IOCTL_DISK_GET_DRIVE_GEOMETRY,
               NULL, 0, &dg, sizeof(DISK_GEOMETRY),
               &dwReturned, NULL);
          ::CloseHandle(hIoCtrl);
```

```
          if(bRet == FALSE || dwReturned != sizeof(DISK_GEOMETRY) || dg.MediaType
!= FixedMedia)
          {
               continue;
          }
// Визначення виробника
          if(GetMain(i, -1, -1, INTERFACE_TYPE_UNKNOWN, VENDOR_UNKNOWN))
          {
               int index = (int)vars.GetCount() - 1;
               CString cmp;
               cmp = vars[index].Model;
               if(cmp.Find(_T("DW C")) == 0 // WDC
               || cmp.Find(_T("iHat")) == 0 // Hitachi
               || cmp.Find(_T("ASSM")) == 0 // SAMSUNG
               || cmp.Find(_T("aMtx")) == 0 // Maxtor
               || cmp.Find(_T("OTHS")) == 0 // TOSHIBA
               || cmp.Find(_T("UFIJ")) == 0 // FUJITSU
               )
               {
                    vars[index].SerialNumber = vars[index].SerialNumberReverse;
                    vars[index].FirmwareRev = vars[index].FirmwareRevReverse;
                    vars[index].Model = vars[index].ModelReverse;
                    vars[index].ModelSerial = vars[index].Model +
vars[index].SerialNumber;
                    vars[index].ModelSerial.Replace(_T("/"), _T(""));
               }
          }
     }
     // сортування
     ATA_SMART_INFO* p = vars.GetData();
     qsort(p, vars.GetCount(), sizeof(ATA_SMART_INFO), Compare);
     DebugPrint(_T("OK:GetMain - PhysicalDrive"));
     // Визначення типу знайдених дисків
     if(advancedDiskSearch)
     {
          // \\\\.\\Scsi%d:
          for(int i = 0; i < MAX_SEARCH_SCSI_PORT; i++)
          {
               for(int j = 0; j < MAX_SEARCH_SCSI_TARGET_ID; j++)
               {
                    if(GetMain(-1, i, j, INTERFACE_TYPE_UNKNOWN, VENDOR_UNKNOWN))
                    {
```

```
                        int index = (int)vars.GetCount() - 1;
                        CString cmp;
                        cmp = vars[index].Model;
                        if(cmp.Find(_T("DW C")) == 0 // WDC
                        || cmp.Find(_T("iHat")) == 0 // Hitachi
                        || cmp.Find(_T("ASSM")) == 0 // SAMSUNG
                        || cmp.Find(_T("aMtx")) == 0 // Maxtor
                        || cmp.Find(_T("OTHS")) == 0 // TOSHIBA
                        || cmp.Find(_T("UFIJ")) == 0 // FUJITSU
                        )
                        {
                             vars[index].SerialNumber =
vars[index].SerialNumberReverse;
                             vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
                             vars[index].Model = vars[index].ModelReverse;
                             vars[index].ModelSerial = vars[index].Model +
vars[index].SerialNumber;
                             vars[index].ModelSerial.Replace(_T("/"), _T(""));
                        }
                   }
              }
         }
         DebugPrint(_T("OK:GetMain - Scsi"));
    }
    MeasuredGetTickCount = GetTickCount();
    DebugPrint(_T("CTestdisk::Init - Complete"));
    if(flagChangeDisk != NULL)
    {
         if(vars.GetCount() != previous.GetCount())
         {
              *flagChangeDisk = TRUE;
         }
         else
         {
              for(int i = 0; i < vars.GetCount(); i++)
              {
                   if(vars.GetAt(i).PhysicalDriveId !=
previous.GetAt(i).PhysicalDriveId
                   || vars.GetAt(i).ScsiTargetId != previous.GetAt(i).ScsiTargetId
                   || vars.GetAt(i).ScsiPort != previous.GetAt(i).ScsiPort
                   )
```

```
                    {
                         *flagChangeDisk = TRUE;
                         break;
                    }
               }
          }
     }
//Додавання нового диску, та робота з ним
BOOL CTestdisk::AddDisk(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
COMMAND_TYPE commandType, IDENTIFY_ПРИСТРІЙ* identify)
{
     ATA_SMART_INFO asi;
     memcpy(&(asi.IdentifyDevice), identify, sizeof(IDENTIFY_ПРИСТРІЙ));
     asi.PhysicalDriveId = physicalDriveId;
     asi.ScsiPort =  scsiPort;
     asi.ScsiTargetId = scsiTargetId;
     asi.CommandType = commandType;
     asi.CommandTypeString = commandTypeString[commandType];
     for(int i = 0; i < MAX_ATTRIBUTE; i++)
     {
          ::ZeroMemory(&(asi.Attribute[i]), sizeof(SMART_ATTRIBUTE));
          ::ZeroMemory(&(asi.Threshold[i]), sizeof(SMART_THRESHOLD));
     }
//Попередні встановлення
     asi.IsSmartEnabled = FALSE;
     asi.IsWord88 = FALSE;
     asi.IsWord64_76 = FALSE;
     asi.IsCheckSumError = FALSE;
     asi.IsSmartSupported = FALSE;
     asi.IsLba48Supported = FALSE;
     asi.IsNcqSupported = FALSE;
     asi.IsAamSupported = FALSE;
     asi.IsApmSupported = FALSE;
     asi.IsAamEnabled = FALSE;
     asi.IsApmEnabled = FALSE;
     asi.IsNvCacheSupported = FALSE;
     asi.IsMaxtorMinute = FALSE;
     asi.TotalDiskSize = 0;
     asi.Cylinder = 0;
     asi.Head = 0;
     asi.Sector = 0;
     asi.Sector28 = 0;
```

```
asi.Sector48 = 0;
asi.DiskSizeChs = 0;
asi.DiskSizeLba28 = 0;
asi.DiskSizeLba48 = 0;
asi.BufferSize = 0;
asi.NvCacheSize = 0;
asi.TransferModeType = 0;
asi.DetectedTimeUnitType = 0;
asi.MeasuredTimeUnitType = 0;
asi.AttributeCount = 0;
asi.DetectedPowerOnHours = -1;
asi.MeasuredPowerOnHours = -1;
asi.PowerOnRawValue = -1;
asi.PowerOnStartRawValue = -1;
asi.PowerOnCount = 0;
asi.Temperature = 0;
asi.Speed = 0.0;
asi.Life = -1;
asi.Major = 0;
asi.Minor = 0;
asi.DiskStatus = 0;
asi.DriveLetterMap = 0;
asi.AlarmTemperature = 0;
asi.VendorId = VENDOR_UNKNOWN;
asi.InterfaceType = INTERFACE_TYPE_UNKNOWN;
asi.VendorId = 0;
asi.ProductId = 0;
asi.SerialNumber = _T("");
asi.FirmwareRev = _T("");
asi.Model = _T("");
asi.ModelReverse = _T("");
asi.ModelWmi = _T("");
asi.ModelSerial = _T("");
asi.DriveMap = _T("");
asi.MaxTransferMode = _T("");
asi.CurrentTransferMode = _T("");
asi.MajorVersion = _T("");
asi.MinorVersion = _T("");
asi.Interface = _T("");
asi.Enclosure = _T("");
CHAR buf[64];
// Перевірка помилок
```

```
    BYTE sum = 0;
    BYTE checkSum[IDENTIFY_BUFFER_SIZE];
    memcpy(checkSum, (void *)identify, IDENTIFY_BUFFER_SIZE);
    for(int j = 0; j < IDENTIFY_BUFFER_SIZE; j++)
    {
        sum += checkSum[j];
    }
    if(sum != 0)
    {
        asi.IsCheckSumError = TRUE;
    }
    // Оборотні дії
    strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify->SerialNumber));
    asi.SerialNumberReverse = buf;
    asi.SerialNumberReverse.TrimLeft();
    asi.SerialNumberReverse.TrimRight();
    strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
    asi.FirmwareRevReverse = buf;
    asi.FirmwareRevReverse.TrimLeft();
    asi.FirmwareRevReverse.TrimRight();
    strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
    asi.ModelReverse = buf;
    asi.ModelReverse.TrimLeft();
    asi.ModelReverse.TrimRight();
    ChangeByteOrder(identify->SerialNumber, sizeof(identify->SerialNumber));
    ChangeByteOrder(identify->FirmwareRev, sizeof(identify->FirmwareRev));
    ChangeByteOrder(identify->Model, sizeof(identify->Model));
    if(CheckAsciiStringError(identify->SerialNumber, sizeof(identify-
>SerialNumber))
    || CheckAsciiStringError(identify->FirmwareRev, sizeof(identify-
>FirmwareRev))
    || CheckAsciiStringError(identify->Model, sizeof(identify->Model)))
    {
        return FALSE;
    }
    // Запис даних у структуру
    strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify->SerialNumber));
    asi.SerialNumber = buf;
    asi.SerialNumber.TrimLeft();
    asi.SerialNumber.TrimRight();
    strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
    asi.FirmwareRev = buf;
```

```
     asi.FirmwareRev.TrimLeft();
     asi.FirmwareRev.TrimRight();
     strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
     asi.Model = buf;
     asi.Model.TrimLeft();
     asi.Model.TrimRight();
     if(asi.Model.IsEmpty() || asi.FirmwareRev.IsEmpty())
     {
          return FALSE;
     }
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//   asi.Model = _T(" MTRON ") + asi.Model;
     asi.ModelSerial = asi.Model + asi.SerialNumber;
     asi.ModelSerial.Replace(_T("/"), _T(""));
     asi.Major = GetAtaMajorVersion(identify->MajorVersion, asi.MajorVersion);
     asi.TransferModeType = GetTransferMode(identify->MultiWordDma, identify-
>SerialAtaCapabilities,
                              identify->UltraDmaMode, asi.CurrentTransferMode,
asi.MaxTransferMode,
                              asi.Interface, &asi.InterfaceType);
     asi.DetectedTimeUnitType = GetTimeUnitType(asi.Model, asi.FirmwareRev,
asi.Major, asi.TransferModeType);
     // Характеристика
     if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported1 & (1 << 0))
     {
          asi.IsSmartSupported = TRUE;
     }
     if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 3))
     {
          asi.IsApmSupported = TRUE;
          if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 3))
          {
               asi.IsApmEnabled = TRUE;
          }
     }
     if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 9))
     {
          asi.IsAamSupported = TRUE;
          if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 9))
          {
               asi.IsAamEnabled = TRUE;
          }
```

```
    }
    if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 10))
    {
        asi.IsLba48Supported = TRUE;
    }
    if(asi.Major >= 6 && asi.IdentifyDevice.SerialAtaCapabilities & (1 << 8))
    {
        asi.IsNcqSupported = TRUE;
    }
    if(asi.Major >= 7 && asi.IdentifyDevice.NvCacheCapabilities & (1 << 0))
    {
        asi.IsNvCacheSupported = TRUE;
    }
    CString model = asi.Model;
    model.MakeUpper();
    if(model.Find(_T("MAXTOR")) == 0 && asi.DetectedTimeUnitType ==
POWER_ON_MINUTES)
    {
        asi.IsMaxtorMinute = TRUE;
    }
    // Розмір диску та розмір буферу
    asi.Cylinder = identify->LogicalCylinders;
    asi.Head = identify->LlogicalHeads;
    asi.Sector = identify->LogicalSectors;
    asi.Sector28 = identify->TotalAddressableSectors;
    asi.Sector48 = identify->MaxUserLba;
    asi.DiskSizeChs   = (DWORD)(((ULONGLONG)identify->LogicalCylinders *
identify->LlogicalHeads * identify->LogicalSectors * 512) / 1000 / 1000  - 50);
    asi.DiskSizeLba28 = (DWORD)(((ULONGLONG)identify->TotalAddressableSectors *
512) / 1000 / 1000 - 50);
    if(asi.IsLba48Supported)
    {
        asi.DiskSizeLba48 = (DWORD)(((ULONGLONG)identify->MaxUserLba * 512) /
1000 / 1000 - 50);
    }
    asi.BufferSize = identify->BufferSize * 512;
    if(asi.IsNvCacheSupported)
    {
        asi.NvCacheSize = identify->NvCacheSizeLogicalBlocks * 512;
    }
    if(asi.DiskSizeChs == 0)
    {
```

```
          asi.TotalDiskSize = 0;
     }
     else if(asi.DiskSizeLba48 > asi.DiskSizeLba28)
     {
          asi.TotalDiskSize = asi.DiskSizeLba48;
     }
     else if(asi.DiskSizeLba28 > asi.DiskSizeChs)
     {
          asi.TotalDiskSize = asi.DiskSizeLba28;
     }
     else
     {
          asi.TotalDiskSize = asi.DiskSizeChs;
     }
     // Перевірка помилок для контролеру External ATA
     if(asi.IsLba48Supported && (identify->TotalAddressableSectors < 268435455 &&
asi.DiskSizeLba28 != asi.DiskSizeLba48))
     {
          asi.DiskSizeLba48 = 0;
     }
     // SSD Життя
     if(asi.Model.Find(_T("MTRON")) == 0)
     {
          asi.VendorId = VENDOR_MTRON;
     }
     // перевірка підтримки S.M.A.R.T.
     switch(asi.CommandType)
     {
     case CMD_TYPE_PHYSICAL_DRIVE:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//        SendAtaCommandPd(physicalDriveId, 0xEF, 0x42, 0xFE);
//        SendAtaCommandPd(physicalDriveId, 0xEF, 0x05, 0xFE);
          if(GetSmartAttributePd(physicalDriveId, &asi))
          {
               GetSmartThresholdPd(physicalDriveId, &asi);
               asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
               asi.IsSmartEnabled = TRUE;
          }
          else if(ControlSmartStatusPd(physicalDriveId, ENABLE_SMART))
          {
               if(GetSmartAttributePd(physicalDriveId, &asi))
```

```
                {
                    GetSmartThresholdPd(physicalDriveId, &asi);
                    asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                    asi.IsSmartEnabled = TRUE;
                }
            }
            break;
```

Наступна структура визначає значення атрибутів параметрів.

```
struct IDENTIFY_DRIVE
{
    WORD    GeneralConfiguration;               //0
    WORD    LogicalCylinders;                   //1  Застарівший
    WORD    SpecificConfiguration;              //2
    WORD    LlogicalHeads;                  //3 Застарівший
    WORD    Retired1[2];                    //4-5
    WORD    LogicalSectors;                     //6 Застарівший
    DWORD       ReservedForCompactFlash;            //7-8
    WORD    Retired2;                       //9
    CHAR    SerialNumber[20];                   //10-19
    WORD    Retired3;                       //20
    WORD    BufferSize;                         //21 Застарівший
    WORD    Obsolute4;                          //22
    CHAR    FirmwareRev[8];                     //23-26
    CHAR    Model[40];                          //27-46
    WORD    MaxNumPerInterupt;              //47
    WORD    Reserved1;                          //48
    WORD    Capabilities1;                  //49
    WORD    Capabilities2;                  //50
    DWORD       Obsolute5;                          //51-52
    WORD    Field88and7064;                     //53
    WORD    Obsolute6[5];                   //54-58
    WORD    MultSectorStuff;                    //59
    DWORD       TotalAddressableSectors;            //60-61
    WORD    Obsolute7;                          //62
    WORD    MultiWordDma;                   //63
    WORD    PioMode;                        //64
    WORD    MinMultiwordDmaCycleTime;           //65
    WORD    RecommendedMultiwordDmaCycleTime;   //66
    WORD    MinPioCycleTimewoFlowCtrl;          //67
    WORD    MinPioCycleTimeWithFlowCtrl;        //68
```

```
 WORD          Reserved2[6];                    //69-74
WORD      QueueDepth;                          //75
WORD      SerialAtaCapabilities;               //76
WORD      ReservedForFutureSerialAta;          //77
WORD      SerialAtaFeaturesSupported;          //78
WORD      SerialAtaFeaturesEnabled;            //79
WORD      MajorVersion;                   //80
WORD      MinorVersion;                   //81
WORD      CommandSetSupported1;                //82
WORD      CommandSetSupported2;                //83
WORD      CommandSetSupported3;                //84
WORD      CommandSetEnabled1;                  //85
WORD      CommandSetEnabled2;                  //86
WORD      CommandSetDefault;              //87
WORD      UltraDmaMode;                   //88
WORD      TimeReqForSecurityErase;             //89
WORD      TimeReqForEnhancedSecure;            //90
WORD      CurrentPowerManagement;              //91
WORD      MasterPasswordRevision;              //92
WORD      HardwareResetResult;                 //93
WORD      AcoustricManagement;                 //94
WORD      StreamMinRequestSize;                //95
WORD      StreamingTimeDma;                    //96
WORD      StreamingAccessLatency;              //97
DWORD          StreamingPerformance;                //98-99
ULONGLONG MaxUserLba;                          //100-103
WORD      StremingTimePio;                     //104
WORD      Reserved3;                           //105
WORD      SectorSize;                          //106
WORD      InterSeekDelay;                      //107
WORD      IeeeOui;                        //108
WORD      UniqueId3;                           //109
WORD      UniqueId2;                           //110
WORD      UniqueId1;                           //111
WORD      Reserved4[4];                   //112-115
WORD      Reserved5;                           //116
DWORD          WordsPerLogicalSector;               //117-118
WORD      Reserved6[8];                   //119-126
WORD      RemovableMediaStatus;                //127
WORD      SecurityStatus;                      //128
WORD      VendorSpecific[31];                  //129-159
WORD      CfaPowerMode1;                  //160
```

```
    WORD      Reserved7[15];                    //161-175
    CHAR      CurrentMediaSerialNo[60];              //176-205
     WORD          SctCommandTransport;                    //206  254
    WORD      ReservedForCeAta1[2];                  //207-208
    WORD      AlignmentOfLogicalBlocks;              //209
    DWORD          WriteReadVerifySectorCountMode3;      //210-211
    DWORD          WriteReadVerifySectorCountMode2;      //212-213
    WORD      NvCacheCapabilities;                   //214
    DWORD          NvCacheSizeLogicalBlocks;              //215-216
    WORD      NominalMediaRotationRate;              //217
    WORD      Reserved8;                             //218
    WORD      NvCacheOptions1;                       //219
    WORD      NvCacheOptions2;                       //220
    WORD      Reserved9;                             //221
    WORD      TransportMajorVersionNumber;           //222
    WORD      TransportMinorVersionNumber;           //223
    WORD      ReservedForCeAta2[10];                 //224-233
    WORD      MinimumBlocksPerDownloadMicrocode;         //234
    WORD      MaximumBlocksPerDownloadMicrocode;         //235
    WORD      Reserved10[19];                        //236-254
    WORD      IntegrityWord;                    //255
};
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою RC6 – симетричний блоковий криптографічний алгоритм, похідний від алгоритму RC5. Був створений Роном Рівестом, Меттом Робшау і Реєм Сіднеєм для задоволення вимог конкурсу Advanced Encryption Standard (AES). Алгоритм був одним з п'яти фіналістів конкурсу, був також представлений NESSIE і CRYPTREC. Є власницьким (пропрієтарним) алгоритмом, і запатентований RSA Security, одначе дія патентів сплила, і зараз алгоритм знаходиться у відкритому доступі. В той же час, "RC6" залишається зареєстрованою торговою маркою RSA.

Варіант шифру RC6, заявлений на конкурс AES, підтримує блоки довжиною 128 біт і ключі довжиною 128, 192 і 256 біт, але сам алгоритм, як і

RC5, може бути налаштований для підтримки більш широкого діапазону довжин як блоків, так і ключів (від 0 до 2040 біт)[1]. RC6 дуже схожий на RC5 за своєю структурою і також досить простий у реалізації.

Є фіналістом AES, проте одна з примітивних операцій – операція множення, повільно виконується на певному обладнанні і ускладнює реалізацію шифру на ряді апаратних платформ і, що виявилося сюрпризом для авторів, на системах з архітектурою Intel IA-64 також реалізована досить погано. В даному випадку алгоритм втрачає одну зі своїх ключових переваг – високу швидкість виконання, що стало причиною для критики і однією з перепон для обрання як нового стандарту.

Деталі RC6

Так само, як і RC5, RC6 – повністю параметризована сім'я алгоритмів шифрування. Для специфікації алгоритму з конкретними параметрами, прийнято позначення RC6-w/r/b, де

– W – довжина машинного слова в бітах.

– R – число раундів.

– B – довжина ключа в байтах. Можливі значення 0 .. 255 байт.

Для того щоб відповідати вимогам AES, блочний шифр повинен працювати з 128-бітовими блоками. Так як RC5 – виключно швидкий блочний шифр, розширення його, щоб працювати з 128-бітовими блоками, привело б до використання двох 64-бітових робочих регістрів. Але архітектура і мови програмування ще не підтримують 64-бітні операції, тому довелося змінити проект так, щоб використовувати чотири 32-бітних регістри замість двох 64-бітних.

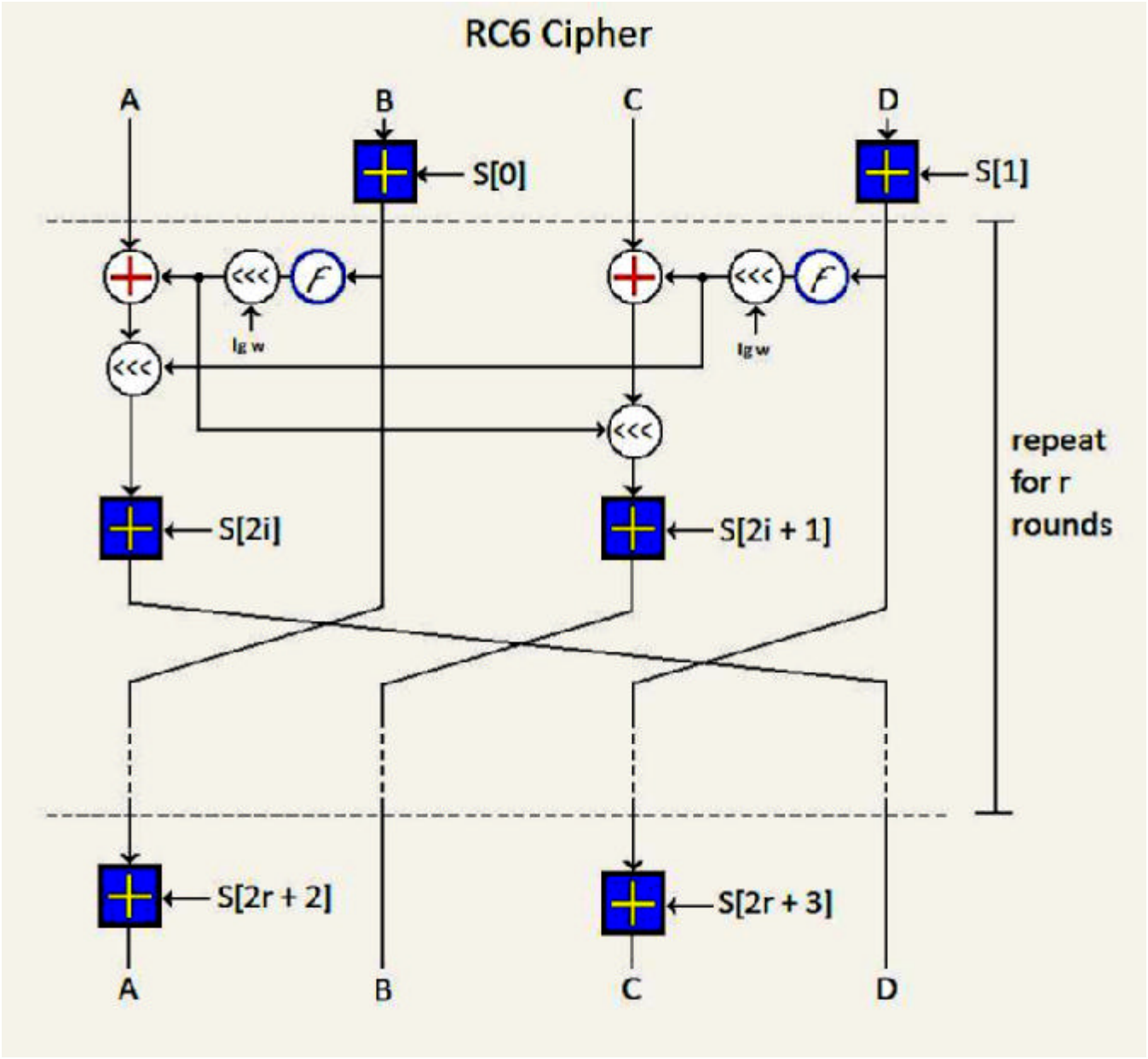

Рисунок 4.3 – Структура RC6

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Інтерфейс користувача розробленого програмного продукту зображено на рисунку 5.1. З нього видно, що програма складається з наступних функціональних блоків:

– Блок управління системою, який включає в себе наступні меню, які розвертаються: Файл, Диски, Функції, Параметри, Довідка.

– Блок загальних даних про носій інформації.

– Блок значень атрибутів та визначення помилок, якщо вони є.

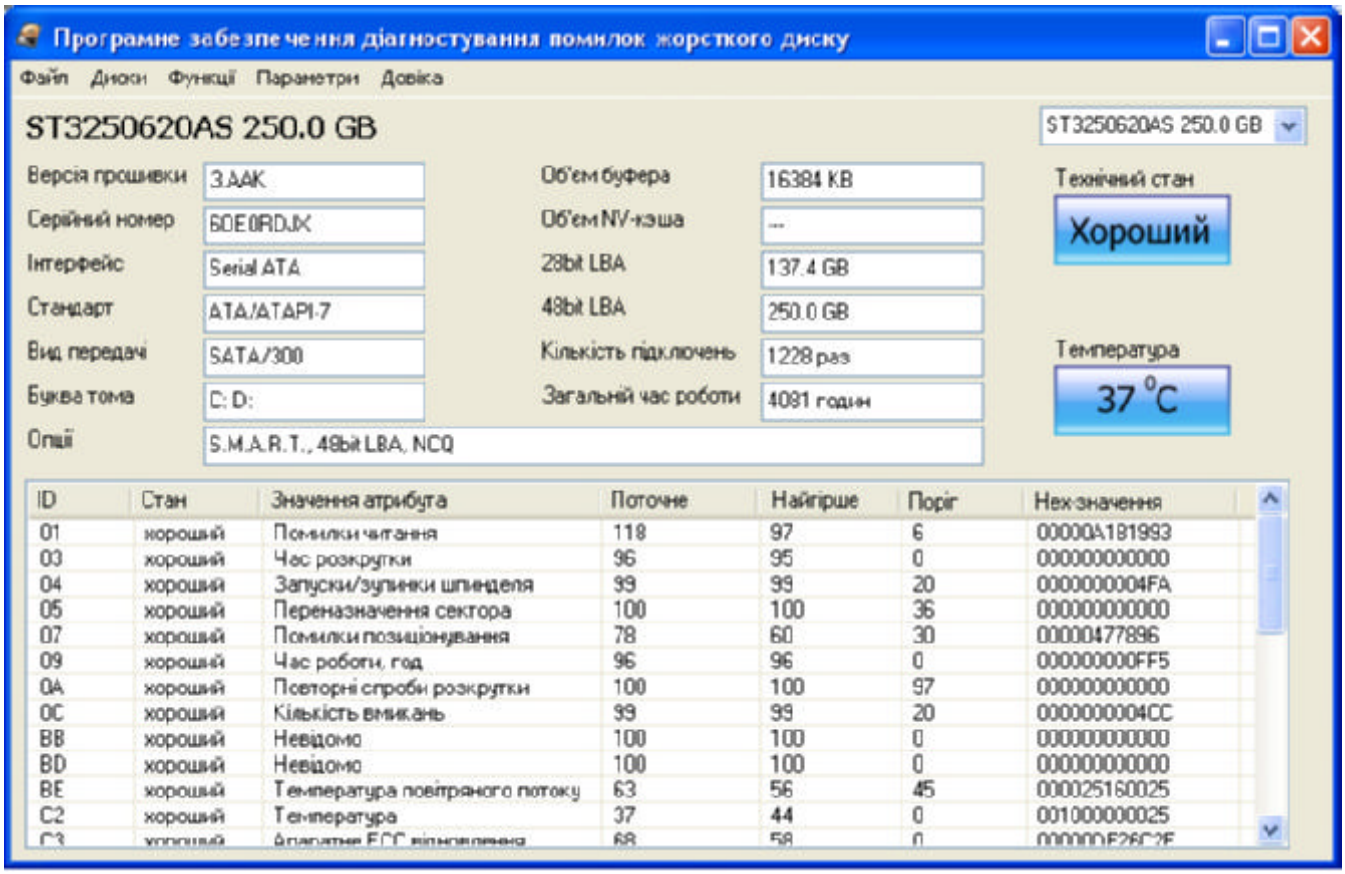

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено інформацію про розробника програмного продукту.

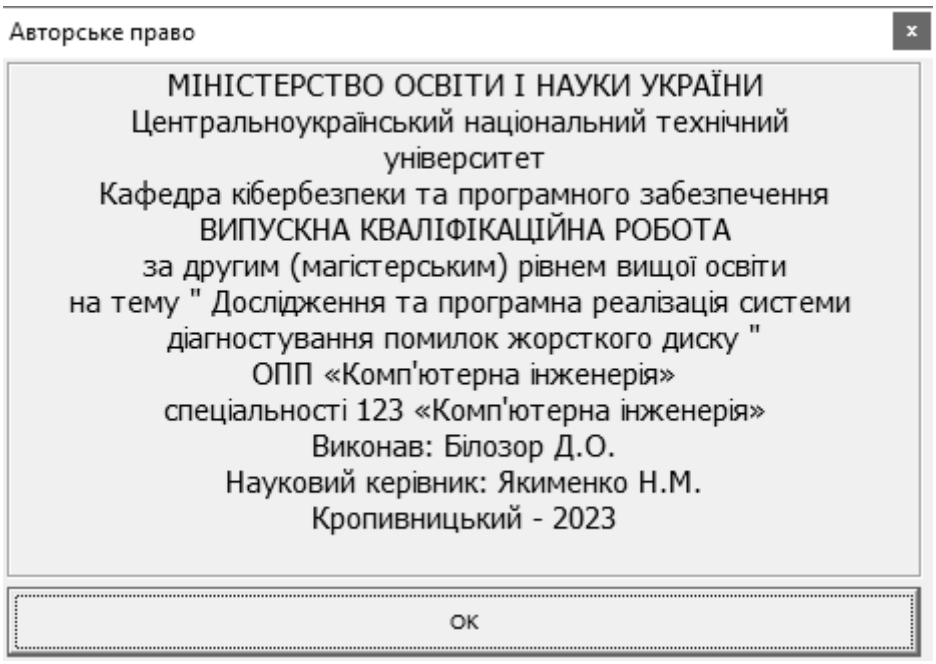

Рисунок 5.2 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи діагностування помилок жорсткого диску.

Метою розробки є дослідження та програмна реалізація системи діагностування помилок жорсткого диску.

Об'єктом дослідження є процес діагностування помилок жорсткого диску.

Предметом дослідження є методи діагностування помилок жорсткого диску.

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

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

– Удосконалено метод діагностування помилок жорсткого диску.

– Розроблено вітчизняний продукт діагностування помилок жорсткого диску, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

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

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи діагностування помилок жорсткого диску.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликі системні потреби;

б) незалежність від встановлених на комп’ютері баз даних;

в) зручність у користуванні;

г) надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	19
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	19000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, $A = 2{,}45$;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

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

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

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

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

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

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4).

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо S =86 %

$$T_{РП} = 0{,}3 \cdot 3{,}22 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 86 = 168 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

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

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	6	540	9
Монітор	60	6	360	6
Клавіатура	30	6	180	3
Маніпулятор «мишка»	30	6	180	3
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	3	360	6
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	4	120	2
Кабельні господарства ЛОМ на 1 м. п.	2,5	550	1375	22,92
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	57,92

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф^{с}_{др} = \frac{58 \cdot 3}{1,2} = 145 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 145/(60 \cdot 8) = 0{,}3 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12475	37425
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	0,3	11500	10350
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 7,35	-	$Ф_{роб}$ = 264600

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{264600}{7,35 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

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

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

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

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

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 12500 = 100000 \text{ грн.}$$

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 15.10.23 – джерело

http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Celeron G5905 (BX80701G5905) 1200, 2 ядра, 2 потоки, 3.5 GHz, TDP – 58 Вт, 14nm box	-
Системна плата	GIGABYTE H470M H 1200, DDR4 64 ГБ, 3200 MHz, LAN – 1 Гбит/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Radeon R7 350 2Gb Afox (AFR7350-2048D5H4-V3) PCI-Express 3.0, 2 ГБ, GDDR5, 128 Bit	-
Жорсткий диск	SSD 2.5" 256GB Mibrand, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2666 MHZ GOODRAM (GR2666D464L19S/8G)	-
DVD-привод	-	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA-489, PSU 350W(FSP Brand: ATX-350PNR, 12cm), black, (front bezel – black+light silver; body material – 0.6mm), 80mm fan (rear), 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	100000	25	25000
Всього по групі	202690	-	45797,5
7. Нематеріальні активи	19000	10	1900
Разом	$К_р$ = 1744575		$А_р$ = 175540

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 2635 USD, що враховуючи курс 37 складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 600 \cdot 209/19 = 6600 \text{ грн.}$$

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

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

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

$$З_д = 6600 \cdot 10 \cdot 0{,}01 = 660 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(6600+660) = 2686 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

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

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

де: $З_{М1}$ – вартість паперу, грн.;

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

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

$N_е$ – кількість екземплярів програм, шт.

Згідно норм приймаємо 0,5 пачки паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 210 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

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

$$З_{М1} = 210\ 0{,}5 = 105 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 28 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 51 грн./шт.

$$З_{М2} = 28 \cdot 2 + 51 = 107 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (105 + 107 + 1702)/19 = 101 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$O_n = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6600
2. Додаткова зарплата виконавців	$З_д$	660
3. Відрахування на соціальні потреби	$С_{оц}$	2686
4. Загальногосподарські витрати	$Г_{осп}$	990
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	990
7. Амортизація основних фондів	$А_м$	2310
8. Повна собівартість програмного забезпечення	$С_п$	14337
9. Плановий прибуток	$П_р$	7168,5
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	21505,5
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4301,1
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	25806,6

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 19 прим.):

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

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

$$A_м = 175540 \cdot 3/(19 \cdot 12) = 2310 \text{ грн.}$$

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

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

$$C_п = 6600+660+2686+990+101+990+2310 = 14337 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 14337 = 7168{,}5 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	25807
Всього капітальних витрат	–	25807

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	32208	13420
2. Витрати на електроенергію	$З_{ел}$	5320	4560
3. Витрати на амортизацію	$З_{ам}$	0	6352
Всього витрат за рік	$І$	37528	24332

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 32208 \text{ грн},$$

до:

$$З_{р\,нов} = 100 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 13420 \text{ грн}.$$

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

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

$$З_{ел\ баз} = 0{,}35 \cdot 4000 \cdot 3{,}8 = 5320 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}3 \cdot 4000 \cdot 3{,}8 = 4560 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	25807	–	6451,75
Всього відрахувань	-	–	25807	–	6451,75

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (21505{,}5 - 14337) \cdot 19 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}25 \cdot 105190 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 19000) \cdot 3/12 = 92317 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: K_p^* – балансова вартість основних фондів розробника.

$$T_в = \frac{1744575}{(21505,5 - 14337) \cdot 19 \cdot 12 / 3} = 3,2 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	19
2. Повна собівартість розробленої програми	Грн.	14337
3. Ціна розробленої програми	Грн.	21505,5
4. Плановий прибуток від реалізації розробленої програми	Грн.	7168,5
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1744575
7. Загальний прибуток від реалізації програмної продукції	Грн.	136201,5
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	92317
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	3,2
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	25807
11.Величина економічного ефекту у користувача програмної продукції	Грн.	6744
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,95

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (37528\text{-}24332)\text{-}0{,}25 \cdot 25807 = 6744 \text{ грн.}$$

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

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

$$T_{сп} = \frac{25807}{37528 - 24332} = 1{,}95 \ року.$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [5], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

До фізичних факторів при роботі на комп’ютері можемо віднести:

– підвищену й знижену температуру повітря;

– підвищену й знижену вологість повітря;

– недостатню освітленість робочого місця;

– перевищуючі припустимі норми шуму;

– підвищений рівень іонізуючого випромінювання;

– підвищений рівень електромагнітних полів;

– підвищений рівень статичної електрики;

– небезпеку враження електричним струмом;

– блякліть екрана дисплея;

до хімічних можемо віднести:.

– виникнення, у результаті іонізації повітря при роботі комп'ютера, активних часток.

До психофізіологічні можемо віднести:.

– розумова напруга;

– втрата реальності;

– виникнення залежності;

– нервово-емоційні перевантаження;

– перенапруга зорового аналізатора.

До біологічних можемо віднести:

– бактеріологічну небезпеку пов’язану з наявністю місць з сприятливим середовищем їх розмноженням (наприклад клавіатура).

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівця

Практичне значення заходів щодо підвищення працездатності випливає із закономірностей її динаміки і зводиться ось до чого:

– збільшення фази стійкого стану у фонді робочого часу;

– прискорення процесу впрацювання;

– віддалення фази розвитку втоми;

– забезпечення високої продуктивності праці за нормальних фізіологічних затрат.

Комплекс заходів щодо підвищення і збереження працездатності працівників на оптимальному рівні реалізується на техніко-організаційному, соціально-економічному, санітарно-гігієнічному, медико-біологічному, психологічному напрямках.

Вагомим фактором високої працездатності і продуктивності праці є оптимізація трудових навантажень на основі механізації і автоматизації виробничих процесів, удосконалення технології, скорочення і ліквідації важкої ручної праці. Доведено, що при правильній організації праці на легких роботах спостерігається найбільша тривалість фази стійкого стану, а на важких роботах вона нетривала.

Високий рівень працездатності безпосередньо залежить від умов праці, оскільки поліпшення їх супроводжується зменшенням енергетичних затрат організму на подолання несприятливого впливу факторів виробничого середовища.

Важливим напрямком підвищення працездатності працюючих є ритмізація трудових процесів, оптимізація темпу роботи, а також раціоналізація трудових рухів на фізіологічній основі, що сприяє формуванню і закріпленню робочих

динамічних стереотипів, а отже зменшенню м’язових і вольових зусиль. Ритмічна робота підвищує функціональні можливості організму, сприяє його тренованості і забезпечує економізацію енергетичних затрат. [1]

Багатьом програмістам постійно доводиться працювати з великою кількістю програм одночасно. Часте перемикання туди-сюди між IDE та довідкою суттєво зменшує продуктивність фахівця. Однак вирішення цієї проблеми досить просте та очевидне: встановлення більшої кількості моніторів.

Оптимальним варіантом є два монітори. Все ж таки це найпростіший з апаратної точки зору варіант. Крім того, якби їх було більше, то ними було б важче керувати, та й столі просто не вистачить місця на ще один монітор. Але тут ще залежить розміру моніторів. Є системи із 4 або 6 відносно невеликими екранами, які кріпляться на кронштейні. Але оптимальнім є два 27-дюймові монітори, на яких все добре видно, особливо коли працювати доводиться в основному з текстом [3].

8.4 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 2 м.; довжина – 3,5 м.; висота – 3 м.

Розрахунок штучного освітлення проводемо за методом коефіцієнта використання світлового потоку.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де: F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; $E = 300$ *Лк;*

S – площа освітлюваного приміщення (у нашому випадку S=2 х 3,5 = 7 $м^2$);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін}$.)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють **$\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%** *[6]***.**

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де: *S* – площа приміщення, *S = 7* м^2;

h – розрахункова висота підвісу, *h* = 3 *м;*

A – ширина приміщення, А = 2 м;

B – довжина приміщення, В = *3,5* м.

Підставимо всі значення у формулу та визначемо індекса приміщення: *i=0,57.*

Знаючи індекс приміщення, за знаходимо *n = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу).* Підставимо всі значення у формулу, визначемо світловий потік: *F=7533 Лм.*

Для штучного освітлення приміщення використовуються *LED панель MAXUS ASSISTANCE PRO 80W 5000K WHITE (M1052480531)*, світловий потік яких $F_л$ = *8000* Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де: *F* – світловий потік,

$F_л$ – світловий потік одного *світильника*.

N= 7533/ 8000=0,94 шт.

Приймаємо необхідну кількість світильників 1 шт.

8.5 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного освітлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

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

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2.007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

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

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

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

6. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи діагностування помилок жорсткого диску.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів діагностування помилок жорсткого диску.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем діагностування помилок жорсткого диску.

– Досліджена система діагностування помилок жорсткого диску.

– На основі отриманих результатів досліджень створена програмна реалізація системи діагностування помилок жорсткого диску.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання діагностування помилок жорсткого диску.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RC6.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 6744 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,95 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білозор Д.О. Дослідження та програмна реалізація системи діагностування помилок жорсткого диску // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.
3. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.
4. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.
5. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
6. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
7. 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.
8. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
9. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.
10. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.
11. 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.

12. 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.

13. 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.

14. 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.

15. 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

16. 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.

17. 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.

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

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

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

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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).

33. 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.

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

35. 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.

36. 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.

37. 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

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

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

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

52. Смірнов О.А., Коваленко О.В., Мелешко Є.В., Константинова Л.В., Кожанова А.С. Інженерія програмного забезпечення. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник"

надано у відповідності з листом Міністерства освіти і науки України від 18.03.2013 року № 1/11-5584. – Кіровоград: КНТУ 2013. – 409с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи діагностування помилок жорсткого диску.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи діагностування помилок жорсткого диску.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи діагностування помилок жорсткого диску;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті пропозиції щодо підвищення працездатності ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 107 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Якименко Н.М.

Дослідження та програмна реалізація системи діагностування помилок жорсткого диску

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 70

Літера: РП

Кропивницький – 2023 року

Main.cpp – основна програма

```
#include "stdafx.h"
#include "Main.h"
#include "MainDlg.h"
#include "GraphDlg.h"

#include "GetFileVersion.h"
#include "GetOsInfo.h"
#include "IsCurrentUserLocalAdministrator.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMainApp

BEGIN_MESSAGE_MAP(CMainApp, CWinApp)
      ON_COMMAND(ID_HELP, &CWinApp::OnHelp)
END_MESSAGE_MAP()


// CMainApp конструктор

CMainApp::CMainApp()
{
      // ПРИМІТКА: Додаємо код конструктору,
      // Встановлюємо усю значиму ініціалізацію в  InitInstance
}


// Опис CMainApp об’єкту

CMainApp theApp;
CString m_Ini;

//-----------------------------------------------------------------------------
// Опис прототипів
//-----------------------------------------------------------------------------
static BOOL IsFileExistEx(const TCHAR* path, const TCHAR* fileName);
static BOOL RunAsRestart();
// CMainApp ініціалізація

BOOL CMainApp::InitInstance()
{
      BOOL flagEarthlight = FALSE;
      DWORD defaultDisk = 0;
      HANDLE hMutex = NULL;

      INITCOMMONCONTROLSEX InitCtrls;
      InitCtrls.dwSize = sizeof(InitCtrls);
      InitCtrls.dwICC = ICC_WIN95_CLASSES;
      InitCommonControlsEx(&InitCtrls);
      CWinApp::InitInstance();

      ZeroMemory(&m_OsVer, sizeof(OSVERSIONINFO));
      m_OsVer.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
      GetVersionEx(&m_OsVer);

      // Якщо версія підтримується.
      if(GetFileVersion(_T("Shdocvw.dll")) < 471)
      {
            AfxMessageBox(_T("CrystalMain має потребу у  IE 6.0 або вище."));
      }

      // ініціалізація m_Ini
      TCHAR *ptrEnd;
```

```
	TCHAR ini[MAX_PATH];
	::GetModuleFileName(NULL, ini, MAX_PATH);
	if((ptrEnd = _tcsrchr(ini, '.')) != NULL)
	{
		*ptrEnd = '\0';
		_tcscat_s(ini, MAX_PATH, _T(".ini"));
	}
	m_Ini = ini;

	int argc;
	LPWSTR *argv = CommandLineToArgvW(GetCommandLineW(), &argc);

	if(argc > 1)
	{
		CString cstr;
		cstr = argv[1];
		if(cstr.Compare(_T("/Earthlight")) == 0)
		{
			flagEarthlight = TRUE;
			if(argc > 2)
			{
				defaultDisk = _tstoi(argv[2]);
			}
		}
		if(cstr.Compare(_T("/Startup")) == 0)
		{
			int time = 0;
			time = GetPrivateProfileInt(_T("Setting"),
_T("StartupWaitTime"), 30, m_Ini);
			if(time >= 0)
			{
				Sleep(time * 1000);
			}
			TCHAR str[MAX_PATH];
			::GetModuleFileName(NULL, str, MAX_PATH);
			ShellExecute(NULL, NULL, str, NULL, NULL, SW_SHOWNORMAL);
			return FALSE;
		}
	}

	// Розшифрування отриманих даних
	//flagEarthlight = TRUE;

	if(! flagEarthlight)
	{
		hMutex = ::CreateMutex(NULL, FALSE, PRODUCT_NAME PRODUCT_VERSION);
		if(GetLastError() == ERROR_ALREADY_EXISTS)
		{
			return FALSE;
		}
	}

	CString DefaultTheme;
	CString DefaultLanguage;
	TCHAR tmp[MAX_PATH];

	GetModuleFileName(NULL, tmp, MAX_PATH);
	if((ptrEnd = _tcsrchr(tmp, '\\')) != NULL)
	{
		*ptrEnd = '\0';
	}

	m_ExeDir.Format(_T("%s\\"), tmp);
	m_ThemeDir.Format(_T("%s\\%s"), tmp, THEME_DIR);
	m_LangDir.Format(_T("%s\\%s"), tmp, LANGUAGE_DIR);
	m_SmartDir.Format(_T("%s\\%s"), tmp, SMART_DIR);

	m_ThemeIndex = MENU_THEME_INDEX;
	m_LangIndex = MENU_LANG_INDEX;
```

```
	DefaultTheme.Format(_T("%s\\%s"), tmp, DEFAULT_THEME);
	DefaultLanguage.Format(_T("%s\\%s"), tmp, DEFAULT_LANGUAGE);

/*
	if(IsClassicSystem())
	{
		m_MainDlgPath.Format(_T("%s\\") DIALOG_DIR CLASSIC_DIALOG, tmp);
	}
*/
	if(! IsFileExist(m_MainDlgPath))
	{
		m_MainDlgPath.Format(_T("%s\\") DIALOG_DIR MAIN_DIALOG, tmp);
	}

	m_AboutDlgPath.Format(_T("%s\\") DIALOG_DIR ABOUT_DIALOG, tmp);
	m_SettingDlgPath.Format(_T("%s\\") DIALOG_DIR SETTING_DIALOG, tmp);
	if(GetIeVersion() == 800)
	{
		m_GraphDlgPath.Format(_T("%s\\") DIALOG_DIR GRAPH_DIALOG_IE8, tmp);
	}
	else
	{
		m_GraphDlgPath.Format(_T("%s\\") DIALOG_DIR GRAPH_DIALOG, tmp);
	}
	m_OptionDlgPath.Format(_T("%s\\") DIALOG_DIR OPTION_DIALOG, tmp);

	if(! IsFileExistEx(m_MainDlgPath, MAIN_DIALOG))
	{
		return FALSE;
	}
	if(! IsFileExistEx(m_AboutDlgPath, ABOUT_DIALOG))
	{
		return FALSE;
	}
	if(! IsFileExistEx(m_SettingDlgPath, SETTING_DIALOG))
	{
		return FALSE;
	}

// для сімейства Windows NT
#ifdef _UNICODE
	if(! IsCurrentUserLocalAdministrator())
	{
		if(m_OsVer.dwMajorVersion < 6)
		{
			AfxMessageBox(_T("CrystalMain is required Administrator
Privileges."));
		}
		RunAsRestart();
		return FALSE;
	}
#endif

	if(flagEarthlight)
	{
		CGraphDlg dlg(NULL, defaultDisk);
		m_pMainWnd = &dlg;
		INT_PTR nResponse = dlg.DoModal();
	}
	else
	{
		CMainDlg dlg;
		m_pMainWnd = &dlg;
		INT_PTR nResponse = dlg.DoModal();
		::ReleaseMutex(hMutex);
		::CloseHandle(hMutex);
	}
```

```
	return FALSE;
}

BOOL IsFileExistEx(const TCHAR* path, const TCHAR* fileName)
{
	if(! IsFileExist(path))
	{
		CString cstr;
		cstr.Format(_T("Не знайдено\"%s\"."), fileName);
		AfxMessageBox(cstr);
		return FALSE;
	}
	return TRUE;
}

BOOL RunAsRestart()
{
	int count;
#ifdef _UNICODE
	TCHAR** cmd = ::CommandLineToArgvW(::GetCommandLine(), &count);
#else
	TCHAR** cmd = ::__argv;
	count = ::__argc;
#endif

	if(count < 2 || _tcscmp(cmd[1], _T("runas")) != 0)
	{
		TCHAR path[MAX_PATH];
		::GetModuleFileName(NULL, path, MAX_PATH);
		if(::ShellExecute(NULL, _T("runas"), path, _T("runas"), NULL,
SW_SHOWNORMAL)
			> (HINSTANCE)32)
		{
			return TRUE;
		}
	}
	return FALSE;
}
```

Main.h – бібліотека для файлу Main.cpp

```
#pragma once

#ifndef __AFXWIN_H__
	#error "include 'stdafx.h' перед включенням цього файлу для PCH"
#endif

#include "resource.h"		// головні символи

#define THEME_DIR                       _T("CdiResource\\theme\\")
#define LANGUAGE_DIR                    _T("CdiResource\\language\\")
#define DIALOG_DIR                            _T("CdiResource\\dialog\\")
#define SMART_DIR                       _T("Smart\\")
#define SMART_INI                       _T("Smart.ini")
#define EXCHANGE_INI                    _T("Exchange.ini")

#define MENU_THEME_INDEX                3
#define MENU_LANG_INDEX                 6
#define MENU_DRIVE_INDEX                4

#define MAIN_DIALOG                           _T("Main.html")
//#define CLASSIC_DIALOG                      _T("Classic.html")
#define ABOUT_DIALOG                    _T("About.html")
#define SETTING_DIALOG                  _T("Setting.html")
#define GRAPH_DIALOG                    _T("Graph.html")
#define GRAPH_DIALOG_IE8                _T("GraphIe8.html")
#define OPTION_DIALOG                   _T("Option.html")

#define DEFAULT_THEME                   THEME_DIR _T("default\\Main.css")
#define DEFAULT_LANGUAGE                LANGUAGE_DIR _T("English.lang")


class CMainApp : public CWinApp
{
public:
	CMainApp();

	OSVERSIONINFO m_OsVer;
	BOOL m_IsNT;

	CString m_MainDlgPath;
	CString m_AboutDlgPath;
	CString m_SettingDlgPath;
	CString m_GraphDlgPath;
	CString m_OptionDlgPath;
	CString m_SmartDir;
	CString m_ExeDir;
	CString m_Ini;

	CString m_ThemeDir;
	CString m_LangDir;
	DWORD m_ThemeIndex;
	DWORD m_LangIndex;

// Аннулюється
	public:
	virtual BOOL InitInstance();

// Реалізація

	DECLARE_MESSAGE_MAP()
};

extern CMainApp theApp;
```

Testdisk.cpp – діагностування помилок жорсткого диску

```
#include "stdafx.h"
#include "Testdisk.h"
#include <wbemcli.h>

#include "DebugPrint.h"
#include "DnpService.h"

#pragma comment(lib, "wbemuuid.lib")
#define SAFE_RELEASE(p) { if(p) { (p)->Release(); (p)=NULL; } }

static const TCHAR *commandTypeString[] =
{
	_T("pd"),
	_T("sm"),
	_T("sa"),
	_T("sp"),
	_T("io"),
	_T("lo"),
	_T("jm"),
	_T("cy")
};
// Визначення ATA
CTestdisk::CTestdisk()
{
	BOOL bosVersionInfoEx;
	ZeroMemory(&m_Os, sizeof(OSVERSIONINFOEX));
	m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFOEX);
	if(!(bosVersionInfoEx = GetVersionEx((OSVERSIONINFO *)&m_Os)))
	{
		m_Os.dwOSVersionInfoSize = sizeof(OSVERSIONINFO);
		GetVersionEx((OSVERSIONINFO *)&m_Os);
	}

	m_FlagAtaPassThrough = FALSE;
	if(m_Os.dwMajorVersion >= 6 || (m_Os.dwMajorVersion == 5 &&
m_Os.dwMinorVersion == 2))
	{
		m_FlagAtaPassThrough = TRUE;
	}
	else if(m_Os.dwMajorVersion == 5 && m_Os.dwMinorVersion == 1)
	{
		CString cstr;
		cstr = m_Os.szCSDVersion;
		cstr.Replace(_T("Service Pack "), _T(""));
		if(_tstoi(cstr) >= 2)
		{
			m_FlagAtaPassThrough = TRUE;
		}
	}
}

CTestdisk::~CTestdisk()
{
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
DWORD CTestdisk::UpdateSmartInfo(DWORD i)
{
	if(vars.GetCount() == 0)
	{
		return SMART_STATUS_NO_CHANGE;
	}

	static SMART_ATTRIBUTE attribute[MAX_DISK][MAX_ATTRIBUTE] = {0};
// Читання таблиці атрибутів
```

```
	if(vars[i].IsSmartEnabled)
	{
		switch(vars[i].CommandType)
		{
		case CMD_TYPE_PHYSICAL_DRIVE:
			GetSmartAttributePd(vars[i].PhysicalDriveId, &(vars[i]));
			vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
			break;

		case CMD_TYPE_SCSI_MINIPORT:
			GetSmartAttributeScsi(vars[i].ScsiPort, vars[i].ScsiTargetId,
&(vars[i]));
			vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
			break;
		case CMD_TYPE_SAT:
		case CMD_TYPE_SUNPLUS:
		case CMD_TYPE_IO_DATA:
		case CMD_TYPE_LOGITEC:
		case CMD_TYPE_JMICRON:
		case CMD_TYPE_CYPRESS:
			GetSmartAttributeSat(vars[i].PhysicalDriveId, &(vars[i]));
			vars[i].DiskStatus = CheckDiskStatus(vars[i].Attribute,
vars[i].Threshold, vars[i].AttributeCount, vars[i].VendorId);
			break;
		default:
			return SMART_STATUS_NO_CHANGE;
		}

		return CheckSmartAttributeUpdate(i, attribute[i],
vars[i].Attribute);
	}

	return SMART_STATUS_NO_CHANGE;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::UpdateIdInfo(DWORD i)
{
	BOOL flag = FALSE;
	switch(vars[i].CommandType)
	{
	case CMD_TYPE_PHYSICAL_DRIVE:
		flag =  DoIdentifyDevicePd(vars[i].PhysicalDriveId,
&(vars[i].IdentifyDevice));
		break;
	case CMD_TYPE_SCSI_MINIPORT:
		flag =  DoIdentifyDeviceScsi(vars[i].ScsiPort, vars[i].ScsiTargetId,
&(vars[i].IdentifyDevice));
		break;
	case CMD_TYPE_SAT:
	case CMD_TYPE_SUNPLUS:
	case CMD_TYPE_IO_DATA:
	case CMD_TYPE_LOGITEC:
	case CMD_TYPE_JMICRON:
	case CMD_TYPE_CYPRESS:
		flag = DoIdentifyDeviceSat(vars[i].PhysicalDriveId,
&(vars[i].IdentifyDevice), vars[i].CommandType);
		break;
	default:
		return FALSE;
		break;
	}

	if(vars[i].Major >= 3 && vars[i].IdentifyDevice.CommandSetSupported2 & (1
<< 3))
	{
		vars[i].IsApmSupported = TRUE;
```

```
            if(vars[i].IdentifyDevice.CommandSetEnabled2 & (1 << 3))
            {
                  vars[i].IsApmEnabled = TRUE;
            }
            else
            {
                  vars[i].IsApmEnabled = FALSE;
            }
      }
      if(vars[i].Major >= 5 && vars[i].IdentifyDevice.CommandSetSupported2 & (1
<< 9))
      {
            vars[i].IsAamSupported = TRUE;
            if(vars[i].IdentifyDevice.CommandSetEnabled2 & (1 << 9))
            {
                  vars[i].IsAamEnabled = TRUE;
            }
            else
            {
                  vars[i].IsAamEnabled = FALSE;
            }
      }

      return flag;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CTestdisk::GetAamValue(DWORD i)
{
      return LOBYTE(vars[i].IdentifyDevice.AcoustricManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CTestdisk::GetApmValue(DWORD i)
{
      return LOBYTE(vars[i].IdentifyDevice.CurrentPowerManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CTestdisk::GetRecommendAamValue(DWORD i)
{
      return HIBYTE(vars[i].IdentifyDevice.AcoustricManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BYTE CTestdisk::GetRecommendApmValue(DWORD i)
{
      return HIBYTE(vars[i].IdentifyDevice.CurrentPowerManagement);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::EnableAam(DWORD i, BYTE param)
{
      return SendAtaCommand(i, 0xEF, 0x42, param);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::DisableAam(DWORD i)
{
      return SendAtaCommand(i, 0xEF, 0xC2, 0);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::EnableApm(DWORD i, BYTE param)
{
      return SendAtaCommand(i, 0xEF, 0x05, param);
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
```

```
BOOL CTestdisk::DisableApm(DWORD i)
{
	return SendAtaCommand(i, 0xEF, 0x85, 0);
}

BOOL CTestdisk::SendAtaCommand(DWORD i, BYTE main, BYTE sub, BYTE param)
{
	switch(vars[i].CommandType)
	{
	case CMD_TYPE_PHYSICAL_DRIVE:
		return SendAtaCommandPd(vars[i].PhysicalDriveId, main, sub, param,
NULL, 0);
		break;
	case CMD_TYPE_SCSI_MINIPORT:
		return SendAtaCommandScsi(vars[i].ScsiPort, vars[i].ScsiTargetId,
main, sub, param);
		break;
	case CMD_TYPE_SAT:
	case CMD_TYPE_SUNPLUS:
	case CMD_TYPE_IO_DATA:
	case CMD_TYPE_LOGITEC:
	case CMD_TYPE_JMICRON:
	case CMD_TYPE_CYPRESS:
		return SendAtaCommandSat(vars[i].PhysicalDriveId, main, sub, param,
vars[i].CommandType);
		break;
	default:
		return FALSE;
		break;
	}
	return FALSE;
}

//Перевірка атрибутів SMART

DWORD CTestdisk::CheckSmartAttributeUpdate(DWORD index, SMART_ATTRIBUTE* pre,
SMART_ATTRIBUTE* cur)
{

	if(memcmp(pre, cur, sizeof(SMART_ATTRIBUTE) * MAX_ATTRIBUTE) == 0)
	{
		return SMART_STATUS_NO_CHANGE;
	}
	else
	{
		for(int i = 0; i < MAX_ATTRIBUTE; i++)
		{
			switch(cur[i].Id)
			{
			case 0x09: // Кількість відпрацьованих годин у включеному
стані
				{
					DWORD preRawValue = MAKELONG(
						MAKEWORD(pre[i].RawValue[0],
pre[i].RawValue[1]),
						MAKEWORD(pre[i].RawValue[2],
pre[i].RawValue[3])
						);
					DWORD curRawValue = MAKELONG(
						MAKEWORD(cur[i].RawValue[0],
cur[i].RawValue[1]),
						MAKEWORD(cur[i].RawValue[2],
cur[i].RawValue[3])
						);

					if(GetPowerOnHours(preRawValue,
vars[index].DetectedTimeUnitType)
					!= GetPowerOnHours(curRawValue,
vars[index].DetectedTimeUnitType))
```

```
                                {
                                      memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                                      return SMART_STATUS_MAJOR_CHANGE;
                                }
                                if(GetPowerOnHours(preRawValue,
vars[index].MeasuredTimeUnitType)
                                != GetPowerOnHours(curRawValue,
vars[index].MeasuredTimeUnitType))
                                {
                                      memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                                      return SMART_STATUS_MAJOR_CHANGE;
                                }
                          }
                          break;
                    case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                          {
                                DWORD preRawValue = MAKELONG(
                                      MAKEWORD(pre[i].RawValue[0],
pre[i].RawValue[1]),
                                      MAKEWORD(pre[i].RawValue[2],
pre[i].RawValue[3])
                                      );
                                DWORD curRawValue = MAKELONG(
                                      MAKEWORD(cur[i].RawValue[0],
cur[i].RawValue[1]),
                                      MAKEWORD(cur[i].RawValue[2],
cur[i].RawValue[3])
                                      );
                                if(preRawValue != curRawValue)
                                {
                                      memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                                      return SMART_STATUS_MAJOR_CHANGE;
                                }
                          }
                          break;
                    case 0xC2: // Температура
                          if(pre[i].RawValue[0] != cur[i].RawValue[0]
                          || pre[i].CurrentValue != cur[i].CurrentValue)
                          {
                                memcpy(pre, cur, sizeof(SMART_ATTRIBUTE) *
MAX_ATTRIBUTE);
                                return SMART_STATUS_MAJOR_CHANGE;
                          }
                          break;
                    default:
                          break;
                    }
              }
              return SMART_STATUS_MINOR_CHANGE;
        }
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::MeasuredTimeUnit()
{
      DWORD getTickCount = GetTickCount();
      if(getTickCount > MeasuredGetTickCount + 155000 || MeasuredGetTickCount +
125000 > getTickCount)
      {
            return FALSE;
      }

      for(int i = 0; i < vars.GetCount(); i++)
      {
            if(vars[i].PowerOnRawValue < 0)
```

```
            {
                  continue;
            }
            UpdateSmartInfo(i);

            DWORD test = vars[i].PowerOnRawValue - vars[i].PowerOnStartRawValue;

            if(vars[i].Model.Find(_T("SAMSUNG")) == 0)
            {
                  if(test >= 2)
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_HALF_MINUTES;
                  }
                  else
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                  }
            }
            else if(vars[i].Model.Find(_T("FUJITSU")) == 0)
            {
                  if(test >= 2)
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_SECONDS;
                  }
                  else
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                  }
            }
            else if(vars[i].Model.Find(_T("MAXTOR")) == 0)
            {
                  if(test >= 2)
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_MINUTES;
                        vars[i].IsMaxtorMinute = TRUE;
                  }
                  else
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                        vars[i].IsMaxtorMinute = FALSE;
                  }
            }
            else
            {
                  if(test >= 2)
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_MINUTES;
                  }
                  else
                  {
                        vars[i].MeasuredTimeUnitType = POWER_ON_HOURS;
                  }
            }
      }

      return TRUE;
}

/* ЗАГАЛЬНІ ФУНКЦІЇ*/
BOOL CTestdisk::Init(BOOL useWmi, BOOL advancedDiskSearch, PBOOL flagChangeDisk)
{
      IsEnabledWmi = FALSE;
      CArray<DISK_POSITION, DISK_POSITION> previous;

      if(flagChangeDisk != NULL)
      {
            *flagChangeDisk = FALSE;
            for(int i = 0; i < vars.GetCount(); i++)
            {
```

```
                DISK_POSITION dp;
                dp.PhysicalDriveId = vars.GetAt(i).PhysicalDriveId;
                dp.ScsiTargetId = vars.GetAt(i).ScsiTargetId;
                dp.ScsiPort = vars.GetAt(i).ScsiPort;

                previous.Add(dp);
          }
     }

     // Ініціалізація
     vars.RemoveAll();
     m_ControllerMap = _T("");

     if(useWmi)
     {
          HRESULT hRes = S_OK;
          ULONG   uReturned = 1;

          IWbemLocator*                pIWbemLocator = NULL;
          IWbemServices*               pIWbemServices = NULL;
          IEnumWbemClassObject*   pEnumCOMDevs = NULL;
          IEnumWbemClassObject*   pEnumCOMDevs2 = NULL;
          IWbemClassObject*       pCOMDev = NULL;

          DebugPrint(_T("CTestdisk::Init WMI on - Start"));

          bool initWmi = true;
          CDnpService cService;

          for(int i = 0; i < 3; i++)
          {
                if(! cService.IsServiceRunning(_T("Winmgmt")))
                {
                     DebugPrint(_T("Зачекайте... Winmgmt"));
                     initWmi = cService.EasyStart(_T("Winmgmt"));
                     continue;
                }
                else
                {
                     break;
                }
          }

          if(initWmi)
          {
                try
                {
                     DebugPrint(_T("CoInitialize()"));
                     CoInitialize(NULL);
                     DebugPrint(_T("CoInitializeSecurity()"));
                     CoInitializeSecurity(NULL, -1, NULL, NULL,
RPC_C_AUTHN_LEVEL_DEFAULT,
                          RPC_C_IMP_LEVEL_IMPERSONATE, NULL, EOAC_NONE,
NULL);
                     DebugPrint(_T("CoCreateInstance()"));
                     if(FAILED(CoCreateInstance(CLSID_WbemLocator, NULL,
CLSCTX_INPROC_SERVER,
                          IID_IWbemLocator, (LPVOID *)&pIWbemLocator)))
                     {
                          CoUninitialize();
                          DebugPrint(_T("NG:WMI Init 1"));
                     }
                     else
                     {
                          long securityFlag = 0;
                          if( m_Os.dwMajorVersion >= 6 // Vista або пізніша
версія
                          || (m_Os.dwMajorVersion == 5 &&
m_Os.dwMinorVersion >= 1) // XP або пізніша версія
```

```
                              )
                              {
                                    securityFlag =
WBEM_FLAG_CONNECT_USE_MAX_WAIT;
                              }

                              DebugPrint(_T("ConnectServer()"));
                              if(FAILED(pIWbemLocator-
>ConnectServer(SysAllocString(L"\\\\.\\root\\cimv2"),
                                    NULL, NULL, 0L,
                                    securityFlag,
                                    NULL, NULL, &pIWbemServices)))
                              {
                                    CoUninitialize();
                                    DebugPrint(_T("NG:WMI Init 2"));
                              }
                              else
                              {
                                    DebugPrint(_T("CoSetProxyBlanket()"));
                                    hRes = CoSetProxyBlanket(pIWbemServices,
RPC_C_AUTHN_WINNT, RPC_C_AUTHZ_NONE,
                                          NULL, RPC_C_AUTHN_LEVEL_CALL,
RPC_C_IMP_LEVEL_IMPERSONATE, NULL, EOAC_NONE);
                                    if(FAILED(hRes))
                                    {
                                          CoUninitialize();
                                          CString cstr;
                                          cstr.Format(_T("NG:WMI Init - %08X"),
hRes);
                                          DebugPrint(cstr);
                                    }
                                    else
                                    {
                                          IsEnabledWmi = TRUE;
                                          DebugPrint(_T("OK:WMI Init"));
                                    }
                              }
                              SAFE_RELEASE(pIWbemLocator);
                        }
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:WMI Init"));
                  }
            }
            else
            {
                  DebugPrint(_T("NG:WMI Init 3"));
            }

            if(IsEnabledWmi)
            {
                  CStringArray csa;
                  CString temp, cstr, cstr1, cstr2;
                  try
                  {// Win32_IDE Контролер
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_IDEController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString name1, deviceId, channel;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
```

```
                                    deviceId = pVal.bstrVal;
                                    if(deviceId.Find(_T("PCIIDE\\IDECHANNEL"))
== 0)
                                    {
                                          channel = deviceId.Right(1);
                                    }
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Name", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    name1 = pVal.bstrVal;
                                    if(! channel.IsEmpty())
                                    {
                                          name1 += _T(" (") + channel + _T(")");
                                    }
                                    m_IdeController.Add(name1);

                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              if(cstr.Find(name1) == -1 || cstr.Find(name1 +
_T(" [ATA]")) >= 0)
                              {
                                    csa.Add(cstr);
                                    cstr = _T("%%%") + name1 + _T(" [ATA]");
                                    cstr += _T("\r\n");
                              }

                              CString mapping;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_IDEController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_IDEControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    CString name2, deviceId, channel;
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;

      if(deviceId.Find(_T("PCIIDE\\IDECHANNEL")) == 0)
                                          {
                                                channel = deviceId.Right(1);
                                          }
                                          VariantClear(&pVal);
                                    }
                                    if(pCOMDev->Get(L"Name", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          name2 = pVal.bstrVal;
                                          if(! channel.IsEmpty())
                                          {
                                                name2 += _T(" (") + channel +
_T(")");
                                          }
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);
```

```
                                    if(cstr.Find(_T("   - ") + name1) >= 0 ||
cstr.Find(_T("   + ") + name1) >= 0)
                                    {
                                          cstr1 = _T("- ") + name1;
                                          cstr2 = _T("+ ") + name1;
                                          cstr.Replace(cstr1, cstr2);

                                          cstr1 = name1 + _T("\r\n     - ") +
name2;
                                          cstr.Replace(name1, cstr1);
                                    }
                                    else
                                    {
                                          cstr += _T("   - ") + name2 +
_T("\r\n");
                                    }
                                    cstr.Replace(_T("%%%"), _T(" + "));
                              }
                              cstr.Replace(_T("%%%"), _T(" - "));
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        csa.Add(cstr);
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_IDEController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_IDEController"));
                  }

                  try
                  {
                        cstr = _T("");
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_SCSIController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString name1, deviceId;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Name", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    name1 = pVal.bstrVal;
                                    m_ScsiController.Add(name1);
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);
                              if(cstr.Find(name1) == -1 || cstr.Find(name1 +
_T(" [SCSI]")) >= 0)
                              {
                                    csa.Add(cstr);
                                    cstr = _T("%%%") + name1 + _T(" [SCSI]");

                                    cstr += _T("\r\n");
                              }

                              CString mapping;
```

```
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_SCSIController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_SCSIControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    CString name2, deviceId;
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    if(pCOMDev->Get(L"Name", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          name2 = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);

                                    if(cstr.Find(_T("   - ") + name1) >= 0 ||
cstr.Find(_T("   + ") + name1) >= 0)
                                    {
                                          cstr1 = _T("- ") + name1;
                                          cstr2 = _T("+ ") + name1;
                                          cstr.Replace(cstr1, cstr2);

                                          cstr1 = name1 + _T("\r\n     - ") +
name2;
                                          cstr.Replace(name1, cstr1);
                                    }
                                    else
                                    {
                                          cstr += _T("   - ") + name2 +
_T("\r\n");
                                    }
                                    cstr.Replace(_T("%%%"), _T(" + "));
                              }
                              cstr.Replace(_T("%%%"), _T(" - "));
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        csa.Add(cstr);
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_SCSIController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_SCSIController"));
                  }

                  for(int i = 0; i < csa.GetCount(); i++)
                  {
                        m_ControllerMap += csa.GetAt(i);
                  }

                  try
                  {// Win32_USB Контролер
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_USBController"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
```

```
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString deviceId, channel;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              CString mapping, enclosure;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_USBController.DeviceID=\"%s\"} WHERE AssocClass =
Win32_USBControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          cstr = pVal.bstrVal;
                                          VariantClear(&pVal);
                                          if(cstr.Find(_T("USBSTOR")) >= 0)
                                          {
                                                EXTERNAL_DISK_INFO edi = {0};
                                                int curPos= 0;
                                                CString resToken;
                                                resToken =
deviceId.Tokenize(_T("\\&"), curPos);
                                                while(resToken != _T(""))
                                                {

      if(resToken.Replace(_T("VID_"), _T("")) > 0)
                                                      {
                                                            edi.VendorId =
_tcstol(resToken, NULL, 16);
                                                      }
                                                      else
if(resToken.Replace(_T("PID_"), _T("")) > 0)
                                                      {
                                                            edi.ProductId =
_tcstol(resToken, NULL, 16);
                                                      }
                                                      resToken =
deviceId.Tokenize(_T("\\&"), curPos);
                                                };

                                                if(pCOMDev->Get(L"Name", 0L,
&pVal, NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                                {
                                                      edi.Enclosure =
pVal.bstrVal;
                                                      VariantClear(&pVal);
                                                }
                                                externals.Add(edi);
                                          }
                                          deviceId = cstr;
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
```

```
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_USBController"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_USBController"));
                  }

                  try
                  {// Win32_1394 Контролер
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"select Name, DeviceID from
Win32_1394Controller"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              VARIANT  pVal;
                              VariantClear(&pVal);
                              CString deviceId, channel;
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              CString mapping, enclosure;
                              mapping.Format(_T("ASSOCIATORS OF
{Win32_1394Controller.DeviceID=\"%s\"} WHERE AssocClass =
Win32_1394ControllerDevice"), deviceId);
                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
SysAllocString(mapping), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs2);
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VARIANT pVal;
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          deviceId = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:Win32_1394Controller"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Win32_1394Controller"));
                  }

                  /* РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                  for(int i = 0; i < externals.GetCount(); i++)
                  {
                        CString cstr;
                        cstr.Format(_T("Enclosure=%s, VID=%04X, PID=%04X"),
                              externals.GetAt(i).Enclosure,
                              externals.GetAt(i).VendorId,
                              externals.GetAt(i).ProductId);
```

```
                        AfxMessageBox(cstr);
                  }
                  */

                  try
                  {
                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"SELECT * FROM Win32_DiskDrive"),
WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY, NULL, &pEnumCOMDevs);
                        DebugPrint(_T("DO:SELECT * FROM Win32_DiskDrive"));
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              CString mapping1, mapping2;
                              CString model, deviceId, diskSize;
                              INT physicalDriveId = -1, scsiPort = -1,
scsiTargetId = -1;

                              VARIANT pVal;
                              VariantClear(&pVal);
                              if(pCOMDev->Get(L"Size", 0L, &pVal, NULL, NULL) ==
WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    diskSize = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    deviceId = pVal.bstrVal;
                                    deviceId.Replace(_T("\\"), _T("\\\\"));
                                    if(_ttoi(deviceId.Right(2)) >= 10)
                                    {
                                          physicalDriveId =
_ttoi(deviceId.Right(2));
                                    }
                                    else
                                    {
                                          physicalDriveId =
_ttoi(deviceId.Right(1));
                                    }
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"Model", 0L, &pVal, NULL, NULL)
== WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    model = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"SCSIPort", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    scsiPort = pVal.intVal;
                                    VariantClear(&pVal);
                              }
                              if(pCOMDev->Get(L"SCSITargetId", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    scsiTargetId = pVal.intVal;
                                    VariantClear(&pVal);
                              }
                              SAFE_RELEASE(pCOMDev);

                              try
                              {
                                    // GetMain - інформація про диск
                                    CString cstr;
                                    cstr.Format(_T("DO:GetMain pd=%d, sp=%d,
st=%d"), physicalDriveId, scsiPort, scsiTargetId);
```

```
                                    DebugPrint(cstr);

                                    INTERFACE_TYPE interfaceType =
INTERFACE_TYPE_UNKNOWN;
                                    VENDOR_ID vendorId = VENDOR_UNKNOWN;

                                    if(model.Find(_T(" USB Пристрій")) >= 0)
                                    {
                                          interfaceType = INTERFACE_TYPE_USB;
                                    }
                                    else if(model.Find(_T(" IEEE 1394 SBP2
Пристрій")) >= 0)
                                    {
                                          interfaceType =
INTERFACE_TYPE_IEEE1394;
                                    }

                                    for(int i = 0; i < externals.GetCount();
i++)
                                    {

      if(model.Find(externals.GetAt(i).Enclosure) == 0)
                                          {
                                                vendorId =
(VENDOR_ID)externals.GetAt(i).VendorId;
                                          }
                                    }

                                    if(GetMain(physicalDriveId, scsiPort,
scsiTargetId, interfaceType, vendorId))
                                    {
                                          int index = (int)vars.GetCount() - 1;
                                          if(! diskSize.IsEmpty())
                                          {
                                                DWORD totalDiskSize =
(DWORD)(_ttoi64(diskSize) / 1000 / 1000 - 50);
                                                if((totalDiskSize <
vars[index].TotalDiskSize - 100
                                                || vars[index].TotalDiskSize +
100 < totalDiskSize)
                                                && totalDiskSize >
vars[index].DiskSizeLba48
                                                )
                                                {
                                                      vars[index].TotalDiskSize =
totalDiskSize;
                                                }
                                          }

                                          BOOL flagSkipModelCheck = FALSE;

                                          vars[index].ModelWmi = model;
                                          // Модель
                                          model.Replace(_T(" SCSI Disk
Пристрій"), _T(""));
                                          model.Replace(_T(" ATA Пристрій"),
_T(""));

                                          if(model.Replace(_T(" USB Пристрій"),
_T("")) > 0)
                                          {
                                                flagSkipModelCheck = TRUE;
                                                cstr.Format(_T("USB (%s)"),
vars[index].Interface);
                                                vars[index].Interface = cstr;
                                                vars[index].InterfaceType =
INTERFACE_TYPE_USB;
```

```
                                                for(int i = 0; i <
externals.GetCount(); i++)
                                                {

      if(externals.GetAt(i).Enclosure.Find(vars[index].ModelWmi) == 0)
                                                      {
                                                            vars[index].Enclosure
= externals.GetAt(i).Enclosure;
                                                            vars[index].VendorId
= externals.GetAt(i).VendorId;
                                                            vars[index].ProductId
= externals.GetAt(i).ProductId;
                                                      }
                                                }
                                          }
                                          else if(model.Replace(_T(" IEEE 1394
SBP2 Пристрій"), _T("")) > 0)
                                          {
                                                flagSkipModelCheck = TRUE;
                                                cstr.Format(_T("IEEE 1394 (%s)"),
vars[index].Interface);
                                                vars[index].Interface = cstr;
                                                vars[index].InterfaceType =
INTERFACE_TYPE_IEEE1394;
                                                for(int i = 0; i <
externals.GetCount(); i++)
                                                {

      if(externals.GetAt(i).Enclosure.Find(vars[index].ModelWmi) == 0)
                                                      {
                                                            vars[index].Enclosure
= externals.GetAt(i).Enclosure;
                                                            vars[index].VendorId
= externals.GetAt(i).VendorId;
                                                            vars[index].ProductId
= externals.GetAt(i).ProductId;
                                                      }
                                                }
                                          }

                                          CString cmp, cmp1, cmp2, cmp3;
                                          cmp = model;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp1 = cmp.Left(16);

                                          cmp = vars[index].Model;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp2 = cmp.Left(16);

                                          cmp = vars[index].ModelReverse;
                                          cmp.Replace(_T(" "), _T(""));
                                          cmp3 = cmp.Left(16);

                                          if(vars[index].Model.IsEmpty())
                                          {
                                                vars.RemoveAt(index);
                                          }
                                          else if(flagSkipModelCheck)
                                          {
                                                // Не використовується
                                          }
                                          else if(model.IsEmpty() ||
cmp1.Compare(cmp2) == 0)
                                          {
                                                // Не використовується
                                          }
                                          else if(cmp1.Compare(cmp3) == 0)
                                          {
```

```
                                                vars[index].SerialNumber =
vars[index].SerialNumberReverse;
                                                vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
                                                vars[index].Model =
vars[index].ModelReverse;
                                                vars[index].ModelSerial =
vars[index].Model + vars[index].SerialNumber;

      vars[index].ModelSerial.Replace(_T("/"), _T(""));
                                          }
                                          else
                                          {
                                                vars.RemoveAt(index);
                                          }

                                          // РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                                          // vars[index].VendorId =
VENDOR_MTRON;


                                                DebugPrint(_T("OK:Check Model Name"));
                                    }
                              }
                              catch(...)
                              {
                                    DebugPrint(_T("EX:GetMain pd=%d, sp=%d,
st=%d"));
                              }
                        }
                        SAFE_RELEASE(pEnumCOMDevs);
                        DebugPrint(_T("OK:SELECT * FROM Win32_DiskDrive"));
                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:SELECT * FROM Win32_DiskDrive"));
                  }

                  // Список дисків
                  try
                  {
                        DWORD driveLetterMap[256] = {0};

                        pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                              SysAllocString(L"SELECT * FROM
Win32_DiskPartition"), WBEM_FLAG_FORWARD_ONLY | WBEM_FLAG_RETURN_IMMEDIATELY,
NULL, &pEnumCOMDevs);
                        while(pEnumCOMDevs && SUCCEEDED(pEnumCOMDevs-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                        {
                              DWORD physicalDriveId = 0;
                              CString partition, drive, mapping, cstr;
                              VARIANT pVal;
                              VariantClear(&pVal);
                              if(pCOMDev->Get(L"DeviceID", 0L, &pVal, NULL,
NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                              {
                                    partition = pVal.bstrVal;
                                    VariantClear(&pVal);
                              }
                              cstr = partition;
                              cstr.Replace(_T("Disk #"), _T(""));
                              physicalDriveId = _ttoi(cstr);
                              SAFE_RELEASE(pCOMDev);

                              pIWbemServices->ExecQuery(SysAllocString(L"WQL"),
                                    SysAllocString(L"SELECT * FROM
Win32_LogicalDisk Where DriveType = 3"), WBEM_FLAG_FORWARD_ONLY |
WBEM_FLAG_RETURN_IMMEDIATELY, NULL, &pEnumCOMDevs2);
```

```
                              while(pEnumCOMDevs2 && SUCCEEDED(pEnumCOMDevs2-
>Next(10000, 1, &pCOMDev, &uReturned)) && uReturned == 1)
                              {
                                    VariantClear(&pVal);
                                    if(pCOMDev->Get(L"DeviceID", 0L, &pVal,
NULL, NULL) == WBEM_S_NO_ERROR && pVal.vt > VT_NULL)
                                    {
                                          drive = pVal.bstrVal;
                                          VariantClear(&pVal);
                                    }
                                    SAFE_RELEASE(pCOMDev);

                                    IWbemContext *pCtx = 0;
                                    IWbemCallResult *pResult = 0;

      mapping.Format(_T("Win32_LogicalDiskToPartition.Antecedent=\"Win32_DiskPar
tition.DeviceID=\\\"%s\\\"\",Dependent=\"Win32_LogicalDisk.DeviceID=\\\"%s\\\"\"
"),
                                          partition, drive);

                                    BSTR bstr;
                                    bstr = mapping.AllocSysString();
                                    pIWbemServices->GetObject(bstr, 0, pCtx,
&pCOMDev, &pResult);
                                    SysFreeString(bstr);
                                    if(pCOMDev)
                                    {
                                          driveLetterMap[physicalDriveId] |= 1
<< (drive.GetAt(0) - 'A');
                                    }
                                    SAFE_RELEASE(pCOMDev);
                              }
                              SAFE_RELEASE(pEnumCOMDevs2);
                        }
                        SAFE_RELEASE(pEnumCOMDevs);

                        for(int i = 0; i < vars.GetCount(); i++)
                        {
                              CString driveLetter = _T("");
                              for(int j = 0; j < 26; j++)
                              {
                                    if(driveLetterMap[vars[i].PhysicalDriveId] &
(1 << j))
                                    {
                                          CString cstr;
                                          cstr.Format(_T("%C"), j + 'A');
                                          driveLetter += cstr + _T(": ");
                                          vars[i].DriveLetterMap += (1 << j);
                                    }
                              }
                              vars[i].DriveMap.Append(driveLetter);
                        }
                        DebugPrint(_T("OK:Список дисків"));

                  }
                  catch(...)
                  {
                        DebugPrint(_T("EX:Список дисків"));
                  }

                  SAFE_RELEASE(pCOMDev);
                  SAFE_RELEASE(pEnumCOMDevs);
                  SAFE_RELEASE(pEnumCOMDevs2);
                  SAFE_RELEASE(pIWbemServices);
                  CoUninitialize();

                  DebugPrint(_T("OK:CoUninitialize()"));
            }
      }
```

```
	else
	{
		DebugPrint(_T("CTestdisk::Init WMI off - Start"));
	}

	// \\\\.\\PhysicalDrive%d
	for(int i = 0; i < MAX_SEARCH_PHYSICAL_DRIVE; i++)
	{
		BOOL  bRet;
		HANDLE	hIoCtrl;
		DWORD dwReturned;
		DISK_GEOMETRY dg;

		hIoCtrl = GetIoCtrlHandle(i);
		if(hIoCtrl == INVALID_HANDLE_VALUE)
		{
			continue;
		}
		bRet = ::DeviceIoControl(hIoCtrl, IOCTL_DISK_GET_DRIVE_GEOMETRY,
			NULL, 0, &dg, sizeof(DISK_GEOMETRY),
			&dwReturned, NULL);
		::CloseHandle(hIoCtrl);
		if(bRet == FALSE || dwReturned != sizeof(DISK_GEOMETRY) ||
dg.MediaType != FixedMedia)
		{
			continue;
		}
// Визначення виробника
		if(GetMain(i, -1, -1, INTERFACE_TYPE_UNKNOWN, VENDOR_UNKNOWN))
		{
			int index = (int)vars.GetCount() - 1;
			CString cmp;
			cmp = vars[index].Model;
			if(cmp.Find(_T("DW C")) == 0 // WDC
			|| cmp.Find(_T("iHat")) == 0 // Hitachi
			|| cmp.Find(_T("ASSM")) == 0 // SAMSUNG
			|| cmp.Find(_T("aMtx")) == 0 // Maxtor
			|| cmp.Find(_T("OTHS")) == 0 // TOSHIBA
			|| cmp.Find(_T("UFIJ")) == 0 // FUJITSU
			)
			{
				vars[index].SerialNumber =
vars[index].SerialNumberReverse;
				vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
				vars[index].Model = vars[index].ModelReverse;
				vars[index].ModelSerial = vars[index].Model +
vars[index].SerialNumber;
				vars[index].ModelSerial.Replace(_T("/"), _T(""));
			}
		}
	}

	// сортування
	ATA_SMART_INFO* p = vars.GetData();
	qsort(p, vars.GetCount(), sizeof(ATA_SMART_INFO), Compare);

	DebugPrint(_T("OK:GetMain - PhysicalDrive"));

	// Визначення типу знайдених дисків
	if(advancedDiskSearch)
	{
		// \\\\.\\Scsi%d:
		for(int i = 0; i < MAX_SEARCH_SCSI_PORT; i++)
		{
			for(int j = 0; j < MAX_SEARCH_SCSI_TARGET_ID; j++)
			{
				if(GetMain(-1, i, j, INTERFACE_TYPE_UNKNOWN,
VENDOR_UNKNOWN))
```

```
                        {
                              int index = (int)vars.GetCount() - 1;
                              CString cmp;
                              cmp = vars[index].Model;
                              if(cmp.Find(_T("DW C")) == 0 // WDC
                              || cmp.Find(_T("iHat")) == 0 // Hitachi
                              || cmp.Find(_T("ASSM")) == 0 // SAMSUNG
                              || cmp.Find(_T("aMtx")) == 0 // Maxtor
                              || cmp.Find(_T("OTHS")) == 0 // TOSHIBA
                              || cmp.Find(_T("UFIJ")) == 0 // FUJITSU
                              )
                              {
                                    vars[index].SerialNumber =
vars[index].SerialNumberReverse;
                                    vars[index].FirmwareRev =
vars[index].FirmwareRevReverse;
                                    vars[index].Model =
vars[index].ModelReverse;
                                    vars[index].ModelSerial = vars[index].Model
+ vars[index].SerialNumber;
                                    vars[index].ModelSerial.Replace(_T("/"),
_T(""));
                              }
                        }
                  }
            }
            DebugPrint(_T("OK:GetMain - Scsi"));
      }

      MeasuredGetTickCount = GetTickCount();
      DebugPrint(_T("CTestdisk::Init - Complete"));

      if(flagChangeDisk != NULL)
      {
            if(vars.GetCount() != previous.GetCount())
            {
                  *flagChangeDisk = TRUE;
            }
            else
            {
                  for(int i = 0; i < vars.GetCount(); i++)
                  {
                        if(vars.GetAt(i).PhysicalDriveId !=
previous.GetAt(i).PhysicalDriveId
                        || vars.GetAt(i).ScsiTargetId !=
previous.GetAt(i).ScsiTargetId
                        || vars.GetAt(i).ScsiPort != previous.GetAt(i).ScsiPort
                        )
                        {
                              *flagChangeDisk = TRUE;
                              break;
                        }
                  }
            }
      }

      return IsEnabledWmi;
}

int CTestdisk::Compare(const void *p1, const void *p2)
{
      return ((ATA_SMART_INFO*)p1)->PhysicalDriveId - ((ATA_SMART_INFO*)p2)-
>PhysicalDriveId;
}

//Додавання нового диску, та робота з ним

BOOL CTestdisk::AddDisk(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
COMMAND_TYPE commandType, IDENTIFY_DRIVE* identify)
```

```
{
	ATA_SMART_INFO asi;

	memcpy(&(asi.IdentifyDevice), identify, sizeof(IDENTIFY_DRIVE));
	asi.PhysicalDriveId = physicalDriveId;
	asi.ScsiPort =  scsiPort;
	asi.ScsiTargetId = scsiTargetId;
	asi.CommandType = commandType;
	asi.CommandTypeString = commandTypeString[commandType];

	for(int i = 0; i < MAX_ATTRIBUTE; i++)
	{
		::ZeroMemory(&(asi.Attribute[i]), sizeof(SMART_ATTRIBUTE));
		::ZeroMemory(&(asi.Threshold[i]), sizeof(SMART_THRESHOLD));
	}

	asi.IsSmartEnabled = FALSE;
	asi.IsWord88 = FALSE;
	asi.IsWord64_76 = FALSE;
	asi.IsCheckSumError = FALSE;

	asi.IsSmartSupported = FALSE;
	asi.IsLba48Supported = FALSE;
	asi.IsNcqSupported = FALSE;
	asi.IsAamSupported = FALSE;
	asi.IsApmSupported = FALSE;
	asi.IsAamEnabled = FALSE;
	asi.IsApmEnabled = FALSE;
	asi.IsNvCacheSupported = FALSE;
	asi.IsMaxtorMinute = FALSE;

	asi.TotalDiskSize = 0;
	asi.Cylinder = 0;
	asi.Head = 0;
	asi.Sector = 0;
	asi.Sector28 = 0;
	asi.Sector48 = 0;
	asi.DiskSizeChs = 0;
	asi.DiskSizeLba28 = 0;
	asi.DiskSizeLba48 = 0;
	asi.BufferSize = 0;
	asi.NvCacheSize = 0;
	asi.TransferModeType = 0;
	asi.DetectedTimeUnitType = 0;
	asi.MeasuredTimeUnitType = 0;
	asi.AttributeCount = 0;
	asi.DetectedPowerOnHours = -1;
	asi.MeasuredPowerOnHours = -1;
	asi.PowerOnRawValue = -1;
	asi.PowerOnStartRawValue = -1;
	asi.PowerOnCount = 0;
	asi.Temperature = 0;
	asi.Speed = 0.0;
	asi.Life = -1;

	asi.Major = 0;
	asi.Minor = 0;

	asi.DiskStatus = 0;
	asi.DriveLetterMap = 0;

	asi.AlarmTemperature = 0;

	asi.VendorId = VENDOR_UNKNOWN;
	asi.InterfaceType = INTERFACE_TYPE_UNKNOWN;

	asi.VendorId = 0;
	asi.ProductId = 0;
```

```
	asi.SerialNumber = _T("");
	asi.FirmwareRev = _T("");
	asi.Model = _T("");
	asi.ModelReverse = _T("");
	asi.ModelWmi = _T("");
	asi.ModelSerial = _T("");
	asi.DriveMap = _T("");
	asi.MaxTransferMode = _T("");
	asi.CurrentTransferMode = _T("");
	asi.MajorVersion = _T("");
	asi.MinorVersion = _T("");
	asi.Interface = _T("");
	asi.Enclosure = _T("");

	CHAR buf[64];

	// Перевірка помилок
	BYTE sum = 0;
	BYTE checkSum[IDENTIFY_BUFFER_SIZE];
	memcpy(checkSum, (void *)identify, IDENTIFY_BUFFER_SIZE);
	for(int j = 0; j < IDENTIFY_BUFFER_SIZE; j++)
	{
		sum += checkSum[j];
	}
	if(sum != 0)
	{
		asi.IsCheckSumError = TRUE;
	}

	// Обооротні дії
	strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify-
>SerialNumber));
	asi.SerialNumberReverse = buf;
	asi.SerialNumberReverse.TrimLeft();
	asi.SerialNumberReverse.TrimRight();
	strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
	asi.FirmwareRevReverse = buf;
	asi.FirmwareRevReverse.TrimLeft();
	asi.FirmwareRevReverse.TrimRight();
	strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
	asi.ModelReverse = buf;
	asi.ModelReverse.TrimLeft();
	asi.ModelReverse.TrimRight();

	ChangeByteOrder(identify->SerialNumber, sizeof(identify->SerialNumber));
	ChangeByteOrder(identify->FirmwareRev, sizeof(identify->FirmwareRev));
	ChangeByteOrder(identify->Model, sizeof(identify->Model));

	if(CheckAsciiStringError(identify->SerialNumber, sizeof(identify-
>SerialNumber))
	|| CheckAsciiStringError(identify->FirmwareRev, sizeof(identify-
>FirmwareRev))
	|| CheckAsciiStringError(identify->Model, sizeof(identify->Model)))
	{
		return FALSE;
	}



	// Запис даних у структуру
	strncpy_s(buf, 64, identify->SerialNumber, sizeof(identify-
>SerialNumber));
	asi.SerialNumber = buf;
	asi.SerialNumber.TrimLeft();
	asi.SerialNumber.TrimRight();
	strncpy_s(buf, 64, identify->FirmwareRev, sizeof(identify->FirmwareRev));
	asi.FirmwareRev = buf;
	asi.FirmwareRev.TrimLeft();
	asi.FirmwareRev.TrimRight();
```

```
	strncpy_s(buf, 64, identify->Model, sizeof(identify->Model));
	asi.Model = buf;
	asi.Model.TrimLeft();
	asi.Model.TrimRight();

	if(asi.Model.IsEmpty() || asi.FirmwareRev.IsEmpty())
	{
		return FALSE;
	}

// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//	asi.Model = _T(" MTRON ") + asi.Model;

	asi.ModelSerial = asi.Model + asi.SerialNumber;
	asi.ModelSerial.Replace(_T("/"), _T(""));

	asi.Major = GetAtaMajorVersion(identify->MajorVersion, asi.MajorVersion);
	asi.TransferModeType = GetTransferMode(identify->MultiWordDma, identify-
>SerialAtaCapabilities,
					identify->UltraDmaMode,
asi.CurrentTransferMode, asi.MaxTransferMode,
					asi.Interface, &asi.InterfaceType);
	asi.DetectedTimeUnitType = GetTimeUnitType(asi.Model, asi.FirmwareRev,
asi.Major, asi.TransferModeType);

	// Характеристика
	if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported1 & (1 << 0))
	{
		asi.IsSmartSupported = TRUE;
	}
	if(asi.Major >= 3 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 3))
	{
		asi.IsApmSupported = TRUE;
		if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 3))
		{
			asi.IsApmEnabled = TRUE;
		}
	}
	if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 9))
	{
		asi.IsAamSupported = TRUE;
		if(asi.IdentifyDevice.CommandSetEnabled2 & (1 << 9))
		{
			asi.IsAamEnabled = TRUE;
		}
	}

	if(asi.Major >= 5 && asi.IdentifyDevice.CommandSetSupported2 & (1 << 10))
	{
		asi.IsLba48Supported = TRUE;
	}

	if(asi.Major >= 6 && asi.IdentifyDevice.SerialAtaCapabilities & (1 << 8))
	{
		asi.IsNcqSupported = TRUE;
	}
	if(asi.Major >= 7 && asi.IdentifyDevice.NvCacheCapabilities & (1 << 0))
	{
		asi.IsNvCacheSupported = TRUE;
	}

	CString model = asi.Model;
	model.MakeUpper();
	if(model.Find(_T("MAXTOR")) == 0 && asi.DetectedTimeUnitType ==
POWER_ON_MINUTES)
	{
		asi.IsMaxtorMinute = TRUE;
	}
```

```
      // Розмір диску та розмір буферу
      asi.Cylinder = identify->LogicalCylinders;
      asi.Head = identify->LlogicalHeads;
      asi.Sector = identify->LogicalSectors;
      asi.Sector28 = identify->TotalAddressableSectors;
      asi.Sector48 = identify->MaxUserLba;
      asi.DiskSizeChs   = (DWORD)(((ULONGLONG)identify->LogicalCylinders *
identify->LlogicalHeads * identify->LogicalSectors * 512) / 1000 / 1000  - 50);
      asi.DiskSizeLba28 = (DWORD)(((ULONGLONG)identify->TotalAddressableSectors
* 512) / 1000 / 1000 - 50);
      if(asi.IsLba48Supported)
      {
            asi.DiskSizeLba48 = (DWORD)(((ULONGLONG)identify->MaxUserLba * 512)
/ 1000 / 1000 - 50);
      }
      asi.BufferSize = identify->BufferSize * 512;
      if(asi.IsNvCacheSupported)
      {
            asi.NvCacheSize = identify->NvCacheSizeLogicalBlocks * 512;
      }

      if(asi.DiskSizeChs == 0)
      {
            asi.TotalDiskSize = 0;
      }
      else if(asi.DiskSizeLba48 > asi.DiskSizeLba28)
      {
            asi.TotalDiskSize = asi.DiskSizeLba48;
      }
      else if(asi.DiskSizeLba28 > asi.DiskSizeChs)
      {
            asi.TotalDiskSize = asi.DiskSizeLba28;
      }
      else
      {
            asi.TotalDiskSize = asi.DiskSizeChs;
      }

      // Перевірка помилок для контролеру External ATA
      if(asi.IsLba48Supported && (identify->TotalAddressableSectors < 268435455
&& asi.DiskSizeLba28 != asi.DiskSizeLba48))
      {
            asi.DiskSizeLba48 = 0;
      }

      // SSD Життя
      if(asi.Model.Find(_T("MTRON")) == 0)
      {
            asi.VendorId = VENDOR_MTRON;
      }

      // перевірка підтримки S.M.A.R.T.
      switch(asi.CommandType)
      {
      case CMD_TYPE_PHYSICAL_DRIVE:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//          SendAtaCommandPd(physicalDriveId, 0xEF, 0x42, 0xFE);
//          SendAtaCommandPd(physicalDriveId, 0xEF, 0x05, 0xFE);

            if(GetSmartAttributePd(physicalDriveId, &asi))
            {
                  GetSmartThresholdPd(physicalDriveId, &asi);
                  asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                  asi.IsSmartEnabled = TRUE;
            }
            else if(ControlSmartStatusPd(physicalDriveId, ENABLE_SMART))
            {
                  if(GetSmartAttributePd(physicalDriveId, &asi))
```

```
                  {
                        GetSmartThresholdPd(physicalDriveId, &asi);
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsSmartEnabled = TRUE;
                  }
            }
            break;
      case CMD_TYPE_SCSI_MINIPORT:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEC, 0x00, 0x00); // ID
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x05, 0x80); // APM
//          SendAtaCommandScsi(scsiPort, scsiTargetId, 0xEF, 0x42, 0x80); // AAM

            if(GetSmartAttributeScsi(scsiPort, scsiTargetId, &asi))
            {
                  GetSmartThresholdScsi(scsiPort, scsiTargetId, &asi);
                  asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                  asi.IsSmartEnabled = TRUE;
            }
            else if(ControlSmartStatusScsi(scsiPort, scsiTargetId,
ENABLE_SMART))
            {
                  if(GetSmartAttributeScsi(scsiPort, scsiTargetId, &asi))
                  {
                        GetSmartThresholdScsi(scsiPort, scsiTargetId, &asi);
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsSmartEnabled = TRUE;
                  }
            }
            break;
      case CMD_TYPE_SAT:
      case CMD_TYPE_SUNPLUS:
      case CMD_TYPE_IO_DATA:
      case CMD_TYPE_LOGITEC:
      case CMD_TYPE_JMICRON:
      case CMD_TYPE_CYPRESS:
// РОЗШИФРУВАННЯ ЗНАЧЕНЬ
//          SendAtaCommandSat(physicalDriveId, 0xEF, 0x05, 0xC0,
asi.CommandType);
//          SendAtaCommandSat(physicalDriveId, 0xEF, 0x42, 0xC0,
asi.CommandType);

            if(GetSmartAttributeSat(physicalDriveId, &asi))
            {
                  GetSmartThresholdSat(physicalDriveId, &asi);
                  asi.DiskStatus = CheckDiskStatus(asi.Attribute, asi.Threshold,
asi.AttributeCount, asi.VendorId);
                  asi.IsSmartEnabled = TRUE;
            }
            else if(ControlSmartStatusSat(physicalDriveId, ENABLE_SMART,
asi.CommandType))
            {
                  if(GetSmartAttributeSat(physicalDriveId, &asi))
                  {
                        GetSmartThresholdSat(physicalDriveId, &asi);
                        asi.DiskStatus = CheckDiskStatus(asi.Attribute,
asi.Threshold, asi.AttributeCount, asi.VendorId);
                        asi.IsSmartEnabled = TRUE;
                  }
            }
            break;
      default:
            return FALSE;
            break;
      }
```

```
	for(int i = 0; i < vars.GetCount(); i++)
	{
		if(asi.Model.Compare(vars[i].Model) == 0 &&
asi.SerialNumber.Compare(vars[i].SerialNumber) == 0)
		{
			return FALSE;
		}
	}

	asi.PowerOnStartRawValue = asi.PowerOnRawValue;

	vars.Add(asi);

	return TRUE;
}

BOOL CTestdisk::GetMain(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
INTERFACE_TYPE interfaceType, VENDOR_ID vendorId)
{
	if(vars.GetCount() > MAX_DISK)
	{
		return FALSE;
	}
	// Перевірка перекриття
	for(int i = 0; i < vars.GetCount(); i++)
	{
		if(physicalDriveId >= 0 && vars[i].PhysicalDriveId ==
physicalDriveId)
		{
			return FALSE;
		}
		else if(scsiPort >= 0 && scsiTargetId >= 0
			&&	vars[i].ScsiPort == scsiPort && vars[i].ScsiTargetId ==
scsiTargetId)
		{
			return FALSE;
		}
	}

	IDENTIFY_DRIVE identify = {0};

	if(interfaceType == INTERFACE_TYPE_UNKNOWN || interfaceType ==
INTERFACE_TYPE_PATA || interfaceType == INTERFACE_TYPE_SATA)
	{
		if(physicalDriveId >= 0 && DoIdentifyDevicePd(physicalDriveId,
&identify))
		{
			return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_PHYSICAL_DRIVE, &identify);
		}
		else if(scsiPort >= 0 && scsiTargetId >= 0 &&
DoIdentifyDeviceScsi(scsiPort, scsiTargetId, &identify))
		{
			return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SCSI_MINIPORT, &identify);
		}
	}
	else if(physicalDriveId >= 0)
	{
		/** РОЗШИФРУВАННЯ ЗНАЧЕНЬ
		if(TRUE)
		{
			DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_DEBUG);
		}
		else
		*/
		if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_IO_DATA)
```

```
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_IO_DATA))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_IO_DATA, &identify);
                  }
            }
            if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_LOGITEC)
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_LOGITEC))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_LOGITEC, &identify);
                  }
            }
            if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_SUNPLUS)
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SUNPLUS))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SUNPLUS, &identify);
                  }
            }
            else if(interfaceType == INTERFACE_TYPE_USB && vendorId ==
USB_VENDOR_CYPRESS)
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_CYPRESS))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_CYPRESS, &identify);
                  }
            }
            else if(interfaceType == INTERFACE_TYPE_USB &&
                  (vendorId == USB_VENDOR_INITIO || vendorId ==
USB_VENDOR_OXFORD)
                  )
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SAT))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SAT, &identify);
                  }
            }
            else
            {
                  if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SAT))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SAT, &identify);
                  }
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_JMICRON))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_JMICRON, &identify);
                  }
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_SUNPLUS))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_SUNPLUS, &identify);
```

```
                  }
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_CYPRESS))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_CYPRESS, &identify);
                  }
                  else if(DoIdentifyDeviceSat(physicalDriveId, &identify,
CMD_TYPE_LOGITEC))
                  {
                        return AddDisk(physicalDriveId, scsiPort, scsiTargetId,
CMD_TYPE_LOGITEC, &identify);
                  }
            }
      }

      return FALSE;
}

/*-------------------------------------------------------------------------*/
// \\\\.\\Фізичний диск X
/*-------------------------------------------------------------------------*/
HANDLE CTestdisk::GetIoCtrlHandle(BYTE index)
{
      CString     strDevice;
      strDevice.Format(_T("\\\\.\\PhysicalDrive%d"), index);

      return ::CreateFile(strDevice, GENERIC_READ | GENERIC_WRITE,
            FILE_SHARE_READ | FILE_SHARE_WRITE,
            NULL, OPEN_EXISTING, 0, NULL);
}

BOOL CTestdisk::DoIdentifyDevicePd(INT physicalDriveId, IDENTIFY_DRIVE* data)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      IDENTIFY_DRIVE_OUTDATA  sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      if(data == NULL)
      {
            return      FALSE;
      }

      if(! SendAtaCommandPd(physicalDriveId, 0xEC, 0x00, 0x00, (PBYTE)data,
sizeof(IDENTIFY_DRIVE)))
      {
            ::ZeroMemory(data, sizeof(IDENTIFY_DRIVE));
            hIoCtrl = GetIoCtrlHandle(physicalDriveId);
            if(hIoCtrl == INVALID_HANDLE_VALUE)
            {
                  return      FALSE;
            }
            ::ZeroMemory(&sendCmdOutParam, sizeof(IDENTIFY_DRIVE_OUTDATA));
            ::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

            sendCmd.irDriveRegs.bCommandReg                 = ID_CMD;
            sendCmd.irDriveRegs.bSectorCountReg       = 1;
            sendCmd.irDriveRegs.bSectorNumberReg      = 1;
            sendCmd.cBufferSize                                   =
IDENTIFY_BUFFER_SIZE;

            bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
                  &sendCmd, sizeof(SENDCMDINPARAMS),
                  &sendCmdOutParam, sizeof(IDENTIFY_DRIVE_OUTDATA),
                  &dwReturned, NULL);
```

```
            ::CloseHandle(hIoCtrl);

            if(bRet == FALSE || dwReturned != sizeof(IDENTIFY_DRIVE_OUTDATA))
            {
                  return      FALSE;
            }

            memcpy_s(data, sizeof(IDENTIFY_DRIVE),
sendCmdOutParam.SendCmdOutParam.bBuffer, sizeof(IDENTIFY_DRIVE));
      }

      return      TRUE;
}

BOOL CTestdisk::GetSmartAttributePd(INT PhysicalDriveId, ATA_SMART_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      SMART_READ_DATA_OUTDATA sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      hIoCtrl = GetIoCtrlHandle(PhysicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sendCmdOutParam, sizeof(SMART_READ_DATA_OUTDATA));
      ::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

      sendCmd.irDriveRegs.bFeaturesReg    = READ_ATTRIBUTES;
      sendCmd.irDriveRegs.bSectorCountReg = 1;
      sendCmd.irDriveRegs.bSectorNumberReg= 1;
      sendCmd.irDriveRegs.bCylLowReg            = SMART_CYL_LOW;
      sendCmd.irDriveRegs.bCylHighReg           = SMART_CYL_HI;
      sendCmd.irDriveRegs.bCommandReg           = SMART_CMD;
      sendCmd.cBufferSize                             =
READ_ATTRIBUTE_BUFFER_SIZE;

      bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
            &sendCmd, sizeof(SENDCMDINPARAMS),
            &sendCmdOutParam, sizeof(SMART_READ_DATA_OUTDATA),
            &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != sizeof(SMART_READ_DATA_OUTDATA))
      {
            return      FALSE;
      }

      CString str;
      asi->AttributeCount = 0;
      int j = 0;
      for(int i = 0; i < MAX_ATTRIBUTE; i++)
      {
            DWORD rawValue = 0;
            memcpy(     &(asi->Attribute[j]),
                        &(sendCmdOutParam.Data[i * sizeof(SMART_ATTRIBUTE) +
1]), sizeof(SMART_ATTRIBUTE));

            if(asi->Attribute[j].Id != 0)
            {
                  switch(asi->Attribute[j].Id)
                  {
                  case 0x09: // Кількість відпрацьованих годин у включеному
стані
```

```
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnRawValue = rawValue;
                        asi->DetectedPowerOnHours = GetPowerOnHours(rawValue,
asi->DetectedTimeUnitType);
                        asi->MeasuredPowerOnHours = GetPowerOnHours(rawValue,
asi->MeasuredTimeUnitType);
                        break;
                  case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnCount = rawValue;
                        break;
                  case 0xC2: // Температура
                        if(asi->Attribute[j].RawValue[0] > 0)
                        {
                              asi->Temperature = asi->Attribute[j].RawValue[0];
                        }
                        else
                        {
                              asi->Temperature = asi->Attribute[j].CurrentValue;
                        }
                  case 0xBB: // Визначення фірми-розробника
                        if(asi->VendorId == VENDOR_MTRON)
                        {
                              asi->Life = asi->Attribute[j].CurrentValue;
                        }
                        break;
                  default:
                        break;
                  }
                  j++;
            }
      }
      asi->AttributeCount = j;

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CTestdisk::GetSmartThresholdPd(INT physicalDriveId, ATA_SMART_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      SMART_READ_DATA_OUTDATA sendCmdOutParam;
      SENDCMDINPARAMS   sendCmd;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }
```

```
	::ZeroMemory(&sendCmdOutParam, sizeof(SMART_READ_DATA_OUTDATA));
	::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));

	sendCmd.irDriveRegs.bFeaturesReg    = READ_THRESHOLDS;
	sendCmd.irDriveRegs.bCylLowReg           = SMART_CYL_LOW;
	sendCmd.irDriveRegs.bCylHighReg          = SMART_CYL_HI;
	sendCmd.irDriveRegs.bCommandReg          = SMART_CMD;
	sendCmd.cBufferSize                           =
READ_THRESHOLD_BUFFER_SIZE;

	bRet = ::DeviceIoControl(hIoCtrl, DFP_RECEIVE_DRIVE_DATA,
		&sendCmd, sizeof(SENDCMDINPARAMS),
		&sendCmdOutParam, sizeof(SMART_READ_DATA_OUTDATA),
		&dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	if(bRet == FALSE || dwReturned != sizeof(SMART_READ_DATA_OUTDATA))
	{
		return	FALSE;
	}

	CString str;
	int j = 0;
	for(int i = 0; i < MAX_ATTRIBUTE; i++)
	{
		memcpy(	&(asi->Threshold[i]),
			&(sendCmdOutParam.Data[i * sizeof(SMART_THRESHOLD) +
1]), sizeof(SMART_THRESHOLD));

		if(asi->Threshold[j].Id != 0)
		{
			j++;
		}
	}

	return	TRUE;
}

BOOL CTestdisk::ControlSmartStatusPd(INT physicalDriveId, BYTE command)
{
	BOOL  bRet;
	HANDLE	hIoCtrl;
	DWORD dwReturned;

	SENDCMDINPARAMS		sendCmd;
	SENDCMDOUTPARAMS  sendCmdOutParam;

	hIoCtrl = GetIoCtrlHandle(physicalDriveId);
	if(hIoCtrl == INVALID_HANDLE_VALUE)
	{
		return	FALSE;
	}

	::ZeroMemory(&sendCmd, sizeof(SENDCMDINPARAMS));
	::ZeroMemory(&sendCmdOutParam, sizeof(SENDCMDOUTPARAMS));

	sendCmd.irDriveRegs.bFeaturesReg    = command;
	sendCmd.irDriveRegs.bSectorCountReg = 1;
	sendCmd.irDriveRegs.bSectorNumberReg= 1;
	sendCmd.irDriveRegs.bCylLowReg           = SMART_CYL_LOW;
	sendCmd.irDriveRegs.bCylHighReg          = SMART_CYL_HI;
	sendCmd.irDriveRegs.bCommandReg          = SMART_CMD;
	sendCmd.cBufferSize                           = 0;

	bRet = ::DeviceIoControl(hIoCtrl, DFP_SEND_DRIVE_COMMAND,
		&sendCmd, sizeof(SENDCMDINPARAMS) - 1,
		&sendCmdOutParam, sizeof(SENDCMDOUTPARAMS) -1,
```

```
            &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      return      bRet;
}

BOOL CTestdisk::SendAtaCommandPd(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, PBYTE data, DWORD dataSize)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      if(m_FlagAtaPassThrough)
      {
            ATA_PASS_THROUGH_EX_WITH_BUFFERS ab;
            ::ZeroMemory(&ab, sizeof(ab));
            ab.Apt.Length = sizeof(ATA_PASS_THROUGH_EX);
            ab.Apt.TimeOutValue = 10;
            DWORD size = offsetof(ATA_PASS_THROUGH_EX_WITH_BUFFERS, Buf);
            ab.Apt.DataBufferOffset = size;

            if(dataSize > 0)
            {
                  if(dataSize > sizeof(ab.Buf))
                  {
                        return FALSE;
                  }
                  ab.Apt.AtaFlags = ATA_FLAGS_DATA_IN;
                  ab.Apt.DataTransferLength = dataSize;
                  size += dataSize;
            }

            ab.Apt.CurrentTaskFile.bFeaturesReg = sub;
            ab.Apt.CurrentTaskFile.bSectorCountReg = param;
            ab.Apt.CurrentTaskFile.bCommandReg = main;

            bRet = ::DeviceIoControl(hIoCtrl, IOCTL_ATA_PASS_THROUGH,
                  &ab, size, &ab, size, &dwReturned, NULL);
            ::CloseHandle(hIoCtrl);
            if(bRet && dataSize && data != NULL)
            {
                  memcpy_s(data, dataSize, ab.Buf, dataSize);
            }
      }
      else if(m_Os.dwMajorVersion == 4)
      {
            return FALSE;
      }
      else
      {
            DWORD size = sizeof(CMD_IDE_PATH_THROUGH) - 1 + dataSize;
            CMD_IDE_PATH_THROUGH* buf =
(CMD_IDE_PATH_THROUGH*)VirtualAlloc(NULL, size, MEM_COMMIT, PAGE_READWRITE);

            buf->reg.bFeaturesReg         = sub;
            buf->reg.bSectorCountReg      = param;
            buf->reg.bSectorNumberReg     = 0;
            buf->reg.bCylLowReg                 = 0;
            buf->reg.bCylHighReg          = 0;
            buf->reg.bDriveHeadReg        = 0;
            buf->reg.bCommandReg          = main;
```

```
            buf->reg.bReserved                = 0;
            buf->length                         = dataSize;

            bRet = ::DeviceIoControl(hIoCtrl, IOCTL_IDE_PASS_THROUGH,
                  buf, size, buf, size, &dwReturned, NULL);
            ::CloseHandle(hIoCtrl);
            if(bRet && dataSize && data != NULL)
            {
                  memcpy_s(data, dataSize, buf->buffer, dataSize);
            }
            VirtualFree(buf, 0, MEM_RELEASE);
      }

      return      bRet;
}

/*-------------------------------------------------------------------------*/
//  \\\\.\\ Диск SCSI X
/*-------------------------------------------------------------------------*/

BOOL CTestdisk::DoIdentifyDeviceScsi(INT scsiPort, INT scsiTargetId,
IDENTIFY_DRIVE* identify)
{
      int done = FALSE;
      int controller = 0;
      int current = 0;
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
IDENTIFY_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;

            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof (SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + IDENTIFY_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_IDENTIFY;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bCommandReg = ID_CMD;
            pin->bDriveNumber = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + IDENTIFY_BUFFER_SIZE,
                                                &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        done = TRUE;
                        memcpy_s(identify, sizeof(IDENTIFY_DRIVE), pOut-
>bBuffer, sizeof(IDENTIFY_DRIVE));
                  }
            }
            CloseHandle(hScsiDriveIOCTL);
      }
      return done;
```

```
}

BOOL CTestdisk::GetSmartAttributeScsi(INT scsiPort, INT scsiTargetId,
ATA_SMART_INFO* asi)
{
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
READ_ATTRIBUTE_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;
            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof(SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + READ_ATTRIBUTE_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_READ_SMART_ATTRIBS;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bFeaturesReg       = READ_ATTRIBUTES;
            pin->irDriveRegs.bSectorCountReg    = 1;
            pin->irDriveRegs.bSectorNumberReg   = 1;
            pin->irDriveRegs.bCylLowReg               = SMART_CYL_LOW;
            pin->irDriveRegs.bCylHighReg        = SMART_CYL_HI;
            pin->irDriveRegs.bCommandReg        = SMART_CMD;
            pin->cBufferSize                          =
READ_ATTRIBUTE_BUFFER_SIZE;
            pin->bDriveNumber                         = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + READ_ATTRIBUTE_BUFFER_SIZE,
                                                &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        CString str;
                        asi->AttributeCount = 0;
                        int j = 0;

                        for(int i = 0; i < MAX_ATTRIBUTE; i++)
                        {
                              DWORD rawValue = 0;

                              memcpy(     &(asi->Attribute[j]),
                                    &(pOut->bBuffer[i * sizeof(SMART_ATTRIBUTE)
+ 2]), sizeof(SMART_ATTRIBUTE));

                              if(asi->Attribute[j].Id != 0)
                              {
                                    switch(asi->Attribute[j].Id)
                                    {
                                    case 0x09: // Кількість відпрацьованих годин
у включеному стані
                                          rawValue = MAKELONG(
                                                MAKEWORD(asi-
>Attribute[j].RawValue[0], asi->Attribute[j].RawValue[1]),
                                                MAKEWORD(asi-
>Attribute[j].RawValue[2], asi->Attribute[j].RawValue[3])
```

```
                                                );
                                          asi->PowerOnRawValue = rawValue;
                                          asi->DetectedPowerOnHours =
GetPowerOnHours(rawValue, asi->DetectedTimeUnitType);
                                          asi->MeasuredPowerOnHours =
GetPowerOnHours(rawValue, asi->MeasuredTimeUnitType);
                                          break;
                                    case 0x0C: // Кількість зафіксованих
повторів включення/вимикання живлення накопичувача
                                          rawValue = MAKELONG(
                                                MAKEWORD(asi-
>Attribute[j].RawValue[0], asi->Attribute[j].RawValue[1]),
                                                MAKEWORD(asi-
>Attribute[j].RawValue[2], asi->Attribute[j].RawValue[3])
                                                );
                                          asi->PowerOnCount = rawValue;
                                          break;
                                    case 0xC2: // Температура
                                          if(asi->Attribute[j].RawValue[0] > 0)
                                          {
                                                asi->Temperature = asi-
>Attribute[j].RawValue[0];
                                          }
                                          else
                                          {
                                                asi->Temperature = asi-
>Attribute[j].CurrentValue;
                                          }
                                          break;
                                    case 0xBB: // Визначення фірми-розробника
                                          if(asi->VendorId == VENDOR_MTRON)
                                          {
                                                asi->Life = asi-
>Attribute[j].CurrentValue;
                                          }
                                          break;
                                    default:
                                          break;
                                    }
                                    j++;
                              }
                        }
                        asi->AttributeCount = j;
                  }
            }
      }
      CloseHandle(hScsiDriveIOCTL);

      if(asi->AttributeCount > 0)
      {
            return      TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CTestdisk::GetSmartThresholdScsi(INT scsiPort, INT scsiTargetId,
ATA_SMART_INFO* asi)
{
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
      if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
      {
```

```
            BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
READ_THRESHOLD_BUFFER_SIZE];
            SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
            SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
            DWORD dummy;
            memset(buffer, 0, sizeof(buffer));
            p->HeaderLength = sizeof(SRB_IO_CONTROL);
            p->Timeout = 2;
            p->Length = sizeof(SENDCMDOUTPARAMS) + READ_THRESHOLD_BUFFER_SIZE;
            p->ControlCode = IOCTL_SCSI_MINIPORT_READ_SMART_THRESHOLDS;
            memcpy((char *)p->Signature, "SCSIDISK", 8);
            pin->irDriveRegs.bFeaturesReg       = READ_THRESHOLDS;
            pin->irDriveRegs.bSectorCountReg    = 1;
            pin->irDriveRegs.bSectorNumberReg   = 1;
            pin->irDriveRegs.bCylLowReg               = SMART_CYL_LOW;
            pin->irDriveRegs.bCylHighReg        = SMART_CYL_HI;
            pin->irDriveRegs.bCommandReg        = SMART_CMD;
            pin->cBufferSize                          =
READ_THRESHOLD_BUFFER_SIZE;
            pin->bDriveNumber                         = scsiTargetId;

            if(DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
                                                buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + READ_THRESHOLD_BUFFER_SIZE,
                                                &dummy, NULL))
            {
                  SENDCMDOUTPARAMS *pOut = (SENDCMDOUTPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
                  if(*(pOut->bBuffer) > 0)
                  {
                        int j = 0;
                        for(int i = 0; i < MAX_ATTRIBUTE; i++)
                        {
                              memcpy(     &(asi->Threshold[j]),
                                    &(pOut->bBuffer[i * sizeof(SMART_THRESHOLD)
+ 2]), sizeof(SMART_THRESHOLD));
                              if(asi->Threshold[j].Id != 0)
                              {
                                    j++;
                              }
                        }
                  }
            }
      }
      CloseHandle (hScsiDriveIOCTL);

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CTestdisk::ControlSmartStatusScsi(INT scsiPort, INT scsiTargetId, BYTE
command)
{
      BOOL  bRet;
      HANDLE hScsiDriveIOCTL = 0;
      CString driveName;
      driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
      hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
                                          FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
```

```
	if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
	{
		BYTE buffer[sizeof(SRB_IO_CONTROL) + sizeof(SENDCMDOUTPARAMS) +
SCSI_MINIPORT_BUFFER_SIZE];
		SRB_IO_CONTROL *p = (SRB_IO_CONTROL *)buffer;
		SENDCMDINPARAMS *pin = (SENDCMDINPARAMS *)(buffer +
sizeof(SRB_IO_CONTROL));
		DWORD dummy;
		memset(buffer, 0, sizeof(buffer));
		p->HeaderLength = sizeof(SRB_IO_CONTROL);
		p->Timeout = 2;
		p->Length = sizeof(SENDCMDOUTPARAMS) + SCSI_MINIPORT_BUFFER_SIZE;
		if(command == DISABLE_SMART)
		{
			p->ControlCode = IOCTL_SCSI_MINIPORT_DISABLE_SMART;
		}
		else
		{
			p->ControlCode = IOCTL_SCSI_MINIPORT_ENABLE_SMART;
		}
		memcpy((char *)p->Signature, "SCSIDISK", 8);
		pin->irDriveRegs.bFeaturesReg	= command;
		pin->irDriveRegs.bSectorCountReg	= 1;
		pin->irDriveRegs.bSectorNumberReg	= 1;
		pin->irDriveRegs.bCylLowReg		= SMART_CYL_LOW;
		pin->irDriveRegs.bCylHighReg	= SMART_CYL_HI;
		pin->irDriveRegs.bCommandReg	= SMART_CMD;
		pin->cBufferSize			=
SCSI_MINIPORT_BUFFER_SIZE;
		pin->bDriveNumber			= scsiTargetId;

		bRet = DeviceIoControl(hScsiDriveIOCTL, IOCTL_SCSI_MINIPORT,
					buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDINPARAMS) - 1,
					buffer, sizeof(SRB_IO_CONTROL) +
sizeof(SENDCMDOUTPARAMS) + SCSI_MINIPORT_BUFFER_SIZE,
					&dummy, NULL);
	}
	CloseHandle(hScsiDriveIOCTL);

	return bRet;
}


BOOL CTestdisk::SendAtaCommandScsi(INT scsiPort, INT scsiTargetId, BYTE main,
BYTE sub, BYTE param)
{
/** Does not work...
	BOOL  bRet;
	HANDLE hScsiDriveIOCTL = 0;
	CString driveName;
	driveName.Format(_T("\\\\.\\Scsi%d:"), scsiPort);
	hScsiDriveIOCTL = CreateFile(driveName, GENERIC_READ | GENERIC_WRITE,
				FILE_SHARE_READ | FILE_SHARE_WRITE,
NULL, OPEN_EXISTING, 0, NULL);
	if(hScsiDriveIOCTL != INVALID_HANDLE_VALUE)
	{
		DWORD dummy;
		CMD_ATA_PASS_THROUGH_WITH_BUFFERS capt;
		::ZeroMemory(&capt, sizeof(CMD_ATA_PASS_THROUGH_WITH_BUFFERS));
		capt.apt.Length = sizeof(CMD_ATA_PASS_THROUGH);
		capt.apt.PathId = 0;
		capt.apt.TargetId = 0;
		capt.apt.Lun = 0;
		capt.apt.TimeOutValue = 10;

		DWORD size = offsetof(CMD_ATA_PASS_THROUGH_WITH_BUFFERS, DataBuf);
		capt.apt.DataBufferOffset = size;
```

```
            capt.apt.AtaFlags = 0x02;
            capt.apt.DataTransferLength = 512;
            size += 512;
            capt.DataBuf[0] = 0xCF;

            capt.apt.CurrentTaskFile.bFeaturesReg= sub;
            capt.apt.CurrentTaskFile.bSectorCountReg = param;
            capt.apt.CurrentTaskFile.bDriveHeadReg = 0xA0;
            capt.apt.CurrentTaskFile.bCommandReg = main;

            bRet = DeviceIoControl(hScsiDriveIOCTL, IOCTL_ATA_PASS_THROUGH,
                                                &capt, size,
                                                &capt, size,
                                                &dummy, NULL);
      }
      CloseHandle(hScsiDriveIOCTL);

      return bRet;
*/
      return FALSE;
}

/*-------------------------------------------------------------------------*/
//  SCSI / ATA переклад (SAT)
/*-------------------------------------------------------------------------*/

BOOL CTestdisk::DoIdentifyDeviceSat(INT physicalDriveId, IDENTIFY_DRIVE* data,
COMMAND_TYPE type)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;
      DWORD length;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      if(data == NULL)
      {
            return      FALSE;
      }

      ::ZeroMemory(data, sizeof(IDENTIFY_DRIVE));

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = IDENTIFY_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);

      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xA1;//ATA проходів через(12) операцій коду(A1h)
```

```
            sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
            sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3] = 0;//FEATURES (7:0)
            sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5] = 0;//LBA_LOW (7:0)
            sptwb.Spt.Cdb[6] = 0;//LBA_MID (7:0)
            sptwb.Spt.Cdb[7] = 0;//LBA_HIGH (7:0)
            sptwb.Spt.Cdb[9] = ID_CMD;//COMMAND
      }
      else if(type == CMD_TYPE_SUNPLUS)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xF8;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0x22;
            sptwb.Spt.Cdb[3] = 0x10;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
            sptwb.Spt.Cdb[9] = 0x00;
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = 0xEC; // ID_CMD
      }
      else if(type == CMD_TYPE_IO_DATA)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xE3;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0x01;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x00;
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = 0xEC;  // ID_CMD
            sptwb.Spt.Cdb[9] = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_LOGITEC)
      {
            sptwb.Spt.CdbLength = 10;
            sptwb.Spt.Cdb[0] = 0xE0;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0x00;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x00;
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = 0xEC;  // ID_CMD
            sptwb.Spt.Cdb[9] = 0x4C;
      }
      else if(type == CMD_TYPE_JMICRON)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xDF;
            sptwb.Spt.Cdb[1] = 0x10;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0x02;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
```

```
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xEC; // ID_CMD
	}
	else if(type == CMD_TYPE_CYPRESS)
	{
		sptwb.Spt.CdbLength = 16;
		sptwb.Spt.Cdb[0] = 0x24;
		sptwb.Spt.Cdb[1] = 0x24;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0xBE;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0x00;
		sptwb.Spt.Cdb[7] = 0x01;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x00;
		sptwb.Spt.Cdb[10] = 0x00;
		sptwb.Spt.Cdb[11] = 0xA0;
		sptwb.Spt.Cdb[12] = 0xEC; // ID_CMD
		sptwb.Spt.Cdb[13] = 0x00;
		sptwb.Spt.Cdb[14] = 0x00;
		sptwb.Spt.Cdb[15] = 0x00;
	}
/*
	else if(type == CMD_TYPE_DEBUG)
	{
		sptwb.Spt.CdbLength = 16;
		for(int i = 0xA0; i <= 0xFF; i++)
		{
			for(int j = 8; j < 16; j++)
			{
				::ZeroMemory(&sptwb.Spt.Cdb, 16);
				sptwb.Spt.Cdb[0] = i;
				sptwb.Spt.Cdb[j - 1] = 0xA0;
				sptwb.Spt.Cdb[j] = 0xEC;

				length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf) + sptwb.Spt.DataTransferLength;

				bRet = ::DeviceIoControl(hIoCtrl,
IOCTL_SCSI_PASS_THROUGH,
					&sptwb, sizeof(SCSI_PASS_THROUGH),
					&sptwb, length,   &dwReturned, NULL);


				if(bRet == FALSE || dwReturned != length)
				{
					continue;
				}

				CString cstr;
				cstr.Format(_T("i = %d, j = %d"), i, j);
				AfxMessageBox(cstr);

				::CloseHandle(hIoCtrl);
				memcpy_s(data, sizeof(IDENTIFY_DRIVE), sptwb.DataBuf,
sizeof(IDENTIFY_DRIVE));

				return TRUE;
			}
		}
	}
*/
	else
	{
		return FALSE;
	}
```

```
	length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;

	bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
		&sptwb, sizeof(SCSI_PASS_THROUGH),
		&sptwb, length,   &dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	if(bRet == FALSE || dwReturned != length)
	{
		return	FALSE;
	}

	memcpy_s(data, sizeof(IDENTIFY_DRIVE), sptwb.DataBuf,
sizeof(IDENTIFY_DRIVE));

	return	TRUE;
}

BOOL CTestdisk::GetSmartAttributeSat(INT PhysicalDriveId, ATA_SMART_INFO* asi)
{
	BOOL  bRet;
	HANDLE	hIoCtrl;
	DWORD dwReturned;
	DWORD length;

	SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

	hIoCtrl = GetIoCtrlHandle(PhysicalDriveId);
	if(hIoCtrl == INVALID_HANDLE_VALUE)
	{
		return	FALSE;
	}

	::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = READ_ATTRIBUTE_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);

	COMMAND_TYPE type = asi->CommandType;
	if(type == CMD_TYPE_SAT)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xA1;//ATA прохід через (12) операцій коду (A1h)
		sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
		sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
		sptwb.Spt.Cdb[3] = READ_ATTRIBUTES;//FEATURES (7:0)
		sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
		sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
		sptwb.Spt.Cdb[6] = SMART_CYL_LOW;//LBA_MID (7:0)
		sptwb.Spt.Cdb[7] = SMART_CYL_HI;//LBA_HIGH (7:0)
		sptwb.Spt.Cdb[9] = SMART_CMD;//COMMAND
	}
	else if(type == CMD_TYPE_SUNPLUS)
	{
		sptwb.Spt.CdbLength = 12;
```

```
        sptwb.Spt.Cdb[0] = 0xF8;
        sptwb.Spt.Cdb[1] = 0x00;
        sptwb.Spt.Cdb[2] = 0x22;
        sptwb.Spt.Cdb[3] = 0x10;
        sptwb.Spt.Cdb[4] = 0x01;
        sptwb.Spt.Cdb[5] = 0xD0;  // READ_ATTRIBUTES
        sptwb.Spt.Cdb[6] = 0x01;
        sptwb.Spt.Cdb[7] = 0x00;
        sptwb.Spt.Cdb[8] = 0x4F;  // SMART_CYL_LOW
        sptwb.Spt.Cdb[9] = 0xC2;  // SMART_CYL_HIGH
        sptwb.Spt.Cdb[10] = 0xA0;
        sptwb.Spt.Cdb[11] = 0xB0; // SMART_CMD
    }
    else if(type == CMD_TYPE_IO_DATA)
    {
        sptwb.Spt.CdbLength = 12;
        sptwb.Spt.Cdb[0] = 0xE3;
        sptwb.Spt.Cdb[1] = 0x00; // ?
        sptwb.Spt.Cdb[2] = 0xD0; // READ_ATTRIBUTES
        sptwb.Spt.Cdb[3] = 0x00; // ?
        sptwb.Spt.Cdb[4] = 0x00; // ?
        sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
        sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
        sptwb.Spt.Cdb[7] = 0xA0; //
        sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
        sptwb.Spt.Cdb[9] = 0x00;
        sptwb.Spt.Cdb[10] = 0x00;
        sptwb.Spt.Cdb[11] = 0x00;
    }
    else if(type == CMD_TYPE_LOGITEC)
    {
        sptwb.Spt.CdbLength = 10;
        sptwb.Spt.Cdb[0] = 0xE0;
        sptwb.Spt.Cdb[1] = 0x00;
        sptwb.Spt.Cdb[2] = 0xD0; // READ_ATTRIBUTES
        sptwb.Spt.Cdb[3] = 0x00;
        sptwb.Spt.Cdb[4] = 0x00;
        sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
        sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
        sptwb.Spt.Cdb[7] = 0xA0;
        sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
        sptwb.Spt.Cdb[9] = 0x4C;
    }
    else if(type == CMD_TYPE_JMICRON)
    {
        sptwb.Spt.CdbLength = 12;
        sptwb.Spt.Cdb[0] = 0xDF;
        sptwb.Spt.Cdb[1] = 0x10;
        sptwb.Spt.Cdb[2] = 0x00;
        sptwb.Spt.Cdb[3] = 0x02;
        sptwb.Spt.Cdb[4] = 0x00;
        sptwb.Spt.Cdb[5] = 0xD0;  // READ_ATTRIBUTES
        sptwb.Spt.Cdb[6] = 0x01;
        sptwb.Spt.Cdb[7] = 0x01;
        sptwb.Spt.Cdb[8] = 0x4F;  // SMART_CYL_LOW
        sptwb.Spt.Cdb[9] = 0xC2;  // SMART_CYL_HIGH
        sptwb.Spt.Cdb[10] = 0xA0;
        sptwb.Spt.Cdb[11] = 0xB0; // SMART_CMD
    }
    else if(type == CMD_TYPE_CYPRESS)
    {
        sptwb.Spt.CdbLength = 16;
        sptwb.Spt.Cdb[0] = 0x24;
        sptwb.Spt.Cdb[1] = 0x24;
        sptwb.Spt.Cdb[2] = 0x00;
        sptwb.Spt.Cdb[3] = 0xBE;
        sptwb.Spt.Cdb[4] = 0x01;
        sptwb.Spt.Cdb[5] = 0x00;
        sptwb.Spt.Cdb[6] = 0xD0;  // READ_ATTRIBUTES
```

```
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x00;
            sptwb.Spt.Cdb[9] = 0x4F;  // SMART_CYL_LOW
            sptwb.Spt.Cdb[10] = 0xC2; // SMART_CYL_HIGH
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);

      ::CloseHandle(hIoCtrl);

      if(bRet == FALSE || dwReturned != length)
      {
            return      FALSE;
      }

      CString str;
      asi->AttributeCount = 0;
      int j = 0;
      for(int i = 0; i < MAX_ATTRIBUTE; i++)
      {
            DWORD rawValue = 0;
            memcpy(     &(asi->Attribute[j]),
                        &(sptwb.DataBuf[i * sizeof(SMART_ATTRIBUTE) + 2]),
sizeof(SMART_ATTRIBUTE));

            if(asi->Attribute[j].Id != 0)
            {
                  switch(asi->Attribute[j].Id)
                  {
                  case 0x09: // Кількість відпрацьованих годин у включеному
стані
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnRawValue = rawValue;
                        asi->DetectedPowerOnHours = GetPowerOnHours(rawValue,
asi->DetectedTimeUnitType);
                        asi->MeasuredPowerOnHours = GetPowerOnHours(rawValue,
asi->MeasuredTimeUnitType);
                        break;
                  case 0x0C: // Кількість зафіксованих повторів
включення/вимикання живлення накопичувача
                        rawValue = MAKELONG(
                              MAKEWORD(asi->Attribute[j].RawValue[0], asi-
>Attribute[j].RawValue[1]),
                              MAKEWORD(asi->Attribute[j].RawValue[2], asi-
>Attribute[j].RawValue[3])
                              );
                        asi->PowerOnCount = rawValue;
                        break;
                  case 0xC2: // Температура
                        if(asi->Attribute[j].RawValue[0] > 0)
                        {
```

```
                              asi->Temperature = asi->Attribute[j].RawValue[0];
                        }
                        else
                        {
                              asi->Temperature = asi->Attribute[j].CurrentValue;
                        }
                        break;
                  case 0xBB: // Визначення фірми-розробника
                        if(asi->VendorId == VENDOR_MTRON)
                        {
                              asi->Life = asi->Attribute[j].CurrentValue;
                        // РОЗШИФРУВАННЯ ЗНАЧЕНЬ
                        //    asi->Life = 0;
                        //    asi->Attribute[j].CurrentValue = 0;
                        }
                        break;
                  default:
                        break;
                  }
                  j++;
            }
      }
      asi->AttributeCount = j;

      if(asi->AttributeCount > 0)
      {
            return TRUE;
      }
      else
      {
            return FALSE;
      }
}

BOOL CTestdisk::GetSmartThresholdSat(INT physicalDriveId, ATA_SMART_INFO* asi)
{
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;
      DWORD length;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = READ_THRESHOLD_BUFFER_SIZE;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);

      COMMAND_TYPE type = asi->CommandType;
      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
```

```
            sptwb.Spt.Cdb[0] = 0xA1; ////ATA Проходів через (12) операцій коду
(A1h)
            sptwb.Spt.Cdb[1] = (4 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=4(PIO
Data-In),Reserved
            sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3] = READ_THRESHOLDS;//FEATURES (7:0)
            sptwb.Spt.Cdb[4] = 1;//SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
            sptwb.Spt.Cdb[6] = SMART_CYL_LOW;//LBA_MID (7:0)
            sptwb.Spt.Cdb[7] = SMART_CYL_HI;//LBA_HIGH (7:0)
            sptwb.Spt.Cdb[9] = SMART_CMD;//COMMAND
      }
      else if(type == CMD_TYPE_SUNPLUS)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xF8;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0x22;
            sptwb.Spt.Cdb[3] = 0x10;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = 0xD1; // READ_THRESHOLD
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x01;
            sptwb.Spt.Cdb[8] = 0x4F; // SMART_CYL_LOW
            sptwb.Spt.Cdb[9] = 0xC2; // SMART_CYL_HIGH
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = 0xB0;// SMART_CMD
      }
      else if(type == CMD_TYPE_IO_DATA)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xE3;
            sptwb.Spt.Cdb[1] = 0x00; // ?
            sptwb.Spt.Cdb[2] = 0xD1; // READ_THRESHOLD
            sptwb.Spt.Cdb[3] = 0x00; // ?
            sptwb.Spt.Cdb[4] = 0x00; // ?
            sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
            sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
            sptwb.Spt.Cdb[7] = 0xA0; //
            sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
            sptwb.Spt.Cdb[9] = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_LOGITEC)
      {
            sptwb.Spt.CdbLength = 10;
            sptwb.Spt.Cdb[0] = 0xE0;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0xD1; // READ_THRESHOLD
            sptwb.Spt.Cdb[3] = 0x00;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
            sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
            sptwb.Spt.Cdb[9] = 0x4C;
      }
      else if(type == CMD_TYPE_JMICRON)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xDF;
            sptwb.Spt.Cdb[1] = 0x10;
            sptwb.Spt.Cdb[2] = 0x00;
            sptwb.Spt.Cdb[3] = 0x02;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0xD1;  // READ_THRESHOLD
            sptwb.Spt.Cdb[6] = 0x01;
```

```
		sptwb.Spt.Cdb[7] = 0x01;
		sptwb.Spt.Cdb[8] = 0x4F;  // SMART_CYL_LOW
		sptwb.Spt.Cdb[9] = 0xC2;  // SMART_CYL_HIGH
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xB0; // SMART_CMD
	}
	else if(type == CMD_TYPE_CYPRESS)
	{
		sptwb.Spt.CdbLength = 16;
		sptwb.Spt.Cdb[0] = 0x24;
		sptwb.Spt.Cdb[1] = 0x24;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0xBE;
		sptwb.Spt.Cdb[4] = 0x01;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = 0xD1;  // READ_THRESHOLD
		sptwb.Spt.Cdb[7] = 0x00;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x4F;  // SMART_CYL_LOW
		sptwb.Spt.Cdb[10] = 0xC2; // SMART_CYL_HIGH
		sptwb.Spt.Cdb[11] = 0xA0;
		sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
		sptwb.Spt.Cdb[13] = 0x00;
		sptwb.Spt.Cdb[14] = 0x00;
		sptwb.Spt.Cdb[15] = 0x00;
	}
	else
	{
		return FALSE;
	}

	length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
	bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
		&sptwb, sizeof(SCSI_PASS_THROUGH),
		&sptwb, length,   &dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	if(bRet == FALSE || dwReturned != length)
	{
		return	FALSE;
	}

	CString str;
	int j = 0;
	for(int i = 0; i < MAX_ATTRIBUTE; i++)
	{
		memcpy(	&(asi->Threshold[i]),
			&(sptwb.DataBuf[i * sizeof(SMART_THRESHOLD) + 2]),
sizeof(SMART_THRESHOLD));

		if(asi->Threshold[j].Id != 0)
		{
			j++;
		}
	}

	return	TRUE;
}

BOOL CTestdisk::ControlSmartStatusSat(INT physicalDriveId, BYTE command,
COMMAND_TYPE type)
{
	BOOL  bRet;
	HANDLE	hIoCtrl;
	DWORD dwReturned;

	SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;
```

```
      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = 0;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);
      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xA1; //ATA Проходів через (12) операцій коду
(A1h)
            sptwb.Spt.Cdb[1] = (3 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=3(Non-
Data),Reserved
            sptwb.Spt.Cdb[2] = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3] = command;//FEATURES (7:0)
            sptwb.Spt.Cdb[4] = 0;//SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5] = 1;//LBA_LOW (7:0)
            sptwb.Spt.Cdb[6] = SMART_CYL_LOW;//LBA_MID (7:0)
            sptwb.Spt.Cdb[7] = SMART_CYL_HI;//LBA_HIGH (7:0)
            sptwb.Spt.Cdb[9] = SMART_CMD;//COMMAND
      }
      else if(type == CMD_TYPE_SUNPLUS)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xF8;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = 0x22;
            sptwb.Spt.Cdb[3] = 0x10;
            sptwb.Spt.Cdb[4] = 0x01;
            sptwb.Spt.Cdb[5] = command;
            sptwb.Spt.Cdb[6] = 0x01;
            sptwb.Spt.Cdb[7] = 0x00;
            sptwb.Spt.Cdb[8] = 0x4F;  // SMART_CYL_LOW
            sptwb.Spt.Cdb[9] = 0xC2;  // SMART_CYL_HIGH
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = 0xB0; // SMART_CMD
      }
      else if(type == CMD_TYPE_IO_DATA)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0] = 0xE3;
            sptwb.Spt.Cdb[1] = 0x00; // ?
            sptwb.Spt.Cdb[2] = command;
            sptwb.Spt.Cdb[3] = 0x00; // ?
            sptwb.Spt.Cdb[4] = 0x00; // ?
            sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
            sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
            sptwb.Spt.Cdb[7] = 0xA0; //
            sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
            sptwb.Spt.Cdb[9] = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
```

```
	else if(type == CMD_TYPE_LOGITEC)
	{
		sptwb.Spt.CdbLength = 10;
		sptwb.Spt.Cdb[0] = 0xE0;
		sptwb.Spt.Cdb[1] = 0x00;
		sptwb.Spt.Cdb[2] = command;
		sptwb.Spt.Cdb[3] = 0x00;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = 0x4F; // SMART_CYL_LOW
		sptwb.Spt.Cdb[6] = 0xC2; // SMART_CYL_HIGH
		sptwb.Spt.Cdb[7] = 0xA0;
		sptwb.Spt.Cdb[8] = 0xB0; // SMART_CMD
		sptwb.Spt.Cdb[9] = 0x4C;
	}
	else if(type == CMD_TYPE_JMICRON)
	{
		sptwb.Spt.CdbLength = 12;
		sptwb.Spt.Cdb[0] = 0xDF;
		sptwb.Spt.Cdb[1] = 0x10;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0x02;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = command;
		sptwb.Spt.Cdb[6] = 0x01;
		sptwb.Spt.Cdb[7] = 0x01;
		sptwb.Spt.Cdb[8] = 0x4F;  // SMART_CYL_LOW
		sptwb.Spt.Cdb[9] = 0xC2;  // SMART_CYL_HIGH
		sptwb.Spt.Cdb[10] = 0xA0;
		sptwb.Spt.Cdb[11] = 0xB0; // SMART_CMD
	}
	else if(type == CMD_TYPE_CYPRESS)
	{
		sptwb.Spt.CdbLength = 16;
		sptwb.Spt.Cdb[0] = 0x24;
		sptwb.Spt.Cdb[1] = 0x24;
		sptwb.Spt.Cdb[2] = 0x00;
		sptwb.Spt.Cdb[3] = 0xBE;
		sptwb.Spt.Cdb[4] = 0x00;
		sptwb.Spt.Cdb[5] = 0x00;
		sptwb.Spt.Cdb[6] = command;
		sptwb.Spt.Cdb[7] = 0x00;
		sptwb.Spt.Cdb[8] = 0x00;
		sptwb.Spt.Cdb[9] = 0x4F;  // SMART_CYL_LOW
		sptwb.Spt.Cdb[10] = 0xC2; // SMART_CYL_HIGH
		sptwb.Spt.Cdb[11] = 0xA0;
		sptwb.Spt.Cdb[12] = 0xB0; // ID_CMD
		sptwb.Spt.Cdb[13] = 0x00;
		sptwb.Spt.Cdb[14] = 0x00;
		sptwb.Spt.Cdb[15] = 0x00;
	}
	else
	{
		return FALSE;
	}

	DWORD length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
	bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
		&sptwb, sizeof(SCSI_PASS_THROUGH),
		&sptwb, length,   &dwReturned, NULL);

	::CloseHandle(hIoCtrl);

	return	bRet;
}

BOOL CTestdisk::SendAtaCommandSat(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, COMMAND_TYPE type)
{
```

```
      BOOL  bRet;
      HANDLE      hIoCtrl;
      DWORD dwReturned;

      SCSI_PASS_THROUGH_WITH_BUFFERS sptwb;

      hIoCtrl = GetIoCtrlHandle(physicalDriveId);
      if(hIoCtrl == INVALID_HANDLE_VALUE)
      {
            return      FALSE;
      }

      ::ZeroMemory(&sptwb,sizeof(SCSI_PASS_THROUGH_WITH_BUFFERS));

    sptwb.Spt.Length = sizeof(SCSI_PASS_THROUGH);
    sptwb.Spt.PathId = 0;
    sptwb.Spt.TargetId = 0;
    sptwb.Spt.Lun = 0;
    sptwb.Spt.SenseInfoLength = 24;
    sptwb.Spt.DataIn = SCSI_IOCTL_DATA_IN;
    sptwb.Spt.DataTransferLength = 0;
    sptwb.Spt.TimeOutValue = 2;
    sptwb.Spt.DataBufferOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
DataBuf);
    sptwb.Spt.SenseInfoOffset = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS,
SenseBuf);
      if(type == CMD_TYPE_SAT)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xA1; //ATA Проходів через (12) операцій коду
(A1h)
            sptwb.Spt.Cdb[1]  = (3 << 1) | 0; //MULTIPLE_COUNT=0,PROTOCOL=3(Non-
Data),Reserved
            sptwb.Spt.Cdb[2]  = (1 << 3) | (1 << 2) |
2;//OFF_LINE=0,CK_COND=0,Reserved=0,T_DIR=1(ToDevice),BYTE_BLOCK=1,T_LENGTH=2
            sptwb.Spt.Cdb[3]  = sub;            //FEATURES (7:0)
            sptwb.Spt.Cdb[4]  = param;          //SECTOR_COUNT (7:0)
            sptwb.Spt.Cdb[5]  = 0x00;           //LBA_LOW (7:0)
            sptwb.Spt.Cdb[6]  = 0x00;           //LBA_MID (7:0)
            sptwb.Spt.Cdb[7]  = 0x00;           //LBA_HIGH (7:0)
            sptwb.Spt.Cdb[8]  = 0xA0;           //ПРИСТРІЙ_HEAD
            sptwb.Spt.Cdb[9]  = main;           //COMMAND
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_SUNPLUS)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xF8;
            sptwb.Spt.Cdb[1]  = 0x00;
            sptwb.Spt.Cdb[2]  = 0x22;
            sptwb.Spt.Cdb[3]  = 0x10;
            sptwb.Spt.Cdb[4]  = 0x01;
            sptwb.Spt.Cdb[5]  = sub;
            sptwb.Spt.Cdb[6]  = param;
            sptwb.Spt.Cdb[7]  = 0x00;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = main;
      }
      else if(type == CMD_TYPE_IO_DATA)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xE3;
            sptwb.Spt.Cdb[1]  = 0x00;
            sptwb.Spt.Cdb[2]  = sub;
            sptwb.Spt.Cdb[3]  = param;
            sptwb.Spt.Cdb[4]  = 0x00;
```

```
            sptwb.Spt.Cdb[5]  = 0x00;
            sptwb.Spt.Cdb[6]  = 0x00;
            sptwb.Spt.Cdb[7]  = 0xA0;
            sptwb.Spt.Cdb[8]  = main;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0x00;
      }
      else if(type == CMD_TYPE_LOGITEC)
      {
            sptwb.Spt.CdbLength = 10;
            sptwb.Spt.Cdb[0] = 0xE0;
            sptwb.Spt.Cdb[1] = 0x00;
            sptwb.Spt.Cdb[2] = sub;
            sptwb.Spt.Cdb[3] = param;
            sptwb.Spt.Cdb[4] = 0x00;
            sptwb.Spt.Cdb[5] = 0x00;
            sptwb.Spt.Cdb[6] = 0x00;
            sptwb.Spt.Cdb[7] = 0xA0;
            sptwb.Spt.Cdb[8] = main;
            sptwb.Spt.Cdb[9] = 0x4C;            // ?
      }
      else if(type == CMD_TYPE_JMICRON)
      {
            sptwb.Spt.CdbLength = 12;
            sptwb.Spt.Cdb[0]  = 0xDF;
            sptwb.Spt.Cdb[1]  = 0x10;
            sptwb.Spt.Cdb[2]  = 0x00;
            sptwb.Spt.Cdb[3]  = 0x02;
            sptwb.Spt.Cdb[4]  = 0x00;
            sptwb.Spt.Cdb[5]  = sub;
            sptwb.Spt.Cdb[6]  = param;
            sptwb.Spt.Cdb[7]  = 0x00;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0xA0;
            sptwb.Spt.Cdb[11] = main;
      }
      else if(type == CMD_TYPE_CYPRESS)
      {
            sptwb.Spt.CdbLength = 16;
            sptwb.Spt.Cdb[0]  = 0x24;
            sptwb.Spt.Cdb[1]  = 0x24;
            sptwb.Spt.Cdb[2]  = 0x00;
            sptwb.Spt.Cdb[3]  = 0xBE;
            sptwb.Spt.Cdb[4]  = 0x00;
            sptwb.Spt.Cdb[5]  = 0x00;
            sptwb.Spt.Cdb[6]  = sub;
            sptwb.Spt.Cdb[7]  = param;
            sptwb.Spt.Cdb[8]  = 0x00;
            sptwb.Spt.Cdb[9]  = 0x00;
            sptwb.Spt.Cdb[10] = 0x00;
            sptwb.Spt.Cdb[11] = 0xA0;
            sptwb.Spt.Cdb[12] = main;
            sptwb.Spt.Cdb[13] = 0x00;
            sptwb.Spt.Cdb[14] = 0x00;
            sptwb.Spt.Cdb[15] = 0x00;
      }
      else
      {
            return FALSE;
      }

      DWORD length = offsetof(SCSI_PASS_THROUGH_WITH_BUFFERS, DataBuf) +
sptwb.Spt.DataTransferLength;
      bRet = ::DeviceIoControl(hIoCtrl, IOCTL_SCSI_PASS_THROUGH,
            &sptwb, sizeof(SCSI_PASS_THROUGH),
            &sptwb, length,   &dwReturned, NULL);
```

```
	::CloseHandle(hIoCtrl);

	return	bRet;
}

/*-----------------------------------------------------------------------------*/
// Підтримка функцій
/*-----------------------------------------------------------------------------*/

DWORD CTestdisk::CheckDiskStatus(SMART_ATTRIBUTE* attribute, SMART_THRESHOLD*
threshold, DWORD attributeCount, DWORD vendorId)
{
	int error = 0;
	int caution = 0;
	BOOL flagUnknown = TRUE;

	for(DWORD j = 0; j < attributeCount; j++)
	{
		if( attribute[j].Id != 0xBE // Температура воздуха
		&&	threshold[j].ThresholdValue != 0
		&&	attribute[j].CurrentValue <= threshold[j].ThresholdValue)
		{
			error++;
		}

		switch(attribute[j].Id)
		{
		case 0x05: // Кількість перепризначених секторів
//		case 0xC4: // Кількість операцій перепризначення (ремаппінгу).
		case 0xC5: // Поточна кількість нестабільних секторів
		case 0xC6: // Кількість нескоректованих помилок
			if(attribute[j].RawValue[0] == 0xFF
			&& attribute[j].RawValue[1] == 0xFF
			&& attribute[j].RawValue[2] == 0xFF
			&& attribute[j].RawValue[3] == 0xFF)
			{
				flagUnknown = FALSE;
			}
			else
			{
				caution += attribute[j].RawValue[0]
						+ attribute[j].RawValue[1];
				flagUnknown = FALSE;
			}
			break;
		case 0xBB: // Визначення фірми-розробника
			if(vendorId == VENDOR_MTRON)
			{
				if(attribute[j].CurrentValue == 0)
				{
					error = 1;
				}
				else if(attribute[j].CurrentValue < 10)
				{
					caution = 1;
				}
				else
				{
					flagUnknown = FALSE;
				}
			}
			break;
		default:
			break;
		}
	}

	if(error > 0)
	{
```

```
		return DISK_STATUS_BAD;
	}
	else if(flagUnknown)
	{
		return DISK_STATUS_UNKNOWN;
	}
	else if(caution > 0)
	{
		return DISK_STATUS_CAUTION;
	}
	else
	{
		return DISK_STATUS_GOOD;
	}
}

VOID CTestdisk::ChangeByteOrder(PCHAR str, DWORD length)
{
	CHAR  temp;
	for(DWORD i = 0; i < length; i += 2)
	{
		temp = str[i];
		str[i] = str[i+1];
		str[i+1] = temp;
	}
}

BOOL CTestdisk::CheckAsciiStringError(PCHAR str, DWORD length)
{
	BOOL flag = FALSE;
	for(DWORD i = 0; i < length; i++)
	{
		if((0x00 < str[i] && str[i] < 0x20) || str[i] >= 0x7f)
		{
			flag = TRUE;
			break;
		}
	}
	return flag;
}

DWORD CTestdisk::GetPowerOnHours(DWORD rawValue, DWORD timeUnitType)
{
	switch(timeUnitType)
	{
	case POWER_ON_UNKNOWN:
		return 0;
		break;
	case POWER_ON_HOURS:
		return rawValue;
		break;
	case POWER_ON_MINUTES:
		return rawValue / 60;
		break;
	case POWER_ON_HALF_MINUTES:
		return rawValue / 120;
		break;
	case POWER_ON_SECONDS:
		return rawValue / 60 / 60;
		break;
	default:
		return rawValue;
		break;
	}
}

DWORD CTestdisk::GetPowerOnHoursEx(DWORD i, DWORD timeUnitType)
{
	DWORD rawValue = vars[i].PowerOnRawValue;
```

```
	switch(timeUnitType)
	{
	case POWER_ON_UNKNOWN:
		return 0;
		break;
	case POWER_ON_HOURS:
		return rawValue;
		break;
	case POWER_ON_MINUTES:
		return rawValue / 60;
		break;
	case POWER_ON_HALF_MINUTES:
		return rawValue / 120;
		break;
	case POWER_ON_SECONDS:
		return rawValue / 60 / 60;
		break;
	default:
		return rawValue;
		break;
	}
}

DWORD CTestdisk::GetTransferMode(WORD w63, WORD w76, WORD w88, CString &current,
CString &max, CString &type, INTERFACE_TYPE* interfaceType)
{
	DWORD tm = TRANSFER_MODE_PIO;
	current = max = _T("");
	type = _T("Parallel ATA");
	*interfaceType = INTERFACE_TYPE_PATA;

	// Слово DMA або PIO
	if(w63 & 0x0700)
	{
		tm = TRANSFER_MODE_PIO_DMA;
		current = max = _T("PIO / DMA");
	}

	// Ultara DMA Максимальний режим передачі
		if(w88 & 0x0040){tm = TRANSFER_MODE_ULTRA_DMA_133; max = _T("Ultra
DMA/133");}
	else if(w88 & 0x0020){tm = TRANSFER_MODE_ULTRA_DMA_100; max = _T("Ultra
DMA/100");}
	else if(w88 & 0x0010){tm = TRANSFER_MODE_ULTRA_DMA_66;  max = _T("Ultra
DMA/66");}
	else if(w88 & 0x0008){tm = TRANSFER_MODE_ULTRA_DMA_44;  max = _T("Ultra
DMA/44");}
	else if(w88 & 0x0004){tm = TRANSFER_MODE_ULTRA_DMA_33;  max = _T("Ultra
DMA/33");}
	else if(w88 & 0x0002){tm = TRANSFER_MODE_ULTRA_DMA_25;  max = _T("Ultra
DMA/25");}
	else if(w88 & 0x0001){tm = TRANSFER_MODE_ULTRA_DMA_16;  max = _T("Ultra
DMA/16");}

	// Ultara DMA поточний режим передачі
		if(w88 & 0x4000){current = _T("Ultra DMA/133");}
	else if(w88 & 0x2000){current = _T("Ultra DMA/100");}
	else if(w88 & 0x1000){current = _T("Ultra DMA/66");}
	else if(w88 & 0x0800){current = _T("Ultra DMA/44");}
	else if(w88 & 0x0400){current = _T("Ultra DMA/33");}
	else if(w88 & 0x0200){current = _T("Ultra DMA/25");}
	else if(w88 & 0x0100){current = _T("Ultra DMA/16");}

	// Serial ATA
	if(w76 != 0x0000 && w76 != 0xFFFF)
	{
		current = max = _T("SATA/150");
		type = _T("Serial ATA");
		*interfaceType = INTERFACE_TYPE_SATA;
```

```
	}

		if(w76 & 0x0010){tm = TRANSFER_MODE_UNKNOWN;  current = max =
_T("Unknown");}
	else if(w76 & 0x0008){tm = TRANSFER_MODE_SATA_600; current = max =
_T("SATA/600");}
	else if(w76 & 0x0004){tm = TRANSFER_MODE_SATA_300; current = max =
_T("SATA/300");}
	else if(w76 & 0x0002){tm = TRANSFER_MODE_SATA_150; current = max =
_T("SATA/150");}

	return tm;
}

DWORD CTestdisk::GetTimeUnitType(CString model, CString firmware, DWORD major,
DWORD transferMode)
{
	model.MakeUpper();

	if(model.Find(_T("FUJITSU")) == 0)
	{
		if(major >= 8)
		{
			return POWER_ON_HOURS;
		}
		else
		{
			return POWER_ON_SECONDS;
		}
	}
	else if(model.Find(_T("HITACHI_DK")) == 0)
	{
		return POWER_ON_MINUTES;
	}
	else if(model.Find(_T("MAXTOR")) == 0)
	{
		if(transferMode >= TRANSFER_MODE_SATA_300
		|| model.Find(_T("MAXTOR 6H")) == 0	// Maxtor DiamondMax 11
сімейство
		|| model.Find(_T("MAXTOR 7H500")) == 0	// Maxtor MaXLine Pro 500
сімейство
		|| model.Find(_T("MAXTOR 6L0")) == 0	// Maxtor DiamondMax Plus
D740X сімейство
		|| model.Find(_T("MAXTOR 4K")) == 0	// Maxtor DiamondMax D540X-
4K сімейство
		)
		{
			return POWER_ON_HOURS;
		}
		else
		{
			return POWER_ON_MINUTES;
		}
	}
	else if(model.Find(_T("SAMSUNG")) == 0)
	{
		if(transferMode >= TRANSFER_MODE_SATA_300)
		{
			return POWER_ON_HOURS;
		}
		else if(-23 >= _tstoi(firmware.Right(3)) &&
_tstoi(firmware.Right(3)) >= -39)
		{
			return POWER_ON_HALF_MINUTES;
		}
		else if(model.Find(_T("SAMSUNG SV")) == 0
		||		model.Find(_T("SAMSUNG SP")) == 0
		||		model.Find(_T("SAMSUNG HM")) == 0
		)
```

```
            {
                  return POWER_ON_HALF_MINUTES;
            }
            else
            {
                  return POWER_ON_HOURS;
            }
      }
      else
      {
            return POWER_ON_HOURS;
      }
}

DWORD CTestdisk::GetAtaMajorVersion(WORD w80, CString &majorVersion)
{
      DWORD major = 0;

      if(w80 == 0x0000 || w80 == 0xFFFF)
      {
            return FALSE;
      }

      for(int i = 14; i > 0; i--)
      {
            if((w80 >> i) & 0x1)
            {
                  major = i;
                  break;
            }
      }

      if(major == 15)
      {
            majorVersion = _T("");
      }
      else if(major >= 8)
      {
            majorVersion.Format(_T("ATA%d-ACS"), major);
      }
      else if(major >= 4)
      {
            majorVersion.Format(_T("ATA/ATAPI-%d"), major);
      }
      else if(major == 0)
      {
            majorVersion = _T("");
      }
      else
      {
            majorVersion.Format(_T("ATA-%d"), major);
      }

      return major;
}
```

Testdisk.h – бібліотека для файлу Testdisk.cpp

```
#pragma once

#include "winioctl.h"
#include "SPTIUtil.h"


class CTestdisk
{
public:
	static const int MAX_DISK = 32;// FIX
	static const int MAX_ATTRIBUTE = 30; // FIX
	static const int MAX_SEARCH_PHYSICAL_DRIVE = 32;
	static const int MAX_SEARCH_SCSI_PORT = 16;
	static const int MAX_SEARCH_SCSI_TARGET_ID = 8;

	static const int SCSI_MINIPORT_BUFFER_SIZE = 512;

public:
	CTestdisk();
	virtual ~CTestdisk();

	enum SMART_STATUS
	{
		SMART_STATUS_NO_CHANGE = 0,
		SMART_STATUS_MINOR_CHANGE,
		SMART_STATUS_MAJOR_CHANGE
	};

	enum TRANSFER_MODE
	{
		TRANSFER_MODE_UNKNOWN = 0,
		TRANSFER_MODE_PIO,
		TRANSFER_MODE_PIO_DMA,
		TRANSFER_MODE_ULTRA_DMA_16,
		TRANSFER_MODE_ULTRA_DMA_25,
		TRANSFER_MODE_ULTRA_DMA_33,
		TRANSFER_MODE_ULTRA_DMA_44,
		TRANSFER_MODE_ULTRA_DMA_66,
		TRANSFER_MODE_ULTRA_DMA_100,
		TRANSFER_MODE_ULTRA_DMA_133,
		TRANSFER_MODE_SATA_150,
		TRANSFER_MODE_SATA_300,
		TRANSFER_MODE_SATA_600
	};

	enum DISK_STATUS
	{
		DISK_STATUS_UNKNOWN = 0,
		DISK_STATUS_GOOD,
		DISK_STATUS_CAUTION,
		DISK_STATUS_BAD
	};

	enum POWER_ON_HOURS_UNIT
	{
		POWER_ON_UNKNOWN = 0,
		POWER_ON_HOURS,
		POWER_ON_MINUTES,
		POWER_ON_HALF_MINUTES,
		POWER_ON_SECONDS,
	};

	enum COMMAND_TYPE
	{
		CMD_TYPE_PHYSICAL_DRIVE = 0,
```

```
            CMD_TYPE_SCSI_MINIPORT,
            CMD_TYPE_SAT,                        // SAT = SCSI_ATA_TRANSLATION
            CMD_TYPE_SUNPLUS,
            CMD_TYPE_IO_DATA,
            CMD_TYPE_LOGITEC,
            CMD_TYPE_JMICRON,
            CMD_TYPE_CYPRESS,
            CMD_TYPE_PROLIFIC,
            CMD_TYPE_DEBUG
      };

      enum VENDOR_ID
      {
            VENDOR_UNKNOWN      = 0x0000,
            VENDOR_MTRON    = 0x0001,

            USB_VENDOR_BUFFALO  = 0x0411,
            USB_VENDOR_IO_DATA  = 0x04BB,
            USB_VENDOR_LOGITEC  = 0x0789,
            USB_VENDOR_INITIO   = 0x13FD,
            USB_VENDOR_SUNPLUS  = 0x04FC,
            USB_VENDOR_JMICRON  = 0x152D,
            USB_VENDOR_CYPRESS  = 0x04B4,
            USB_VENDOR_OXFORD   = 0x0928,
            USB_VENDOR_PROLIFIC = 0x067B,

      };

      enum INTERFACE_TYPE
      {
            INTERFACE_TYPE_UNKNOWN = 0,
            INTERFACE_TYPE_PATA,
            INTERFACE_TYPE_SATA,
            INTERFACE_TYPE_USB,
            INTERFACE_TYPE_IEEE1394
      };

protected:
      enum IO_CONTROL_CODE
      {
            DFP_SEND_DRIVE_COMMAND  = 0x0007C084,
            DFP_RECEIVE_DRIVE_DATA  = 0x0007C088,
            IOCTL_SCSI_MINIPORT     = 0x0004D008,
            IOCTL_IDE_PASS_THROUGH  = 0x0004D028, // 2000 або пізніша версія
            IOCTL_ATA_PASS_THROUGH  = 0x0004D02C, // XP SP2 and 2003 або пізніша
версія
      };

#pragma pack(push,1)

      typedef     struct _IDENTIFY_DRIVE_OUTDATA
      {
            SENDCMDOUTPARAMS  SendCmdOutParam;
            BYTE                    Data[IDENTIFY_BUFFER_SIZE - 1];
      } IDENTIFY_DRIVE_OUTDATA, *PIDENTIFY_DRIVE_OUTDATA;

      typedef     struct _SMART_READ_DATA_OUTDATA
      {
            SENDCMDOUTPARAMS  SendCmdOutParam;
            BYTE                    Data[READ_ATTRIBUTE_BUFFER_SIZE - 1];
      } SMART_READ_DATA_OUTDATA, *PSMART_READ_DATA_OUTDATA;

      typedef struct _CMD_IDE_PATH_THROUGH
      {
            IDEREGS     reg;
            DWORD   length;
            BYTE    buffer[1];
      } CMD_IDE_PATH_THROUGH, *PCMD_IDE_PATH_THROUGH;
```

```
static const int ATA_FLAGS_DRDY_REQUIRED = 0x01;
static const int ATA_FLAGS_DATA_IN       = 0x02;
static const int ATA_FLAGS_DATA_OUT      = 0x04;
static const int ATA_FLAGS_48BIT_COMMAND = 0x08;

typedef struct _ATA_PASS_THROUGH_EX
{
      WORD    Length;
      WORD    AtaFlags;
      BYTE    PathId;
      BYTE    TargetId;
      BYTE    Lun;
      BYTE    ReservedAsUchar;
      DWORD   DataTransferLength;
      DWORD   TimeOutValue;
      DWORD   ReservedAsUlong;
      DWORD_PTR   DataBufferOffset;
      IDEREGS PreviousTaskFile;
      IDEREGS CurrentTaskFile;
} ATA_PASS_THROUGH_EX, *PCMD_ATA_PASS_THROUGH_EX;

typedef struct
{
      ATA_PASS_THROUGH_EX Apt;
      BYTE  Buf[512];
} ATA_PASS_THROUGH_EX_WITH_BUFFERS;

typedef     struct SMART_ATTRIBUTE
{
      BYTE  Id;
      WORD  StatusFlags;
      BYTE  CurrentValue;
      BYTE  WorstValue;
      BYTE  RawValue[6];
      BYTE  Reserved;
};

typedef     struct SMART_THRESHOLD
{
      BYTE  Id;
      BYTE  ThresholdValue;
      BYTE  Reserved[10];
};

typedef struct SRB_IO_CONTROL
{
   ULONG    HeaderLength;
   UCHAR    Signature[8];
   ULONG    Timeout;
   ULONG    ControlCode;
   ULONG    ReturnCode;
   ULONG    Length;
};

typedef struct SRB_IO_COMMAND
{
      SRB_IO_CONTROL    Cntrol;
      IDEREGS                 IdeRegs;
      BYTE              Data[512];
};

struct IDENTIFY_DRIVE
{
      WORD        GeneralConfiguration;                   //0
      WORD        LogicalCylinders;                       //1
Застарівший
      WORD        SpecificConfiguration;                  //2
      WORD        LlogicalHeads;                                //3
Застарівший
```

```
            WORD        Retired1[2];                                    //4-5
            WORD        LogicalSectors;                                 //6
Застарівший
            DWORD       ReservedForCompactFlash;                //7-8
            WORD        Retired2;                                       //9
            CHAR        SerialNumber[20];                       //10-19
            WORD        Retired3;                                       //20
            WORD        BufferSize;                                     //21
Застарівший
            WORD        Obsolute4;                                      //22
            CHAR        FirmwareRev[8];                                 //23-26
            CHAR        Model[40];                                      //27-46
            WORD        MaxNumPerInterupt;                              //47
            WORD        Reserved1;                                      //48
            WORD        Capabilities1;                                  //49
            WORD        Capabilities2;                                  //50
            DWORD       Obsolute5;                                      //51-52
            WORD        Field88and7064;                                 //53
            WORD        Obsolute6[5];                                   //54-58
            WORD        MultSectorStuff;                        //59
            DWORD       TotalAddressableSectors;                //60-61
            WORD        Obsolute7;                                      //62
            WORD        MultiWordDma;                                   //63
            WORD        PioMode;                                        //64
            WORD        MinMultiwordDmaCycleTime;               //65
            WORD        RecommendedMultiwordDmaCycleTime;   //66
            WORD        MinPioCycleTimewoFlowCtrl;              //67
            WORD        MinPioCycleTimeWithFlowCtrl;        //68
            WORD        Reserved2[6];                                   //69-74
            WORD        QueueDepth;                                     //75
            WORD        SerialAtaCapabilities;                  //76
            WORD        ReservedForFutureSerialAta;             //77
            WORD        SerialAtaFeaturesSupported;             //78
            WORD        SerialAtaFeaturesEnabled;               //79
            WORD        MajorVersion;                                   //80
            WORD        MinorVersion;                                   //81
            WORD        CommandSetSupported1;                   //82
            WORD        CommandSetSupported2;                   //83
            WORD        CommandSetSupported3;                   //84
            WORD        CommandSetEnabled1;                             //85
            WORD        CommandSetEnabled2;                             //86
            WORD        CommandSetDefault;                              //87
            WORD        UltraDmaMode;                                   //88
            WORD        TimeReqForSecurityErase;                //89
            WORD        TimeReqForEnhancedSecure;               //90
            WORD        CurrentPowerManagement;                 //91
            WORD        MasterPasswordRevision;                 //92
            WORD        HardwareResetResult;                    //93
            WORD        AcoustricManagement;                    //94
            WORD        StreamMinRequestSize;                   //95
            WORD        StreamingTimeDma;                       //96
            WORD        StreamingAccessLatency;                 //97
            DWORD       StreamingPerformance;                   //98-99
            ULONGLONG   MaxUserLba;                                     //100-103
            WORD        StremingTimePio;                        //104
            WORD        Reserved3;                                      //105
            WORD        SectorSize;                                     //106
            WORD        InterSeekDelay;                                 //107
            WORD        IeeeOui;                                        //108
            WORD        UniqueId3;                                      //109
            WORD        UniqueId2;                                      //110
            WORD        UniqueId1;                                      //111
            WORD        Reserved4[4];                                   //112-115
            WORD        Reserved5;                                      //116
            DWORD       WordsPerLogicalSector;                  //117-118
            WORD        Reserved6[8];                                   //119-126
            WORD        RemovableMediaStatus;                   //127
            WORD        SecurityStatus;                                 //128
            WORD        VendorSpecific[31];                             //129-159
```

```
            WORD        CfaPowerMode1;                                  //160
            WORD        Reserved7[15];                                  //161-175
            CHAR        CurrentMediaSerialNo[60];               //176-205
            WORD        SctCommandTransport;                    //206  254
            WORD        ReservedForCeAta1[2];                   //207-208
            WORD        AlignmentOfLogicalBlocks;               //209
            DWORD       WriteReadVerifySectorCountMode3;    //210-211
            DWORD       WriteReadVerifySectorCountMode2;    //212-213
            WORD        NvCacheCapabilities;                    //214
            DWORD       NvCacheSizeLogicalBlocks;               //215-216
            WORD        NominalMediaRotationRate;               //217
            WORD        Reserved8;                                      //218
            WORD        NvCacheOptions1;                        //219
            WORD        NvCacheOptions2;                        //220
            WORD        Reserved9;                                      //221
            WORD        TransportMajorVersionNumber;        //222
            WORD        TransportMinorVersionNumber;        //223
            WORD        ReservedForCeAta2[10];                  //224-233
            WORD        MinimumBlocksPerDownloadMicrocode;  //234
            WORD        MaximumBlocksPerDownloadMicrocode;  //235
            WORD        Reserved10[19];                                 //236-254
            WORD        IntegrityWord;                                  //255
      };
#pragma     pack(pop)

public:
      DWORD UpdateSmartInfo(DWORD index);
      BOOL UpdateIdInfo(DWORD index);
      BYTE GetAamValue(DWORD index);
      BYTE GetApmValue(DWORD index);
      BOOL EnableAam(DWORD index, BYTE param);
      BOOL EnableApm(DWORD index, BYTE param);
      BOOL DisableAam(DWORD index);
      BOOL DisableApm(DWORD index);
      BYTE GetRecommendAamValue(DWORD index);
      BYTE GetRecommendApmValue(DWORD index);

      BOOL Init(BOOL useWmi, BOOL advancedDiskSearch, PBOOL flagChangeDisk);
      BOOL MeasuredTimeUnit();
      DWORD GetPowerOnHours(DWORD rawValue, DWORD timeUnitType);
      DWORD GetPowerOnHoursEx(DWORD index, DWORD timeUnitType);

      struct DISK_POSITION
      {
            INT                         PhysicalDriveId;
            INT                         ScsiPort;
            INT                         ScsiTargetId;
      };

      struct ATA_SMART_INFO
      {
            IDENTIFY_DRIVE          IdentifyDevice;
            SMART_ATTRIBUTE         Attribute[MAX_ATTRIBUTE];
            SMART_THRESHOLD         Threshold[MAX_ATTRIBUTE];

            BOOL                    IsSmartEnabled;
            BOOL                    IsCheckSumError;
            BOOL                    IsWord88;
            BOOL                    IsWord64_76;

            BOOL                    IsSmartSupported;
            BOOL                    IsLba48Supported;
            BOOL                    IsAamSupported;
            BOOL                    IsApmSupported;
            BOOL                    IsAamEnabled;
            BOOL                    IsApmEnabled;
            BOOL                    IsNcqSupported;
            BOOL                    IsNvCacheSupported;
            BOOL                    IsMaxtorMinute;
```

```
		INT			PhysicalDriveId;
		INT			ScsiPort;
		INT			ScsiTargetId;
//		INT			AccessType;

		DWORD			TotalDiskSize;
		DWORD			Cylinder;
		DWORD			Head;
		DWORD			Sector;
		DWORD			Sector28;
		ULONGLONG		Sector48;
		DWORD			DiskSizeChs;
		DWORD			DiskSizeLba28;
		DWORD			DiskSizeLba48;
		DWORD			BufferSize;
		ULONGLONG		NvCacheSize;
		DWORD			TransferModeType;
		DWORD			DetectedTimeUnitType;
		DWORD			MeasuredTimeUnitType;
		DWORD			AttributeCount;
		INT			DetectedPowerOnHours;
		INT			MeasuredPowerOnHours;
		INT			PowerOnRawValue;
		INT			PowerOnStartRawValue;
		DWORD			PowerOnCount;
		DWORD			Temperature;
		double			Speed;

		INT			Life;

		DWORD			Major;
		DWORD			Minor;

		DWORD			DiskStatus;
		DWORD			DriveLetterMap;
		//
		DWORD			AlarmTemperature;
		BOOL			AlarmHealthStatus;

		INTERFACE_TYPE		InterfaceType;
		COMMAND_TYPE		CommandType;

		DWORD			VendorId;
		DWORD			ProductId;

		CString			SerialNumber;
		CString			SerialNumberReverse;
		CString			FirmwareRev;
		CString			FirmwareRevReverse;
		CString			Model;
		CString			ModelReverse;
		CString			ModelWmi;
		CString			ModelSerial;
		CString			DriveMap;
		CString			MaxTransferMode;
		CString			CurrentTransferMode;
		CString			MajorVersion;
		CString			MinorVersion;
		CString			Interface;
		CString			Enclosure;
		CString			CommandTypeString;
	};

	struct EXTERNAL_DISK_INFO
	{
		CString Enclosure;
		DWORD VendorId;
		DWORD ProductId;
```

```
	};

	CArray<ATA_SMART_INFO, ATA_SMART_INFO> vars;
	CArray<EXTERNAL_DISK_INFO, EXTERNAL_DISK_INFO> externals;

	CStringArray m_IdeController;
	CStringArray m_ScsiController;
	CStringArray m_UsbController;
	CString m_ControllerMap;

	BOOL IsEnabledWmi;
	DWORD MeasuredGetTickCount;

protected:
	OSVERSIONINFOEX m_Os;
	CString m_SerialNumberA_Z[26];
	BOOL m_FlagAtaPassThrough;

	BOOL GetMain(INT physicalDriveId, INT scsiPort, INT scsiTargetId,
INTERFACE_TYPE interfaceType, VENDOR_ID vendorId);
	BOOL AddDisk(INT PhysicalDriveId, INT ScsiPort, INT scsiTargetId,
COMMAND_TYPE commandType, IDENTIFY_DRIVE* identify);
	DWORD CheckSmartAttributeUpdate(DWORD index, SMART_ATTRIBUTE* pre,
SMART_ATTRIBUTE* cur);

	VOID InitAtaInfo();
	VOID InitAtaInfoByWmi();
	VOID InitStruct();
	VOID ChangeByteOrder(PCHAR str, DWORD length);
	BOOL CheckAsciiStringError(PCHAR str, DWORD length);
	HANDLE GetIoCtrlHandle(BYTE index);
	BOOL SendAtaCommand(DWORD i, BYTE main, BYTE sub, BYTE param);

	BOOL DoIdentifyDevicePd(INT physicalDriveId, IDENTIFY_DRIVE* identify);
	BOOL GetSmartAttributePd(INT physicalDriveId, ATA_SMART_INFO* asi);
	BOOL GetSmartThresholdPd(INT physicalDriveId, ATA_SMART_INFO* asi);
	BOOL ControlSmartStatusPd(INT physicalDriveId, BYTE command);
	BOOL SendAtaCommandPd(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, PBYTE data, DWORD dataSize);

	BOOL DoIdentifyDeviceScsi(INT scsiPort, INT scsiTargetId, IDENTIFY_DRIVE*
identify);
	BOOL GetSmartAttributeScsi(INT scsiPort, INT scsiTargetId, ATA_SMART_INFO*
asi);
	BOOL GetSmartThresholdScsi(INT scsiPort, INT scsiTargetId, ATA_SMART_INFO*
asi);
	BOOL ControlSmartStatusScsi(INT scsiPort, INT scsiTargetId, BYTE command);
	BOOL SendAtaCommandScsi(INT scsiPort, INT scsiTargetId, BYTE main, BYTE
sub, BYTE param);

	BOOL DoIdentifyDeviceSat(INT physicalDriveId, IDENTIFY_DRIVE* identify,
COMMAND_TYPE commandType);
	BOOL GetSmartAttributeSat(INT physicalDriveId, ATA_SMART_INFO* asi);
	BOOL GetSmartThresholdSat(INT physicalDriveId, ATA_SMART_INFO* asi);
	BOOL ControlSmartStatusSat(INT physicalDriveId, BYTE command, COMMAND_TYPE
commandType);
	BOOL SendAtaCommandSat(INT physicalDriveId, BYTE main, BYTE sub, BYTE
param, COMMAND_TYPE commandType);
	DWORD CheckDiskStatus(SMART_ATTRIBUTE* attribute, SMART_THRESHOLD*
threshold, DWORD attributeCount, DWORD vendorId);
	DWORD GetTransferMode(WORD w63, WORD w76, WORD w88, CString
&currentTransferMode, CString &maxTransferMode, CString &Interface,
INTERFACE_TYPE *interfaceType);
	DWORD GetTimeUnitType(CString model, CString firmware, DWORD major, DWORD
transferMode);
	DWORD GetAtaMajorVersion(WORD w80, CString &majorVersion);
//	DWORD GetMaxtorPowerOnHours(DWORD currentValue, DWORD rawValue);
	static int Compare(const void *p1, const void *p2);
};
```

AboutDlg.cpp – довідка

```
#include "stdafx.h"
#include "Main.h"
#include "AboutDlg.h"

IMPLEMENT_DYNCREATE(CAboutDlg, CDHtmlDialog)

CAboutDlg::CAboutDlg(CWnd* pParent /*=NULL*/)
                  : CDHtmlDialogEx(CAboutDlg::IDD, CAboutDlg::IDH, pParent)
{

}

CAboutDlg::~CAboutDlg()
{
}

BOOL CAboutDlg::OnInitDialog()
{
                  CDHtmlDialogEx::OnInitDialog();

                  InitDHtmlDialog(SIZE_X, SIZE_Y, ((CMainApp*)AfxGetApp())-
>m_AboutDlgPath);

                  return TRUE;
}

void CAboutDlg::OnDocumentComplete(LPDISPATCH pDisp, LPCTSTR szUrl)
{
                  CString cstr;
                  cstr = szUrl;
                  if(cstr.Find(_T("html")) != -1 || cstr.Find(_T("dlg")) != -1)
                  {
                        m_FlagShowWindow = TRUE;
                       m_Copyright = PRODUCT_COPYRIGHT;
                        UpdateData(FALSE);
                        ShowWindow(SW_SHOW);
                  }
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDHtmlDialogEx)
END_MESSAGE_MAP()

BEGIN_DHTML_EVENT_MAP(CAboutDlg)
                  DHTML_EVENT_ONCLICK(_T("CrystalDewWorld"), OnCrystalDewWorld)
END_DHTML_EVENT_MAP()

HRESULT CAboutDlg::OnCrystalDewWorld(IHTMLElement* /*pElement*/)
{
                  if(GetUserDefaultLCID() == 0x0411)//
                  {
                        ShellExecute(NULL, NULL, URL_CRYSTAL_DEW_WORLD_JA, NULL,
NULL, SW_SHOWNORMAL);
                  }
                  else // Other Language
                  {
                        ShellExecute(NULL, NULL, URL_CRYSTAL_DEW_WORLD_EN, NULL,
NULL, SW_SHOWNORMAL);
                  }

                  return S_FALSE;
}
```

AboutDlg.h – бібліотека для файлу AboutDlg.cpp

```
#pragma once

class CAboutDlg : public CDHtmlDialogEx
{
	DECLARE_DYNCREATE(CAboutDlg)

	static const int SIZE_X = 240;
	static const int SIZE_Y = 200;

public:
	CAboutDlg(CWnd* pParent = NULL);
	virtual ~CAboutDlg();

	enum { IDD = IDD_ABOUT, IDH = IDR_HTML_DUMMY };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);
	virtual BOOL OnInitDialog();
	virtual void OnDocumentComplete(LPDISPATCH pDisp, LPCTSTR szUrl);

	CString m_Version;
	CString m_Release;
	CString m_Copyright;

	HRESULT OnCrystalDewWorld(IHTMLElement *pElement);

	DECLARE_MESSAGE_MAP()
	DECLARE_DHTML_EVENT_MAP()
};
```