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 аналізу працездатності жорсткого диску ПК”

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

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

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

Форостяному Олександру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. 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 аналізу працездатності жорсткого диску ПК*

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

Форостяний О.А. Дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК. 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 аналізу працездатності жорсткого диску ПК.

Метою розробки є дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК.

Об'єктом дослідження є процес аналізу працездатності жорсткого диску ПК.

Предметом дослідження є методи аналізу працездатності жорсткого диску ПК.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи аналізу працездатності жорсткого диску ПК.

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 RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, жорсткий диск

ABSTRACT

Forostianyi O.A. Research and software implementation of a PC hard disk performance analysis 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 was developed, which is intended for the system of analyzing the performance of the hard disk of a PC.

The purpose of the development is the research and software implementation of the PC hard disk performance analysis system.

The object of the study is the process of analyzing the performance of the PC hard disk.

The subject of the study is the methods of analyzing the performance of the PC hard disk.

Research methods are based on computer architecture methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the PC hard disk performance analysis 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 RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, hard disk

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи..... 6
1.2 Область застосування..... 7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 15
2.3 Розгорнута постановка завдання 21
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 22
3.1 Опис функціонування системи 22
3.2 Розробка структурної схеми..... 24
3.3 Розробка функціональної схеми 29
3.4 Розробка діаграми процесів..... 33
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 35
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 35
4.2 Захист розробленого програмного забезпечення..... 47
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 51
6 НАУКОВА НОВИЗНА 53

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ 54
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 54
7.2 Розрахунок трудомісткості розробки програмної продукції 56
7.3 Визначення чисельності виконавців і планового фонду зарплати 58
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника 62
7.5 Визначення собівартості розробки та ціни програмної продукції. 66
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції. 70
7.7 Визначення експлуатаційних витрат 70
7.8 Визначення економічної ефективності програмної продукції. 72
7.9 Висновок 74
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 75
8.1 Вступ 75
8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером 76
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 77
8.4 Розробка заходів з умов поліпшення охорони праці 80
8.5 Розрахунок занулення 81
9 ОСНОВНІ ВИСНОВКИ 85
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 87

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. Більшості користувачів комп'ютера відомо, що жорсткий диск – це не найнадійніше місце для зберігання інформації. По-перше, він схильний таким руйнівним факторам, як діяльність шкідливих програм і вірусів, некваліфіковані дії користувачів, апаратні збої і інше. По-друге, жорсткий диск, на відміну від компакт дисків або інших носіїв інформації, являє собою механічний пристрій – а будь механізм рано чи пізно зношується. Тому ніхто не застрахований від такої неприємності, як ушкодження жорсткого диска. У подібній ситуації найголовніше – не панікувати і не робити ніяких необдуманих дій, оскільки це може тільки посилити і без того складне становище. Пам'ятайте, що в більшості випадків можна врятувати дані, що зберігаються на пошкодженому жорсткому диску. Найпростіший спосіб – це звернутися в спеціалізовану фірму, співробітники якої зроблять все можливе для порятунку даних. Але, по-перше, це дорого, а по-друге – якщо на вашому жорсткому диску зберігалися якісь конфіденційні відомості, вони можуть потрапити до рук сторонніх. Самостійно відновлювати інформацію з пошкодженого диска слід тільки користувачам, які вже мають певний досвід роботи на комп'ютері. Новачкам для вирішення подібних проблем все ж порекомендуємо звернутися до фахівців, оскільки некваліфіковані дії можуть призвести до остаточної і безповоротної втрати даних.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК.

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

– Огляд існуючих систем аналізу працездатності жорсткого диску ПК.

– Дослідження системи аналізу працездатності жорсткого диску ПК.

– Програмна реалізація системи аналізу працездатності жорсткого диску ПК.

Об'єктом дослідження є процес аналізу працездатності жорсткого диску ПК.

Предметом дослідження є методи аналізу працездатності жорсткого диску ПК.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод аналізу працездатності жорсткого диску ПК.

– Розроблено вітчизняний продукт аналізу працездатності жорсткого диску ПК, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі аналізу працездатності жорсткого диску ПК.

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 аналізу працездатності жорсткого диску ПК, 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

SMART – це абревіатура від Self-Monitoring, Analysis and Reporting Technology.

SMART – це галузевий стандарт для моніторингу та звітування про несправності периферійного пристрою зберігання даних, пов’язані з пошкодженням або звичайним зносом. SMART має на меті зменшити ймовірність втрати даних шляхом відповідного попередження про ці умови, але він не замінить якісне планування аварійного відновлення.

Як працює SMART?

Диск SMART відстежує внутрішню продуктивність двигунів, носія, головок і електроніки приводу, тоді як наше програмне забезпечення контролює загальний стан надійності приводу. Статус надійності визначається за допомогою аналізу внутрішнього рівня продуктивності накопичувача та порівняння внутрішніх рівнів продуктивності із заздалегідь визначеними пороговими значеннями. Накопичувачі з підтримкою SMART можуть контролювати кілька ключових факторів продуктивності, щоб оцінити надійність і передбачити загрозливий вихід пристрою з ладу. Однак SMART не може виявити всі загрозливі збої. SMART слід розглядати як консультаційну службу, а не як заміну регулярного резервного копіювання ваших файлів. Безпека ваших даних може бути забезпечена лише регулярним створенням резервних копій. Функції SMART будь-якого пристрою не можна розглядати як заміну плануванню наперед.

1.2 Область застосування

Щоб допомогти користувачам уникнути втрати даних, виробники накопичувачів тепер включають у свої накопичувачі логіку, яка діє як «система раннього попередження» про невирішені проблеми з приводом. Ця система називається *технологією аналізу та звітування самоконтролю* або *SMART*. Інтегрований контролер жорсткого диска працює з різними датчиками, щоб відстежувати різні аспекти продуктивності накопичувача, визначає на основі цієї інформації, чи поводиться він нормально чи ні, і надає інформацію про стан програмному забезпеченню, яке перевіряє диск і переглядає його.

Основний принцип SMART полягає в тому, що багато проблем із жорсткими дисками не виникають раптово. Вони є результатом повільної деградації різних механічних або електронних компонентів. SMART розвинувся з технології, розробленої IBM під назвою *Predictive Failure Analysis* або *PFA*. PFA поділяє невдачі на дві категорії: ті, які можна передбачити, і ті, які неможливо. Передбачувані збої відбуваються повільно з часом і часто дають *підказки* про їх поступовий збій, який можна виявити. Прикладом такої передбачуваної несправності є перегорання підшипника шпиндельного двигуна: це часто відбувається протягом тривалого часу, і його можна виявити, звернувши увагу на те, скільки часу потрібно обертанню приводу вгору або вниз, відстежуючи температуру підшипників або відстежуючи, скільки струму споживає двигун шпинделя. Прикладом непередбачуваної несправності може бути перегорання чіпа на логічній платі жорсткого диска: часто це «просто станеться» одного дня. Очевидно, що такі непередбачувані збої неможливо спланувати.

Інженери з надійності виробника накопичувачів аналізують несправні диски та різні механічні й електронні характеристики приводу, щоб визначити різні кореляції: зв’язки між передбачуваними відмовами, а також значеннями та тенденціями в різних характеристиках приводу, які свідчать про можливість повільного погіршення якості приводу. Точні характеристики, що

контролюються, залежать від конкретного виробника та моделі. Ось деякі з них, які зазвичай використовуються:

– Висота польоту голови: Тенденція до зниження висоти польоту часто передвіщає падіння голови.

– Кількість перепризначених секторів: якщо диск перепризначає багато секторів через внутрішньо виявлені помилки, це може означати, що диск починає працювати.

– Використання ECC і підрахунок помилок: кількість помилок, з якими стикається накопичувач, навіть якщо вони виправлені всередині, часто вказує на проблеми, що розвиваються з приводом. Тенденція в деяких випадках важливіша за фактичну кількість.

– Час розкручування: Зміни часу розкручування можуть вказувати на проблеми з двигуном шпинделя.

– Температура: Підвищення температури приводу часто свідчить про проблеми з мотором шпинделя.

– Пропускна здатність даних: Зниження швидкості передачі даних накопичувача може сигналізувати про різні внутрішні проблеми.

Деякі характеристики якості та надійності, які я описую в цій частині, насправді використовуються для передачі даних у програмне забезпечення SMART.

За допомогою статистичного аналізу «прийнятні» значення різних характеристик програмуються в приводі. Якщо вимірювання для різних атрибутів, що контролюються, виходять за межі допустимого діапазону, або якщо тенденція *характеристики* демонструє неприйнятне зниження, умова попередження записується в реєстр стану SMART диска, щоб попередити, що проблема з приводом може бути що відбуваються.

Для SMART потрібен жорсткий диск, який підтримує цю функцію, і якесь програмне забезпечення для перевірки стану диска. Усі основні виробники накопичувачів тепер включають функцію SMART у свої накопичувачі, і більшість новітніх комп’ютерних систем і материнських плат мають підпрограми

BIOS, які перевіряють стан SMART приводу. Якщо ваш комп'ютер не має вбудованої підтримки SMART, можна налаштувати деякі службові програми (наприклад, Norton Utilities та подібні пакети), щоб перевіряти стан дисків SMART. Це важливий момент, про який слід пам'ятати: жорсткий диск не генерує сповіщень SMART, він лише надає інформацію про стан. Ці дані про стан необхідно регулярно перевіряти, щоб ця функція мала будь-яку цінність.

Зрозуміло, що SMART є корисним інструментом, але не надійним: він може виявити деякі проблеми, але про інші не має жодного поняття. Хорошою аналогією для цієї функції було б розглядати її як сигнальні лампи на панелі приладів вашого автомобіля: те, на що варто звернути увагу, але не покладатися на це. Ви не повинні припускати, що через те, що SMART створив сповіщення, виникла проблема диска, або навпаки, що відсутність сигналу означає, що диск не може мати проблеми. Це, звичайно, не замінить належний догляд за жорстким диском і технічне обслуговування, або звичайне та поточне резервне копіювання.

Якщо під час використання накопичувача ви бачите попередження SMART, вам слід негайно припинити його використання та звернутися до відділу технічної підтримки виробника приводу, щоб отримати інструкції. Деякі компанії вважають сповіщення SMART достатнім доказом того, що диск несправний, і негайно видають RMA для його заміни; інші вимагають виконання інших кроків, наприклад запуску діагностичного програмного забезпечення на диску. Ні в якому разі не ігноруйте сповіщення. Іноді я бачу, як люди запитують інших, «як вони можуть вимкнути ці дратівливі SMART-повідомлення» на своїх ПК. Зробити це все одно, що наклеїти ізоляційну стрічку на лампу тиску масла вашого автомобіля, щоб вона не турбувала вас під час їзди!

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

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Якщо ПК останнім часом почав пригальмовувати в роботі, досить повільно відбувається процес зчитування-запису інформації, або взагалі кудись зникають файли з жорсткого диску, значить настав момент протестувати HDD.

Victoria

Перша з вище згаданих утиліт тестує жорсткі диски з IDE і SATA інтерфейсами. Сам тест HDD потрібно проводити з загрузочного диску, тому перед його початком заходимо в BIOS і встановлюємо параметри завантаження з CD.

Victoria виводить на монітор повну інформацію про жорсткий диск, що піддається тестуванню. Ви маєте змогу переглянути S.M.A.R.T параметри. В програмі існує п'ять ступенів діагностики поверхні вінчестерів згідно яких можна проводити тест HDD.

Отже вставляємо диск з утилітою і перезавантажуємо ПК. Програма повинна завантажитись автоматично. Тиснемо F2 для визначення жорсткого диску. Якщо Victoria не змогла визначити HDD, або на вашому комп'ютері стоїть декілька вінчестерів, тоді тиснемо "P" й здійснюємо вибір в ручну, з представленого меню. В меню "Выбор порта HDD" будуть такі опції, як: Primary Master, Primary Slave, Secondary Master, Secondary Slave – для вінчестерів з IDE інтерфейсом і Ext. PCI ATA/SATA – для HDD з SATA інтерфейсом. Тобто для IDE вінчестерів з меню потрібно вибрати цей режим в якому стоїть потрібний нам жорсткий диск, наприклад Primary Master. Хочу наголосити, що Master/Slave

це положення джампера жорсткого диску, а Primary/Secondary – вид підключення. Ну а з SATA думаю все зрозуміло.

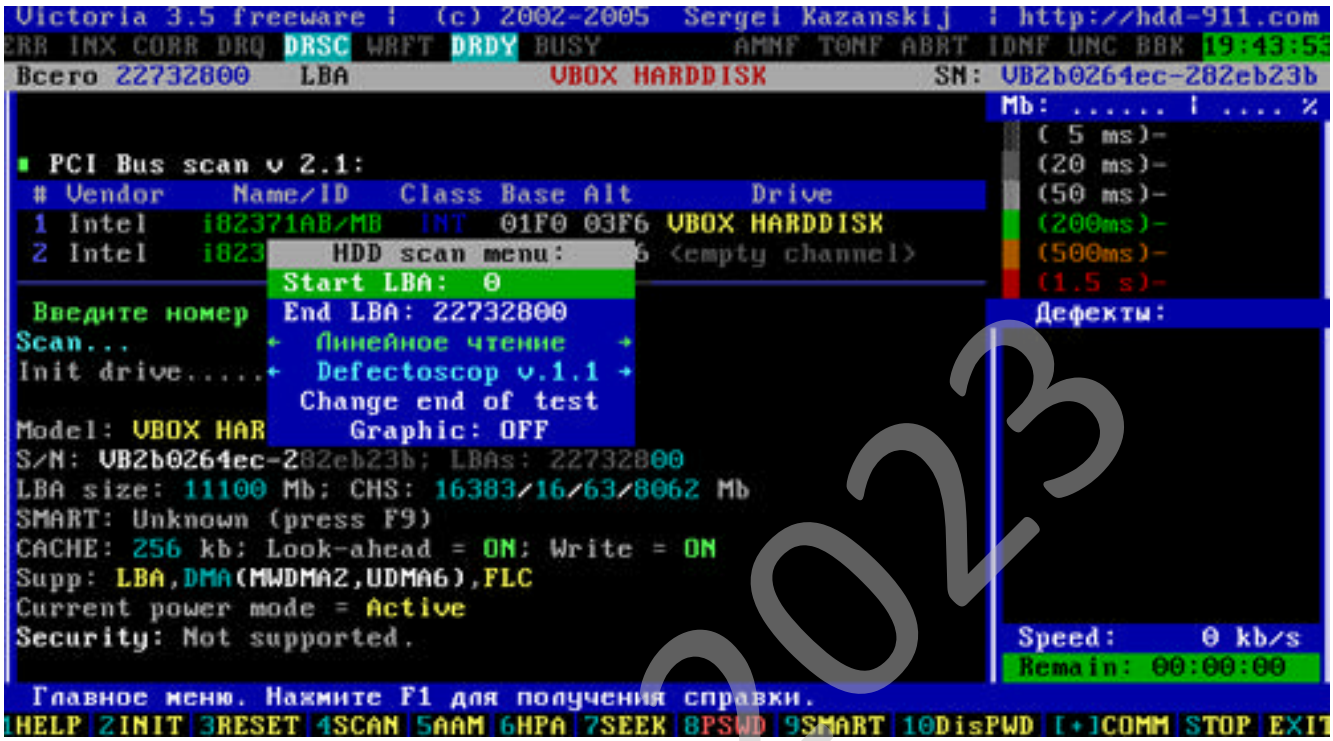

Рисунок 2.1 – Інтерфейс користувача Victoria

Далі ми побачимо перелік моделей вінчестерів змонтованих в системному блоці, якщо їхній тип інтерфейсу – SATA. Потрібно лише ввести номер каналу вінчестера обраного для діагностики, наприклад – "2". Що стосується IDE вінчестерів то в одному з вище описаних режимів може працювати тільки один жорсткий диск на ПК. Тепер знову тиснемо F2. На екрані появиться технічний паспорт жорсткого диску. Наступними кроком буде відкриття "HDD scan menu" з допомогою клавіші F4.

Для цього, щоб просканувати весь жорсткий диск вибираємо пункт "Start LBA: 0". Для вибору місця завершення тесту слід обрати в меню пункт "End LBA:...". Далі тиснемо клавішу пробіл. Тепер вводимо об'єм цілим числом в Гб і натискаємо Enter.

Наступний пункт меню "Линейное чтение" стоїть за замовчуванням, залишаємо його без змін. В наступному рядку натиском клавіші Space задаємо параметр "BB = Advanced REMAP". Залишається запустити процес тестування з допомогою клавіші Enter.

Всі дефекти, (тобто перелік вилучених бед-блоків жорсткого диску), будуть відображатись в правій стороні вікна програми. В такому випадку утиліта автоматично здійснить заміну пошкоджених секторів на резервні.

Ось в такому форматі відбувається тестування вінчестера. Тривалість тесту залежить від об'єму жорсткого диску і записаної на нього інформації.

MHDD

В даний момент MHDD є, мабуть, найпопулярнішою програмою для глибокої діагностики, відновлення, і форматування жорстких дисків. MHDD вміє працювати практично з усіма існуючими інтерфейсами. Вона успішно працює з інтерфейсами IDE, SCCI, SATA. Також програма використовує спеціальний драйвер, який дає їй можливість працювати і з USB-пристроями. За допомогою MHDD можна надзвичайно точно визначити кількість збійних і дефектних секторів, можна подивитися SMART-атрибути, можна проробити масу інших операцій, необхідних для відновлення жорстких дисків.

Коли потрібно задуматися про використання цієї програми:

– Коли з комп’ютером починає відбуватися щось незрозуміле. Наприклад, коли операційна система без всяких причин падає, і показує синій екран (який багато хто комп’ютерники називають “екраном смерті”). І якщо після того, як користувач переустановив операційну систему заново, «сині екрани» продовжують виникати – це перша ознака несправності жорсткого диска.

– Ще одна причина задуматися про перевірку диска – «гальмування» комп’ютера. Комп’ютер може почати гальмувати тоді, коли приводів для цього у нього немає: оперативна пам’ять не забита, сторонніх процесів не запущено. А машина – гальмує.

– І нарешті, третя, і найсерйозніша причина провести діагностику: жорсткий диск комп’ютера перестав відображатися в БІОС-і.

Як бути, якщо все, вказане вище, вже відбулося:

1. Насамперед слід витягти несправний жорсткий диск із комп’ютера, і підключити його до іншого комп’ютера, який гарантовано справний. Як варіант – можна підключити несправний жорсткий диск до справного ноутбуку, а для підключення можна скористатися USB-перехідником.

2. Після підключення – слід повністю відформатувати несправний диск прямо з-під операційної системи, яка встановлена на справній машині.

3. Після повного форматування – необхідно повернути несправний диск на місце, і знову встановити операційну систему. І якщо комп’ютер після установки системи все одно продовжує відчайдушно гальмувати – необхідно провести діагностику жорсткого диска за допомогою MHDD.

4. Перше, що для цього потрібно зробити – зробити перезавантаження комп’ютера, після чого увійти в БІОС, і налаштувати завантаження пристроїв таким чином, щоб першим завантажувався дисковод.

5. Далі слід завантажити iso-образ програми з офіційного сайту, і записати скачаний образ на CD-диск. (Програма MHDD є безкоштовною, і поширюється абсолютно вільно).

Після цього – завантажуємося з CD-диска, і бачимо меню програми.

Запускаємо сканування поверхні жорсткого диска простим натисненням кнопки F4. Тобто наочно видно, що час відгуку у деяких секторів жорсткого диска перевищує 500 мс. А це означає, що жорсткий диск – дуже сильно зношений.

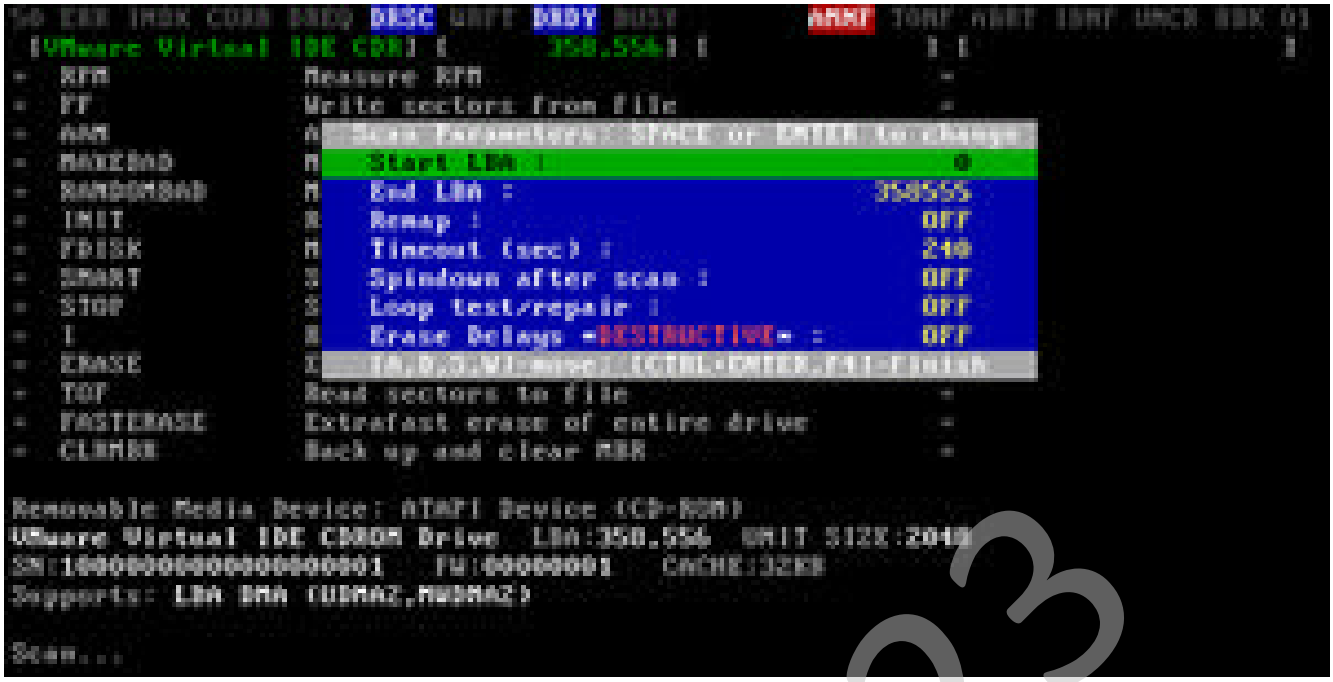

Рисунок 2.2 – Інтерфейс користувача MHDD

Тим не менш, можна спробувати дати йому друге життя:

– Для цього в меню слід виконати команду «Erase», яка зітре жорсткий диск повністю, причому прати вона його буде з використанням низькорівневого форматування.

– Після того, як диск буде відформатований, слід зробити ще одне сканування, однак перед початком цього сканування – потрібно включити функцію «Remap», встановивши її в положення «ON». Включення цієї функції дозволяє програмі MHDD блокувати збійні та пошкоджені сектора жорсткого диска, і вони надалі – не будуть використовуватися в роботі жорсткого диска.

– Після повторного сканування всі збійні сектора будуть заблоковані програмою, про що вона видасть відповідне повідомлення. Далі можна переходити до установки операційної системи на жорсткий диск.

Однак слід сказати, що програма MHDD, хоча і є дуже корисною, не є панацеєю від усіх бід. Якщо жорсткий диск використовується вже дуже давно, і практично повністю виробив свій ресурс, то використання програми вирішить проблему лише на час. А потім виникнуть нові збійні сектора, які знову почнуть

заважати роботі. І в цій ситуації без покупки нового жорсткого диска вже не обійтися.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких

же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

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

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

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

Єдине керування пам'яттю

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

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

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє

використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

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

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

Багато з нас стикалися з поломкою жорсткого диска або SSD. Деякі з нас навіть намагалися дізнатися більше про надійність жорстких дисків та їх приховану функцію прогнозування, яка є частиною технології під назвою SMART. Хтось може заперечити, що SMART не настільки надійний, оскільки він не передбачає невдач у всіх випадках. Цей факт частково вірний, але насправді внутрішня робота цієї системи самоконтролю не така проста, тому давайте розглянемо, як працює SMART. Ми також збираємося показати вам, як перевірити статус HDD SMART, а також стан твердотільного накопичувача SMART:

SMART – це система, яка контролює внутрішню інформацію вашого накопичувача. Його розумна назва насправді є абревіатурою від Self-Monitoring, Analysis, and Reporting Technology. SMART, також пишеться як SMART, – це технологія, яка міститься в жорстких і твердотільних накопичувачах. Це не залежить від вашої операційної системи, BIOS чи іншого програмного забезпечення.

SMART був винайдений тому, що комп’ютерам було потрібно щось, що могло б відстежувати стан їхніх жорстких дисків. Це означає, прямо кажучи, що SMART нібито має бути в змозі сказати вам, якщо ваш жорсткий або твердотільний накопичувач ось-ось перестане працювати.

Інформацію про стан диска надає SMART

Як SMART це робить? У вас може виникнути спокуса подумати, що SMART магічним чином здогадається, чи справний ваш диск. Але те, що він робить, це зовсім інша історія. SMART відстежує низку змінних , кількість і тип яких змінюються від приводу до приводу, що є показниками його надійності.

Якщо ви хочете отримати детальне уявлення про всі атрибути SMART, оскільки їх близько 50 (рівень помилок необробленого читання, час розкручування, повідомлення про невиправні помилки, час увімкнення, кількість циклів завантаження тощо) відвідайте цю веб-сторінку.

Однак знайте, що, за винятком окремих спроб (Google , Backblaze), більшість даних SMART є незадокументованими. Система надає велику кількість внутрішніх даних. Проте в статистиці є багато невідповідностей, оскільки багато виробників жорстких дисків використовують різні визначення та вимірювання. Наприклад, деякі виробники зберігають дані про час увімкнення в годинах, а інші вимірюють їх у хвилинах або секундах. Крім того, вони не пояснюють, які з різних атрибутів або змінних варті нашої уваги, що змушує нас потонути в даних.

Перш ніж спробувати зрозуміти, які атрибути SMART є доречними, ми спочатку повинні розрізнити основні типи збоїв SSD і HDD: передбачувані та непередбачувані.

Деталі SMART для твердотільного накопичувача

Передбачувані збої включають поломки, які виникають вчасно і викликані несправністю дискової механіки або пошкодженням поверхні диска у випадку жорстких дисків. Для твердотільних накопичувачів передбачувані збої можуть включати звичайний знос із часом або велику кількість невдалих спроб стирання. Проблеми з часом посилюються, і диск зрештою виходить з ладу.

Непередбачувані збої викликані раптовими подіями, серед яких ми можемо згадати, наприклад, раптові стрибки напруги або несподіване пошкодження схем всередині жорсткого диска або твердотільного накопичувача. Важливо розуміти, що SMART може допомогти вам лише виявити передбачувані збої.

Тепер, коли ви маєте основне розуміння того, що таке SMART і що він робить, давайте подивимося, як перевірити стан SMART ваших дисків у Windows, а також як читати та інтерпретувати деталі SMART:

Як перевірити стан SSD і HDD SMART

На комп'ютерах і пристроях з ОС Windows найпростіший спосіб читати дані SMART із жорсткого диска чи SSD – це використовувати спеціалізовані програми. Їх досить багато, але багато з них або погано розроблені, або коштують грошей. З усіх програм, які можуть зчитувати дані SMART, найкращою, яку ми рекомендуємо вам використовувати, є CrystalDiskInfo. Він безкоштовний, може зчитувати атрибути SMART, а також це одна з небагатьох таких програм, яка може отримувати дані SMART як з дисків IDE(PATA), SATA та NVMe, так і з портативних дисків, які використовують eSATA, USB, або IEEE 1394.

Ще один чудовий спосіб перевірити стан SMART і деталі жорсткого диска або твердотільного накопичувача – це використовувати програми, надані їх виробником. Наприклад, більшість твердотільних накопичувачів супроводжуються додатками підтримки, які дозволяють перевіряти інформацію про них, перевіряти їх справність, запускати діагностику тощо. Ці програми зазвичай містять опції для перевірки стану SMART.

Третій спосіб перевірки стану SMART вашого жорсткого диска або SSD пропонує Windows 10. Він не показує подробиці, але може сказати вам, чи стан SMART ваших дисків нормальний чи ні. Щоб перевірити SMART, відкрийте командний рядок і виконайте цю команду: wmic diskdrive get model, status. Команда виводить список дисків, підключених до вашого ПК, і показує статус SMART для кожного з них.

Цей останній метод перевірки стану SMART є, мабуть, найшвидшим способом у Windows 10 перевірити, чи ваші диски виходять з ладу.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленої системи аналізу працездатності жорсткого диску ПК. Структурна схема складається з наступних блоків:

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

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

– Блок одержання таблиці в буфер додатка.

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

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

– Блок виводу числових значень в зручному для сприйняття вигляді.

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

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

Якщо вас не задовольняє просто читання SMART-статусу ваших дисків, ви також можете запустити SMART-тест SSD або HDD. Це легше сказати, ніж зробити, оскільки для цього вам потрібен спеціальний додаток. Відповідно, ми вважаємо, що це тема, яка заслуговує на окрему статтю, доступ до якої ви можете отримати за цим посиланням: Перевірте свій HDD або SSD і перевірте стан його працездатності.

Як читати значення та атрибути SMART

Стан жорсткого диска постійно перевіряється та контролюється кількома датчиками. Значення вимірюються за допомогою типових алгоритмів, а потім відповідні атрибути налаштовуються відповідно до результатів.

У будь-якій програмі моніторингу SMART ви повинні побачити атрибути, які містять принаймні деякі з цих полів:

– Ідентифікатор: визначення атрибута. Зазвичай воно має стандартне значення та позначається числом від 1 до 250 (наприклад, 9 означає кількість увімкнень). Тим не менш, усі інструменти моніторингу та тестування диска надають назву та текстовий опис атрибута.

– Поріг: мінімальне значення для атрибута. Якщо це значення досягнуто, ваш диск скоро виходить з ладу.

– Значення: поточне значення атрибута. Алгоритм обчислює це число на основі необроблених даних. Новий жорсткий диск повинен мати високий теоретичний максимум (100, 200 або 253 залежно від виробника), який зменшується протягом терміну служби.

– Найгірше: найменше значення атрибута, яке будь-коли зареєстроване.

– Дані: необроблені виміряні значення, надані датчиком або лічильником. Це дані, які використовуються алгоритмом, розробленим виробником HDD або SSD. Його вміст залежить від атрибута та виробника накопичувача. Звичайні користувачі повинні пропустити цей.

– Прапори: призначення атрибута. Зазвичай це встановлюється виробником і тому залежить від диска. Кожен із атрибутів є критичним і може передбачати неминучий збій (наприклад, кількість перерозподілених секторів ID 5), або статистичним, що не впливає безпосередньо на стан (наприклад, кількість неочікуваних втрат живлення ID 174).

Атрибути SMART описуються такими даними, як їхній ідентифікатор, поточне значення, найгірше значення та порогове значення.

Намагаючись зрозуміти статус будь-якого атрибута SMART, перевірте значення цих трьох полів: значення, порогове значення та прапорці. Крім того, пам’ятайте, що зазвичай менші значення вказують на зниження надійності.

Як використовувати SMART для прогнозування несправності жорсткого диска або SSD (основні значення для перевірки).

Не всі атрибути SMART критичні для прогнозування помилок. Два вищезгаданих дослідження частоти відмов жорстких дисків та інші джерела сходяться на тому, що важливою допомогою у виявленні несправних дисків є:

– Кількість перерозподілених секторів. Перерозподіл відбувається, коли логіка накопичувача переназначає пошкоджений сектор у результаті повторюваних м’яких або жорстких помилок на новий фізичний сектор із

резервних. Цей атрибут відображає кількість разів, коли відбулося перевідображення. Якщо його значення збільшується, це свідчить про знос HDD або SSD.

– Поточна кількість секторів, що очікують на розгляд. Тут враховуються «нестабільні» сектори, тобто пошкоджені з помилками читання, які очікують на перевідображення, свого роду система «випробувального терміну». Алгоритми SMART неоднозначно розуміють цей конкретний атрибут, оскільки він іноді непереконливий. Тим не менш, це може забезпечити раннє попередження про можливі проблеми.

– Повідомлено про невиправні помилки. Це підрахунок помилок, які неможливо відновити, і він корисний, оскільки, здається, має однакове значення для всіх виробників.

– Кількість помилок видалення. Це чудовий показник передчасної смерті твердотільного накопичувача. Він підраховує кількість невдалих спроб видалення даних, і значення, яке збільшується, говорить про те, що термін служби флеш-пам’яті всередині SSD майже закінчився.

– Підрахунок вирівнювання зносу. Це також особливо корисно для SSD. Виробники встановлюють очікуваний термін служби SSD у даних SMART. Підрахунок *вирівнювання зносу* – це оцінка справності вашого диска. Він розраховується за допомогою алгоритму, який враховує заздалегідь визначений очікуваний термін служби та кількість циклів (запис, стирання тощо), які може виконати кожен блок флеш-пам’яті до досягнення кінця свого життєвого циклу.

– Температура диска є параметром, який дуже дискутується. Проте вважається, що значення вище 60°C можуть скоротити термін служби жорсткого диска або SSD і збільшити ймовірність пошкодження. Ми рекомендуємо використовувати вентилятор, щоб знизити температуру ваших дисків і, сподіваємося, подовжити термін їх служби.

Значення SMART, які перевищують порогове значення, можуть вказувати на можливі майбутні збої диска.

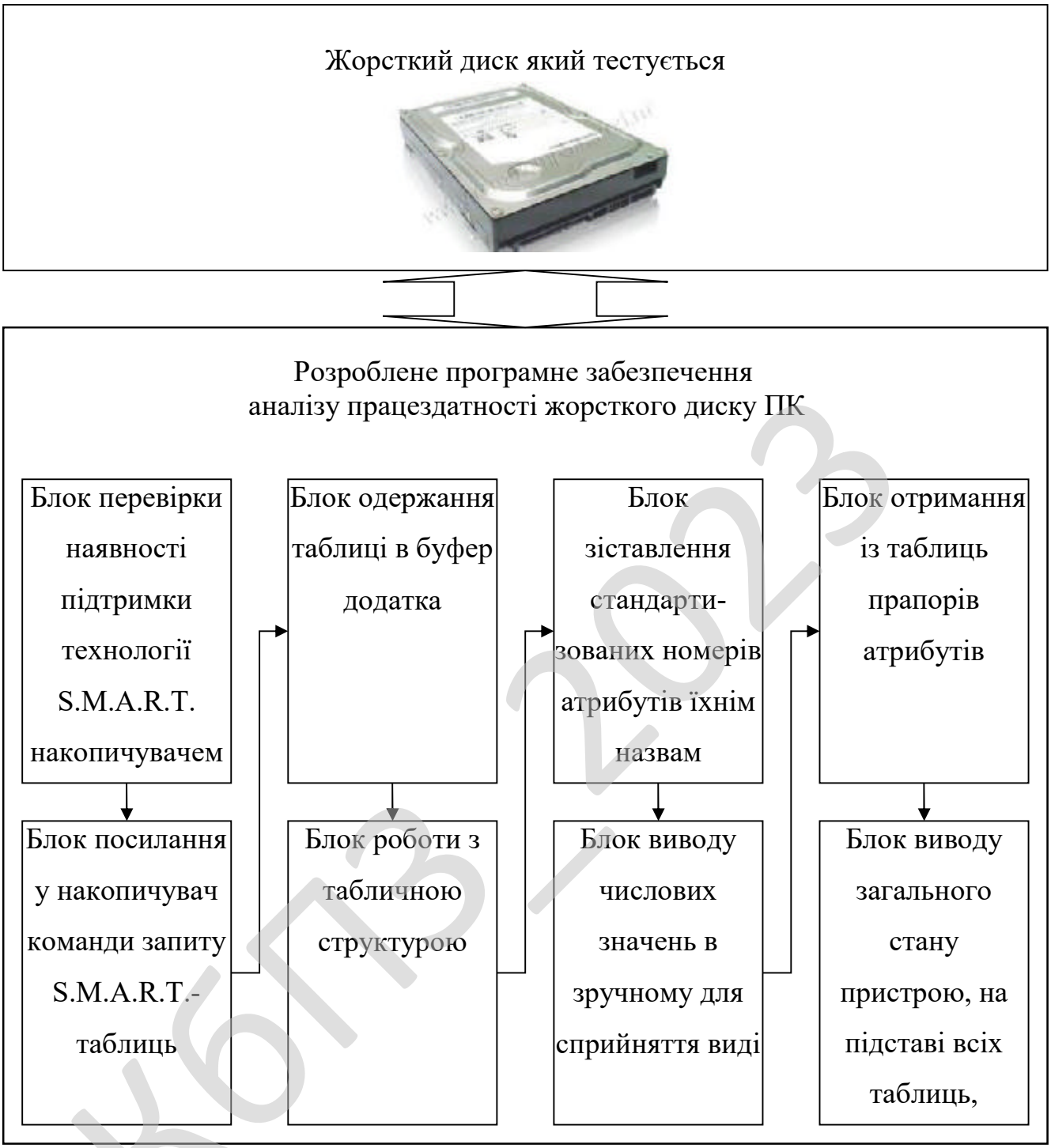

Рисунок 3.1 – Структурна схема системи

Вищезгадані атрибути SMART відносно легко інтерпретувати. Якщо ви помітили збільшення їх значень, можливо, ваш накопичувач вийшов з ладу, тому вам краще почати резервне копіювання. Однак, хоча це корисні показники надійності диска, не забувайте, що вони не є надійними.

Історична довідка про SMART

SMART було розроблено з 1992 року, хоча тепер ви знаєте, що він є частиною всіх сучасних твердотільних накопичувачів і жорстких дисків. Його історія охоплює низку назв, таких як Predictive Failure Analysis або IntelliSafe, а також дані від усіх основних виробників жорстких дисків: IBM, Seagate, Quantum, Western Digital. Нарешті, його документація була представлена вперше в 2004 році в рамках стандарту Parallel ATA і отримувала регулярні перегляди згодом. Останній був виданий у 2011 році.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Програма надає можливість стежити за практично всіма атрибутами S.M.A.R.T., має приємний інтерфейс, і має велику кількість налаштувань.

Функціонально вона складається з наступних блоків:

– Головне вікно інтерфейсу користувача програмного забезпечення аналізу працездатності жорсткого диску ПК.

– Загальні відомості про диски.

– Загальні відомості S.M.A.R.T.

– S.M.A.R.T. докладно.

– Налаштування.

Розглянемо ці функціональні блоки більш детально.

Загальні відомості про диски

У лівій половині зазначені: технічні характеристики, такі як обсяг диска, кількість циліндрів, головок і т.п.; режим роботи диска в даний момент (PIO, multiword DMA, UDMA); підтримувані режими роботи диска (тільки в Розширеному режимі). У правій половині показується логотип фірми-виробника жорсткого диска й нижче – загальна інформація про диск: модель диска, серійний номер диска, дата/ревізія прошивання мікропрограми.

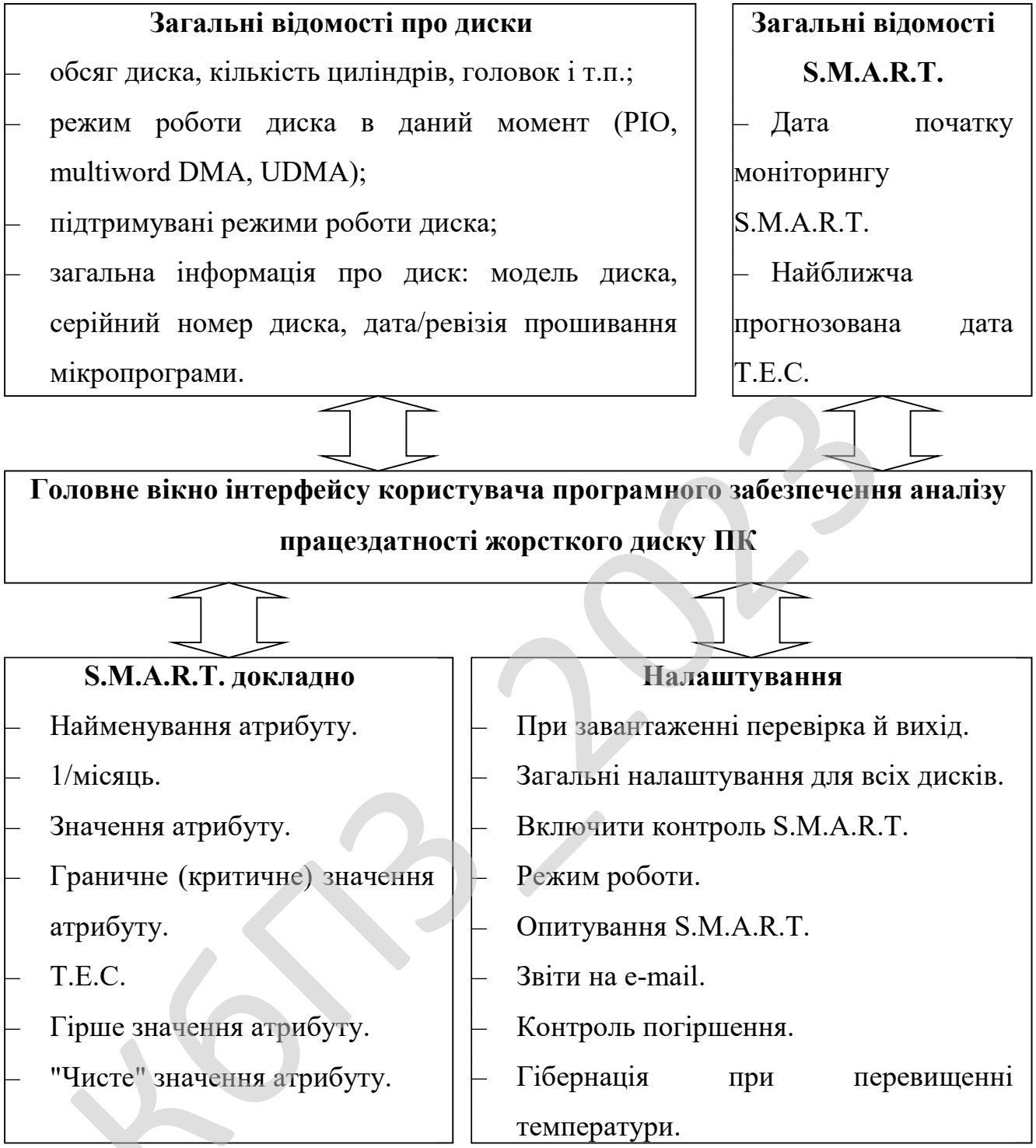

Рисунок 3.2 – Функціональна схема системи

Загальні відомості S.M.A.R.T.

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

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

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

S.M.A.R.T. докладно

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

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

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

– Значення атрибуту – поточне значення даного атрибуту S.M.A.R.T.

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

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

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

– "Чисте" значення атрибуту – просто числове значення атрибуту в чистому, неопрацьованому виді.

Налаштування

Закладка "Налаштування" призначена для самостійного налаштування користувачем параметрів розробленого програмного продукту для роботи на комп'ютері. Якщо ви не вважаєте себе досвідченим користувачем, рекомендуємо скористатися "Майстром налаштування" – він допоможе вам вибрати найбільш підходящі параметри роботи.

Основні й найбільш важливі налаштування:

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

– Загальні налаштування для всіх дисків – Розроблений програмний продукт буде використовувати загальні налаштування для всіх дисків у комп'ютері. Вони включають: контроль S.M.A.R.T., період опитування S.M.A.R.T. і адреса електронної пошти для повідомлень. Ви можете встановити загальні або індивідуальні для кожного диска параметри.

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

– Режим роботи – Звичайний або Розширений – Звичайний режим – основної для користувачів. У цьому режимі розроблений програмний продукт показує значення атрибуту, граничне значення й Т.Е.С., швидкість падіння атрибуту. На закладці "Загальне" Ви не побачите інформації про підтримуваний диском режимах роботи (передачі даних). У розширеному режимі додатково показують Гірше й Чисте (Raw) значення атрибуту й повну інформацію про диск на закладці "Загальне".

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

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

– Контроль погіршення – інформація тільки про значні зміни (погіршення) параметрів S.M.A.R.T.

– Гібернація при перевищенні температури – якщо температура HDD перевищує встановлене значення, комп'ютер переходить у режим гібернації (hibernate).

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3. Після початку роботи ПЗ ми потрапляємо до інтерфейсу ПЗ (головне вікно) звідки ми можемо переглянути загальні відомості S.M.A.R.T. з загальними відомостями HDD та налаштування ПЗ, далі через обробник помилкових запитів проводиться S.M.A.R.T запит до обраного HDD з подальшим отриманням таблиць прапорів атрибутів та отримання самої таблиці з якої проводяться дії з пошуку несправностей, аналізу загального стану пристрою, порівняння номерів атрибутів їхнім назвам.

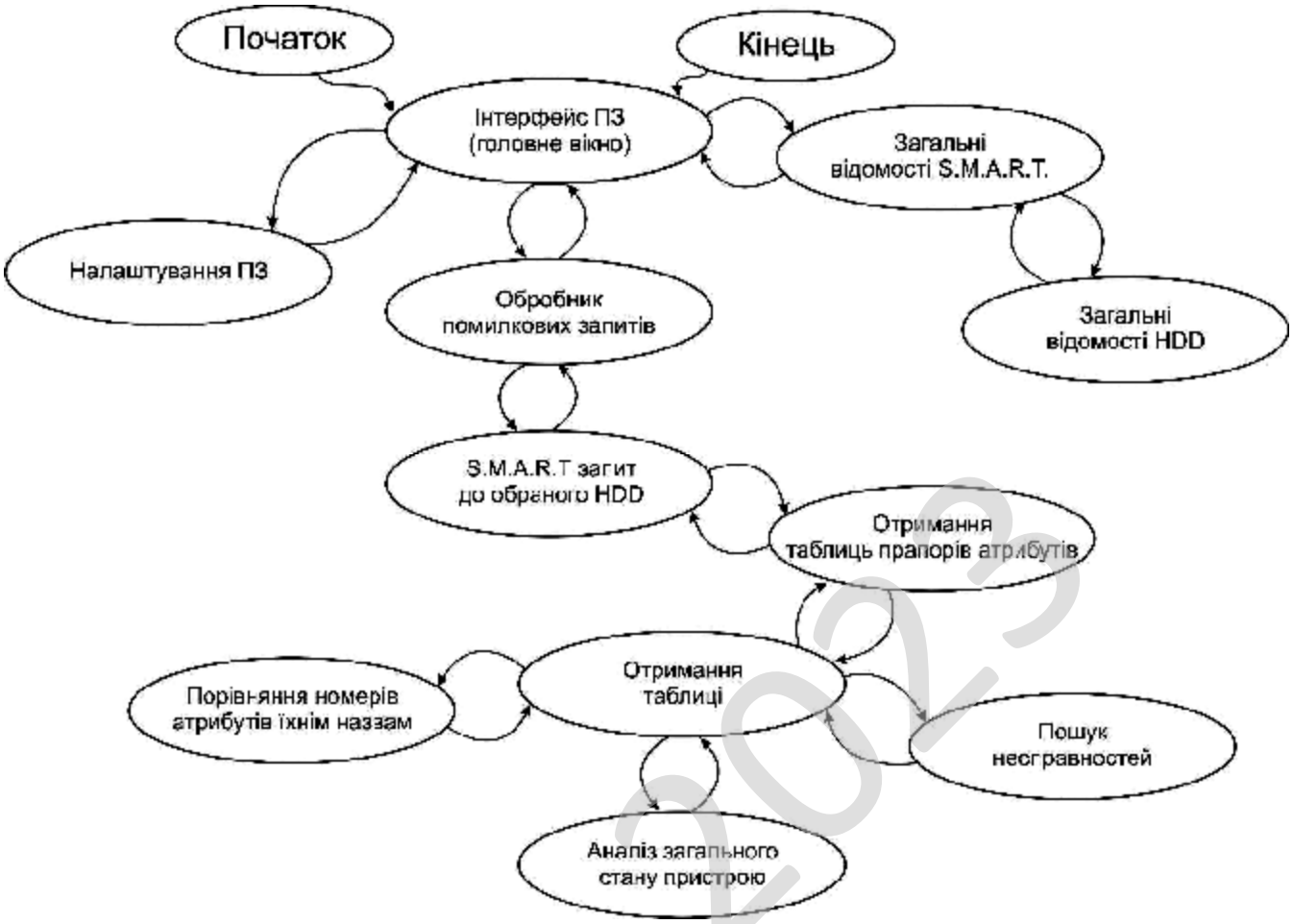

Рисунок 3.3 – Діаграма взаємодії процесів

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.

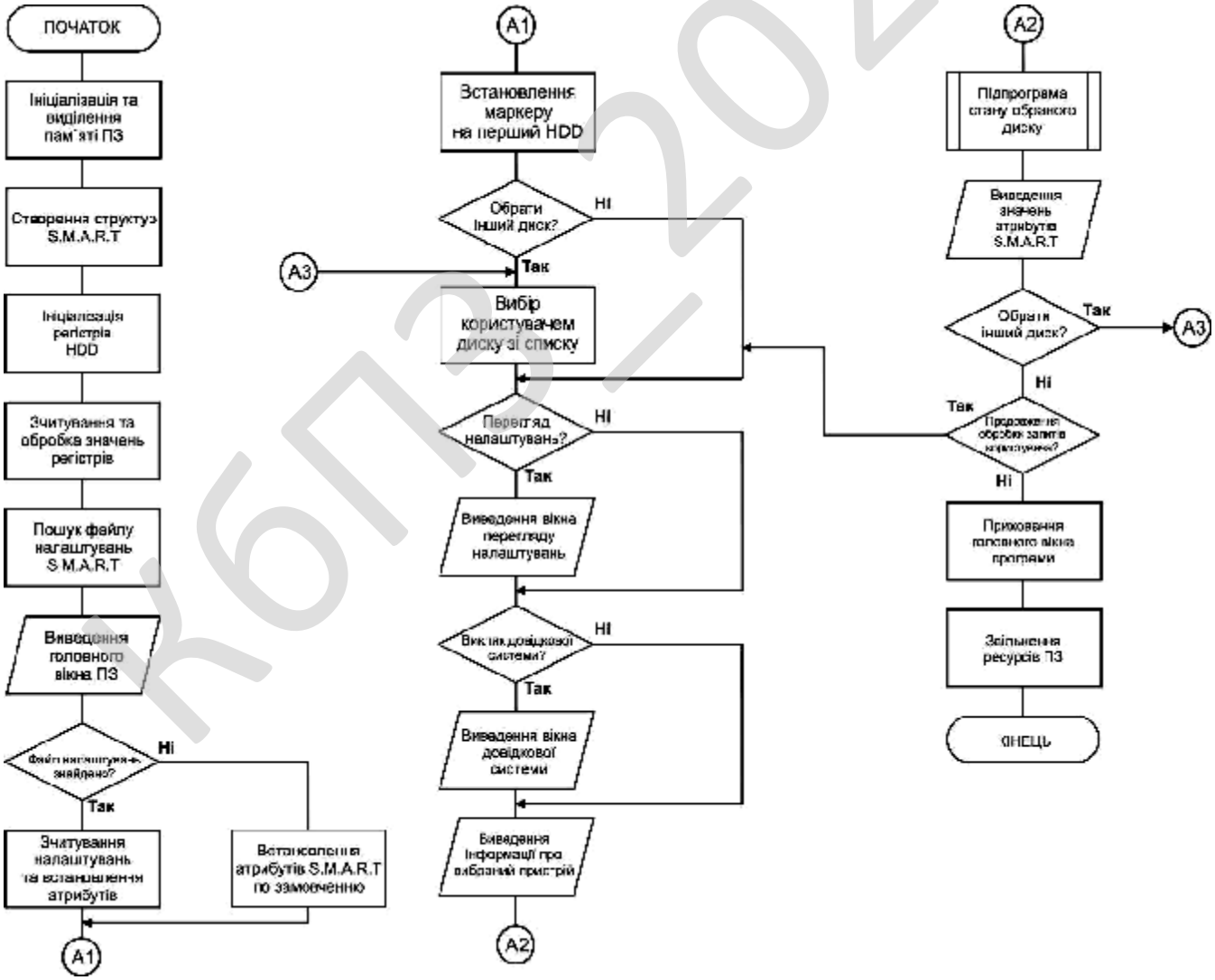

Рисунок 4.1 – Блок-схема основної програми

Після цього проводиться ініціалізація та виділення пам`яті ПЗ, створення структур S.M.A.R.T, ініціалізація регістрів HDD, зчитування та обробка значень регістрів та пошук файлу налаштувань S.M.A.R.T після чого проводиться виведення головного вікна ПЗ. Далі проходе запит файл налаштувань знайдено з подальшим зчитуванням налаштувань та встановленням атрибутів, встановлення маркеру на перший HDD у списку.

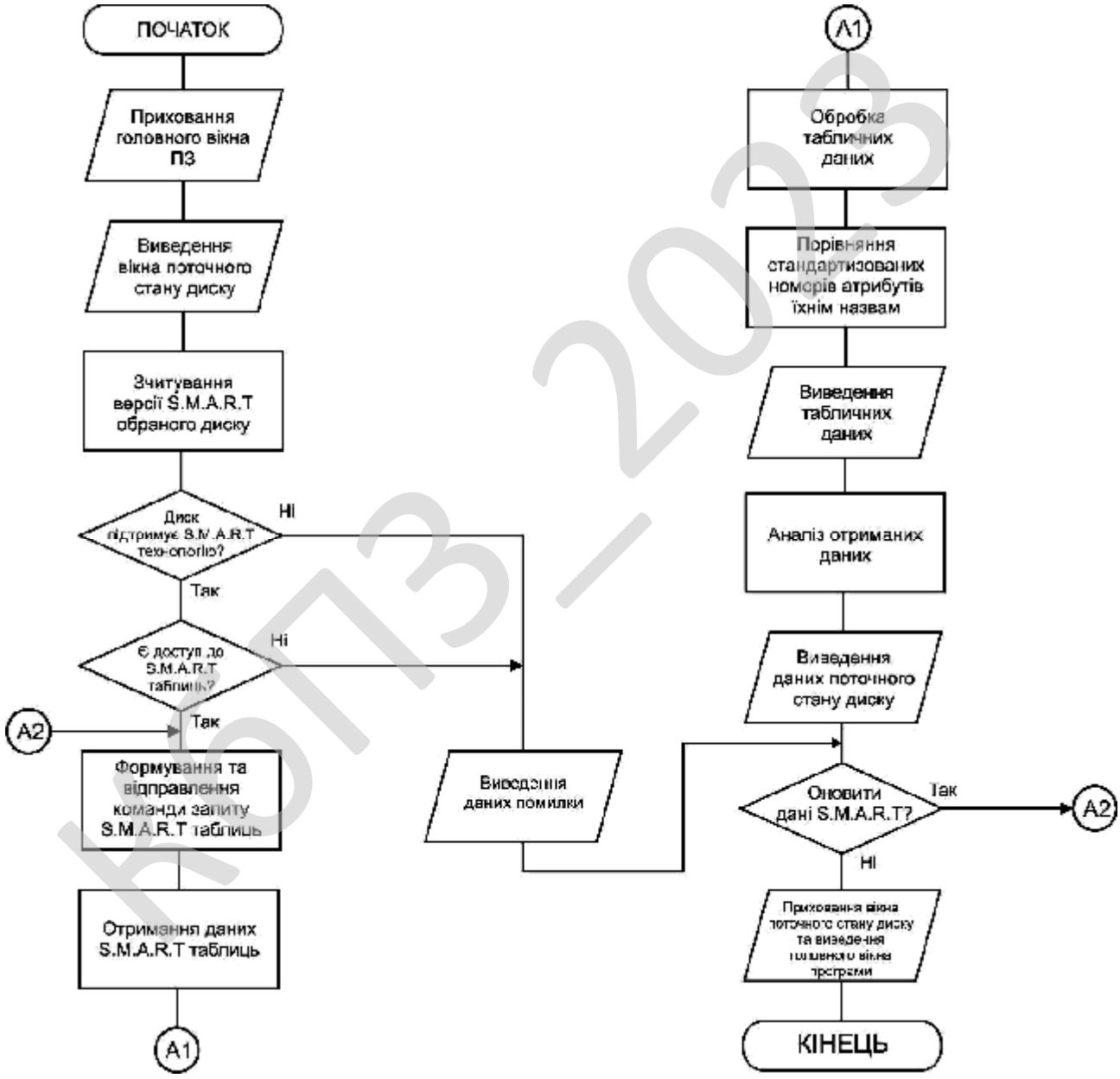

Рисунок 4.2 – Блок-схема підпрограми

Якщо було обрано інший диск проходе встановлення диску зі списку. Якщо є запит перегляд налаштувань – виведення вікна перегляду налаштувань. Якщо є запит виклик довідкової системи – проходе виведення вікна довідкової системи.

Далі проходе виведення інформації про вибраний пристрій та виконання підпрограми стану обраного диску яка зображена на рисунку 4.2. Де проходе приховання головного вікна ПЗ, виведення вікна поточного стану диску, зчитування версії S.M.A.R.T обраного диску. Якщо диск підтримує S.M.A.R.T технологію проводиться та є доступ до S.M.A.R.T таблиць проводиться формування та відправлення команди запиту S.M.A.R.T таблиць, отримання даних S.M.A.R.T таблиць, обробка табличних даних, порівняння стандартизованих номерів атрибутів їхнім назвам виведення табличних даних, аналіз отриманих даних та виведення даних поточного стану диску.

Якщо є запит на оновлення поточних даних S.M.A.R.T проходе прехід по мітці, якщо такого запиту немає проводиться приховання вікна поточного стану диску та виведення головного вікна програми. Де проходе виведення значень атрибутів S.M.A.R.T та запит на обрання іншого диску. Якщо запита немає та проходе негативна відповідь на продовження обробки запитів користувача проводиться приховання головного вікна програми та звільнення ресурсів ПЗ.

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

При розробці я використовував стандартну документацію «Small Form Factor Committee. Specification for Self-Monitoring, Analysis and Reporting Technology», затверджений такими компаніями, як Compaq Computer Corporation, Hitachi Ltd., IBM Storage Products Company, Maxtor Corporation, Quantum Corporation, Seagate Technology, Toshiba Corporation і Western Digital Corporation.

Слід також зазначити, що на сьогоднішній день стандарт на технологію SMART не затверджено. Однак у стандарті ATA, починаючи з версії 3, описаний обов'язковий мінімум для технології SMART, і якщо ваш жорсткий диск

відповідає ATA (3-8), то він буде підтримувати дану технологію у відповідності з цим стандартом.

Аналіз стану диска проводиться за допомогою вивчення атрибутів SMART Максимальна кількість атрибутів на одному диску залежить від виробника і не перевищує 30. Атрибути можуть приймати значення в діапазоні від 1 до 253. Для кожного атрибута існує граничне значення, ґрунтуючись на якому можна судити про близькість моменту виходу приводу з ладу.

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

Розглянемо розроблений код.

Код, який визначає тип диска і намагається активувати SMART:

```
       for i: = 0 to 3 do
// Кількість і тип пристроїв визначається параметром bIDEDeviceMap
// Структури TGetVersionOutParams
       begin
// Якщо пристрій з номером "i" - IDE, передаємо йому команди.
         if VersionParams.bIDEDeviceMap shr i and 1 = 1 then
         begin
// Ігноруємо ATAPI - пристрої.
           if VersionParams.bIDEDeviceMap shr i and $ 10 = 0 then
           begin
             ZeroMemory (@ scip, sizeof (scip));
// Обнуляємо TSendCmdInParams
             ZeroMemory (@ OutCmd, sizeof (OutCmd));
// Обнуляємо TSendCmdOutParams
// Намагаємося активувати SMART.
             if DoEnableSMART (hSMARTIOCTL, @ scip, @ OutCmd, i) then
               ShowMessage ('Команда запуску SMART виконана, диск:'
                 + Inttostr (i))
             else ShowMessage ('Команда запуску SMART не виконана, диск:'
               + Inttostr (i));
           end;
         end;
```

```
      end;
```

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

Перший – для читання значень атрибутів:

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

Параметр wStatusFlags може приймати наступні значення або їх комбінації:

```
      const
// Життєво важливий
        PRE_FAILURE_WARRANTY = $ 01;
// Колекція реального часу
      ON_LINE_COLLECTION = $ 02;
// Атрибут, що відображає продуктивність диска
        PERFORMANCE_ATTRIBUTE = $ 04;
// Атрибут, що відображає частоту появи помилок
        ERROR_RATE_ATTRIBUTE = $ 08;
// Лічильник подій
        EVENT_COUNT_ATTRIBUTE = $ 10;
// атрибут що проводе самозбереження
        SELF_PRESERVING_ATTRIBUTE = $ 20;
```

Другий тип призначений для читання порогових значень:

```
      type
```

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

Функція читання значень атрибутів виглядає наступним чином:

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

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

Створення дескриптору для роботи з функціями SMART.

```
function OpenSMART (DrvNum: Byte):THandle;
  var
    hSMARTIOCTL: THandle;
```

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

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

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

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

Змінна OSVersionInfo є глобальною, і її тип визначений як TOSVersionInfo. Вона заповнюється до виклику функції OpenSMART. Отримавши дескриптор SMART, необхідно визначити версію SMART IOCTL. За це відповідає функція:

```
function GetVersionSMART (hSMARTIOCTL: THandle): TGetVersionOutParams;
var
  VersionParams: TGetVersionOutParams;
  cbBytesReturned: DWORD;
begin
  ZeroMemory (@ VersionParams, sizeof (TGetVersionOutParams));
    if not DeviceIoControl (hSMARTIOCTL, DFP_GET_VERSION, nil, 0,
```

```
            @ VersionParams, sizeof (VersionParams), cbBytesReturned, nil)
          then
            ShowMessage (SysErrorMessage (GetLastError));
        Result: = VersionParams;
      end;
```

В якості параметра передається дескриптор SMART Функція повертає структуру типу:

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

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

```
       type
        TIDERegs = packed record
// Використовується для визначення "підкоманди" SMART
          bFeaturesReg: BYTE;
// Регістр кількості секторів IDE
          bSectorCountReg: BYTE;
// Регістр номера сектора IDE
          bSectorNumberReg: BYTE;
// Молодший розряд номера циліндра IDE
          bCylLowReg: BYTE;
// Старший розряд номера циліндра IDE
          bCylHighReg: BYTE;
```

```
// Регістр диска / головки IDE
          bDriveHeadReg: BYTE;
// Фактична команда IDE
          bCommandReg: BYTE;
// Зарезервовано для майбутнього використання. Повинне бути 0.
          bReserved: BYTE;
      end;
        IDEREGS = TIDERegs;
        PIDERegs = ^ TIDERegs;
```

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

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

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

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

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

```
      type
```

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

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

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

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

```
      function DoEnableSMART (hSMARTIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
        pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
      var
        lpcbBytesReturned: DWORD;
      begin
        pSCIP.cBufferSize: = 0;
// Активувати S.M.A.R.T.
        pSCIP.irDriveRegs.bFeaturesReg: = SMART_ENABLE_SMART_OPERATIONS ($ D8);
        pSCIP.irDriveRegs.bSectorCountReg: = 1;
        pSCIP.irDriveRegs.bSectorNumberReg: = 1;
        pSCIP.irDriveRegs.bCylLowReg: = SMART_CYL_LOW;
```

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

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

Параметр bDriveNum – це номер диска в межах від 0 до 3. Після заповнення необхідних параметрів структури PSENDCMDOUTPARAMS виконуємо функцію DeviceIoControl з керуючим кодом DFP_SEND_DRIVE_COMMAND ($ 0007C084). Якщо функція виконана успішно, результат який повертається – TRUE.

Приклад читання поточних і порогових значень атрибутів диска «i» наведено нижче:

```
      var
// Два буфера для отримання даних
        AttrOutCmd, ThreshOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) - 1)
        +(READ_ATTRIBUTE_BUFFER_SIZE - 1)] of BYTE;
        bSuccess: bool;
      begin
        ZeroMemory (@ AttrOutCmd, sizeof (AttrOutCmd));
        ZeroMemory (@ ThreshOutCmd, sizeof (ThreshOutCmd));
        bSuccess: = DoReadAttributesCmd (hSMARTIOCTL, @ scip,
          PSENDCMDOUTPARAMS (@ AttrOutCmd), i);
        if bSuccess = false then ShowMessage (
          'Помилка при виконанні команди читання атрибутів SMART на диску: '
          + Inttostr (i))
// Команда читання атрибутів виконана успішно.
// Намагаємося прочитати порогові значення атрибутів.
        else if not DoReadThresholdsCmd (hSMARTIOCTL, @ scip,
          PSENDCMDOUTPARAMS (@ ThreshOutCmd), i)
        then
```

```
            ShowMessage ('Помилка при виконанні команди читання порогових значень'
              + 'Атрибутів S.M.A.R.T. на диску: '+ inttostr (i));
          if bSuccess <> false then
// Виводимо інформацію про атрибути та їх порогових значеннях
            DoPrintData (@ PSENDCMDOUTPARAMS (@ AttrOutCmd). BBuffer,
              @ PSENDCMDOUTPARAMS (@ ThreshOutCmd). BBuffer);
        end;

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

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

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

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

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоанализу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard, покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються

різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.
- $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – ($K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)}$)

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.3.

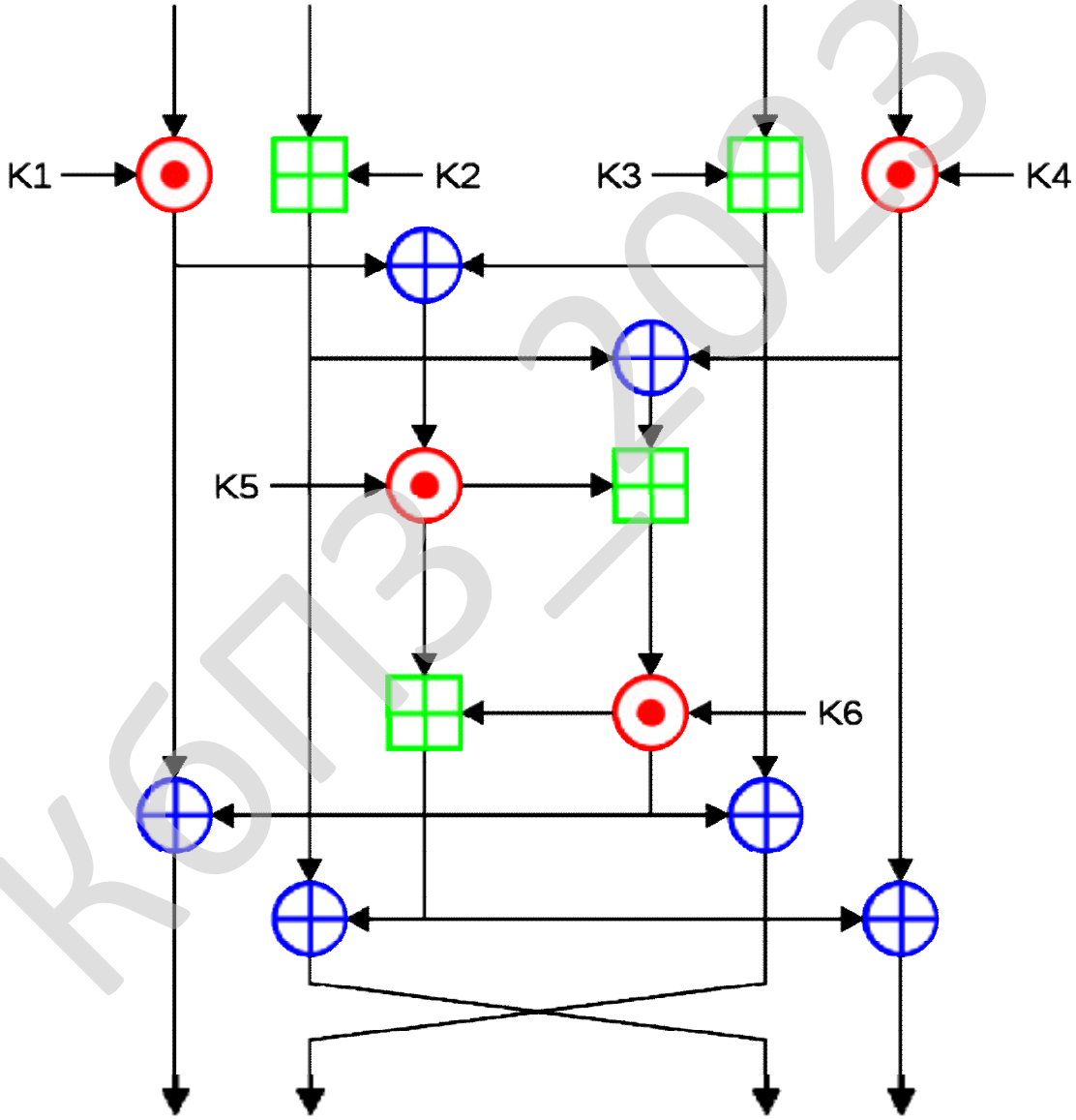

Рисунок 4.3 – Структура алгоритму IDEA

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів. Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню програми. Пункти меню: Файл; Налаштування; Формат; Довідкова.

– Бігунок зчитування даних S.M.A.R.T.

– Основний блок програми. Складається зі вкладок: Звіт (рисунок 5.1); Карта диску; Особливостей диску; Атрибутів S.M.A.R.T. (рисунок 5.2).

– Списку дисків системи з серійним номером.

– Кнопок: Збереження; оновлення; Очищення даних S.M.A.R.T.

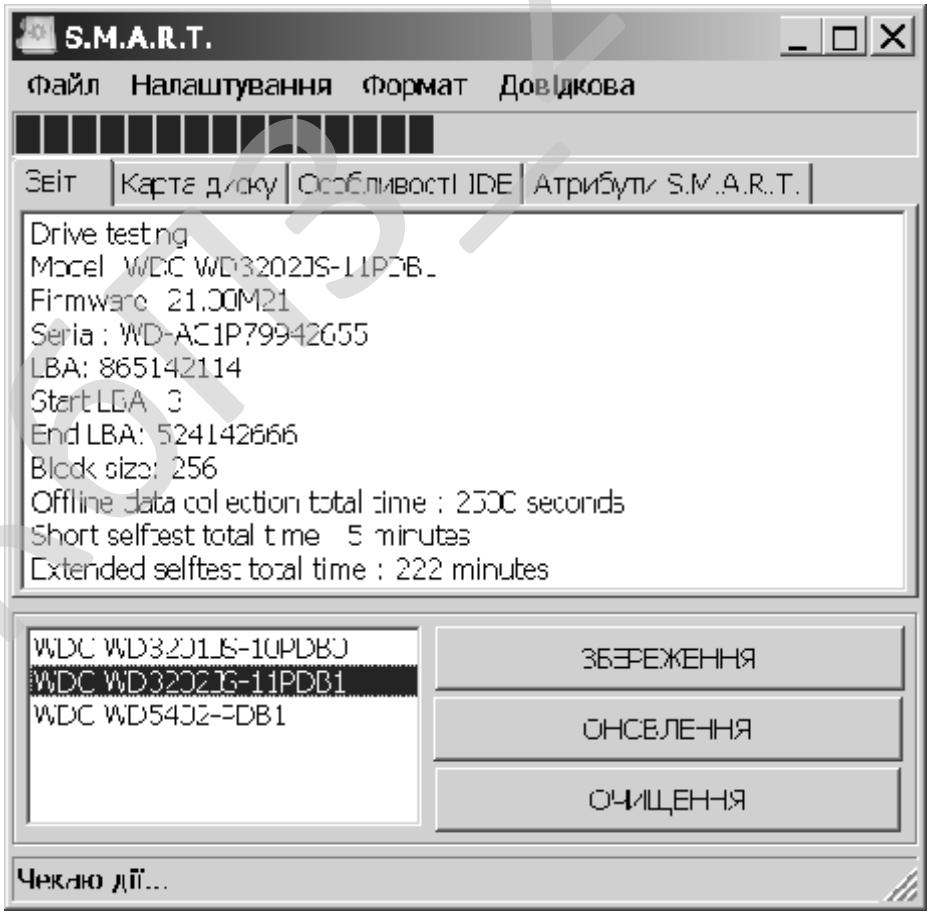

Рисунок 5.1 – Головне вікно програми, вкладка звіту

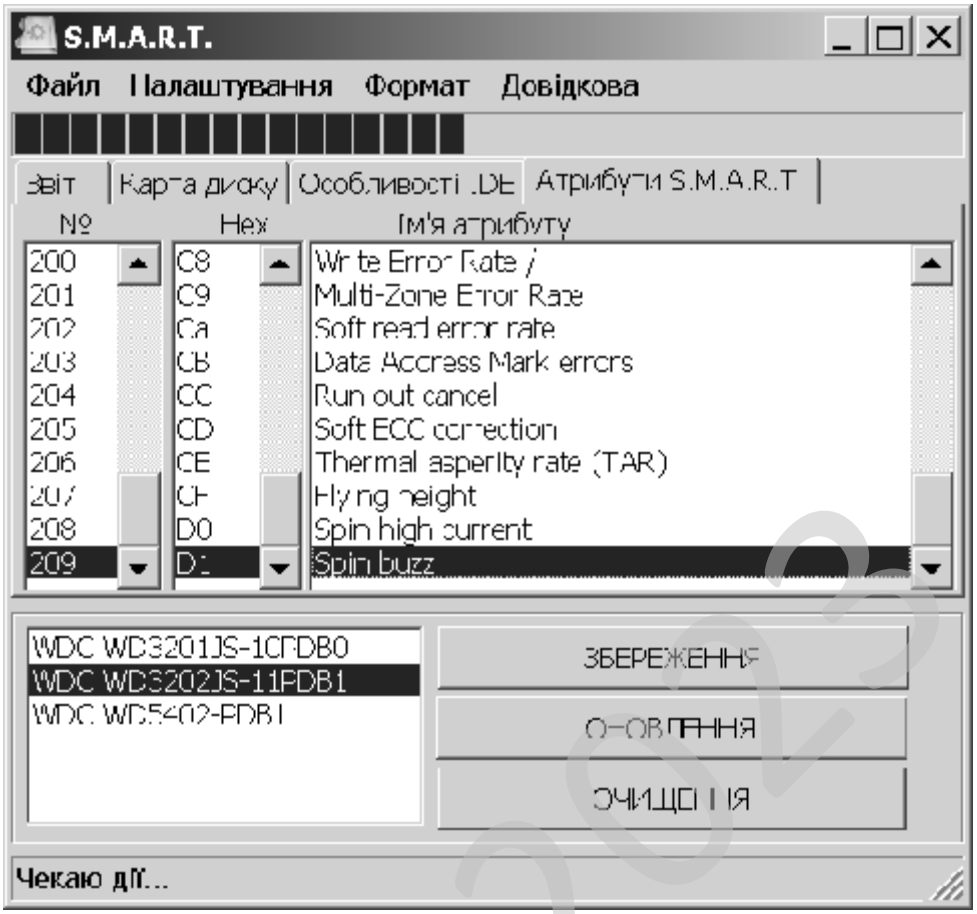

Рисунок 5.2 – Головне вікно програми, вкладка атрибутів S.M.A.R.T.

На рисунку 5.3 зображено форму авторського права. Обраний тип ліцензії – умовно безкоштовне розповсюдження.

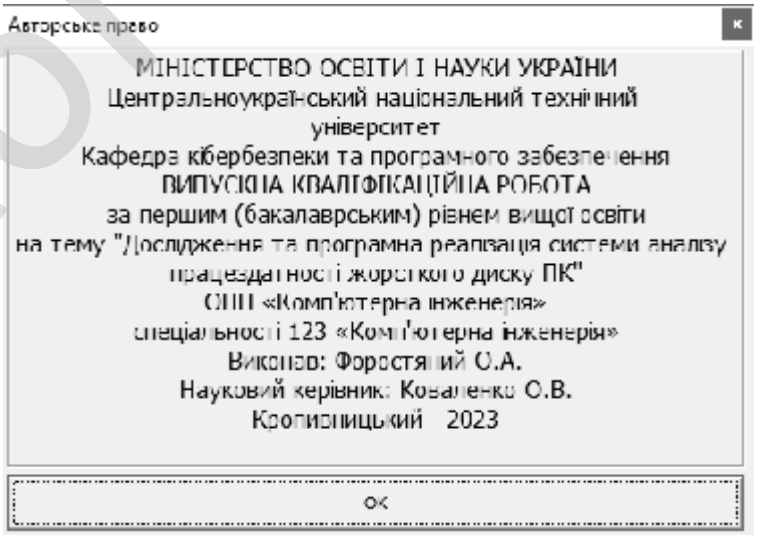

Рисунок 5.3 – Довідка розробника

6 НАУКОВА НОВИЗНА

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

Метою розробки є дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК.

Об'єктом дослідження є процес аналізу працездатності жорсткого диску ПК.

Предметом дослідження є методи аналізу працездатності жорсткого диску ПК.

Методи дослідження базуються на методах архітектури комп'ютерів, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод аналізу працездатності жорсткого диску ПК.

– Розроблено вітчизняний продукт аналізу працездатності жорсткого диску ПК, 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 аналізу працездатності жорсткого диску ПК.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	24
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	24000
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)%.

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

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	5	150	2,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	45,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф^{с}_{др} = \frac{45 \cdot 3}{1{,}2} = 112{,}5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 112,5/(60 \cdot 8) = 0,25 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	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	12450	37350
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	0,25	11500	8625
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	$R_{сп}$ = 7,3	-	$Ф_{роб}$ = 262800

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.

$$З_{сд} = \frac{262800}{7,3 \cdot 60} = 600 \; грн.$$

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

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

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

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Brain за 24.10.23 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	AMD Ryzen 5 4500 (100-000000644) AM4, 6 ядер, 12 потоків, 3.6 GHz, 4.1 GHz, (TDP) – 65 Вт, 7nm	-
Системна плата	GIGABYTE A520M H AM4, DDR4, LAN – 1 Гбіт/с, DVI, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Жорсткий диск	SSD M.2 2280 240GB Apacer 240 GB, TLC, M.2, SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot	-
Система охолодження	Deepcool BETA 10	-
Корпус	ProLogix E105 Minitower, Micro – ATX, Mini – ITX,	-
Блок живлення	Vinga 450W OEM	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	8	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

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

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	20	-
5. Транспортні засоби	143000	25	
6. Господарський інвентар	28000	20	-
Всього по групам 5, 6	176190	-	35238
7. Нематеріальні активи	24000	10	2400
Разом	$К_р$= 1733406		$А_р$= 170646

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

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

де: N_e – кількість екземплярів програм, шт.

$$З_о = 600 \cdot 209/24 = 5225 \text{ грн.}$$

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, %.

$$З_д = 5225 \cdot 10 \cdot 0{,}01 = 523 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(5225+523) = 1265 \text{ грн.}$$

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, %.

$$Г_{осп} = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

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

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

де: $З_{М1}$ – вартість паперу, грн.;

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

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

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм приймаємо 1/2 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п = 210$ грн., визначаємо вартість паперу за період розробки $N_м = 3$ міс:

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

$$З_{М1} = 210 \cdot 1/2 \cdot 3 = 315 \text{ грн.}$$

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

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 30 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 30 грн./шт.

$$З_{М2} = 30 \cdot 10 = 300 \text{ грн.}$$

Згідно виданих 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.

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

$$З_М = (315 + 300 + 1702)/24 = 97 \text{ грн.}$$

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

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

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

$$О_п = 5225 \cdot 15 \cdot 0{,}01 = 784 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 24 прим.):

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

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

$$А_м = 170646 \cdot 3/(24 \cdot 12) = 1777 \text{ грн.}$$

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

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

$$С_п = 5225+523+1265+784+97+784+1777 = 10455 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($Р_п$) 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, %.

$$П_р = 0{,}01 \cdot 50 \cdot 10455 = 5228 \text{ грн.}$$

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	$З_о$	5225
2. Додаткова зарплата виконавців	$З_д$	523
3. Відрахування на соціальні потреби	$С_{оц}$	1265
4. Загальногосподарські витрати	$Г_{осп}$	784
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	784
7. Амортизація основних фондів	$А_м$	1777
8. Повна собівартість програмного забезпечення	$С_п$	10455
9. Плановий прибуток	$П_р$	5228
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	15683
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3136,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	18819,6

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	–	18820
Всього капітальних витрат	–	18820

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.

$$З_о = Ч_е О_е М_е (1 + 0{,}01 Н_q)(1 + 0{,}01 Н_c), \qquad (7.23)$$

де $Ч_е$ – чисельність працівників, чол.;

$О_е$ – заробітна плата, грн./год;

$М_е$ – час, що витрачається на нарахування ЗП.

Після купівлі нового програмного забезпечення час на нарахування зменшився з 630 годин на рік до 393 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{о\,баз} = 1 \cdot 100 \cdot 630 \cdot 1{,}1 \cdot 1{,}22 = 84546\ грн,$$

до:

$$З_{о\,нов} = 1 \cdot 100 \cdot 393 \cdot 1{,}1 \cdot 1{,}22 = 52741\ грн.$$

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Заробітна плата(основна, додаткова, відрахування в соціальні фонди)	$З_р$	84546	52741
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	4705
Всього витрат за рік	$І$	84546	57446

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

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

Після впровадження ПЗ витрати на електроенергію не зміняться, тому: $З_{ел\ баз} = З_{ел\ нов}$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	18820	–	4705
Всього відрахувань	-	–	18820	–	4705

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.

$$Ев = (15683 - 10455) \cdot 24 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,25 \cdot 33190 + 0,2 \cdot 143000 + 0,1 \cdot 24000) \cdot 3/12 = 82396 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{325406}{(15683 - 10455) \cdot 24 \cdot 12 / 3} = 0,7 \text{ років}.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

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

$$E_{сп} = (84546\text{-}57446)\text{-}0,25 \cdot 18820 = 22395 \text{ грн.}$$

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.	24
2. Повна собівартість розробленої програми	Грн.	10455
3. Ціна розробленої програми	Грн.	15683
4. Плановий прибуток від реалізації розробленої програми	Грн.	5228
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1733406
7. Загальний прибуток від реалізації програмної продукції	Грн.	125472
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	82396
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,7
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	18820
11.Величина економічного ефекту у користувача програмної продукції	Грн.	22395
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,7

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

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

$$T_{сп} = \frac{18820}{84546 - 57446} = 0,7 \text{ років}.$$

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

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

Ці шкідливі фактори можуть привести до професійних захворювань.

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

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

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

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

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

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

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

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,4
Довжина	6
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8,1
Об'єм, V	$м^3$	не менше 20.0	22,7

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

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми

роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

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

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

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

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

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

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

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого

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

8.4 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці користувача ПК.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

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

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

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

8.5 Розрахунок занулення

Розрахунок занулення складається з трьох частин:

1. Розрахунок на відключаючу спроможність;

2. Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус;

3. Розрахунок робочого і повторного заземлювачів

Початкові дані

1. Потужність електроприладів, які підлягають зануленню:

$$P = 8 \text{ кВт.}$$

2. Довжина магістрального кабеля: L_M=32 м.

3. Довжина розгалудження (від розподільчого щита до електроприладів) : l=15 м.

4. Матеріал провідників кабеля−алюміній.

5. Лінійна напруга U=380 В.

6. Фазна напруга Uф=220 В.

Розрахунок проводиться відповідно до схеми електромережі, яка зображенана рис. 8.1:

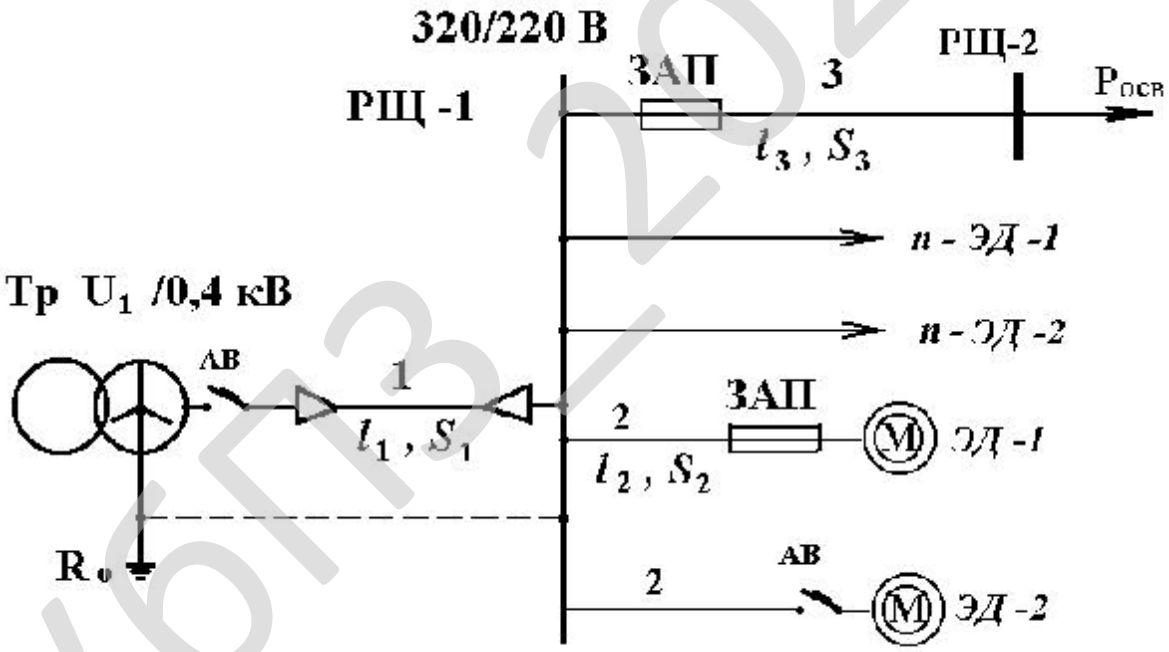

Рисунок 8.1 – Схема електромережі.

Тр – трансформатор; РЩ – розподільчий щит; АВ – автоматичний вимикач; ЗАП – запобіжник; 1 – магістральний кабель; 2 – розгалуження до електроприладів.

У результаті розрахунку отримали:

– Номінальна силу струму апарата захисту 25 А.

– Найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання 35 А.

– Площа перерізу магістрального кабеля (провідника) 6 мм$^{2.}$.

– Максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус 18,5 В.

Висновки до розділу

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

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.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного занулення, як одного з ключових факторів безпеки програміста. Розроблено заходи з охорони праці.

Список використаних джерел інформації

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

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

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

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

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

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

7. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення 19.09.22).

8. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

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

10. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу : http://dspace.kntu.kr.ua/jspui/handle/123456789/12240

9 ОСНОВНІ ВИСНОВКИ

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

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 аналізу працездатності жорсткого диску ПК.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем аналізу працездатності жорсткого диску ПК.

– Досліджена система аналізу працездатності жорсткого диску ПК.

– На основі отриманих результатів досліджень створена програмна реалізація системи аналізу працездатності жорсткого диску ПК.

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

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

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. 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.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм IDEA.

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 22395 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Форостяний О.А. Дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

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

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

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

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

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

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

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

9. 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.

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

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

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

13. 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.

14. 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.

15. 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.

16. 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.

17. 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

18. 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.

19. 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.

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

21. 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.

22. 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.

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

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

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

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

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

28. 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.

29. 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.

30. 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.

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

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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).

40. 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.

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

42. 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.

43. 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.

44. 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

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи аналізу працездатності жорсткого диску ПК.

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 аналізу працездатності жорсткого диску ПК.

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 аналізу працездатності жорсткого диску ПК;

– 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 RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

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 – 93 аркуші.

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 19.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Коваленко О.В.

Дослідження та програмна реалізація системи аналізу працездатності жорсткого диску ПК

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 55

Літера: РП

Кропивницький – 2023 року

ФАЙЛ МОДУЛЮ UNIT2.PAS

```
unit Unit2; // об'ява модулю

interface

uses windows;

type
   USHORT = Word;

const
MAX_IDE_DRIVES = 4;
// Максимальне число дисків, що приймають
// Primary / secondary, master / slave топологію

READ_ATTRIBUTE_BUFFER_SIZE = 512;
IDENTIFY_BUFFER_SIZE = 512;
READ_THRESHOLD_BUFFER_SIZE = 512;

// IOCTL команди
DFP_GET_VERSION = $ 00074080;
DFP_SEND_DRIVE_COMMAND = $ 0007C084;
DFP_RECEIVE_DRIVE_DATA = $ 0007C088;

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

// Біти, що повертаються в fCapabilities структури GETVERSIONOUTPARAMS
const
CAP_IDE_ID_FUNCTION = 1; // Підтримується команда ATA ID
CAP_IDE_ATAPI_ID = 2; // Підтримується команда ATAPI ID
CAP_IDE_EXECUTE_PCHardDrive_FUNCTION = 4; // Підтримуються команди PCHardDrive
// ------------------------------------------------ ----------------------
// Регістри IDE
// ------------------------------------------------ ----------------------
type
   TIDERegs = packed record
      bFeaturesReg: BYTE; // Використовується для визначення PCHardDrive "під
команди".
      bSectorCountReg: BYTE; // Регістр кількості секторів IDE
      bSectorNumberReg: BYTE; // Регістр номера сектора IDE
      bCylLowReg: BYTE; // Молодший розряд номера циліндра IDE
      bCylHighReg: BYTE; // Старший розряд номера циліндра IDE
      bDriveHeadReg: BYTE; // Регістр диска / головки IDE
      bCommandReg: BYTE; // Фактична команда IDE
      bReserved: BYTE; // Зарезервовано для
                                    // Майбутнього використання. Повинне бути
0.
   end;
   IDEREGS = TIDERegs;
   PIDERegs = ^ TIDERegs;
```

```
// Допустимі значення для параметра bCommandReg структури IDEREGS.
const
IDE_ATAPI_ID = $ A1; // Повертає ID сектора для ATAPI.
IDE_ID_FUNCTION = $ EC; // Повертає ID сектора для ATA.
IDE_EXECUTE_PCHardDrive_FUNCTION = $ B0;
 // Виконує команду PCHardDrive.
// Вимагає правильних значень для параметрів bFeaturesReg
// BCylLowReg, і bCylHighReg.
// Значення регістра циліндра потрібні при використанні команди PCHardDrive
PCHardDrive_CYL_LOW = $ 4F;
PCHardDrive_CYL_HI = $ C2;
// ------------------------------------------------ --------------------
// SENDCMDINPARAMS містить вхідні параметри для функції Send Command to Drive.
// ------------------------------------------------ --------------------
type
   TSendCmdInParams = packed record
      cBufferSize: DWORD; // Розмір буфера в байтах.
      irDriveRegs: TIDERegs; // Структура зі значеннями регістрів диска.
      bDriveNumber: BYTE; // Фізичний номер диска
                                          // команд (0,1,2,3).
      bReserved: array [0 .. 2] of Byte; // Зарезервовано для майбутнього
                                                        // розширення.
      dwReserved: array [0 .. 3] of DWORD; // Зарезервовано для майбутнього
                                                              //
використання.
      bBuffer: array [0 .. 0] of Byte; // Вхідний буфер.
   end;
   SENDCMDINPARAMS = TSendCmdInParams;
   PSendCmdInParams = ^ TSendCmdInParams;

// ------------------------------------------------ --------------------
// Стан повертане драйвером
// ------------------------------------------------ --------------------

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

// Значення bDriverError
const
PCHardDrive_NO_ERROR = 0; // Без помилок
PCHardDrive_IDE_ERROR = 1; // Помилка контролера IDE
PCHardDrive_INVALID_FLAG = 2; // Неправильний прапор команди
PCHardDrive_INVALID_COMMAND = 3; // Неправильний байт команди
PCHardDrive_INVALID_BUFFER = 4; // Неправильний буфер (нульовий, невірна адреса
і т.д.)
PCHardDrive_INVALID_DRIVE = 5; // Неправильний номер привода
PCHardDrive_INVALID_IOCTL = 6; // Неправильна IOCTL-команда
PCHardDrive_ERROR_NO_MEM = 7; // Неможливо захопити буфер користувача
PCHardDrive_INVALID_REGISTER = 8; // Деякі регістри IDE неправильні
PCHardDrive_NOT_SUPPORTED = 9; // Неприпустимий набір прапорів команди.
PCHardDrive_NO_IDE_DEVICE = 10; // Команда, послана пристрою не існує, хоча
номер диска правильний.
// Значення з 11 по 255 зарезервовані

// ------------------------------------------------ --------------------
// Структура, яка повертається PCHardDrive IOCTL для декількох команд
// ------------------------------------------------ --------------------
type
```

```
   TSendCmdOutParams = packed record
      cBufferSize: DWORD; // Розмір bBuffer в байтах
      DriverStatus: TDriverStatus; // Структура стану драйвера.
      bBuffer: array [0 .. 0] of BYTE; // Буфер, довільної довжини,
                                                        // для збереження
даних, прочитаних з диска.
   end;
   SENDCMDOUTPARAMS = TSendCmdOutParams;
   PSendCmdOutParams = ^ TSendCmdOutParams;
// ------------------------------------------------ ---------------------
// Константи для параметра bFeaturesReg структури TIDERegs для "під-команнд"
PCHardDrive
// ------------------------------------------------ ---------------------
const
PCHardDrive_READ_ATTRIBUTE_VALUES = $ D0;
PCHardDrive_READ_ATTRIBUTE_THRESHOLDS = $ D1;
PCHardDrive_ENABLE_DISABLE_ATTRIBUTE_AUTOSAVE = $ D2;
PCHardDrive_SAVE_ATTRIBUTE_VALUES = $ D3;
PCHardDrive_EXECUTE_OFFLINE_IMMEDIATE = $ D4
PCHardDrive_ENABLE_PCHardDrive_OPERATIONS = $ D8;
PCHardDrive_DISABLE_PCHardDrive_OPERATIONS = $ D9;
PCHardDrive_RETURN_PCHardDrive_STATUS = $ DA;
// ------------------------------------------------------------ ---------------------
// Наступний тип визначає структуру атрибутів диска
// ------------------------------------------------ ---------------------
type
   TDriveAttribute = packed record
      bAttrID: BYTE; // Ідентифікатор атрибуту
      wStatusFlags: WORD; // Прапори стану
      bAttrValue: BYTE; // Поточне нормалізоване значення
      bWorstValue: BYTE; // Найгірше значення
      bRawValue: array [0 .. 5] of BYTE; // ненормалізованного значення
      bReserved: BYTE; // Зарезервовано
   end;
   DRIVEATTRIBUTE = TDriveAttribute;
   PDriveAttribute = ^ TDriveAttribute;

// ------------------------------------------------ ---------------------
// Значення параметра wStatusFlags структури TDriveAttribute
// ------------------------------------------------ ---------------------
const
PRE_FAILURE_WARRANTY = $ 01; // Життєво-важливий
ON_LINE_COLLECTION = $ 02; //
PERFORMANCE_ATTRIBUTE = $ 04; // Атрибут, що відображає продуктивність диска
ERROR_RATE_ATTRIBUTE = $ 08; / / Атрибут, що відображає частоту появи помилок
EVENT_COUNT_ATTRIBUTE = $ 10; // Лічильник подій
SELF_PRESERVING_ATTRIBUTE = $ 20; // самозберігається атрибут

// ------------------------------------------------ ---------------------
// Наступна структура визначає структуру гарантійного порогу
// ------------------------------------------------ ---------------------
type
   TAttrThreshold = packed record
      bAttrID: BYTE; // Ідентифікатор атрибуту
      bWarrantyThreshold: BYTE; // Граничне значення
      bReserved: array [0 .. 9] of BYTE; // Зарезервовано
   end;
   ATTRTHRESHOLD = TAttrThreshold;
   PAttrThreshold = ^ TAttrThreshold;
// ------------------------------------------------ ---------------------
// Наступна структура визначає частину буфера IDENTIFY:
// ------------------------------------------------ ---------------------
   TIdSector = packed record
      wGenConfig: USHORT;
      wNumCyls: USHORT;
      wReserved: USHORT;
      wNumHeads: USHORT;
      wBytesPerTrack: USHORT;
      wBytesPerSector: USHORT;
```

```
      wSectorsPerTrack: USHORT;
      wVendorUnique: array [0 .. 2] of USHORT;
      sSerialNumber: array [0 .. 19] of CHAR;
      wBufferType: USHORT;
      wBufferSize: USHORT;
      wECCSize: USHORT;
      sFirmwareRev: array [0 .. 7] of CHAR;
      sModelNumber: array [0 .. 39] of CHAR;
      wMoreVendorUnique: USHORT;
      wDoubleWordIO: USHORT;
      wCapabilities: USHORT;
      wReserved1: USHORT;
      wPIOTiming: USHORT;
      wDMATiming: USHORT;
      wBS: USHORT;
      wNumCurrentCyls: USHORT;
      wNumCurrentHeads: USHORT;
      wNumCurrentSectorsPerTrack: USHORT;
      ulCurrentSectorCapacity: ULONG;
      wMultSectorStuff: USHORT;
      ulTotalAddressableSectors: ULONG;
      wSingleWordDMA: USHORT;
      wMultiWordDMA: USHORT;
      bReserved: array [0 .. 127] of BYTE;
   end;
   PIdSector = ^ TIdSector;
   IDSECTOR = TIdSector;
   _IDSECTOR = TIdSector;

const
NUM_ATTRIBUTE_STRUCTS = 30;

implementation

end.
```

ФАЙЛ ПРОЕКТУ РОЗРОБЛЕНОГО ПЗ PROGRAM PROJECT_PCHardDrive.DPR

```
program Project_PCHardDrive; // Назва ПЗ

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

{$R *.res} // Ресурси

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

ФАЙЛ МОДУЛЮ UNIT1.PAS

```
unit Unit1; // об'ява модулю

interface

uses
  Windows, PCHardDrive, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Grids;

type
  TForm1 = class (TForm)
    StringGrid1: TStringGrid;
    ComboBox1: TComboBox;
    Label1: TLabel;
    GroupBox1: TGroupBox;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox2: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label7: TLabel;
    GroupBox3: TGroupBox;
    Label9: TLabel;
    Label8: TLabel;
    procedure DoPrintData (pAttrBuffer: PCHAR; pThrsBuffer: PCHAR);
    procedure FormCreate (Sender: TObject);
    procedure FormClose (Sender: TObject; var Action: TCloseAction);
    procedure ComboBox1Change (Sender: TObject);
  private
    {Private declarations}
  public
    {Public declarations}
  end;

var
  Form1: TForm1;
  pAttrNames: TStringList;
  CurrentHandle: THandle;
  OSVersionInfo: TOSVersionInfo;

// Визначення глобальних буферів

AttrOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) +
(READ_ATTRIBUTE_BUFFER_SIZE-1)] of BYTE;
ThreshOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) +
(READ_THRESHOLD_BUFFER_SIZE-1)] of BYTE;
IdOutCmd: array [0 .. (sizeof (SENDCMDOUTPARAMS) -1) + (IDENTIFY_BUFFER_SIZE-1)]
of BYTE;

implementation

{$ R *. Dfm}

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду IDENTIFY
// * BDriveNum = 0-3
// * BIDCmd = IDE_ID_FUNCTION або IDE_ATAPI_ID
// ********************************************** ***************************

function DoIDENTIFY (hPCHardDriveIOCTL: THandle; pSCIP: PSENDCMDINPARAMS;
pSCOP: PSENDCMDOUTPARAMS; bIDCmd: BYTE; bDriveNum: BYTE): BOOL;
var
lpcbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = IDENTIFY_BUFFER_SIZE;
```

```
pSCIP.irDriveRegs.bFeaturesReg: = 0;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = 0;
pSCIP.irDriveRegs.bCylHighReg: = 0;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = bIDCmd;
// Команда може ідентифікувати IDE або ATAPI.
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду
PCHardDrive_ENABLE_PCHardDrive_OPERATIONS
// * BDriveNum = 0-3
// ********************************************** ***************************

function DoEnablePCHardDrive (hPCHardDriveIOCTL: THandle; pSCIP:
PSENDCMDINPARAMS; pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
lpcbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = 0;
pSCIP.irDriveRegs.bFeaturesReg: = PCHardDrive_ENABLE_PCHardDrive_OPERATIONS;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = PCHardDrive_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = PCHardDrive_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_PCHardDrive_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду PCHardDrive_READ_ATTRIBUTE_VALUES
// * BDriveNum = 0-3
// ********************************************** ***************************

function DoReadAttributesCmd (hPCHardDriveIOCTL: THandle; pSCIP:
PSENDCMDINPARAMS; pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
cbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = READ_ATTRIBUTE_BUFFER_SIZE;
pSCIP.irDriveRegs.bFeaturesReg: = PCHardDrive_READ_ATTRIBUTE_VALUES;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = PCHardDrive_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = PCHardDrive_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_PCHardDrive_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: =
end;

// ********************************************** ***************************
// *
// * Призначення: Посилає диску команду PCHardDrive_READ_ATTRIBUTE_THRESHOLDS
// * BDriveNum = 0-3

// ********************************************** ***************************
```

```
function DoReadThresholdsCmd (hPCHardDriveIOCTL: THandle; pSCIP:
PSENDCMDINPARAMS; pSCOP: PSENDCMDOUTPARAMS; bDriveNum: BYTE): BOOL;
var
cbBytesReturned: DWORD;
begin
pSCIP.cBufferSize: = READ_THRESHOLD_BUFFER_SIZE;
pSCIP.irDriveRegs.bFeaturesReg: = PCHardDrive_READ_ATTRIBUTE_THRESHOLDS;
pSCIP.irDriveRegs.bSectorCountReg: = 1;
pSCIP.irDriveRegs.bSectorNumberReg: = 1;
pSCIP.irDriveRegs.bCylLowReg: = PCHardDrive_CYL_LOW;
pSCIP.irDriveRegs.bCylHighReg: = PCHardDrive_CYL_HI;
pSCIP.irDriveRegs.bDriveHeadReg: = $ A0 or ((bDriveNum and 1) shl 4);
// Обчислюємо номер накопичувача.
pSCIP.irDriveRegs.bCommandReg: = IDE_EXECUTE_PCHardDrive_FUNCTION;
pSCIP.bDriveNumber: = bDriveNum;
result: = DeviceIoControl (hPCHardDriveIOCTL, DFP_RECEIVE_DRIVE_DATA, pSCIP,
sizeof (SENDCMDINPARAMS) -1, pSCOP, sizeof (SENDCMDOUTPARAMS) +
READ_THRESHOLD_BUFFER_SIZE-1, cbBytesReturned, nil);
end;

// ********************************************** **************************
// * DoPrintData
// *
// * FUNCTION: Відображає атрибути та порогові значення PCHardDrive
// ********************************************** **************************

procedure TForm1.DoPrintData (pAttrBuffer: PCHAR; pThrsBuffer: PCHAR);
var
i: integer;
pDA: PDRIVEATTRIBUTE;
pAT: PATTRTHRESHOLD;
begin
Label8.Caption: = 'Версія структури атрибутів:' + inttostr (WORD (pAttrBuffer
[0]));
Label9.Caption: = 'Версія структури порогових значень атрибутів:' + inttostr
(WORD (pThrsBuffer [0]));
// Виводимо інформацію: ідентифікатор і назву
// атрибута, його поточне і граничне значення

pDA: = PDRIVEATTRIBUTE (@pAttrBuffer [2]);
pAT: = PATTRTHRESHOLD (@pThrsBuffer [2]);
for i: = 0 to NUM_ATTRIBUTE_STRUCTS-1 do
begin
StringGrid1.Rows [i +1]. Strings [0]: = inttostr (pDA.bAttrID);
if (pDA.bAttrID = 194) or (pDA.bAttrID = 231) then Label7.Caption: =
'Температура:' + inttostr ((84 - (pDA.bAttrValue-1) div 3)) + # 176 + 'C';
StringGrid1.Rows [i +1]. Strings [1]: = pAttrNames [pDA.bAttrID];
StringGrid1.Rows [i +1]. Strings [2]: = inttostr (pDA.bAttrValue);
StringGrid1.Rows [i +1]. Strings [3]: = inttostr (pAT.bWarrantyThreshold);
StringGrid1.Rows [i +1]. Strings [4]: = inttostr (pDA.bWorstValue);
inc (pDA);
inc (pAT);
end;
end;

// Міняємо WORD-масив на BYTE-масив
procedure ChangeByteOrder (szString: PCHAR; uscStrSize: USHORT);
var
i: USHORT;
temp: CHAR;
begin
i: = 0;
while i <uscStrSize do
begin
temp: = szString [i];
szString [i]: = szString [i +1];
szString [i +1]: = temp;
i: = i + 2;
end;
```

```
end;

// ------------------------------------------------ ---------------------
// Відкриваємо дескриптор (хендл) PCHardDrive для операцій за допомогою
DeviceIoControl.
// ------------------------------------------------ ---------------------

function OpenPCHardDrive (DrvNum: Byte): THandle;
var
hPCHardDriveIOCTL: THandle;
begin
if OSVersionInfo.dwPlatformId = VER_PLATFORM_WIN32_NT then
begin
hPCHardDriveIOCTL: = CreateFile (PChar ('\ \. \ PhysicalDrive' + inttostr
(DrvNum)), GENERIC_READ or GENERIC_WRITE, FILE_SHARE_READ or FILE_SHARE_WRITE,
nil, OPEN_EXISTING, 0,0);
end
else // Windows 95 OSR2, Windows 98
begin
hPCHardDriveIOCTL: = CreateFile ('\ \. \ PCHardDriveVSD', 0,0, nil, CREATE_NEW,
0,0);
if hPCHardDriveIOCTL = INVALID_HANDLE_VALUE then
ShowMessage ('Неможливо відкрити PCHardDriveVSD, код помилки:' + inttostr
(GetLastError) + '-' + SysErrorMessage (GetLastError))
end;
result: = hPCHardDriveIOCTL;
end;

// ********************************************** *************************
// * DisplayIdInfo
// *
// * Відображає вміст ID буфера
// ********************************************** *************************
procedure DisplayIdInfo (pids: PIDSECTOR; bIDCmd: BYTE; bDriveNum: BYTE);
begin
if bIDCmd = IDE_ID_FUNCTION then
begin
Form1.Label5.Caption: = 'Диск' + inttostr (bDriveNum) + 'є IDE пристроєм, який
підтримує PCHardDrive';
Form1.Label6.Caption: = 'Циліндрів:' + inttostr (pids.wNumCyls) + '; голівок:' +
inttostr (pids.wNumHeads) + '; Секторів на доріжку:' + inttostr
(pids.wSectorsPerTrack);
end
else Form1.Label5.Caption: = 'Диск' + inttostr (bDriveNum) + 'є ATAPI
пристроєм.';
end;

function GetVersionPCHardDrive (hPCHardDriveIOCTL: THandle):
TGetVersionOutParams;
var
VersionParams: TGetVersionOutParams;
cbBytesReturned: DWORD;
begin
ZeroMemory (@VersionParams, sizeof (TGetVersionOutParams));
if not then ShowMessage (SysErrorMessage (GetLastError));
result: = VersionParams;
end;

procedure TForm1.FormCreate (Sender: TObject);
var
hPCHardDriveIOCTL: THandle;
i: integer;
VersionParams: TGetVersionOutParams;
bIDCmd, bDfpDriveMap: BYTE;
// Команда ідентифікації IDE або ATAPI
scip: TSendCmdInParams;
OutCmd: TSendCmdOutParams;
bSuccess: bool;
pids: PIDSECTOR;
```

```
begin
StringGrid1.ColWidths [0]: = 30;
StringGrid1.Cols [0]. Strings [0]: = 'ID';
StringGrid1.ColWidths [1]: = 260;
StringGrid1.Cols [1]. Strings [0]: = 'Назва атрибуту';
StringGrid1.ColWidths [2]: = 130;
StringGrid1.Cols [2]. Strings [0]: = 'Поточне значення';
StringGrid1.ColWidths [3]: = 130;
StringGrid1.Cols [3]. Strings [0]: = 'Граничне значення';
StringGrid1.ColWidths [4]: = 130;
StringGrid1.Cols [4]. Strings [0]: = 'Найгірше значення';
pAttrNames: = TStringList.Create;
pAttrNames.LoadFromFile (ExtractFilePath (Application.ExeName) +
'attributes.txt');

OSVersionInfo.dwOSVersionInfoSize: = SizeOf (OSVersionInfo);
GetVersionEx (OSVersionInfo);
bDfpDriveMap: = 0;
CurrentHandle: = OpenPCHardDrive (0);
// Отримуємо версію і т.п. PCHardDrive IOCTL
VersionParams: = GetVersionPCHardDrive (CurrentHandle);

case VersionParams.fCapabilities of
CAP_IDE_ID_FUNCTION: Label3.Caption: = 'Підтримується команда ATA ID';
CAP_IDE_ATAPI_ID: Label3.Caption: = 'Підтримується команда ATAPI ID';
CAP_IDE_EXECUTE_PCHardDrive_FUNCTION: Label3.Caption: =
'Підтримуються команди PCHardDrive';
CAP_IDE_ID_FUNCTION or CAP_IDE_ATAPI_ID or CAP_IDE_EXECUTE_PCHardDrive_FUNCTION:
Label3.Caption: =
'Підтримуються команди PCHardDrive, ATA ID і ATAPI ID';
CAP_IDE_ID_FUNCTION or CAP_IDE_EXECUTE_PCHardDrive_FUNCTION: Label3.Caption: =
'Підтримуються команди PCHardDrive і ATA ID ';
CAP_IDE_ATAPI_ID or CAP_IDE_EXECUTE_PCHardDrive_FUNCTION: Label3.Caption: =
'Підтримуються команди PCHardDrive і ATAPI ID ';
CAP_IDE_ID_FUNCTION or CAP_IDE_ATAPI_ID: Label3.Caption: =
'Підтримуються команди ATA ID і ATAPI ID';
end;
for i: = 0 to MAX_IDE_DRIVES-1 do
begin
hPCHardDriveIOCTL: = OpenPCHardDrive (i);
// Якщо пристрій з номером "i" - IDE, передаємо йому команди.
if VersionParams.bIDEDeviceMap shr i and 1 = 1 then
begin
// Ігноруємо ATAPI-пристрої.
if VersionParams.bIDEDeviceMap shr i and $ 10 = 0 then
begin
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@OutCmd, sizeof (OutCmd));
// Намагаємося активувати PCHardDrive.
if DoEnablePCHardDrive (hPCHardDriveIOCTL, @scip, @OutCmd, i) then
begin
Label4.Caption: = Label4.Caption + '' + inttostr (i) + ';';
// Позначаємо диск, як накопичувач з активним PCHardDrive
bDfpDriveMap: = bDfpDriveMap or (1 shl i);

// Тепер отримуємо ID сектора для всіх пристроїв IDE в системі.
// якщо пристрій - ATAPI, використовуємо команду IDE_ATAPI_ID,
// в іншому випадку використовуємо команду IDE_ID_FUNCTION.
if VersionParams.bIDEDeviceMap shr i and $ 10 = 1 then bIDCmd: = IDE_ATAPI_ID
else bIDCmd: = IDE_ID_FUNCTION;
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@IdOutCmd, sizeof (IdOutCmd));
if DoIDENTIFY (hPCHardDriveIOCTL, @scip, PSENDCMDOUTPARAMS (@IdOutCmd), bIDCmd,
i) then
begin
pids: = @PSENDCMDOUTPARAMS (@IdOutCmd). bBuffer;
if bIDCmd = IDE_ID_FUNCTION then ComboBox1.Items.Add (inttostr (i));
ChangeByteOrder (pids.sModelNumber, sizeof (pids.sModelNumber));
```

```
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sModelNumber;
ChangeByteOrder (pids.sFirmwareRev, sizeof (pids.sFirmwareRev));
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sFirmwareRev;
ChangeByteOrder (pids.sSerialNumber, sizeof (pids.sSerialNumber));
ComboBox1.Items.Strings [i]: = ComboBox1.Items.Strings [i] + '' +
pids.sSerialNumber;
end
else ShowMessage ('Команда Identify не виконана на диску:' + inttostr (i) + # 10
# 13 + 'DriverStatus: bDriverError =' + inttostr (PSENDCMDOUTPARAMS (@IdOutCmd).
DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (PSENDCMDOUTPARAMS
(@IdOutCmd). DriverStatus.bIDEStatus));
end
else ShowMessage ('Команда запуску PCHardDrive не виконана, диск:' + inttostr
(i) + # 10 # 13 + 'DriverStatus: bDriverError =' + inttostr
(OutCmd.DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (OutCmd.
DriverStatus.bIDEStatus));
end;
end;
end;
// Перебираємо всі можливі IDE-приводи і посилаємо
// команди тим, які підтримують PCHardDrive.
for i: = 0 to MAX_IDE_DRIVES-1 do
begin
if bDfpDriveMap shr i and 1 = 1 then
begin
ZeroMemory (@AttrOutCmd, sizeof (AttrOutCmd));
ZeroMemory (@ThreshOutCmd, sizeof (ThreshOutCmd));
bSuccess: = DoReadAttributesCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@AttrOutCmd), i);
if bSuccess = false then ShowMessage ('Помилка при виконанні команди читання
атрибутів PCHardDrive на диску:' + inttostr (i) + # 10 # 13 + 'DriverStatus:
bDriverError =' + inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd).
DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr (PSENDCMDOUTPARAMS
(@AttrOutCmd). DriverStatus.bIDEStatus))
// Команда читання атрибутів виконана успішно.
// Намагаємося прочитати порогові значення атрибутів.
else
if not DoReadThresholdsCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@ThreshOutCmd), i) then ShowMessage ('Помилка при виконанні команди читання
порогових значень атрибутів PCHardDrive на диску:' + inttostr (i) + # 10 # 13 +
'DriverStatus: bDriverError = '+ inttostr (PSENDCMDOUTPARAMS (@ThreshOutCmd).
DriverStatus.bDriverError) +', bIDEStatus = '+ inttostr (PSENDCMDOUTPARAMS
(@ThreshOutCmd). DriverStatus.bIDEStatus));
// Якщо функції DoReadAttributesCmd і DoReadThresholdsCmd виконані успішно,
// Процедура DoPrintData виводить обидва
// Значення атрибутів. Якщо DoReadThresholdsCmd
// Не підтримується, виводяться тільки поточні значення атрибутів
if bSuccess <> false then DoPrintData (@PSENDCMDOUTPARAMS (@AttrOutCmd).
bBuffer, @PSENDCMDOUTPARAMS (@ThreshOutCmd). bBuffer);
end;
end;
DisplayIdInfo (@PSENDCMDOUTPARAMS (@IdOutCmd). BBuffer, IDE_ID_FUNCTION, 0);
ComboBox1.ItemIndex: = 0;
end;

procedure TForm1.FormClose (Sender: TObject; var Action: TCloseAction);
begin
pAttrNames.Free;
// Закриваємо дескриптор (хендл) PCHardDrive.
CloseHandle (CurrentHandle);
end;

procedure TForm1.ComboBox1Change (Sender: TObject);
var
Num: string [1];
scip: TSendCmdInParams;
bSuccess: bool;
```

```
begin
Num: = ComboBox1.Items.Strings [ComboBox1.ItemIndex];
CurrentHandle: = OpenPCHardDrive (strtoint (Num));
ZeroMemory (@scip, sizeof (scip));
ZeroMemory (@IdOutCmd, sizeof (IdOutCmd));
DoIDENTIFY (CurrentHandle, @scip, PSENDCMDOUTPARAMS (@IdOutCmd),
IDE_ID_FUNCTION, strtoint (Num));
ZeroMemory (@AttrOutCmd, sizeof (AttrOutCmd));
ZeroMemory (@ThreshOutCmd, sizeof (ThreshOutCmd));
bSuccess: = DoReadAttributesCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@AttrOutCmd), strtoint (Num));
if bSuccess = false then ShowMessage ('Помилка при виконанні команди читання
атрибутів PCHardDrive на диску:' + Num + # 10 # 13 + 'DriverStatus: bDriverError
=' + inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd). DriverStatus.bDriverError) + ',
bIDEStatus = '+ inttostr (PSENDCMDOUTPARAMS (@AttrOutCmd).
DriverStatus.bIDEStatus))
// Команда читання атрибутів виконана успішно.
// Намагаємося прочитати порогові значення атрибутів.
else
if not DoReadThresholdsCmd (CurrentHandle, @scip, PSENDCMDOUTPARAMS
(@ThreshOutCmd), strtoint (Num)) then ShowMessage ('Помилка при виконанні
команди читання порогових значень атрибутів PCHardDrive на диску:' + Num + # 10
# 13 + 'DriverStatus: bDriverError =' + inttostr (PSENDCMDOUTPARAMS
(@ThreshOutCmd). DriverStatus.bDriverError) + ', bIDEStatus =' + inttostr
(PSENDCMDOUTPARAMS (@ThreshOutCmd). DriverStatus.bIDEStatus));
// Якщо функції DoReadAttributesCmd і DoReadThresholdsCmd виконані успішно,
// процедура DoPrintData виводить обидва значення атрибутів. Якщо
DoReadThresholdsCmd
// не підтримується, виводяться тільки поточні значення атрибутів
if bSuccess <> false then DoPrintData (@PSENDCMDOUTPARAMS (@AttrOutCmd).
bBuffer, @PSENDCMDOUTPARAMS (@ThreshOutCmd). bBuffer);
DisplayIdInfo (@PSENDCMDOUTPARAMS (@IdOutCmd). BBuffer, IDE_ID_FUNCTION,
strtoint (Num));
end;

// Об'єм жорсткого диска
function TForm1.GetTotalSpaceMb (Root: string): string;
var
  free, total: int64;
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = total div 1000000000;
   Mb: = (total mod 1000000000) div 1000000;
   Kb: = (total mod 1000000) div 1000;
   b: = total mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Обсяг вільного місця жорсткого диска
function TForm1.GetFreeSpaceMb (Root: string): string;
var
  free, total: int64;
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = free div 1000000000;
   Mb: = (free mod 1000000000) div 1000000;
   Kb: = (free mod 1000000) div 1000;
   b: = free mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Обсяг зайнятого місця жорсткого диска
function TForm1.GetFullSpaceMb (Root: string): string;
var
  free, total: int64;
```

```
  call: int64;
  Gb, Mb, Kb, b: integer;
begin
   GetDiskFreeSpaceEx (PChar (Root), free, total, @call);
   Gb: = (total - free) div 1000000000;
   Mb: = ((total - free) mod 1000000000) div 1000000;
   Kb: = ((total - free) mod 1000000) div 1000;
   b: = (total - free) mod 1000;
   Result: = Format ('% d% .3 d% .3 d% .3 d', [Gb, Mb, Kb, b]);
end;

// Як визначити тип файлової системи на вказаному диску
function TForm1.GetHDDFileSystem (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = FSName;
end;

// Мітка тому на вказаному диску
function TForm1.GetHDDLabel (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = VolumeName;
end;

// Зміна мітки томи на вказаному диску
// Перший аргумент - те, мітку якого потрібно змінити
// Другий аргумент - нова мітка
// SetVolumeLabel (PChar ('c: \'), PChar ('Win98'));

// Серійний номер тому
function TForm1.GetHDDSerialNumber (ADisk: char): String;
var
  SerialNum: dword;
  VolumeName, FSName: array [0 .. 255] of char;
  MaximumFNameLength,
  FileSystemFlags: dword;
begin
   Result: ='';
   if GetVolumeInformation (PChar (ADisk + ': \'),
                            VolumeName, SizeOf (VolumeName),
                            @SerialNum,
                            MaximumFNameLength,
                            FileSystemFlags,
                            FSName, SizeOf (FSName)) then
   Result: = Format ('% .8 x', [SerialNum]);
end;
```

```
// Загальна кількість кластерів на вказаному диску
function GetTotalNumberClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = TotalNumberClusters;
end;

// Кількість вільних кластерів на вказаному диску
function GetNumberOfFreeClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = NumberOfFreeClusters;
end;

// Кількість зайнятих кластерів на вказаному диску
function GetNumberOfFullClusters (Disk: char): Cardinal;
var
  SectorsPerCluster: Cardinal; // Кількість секторів в кластері
  BytesPerSector: Cardinal; // Кількість байт у секторі
  NumberOfFreeClusters: Cardinal; // Кількість вільних кластерів
  TotalNumberClusters: Cardinal; // Загальна кількість кластерів
begin
   GetDiskFreeSpace (PChar (Disk + ': \'),
                     SectorsPerCluster, BytesPerSector,
                     NumberOfFreeClusters, TotalNumberClusters);
   Result: = TotalNumberClusters - NumberOfFreeClusters;
end;

end.
```

ФАЙЛ МОДУЛЮ UNIT3.PAS

```
unit Unit3; // Об'ява модулю

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, magfmtdisk, FileCtrl, ComCtrls, magsubs1;

type
  TMainF2 = class(TForm)
    Log: TMemo;
    Panel1: TPanel;
    doChkDsk: TButton;
    doExit: TButton;
    DriveBox: TDriveComboBox;
    OptCorrectErrors: TCheckBox;
    OptVerbose: TCheckBox;
    OptCheckDirty: TCheckBox;
    OptScanDrive: TCheckBox;
    OptQuickFmt: TCheckBox;
    doAbort: TButton;
    doFrmtDsk: TButton;
    ProgressBar: TProgressBar;
    FileSystem: TComboBox;
    Label1: TLabel;
    VolumeLabel: TEdit;
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure doChkDskClick(Sender: TObject);
    procedure doExitClick(Sender: TObject);
    procedure doAbortClick(Sender: TObject);
    procedure doFrmtDskClick(Sender: TObject);
  private
    { Private declarations }
    procedure LogInfo (Info: String);
    procedure ProgressEvent (Percent: integer; var Cancel: boolean);
    procedure InfoEvent (Info: string; var Cancel: boolean);
  public
    { Public declarations }
  end;

var
  MainForm: TMainF2;
  MagFmtChkDsk: TMagFmtChkDsk;
  CancelFlag: boolean;

implementation

{$R *.dfm}

procedure TMainF2.FormDestroy(Sender: TObject);
begin
    FreeAndNil (MagFmtChkDsk);
end;

procedure TMainF2.FormCreate(Sender: TObject);
begin
    MagFmtChkDsk := TMagFmtChkDsk.Create (self);
    MagFmtChkDsk.onProgressEvent := ProgressEvent;
    MagFmtChkDsk.onInfoEvent := InfoEvent;
    if NOT IsProgAdmin then
end;

procedure TMainF2.LogInfo (Info: String);
begin
    Log.Lines.Add (Info);
```

```
end;

procedure TMainF2.ProgressEvent (Percent: integer; var Cancel: boolean);
begin
    ProgressBar.Position := Percent;
    Application.ProcessMessages;
    Cancel := CancelFlag;
end;

procedure TMainF2.InfoEvent (Info: string; var Cancel: boolean);
begin
    LogInfo (Info);
    Application.ProcessMessages;
    Cancel := CancelFlag;
end;

procedure TMainF2.doChkDskClick(Sender: TObject);
begin
    CancelFlag := false;
    doChkDsk.Enabled := false;
    doFrmtDsk.Enabled := false;
    try
        try
            ProgressBar.Position := 0;
if NOT MagFmtChkDsk.CheckDisk (DriveBox.Drive + ':\', OptCorrectErrors.Checked,
                OptVerbose.Checked, OptCheckDirty.Checked,
                OptScanDrive.Checked) then
LogInfo ('Check Disk Failed');
            ProgressBar.Position := 0;
            if MagFmtChkDsk.FileSysProblem then
LogInfo ('!!! Check Disk Found Problems with ' +
Uppercase (DriveBox.Drive) + ':');
        except
            on E:Exception do LogInfo ('Error: ' + E.Message);
        end;
    finally
        doChkDsk.Enabled := true;
        doFrmtDsk.Enabled := true;
        LogInfo ('');
    end;
end;

procedure TMainF2.doFrmtDskClick(Sender: TObject);
var
    MediaType: TMediaType;
begin
    if Trim (VolumeLabel.Text) = '' then
    begin
        exit;
    end;
    CancelFlag := false;
    doChkDsk.Enabled := false;
    doFrmtDsk.Enabled := false;
    MediaType := mtHardDisk;
    if UpperCase (DriveBox.Drive) < 'C' then MediaType := mtFloppy;
    try
        try
            ProgressBar.Position := 0;

            if NOT MagFmtChkDsk.DATADisk (DriveBox.Drive + ':\', MediaType,
                   TFileSystem (FileSystem.ItemIndex), Trim (VolumeLabel.Text),
                                                        OptQuickFmt.Checked, 0)
then
            ProgressBar.Position := 0;
        except
            on E:Exception do LogInfo ('Error: ' + E.Message);
        end;
    finally
        doChkDsk.Enabled := true;
```

```
        doFrmtDsk.Enabled := true;
        LogInfo ('');
    end;
end;

procedure TMainF2.doExitClick(Sender: TObject);
begin
    CancelFlag := true;
    Close;
end;

procedure TMainF2.doAbortClick(Sender: TObject);
begin
    CancelFlag := true;
end;


end.
```

ФАЙЛ МОДУЛЮ UNIT4.PAS

```
unit Unit4; // Об’ява модулю

interface

uses
  Windows, Messages, SysUtils, Classes;

const
    fmifs = 'fmifs.dll';
    WM_GETOBJ = WM_USER + 701;

// прапори
    FMIFS_HARDDISK = $0C;
    FMIFS_FLOPPY   = $08;

type
    TextOutput = record
          Lines:  DWORD;
          Output: PAnsiChar;  // unicode
    end;
   PTextOutput = ^TextOutput;

// Callback функція
    TCallBackCommand = (
        PROGRESS,
        DONEWITHSTRUCTURE,
      UNKNOWN2,
          UNKNOWN3,
      UNKNOWN4,
          UNKNOWN5,
      INSUFFICIENTRIGHTS,
          FSNOTSUPPORTED,
      VOLUMEINUSE,
      UNKNOWN9,
          UNKNOWNA,
      DONE,
          UNKNOWNC,
      UNKNOWND,
      OUTPUT,
        STRUCTUREPROGRESS,
        CLUSTERSIZETOOSMALL,
        UNKNOWN11,
        UNKNOWN12,
        UNKNOWN13,
        UNKNOWN14,
        UNKNOWN15,
        UNKNOWN16,
        UNKNOWN17,
        UNKNOWN18,
        PROGRESS2,
        UNKNOWN1A);
var

// Chkdsk процедура

Chkdsk: procedure (
    DriveRoot: PWCHAR;
    DATA: PWChar;
    CorrectErrors: BOOL;
      Verbose: BOOL;
      CheckOnlyIfDirty: BOOL;
      ScanDrive: BOOL;
      Unused2: DWORD;
      Unused3: DWORD;
      Callback: Pointer); stdcall;
```

```
DATAEx: procedure (
    DriveRoot: PWCHAR;
      MediaFlag: DWORD;
      DATA: PWCHAR;
      DiskLabel: PWCHAR;
      QuickDATA: BOOL;
      ClusterSize: DWORD;
      Callback: Pointer); stdcall;

// Enable/Disable функція

EnableVolumeCompession: function (
    DriveRoot: PWCHAR;
      Enable: BOOL): BOOLEAN; stdcall;

type

  TMediaType = (mtHardDisk, mtFloppy);
  TFileSystem = (fsNTFS, fsFAT, fsFAT32);
  TProgressEvent = Procedure (Percent: integer; var Cancel: boolean) of object;
  TInfoEvent = Procedure (Info: string; var Cancel: boolean) of object;

  TMagFmtChkDsk = class(TComponent)
  private
    { Private declarations }
    fProgressEvent: TProgressEvent;
    fInfoEvent: TInfoEvent;
    fDoneOK: boolean;
    fFileSysProblem: boolean;
    fFreeSpaceAlloc: boolean;
    fFirstErrorLine: string;
  protected
    { Protected declarations }
    function CheckDriveExists (const WDrive: WideString;
                               CheckInUse: boolean; var WDATA: WideString):
boolean;
    function doProgressEvent (const Percent: integer): boolean;
    function doInfoEvent (const Info: string): boolean;
    procedure WMGETOBJ (var msg: TMessage); message WM_GETOBJ;
  public
    { Public declarations }
    function LoadFmifs: boolean;
    function DATADisk (const DrvRoot: string; MediaType: TMediaType;
FileSystem: TFileSystem;
        const
DiskLabel: string; QuickDATA: boolean; ClusterSize: integer): boolean;
    function CheckDisk (const DrvRoot: string; CorrectErrors, Verbose,
                        CheckOnlyIfDirty, ScanDrive: boolean): boolean;
    function VolumeCompression (const DrvRoot: string; Enable: boolean):
boolean;
  published
    { Published declarations }
    property FileSysProblem: boolean          read fFileSysProblem;
    property FreeSpaceAlloc: boolean          read fFreeSpaceAlloc;
    property FirstErrorLine: string           read fFirstErrorLine;
    property onProgressEvent: TProgressEvent  read fProgressEvent write
fProgressEvent;
    property onInfoEvent:     TInfoEvent      read fInfoEvent     write
fInfoEvent;

  end;

  FmtChkException = class(Exception);

var
  MagFmifsib: THandle = 0;
  MagFmifs_Loaded: Boolean = false;   // DLL functions завантажено
  MagFmtObj: TObject;
```

```
implementation

procedure Register;
begin
  RegisterComponents('Samples', [TMagFmtChkDsk]);
end;

// об'ява

function DATACallback (Command: TCallBackCommand; SubAction: DWORD;
                                                  ActionInfo: Pointer): Boolean;
stdcall;
var
    flag: pboolean;
    percent: pinteger;
    toutput: PTextOutput;
    Obj: TObject;
    cancelflag: boolean;
    info: string;
    xlatbuf : AnsiString;
    progper, slen: integer;
begin
    result := true;
    cancelflag := false;
    Obj := TObject (SendMessage (HInstance, WM_GETOBJ, 0, 0));
    Obj := MagFmtObj;
    progper := -1;
    info := '';
    if NOT Assigned (TMagFmtChkDsk (Obj)) then exit;
    case Command of
        Progress:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Output:
            begin
                toutput := ActionInfo;
                slen := StrLen (toutput^.Output);
                SetLength (xlatbuf, slen);
                info := Trim (String (xlatBuf));
            end;
    if progper >= 0  then cancelflag :=
            TMagFmtChkDsk (Obj).doProgressEvent (progper);
    if info <> '' then cancelflag := TMagFmtChkDsk (Obj).doInfoEvent (info);
    result := NOT cancelflag;
end;

function ChkDskCallback (Command: TCallBackCommand; SubAction: DWORD;
                                                  ActionInfo: Pointer): Boolean;
stdcall;
var
    flag: pboolean;
    percent: pinteger;
    toutput: PTextOutput;
    Obj: TObject;
    info: string;
    progper, slen: integer;
    cancelflag: boolean;
    xlatbuf : AnsiString;
begin
    result := true;
    cancelflag := false;
    progper := -1;
    info := '';
    Obj := TObject (SendMessage (HInstance, WM_GETOBJ, 0, 0));
    Obj := MagFmtObj;
    if NOT Assigned (TMagFmtChkDsk (Obj)) then exit;
```

```
    case Command of
        Progress:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Progress2:
            begin
                percent := ActionInfo;
                progper := percent^;
            end;
        Output:
            begin
                toutput := ActionInfo;
                slen := StrLen (toutput^.Output);
                SetLength (xlatbuf, slen);
                OemToCharBuffA (PAnsiChar (toutput^.Output), PAnsiChar
                (xlatBuf), slen);
                info := Trim (String (xlatBuf));
                if (Pos ('found problems', info) > 0) or
                     (Pos ('Correcting errors', info) > 0) or
                       (Pos ('Errors found', info) > 0) or
                         (Pos ('(fix) option', info) > 0) then
                begin
                     TMagFmtChkDsk (Obj).fFileSysProblem := true;
                     if TMagFmtChkDsk (Obj).fFirstErrorLine = '' then
                                  TMagFmtChkDsk (Obj).fFirstErrorLine:=info;
                end;
                if (Pos ('free space marked as allocated', info) > 0) then
                begin
                     TMagFmtChkDsk (Obj).fFreeSpaceAlloc := true;
                     if TMagFmtChkDsk (Obj).fFirstErrorLine = '' then
                                  TMagFmtChkDsk(Obj).fFirstErrorLine:=info;
                end;
            end;
        Done:
            begin
                flag := ActionInfo;
                TMagFmtChkDsk (Obj).fDoneOK := flag^;
                if flag^ then
                  info := 'Check Disk: Finished OK'
                else
                    info := 'Check Disk: Unable to Finish';
            end;
          FSNotSupported: info := 'Check Disk: FS Not Supported';
      VolumeInUse: info := 'Check Disk: Volume In-Use';
        InsufficientRights: info := 'Check Disk: Insufficient Rights';
        else
            info := 'Check Disk Callback: ' + IntToStr (Ord (Command));
    end;
    if progper >= 0  then cancelflag:=
TMagFmtChkDsk (Obj).doProgressEvent (progper);
    if info <> '' then cancelflag := TMagFmtChkDsk (Obj).doInfoEvent (info);
    result:=NOT cancelflag;
end;

procedure TMagFmtChkDsk.WMGETOBJ (var msg: TMessage);
begin
    msg.Result := Integer (TMagFmtChkDsk);
end;

function TMagFmtChkDsk.doProgressEvent (const Percent: integer): boolean;
begin
    result := false;
    if Assigned (fProgressEvent) then fProgressEvent (Percent, result);
end;

function TMagFmtChkDsk.doInfoEvent (const Info: string): boolean;
begin
```

```
    result := false;
    if Assigned (fInfoEvent) then fInfoEvent (Info, result);
end;


function TMagFmtChkDsk.CheckDriveExists (const WDrive: WideString;
                           CheckInUse: boolean; var WDATA: WideString): boolean;
var
    FileSysName  : Array[0..MAX_PATH] of WChar;
    VolumeName   : Array[0..MAX_PATH] of WChar;
    maxcomlen, flags: longword;
    handle: THandle;
    voldev: WideString;
begin

    if NOT GetVolumeInformationW (PWChar (WDrive), VolumeName,
SizeOf(VolumeName) div 2,
                Nil, maxcomlen, flags, FileSysName, SizeOf(FileSysName) div 2)
then
    begin
        raise FmtChkException.Create('Drive Not Found: ' + WDrive);
        exit;
    end;
    WFormat := FileSysName;
    doInfoEvent (WDrive + ' Volume Label: ' + VolumeName + ', File System: ' +
FileSysName);

    // спроба отримання доступу до тому
    if CheckInUse then
    begin
        voldev := '\\.\' + WDrive [1] + ':';
        handle := CreateFileW (PWChar (voldev), Generic_Write, 0, nil,
Open_Existing, 0, 0);
        if handle = INVALID_HANDLE_VALUE then
        begin
            raise FmtChkException.Create('Drive In Use: ' + WDrive);
            exit;
        end;
        CloseHandle (handle);
    end;
    result := true;
end;

function TMagFmtChkDsk.DATADisk (const DrvRoot: string; MediaType: TMediaType;
                          FileSystem: TFileSystem; const DiskLabel: string;
                             QuickDATA: boolean; ClusterSize: integer):
boolean;
var
    wdrive, wformat, wfilesystem, wdisklabel: widestring;
    mediaflags, newsize: DWORD;

begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    wdisklabel := Uppercase (DiskLabel);
    if MediaType = mtHardDisk then
        mediaflags := FMIFS_HARDDISK
    else if MediaType = mtFloppy then
        mediaflags := FMIFS_FLOPPY
    else
        exit;
    if FileSystem = fsFAT then
        wfilesystem := 'FAT'
    else if FileSystem = fsFAT32 then
        wfilesystem := 'FAT32'
    else if FileSystem = fsNTFS then
        wfilesystem := 'NTFS'
    else
```

```
        exit;
    newsize := 0;
    if ((ClusterSize = 512) or (ClusterSize = 1024) or (ClusterSize = 2048) or
        (ClusterSize = 4096) or (ClusterSize = 8192) or (ClusterSize = 16384) or
            (ClusterSize = 32768) or (ClusterSize = 65536)) then newsize :=
ClusterSize;
    fDoneOK := false;
    if DiskSize (Ord (WDrive [1]) - 64) > 100 then
    begin
        doInfoEvent (WDrive + ' Checking Existing Drive Format');
        if NOT CheckDriveExists (wdrive, true, wformat) then exit;
        if wformat <> wfilesystem then QuickFormat := false;
    end
    else
    begin
        if (Length (WDrive) < 2) or (WDrive [2] <> ':') then
        begin
raise FmtChkException.Create('Invalid Drive Specification: ' + WDrive);
            exit;
        end;
                QuickDATA := false;
    end;
    MagFmtObj := Self;
    fFirstErrorLine := '';
    DATAEx (PWchar (wdrive), mediaflags, PWchar (wfilesystem), PWchar
(wdisklabel), QuickDATA, newsize, @DATACallback);
    result := fDoneOK;
    if NOT result then exit;
    doInfoEvent (WDrive + ' Checking New Drive DATA');
    if NOT CheckDriveExists (wdrive, false, wformat) then exit;
    doInfoEvent (WDrive + ' New Volume Space: ' + IntToStr (DiskFree (Ord
(WDrive [1]) - 64)));
end;

function TMagFmtChkDsk.CheckDisk (const DrvRoot: string; CorrectErrors, Verbose,
                                  CheckOnlyIfDirty, ScanDrive: boolean):
boolean;
var
    wdrive, wDATA: widestring;
begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    if NOT CheckDriveExists (wdrive, CorrectErrors, wDATA) then exit;
    MagFmtObj := Self;
    fDoneOK := false;
    fFileSysProblem := false;
    fFreeSpaceAlloc := false;
    fFirstErrorLine := '';
    Chkdsk (PWchar (wdrive), PWchar (wDATA), CorrectErrors, Verbose,
                             CheckOnlyIfDirty, ScanDrive, 0, 0,
@ChkDskCallback);
    if fFileSysProblem then
        result := true  // припинення при помилці
    else
        result := fDoneOK;
end;

function TMagFmtChkDsk.VolumeCompression (const DrvRoot: string; Enable:
boolean): boolean;
var
    wdrive, wDATA: widestring;
begin
    result := false;
    if NOT LoadFmifs then exit;
    wdrive := Uppercase (DrvRoot);
    if NOT CheckDriveExists (wdrive, true, wDATA) then exit;
    result := EnableVolumeCompession (PWchar (wdrive), Enable);
end;
```

```
function TMagFmtChkDsk.LoadFmifs: boolean;
begin
    result := Assigned (Chkdsk);
    if MagFmifs_Loaded then exit;
    result := false;
    if Win32Platform <> VER_PLATFORM_WIN32_NT then exit;

// завантаження бібліотеки
    result := false;
    MagFmifs_Loaded := True;
    MagFmifsib := LoadLibrary (fmifs);
    if MagFmifsib = 0 then exit;

// встановлення адрес функції у DLL
    Chkdsk := GetProcAddress (MagFmifsib, 'Chkdsk');
    DATAEx := GetProcAddress (MagFmifsib, 'DATAEx');
    EnableVolumeCompession := GetProcAddress (MagFmifsib,
                                              'EnableVolumeCompession');
    result := Assigned (Chkdsk);
end;

Initialization
    MagFmifsib := 0;
    MagFmifs_Loaded := false;
finalization
    if MagFmifs_Loaded then FreeLibrary (MagFmifsib);
end.
```

ФАЙЛ МОДУЛЮ UNIT5.PAS

```
unit Unit5; // Об’ява модулю

interface

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


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

const

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

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

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

```
function StripAllCntls (const AString: String): String;

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

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

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

function GetExceptMess (ExceptObject: TObject): string;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
    wSuiteMask: WORD;
    wProductType: BYTE;
    wReserved: BYTE;
  end;

// handle for DLL
var
    SensapiModule: THandle;

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

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

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

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

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

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

function SHEmptyRecycleBin; external shell32 name 'SHEmptyRecycleBinW';


implementation


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

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

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

```
  Result := Copy(S, 1, I);
end;

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

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

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

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

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

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

```
    until V = 0;
    while Digits > Length(Buf) - I do begin
       Dec(I);
       Buf[I] := $30;
    end;
    SetLength(Result, Length(Buf) - I);
    Move(Buf[I], Pointer(Result)^, Length(Buf) - I);
end;

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

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

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

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

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

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

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

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

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

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

```
    SHGetSpecialFolderLocation (HInstance, location, @PIDL);
      if SHGetPathFromIDList (PIDL, Path) then Result := Path;
end;

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

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

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

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

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

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

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

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

```
    end;
end;

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

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

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

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

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

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

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

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

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

function IsSpace (Ch: Char): Boolean;
begin
```

```
    Result := (Ch = ' ') or (Ch = Char($09));
end;

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

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

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

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

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

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

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

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

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


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

```
                result := copy (filename, 2, flen - 2)
            else
                result := '';
        end;
    end;
end;

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

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

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

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

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

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

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

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

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

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

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

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




function StripSpacesAnsi (const AString: AnsiString): AnsiString;
begin
```

```
    result := StripCharAnsi (AString, space);
end;

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

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

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

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

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

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

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

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

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

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

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

```
    result := S;
    StringFileTranEx (result);
end;

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

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

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

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

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

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

```
end;

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

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

    Result := 0;
  End;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
end;

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

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

function GetUAgeSizeFile (filename: string; var FileDT: TDateTime;
                                             var FSize: Int64): boolean;
var
   UTCFileTime: TFileTime;
begin
   Result := GetFUAgeSizeFile (filename, UTCFileTime, FSize);
   if Result then FileDT := FileTimeToDateTime (UTCFileTime);
end;

function TrimSpRight(const S: string): string;
var
  I: Integer;
begin
  I := Length(S);
  while (I > 0) and (S[I] = ' ') do Dec(I);
  Result := Copy(S, 1, I);
end;

function ExtractNameOnly(FileName: string): string;
var
  I: Integer;
begin
  FileName := ExtractFileName (FileName);  // remove path
  I := Length(FileName);
  while (I > 0) and not (IsPathSep (FileName[I])) do Dec(I);  // Unicode
  if (I = 0) or (FileName[I] <> '.') then I := MaxInt;
  Result := Copy(FileName, 1, I - 1);
end;

function GetExceptMess (ExceptObject: TObject): string;
var
  MsgPtr: PChar;
  MsgEnd: PChar;
  MsgLen: Integer;
  MessEnd: String;
begin
  MsgPtr := '';
  MsgEnd := '';
```

```
  if ExceptObject is Exception then
  begin
    MsgPtr := PChar(Exception(ExceptObject).Message);
    MsgLen := StrLen(MsgPtr);
    if (MsgLen <> 0) and (MsgPtr[MsgLen - 1] <> '.') then MsgEnd := '.';
  end;
  result := Trim (MsgPtr);
  MessEnd := Trim (MsgEnd);
  if Length (MessEnd) > 5 then result := result + ' - ' + MessEnd;
end;

function AscToInt64Ansi (value: AnsiString): Int64;
var
    E: Integer;
begin
    Val (value, result, E);
end;

function Str2LInt (const S: String): LongInt;
begin
    result := AscToInt (Trim (S));
end;

function Str2Byte (const S: String): Byte;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxByte then
        Result := MaxByte
    else if L < MinByte then
        Result := MinByte
    else
        Result := L;
end;

function Str2SInt (const S: String): ShortInt;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxShortInt then
        Result := MaxShortInt
    else if L < MinShortInt then
        Result := MinShortInt
    else
        Result := L;
end;

function Str2Int (const S: String): Integer;
begin
    result := Str2LInt (S);
end;


function Str2Word (const S: String): Word;
var
    L: LongInt;
begin
    L := Str2LInt (S);
    if L > MaxWord then
        Result := MaxWord
    else if L < MinWord then
        Result := MinWord
    else
        Result := L;
end;

function AddThouSeps (const S: string): string;
```

```
var
    LS, L2, I, N: Integer;
    Temp : string;
begin
    result := S;
    LS := Length (S);
    N := 1;
    if LS > 1 then
    begin
        if S [1] = '-' then
        begin
            N := 2;
            LS := LS - 1;
        end;
    end;
    if LS <= 3 then exit;
    L2 := (LS - 1) div 3;
    Temp := '';
    for I := 1 to L2 do
Temp := ThousandSeparator + Copy (S, LS - 3 * I + 1, 3) + Temp;
    Result := Copy (S, N, (LS - 1) mod 3 + 1) + Temp;
    if N > 1 then Result := '-' + Result;
end;

function IntToCStr (const N: integer): string;
begin
    result := AddThouSeps (IntToStr (N));
end;

function Int64ToCStr (const N: int64): string;
begin
    result := AddThouSeps (IntToStr (N));
end;

function AddThouSepsAnsi (const S: AnsiString): AnsiString;
var
    LS, L2, I, N: Integer;
    Temp : AnsiString;
begin
    result := S;
    LS := Length (S);
    N := 1;
    if LS > 1 then
    begin
        if S [1] = '-' then
        begin
            N := 2;
            LS := LS - 1;
        end;
    end;
    if LS <= 3 then exit;
    L2 := (LS - 1) div 3;
    Temp := '';
    for I := 1 to L2 do
Temp := ThousandSeparator + Copy (S, LS - 3 * I + 1, 3) + Temp;
    Result := Copy (S, N, (LS - 1) mod 3 + 1) + Temp;
    if N > 1 then Result := '-' + Result;
end;

function IntToCStrAnsi (const N: integer): AnsiString;
begin
    result := AddThouSepsAnsi (IntToStrAnsi (N));
end;

function Int64ToCStrAnsi (const N: int64): AnsiString;
begin
    result := AddThouSepsAnsi (IntToStrAnsi (N));
end;
```

```
function LInt2Str (const L: LongInt; const Len: Byte): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2EStr (const L: LongInt): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
end;

function LInt2ZBEStr (const L: LongInt): String;
begin
    if L = 0 then
        Result := ''
    else
        try
            Result := IntToStr (L);
        except
            Result := '';
    end;
end;

function FillStr (const Ch : Char; const N : Integer): string;
var
  I: integer;
begin
    SetLength (Result, N);
    for I := 1 to N do Result [I] := Char (Ch);
end;

function BlankStr (const N : Integer): string;
begin
    Result := FillStr (' ', N);
end;

function PadRightStr (const S : string; const Len : Integer): string;
var
    N: Integer;
begin
    N := Length (S);
    if N < Len then
        Result := S + BlankStr (Len - N)
    else
        Result := S;
end;

function PadLeftStr (const S : string; const Len : Integer): string;
var
    N: Integer;
begin
    N := Length (S);
    if N < Len then
        Result := BlankStr (Len - N) + S
    else
        Result := S;
end;

function PadChLeftStr (const S : string; const Ch : Char; const Len : Integer):
string;
var
```

```
    N: Integer;
begin
    N := Length (S);
    if N < Len then
        Result := FillStr (Ch, Len - N) + S
    else
        Result := S;
end;

function Int2StrZ (const L: LongInt; const Len: Byte): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), '0', Len);
end;

function Byte2Str (const L: LongInt; const Len: Byte): String;
begin
    try
        Result := IntToStr (L);
    except
        Result := '';
    end;
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2ZBStr (const L: LongInt; const Len: Byte): String;
begin
    Result := LInt2ZBEStr (L);
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2CStr (const L : LongInt; const Len : Byte): string;
begin
    Result := LInt2CEStr (L);
    Result := PadChLeftStr (CopyLeft (Result, Len), NumPadCh, Len);
end;

function LInt2CEStr (const L : LongInt): string;
begin
    try
        Result := AddThouSeps (IntToStr (L));
    except
        Result := '';
    end;
end;

function Int642CEStr (const L : Int64): string;
begin
    try
        Result := AddThouSeps (IntToStr (L));
    except
        Result := '';
    end;
end;

function Packed2Date (info: string): TDateTime;
var
    yy, mm, dd: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 8 then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 5, 2));
```

```
    dd := Str2Word (copy (info, 7, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
    if length (info) < 15 then exit;
    if info [9] <> '-' then exit;
    timeDT := Packed2Time (copy (info, 10, 10));
    if timeDT < 0 then exit;
    result := result + timeDT;
end;

function PackedISO2Date (info: string): TDateTime;
var
    yy, mm, dd: word;
    hh, nn, ss: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 10 then exit;
    if info [5] <> '-' then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 6, 2));
    dd := Str2Word (copy (info, 9, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
    if length (info) <> 19 then exit;
    if info [14] <> ':' then exit;
    hh := Str2Word (copy (info, 12, 2));
    nn := Str2Word (copy (info, 15, 2));
    ss := Str2Word (copy (info, 18, 2));
    if NOT TryEncodeTime (hh, nn, ss, 0, timeDT) then exit;
    result := result + timeDT;
end;

function PackedISO2UKStr (info: string): string;
begin
    result := '';
    info := trim (info);
    if length (info) < 10 then exit;
    if info [5] <> '-' then exit;
    result := copy (info, 9, 2) + '/' +copy(info, 6, 2)+'/'+copy(info, 1, 4);
    if length (info) <> 19 then exit;
    if info [14] <> ':' then exit;
    result := result + ' ' + copy (info, 12, 2) + ':' + copy (info, 15, 2);
    if copy (info, 18, 2) <> '00' then
    result := result + ':' + copy (info, 18, 2);
end;

function Packed2Secs (info: string): integer;
var
    len: integer;
begin
    result := 0;
    info := trim (info);
    len := length (info);
    if len < 4 then exit;
    while length (info) < 8 do info := '0' + info;
    if info [6] <> TimeSeparator then exit;
    result := AscToInt (copy (info, 1, 2)) * 60;
    result := (result + AscToInt (copy (info, 4, 2))) * 60;
    result := result + AscToInt (copy (info, 7, 2));
end;
```

```
function ConvLongDate (info: string): TDateTime;
var
    yy, mm, dd: word;
begin
    result := 0;
    info := trim (info);
    if length (info) <> 10 then exit;
    yy := Str2Word (copy (info, 1, 4));
    mm := Str2Word (copy (info, 6, 2));
    dd := Str2Word (copy (info, 9, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := -1;
        exit;
    end;
end;

function ConvUSADate (info: string): TDateTime;
var
    yy, mm, dd: word;
begin
    result := 0;
    info := trim (info);
    if length (info) <> 10 then exit;
    yy := Str2Word (copy (info, 7, 4));
    mm := Str2Word (copy (info, 1, 2));
    dd := Str2Word (copy (info, 4, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then result := 0;
end;

function ConvUKDate (info: string): TDateTime;
var
    yy, mm, dd: word;
    hh, nn, ss: word;
    timeDT: TDateTime;
begin
    result := 0;
    info := trim (info);
    if length (info) < 10 then exit;
    if info [3] <> '/' then exit;
    yy := Str2Word (copy (info, 7, 4));
    mm := Str2Word (copy (info, 4, 2));
    dd := Str2Word (copy (info, 1, 2));
    if NOT TryEncodeDate (yy, mm, dd, result) then
    begin
        result := 0;
        exit;
    end;
    if length (info) < 16 then exit;
    if info [14] <> ':' then exit;
    hh := Str2Word (copy (info, 12, 2));
    nn := Str2Word (copy (info, 15, 2));
    ss := 0;
    if length (info) >= 19 then ss := Str2Word (copy (info, 18, 2));
    if NOT TryEncodeTime (hh, nn, ss, 0, timeDT) then exit;
    result := result + timeDT;
end;

function TestTrgTick (const TrgTick: longword): boolean;
var
    curtick: longword;
begin
    result := false;
    if TrgTick = TriggerDisabled then exit;
disabled
    if TrgTick = TriggerImmediate then
    begin
        result := true;
        exit;
```

```
    end;
    curtick := GetTickCountX;
    if curtick <= MaxInteger then
    begin
        if curtick >= TrgTick then result := true;
        exit;
    end;
    if TrgTick <= MaxInteger then exit;
    if curtick >= TrgTick then result := true;
end;

procedure FreeAndNilEx(var Obj);
var
  Temp: TObject;
begin
  if Pointer(Obj) = Nil then exit;
  Temp := TObject(Obj);
  Pointer(Obj) := nil;
  Temp.Free;
end;

end.
```