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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0”

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

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 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

Золотухіну Богдану Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0*

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

Золотухін Б.Є. Дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0. 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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Метою розробки є дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Об'єктом дослідження є процес управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Предметом дослідження є методи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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

Результат роботи – програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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, USB 3.0

ABSTRACT

Zolotukhin B.E. Research and software implementation of the data management system of external hard drives with USB 3.0 interface. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the data management system of external hard drives with USB 3.0 interface.

The purpose of the development is research and software implementation of the data management system of external hard drives with USB 3.0 interface.

The object of the study is the data management process of external hard drives with USB 3.0 interface.

The subject of the study is methods of data management of external hard drives with USB 3.0 interface.

The research methods are based on the methods of personal computer architecture theory, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of the data management system for external hard drives with a USB 3.0 interface.

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, USB 3.0

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АПШ	–	асинхронні потокові шифри
ГПВП	–	генератор псевдовипадкових послідовностей
БШ	–	блокові шифри
ПШ	–	потокові шифри
РЗЛЗЗ	–	регістри зрушення з лінійним зворотним зв'язком
СПШ	–	синхронні потокові шифри

ВСТУП

Актуальність теми. Зовнішні жорсткі диски, флеш-накопичувачі (USB) і карти пам’яті – усі ці пристрої роблять резервне копіювання та обмін даними дуже простими.

Вони стають більш цінними, оскільки сучасне навчання, робота та життя переповнені даними.

Завдяки високій місткості, невеликому розміру та портативності вони є чудовими варіантами для передачі або перегляду даних з одного ПК на інший. Однак що станеться, якщо ви втратите або загубите будь-який із цих пристроїв?

У багатьох випадках це призведе до витоку даних. Цих порушень даних можна уникнути, якщо ви зашифруєте свої зовнішні жорсткі диски або USB-накопичувачі.

У разі шифрування хакерам важко отримати доступ до даних, які містять ці пристрої, якщо їх викрадуть або заблукають.

Зовнішні жорсткі диски, флеш-накопичувачі (USB) і карти пам’яті спрощують резервне копіювання та обмін даними. Однак якщо ви втратите або загубите будь-який із цих пристроїв, це може призвести до витоку даних.

Цих порушень даних можна уникнути, якщо ви зашифруєте свої зовнішні жорсткі диски або USB-накопичувачі. У разі шифрування хакерам важко отримати доступ до даних, які містять ці пристрої, якщо їх викрадуть або заблукають.

Проведені дослідження показали, що одним з найбільш перспективних напрямків управління даними з ціллю збереження конфіденційності інформації на зовнішніх носіях, зокрема на зовнішніх жорстких дисках інтерфейсу USB 3.0, є використання потокових шифрів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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

– Огляд існуючих систем управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– Дослідження системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– Програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Об'єктом дослідження є процес управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Предметом дослідження є методи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Методи дослідження базуються на методах теорії архітектури персональних 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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– Розроблено вітчизняний продукт управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0, 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

Призначанням системи є управління даними з ціллю збереження конфіденційності інформації на зовнішніх жорстких дисках інтерфейсу USB 3.0 із застосуванням потокових шифрів**.**

Те, що інтерфейс USB 2.0 уже застарілий, ніким сумніву не піддається. Пристроїв з USB 3.0 виходить усе більше й більше, а порти, що підтримують цю версію інтерфейсу, з'являються навіть у недорогих ноутбуках і материнських платах. І з технічної точки зору це повністю виправдано. Як показало тестування, USB 2.0 уже давно обмежував швидкість передачі даних у портативних HDD. Найчастіше накопичувачі працювали в 3-4 рази повільніше, ніж могли б. Історія з USB-флеш-дисками трохи інша, адже там швидкість запису або читання дуже рідко досягає поріг в 20 Мбайт/с, тому флешки не поспішають переводити на нову версію USB.

Ну й нарешті, USB 3.0 вирішив одну із застарілих проблем зовнішніх дисків, що часто нівелювала всю зручність від їхнього використання, – тепер ніяке зовнішнє живлення не потрібно, адже максимальний струм, що може забезпечити один роз’єм USB 3.0, дорівнює 900 мА, чого для роботи механіки накопичувача вистачає із запасом.

Розглянемо принцип роботи потокових шифрів, які застосуємо для збереження конфіденційності інформації на зовнішніх жорстких дисках інтерфейсу USB 3.0. Потоковий шифр виконує операції над бітами або символами (наприклад 8-, 16- або 32-бітовими). Потоковий шифр перетворить той самий символ відкритого тексту в різні символи шифртекста, наприклад залежно від того, скільки і які символи було оброблено раніше.

У багатьох потокових шифрах шифрування виробляється в такий спосіб. Генератор гами, заснований на генераторі псевдовипадкових чисел, видає послідовність бітів (гаму). Ця гама накладається на відкритий текст за допомогою побітової операції XOR. У результаті виходить шифртекст. Для розшифрування необхідно виконати в точності ту ж процедуру, тільки накласти гаму, отриману з використанням ідентичного генератора з точно таким же початковим станом, на шифртекст.

Таким чином, стійкість алгоритму залежить винятково від характеристик гами, видаваної генератором. Якщо гама складається з одних нулів (вирождений випадок), то дані при шифруванні взагалі не змінюються. Якщо гама має короткий період (наприклад 32 біта), то шифрування зводиться до операції XOR З 32-бітовою константою. Якщо ж гама являє собою випадковий набір бітів, що не підкоряється ніякій закономірності, виходить аналог одноразового шифрувального блокнота, що забезпечує абсолютний захист. Зрозуміло, детермінований алгоритм, використовуваний у генераторі гами, не може видавати істинно випадкову послідовність. Якщо послідовність не вдасться повторити, то не вдасться й розшифрувати повідомлення.

Якщо два повідомлення були зашифровані з використанням однієї й тієї ж гами й для одного з повідомлень (більше довгого) удалося яким-небудь образом одержати відкритий текст, то легко одержати відкритий текст і для іншого повідомлення. Застосувавши операцію XOR до відкритого тексту й шифртексту першого повідомлення, ми одержимо фрагмент гами. А наклавши гаму на шифртекст другого повідомлення, ми одержимо його відкритий текст. Саме тому не можна допускати, щоб та сама гама використовувалася при шифруванні двох різних потоків або повідомлень.

Якщо гама генерується незалежно від умісту повідомлення (як у наведеному раніше прикладі), то такий потоковий алгоритм називається синхронним. Як правило, у синхронних потокових шифрах ключ шифрування

використовується для установки початкового внутрішнього стану генератора гами.

У потокових шифрах, що самосинхронізуються, кожний біт гами залежить від фіксованого числа попередніх бітів шифртекста.

1.2 Область застосування

Областю застосування системи є зовнішні жорсткі диски інтерфейсу USB 3.0, у яких реалізується управління даними з ціллю збереження конфіденційності інформації. Включення системи управління даними з ціллю збереження конфіденційності інформації на зовнішніх жорстких дисках інтерфейсу USB 3.0 дозволяє захистити всі файли, що зберігаються на диску. На відміну від файлової системи EFS, що дозволяє шифрувати окремі файли, програмне забезпечення, що розробляється під час виконання дипломного проектування, шифрує весь диск цілком. Користувач може входити в систему й працювати з файлами як звичайно, а програмне забезпечення, що розробляється під час виконання дипломного проектування, буде перешкоджати хакерам, що намагається одержати доступ до системних файлів для пошуку паролів, а також до жорсткого диску шляхом добування його з даного комп'ютера й установки в іншій. Програмне забезпечення, що розробляється під час виконання дипломного проектування, може захищати тільки файли, що зберігаються на зовнішньому диску. Програмне забезпечення, автоматично шифрує всі файли, що додаються в розділ, що захищається. Файли залишаються зашифрованими тільки при зберіганні на зашифрованому диску. При їхньому копіюванні на незашифрований диск або інший комп'ютер вони не будуть прочитані. При наданні загального доступу до файлів по мережі вони будуть зашифровані на зашифрованому диску, але авторизовані користувачі зможуть одержувати до них доступ звичайним образом.

Якщо при завантаженні комп'ютера програмне забезпечення, що розробляється під час виконання дипломного проектування, виявить можливу

погрозу безпеки (наприклад помилки диску, зміни BIOS або файлів завантаження), розділ буде заблокований і для його розблокування буде потрібно особливий пароль відновлення. Не забудьте створити пароль відновлення при першому запуску програмного забезпечення, що розробляється під час виконання дипломного проектування. У противному випадку доступ до файлів може бути загублений. Для зберігання ключів, які використовуються для розблокування зашифрованого жорсткого диску, програмне забезпечення, що розробляється під час виконання дипломного проектування, звичайно використовує встановлену в комп'ютері мікросхему довіреного платформного модуля (TPM). Коли користувач входить у систему, програмне забезпечення запитує в TPM ключі для доступу до жорсткого диску й розблокує його. Оскільки відразу після входу користувача в систему TPM надає програмному забезпеченню ключі, безпека комп'ютера залежить від надійності пароля входу в систему. Якщо є надійний пароль, що запобігає вхід у систему сторонніх осіб, то захищений за допомогою програмного забезпечення жорсткий диск залишиться заблокованим.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0, 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

Шифрування зовнішнього жорсткого диска в Windows

Виконайте наведені нижче дії, щоб зашифрувати зовнішній жорсткий диск у Windows за допомогою BitLocker.

Крок 1. Відкрийте Провідник Windows і клацніть правою кнопкою миші зовнішній жорсткий диск. Потім натисніть Формат.

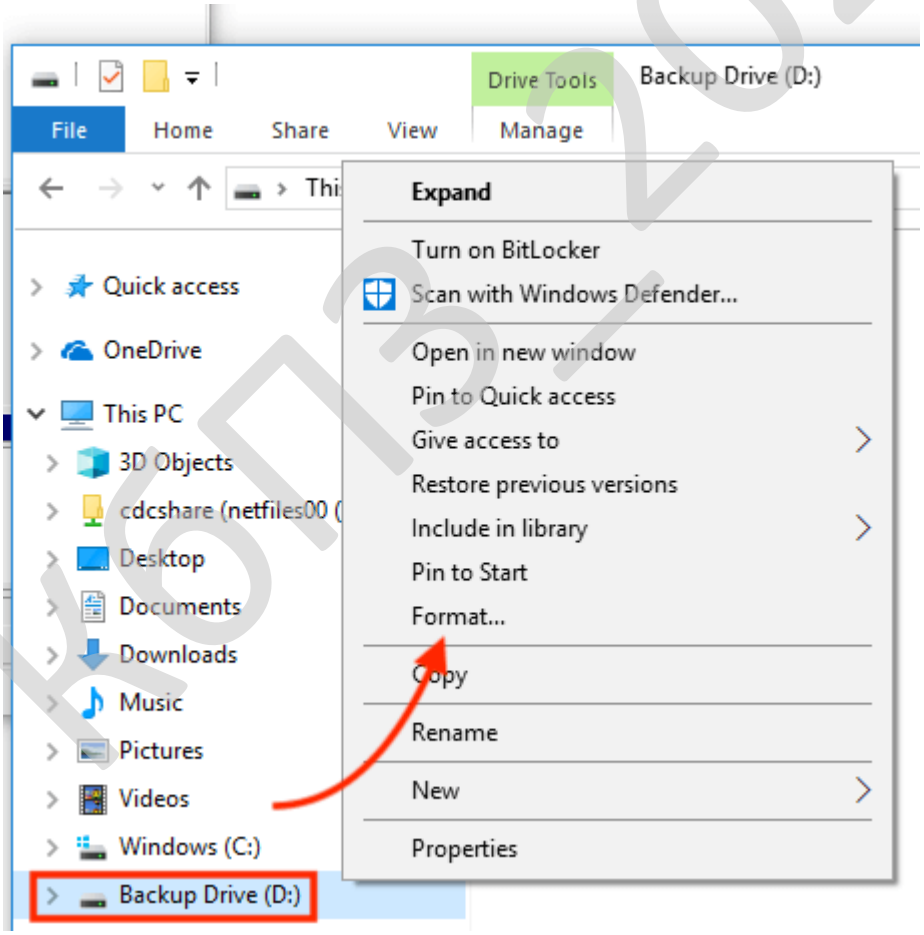


Рисунок 2.1 – Інтнрфейс користувача BitLocker

Крок 2. Вищезазначений крок повністю видалить усі дані на зовнішньому диску. Якщо це прийнятно для вас, натисніть OK.

Крок 3. З’явиться наступне вікно. Переконайтеся, що в розділі «Файлова система» вибрано NTFS, а потім натисніть «Пуск».

Крок 4: Після успішного форматування диска; з'явиться таке повідомлення. Натисніть «ОК», щоб продовжити наступні дії.

Крок 5. Знову відкрийте Провідник Windows і клацніть правою кнопкою миші зовнішній жорсткий диск. Потім виберіть «Увімкнути BitLocker» зі списку параметрів.

Крок 6. Поставте прапорець біля опції «Використовувати пароль, щоб розблокувати диск» у вікні, що з’явиться, а також введіть там надійний пароль, який запам’ятовується, якщо потрібно. Потім натисніть «Далі».

Крок 7: Натисніть «Зберегти у файл» у наступному вікні та виберіть місце для збереження ключа відновлення. Цей ключ відновлення стане в нагоді для доступу до накопичувача, якщо ви забудете свій пароль.

Крок 8: Прочитайте текст у вікні нижче. Потім виберіть потрібний варіант шифрування диска та натисніть «Далі».

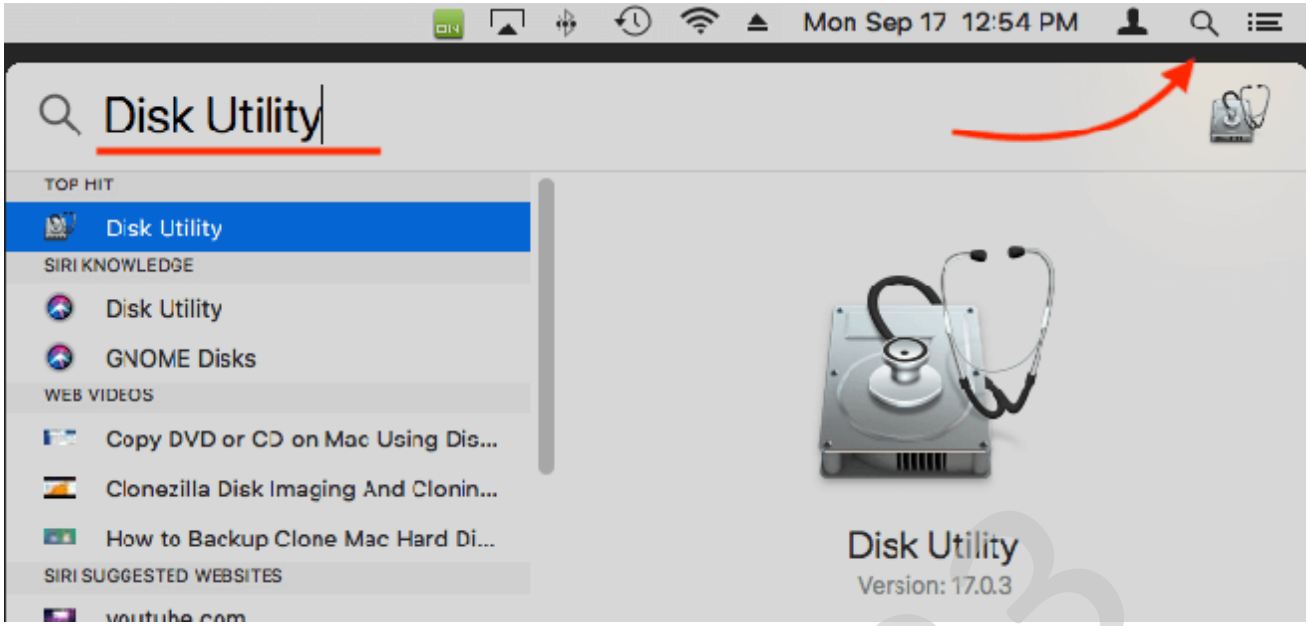
Крок 9. Натисніть «Почати шифрування».

Крок 10: Нарешті, вам доведеться дочекатися завершення шифрування, а потім натиснути «Закрити».

Шифрування зовнішнього жорсткого диска в macOS

Для цього потрібно зашифрувати зовнішній жорсткий диск у macOS за допомогою FileVault.

Крок 1: спочатку відкрийте Finder і отримайте Disk Utility, ввівши там «Disk Utility» і натиснувши enter. Потім запуститься програма Disk Utility, як показано нижче.



Рисунко 2.2 – Інтерфейс користувача Disk Utility

Крок 2. Виберіть свій зовнішній диск у Disk Utility, а потім натисніть «Стерти».

Крок 3. Клацніть розкривне меню «Формат» і виберіть у списку доступних форматів Mac OS Extended (Journaled, Encrypted) або APFS (Encrypted).

Крок 4. Введіть надійну та запам’ятовувану парольну фразу та підказку до парольної фрази, якщо потрібно, потім натисніть «Вибрати», а потім натисніть «Стерти».

Крок 5: Тепер необхідно відформатувати диск для FileVault.

Крок 6: Тепер настає етап шифрування диска для FileVault. Ви зможете помітити значок на робочому столі, схожий на зображення нижче.

Крок 7: Клацніть правою кнопкою миші на піктограмі зовнішнього диска, який потрібно зашифрувати, а потім виберіть опцію під назвою «Шифрувати «Машину часу», як показано нижче.

Крок 8: Введіть надійний пароль, який легко запам’ятовується, і введіть підказку до пароля, якщо потрібно. Після цього натисніть Зашифрувати диск.

Крок 9: Нарешті, диск продовжить процес шифрування. Виберіть і

перевірте диск у Disk Utility, щоб переконатися, що він успішно зашифрований.

Шифрування зовнішнього жорсткого диска в Linux

У Linux існує багато способів шифрування. Нижче наведено кроки для шифрування зовнішнього жорсткого диска в Linux за допомогою Raspberry Pi.

Крок 1: Підключіть жорсткий диск до системи та виконайте таку команду.

sudo fdisk -l

Крок 2. Використовуйте наведену нижче команду, щоб подрібнити жорсткий диск, якщо це традиційний шпиндельний диск і є дані.

sudo shred -v -n 1 /dev/sda

Крок 3. Виконайте наведену нижче команду, щоб видалити існуючі розділи та почати з нового основного розділу.

sudo fdisk /dev/sda

Крок 4. Вам буде запропоновано кілька запитань у інструменті **fdisk** . Дайте відповіді на них належним чином, і ви автоматично вийдете з інструменту fdisk. Тепер виконайте наступну команду.

sudo mkfs.ext4 /dev/sda1

Крок 5. Занотуйте свій UUID, оскільки він знадобиться для монтування зашифрованого диска пізніше. Ви можете використовувати **blkid** , щоб знайти його, як показано нижче.

sudo blkid

Крок 6: Якщо ви ще не встановили інструмент шифрування, запустіть **modprobe** наприкінці, щоб завантажити необхідні модулі без перезапуску.

sudo apt-get install cryptsetup sudo modprobe dm-crypt sha256 aes

Крок 7. Тепер налаштуйте свій зашифрований диск. Для цього вам потрібно буде ввести парольну фразу. Занотуйте його в безпечне місце, оскільки він знадобиться вам пізніше, щоб отримати доступ до зашифрованих даних або налаштувати зашифрований жорсткий диск.

sudo cryptsetup –verify-passphrase luksFormat /dev/sda1 -c aes -s 256 -h

sha256 sudo cryptsetup luksOpen /dev/sda1 YourPreferredName

Наведений вище код створить програму відображення, яка розпізнає ваш зашифрований том, і в цьому випадку він називається «YourPreferredName».

Шифрування жорсткого диска за допомогою стороннього програмного забезпечення для шифрування жорстких дисків і дисків

Іншим способом шифрування жорсткого диска є використання стороннього програмного забезпечення або інструменту шифрування. Нижче наведено деякі з найпопулярніших сторонніх програм шифрування, які використовуються для шифрування жорсткого диска.

VeraCrypt

VeraCrypt – дуже популярна програма, яка часто отримує найвищі оцінки від користувачів. VeraCrypt – це безкоштовне програмне забезпечення для шифрування з відкритим кодом, яке використовується в Windows, Mac OS X і Linux. Він забезпечує безпеку корпоративного рівня для всіх ваших важливих даних.

AES Crypt

AES Crypt – це зручна програма для шифрування, яка має як безкоштовну, так і платну версії. Він поставляється з менеджером паролів і функцією співпраці, що дозволяє обмінюватися зашифрованими даними з іншими. AES Crypt доступний для Windows, macOS і Linux, а версії сторонніх розробників – для Android і iOS.

AxCrypt

AxCrypt – це дуже потужне та надійне програмне забезпечення для шифрування, спеціально розроблене для окремих осіб і команд малого бізнесу. За допомогою AxCrypt файли можна захистити за допомогою 128- або 256-бітного шифрування AES. Він постачається з усіма інструментами для захисту файлів на зовнішніх жорстких дисках.

DiskCryptor

Diskcryptor – одне з найкращих рішень для шифрування даних із

відкритим кодом для Microsoft Windows. Він дозволяє шифрувати цілі жорсткі диски або окремі розділи. DiskCryptor підтримує багато схем шифрування, операційні системи Windows і файлові системи. Вражаючою особливістю DiskCryptor є те, що він дозволяє призупинити шифрування та відновити його пізніше або навіть на іншому комп'ютері.

TrueCrypt

TrueCrypt – потужне програмне забезпечення для шифрування, яке підтримує приховані томи, шифрування на льоту, ключові файли та комбінації клавіш. Його можна використовувати на цілих дисках і системних розділах із встановленою ОС.

Більше того, TrueCrypt можна використовувати для створення єдиного файлу, який працює як диск із зашифрованими папками та файлами. TrueCrypt підтримує Windows 10, 8, 7, Vista, XP і Linux, а також macOS.

Жорсткі диски з апаратним шифруванням

У цих пристроях зазвичай використовується програмне та апаратне шифрування, яке іноді потребує встановлення пароля на фізичній клавіатурі для захисту ваших даних. Але вони також залежать від власного коду, що ускладнює перевірку їхніх заяв про безпеку.

Неможливо переконатися, чи жорсткі диски з апаратним шифруванням мають бекдори чи ні, і це типово для будь-якого обладнання. Тому життєво важливо купувати жорсткі диски лише в перевірених постачальників або брендів.

Шифрування USB з операційними системами

Тут ви дізнаєтесь, як зашифрувати флешку.

Шифрування USB-накопичувачів у Windows

Виконайте наведені нижче дії, щоб зашифрувати флеш-накопичувач USB в ОС Windows за допомогою BitLocker.

Крок 1. Підключіть USB-накопичувач до комп'ютера з Windows і відкрийте файловий провідник. Клацніть там правою кнопкою миші свій USB-накопичувач і виберіть «Увімкнути BitLocker».

Крок 2. Тут вам буде запропоновано вибрати спосіб розблокування диска. Поставте галочку біля пункту «Використовувати пароль, щоб розблокувати цей диск», як показано нижче.

Крок 3: Введіть і підтвердьте свій пароль у відповідних місцях вікна вище, щоб розблокувати диск (можна змінити цей пароль пізніше, надавши оригінальний пароль).

Крок 4. Коли вас запитають «Як ви хочете зберегти ключ відновлення», виберіть опцію «Зберегти ключ відновлення у файл». Збережіть цей файл відновлення в безпечному місці, доступ до якого є лише у вас (цей файл не міститиме вашого пароля, і його можна використовувати, якщо ви забудете пароль).

Крок 5: виберіть шифрування всього диска або шифрування лише використаного простору.

Крок 6: Нарешті, Windows успішно зашифрує вашу флешку. Цей процес займе всього кілька хвилин, і після завершення шифрування з'явиться повідомлення.

Шифрування USB-накопичувачів у macOS

Як ви, можливо, вже знаєте, macOS має деякі вбудовані інструменти для шифрування USB, інтегровані в програмне забезпечення. Вони дозволяють шифрувати або розшифровувати флеш-накопичувачі USB та інші пристрої зберігання на льоту. Нижче наведено кілька простих кроків щодо їх використання.

Цей процес використовує шифрування XTS-AES, той самий тип шифрування, який використовує система macOS FileVault 2. Важливо пам'ятати, що використання Finder для шифрування USB-накопичувача обмежить його використання macOS.

Ви не зможете отримати доступ до диска на машині з іншою ОС, наприклад Windows або Linux.

Крок 1. Підключіть флеш-накопичувач до ПК MAC і відкрийте

Finder. Потім клацніть правою кнопкою миші USB-накопичувач на лівій бічній панелі та натисніть «Шифрувати + ім’я USB-накопичувача», як показано нижче.

Заповнивши всі необхідні поля, натисніть «Зашифрувати диск». **Крок 2:** Тепер Finder запропонує вам ввести пароль і підказку. Цей пароль використовуватиметься для доступу до USB-накопичувача пізніше, тому не втрачайте його.

Крок 3: Нарешті, Finder зашифрує ваш флеш-накопичувач. Цей процес може тривати деякий час, залежно від обсягу збережених даних.

Шифрування USB-накопичувачів у Linux

Нижче наведено кроки для шифрування USB-накопичувача за допомогою LUKS у Linux.

Крок 1: Перегляньте доступні файлові системи за допомогою наведеної нижче команди.

df -hl

Крок 2: Підключіть USB-накопичувач до ПК.

Крок 3: Визначте щойно підключений пристрій за допомогою наступної команди.

df -hl # у моєму випадку це був /dev/sdb1

Крок 4: Відключіть USB-накопичувач за допомогою наведеної нижче команди.

umount /dev/sdb1

Крок 4: Відключіть USB-накопичувач за допомогою наведеної нижче команди.

umount /dev/sdb1

Крок 5. Очистіть файлову систему з USB-накопичувача.

sudo wipefs -a /dev/sdb1

Крок 6: Створіть розділ LUKS, як показано нижче.

sudo cryptsetup luksFormat /dev/sdb1

УВАГА!

========

Це безповоротно перезапише дані на /dev/sdb1.

Ти впевнений? (Введіть у верхньому регістрі yes): YES

Введіть парольну фразу:

Перевірте парольну фразу:

Крок 7. Відкрийте зашифрований диск за допомогою наведених нижче команд.

sudo cryptsetup luksOpen /dev/sdb1 reddrive

Введіть парольну фразу для /dev/sdb1:

ls -l /dev/mapper/reddrive

lrwxrwxrwx 1 root root 7 липня 26 13:32 /dev/mapper/reddrive -> ../dm-0

Крок 8: Створіть файлову систему, як показано нижче. Тут «EXT4» – файлова система, і ви можете створити будь-яку іншу файлову систему, яка вам потрібна.

sudo mkfs.ext4 /dev/mapper/reddrive -L reddrive

Крок 9. Використання зашифрованого USB

Нижче наведено код для монтування та демонтування зашифрованого USB-накопичувача в Linux CLI.

sudo mount /dev/mapper/reddrive /mnt/red

su -c "echo hello > /mnt/red/hello.txt"

Пароль:

ls -l /mnt/red

total 20

-rw-rw-r–. 1 root root 6 липня 17 10:26 hello.txt

drwx——. 2 root root 16384 17 липня 10:21 втрачено+знайдено

sudo umount /mnt/red

sudo cryptsetup luksЗакрийте reddrive

Якщо ви використовуєте графічний інтерфейс для доступу до зашифрованого USB, з'явиться такий екран.

Введіть там свій пароль, потім збережіть дані на USB-накопичувачі та безпечно вийміть його.

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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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

Опис USB 3.0

USB Implementers Forum фіналізував специфікації стандарту USB 3.0 наприкінці 2008 року. Як і можна було очікувати, новий стандарт збільшив пропускну здатність, хоча приріст не так значно, як 40-кратне збільшення швидкості при переході від USB 1.1 на USB 2.0. У кожному разі, 10-кратне підвищення пропускної здатності можна привітати. USB 3.0 підтримує максимальну швидкість передачі 5 Гбіт/с. Пропускна здатність майже у два рази перевищує сучасний стандарт Serial ATA (3 Гбіт/с із урахуванням передачі інформації надмірності).

Кожний спеціаліст підтвердить, що інтерфейс USB 2.0 є основним «вузьким місцем» сучасних комп'ютерів і ноутбуків, оскільки його пікова «чиста» пропускна здатність становить від 30 до 35 Мбайт/с. Але в сучасних 3,5” жорстких дисків для настільних ПК швидкість передачі вже перевищила 100 Мбайт/з (з'являються й 2,5” моделі для ноутбуків, що наближаються до даного рівня). Швидкісні твердотільні накопичувачі успішно перевершили поріг 200 Мбайт/с. А 5 Гбіт/с (або 5120 Мбіт/с) відповідає 640 Мбайт/с.

Ми не думаємо, що в доступному для огляду майбутньому жорсткі диски наблизяться до рівня 600 Мбайт/с але наступні покоління твердотільних накопичувачів можуть перевищити це число вже через кілька років. Збільшення пропускної здатності стає усе більше важливим, оскільки кількість інформації збільшується, відповідно, росте й час її резервування. Ніж швидше працює сховище, тим менше буде час резервування, тим простіше буде зробити «вікна» у розкладі резервування.

Цифрові відеокамери сьогодні можуть записувати й зберігати гігабайти відеоданих. Частка HD-відеокамер збільшується, а їм потрібні більше ємні й швидкі сховища для запису великої кількості даних. Якщо використовувати USB 2.0, то на передачу декількох десятків гігабайт відеоданих на комп'ютер для монтажу буде потрібно значний час. USB Implementers Forum уважає, що пропускна здатність залишиться принципово важливою, й USB 3.0 буде досить для всіх споживчих пристроїв протягом найближчих п'яти років.

Кодування 8/10 біт

Щоб гарантувати надійну передачу даних інтерфейс USB 3.0 використовує кодування 8/10 біт, знайоме нам, наприклад, по Serial ATA. Один байт (8 біт) передається за допомогою 10-бітного кодування, що поліпшує надійність передачі на шкоду пропускної здатності. Тому перехід з бітів на байти здійснюється зі співвідношенням 10:1 замість 8:1.

Режими енергозбереження

Звичайно, основною метою інтерфейсу USB 3.0 є підвищення доступної пропускної здатності, однак новий стандарт ефективно оптимізує енергоспоживання. Інтерфейс USB 2.0 постійно опитує доступність пристроїв, на що витрачається енергія. Навпроти, в USB 3.0 є чотири стани підключення, названі U0-U3. Стан підключення U0 відповідає активній передачі даних, а U3 занурює пристрій в «сон».

Якщо підключення не діє, то в стані U1 будуть відключені можливості прийому й передачі даних. Стан U2 іде ще на крок далі, відключаючи внутрішні тактові імпульси. Відповідно, підключені пристрої можуть переходити в стан U1 відразу ж після завершення передачі даних, що, як передбачається, дасть відчутні переваги по енергоспоживанню, якщо порівнювати з USB 2.0.

Більший струм

Крім різних станів енергоспоживання стандарт USB 3.0 відрізняється від USB 2.0 й більш високим підтримуваним струмом. Якщо USB 2.0 передбачав поріг струму 500 мА, то у випадку нового стандарту обмеження було зрушено до

планки 900 мА. Струм при ініціації з'єднання був збільшений з рівня 100 мА в USB 2.0 до 150 мА в USB 3.0. Обидва параметри досить важливі для портативних жорстких дисків, які звичайно вимагають ледве більші струми. Раніше проблему вдавалося вирішити за допомогою додаткової вилки USB, одержуючи живлення від двох портів, але використовуючи тільки один для передачі даних, нехай навіть це порушувало специфікації USB 2.0.

Нові кабелі, рознімання, колірне кодування

Стандарт USB 3.0 назад сполучимо з USB 2.0, тобто вилки здаються такими ж, як і звичайні вилки типу А. Контакти USB 2.0 залишилися на колишнім місці, але в глибині роз'єму тепер розташовуються п'ять нових контактів. Це означає, що вам потрібно повністю вставляти вилку USB 3.0 у порт USB 3.0, щоб упевнитися в режимі роботи USB 3.0, для якого потрібні додаткові контакти. Інакше ви одержите швидкість USB 2.0. USB Implementers Forum рекомендує виробникам використовувати колірне кодування Pantone 300C на внутрішній частині роз'єму.

Ситуація вийшла схожою й для USB-вилки типу В, хоча розходження візуально більше помітні. Вилку USB 3.0 можна визначити по п'яти додаткових контактах.

Кабелі USB 3.0

Так як USB 3.0 є інтерфейсом масового користування, кабелі USB 3.0 використовують класичні мідні провідники. Волоконна оптика усе ще занадто дорога для масового застосування. На відміну від USB 2.0, кабелі нового інтерфейсу SuperSpeed USB мають дев'ять, а не чотири проводи. Передача даних здійснюється по чотирьох з п'яти додаткових проводів у диференціальному режимі SDP (Shielded Differential Pair – Екранована Диференційована Пара). Одна пара проводів відповідає за прийом інформації, інша – за передачу. Принцип роботи схожий на Serial ATA, при цьому пристрою одержують повну пропускну здатність в обох напрямках. П'яте проведення – «земля».

Роз’єми й коннектори USB 3.0

Керуючись міркуваннями зворотної сумісності з USB 2.0, при проектуванні вилок і розеток USB 3.0 була застосована концепція «удосконалення» (upgrade) можливості роз’ємів за рахунок додавання до вже існуючої USB 2.0 частини коннекторів додаткової групи контактів, що обслуговують шину SuperSpeed USB. Тобто пристрої й кабелі USB 2.0 підключити до рознімання SuperSpeed USB 3.0 можна, а от навпаки – уже не можна.

Далі в деталях розглядаються USB 3.0 коннектори типу A (USB 3.0 Type A connectors) – установлені на стороні хост-пристрою – комп'ютера або хаба) і рознімання SuperSpeed типу B (USB 3.0 Type B connectors) – установлені на USB 3.0 периферії.

Крім них, існує ще два типи роз’ємів USB 3.0 – це сімейство коннекторів USB 3.0 Micro (призначені для мобільних пристроїв) і новий тип USB 3.0 Powered-B (забезпечує додатковим живленням підключені до нього пристрої).

Вилка й розетка типу A (USB-хрност або Хаб)

Стандарт USB 3.0 назад сполучимо з USB 2.0, тобто вилки здаються такими ж, як і звичайні вилки типу A. Контакти USB 2.0 залишилися на колишнім місці, але в глибині рознімання тепер розташовуються п'ять нових контактів. Це означає, що вам потрібно повністю вставляти вилку USB 3.0 у порт USB 3.0, щоб упевнитися в режимі роботи USB 3.0, для якого потрібні додаткові контакти. Інакше ви одержите швидкість USB 2.0. USB Implementers Forum рекомендує виробникам використовувати колірне кодування Pantone 300C на внутрішній частині роз’єму.

Вилка й розетка типу B

Вилку типу B USB 3.0 можна визначити по п'яти додаткових контактах. Вона хоч і схожа на попередницю USB 2.0, але розходження візуально більше помітні.

Вилка й розетка типу Micro-B

У роз'ємів для мобільних пристроїв зміни більше помітні. Старий роз'єм Micro-B USB 2.0 мав ширину 6,86 мм, однак тепер ширина роз'єму USB 3.0 Micro-B для мобільних телефонів, плеєрів і смартфонів збільшилася до 12,25 мм. Знову ж, роз'єми були зроблені таким чином, щоб забезпечити сумісність с USB 2.0.

Також існують роз'єми USB 3.0 Micro ще двох типів: вилки USB 3.0 Micro-A й розетки USB 3.0 Micro-AB. Візуально відрізняються від USB 3.0 Micro-B «прямокутною» (не зрізаною) частиною роз'єму з USB 2.0 контактами, що дозволяє уникнути підключення вилки Micro-A у розетку Micro-B, а розетку Micro-AB робить сумісної з обома вилками.

Розетка Micro-AB буде застосовуватися в мобільних пристроях, що мають бортовий USB 3.0 host контролер. Для ідентифікації режиму хост/клієнт використовується пін 4 (ID) – в вилці Micro-A він замкнутий на «землю».

Коннектори USB 3.0 Powered-B

Новий роз'єм Powered-B зпроєктований з використанням двох додаткових контактів, що дозволяє пристроям надавати до 1000 мА іншому пристрою, наприклад адаптеру Wireless USB. Це дозволяє уникнути необхідності в джерелі живлення для пристрою, що підключається до Wireless USB адаптера, роблячи ще один крок до ідеальної системи бездротового зв'язку без проводів (навіть для живлення). При звичайних провідних підключеннях до хосту або хабу ці два додаткових контакти не використовуються.

Поточні шифри

Далі наведемо опис поточних шифрів які використовуються для управління даними з ціллю збереження конфіденційності інформації на зовнішніх жорстких дисках інтерфейсу USB 3.0

Поточний шифр – це симетричний шифр, у якому кожний символ відкритого тексту перетвориться в символ шифрованого тексту в залежності не тільки від використовуваного ключа, але й від його розташування в потоці

відкритого тексту. Потоковий шифр реалізує інший підхід до симетричного шифрування, ніж блокові шифри.

Режим гамування для потокових шифрів

Генератор гами видає ключовий потік (гаму):

$$k_1, k_2, k_3, \ldots, k_L.$$

Позначимо потік бітів відкритого тексту:

$$m_1, m_2, m_3, \ldots, m_L.$$

Тоді потік бітів шифротексту виходить за допомогою застосування операції XOR:

$$c_1, c_2, c_3, \ldots, c_L,$$

де:

$$c_i = m_i \oplus k_i.$$

Розшифрування виробляється операцією XOR між тією же самою гамою й зашифрованим текстом:

$$m_i = c_i \oplus k_i.$$

Очевидно, що якщо послідовність бітів гами не має періоду й вибирається випадково, то зламати шифр неможливо. Але в даного режиму шифрування є й негативні особливості. Так ключі, порівнянні по довжині з переданими повідомленнями, важко використовувати на практиці. Тому звичайно застосовують ключ меншої довжини (наприклад, 128 біт). За допомогою його генерується псевдовипадкова гаммуюча послідовність (вона повинна задовольняти постулатам Голомба). Природно, псевдовипадковість гами може бути використана при атаці на потоковий шифр.

Класифікація потокових шифрів

Допустимо, наприклад, що в режимі гамування для потокових шифрів при передачі по каналу зв'язку відбулося перекручування одного знака шифротексту. Очевидно, що в цьому випадку всі знаки, прийняті без перекручування, будуть розшифровані правильно. Відбудеться втрата лише одного знака тексту. А тепер представимо, що один зі знаків шифротексту при передачі по каналу зв'язку був

загублений. Це приведе до неправильного розшифрування всього тексту, що випливає за загубленим знаком.

Практично у всіх каналах передачі даних для потокових систем шифрування присутні перешкоди. Тому для запобігання втрати інформації вирішують проблему синхронізації шифрування й розшифрування тексту. По способі рішення цієї проблеми шифросистеми підрозділяються на синхронні й системи із самосинхронізацією.

Синхронні потокові шифри

Синхронні потокові шифри (СПШ) – шифри, у яких потік ключів генерується незалежно від відкритого тексту й шифротексту.

При шифруванні генератор потоку ключів видає біти потоку ключів, які ідентичні биткам потоку ключів при дешифруванні. Втрата знака шифротексту приведе до порушення синхронізації між цими двома генераторами й неможливості розшифрування частини, що залишилася, повідомлення. Очевидно, що в цій ситуації відправник і одержувач повинні повторно сінхронізуватися для продовження роботи.

Звичайно синхронізація виробляється вставкою в передане повідомлення спеціальних маркерів. У результаті цього пропущений при передачі знак приводить до невірного розшифрування лише доти, поки не буде прийнятий один з маркерів.

Помітимо, що виконуватися синхронізація повинна так, щоб жодна частина потоку ключів не була повторена. Тому переводити генератор у більше ранній стан не має змісту.

Плюси СПШ:

– відсутність ефекту поширення помилок (тільки перекручений біт буде розшифрований невірно);

– охороняють від будь-яких вставок і видалень шифротексту, тому що вони приведуть до втрати синхронізації й будуть виявлені.

Мінуси СПШ:

– уразливі до зміни окремих біт шифрованого тексту. Якщо зловмисникові відомий відкритий текст, він може змінити ці біти так, щоб вони розшифровувалися, як йому треба.

Потокові шифри, що самосинхронізуються

Потокові шифри, що самосинхронізуються (асинхронні потокові шифри (АПШ)) – шифри, у яких потік ключів створюється функцією ключа й фіксованого числа знаків шифротексту.

Отже, внутрішній стан генератора потоку ключів є функцією попередніх N бітів шифротексту. Тому розшифруючий генератор потоку ключів, прийнявши N бітів, автоматично синхронізується з генератором, що шифрує.

Реалізація цього режиму відбувається в такий спосіб: кожне повідомлення починається випадковим заголовком довжиною N бітів; заголовок шифрується, передається й розшифровується; розшифровка є неправильної, зате після цих N біт обидва генератори будуть синхронізовані.

Плюси АПШ:

Розмішування статистики відкритого тексту. Тому що кожний знак відкритого тексту впливає на наступний шифротекст, статистичні властивості відкритого тексту поширюються на весь шифротекст. Отже, АПШ може бути більше стійким до атак на основі надмірності відкритого тексту, чим СПШ.

Мінуси АПШ:

– поширення помилки (кожному неправильному біту шифротексту відповідають N помилок у відкритому тексті);

– чутливі до розкриття повторною передачею.

Потокові шифри на регістрах зрушення з лінійним зворотним зв'язком (РЗЛЗЗ)

Є кілька причин використання лінійних регістрів зрушення в криптографії:

– висока швидкодія криптографічних алгоритмів

– застосування тільки найпростіших операцій додавання й множення, апаратно реалізованих практично у всіх обчислювальних пристроях

– гарні криптографічні властивості (генеруємі послідовності мають великий період і гарні статистичні властивості)

– легкість аналізу з використанням алгебраїчних методів за рахунок лінійної структури

Потокові шифри засновані на РЗЛЗЗ. Ускладнення

На жаль, вихідна послідовність РЗЛЗЗ легко передбачувана. Так, знаючи 2L знаків вихідної послідовності, легко знайти вихідне заповнення регістра, вирішивши систему лінійних рівнянь.

Уважається, що для криптографічного використання вихідна послідовність РЗЛЗЗ повинна мати наступні властивості:

– великий період;

– високу лінійну складність;

– гарні статистичні властивості.

Існує кілька методів проектування генераторів ключового потоку, які руйнують лінійні властивості РЗЛЗЗ і тим самим роблять такі системи криптографічнобільше стійкими:

– використання нелінійної функції, що поєднує виходи декількох РЗЛЗЗ;

– використання нелінійної фільтруючої функції для вмісту кожного осередку єдиного РЗЛЗЗ;

– використання виходу одного РЗЛЗЗ для керування синхросигналом одного (або декількох) РЗЛЗЗ.

Нелінійна комбінація генераторів

Відомо, що кожна булева функція:

$$f(x_1, x_2, \ldots, x_n),$$

може бути записана як сума за модулем 2 добутків порядків m незалежних змінних:

$$0 \leqslant m \leqslant n.$$

Це вираження називається алгебраїчною нормальною формою функції f. Нелінійним порядком функції f називається максимальний порядок членів у записі її алгебраїчної нормальної форми.

Генератор Геффа

У цьому генераторі використовуються три РЗЛЗЗ, об'єднані нелінійним образом. Довжини цих регістрів:

$$L_1, L_2, L_3,$$

попарно прості числа.

Нелінійну функцію для даного генератора можна записати в такий спосіб:

$$f(x_1, x_2, x_3) = x_1x_2 \oplus (1 + x_2)x_3 = x_1x_2 \oplus x_2x_3 \oplus x_3.$$

Довжина періоду:

$$(2^{L_1} - 1) \cdot (2^{L_2} - 1) \cdot (2^{L_3} - 1).$$

Лінійна складність:

$$L = L_1 \cdot L_2 + L_2 \cdot L_3 + L_3.$$

Генератор Геффа криптографічнослабшав, тому що інформація про стани генераторів РЗЛЗЗ 1 і РЗЛЗЗ 3 утримується в його вихідній послідовності.

Генератор на нелінійному фільтрі

Вихід кожного осередку подається на вхід деякої нелінійної булевої фільтруючої функції f. Припустимо, що фільтруюча функція порядку m, тоді лінійна складність потоку ключів не більше:

$$L_m = \sum_{i=1}^{m} \binom{L}{i}.$$

Генератори засновані на керуванні синхросигналом

У нелінійних комбінаціях генераторів і генераторах на нелінійних фільтрах переміщення даних у всіх РЗЛЗЗ контролюється одним синхросигналом.

Основна ідея функціонування розглянутого типу генераторів – внести нелінійність у роботу генераторів потоку ключів, заснованих на РЗЛЗЗ, шляхом керування синхросигналом одного регістра вихідною послідовністю іншого.

Є 2 типи генераторів заснованих на керуванні синхросигналом:

– генератор змінного кроку;

– стискаючий генератор.

Генератор змінного кроку

РЗЛЗЗ 1 використовується для керування пересуванням бітів двох інших РЗЛЗЗ 2 і 3.

Генератор змінного кроку

Алгоритм роботи:

1. Регістр РЗЛЗЗ 1 синхронізований зовнішнім синхросигналом.

2. Якщо на виході регістра РЗЛЗЗ 1 одиниця, то на регістр РЗЛЗЗ 2 подається синхросигнал, а РЗЛЗЗ 3 повторює свій попередній вихідний біт (для початкового моменту часу попередній вихідний біт РЗЛЗЗ 3 приймається рівним 0).

3. Якщо на виході регістра РЗЛЗЗ 1 нуль, то на регістр РЗЛЗЗ 3 подається синхросигнал, а РЗЛЗЗ 2 повторює свій попередній вихідний біт (для початкового моменту часу попередній вихідний біт РЗЛЗЗ 2 також приймається рівним 0).

4. Вихідна послідовність бітів генератора зі змінним кроком є результатом застосування операції побітового АБО, що виключає, до вихідних послідовностей регістрів РЗЛЗЗ 2 і РЗЛЗЗ 3.

Збільшення безпеки генераторів зі змінним кроком:

– довжини регістрів РЗЛЗЗ 1, РЗЛЗЗ 2, РЗЛЗЗ 3 повинні бути обрані попарно простими числами;

– довжини цих регістрів повинні бути близькими числами.

Стискаючий генератор

Контролюючий регістр РЗЛЗЗ 1 використовується для керування виходом РЗЛЗЗ 2.

Алгоритм:

– Регістри РЗЛЗЗ 1 і РЗЛЗЗ 2 синхронізовані загальним синхросигналом.

– Якщо вихідний біт РЗЛЗЗ 1 дорівнює 1, вихід генератора формується вихідним бітом регістра РЗЛЗЗ 2.

– Якщо вихідний біт РЗЛЗЗ 1 дорівнює 0, вихідний біт регістра РЗЛЗЗ 2 відкидається.

Стискаючий генератор простий, масштабуємий й має гарні захисні властивості. Його недолік полягає в тому, що швидкість генерації ключа не буде постійної, якщо не прийняти деяких обережностей.

Для збільшення безпеки стискаючого генератора:

– довжини регістрів РЗЛЗЗ 1 і РЗЛЗЗ 2 повинні бути взаємно простими числами;

– бажано використовувати сховане з'єднання між регістрами РЗЛЗЗ 1 і РЗЛЗЗ 2.

Основні відмінності потокових шифрів від блокових

Більшість існуючих шифрів із секретним ключем однозначно можуть бути віднесені або до потокових, або до блокових шифрів. Але теоретична границя між ними є досить розмитою. Наприклад, використовуються алгоритми блокового шифрування в режимі потокового шифрування (приклад: для алгоритму DES режими CFB і OFB).

Розглянемо основні розходження між потоковими й блоковими шифрами не тільки в аспектах їхньої безпеки й зручності, але й з погляду їхнього вивчення у світі:

– найважливішим достоїнством потокових шифрів перед блоковими є висока швидкість шифрування, порівнянна зі швидкістю надходження вхідної інформації; тому, забезпечується шифрування практично в реальному масштабі часу поза залежністю від обсягу й розрядності потоку преутворених даних;

– у синхронних потокових шифрах (на відміну від блокових) відсутній ефект розмноження помилок, тобто число перекручених елементів у розшифрованій послідовності дорівнює числу перекручених елементів зашифрованої послідовності, що прийшла з каналу зв'язку;

– структура потокового ключа може мати уразливі місця, які дають можливість криптоаналітику одержати додаткову інформацію про ключ (наприклад, при малому періоді ключа криптоаналітик може використовувати

знайдені частини потокового ключа для дешифрування наступного закритого тексту).

ПШ на відміну від БШ часто можуть бути атаковані за допомогою лінійної алгебри (тому що виходи окремих регістрів зрушення зі зворотним лінійним зв'язком можуть мати кореляцію з гамою). Також для взлому потокових шифрів досить успішно застосовується лінійний і диференціальний аналіз.

Сучасний стан потокових шифрів:

– у більшості робіт з аналізу й взлому блокових шифрів розглядаються алгоритми шифрування, засновані на стандарті DES; для потокових же шифрів немає виділеного напрямку вивчення; методи взлому ПШ досить різноманітні;

– для потокових шифрів установлений набір вимог, що є критеріями надійності (великі періоди вихідних послідовностей, постулати Голомба, нелінійність), для БШ таких чітких критеріїв немає;

– дослідженням і розробкою потокових шифрів в основному займаються європейські криптографічні центри, блокових – американські;

– дослідження потокових шифрів відбувається більш динамічно, ніж блокових; останнім часом не було зроблено ніяких помітних відкриттів у сфері DES-Алгоритмів, у той час як в області потокових шифрів трапилася безліч успіхів і невдач (деякі схеми, які вважалися стійкими, при подальшому дослідженні не виправдали надій винахідників).

Проектування потокових шифрів

Згідно Райнера Рюппеля можна виділити чотири основних підходи до проектування потокових шифрів:

– Системно-теоретичний підхід заснований на створенні для криптоаналітика складної, раніше недослідженої проблеми.

– Складністно-теоретичний підхід заснований на складній, але відомій проблемі (наприклад, факторизація чисел або дискретне логарифмування).

– Інформаційно-технічний підхід заснований на спробі приховати відкритий текст від криптоаналітика – поза залежністю від того скільки часу витрачено на дешифрування, криптоаналітик не знайде однозначного рішення.

– Рандомізований підхід заснований на створенні об'ємного завдання; криптограф тим самим намагається зробити рішення завдання розшифрування фізично неможливої. Наприклад, криптограф може зашифрувати деяку статтю, а ключем будуть вказівки на те, які частини статті були використані при шифруванні. Криптоаналітику прийде перебирати всі випадкові комбінації частин статті, перш ніж йому повезе, і він визначить ключ.

Теоретичні критерії Райнера Рюппеля для проектування потокових систем:

– довгі періоди вихідних послідовностей;

– більша лінійна складність;

– дифузія – розсіювання надмірності в підструктурах, «розмазування» статистики по всьому тексту;

– кожний біт потоку ключів повинен бути складним перетворенням більшості бітів ключів;

– критерій нелінійності для логічних функцій.

Дотепер не доведено, що ці критерії необхідні або достатні для безпеки потокової системи шифрування. Варто також помітити, що, якщо криптоаналітик має необмежений час і обчислювальну потужність, те єдиним реалізованим потоковим шифром, захищеним від такого супротивника є одноразовий блокнот.

Криптоаналіз. Атаки на потокові шифри

Всі методи криптоаналізу потокових шифрів звичайно підрозділяють на силові (атака «грубою силою»), статистичні й аналітичні.

Силові атаки

До цього класу відносяться атаки, що здійснюють повний перебір всіх можливих варіантів. Складність повного перебору залежить від кількості всіх можливих рішень завдання (зокрема, розміру простору ключів або простору відкритих текстів). Цей клас атак застосуємо до всіх видів систем потокового шифрування. При розробці систем шифрування розроблювачі прагнуть зробити так, щоб цей вид атак був найбільш ефективним у порівнянні з іншими існуючими методами взлому.

Статистичні атаки

Статистичні атаки діляться на два підкласи:

– метод криптоаналізу статистичних властивостей гами, що шифрує: спрямований на вивчення вихідної послідовності криптосистеми; криптоаналітик намагається встановити значення наступного біта послідовності з імовірністю вище ймовірності випадкового вибору за допомогою різних статистичних тестів;

– метод криптоаналізу складності послідовності: криптоаналітик намагається знайти спосіб генерувати послідовність, аналогічну гамі, але більш просто реалізованим способом.

Обидва методи використовують принцип лінійної складності.

Аналітичні атаки

Цей вид атак розглядається в припущенні, що криптоаналітику відомо опис генератора, відкритий і відповідний закритий тексти. Завдання криптоаналітика визначити використаний ключ (початкове заповнення регістрів). Види аналітичних атак, застосовувані до синхронних потокових шифрів:

– кореляційні;

– компроміс “час-пам'ять”;

– інверсійна;

– “припускай і визначай”;

– на ключове завантаження й реініціалізацію;

– XSL-атака.

Кореляційні атаки

Є найпоширенішими атаками для взлому потокових шифрів.

Відомо, що робота з розкриття криптосистеми може бути значно скорочена, якщо нелінійна функція пропускає на вихід інформацію про внутрішні компоненти генератора. Тому для відновлення початкового заповнення регістрів кореляційні атаки досліджують кореляцію вихідної послідовності шифросистеми з вихідною послідовністю регістрів.

Існують наступні підкласи кореляційних атак:

– Базові кореляційні атаки.

– Атаки, засновані на низько-вагових перевірках парності.

– Атаки, засновані на використанні згорточних кодів.

– Атаки, що використовують техніку турбо кодів.

– Атаки, засновані на відновленні лінійних поліномів.

– Швидкі кореляційні атаки.

Компроміс « час-пам'ять»

Ціль даної атаки – відновлення вихідного стану регістра зрушення (знаходження ключа), використовуючи відому схема пристрою й фрагмент послідовності, що шифрує. Складність атаки залежить від розміру шифру й довжини перехопленої гами.

Складається із двох етапів:

– побудова великого словника, у якому записані всілякі пари «стан-вихід»;

– припущення про початкове заповнення регістра зрушення, генерація виходу, перегляд перехопленої вихідної послідовності й пошук відповідності зі згенерированим виходом. Якщо відбувся збіг, то дане можливе заповнення з великою ймовірністю є початковим.

Прикладами цього класу атак є атака Стіва Беббіджа й атака Бірюкова-Шаміра.

«Припускай і визначай»

Атака ґрунтується на припущенні, що криптоаналітику відомо гаму, поліном зворотного зв'язка, кількість зрушень регістра між виходами схеми й фільтруюча функція. Складається із трьох етапів:

– припущення про заповнення деяких осередків регістра;

– визначення повного заповнення регістра на підставі припущення про знання криптоаналітика;

– генерація вихідної послідовності; якщо вона збігається з гамою, то припущення на першому етапі було вірно; якщо не збігається, то вертаємося до етапу 1.

Складність алгоритму залежить від пристрою генератора й від кількості припущень.

3.2 Розробка структурної схеми

Структурна схема зображена на рисунку 3.1. У процесі управління даними з ціллю збереження конфіденційності інформації використовується певний алгоритм управління даними з ціллю збереження конфіденційності інформації, на вхід якому подаються вихідні незашифровані дані, називані також plaintext, і ключ. Виходом алгоритму є зашифровані дані, називані також ciphertext. Ключ є значенням, що не залежить від шифруємих даних. Зміна ключа повинна приводити до зміни зашифрованого повідомлення.

Зашифровані дані зберігаються на зовнішньому жорсткому диску інтерфейсу USB 3.0 й передаються одержувачеві, при читанні даних з зовнішнього жорсткого диску інтерфейсу USB 3.0. Одержувач перетворить зашифровані дані у вихідні незашифровані дані за допомогою алгоритму дешифрування даних з ціллю збереження конфіденційності інформації й того ж самого ключа, що використовувався при шифруванні, або ключа, легко одержуваного із ключа управління даними з ціллю збереження конфіденційності інформації.

Незашифровані дані будемо позначати P або M, від слів plaintext і message. Зашифровані дані будемо позначати C, від слова ciphertext.

Безпека, забезпечувана криптографією, яка використовує поточні шифри, залежить від декількох факторів:

– Криптографічний алгоритм повинен бути досить сильним, щоб передані зашифровані дані неможливо було розшифрувати без ключа, використовуючи тільки різні статистичні закономірності зашифрованих даних або які-небудь інші способи їх аналізу.

– Безпека переданих даних повинна залежати від таємності ключа, але не від таємності алгоритму. Алгоритм повинен бути проаналізований фахівцями, щоб виключити наявність слабких місць, при яких погано схований взаємозв'язок між незашифрованими і зашифрованими даними. До того ж при виконанні цієї

умови виробники можуть створювати дешеві апаратні чипи й вільно розповсюджувані програми, що реалізують даний алгоритм управління даними з ціллю збереження конфіденційності інформації.

– Алгоритм повинен бути таким, щоб не можна було довідатися ключ, навіть знаючи досить багато пар (зашифровані дані, незашифровані дані), отриманих при шифруванні з використанням даного ключа.

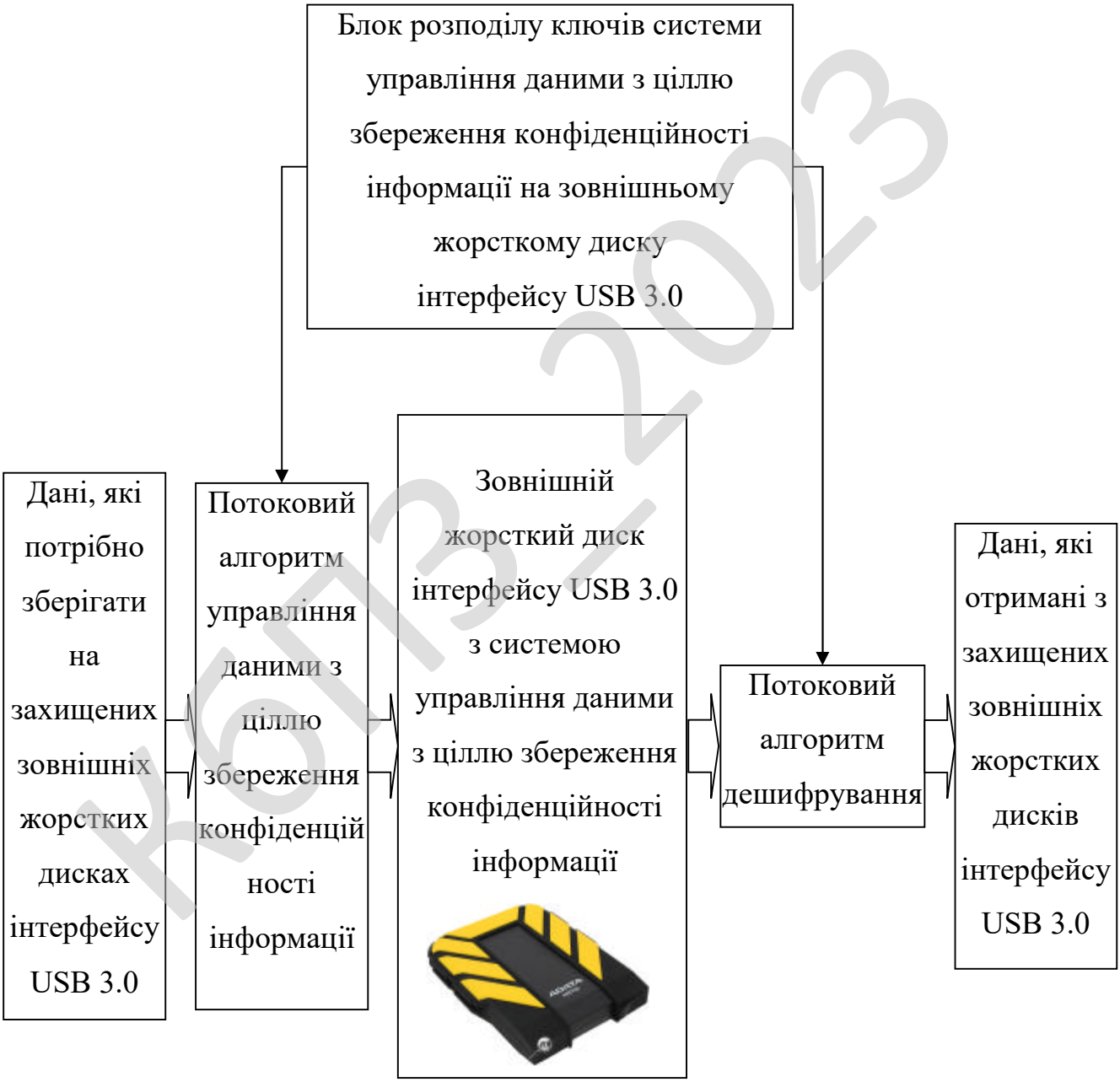


Рисунок 3.1 – Структурна схема системи

Клод Шеннон увів поняття дифузії й конфузії для опису стійкості алгоритму управління даними з ціллю збереження конфіденційності інформації.

Дифузія – це розсіювання статистичних особливостей незашифрованого тексту в широкому діапазоні статистичних особливостей зашифрованого тексту. Це досягається тим, що значення кожного елемента незашифрованого тексту впливає на значення багатьох елементів зашифрованого тексту або, що той же саме, будь-який елемент зашифрованого тексту залежить від багатьох елементів незашифрованого тексту.

Конфузія – це знищення статистичного взаємозв'язку між зашифрованим текстом і ключем.

Якщо X – це вихідне повідомлення й K – криптографічний ключ, то зашифрований переданий текст можна записати у вигляді:

$$Y = E_K[X].$$

Одержувач, використовуючи той же ключ, розшифровує повідомлення:

$$X = D_K[Y]$$

Супротивник, не маючи доступу до K і X, повинен спробувати довідатися X, K або й те, і інше.

3.3 Розробка функціональної схеми

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

До потокових алгоритмів управління даними на зовнішніх жорстких дисках інтерфейсу USB 3.0, які використовуються у розробленій системі відносяться наступні:

– A5.

– A8.

– Crypto-1.

– Decim.

– MICKEY.

– Mosquito.

– Phelix.

– Rabbit.

– RC4.

– SEAL.

– SOSEMANUK.

– Trivium.

– VMPC.

Розглянемо деякі із потокових алгоритмів, які використовуються для управління даними з ціллю збереження конфіденційності інформації даних на зовнішньому жорсткому диску інтерфейсу USB 3.0.

A5

A5 – це потоковий алгоритм управління даними з ціллю збереження конфіденційності інформації, використовуваний для забезпечення конфіденційності переданих даних між телефоном і базовою станцією в європейській системі мобільного цифрового зв'язку GSM (Group Special Mobile).

Шифр заснований на побітовому додаванні за модулем два (булева операції XOR) генеруємої псевдовипадкової послідовності й шифруємої інформації. В A5 псевдовипадкова послідовність реалізується на основі трьох лінійних регістрів зрушення зі зворотним зв'язком. Регістри мають довжини 19, 22 і 23 біта відповідно. Зрушеннями управляє спеціальна схема, що організує на кожному кроці зсув як мінімум двох регістрів, що приводить до їхнього нерівномірного руху. Послідовність формується шляхом операції XOR над вихідними бітами регістрів.

Рисунок 3.2 – Функціональна схема системи

Споконвічно французькими військовими фахівцями-криптографами був розроблений потоковий шифр для використання винятково у військових цілях. Наприкінці 80х для стандарту GSM треба було створення нової, сучасної системи безпеки. У її основу лягли три секретних алгоритми: автентифікації – A3, управління даними з ціллю збереження конфіденційності інформації потоку – A5, генерації сесійного ключа – A8. Як алгоритм A5, була використана французька розробка. Цей шифр забезпечував досить гарну захищеність потоку, що забезпечувало конфіденційність розмови. Споконвічно експорт стандарту з Європи не передбачався, але незабаром у цьому з'явилася необхідність. Саме тому, A5 перейменували в A5/1 і стали поширювати в Європі й США. Для інших країн (у тому числі й Росії) алгоритм модифікували, значно понизивши криптостійкість шифру. A5/2 був спеціально розроблений як експортний варіант для країн, що не входили в Євросоюз. Криптостійкість A5/2 була знижена додаванням ще одного регістра (17 біт), керуючого зрушеннями інших. В A5/0 управління даними з ціллю збереження конфіденційності інформації відсутній зовсім. У цей час розроблені також алгоритм A5/3, заснований на алгоритмі Касумі й затверджений для використання в мережах 3G. Ці модифікації позначають A5/x.

Офіційно дана криптосхема не публікувалася і її структура не віддавалася гласності. Це пов'язано з тим, що розроблювачі покладалися на безпеку за рахунок невідомості, тобто алгоритми сутужніше зламати, якщо вони не доступні привселюдно. Дані надавалися операторам GSM тільки по необхідності. Проте, до 1994 року деталі алгоритму A5 стали відомі: британська телефонна компанія (British Telecom) передала всю документацію, що стосується стандарту, Бредфордскому університету для аналізу, не уклавши угоду про нерозголошення інформації. Крім того, матеріали про стандарт з'явилися на одній конференції в Китаї. У результаті, його схема поступово просочилася в широкі кола. У цьому ж році кембриджські вчені Ross Anderson і Michael Roe опублікували відновлену за цим даними криптосхему й дали оцінку її криптостійкості [1]. Остаточно алгоритм був представлений у роботі Йована Голича на конференції Eurocrypt'97.

Структура A5

Алгоритм A5 у цей час – це ціле сімейство шифрів. Для опису візьмемо A5/1, як родоначальника цього сімейства. Зміни в похідних алгоритмах опишемо окремо.

Потокове управління даними з ціллю збереження конфіденційності інформації

Схема потокового шифру: додавання відкритого тексту й послідовності біт дає шифротекст.

У цьому алгоритмі кожному символу відкритого тексту відповідає символ шифротексту. Текст не ділиться на блоки (як у блоковому шифруванні) і не змінюється в розмірі. Для спрощення апаратної реалізації й, отже, збільшення швидкодії використовуються тільки найпростіші операції: додавання за модулем 2 (XOR) і зрушення регістра.

Формування вихідної послідовності відбувається шляхом додавання потоку вихідного тексту з генеруємої послідовністю (гамою). Особливість операції XOR полягає в тому, що застосована парне число раз, вона приводить до початкового значення. Звідси, декодування повідомлення відбувається шляхом додавання шифротексту з відомою послідовністю.

Таким чином, безпека системи повністю залежить від властивостей послідовності. В ідеальному випадку кожний біт гами – це незалежна випадкова величина, і сама послідовність є випадковою. Така схема була винайдена Вернамом в 1917 році й названа в його честь. Як довів Клод Шеннон в 1949 році, це забезпечує абсолютну криптостійкість. Але використання випадкової послідовності означає передачу по захищеному каналі повідомлення рівного по обсязі відкритому тексту, що значно ускладнює завдання й практично ніде не використовується. У реальних системах створюється ключ заданого розміру, що без праці передається по закритому каналі. Послідовність генерується на його основі і є псевдовипадковою. Великий клас потокових шифрів (у тому числі A5)

становлять шифри, генератор псевдовипадкової послідовності якої заснований на регістрах зрушення з лінійним зворотним зв'язком.

РЗЛЗЗ

Регістр зрушення з лінійним зворотним зв'язком, багаточлен зворотного зв'язка $x^{32}+x^{29}+x^{25}+x^5+1$.

Регістр зрушення з лінійним зворотним зв'язком складається із власно регістра (послідовності біт заданої довжини) і зворотного зв'язка. На кожному такті відбуваються наступні дії: крайній лівий біт (старший біт) витягає, послідовність зрушується вліво й у спустілий правий осередок (молодший біт) записується значення функції зворотного зв'язка. Ця функція є підсумовуванням за модулем два певних біта регістри й записується у вигляді багаточлена, де ступінь указує номер біта. Витягнуті біти формують вихідну послідовність.

Для РЗЛЗЗ основним показником є період псевдовипадкової послідовності. Він буде максимальний (і дорівнює 2n-1), якщо багаточлен функції зворотного зв'язка примітивний за модулем 2. Вихідна послідовність у такому випадку називається М-послідовністю.

Система РЗЛЗЗ в А5

Сам по собі РЗЛЗЗ легко піддається криптоаналізу й не є досить надійним, для використання в шифруванні. Практичне застосування мають системи регістрів змінного тактування, з різними довжинами й функціями зворотного зв'язку.

Структура алгоритму А5 виглядає в такий спосіб:

– три регістри(R1, R2, R3) мають довжини 19, 22 і 23 біта;

– багаточлени зворотних зв'язків:

$$X^{19} + X^{18} + X^{17} + X^{14} + 1 \text{ для R1,}$$

$$X^{22} + X^{21} + 1 \text{ для R2 і}$$

$$X^{23} + X^{22} + X^{21} + X^{8} + 1 \text{ для R3.}$$

Керування тактуванням здійснюється спеціальним механізмом:

– у кожному регістрі є біти синхронізації: 8 (R1), 10 (R2), 10 (R3);

– обчислюється функція F = x&y|x&z|y&z, де & – булево AND, | – булево OR, а x, y і z – біти синхронізації R1, R2 і R3 відповідно;

– зрушуються тільки ті регістри, у яких біт синхронізації дорівнює F;

– фактично, зрушуються регістри, синхробіт яких належить більшості;

– вихідний біт системи – результат операції XOR над вихідними бітами регістрів.

Функціонування алгоритму A5

Розглянемо особливості функціонування алгоритму, на основі відомої схеми. Передача даних здійснюється в структурованому виді – з розбивкою на кадри (114 біт). Перед ініціалізацією регістри обнуляються, на вхід алгоритму надходять сесійний ключ (K – 64 біта), сформований A8, і номер кадру (Fn – 22 біта). Далі послідовно виконуються наступні дії:

Ініціалізація:

– 64 такту, при яких черговий біт ключа XOR з молодшим бітом кожного регістра, регістри при цьому зрушуються на кожному такті;

– аналогічні 22 такту, тільки операція XOR виробляється з номером кадру;

– 100 тактів з керуванням зрушеннями регістрів, але без генерації послідовності;

– 228 (114 + 114) тактів робітники, відбувається управління даними з ціллю збереження конфіденційності інформації переданого кадру (перші 114 біт) і дешифрування даних з ціллю збереження конфіденційності інформації (останні 114 біт) прийнятого;

– далі ініціалізація виробляється заново, використовується новий номер кадру.

Відмінності похідних алгоритмів A5/x

Система регістрів в алгоритмі A5/2

В алгоритм A5/2 доданий ще один регістр на 17 біт (R4), керуючий рухом інших. Зміни структури наступні:

– додано регістр R4 довжиною 17 біт,

– багаточлен зворотного зв'язка для R4:

$$X^{17} + X^{12} + 1.$$

Керування тактуванням здійснює R4:

– в R4 біти 3, 7, 10 є біти синхронізації;

– обчислюється мажоритарна функція F = x&y|x&z|y&z (дорівнює більшості), де & – булево AND, | – булево OR, а x, y і z – бітів синхронізації R4(3), R4(7) і R4(10) відповідно;

– R1 зрушується якщо R4(10) = F;

– R2 зрушується якщо R4(3) = F;

– R3 зрушується якщо R4(7) = F;

– вихідний біт системи – результат операції XOR над старшими бітами регістрів і мажоритарних функцій від певних бітів регістрів:

– R1 – 12, 14, 15;

– R2 – 9, 13, 16;

– R3 – 13, 16, 18.

Зміни у функціонуванні не такі істотні й стосуються тільки ініціалізації:

– 64+22 такту заповнюється сесійним ключем і номером кадру також R4;

– 1 такт R4(3), R4(7) і R4(10) заповнюються 1;

– 99 тактів з керуванням зрушеннями регістрів, але без генерації послідовності.

Видно, що ініціалізація займає такий же час (100 тактів без генерації розбиті на дві частини).

Алгоритм A5/3 розроблений в 2001 році й повинен перемінити A5/1 у третім поколінні мобільних систем. Також він називається алгоритм Касумі. При його створенні за основи взятий шифр MISTY, корпорації Mitsubishi. У цей час уважається, що A5/3 забезпечує необхідну стійкість.

Алгоритм A5/0 не містить управління даними з ціллю збереження конфіденційності інформації.

Криптостійкість

Розробка стандарту GSM мала на увазі потужний апарат управління даними з ціллю збереження конфіденційності інформації, що не піддається взлому (особливо в реальному часі). Використовувані розробки при належній реалізації забезпечували якісне управління даними з ціллю збереження конфіденційності інформації переданих даних. Саме таку інформацію можна одержати від компаній які розповсюджують цей стандарт. Але варто відзначити важливий нюанс: прослуховування розмов – невід'ємний атрибут, використовуваний спецслужбами. Вони були зацікавлені в можливості прослуховування телефонних розмов для своїх цілей. Таким чином, в алгоритм були внесені зміни, що дають можливість взлому за прийнятний час. Крім цього, для експорту A5 модифікували в A5/2. В MoU (Memorandum of Understand Group Special Mobile standard) визнають, що метою розробки A5/2 було зниження криптостійкості управління даними з ціллю збереження конфіденційності інформації, однак в офіційних результатах тестування говориться, що невідомо про яких-небудь недоліки алгоритму[2].

Відомі уразливості

З появою даних про стандарт A5, почалися спроби взлому алгоритму, а також пошуку уразливостей. Величезну роль зробили особливості стандарту, що різко послабляють захист, а саме:

– 10 біт ключа примусово онулені;

– відсутність перехресних зв'язків між регістрами (крім керування зрушеннями);

– зайва надмірність шифруємої службової інформації, відомої криптоаналітику;

– понад 40 % ключів приводить до мінімальної довжини періоду генеруємої послідовності;

– спочатку сеансу здійснюється обмін нульовими повідомленнями (по одному кадрі);

– в A5/2 рух здійснюється окремим регістром довжиною 17 біт.

На основі цих «дір» в алгоритмі, побудовані схеми взлому.

Відомі атаки

Ключем є сесійний ключ довжиною 64 біта, номер кадру вважається відомим. Таким чином, складність атаки заснованої на прямому переборі дорівнює 264.

Перші огляди шифру (робота Росса Андерсона) відразу виявили уразливість алгоритму – через зменшення ефективної довжини ключа (занулення 10 біт) складність упала до 245 (відразу на 6 порядків). Атака Андерсона заснована на припущенні про початкове заповнення коротких регістрів і за вихідним даними одержання заповнення третього.

В 1997 році Йован Голич опублікував результати аналізу A5. Він запропонував спосіб визначення первісного заповнення регістрів по відомому відрізку гами довжиною всього 64 біта. Цей відрізок одержують із нульових повідомлень. Атака має середню складність 240.

В 1999 році Вагнерові й Голдбергу вдалося продемонструвати, що для розкриття системи, досить перебором визначити початкове заповнення R4. Перевірка здійснюється за рахунок нульових кадрів. Складність цієї атаки дорівнює 217, таким чином, на сучасному комп'ютері розкриття шифру займає кілька секунд.

У грудні 1999 року група ізраїльських учених (Ади Шамір, Алекс Бірюков, а пізніше й американець Дэвид Вагнер (англ.)) опублікували досить нетривіальний, але теоретично дуже ефективний метод розкриття A5/1.

A8

A8 – алгоритм формування ключа управління даними з ціллю збереження конфіденційності інформації, що згодом використовується для забезпечення конфіденційності переданої по радіоканалу інформації в стандарті мобільного стільникового зв'язка GSM. A8 є одним з алгоритмів забезпечення таємності розмови в GSM разом з A5 і A3. Його завдання – генерація сеансового ключа Kc

для потокового управління даними з ціллю збереження конфіденційності інформації інформації в каналі зв'язку між стільниковим телефоном (MS – Mobile Station) і базовою станцією (BTS – Basic Transmitter Station) після автентифікації. Через безпеку формування Kc відбувається в Sim-карті.

Забезпечення безпеки

Під «безпекою» в GSM будемо розуміти неможливість несанкціонованого використання системи й таємність переговорів абонентів. У цьому розділі розглядаються деякі механізми безпеки:

– автентифікація;

– таємність передачі даних.

Механізм автентифікації

Для виключення несанкціонованого використання ресурсів системи зв'язки вводяться механізми автентифікації. У кожного рухливого абонента є стандартний модуль дійсності абонента (SIM-карта), що містить:

– міжнародний ідентифікаційний номер рухливого абонента (IMSI – International Mobile Subscriber Identity);

– свій індивідуальний 128-бітний ключ автентифікації (Ki);

– алгоритм автентифікації (A3), і генерації сеансового ключа (A8).

Ключ автентифікації користувача Ki унікальний і однозначно пов'язаний з IMSI, оператор зв'язку за значенням IMSI «уміє» визначати Ki і обчислює очікуваний результат. Від несанкціонованого використання SIM захищена уведенням індивідуального ідентифікаційного номера (PIN-код – Personal Identification Number), що привласнюється користувачеві разом із самою картою.

Розглянемо процедуру перевірки дійсності абонента. Мережа генерує оказію – випадковий номер (RAND) і передає його на мобільний пристрій. В Sim-карті відбувається обчислення значення відгуку (SRES – Signed Response) і сеансового ключа, використовуючи RAND, Ki і алгоритми A3,A8. Мобільний пристрій обчислює SRES і посилає його в мережу, що звіряє його з тим, що обчислила сама. Якщо обоє значення збігаються, то автентифікація пройдена

успішно й мобільний пристрій одержує від мережі команду ввійти в шифрований режим роботи. Через таємність всі обчислення відбуваються усередині SIM. Секретна інформація (така як Ki) не надходить поза SIM-картою. Ключ Kc також не передається по радіоканалу. Рухлива станція й мережа обчислюють їх окремо друг від друга.

Реалізація алгоритму

Формат вхідних і вихідних даних для алгоритму A8 строго визначений консорціумом 3GPP. Але A8 не є стандартизованим, а визначається оператором. Алгоритми A3 і A8 реалізовані як єдине обчислення, вихідні дані якого (96 біт) трактуються так: 32 біта для утворення SRES і 64 біта для утворення Kc.[1] Довжина значимої частини ключа Kc, видана алгоритмом A8 може бути менше 64 біт. Тоді значимі біти доповнюються нулями до кількості 64, зазначеного в специфікації алгоритму. У цей час відомі наступні стандартні реалізації алгоритму A3/A8:

– COMP128.

– COMP 128-2.

– COMP 128-3.

– MILENAGE.

Хоча існують альтернативи COMP128, але цей протокол як і раніше підтримується в переважній більшості мереж GSM[1]. По даним SDA (Smarcard Developer Assosiation), більшість операторів зв'язку не робить перевірку на одночасне включення «однакових» абонентів, настільки вони впевнені в неможливості клонування SIM-карт.

COMP128

COMP128 – ключова хеш-функція, що генерує за один прохід SRES і Kc. A3, A5, A8 минулого розроблені у Великобританії, і виготовлювачі мобільних телефонів, що бажають реалізувати цю технологію управління даними з ціллю збереження конфіденційності інформації у своїх продуктах, повинні погоджуватися на нерозголошення таємниці й одержувати спеціальні ліцензії від

британського уряду. Розбіжності між виготовлювачами мобільних телефонів і британським урядом навколо експорту технології управління даними з ціллю збереження конфіденційності інформації в GSM були влагоджені в 1993. Але в 1998 році деякі документи з описом були опубліковані в Інтернеті. Незважаючи на неповний опис, було встановлено, які саме криптографічні методи використовуються в GSM. Дэвид Вагнер і Айан Голдберг буквально за день зламали алгоритм COMP128, тому що його ключ був занадто коротким. Криптографічний алгоритм COMP128 усе ще використовується, але в поліпшеній формі за назвою COMP 128-2. Криптографічні алгоритми A3 і A8 специфіковані для мережних операторів, хоча деякі параметри стандартизовані для забезпечення взаємодії між мережами.

На вхід алгоритму подається 128-бітний (16 байт) RAND, отриманий від базової станції й 128-бітний Ki, що пройшов в SIM-ці. Виходом є 96-бітна (12 байт) послідовність. Специфікація стандарту [2] затверджує, що перші 4 байти є SRES, що мобільний апарат відправляє для автентифікації, а байти з 5-го по 12-й – це сесійний ключ Kc. Варто помітити, що біти ключа: з 42 по 95, за яких ідуть 10 нулів. Тобто в 64-бітного ключа Kc ентропія не перевершує 54 біта. Це представляє істотне ослаблення криптостійкості шифру A5 більш ніж в 1000 разів.

Ослаблення криптостійкості

Однією із причин, що пояснюють, чому розроблювачі GSM тримали в секреті алгоритми, можливо, є їхнє співробітництво зі службами контролю.

3GPP

В архітектурі 3GPP (як і в GSM) всім операторам не обов'язково використовувати однакові алгоритми автентифікації й генерації ключа. Звичайно існують рекомендації і як приклад приводиться один алгоритм. Однак, як показує практика, саме він стає повсюдно використовуваним. В 3GPP таким прикладом став MILENAGE. MILENAGE побудований на основі шифру Rijndael (переможець конкурсу AES на кращий американський криптостандарт, що

замінив DES). Що стосується A5 – управління даними з ціллю збереження конфіденційності інформації переговорів і забезпечення цілісності повідомлень, то він неодмінно повинен бути однаковий у всіх операторів, щоб вони могли надавати послугу роумінг. Цей алгоритм в 3GPP побудований на основі блокового шифру KASUMI.

Проблеми безпеки

Активні атаки – зловмисник виконує роль мережного елемента (наприклад, роль BST).

Небезпечна передача ключової інформації: RAND, SRES передаються в явному виді усередині й між мережами.

Однобічна ідентифікація:

– Забезпечується тільки автентифікація користувача для мережі.

– Немає засобів ідентифікації мережі для користувача.

Слабкі алгоритми управління даними з ціллю збереження конфіденційності інформації. Довжина ключа занадто мала, у той час як швидкості обчислень ростуть

Негнучкість. Неадекватна гнучкість, що заважає модернізувати й поліпшувати функціональні можливості захисту згодом.

Crypto-1

Crypto-1 – пропрієтарний алгоритм управління даними з ціллю збереження конфіденційності інформації, створений NXP Semiconductors для використання в RFID-картах стандарту Mifare (Classic). Даний стандарт використовується різними картами, зокрема: соціальною картою москвича, Oyster card, CharlieCard, OV-chipkaart.

Кілька досліджень, проведені в 2008-2009 роках, показали що безпека даного алгоритму є невисокою. Crypto-1 є потоковим шифром, дуже схожим на попередній Hitag2. Crypto-1 складається з:

– одного 48-бітного зсувного регістра зі зворотним зв'язком для зберігання секретного стану;

– лінійної функції;;

– дворівневої нелінійної функції B-1

– 16-бітного LFSR, що використовується при автентифікації. Деякими картами може використовуватися як ГПВП.

Алгоритм може використовуватися як NLFSR або LFSR залежно від вхідних параметрів. Звичайно шифри Crypto-1 і Hitag2 використовують режим NLFSR тільки при ініціалізації й автентифікації, перемикаючись на режим LFSR з нелінійним вихідним фільтром для управління даними з ціллю збереження конфіденційності інформації переданих даних.

RC4

RC4 (англ. Rivest Cipher 4 або англ. Ron’s Code, також відомий як ARCFOUR або ARC4 (англ. Alleged RC4)) – потоковий шифр, що широко застосовується в різних системах захисту інформації в комп'ютерних мережах.

Шифр розроблений компанією RSA Security і для його використання потрібна ліцензія.

Алгоритм RC4, як і будь-який потоковий шифр, будується на основі параметризованного ключем генератора псевдовипадкових бітів з рівномірним розподілом. Довжина ключа може становити від 40 до 256 біт.

Основні переваги шифру – висока швидкість роботи й змінний розмір ключа. RC4 досить уразливий, якщо використовуються не випадкові або зв'язані ключі, один ключовий потік використовується двічі. Ці фактори, а також спосіб використання можуть зробити криптосистему небезпечної (наприклад WEP).

Потоковий шифр RC4 був створений Роном Рівестом з RSA Security в 1987 році. Хоча офіційно скорочення позначає Rivest Cipher 4, його часто вважають скороченням від Ron's Code.

Шифр був комерційною таємницею, але у вересні 1994 року його опис був анонімно відправлений у розсилання Cypherpunks. Незабаром опис RC4 було опубліковано в sci.crypt. Саме звідти вихідний код потрапив на безліч сайтів у мережі Інтернет. Опублікований шифр давав ті ж шифротексти на виході, які

давав справжній RC4. Очевидно, даний текст був отриманий у результаті аналізу коду, що виконується. Опублікований шифр сполучимо з наявними продуктами, що використовують RC4, а деякі учасники телеконференції, що мали, за їхніми словами, доступ до вихідного коду RC4, підтвердили ідентичність алгоритмів при розходженнях у позначеннях і структурі програми.

Оскільки даний алгоритм відомий, він більше не є комерційною таємницею. Однак, назва «RC4» є торговельною маркою компанії RSA. Тому іноді шифр називають «ARCFOUR» або «ARC4» (маючи на увазі Alleged RC4 – передбачуваний RC4, оскільки RSA офіційно не опублікувала алгоритм), щоб уникнути можливих претензій з боку власника торговельної марки.

Шифр RC4 застосовується в деяких широко розповсюджених стандартах і протоколах управління даними з ціллю збереження конфіденційності інформації таких, як WEP, WPA і TLS.

Головними факторами, що сприяли широкому застосуванню RC4, були простота його апаратної й програмної реалізації, а також висока швидкість роботи алгоритму в обох випадках.

У США довжина ключа для використання усередині країни рекомендується рівною 128 биткам, але угода, укладена між Software Publishers Association (SPA) і урядом США дає RC4 спеціальний статус, що означає, що дозволено експортувати шифри довжиною ключа до 40 біт. 56-бітні ключі дозволено використовувати закордонним відділенням американських компаній.

Опис алгоритму

Генератор ключового потоку RC4

Ядро алгоритму складається з функції генерації ключового потоку. Ця функція генерує послідовність бітів (k_i), що потім поєднується з відкритим текстом (m_i) за допомогою підсумовування за модулем два. Так виходить шифрограма (c_i):

$$c_i = m_i \oplus k_i.$$

Розшифровка полягає в регенерації цього ключового потоку (k_i) і додаванні його й шифрограми (c_i) за модулем два. У силу властивостей підсумовування за модулем два на виході ми одержимо вихідний незашифрований текст (m_i):

$$m_i = c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i.$$

Інша головна частина алгоритму – функція ініціалізації, що використовує ключ змінної довжини для створення початкового стану генератора ключового потоку.

RC4 – фактично клас алгоритмів, обумовлених розміром його блоку. Цей параметр *n* є розміром слова для алгоритму. Звичайно, *n* = 8, але з метою аналізу можна зменшити його. Однак для підвищення безпеки необхідно збільшити цю величину.

Внутрішній стан RC4 представляється у вигляді масиву слів розміром 2*n* і двох лічильників, кожний розміром в одне слово. Масив відомий як *S*-бокс, і далі буде позначатися як *S*. Він завжди містить перестановку 2*n* можливих значень слова. Два лічильники позначені через *i* і *j*.

Алгоритм ініціалізації RC4 наведений нижче. Цей алгоритм також називається алгоритмом ключового розкладу (англ. Key-Scheduling Algorithm or KSA). Цей алгоритм використовує ключ, збережений в Key, і байт, що має довжину *L*. Ініціалізація починається із заповнення масиву *S*, далі цей масив перемішується шляхом перестановок, обумовлених ключем. Тому що тільки одна дія виконується над S, те повинне виконуватися твердження, що *S* завжди містить всі значення кодового слова.

Початкове заповнення масиву:

```
for i = 0 to 2n – 1
  S[ i ] = i
```

Скремблювання:

```
j = 0
for i = 0 to 2n – 1
```

j = (j + S[i] + Key[i mod L]) mod 2n

Перестановка(S[i], S[j])

Генератор ключового потоку RC4 переставляє значення, що зберігаються в S, і щораз вибирає різне значення з *S* як результат. В одному циклі RC4 визначається одне *n*-бітне слово *K* із ключового потоку, що надалі підсумується з вихідним текстом для одержання зашифрованого тексту. Ця частина алгоритму називається генератором псевдовипадкової послідовності (англ. Pseudo-Random Generation Algorithmor PRGA).

Ініціалізація:

i = 0

j = 0

Цикл генерації:

i = (i + 1) mod 2n

j = (j + S[i]) mod 2n

Перестановка(S[i], S[j])

Результат: K = S[(S[i] + S[j]) mod 2n]

Безпека

На відміну від сучасних шифрів (таких, як в eSTREAM), RC4 не використовує окремої оказії (англ. nonce) поряд із ключем. Це значить, що якщо один ключ повинен використовуватися протягом довгого часу для управління даними з ціллю збереження конфіденційності інформації декількох потоків, сама криптосистема, що використовує RC4, повинна комбінувати оказію й довгостроковий ключ для одержання потокового ключа для RC4. Один з можливих виходів – генерувати новий ключ для RC4 за допомогою хеш-функції від довгострокового ключа й оказії. Однак багато додатків, що використовують RC4, просто конкатенують ключ і оказію. Через цей і слабкий розклад ключів, використовуваного в RC4, додаток може стати уразливим.

Тут будуть розглянуті деякі атаки на шифр і методи захисту від них.

Маніпуляція бітами

Шифр RC4 украй уразливий до маніпуляції бітами, якщо він не використовується вірним образом, як і будь-який потоковий шифр. Тому він був визнаний застарілим багатьма софтверними компаніями, такими як Microsoft. Наприклад, в.NET Framework від Microsoft відсутня реалізація RC4.

Дослідження Руза й відновлення ключа з перестановки

В 1995 році Андрю Руз (англ. Andrew Roos) експериментально визначив, що перший байт ключового потоку корельований з першими трьома байтами ключа, а перші декілька байт перестановки після алгоритму розкладу ключів (англ. KSA) корельовані з деякою лінійною комбінацією байт ключа. Ці зсуви не були доведені до 2007 року, коли Піл, Рафі й Мейтре довели корельованість ключа й ключового потоку. Також Підлога й Мейтре довели корельованість перестановки й ключа. Остання робота також використовує корельованість ключа й перестановки для того, щоб створити перший алгоритм повного відновлення ключа з останньої перестановки після KSA, не роблячи припущень про ключ і вектор ініціалізації (англ. IV or Initial Vector). Цей алгоритм має постійну ймовірність успіху залежно від часу, що відповідає квадратному кореню зі складності повного перебору. Пізніше було зроблено багато робіт про відновлення ключа із внутрішнього стану RC4.

Атака Флурера, Мантина й Шаміра (ФМШ)

В 2001 році, Флурер, Мантин і Шамір опублікували роботу про уразливість ключового розкладу RC4. Вони показали, що серед всіх можливих ключів, перші декілька байт ключового потоку є зовсім невипадковими. Із цих байт можна з високою ймовірністю одержати інформацію про використовуваний шифром ключі. І якщо довгостроковий ключ і оказія (англ. nonce) просто конкатенуються для створення ключа шифру RC4, те цей довгостроковий ключ може бути отриманий за допомогою аналізу досить великої кількості повідомлень, зашифрованих з використанням даного ключа [4]. Ця уразливість і деякі пов'язані з нею ефекти були використані при взломі управління даними з ціллю збереження конфіденційності інформації WEP у бездротових мережах

стандарту IEEE 802.11. Це показало необхідність якнайшвидшої заміни WEP, що спричинило розробку нового стандарту безпеки бездротових мереж WPA.

Криптосистему можна зробити несприйнятливої до цієї атаки, якщо відкидати початок ключового потоку. Таким чином, модифікований алгоритм називається «RC 4-drop[n]», де n – кількість байт із початку ключового потоку, які варто відкинути. Рекомендовано використовувати n = 768, консервативна оцінка становить n = 3072 [5][6].

Атака Кляйна

В 2005 році Андреас Кляйн представив аналіз шифру RC4, у якому він указав на сильну корельованість ключа й ключового потоку RC4. Кляйн проаналізував атаки на першому раунді (подібні до атаки ФМШ), на другому раунді й можливі їхні поліпшення. Він також запропонував деякі зміни алгоритму для посилення стійкості шифру. Зокрема, він затверджує, що якщо поміняти напрямок циклу на зворотне в алгоритмі ключового розкладу, то можна зробити шифр більше стійким до атак типу ФМШ [1].

Комбінаторна проблема

В 2001 році Аді Шамір і Іцхак Мантін першими поставили комбінаторну проблему, пов'язану з кількістю всіляких вхідних і вихідних даних шифру RC4. Якщо із усіляких 256 елементів внутрішнього стану шифру відомо x елементів зі стану (x ? 256), те, якщо припустити, що інші елементи нульові, максимальне кількість елементів, які можуть бути отримані детермінованим алгоритмом за наступних 256 раундів, також дорівнює x. В 2004 році це припущення було доведено Сорадюти Полом (англ. Souradyuti Paul) і Бартом Пренілом (англ. Bart Preneel) [7].

Реалізація

Робота багатьох потокових шифрів заснована на лінійних регістрах зрушення зі зворотним зв'язком (LFSR). Це дозволяє досягти високої ефективності реалізацій шифру у вигляді ІС, але утрудняє програмну реалізацію таких шифрів. Оскільки шифр RC4 не використовує LFSR і заснований на

байтових операціях, його зручно реалізовувати програмно. Типова реалізація виконує від 8 до 16 машинних команд на кожний байт тексту, тому програмна реалізація шифру повинна працювати дуже швидко [8].

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання дипломного проектування, наведена на рисунку 3.3.

Рисунок 3.3 – Діаграма взаємодії процесів

Наведемо покрокову послідовність взаємодії процесів, яка виконується після початку роботи ПЗ:

– Процес роботи головного вікна розробленого ПЗ.

– Процес роботи бібліотеки потокового алгоритму управління даними.

– Процес роботи модуля захисту ПЗ.

– Процес введення шляхів збереження даних.

– Процес моніторингу даних зовнішнього жорсткого диску інтерфейсу USB 3.0.

– Процес роботи блоку розподілу ключів системи управління даними.

– Процес отримання даних з зовнішнього жорсткого диску.

– Процес відправлення даних на зовнішній жорсткий диск.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання послідовності кроків.

Розглянемо по блокам ці послідовності з описом стадії роботи.

1 стадія завантаження ПЗ. Виконуються наступні кроки:

– Підключення модулів ПЗ.

– Підключення потокового алгоритму управління даними.

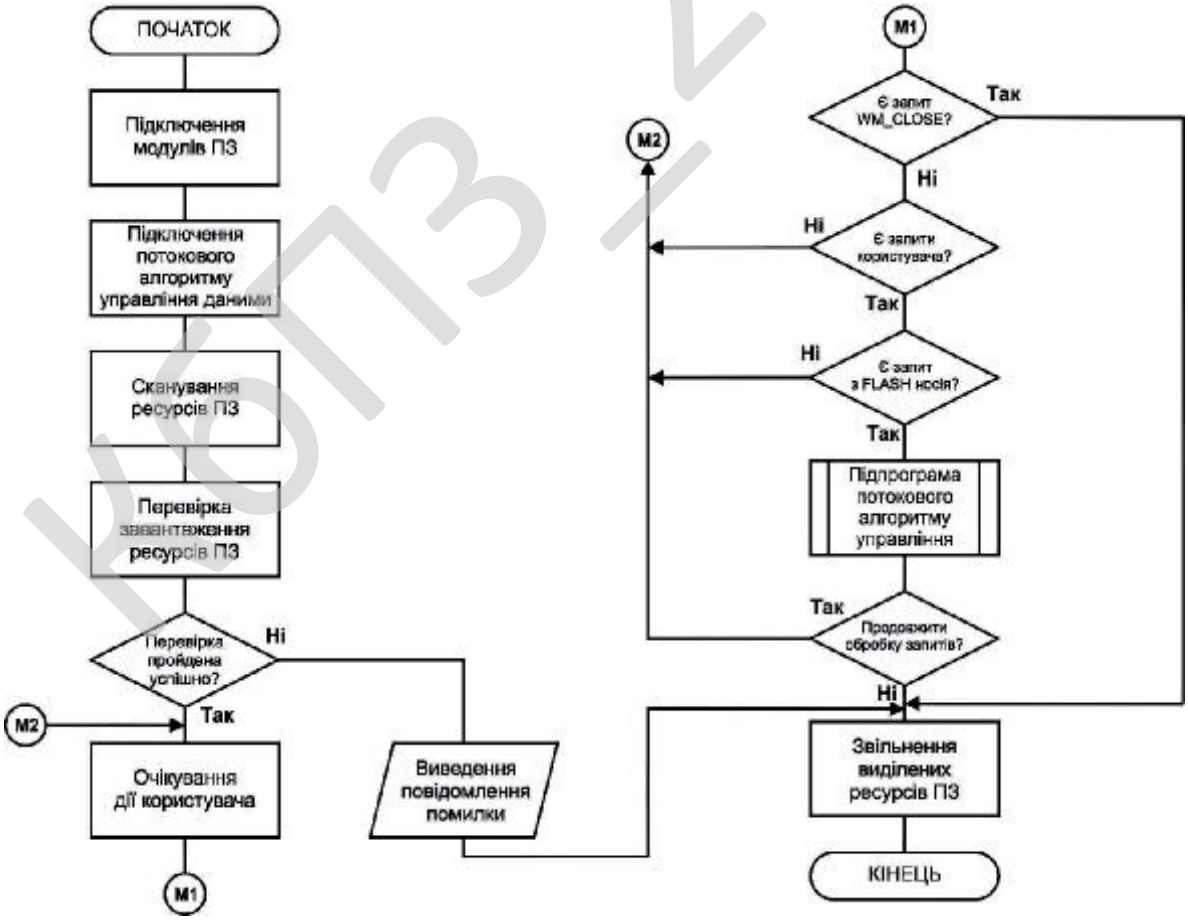


Рисунок 4.1 – Блок-схема основної програми

– Сканування ресурсів ПЗ.

– Перевірка завантаження ресурсів ПЗ.

– Перевірка пройдена успішно?

2 стадія очікування дій користувача. Виконуються наступні кроки:

– Очікування дії користувача.

– Є запит WM_CLOSE?

– Є запити користувача?

– Є запит з FLASH носія?

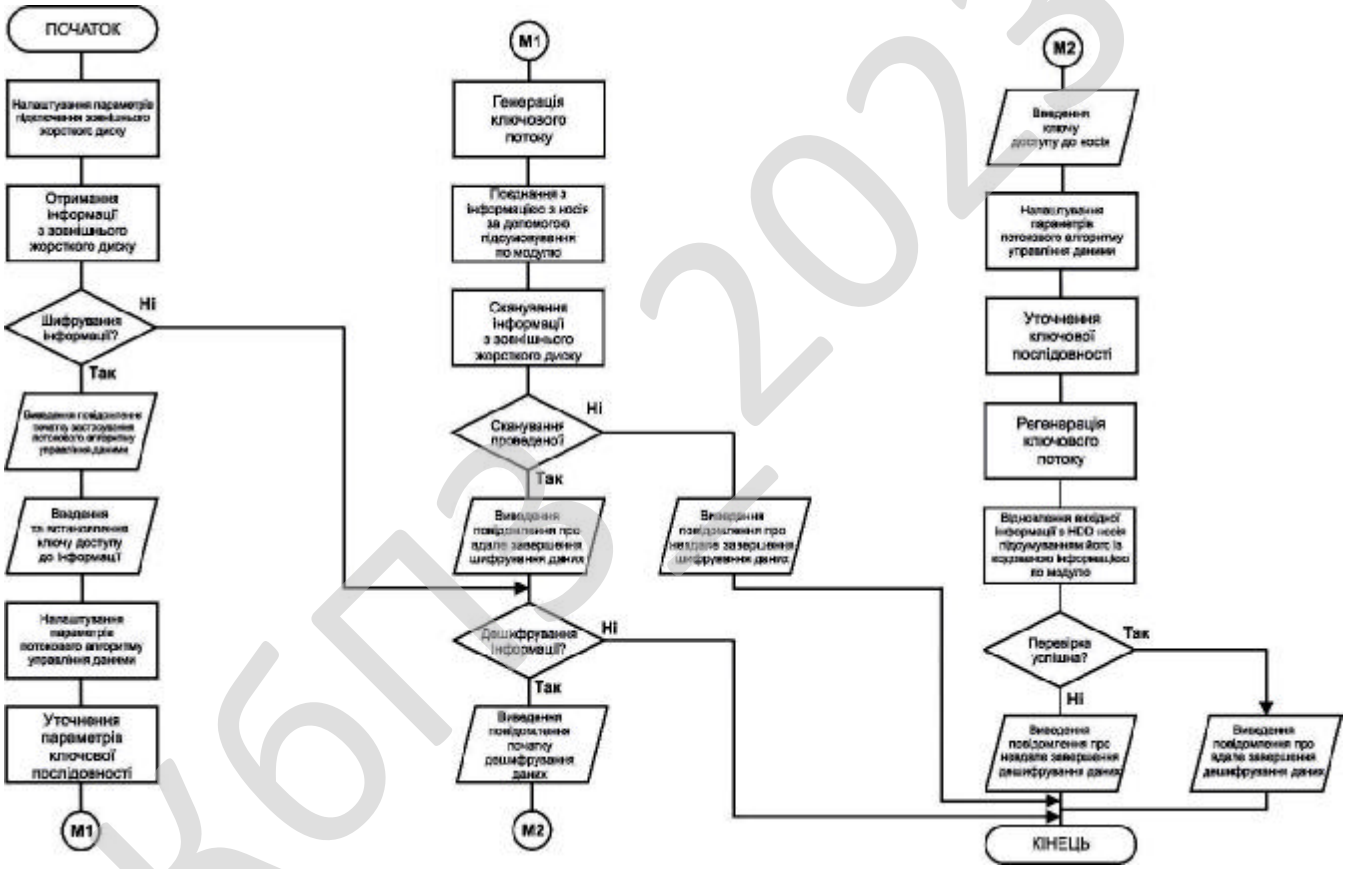


Рисунок 4.2 – Блок-схема роботи підпрограми

3 стадія виклик і робота підпрограми. Виконуються наступні кроки:

– Виклик підпрограми потокового алгоритму управління.

– Налаштування параметрів підключення зовнішнього жорсткого диску.

– Отримання інформації з зовнішнього жорсткого диску.

– Шифрування інформації?

– Виведення повідомлення початку застосування потокового алгоритму управління даними.

– Введення та встановлення ключу доступу до інформації.

– Налаштування параметрів потокового алгоритму управління даними.

– Уточнення параметрів ключової послідовності.

– Генерація ключового потоку.

– Поєднання з інформацією з носія за допомогою підсумовування по модулю.

– Сканування інформації з зовнішнього жорсткого диску.

– Сканування проведено?

– Виведення повідомлення про вдале завершення шифрування даних.

– Дешифрування інформації?

– Виведення повідомлення початку дешифрування даних.

– Введення ключу доступу до носія.

– Налаштування параметрів потокового алгоритму управління даними.

– Уточнення ключової послідовності.

– Регенерація ключового потоку.

– Відновлення вихідної інформації з HDD носія підсумуванням його із кодованою інформацією по модулю.

– Перевірка успішна?

– Виведення повідомлення про вдале завершення дешифрування даних.

4 стадія завершення роботи ПЗ. Виконуються наступні кроки:

– Продовжити обробку запитів?

– Звільнення виділених ресурсів ПЗ.

Розглянемо теорію по темі дипломного проектування.

Функція копіювання файлів. У Delphi є функція CopyFile.

```
BOOL CopyFile (
LPCTSTR lpExistingFileName, // pointer to name of an existing file
LPCTSTR lpNewFileName, // pointer to filename to copy to
BOOL bFailIfExists // flag for operation if file exists
);
```

Параметри передаються в цю функцію:

1. Покажчик на ім'я існуючого файлу (нуль термінований рядок тобто тип PChar).

2. Покажчик на ім'я файлу, який буде створений/перезаписаний після копіювання (нуль термінований рядок – тип PChar).

3. Якщо цей параметр True і файл з таким ім'ям вже існує, то функція поверне False. Якщо ж файл, з ім'ям зазначеним у другому параметрі існує і в якості третього параметра переданий False – то функція перезапише файл і благополучно завершиться.

Як приклад.

```
procedure TForm1.Button1Click (Sender: TObject);
begin
if CopyFile ('c:\1.Txt','c:\2.Txt',true) then
ShowMessage ('Файл успішно скопійований!')
else ShowMessage ('Невдача!');
end;
```

Для того, щоб точніше довідатися при виникненні помилки, що ж все таки сталося, треба скористатися функцією GetLastError, яка повертає код останньої помилки (формат DWORD). Тепер ми трохи змінимо приклад:

```
procedure TForm1.Button1Click (Sender:  TObject);
begin
if CopyFile ('c:\1.Txt','c:\2.Txt',true) then
ShowMessage ('Файл успішно скопійований!')
else ShowMessage('Помилка! Ось її код:'+IntToStr(GetLastError));
end;
```

Таким чином натиснувши вдруге на кнопку ми отримаємо повідомлення: "Помилка! Ось її код: 80". Це говорить нам, що файл існує. Коди всіх помилок можна легко знайти у файлі допомоги.

Розглянемо приклад копіювання файлів за допомогою файлового потоку (TFileStream). В наведеної користувача функції введені два додаткові параметри From і Count, які вказують, відповідно, від якого і з якою байт потрібно копіювати файл. Якщо необхідно скопіювати весь файл, то необхідно передати нулі. Ось код цієї функції:

```
function MyCopyFile(InFile, OutFile:String; From, Count:Longint):Longint;
var
InFS, OutFS:TFileStream;
begin
InFS:= TFileStream.Create(InFile, fmOpenRead); // створюємо потік
OutFS:= TFileStream.Create(OutFile, fmCreate); // створюємо потік
InFS.Seek(From, soFromBeginning) ;// переміщаємо покажчик в From
Result:=OutFS.CopyFrom (InFS, Count);
InFS.Free ;// звільняємо
OutFS.Free ;// звільняємо
end;
```

Видалення файлів. Для видалення файлів в Delphi так само передбачена спеціальна процедура DeleteFile. В якості параметра, переданого у функцію, виступає рядок типу PChar, яка вказує ім'я файлу, який потрібно видалити.

```
procedure TForm1.Button1Click (Sender: TObject);
begin
if DeleteFile ('c:\2.txt') then
ShowMessage('Файл успішно знищений!')
else
ShowMessage ('Помилка! Ось її код:' + IntToStr (GetLastError));
end;
```

Видалення порожньої директорії. Щоб видалити порожню директорію за допомогою Delphi досить звернутися до функції RemoveDir.

```
function RemoveDir (const Dir:  string): Boolean;
```

Ця функція повертає True якщо директорія, зазначена в єдиному параметрі, переданому у функцію, успішно видалена, в іншому випадку функція поверне False.

Часто виникає необхідність видалити не порожню папку, яка містить не лише файли, але й інші вкладені папки. Для цього була написана користувальницька функція, що видаляє папку з усіма файлами і піддиректоріями.

```
Function MyRemoveDir (sDir:  String): Boolean;
Var
iIndex: Integer;
SearchRec: TSearchRec;
sFileName: String;
begin
```

```
Result:= False;
sDir:= sDir+'\*.*';
iIndex:= FindFirst (sDir, faAnyFile, SearchRec);
while iIndex = 0 do begin
sFileName:= ExtractFileDir (sDir) + '\' + SearchRec.Name;
if SearchRec.Attr = faDirectory then begin
if (SearchRec.Name <>'') and (SearchRec.Name <> '.') and (SearchRec.Name <> '..')
then
MyRemoveDir (sFileName);
end else
begin
if SearchRec.Attr<>faArchive then  FileSetAttr (sFileName, faArchive);
if NOT DeleteFile (sFileN ame) then Sh owMessage ('Could NOT delete' + sFileName);
end;
iIndex:=FindNext(SearchRec);
end;
FindClose(SearchRec);
RemoveDir(Extr actFileDir(sDir));
Result:=True;
end;
```

Приклад використання цієї функції:

```
procedure TForm1.Button1Click (Sender:  TObject);
begin
if MyRemoveDir ('C:\testDir') then ShowMessage ('Директорія успішно видалена')
else ShowMessage ('Не виходить видалити директорію');
end;
```

При роботі з файлами необхідно.

1. Перед копіюванням або видаленням файлів завжди перевіряти його наявність функцією FileExists:

```
if FileExists ('c:\1.txt') then
if CopyFile ('c:\1.txt', 'c:\2.txt', true) then
ShowMessage ('Файл успішно скопійований!')
```

2. Використовувати в функціях CopyFile і DeleteFile імена файлів отримані за допомогою, наприклад, OpenDialog:

```
if CopyFile (Pchar (OpenDialog1.FileName),Pchar(SaveDialog 1. FileName), true)
then ...
```

3. Завжди стежити за іменами файлів, використовуваних у функціях. Звертати увагу на абсолютні та відносні шляхи. Через це часто виникають проблеми, які складно відстежити.

Розглянемо реалізований метод в дипломі.

```
unit uShell; // назва модулю
interface // інтерфейс
// Модуль копіювання файлів в буфер для провідника
uses Forms, Classes, SysUtils, ShellApi, ShlObj, ClipBrd, Windows;
type
  TWinMoveTy pe = (wCopy = 5, wMove = 2); // Копіювання або перенос
  WinExplorer = class
  // Файли в буфер
class procedure FilesToClipboard(const Files:TStrings;MoveType:TWinMoveType =
                                 wCopy); overload;
class procedure FilesToClipboard(const Folder:string; Files:TStrings;
                                 MoveType:TWinMoveType = wCopy); overload;
class procedure FilesToClipboard(const FilePath:string; MoveType:TWinMoveType=
                                 wCopy); overload;
// Виконати операцію вставки в папку з буфера
class procedure ClipboardPaste (const Target: string);
// Отримати список файлів в буфері
class procedure ClipBoardGetFiles (const Files: TStrings; h: THandle = 0);
class function DeleteToRecycle (const FileName: string; Wnd: HWND = 0):Boolean;
                                 overload;
class function DeleteToRecycle (const Files: TStrings; Wnd: HWND = 0): Boolean;
                                 overload;
class function DeleteToRecycle (const Folder:string; const Files:TStrings;
                                 Wnd:HWND = 0): Boolean; overload;
end;

implementation //реалізація
```

Функція перетворює TStirngs в спеціальний рядок для Shell цей рядок для буфера обміну, де рядки розділені знаком #0 і весь рядок закінчується #0 #0.

```
function Shell_Str (Strs: TStrings): string;
var
  i:Integer;
begin
  for i:= 0 to Strs.Count - 1 do
    Result:= Result + Strs.Strings [i] + #0;
```

```
   Result:= Result +  #0;
end;
function Shell_DataOperations (const Source, Target:string; Operacia,
                                Flags:Integer):Boolean;
```

Функція для копіювання/вирізування/перейменування/видалення даних засобами API.

Операція:

– FO_COPY.

– FO_MOVE.

– FO_RENAME.

– FO_DELETE.

Прапори:

– FOF_ALLOWUNDO. Якщо можливо, зберігає інформацію для можливості UnDo. Якщо необхідно не просто видалити файли, а перемістити їх у кошик, повинен бути встановлений прапор.

– FOF_CONFIRMMOUSE. Не реалізовано.

– FOF_FILESONLY. Якщо в полі pFrom встановлено *. *, то операція буде проводитись тільки з файлами.

– FOF_MULTIDESTFILES. Вказує, що для кожного вихідного файлу в полі pFrom вказана своя директорія – адресат.

– FOF_NOCONFIRMATION. Відповідає "yes to all" на всі запити в ході операції.

– FOF_NOCONFIRMMKDIR. Не підтверджує створення нового каталогу, якщо операція вимагає, щоб він був створений.

– FOF_RENAMEONCOLLISION. У випадку, якщо вже існує файл з даними ім'ям, створюється файл з ім'ям "Copy # N of ...".

– FOF_SILENT. Не показувати діалог з індикатором прогресу.

– FOF_SIMPLEPROGRESS. Показувати діалог з індикатором прогресу, але не показувати імен файлів.

– FOF_WANTMAPPINGHANDLE. Вносить hNameMappings елемент. Дескриптор повинен бути звільнений функцією SHFreeNameMappings.

```
var
  SHOS:TSHFileOpStruct;
begin
  FillChar (SHOS, SizeOf (SHOS),  #0);
  SHOS.Wnd:= 0;
  SHOS.wFunc:= Operacia;
  SHOS.pFrom:= PCHAR (Source);
  SHOS.pTo:= PCHAR (Target);
  SHOS.fFlags:= Flags;
  Result:= (SHFileOperation (SHOS) = 0) and (not SHOS.fAnyOperationsAborted);
end;
```

Відправляє файли/папки в буфер обміну. 5 – копіювання (ніби натиснуто Ctrl + C). 2 – вирізка (ніби натиснута Ctrl + X). DROPEFFECT_MOVE щоб потім можна було вставити (Ctrl + V) ці дані в якому файловому менеджері.

```
procedure Clipboard_DataSend (const DataPaths: TStrings; const MoveType: Integer);

var
  DropFiles: PDropFiles;
  hGlobal: THandle;
  iLen: Integer;
  f: Cardinal;
  d: PCardinal;
  DataSpecialList: string;
  // Список адрес (FullPaths) файлів / папок які треба копіювати
begin
  Clipboard. Open;
  Clipboard. Clear;
  // Перетворюємо в спеціальний рядок
  DataSpecialList:= Shell_Str (DataPaths);
  iLen:= Length (DataSpecialList) * SizeOf (Char);
  hGlobal:= GlobalAlloc (GMEM_SHARE or GMEM_MOVEABLE or GMEM_ZEROINIT,
    SizeOf (TDropFiles) + iLen);
  Win32Check (hGlobal <> 0);
  DropFiles:= GlobalLock (hGlobal);
  DropFiles ^. PFiles:= SizeOf (TDropFiles);
{$ IFDEF UNICODE}
  DropFiles ^. FWide:= true;
{$ ENDIF}
Move(DataSpecialList [1],(PansiChar (DropFiles)+SizeOf(TDropFiles))^,iLen);
  SetClipboardData (CF_HDROP, hGlobal);
  GlobalUnlock (hGlobal);
```

```
  // Для копіювання
  begin
    f:= RegisterClipboardFormat (CFSTR_PREFERREDDROPEFFECT);
    hGlobal:= GlobalAlloc (GMEM_SHARE or GMEM_MOVEABLE or GMEM_ZEROINIT,
      SizeOf (dword));
    d:= PCardinal (GlobalLock (hGlobal));
    d ^:= MoveType; // 2-Cut, 5-Copy
    SetClipboardData (f, hGlobal);
    GlobalUnlock (hGlobal);
  end;
  Clipboard.Close;
end;
```

Функція визначає, на що послані дані в буфер: на вирізку або копіювання. Ця функція створювалася спеціально для функції ClipBoard_DataPaste, щоб було зрозуміло що робити: копіювати або вирізати.

```
function Clipboard_SendType: Integer;
{
  5 - copy
  2 - cut
}
var
  ClipFormat, hn: Cardinal;
  szBuffer: array [0 .. 511] of Char;
  FormatID: string;
  pMem: Pointer;
begin
  Result:= 0;
  if not OpenClipboard (Application.Handle) then exit;
try
    ClipFormat:= EnumClipboardFormats (0);
    while (ClipFormat <> 0) do
    begin
      GetClipboardFormatName (ClipFormat, szBuffer, SizeOf (szBuffer));
      FormatID:= string (szBuffer);
      if SameText (FormatID, 'Preferred DropEffect') then
      begin
        hn:= GetClipboardData (ClipFormat);
        pMem:= GlobalLock (hn);
        Move (pMem ^, Result, 4); // <- тепер в Result тип операції
        GlobalUnlock (hn);
        Break;
```

```
      end;
      ClipFormat:= EnumClipboardFormats (ClipFormat);
    end;
  finally
    CloseClipboard;
  end;
end;
{WinExplorer}
class procedure WinExplorer.FilesToClipboard (const Files: TStrings; MoveType
                                              :TWinMoveType);
begin
  Clipboard_DataSend (Files, Integer (MoveType));
end;
class procedure WinExplorer.FilesToClipboard (const Folder:string; Files
                                               :TStrings; MoveType:TWinMoveType);
var
  i: Integer;
  s: TStringList;
begin
  s:= TStringList.Create;
  for i:= 0 to Files.Count - 1 do
    s.Add (Folder + Files [i]);
  Clipboard_DataSend (s, Integer (MoveType));
  s.free;
end;
```

Натиснуто Ctrl+C або Ctrl+X – послали дані в буфер обміну. Ця функція повертає список файлів/папок, які надіслані в буфер.

```
class procedure WinExplorer.ClipBoardGetFiles(const Files:TStrings; h:THandle);
var
  FilePath: array [0 .. MAX_PATH] of Char;
  i, FileCount: Integer;
begin
  Files.Clear;
  if h = 0 then
  begin
    if not Clipboard.HasFormat (CF_HDROP) then
      exit;
    Clipboard.Open;
    try
      h:= Clipboard.GetAsHandle (CF_HDROP);
    finally
```

```
      Clipboard.Close;
    end;
  end;
  if h = 0 then
    exit;
  FileCount:= DragQueryFile (h, $ FFFFFFFF, nil, 0);
  for i:= 0 to FileCount - 1 do
  begin
    DragQueryFile (h, i, FilePath, SizeOf (FilePath));
    Files.Add (FilePath);
  end;
end;
```

Ця функція вставить з буфера обміну файли / папки, які копіювали/вирізали (Ctrl+C / Ctrl+X) в буфер в якомусь файловому менеджері (Провідник, Тотал-коммандер). Target – папка в яку буде вставлено дані. Clipboard_OperationType – підфункція яка визначає що треба зробити, копіювати або Вирізати.

```
class procedure WinExplorer.ClipboardPaste (const Target: string);
var
  h: THandle;
  Sourse, sr: string;
  s: TStringList;
begin
  // Якщо те, що перебуває в буфері
  // не є файлами/папками, які треба копіювати/вирізати, то виходимо
  // CF_HDROP - дескриптор який ідентифікує список файлів.
  // прикладна програма може витягувати інформацію про файли, передаючи
  // дескриптор функції DragQueryFile.
if not Clipboard.HasFormat (CF_HDROP) then
    exit;
Clipboard.Open;
  try
    h:= Clipboard.GetAsHandle (CF_HDROP);
    if h <> 0 then
    begin
      s:= TStringList.Create;
      WinExplorer.ClipBoardGetFiles (s, h);
      Sourse:= Shell_Str (s);
      s.free;
      sr:= Copy (Sourse, 0, Pos (#0, Sourse) - 1); // Path № 1
```

```
      sr:= ExtractFilePath (sr); // Батьківська папка Data
      if IncludeTrailingBackslash (sr) = IncludeTrailingBackslash (Target)
      then
// Робимо копію фала: звідки copy туди і paste
      begin
        case Clipboard_SendType of
          5:
            Shell_DataOperations (Sourse, Target, FO_COPY, FOF_SIMPLEPROGRESS or
              FOF_RENAMEONCOLLISION);
          2:
            Shell_DataOperations (Sourse, Target, FO_MOVE, FOF_SIMPLEPROGRESS);
        end;
      end
      else
      begin
        case Clipboard_SendType of
         5: Shell_DataOperations (Sourse, Target, FO_COPY, FOF_SIMPLEPROGRESS);
         2: Shell_DataOperations (Sourse, Target, FO_MOVE, FOF_SIMPLEPROGRESS);
        end;
      end;
    end;
  finally
    Clipboard.Close;
  end;
end;
class function WinExplorer.DeleteToRecycle (const Files: TStrings;
  Wnd: HWND): Boolean;
begin
  result:= WinExplorer.DeleteToRecycle (Shell_Str (Files), wnd);
end;
class procedure WinExplorer.FilesToClipboard (const FilePath: string;
  MoveType: TWinMoveType);
var
  s: TStringList;
begin
  s:= TStringList.Create;
  s.Add (FilePath);
  Clipboard_DataSend (s, Integer (MoveType));
  s.free;
end;
class function WinExplorer.DeleteToRecycle (const FileName: string; Wnd: HWND):
Boolean;
```

```
var
  FileOp: TSHFileOpStruct;
begin
  FillChar (FileOp, SizeOf (FileOp), 0);
  if Wnd = 0 then
    Wnd:= Application.Handle;
  FileOp.Wnd:= Wnd;
  FileOp.wFunc:= FO_DELETE;
  FileOp.pFrom:= PChar (FileName);
  FileOp.fFlags:= FOF_ALLOWUNDO or FOF_NOERRORUI or FOF_SILENT;
  Result:= (SHFileOperation (FileOp) = 0) and (not FileOp.fAnyOperationsAborted);
end;
class function WinExplorer.DeleteToRecycle (const Folder: string;
  const Files: TStrings; Wnd: HWND): Boolean;
var
  i: Integer;
  s: TStringList;
begin
  s:= TStringList.Create;
  for i:= 0 to Files.Count - 1 do
    s.Add (Folder + Files [i]);
  result:= self.DeleteToRecycle (s, Wnd);
  s.free;
end;
end.
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році. В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки. Агентством по теорії архітектури

персональних комп'ютерів специфіковане використання шифру SEED у протоколах TLS і S/MIME. У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

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

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

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

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ, яке зображено на рисунку 5.1. З рисунку можна побачити що головне вікно ПЗ розподілено на декілька частин.

– Ліва частина вікна – основний функціонал ПЗ:

– Шукати під'єднанні зовнішні диски.

– Видалити зовнішній диски.

– Корегувати швидкість передачі даних.

– Статистика передачі даних.

– Налаштування інтерфейсу USB 3.0.

– Налаштування ПЗ.

– Довідка.

– Авторське право.

Права частина вікна – основна інформація з диску:

– Номер зовнішнього жорсткого диску.

– Ємність диску.

– Інтерфейс обміну.

– Швидкість передачі даних – читання.

– Швидкість передачі даних – запис.

– Час доступу – читання.

– Час доступу – запис.

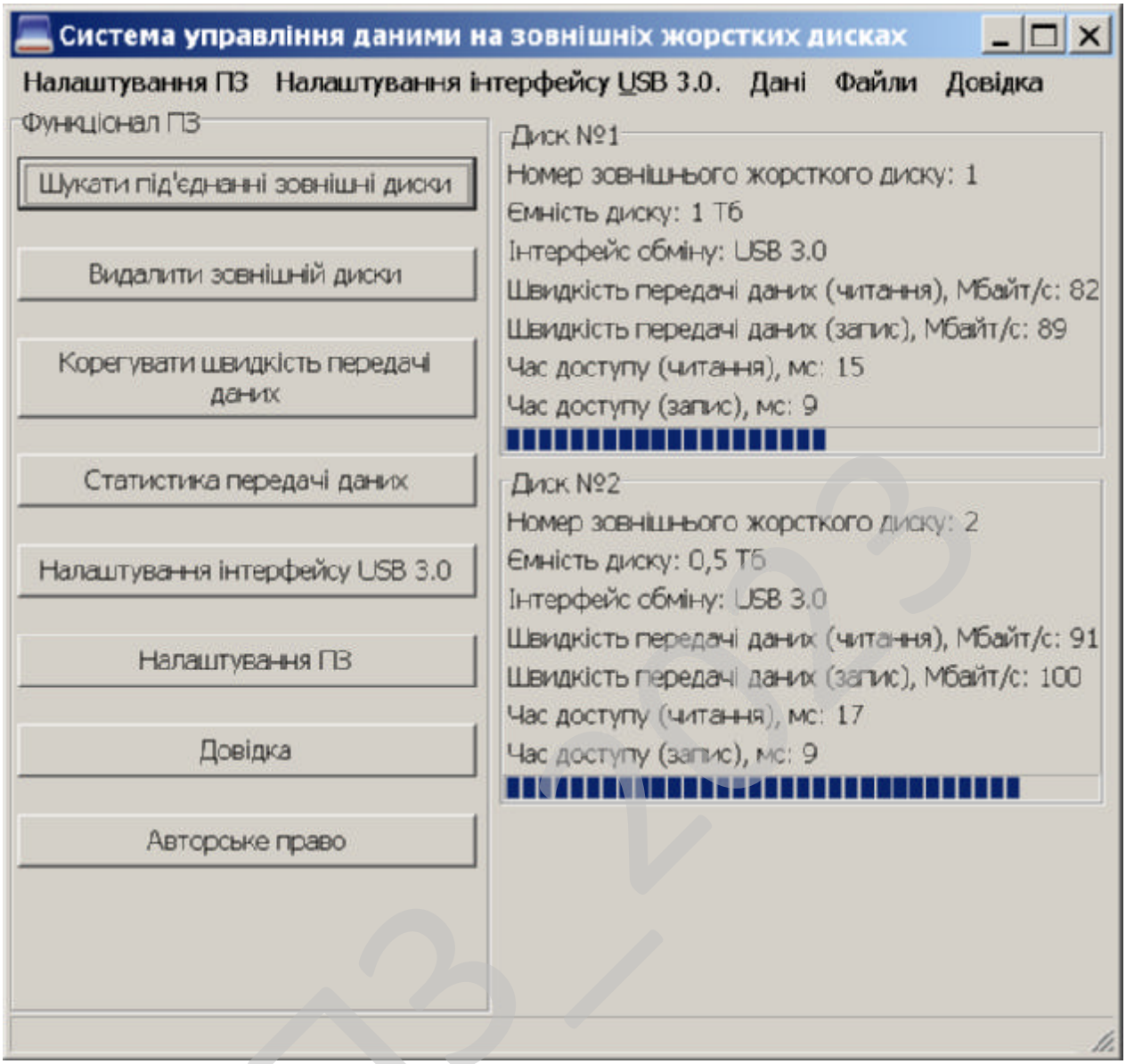


Рисунок 5.1 – Головне вікно ПЗ

Верхнє меню – розширений функціонал розробленого ПЗ:

– Налаштування ПЗ.

– Налаштування інтерфейсу USB 3.0.

– Дані.

– Файли.

– Довідка.

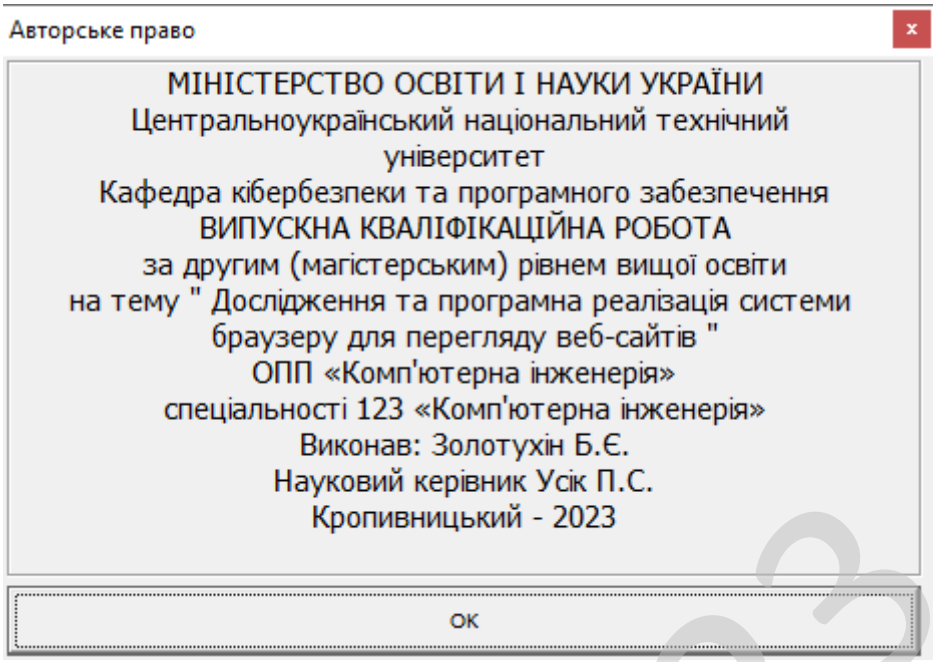


Рисунок 5.2 – Форма авторського права (розробник)

Обраний тип ліцензії – Freeware. Тобто безкоштовне програмне забезпечення це власницьке програмне забезпечення, котре можна безкоштовно використовувати протягом необмеженого терміну без обмежень у функціональності, поширюване без початкових кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають джерельний код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Безкоштовне програмне забезпечення можна безоплатно встановлювати та використовувати, в той час як вільне програмне забезпечення, можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення джерельних кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Метою розробки є дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Об'єктом дослідження є процес управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Предметом дослідження є методи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Методи дослідження базуються на методах теорії архітектури персональних 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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– Розроблено вітчизняний продукт управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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	280
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.	–	28000
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.

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

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

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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 93 = 157 \text{ люд/день.}$$

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	157	Ф 7.1-7.4
Впровадження	13	Д13
Всього	198	–

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.

$$Ч = \frac{198 \cdot 1}{60 - 5} = 3,6 \text{ ставки.}$$

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	2	60	1
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4,17
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	35,33

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{35 \cdot 3}{1,2} = 87,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 87,5/(60 \cdot 8) = 0,18 \text{ ставки.}$$

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

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 2019, 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	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
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	

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.

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,6	8084,5	87312,6
Інженер-електронщик	0,18	7000	3780
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	9000	13500
Всього за період розробки	$R_{сп}$ = 7,03	-	$Ф_{роб}$ = 178842,6

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.

$$З_{сд} = \frac{178843}{7,03 \cdot 60} = 424 \; грн.$$

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:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

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

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

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

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

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

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 по прайсу фірми Компбест за 06.10.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		7347
Процесор	Intel Core i7-4770K 3.5GHz/5GT/s/8MI s1150	-
Системна плата	MSI H81M-S03 (s1150, DDR3, USB 3.(INTEL H81 , PCI-Ex16) з підтримкоі відеоядра процесора	-
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	Samsung DDR3-1333 16Gb PC3-10600R EC(Registered (2x8Gb)	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 450W(FSP Brand: ATX-450PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

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

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 424 \cdot 198/280 = 300 \text{ грн.}$$

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, %.

$$З_д = 300 \cdot 10 \cdot 0{,}01 = 30 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(300+30) = 73 \text{ грн.}$$

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, %.

$$Г_{осп} = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

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, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 22,4 грн./шт., DVD-R box – 33,6 грн./шт.

$$З_{М2} = 100 \cdot 33{,}6 = 3360 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з.}, \quad (7.18)$$

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

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

$$З_М = (105 + 3360 + 1702)/280 = 18 \text{ грн.}$$

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, %.

$$О_п = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 280 прим.):

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

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

$$A_м = 190286 \cdot 3/(280 \cdot 12) = 170 \text{ грн.}$$

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

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

$$C_п = 300+30+73+45+18+45+170 = 681 \text{ грн.}$$

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. Основна зарплата виконавців	$З_о$	300
2. Додаткова зарплата виконавців	$З_д$	30
3. Відрахування на соціальні потреби	$C_{оц}$	73
4. Загальногосподарські витрати	$Г_{осп}$	45
5. Витрати на матеріали	$З_М$	18
6. Освоєння нових операційних систем, мов програмування	$О_п$	45
7. Амортизація основних фондів	$A_м$	170
8. Повна собівартість програмного забезпечення	$C_п$	681
9. Плановий прибуток	$П_р$	341
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	1022
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	204,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1226,4

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 681 = 341 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1226
Всього капітальних витрат	–	1226

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	30000	25000
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	31488	26337

Витрати на профілактичні роботи:

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

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

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

Після купівлі нового програмного забезпечення витрати на обслуговування системи зменшились з 30000 грн до 25000 грн на рік.

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	–	1226	–	306,5
Всього відрахувань	-	–	1226	–	306,5

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

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

$$З_{ел\ баз} = 0{,}545 \cdot 1300 \cdot 2{,}1 = 1487{,}85 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 900 \cdot 2{,}1 = 1030{,}05 \text{ грн.}$$

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:

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

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

$$Ев = (1022 - 681) \cdot 280 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 28000) \cdot 3/12 = 50208 \text{ грн.}$$

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.

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}5 \text{ роки}$$

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; $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (31488\text{-}26337)\text{-}0{,}25 \cdot 1226 = 4845 \text{ грн.}$$

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.	280
2. Повна собівартість розробленої програми	Грн.	681
3. Ціна розробленої програми	Грн.	1022
4. Плановий прибуток від реалізації розробленої програми	Грн.	341
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	95480
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	50208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	4845
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,24

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{1226}{31488 - 26337} = 0,24 \ року .$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Комп’ютер – невід’ємна складова сучасного життя. За допомогою обчислювальної техніки вирішують складні робочі задачі, ведуться наукові дослідження, створюються архітектурні креслення і твори мистецтва. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп’ютерної техніки. Незважаючи на видиму безпеку та розвитку сучасних технологій, при роботі за комп'ютером є ряд чинників, які можу вплинути на здоров'я людини. Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп’ютером.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями, затверджені наказом Мінсоцполітики від 14.02.2018р. № 207, зареєстровані в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 [2].

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ.

У розділі даної магістерської роботи висвітлюються основні питання охорони праці працівників, робота яких пов'язана з роботою за комп'ютером, планування робочого приміщення, де працюють користувачі ПК; параметри мікроклімату, освітленість робочих місць та виробничих приміщень; шумові завади.

Правильна організація і раціональне устаткування робочого місця можливість ефективно і з якнайменшими витратами праці виконувати свої функції, плідно спілкуватися співробітниками і підлеглими, підтримувати високу працездатність і робочий настрій.

Велике значення має раціональна конструкція і розташовує елементів робочого місця, що важливе для підтримки оптимальної робочої пози людини-оператора, а також необхідно дотримувати правильний режим праці і відпочинку.

Що стосується питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності.

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

Робота працівників пов'язана з роботою за комп'ютером, тому актуальною є розгляд саме умов праці та стану охорони праці працівників які постійно працюють з комп'ютерною технікою.

Завдання даного розділу полягає у тому, щоб розробити якісний програмний продукт необхідно організувати безпеку на робочому місці програміста. Під час проектування безпеки робочому місці з ПК необхідно домагатися високої якості та надійності технічного забезпечення, але й створювати комфортні параметри довкілля для розробників.

8.2 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	2,75

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
Обсяг, V	$м^3$	не менше 20.0	24,3

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працюють 4 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], 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.

8.3 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

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

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

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

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

8.4 Розрахункова частина

Для захисного штучного заземлення будем застосовувати ветикальні електроди з сталевого прокату круглого перерізу діаметром 35 мм., довжиною *L=1,5 м.,* та горизонтальний електрод – метелева полоса з перетином 35·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,5 м*. Відстань між вертикальними заземлювачами (електродами) *A*=2 *м*. Глибина закладення горизонтального контура заземлення *t=0,7 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+1,5/2=1,45 \text{ м.}$$

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

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

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м*. – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр ветикальнього електрода (задан):

$$Dв = 35 \text{ мм.} = 0,035 \text{ м.}$$

Відношення *A/L=2/1,5=1,33*.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O=0,366(\rho/L)[lg(2L/Dв)+(1/2)lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/1,5)[(lg(2\cdot1,5/0,035)+(1/2\ )lg((4\cdot1,45+1,5)/(4\cdot1,45-1,5))]=$$

$$=27,2\ Ом.$$

Визначаємо коефіцієнт екранування ветикальньних електродів $Кев=0,53$ при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [11].

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

$$N=R_O\ /(Кев\ R_{ЗН})= 27,2\ /\ (0,53\cdot4)=12,8 \approx 13\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot13=26,9 \approx 30\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П =0,366(\rho\cdot K_П/L_П\ )lg(2(L_П\cdot L_П\ )/(B\cdot t))=$$

$$=0,366(40\cdot5/30)\cdot lg((2\cdot30^2)/(0,035\cdot0,7))=20,6\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [11]:

$B = 35\ мм. = 0,035\ м.$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [11]:

$$R =(R_O\cdot R_П)/(\ R_O\cdot\eta_П+\ N\cdot R_П\cdot Кев)=$$

$$=(27,2\cdot\ 20,6)/(27,2\cdot\ 0,55+\ 7\cdot\ 20,6\cdot\ 0,53)=3,38\ Ом.$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \le R_{ЗН}$ виконується ($3,38 \le 4$).

Так як R суттево быльше $R_{ЗН}$, зменшимо кількість вертикальних електродів до 11 і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює *11* при $R =3,9\ Ом.$

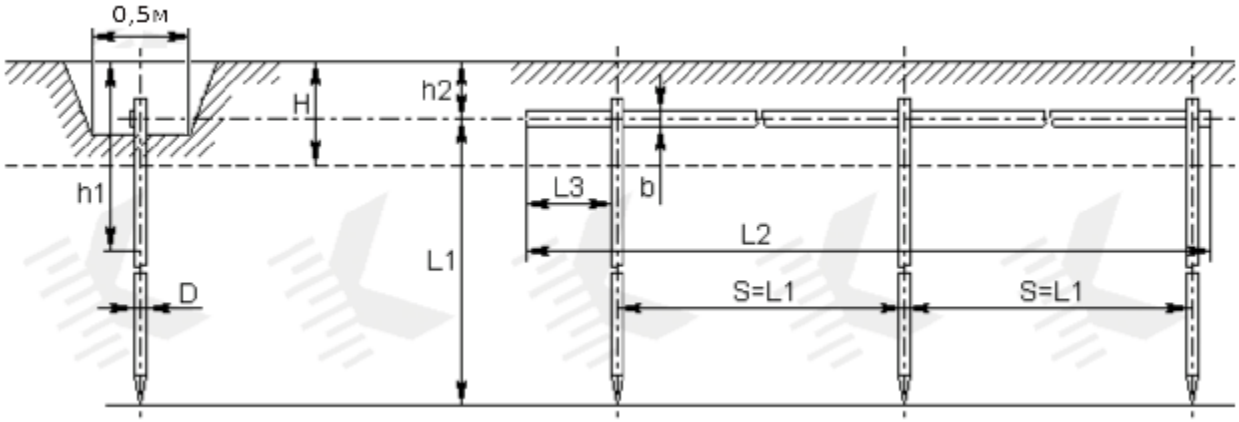


Рисунок 8.1 – Схема штучного заземлення.

При необхідності можна зменшити кількість електродів заземлювача зменшивши загальньний опір розтіканню електричного струму заземлювача методом зменшення питомого опору ґрунта, домішуючи у грунт безпосередньо навколи електродів заземлювача розчини солей NaCl, CaCl, сажу, соду, шлак, коксову дрібницю, або спеціальні суміші.

8.5 Висновки до розділу

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– Досліджена система управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SEED.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 4845 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,24 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Золотухін Б.Є. Дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0 // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. M. Holmes He. Creating Apps with React Native. Apress Media. 2022. 445 p.
3. Maurício Aniche. Effective Software Testing. Manning Publications. 2021. 372 p
4. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.
5. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.
6. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.
7. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.
8. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.
9. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.
10. 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.
11. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
12. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

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

14. 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.

15. 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.

16. 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.

17. 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.

18. 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

19. 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.

20. 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.

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

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

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

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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).

39. 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.

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

41. 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.

42. 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.

43. 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

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0.

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 управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0;

– 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 – 124 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 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Усік П.С.

Дослідження та програмна реалізація системи управління даними зовнішніх жорстких дисків з інтерфейсом USB 3.0

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 38

Літера: РП

Кропивницький – 2023 року

ПРОЕКТ ПРОГРАМИ FLASH_HDD.DPR

```
program FLASH_HDD; // Назва програми

{
Золотухін Богдан Євгенійович, 2023
}

Uses // Підключаємо бібліотеки
  Forms,
  Frm1 in 'Frm1.pas' {Form1},
  Frm2 in 'Frm2.pas' {Form2},
  _LOG in 'HDD_.pas' {Form3},
  Frm3 in 'HDD__DATA.pas'{Form4},
  Frm5 in 'Frm2.pas' {AboutBox};
Var // Об’ява типів даних
  M: HWND;
{$R *.res} // Підлючення файлу ресурсів
Begin // Початок роботи
  Application.Initialize; // Ініціалізація
M:=CreateFileMapping(HWND($FFFFFFFF), Nil, PAGE_READWRITE, 0, MemFileSize,
MemFileName);
// Створення файлу відображеного у пам’ять
  Application.CreateForm(TForm1, Form1); // Створення форми №1
  Application.CreateForm(TForm2, Form2); // Створення форми №2
  Application.CreateForm(TForm3, Form3); // Створення форми №3
  Application.CreateForm(TForm4, Form4); // Створення форми №4
  Application.CreateForm(TAboutBox, AboutBox); // Створення форми №5
  Application.Run; // Запуск
end.
```

Модуль моніторингу даних HDD_DATA.pas

```
Unit HDD_DATA; // Назва модулю
{
Золотухін Богдан Євгенійович, 2023
}
Interface // Інтерфейс модулю
Type // Об’ява власних структур
TDA = ^longint;
TCompare = (Lt, St, Eq, Er);
TSign = (negative, positive);
THDD_DATA = Record // структура
Sign: TSign;
Number: TDA;
End;
Procedure HDD_DATADestroy(Var GInt: THDD_DATA);
Procedure HDD_DATACopy(Var GInt, Copied: THDD_DATA);
Procedure zeronetochar8(Var g: char; Var x: String);
Procedure zeronetochar6(Var g: integer; Var x: String);
Function IntToStr1(Var b: integer): String;
// Опис структури
Function StrToInt4(Var S: String): longint;
Function IntToStr4(i: longint): String;
Function min(i1, i2: longint): longint;
Procedure Setlength(Var number: TDA; oldsize, newsize: longint);
Procedure initialize8(Var trans: Array Of String);
Procedure ConvertBase2to256(str2: String; Var str256: String);
Procedure ConvertBase2to64(str2: String; Var str64: String);
Procedure ConvertBase256StringToHexString(Str256: String; Var HexStr: String);
Procedure ConvertHexStringToBase256String(HexStr: String; Var Str256: String);
Procedure ConvertBase256to64(Var str256, str64: String);
Procedure ConvertBase64to256(str64: String; Var str256: String);
Procedure ConvertBase64to2(str64: String; Var str2: String);
Procedure HDD_DATAToBase2String(HDD_DATA: THDD_DATA; Var S: String);
Procedure Base2StringToHDD_DATA(S: String; Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAToBase256String(Const HDD_DATA: THDD_DATA; Var str256:
String);
Procedure Base256StringToHDD_DATA(str256: String; Var HDD_DATA: THDD_DATA);
Procedure MPIToHDD_DATA(MPI: String; Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAToBase10String(Var HDD_DATA: THDD_DATA; Var Base10: String);
Function HDD_DATACompareAbs(Var HDD_DATA1, HDD_DATA2: THDD_DATA): TCompare;
Procedure HDD_DATAChangeSign(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAAdd(Var HDD_DATA1, HDD_DATA2, Sum: THDD_DATA);
Procedure HDD_DATASub(Var HDD_DATA1, HDD_DATA2, dif: THDD_DATA);
Procedure HDD_DATAMulByInt(Var HDD_DATA, res: THDD_DATA; by: longint);
Procedure HDD_DATAMulByIntbis(Var HDD_DATA: THDD_DATA; by: longint);
Procedure HDD_DATADivByInt(Var HDD_DATA, res: THDD_DATA; by: longint; Var
modres:
Procedure HDD_DATAAbs(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAShiftLeft(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAShiftRight(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAShiftLeftBy15(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAShiftRightBy15(Var HDD_DATA: THDD_DATA);
Procedure HDD_DATAAddBis(Var HDD_DATA1, HDD_DATA2: THDD_DATA);
Procedure HDD_DATASubBis(Var HDD_DATA1, HDD_DATA2: THDD_DATA);
Procedure HDD_DATAMul(Var HDD_DATA1, HDD_DATA2, Prod: THDD_DATA);
```

```
Procedure HDD_DATASquare(Var HDD_DATA, Square: THDD_DATA);
Procedure HDD_DATAExp(Var HDD_DATA, exp, res: THDD_DATA);
Procedure HDD_DATAFac(Var HDD_DATA, res: THDD_DATA);
Procedure HDD_DATADivMod(Var HDD_DATA1, HDD_DATA2, QHDD_DATA, MHDD_DATA:
THDD_DATA);
Procedure HDD_DATADiv(Var HDD_DATA1, HDD_DATA2, QHDD_DATA: THDD_DATA);
Procedure HDD_DATAMod(Var HDD_DATA1, HDD_DATA2, MHDD_DATA: THDD_DATA);
Procedure HDD_DATAModBis(Var HDD_DATA, HDD_DATAOut: THDD_DATA; b, head:
longint);
Procedure HDD_DATAMontgomeryModExp(Var HDD_DATA, exp, modb, res: THDD_DATA);
Procedure HDD_DATAModExp(Var HDD_DATA, exp, modb, res: THDD_DATA);
Procedure HDD_DATATrialDiv9999(Var HDD_DATA: THDD_DATA; Var ok: boolean);
Procedure HDD_DATARandom1(Var Seed, RandomHDD_DATA: THDD_DATA);
Procedure HDD_DATALegendreSymbol(Var a, p: THDD_DATA; Var L: integer);
Procedure HDD_DATASquareRootModP(Square, Prime: THDD_DATA; Var SquareRoot:
THDD_DATA);
Implementation // Реалізація
Const // Константи
chr64: Array[1..64] Of char = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z',
'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5',
'6', '7', '8', '9', '+', '/');
primes: Array[1..1228] Of integer = (3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37,
41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113,
127,
131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211,
223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307,
311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401,
409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499,
503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607,
613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677);
Var // Об’ява типів даних
trans: Array[0..255] Of String;
// Знищення об’єкту потоку
Procedure HDD_DATADestroy(Var GInt: THDD_DATA);
Begin
If GInt.Number <> Nil Then Freemem(GInt.Number);
GInt.Number:= Nil;
End;
// Копіювання
Procedure HDD_DATACopy(Var GInt, Copied: THDD_DATA);
Var // Об’ява типів даних
i: longint;
Begin
Copied.Sign:= GInt.Sign;
Getmem(Copied.Number, 4 * (GInt.Number[0] + 1));
For i:= 0 To GInt.Number[0] Do Copied.Number[i]:= GInt.Number[i];
End;
// Перетворення типів даних
Procedure zeronetochar8(Var g: char; Var x: String);
Var
i: Integer;
b: byte;
Begin
b:= 0;
For i:= 1 To 8 Do
Begin
If copy(x, i, 1) = '1' Then
```

```
 b:= b Or (1 Shl (8 - I));
End;
g:= chr(b);
End;
// Перетворення типів даних
Procedure zeronetochar6(Var g: integer; Var x: String);
Var // Об’ява типів даних
I: Integer;
Begin
G:= 0;
For I:= 1 To Length(X) Do
Begin
If I > 6 Then
 Break;
If X[I] <> '0' Then
 G:= G Or (1 Shl (6 - I));
End;
Inc(G);
End;
// Допоміжна функція
Function IntToStr1(Var b: integer): String;
Begin
If b = 0 Then IntToStr1:= '0' Else IntToStr1:= '1';
End;
Function StrToInt4(Var S: String): longint;
Var // Об’ява типів даних
i, r: integer;
Begin
r:= 0;
For i:= 1 To length(S) Do
Begin
r:= r * 10;
r:= r + (ord(S[i]) - 48);
End;
StrToInt4:= r;
End;
// Допоміжна функція
Function IntToStr4(i: longint): String;
Var // Об’ява типів даних
j: integer;
r: String;
Begin
r:= '';
For j:= 1 To 4 Do
Begin
r:= chr((i Mod 10) + 48) + r;
i:= i Div 10;
End;
IntToStr4:= r;
End;
Function min(i1, i2: longint): longint;
Begin
If i1 < i2 Then min:= i1 Else min:= i2;
End;

// Встановлення довжини пакету даних
Procedure Setlength(Var number: TDA; oldsize, newsize: longint);
```

```
Var // Об’ява типів даних
temp: TDA;
i, t: longint;
Begin
t:= min(oldsize, newsize);
getmem(temp, newsize * 4);
For i:= 0 To (t - 1) Do temp[i]:= number[i];
freemem(number);
number:= temp;
End;
// ініціалізація
Procedure initialize8(Var trans: Array Of String);
Var // Об’ява типів даних
c1, c2, c3, c4, c5, c6, c7, c8: integer;
x: String;
g: char;
Begin
For c1:= 0 To 1 Do
For c2:= 0 To 1 Do
For c3:= 0 To 1 Do
  For c4:= 0 To 1 Do
     For c5:= 0 To 1 Do
        For c6:= 0 To 1 Do
           For c7:= 0 To 1 Do
              For c8:= 0 To 1 Do
              Begin
                 x:= '';
x:=inttostr1(c1)+inttostr1(c2)+inttostr1(c3)+inttostr1(c4)+inttostr1(c5)+inttost
r1(c6)+inttostr1(c7)+inttostr1(c8);
                 zeronetochar8(g, x);
                 trans[ord(g)]:= x;
              End;
End;

Procedure initialize6(Var trans: Array Of String);
Var // Об’ява типів даних
c1, c2, c3, c4, c5, c6: integer;
x: String;
g: integer;
Begin
For c1:= 0 To 1 Do
For c2:= 0 To 1 Do
 For c3:= 0 To 1 Do
   For c4:= 0 To 1 Do
      For c5:= 0 To 1 Do
         For c6:= 0 To 1 Do
         Begin
            x:= '';
            x:= inttostr1(c1) + inttostr1(c2) + inttostr1(c3) + inttostr1(c4) +
inttostr1(c5) + inttostr1(c6);
            zeronetochar6(g, x);
            trans[ord(chr64[g])]:= x;
         End;
End;

Procedure initialize6(Var trans: Array Of String);
Var // Об’ява типів даних
```

```
c1, c2, c3, c4, c5, c6: integer;
x: String;
g: integer;
Begin
For c1:= 0 To 1 Do
For c2:= 0 To 1 Do
 For c3:= 0 To 1 Do
    For c4:= 0 To 1 Do
       For c5:= 0 To 1 Do
          For c6:= 0 To 1 Do
          Begin
      x:= '';     x:=inttostr1(c1)+inttostr1(c2)+inttostr1(c3)+
                     inttostr1(c4)+inttostr1(c5)+inttostr1(c6);
             zeronetochar6(g, x);
             trans[ord(chr64[g])]:= x;
          End;
End;

// Перетворення типів даних
Procedure ConvertBase256to64(Var str256, str64: String);
Var // Об’ява типів даних
temp, x: String;
i, len6: longint;
g: integer;
Begin
initialize8(trans);
temp:= '';
For i:= 1 To length(str256) Do temp:= temp + trans[ord(str256[i])];
While (length(temp) Mod 6) <> 0 Do temp:= temp + '0';
len6:= length(temp) Div 6;
str64:= '';
For i:= 1 To len6 Do
Begin
x:= copy(temp, 1, 6);
zeronetochar6(g, x);
str64:= str64 + chr64[g];
delete(temp, 1, 6);
End;
End;

// Перетворення типів даних
Procedure ConvertBase64to256(Var str64, str256: String);
Var // Об’ява типів даних
temp, x: String;
i, len8: longint;
g: char;
Begin
initialize6(trans);
temp:= '';
For i:= 1 To length(str64) Do temp:= temp + trans[ord(str64[i])];
str256:= '';
len8:= length(temp) Div 8;
For i:= 1 To len8 Do
Begin
x:= copy(temp, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
```

```
delete(temp, 1, 8);
End;
End;

// Перетворення типів даних
Procedure ConvertBase256to2(str256: String; Var str2: String);
Var // Об’ява типів даних
i: longint;
Begin
str2:= '';
initialize8(trans);
For i:= 1 To length(str256) Do str2:= str2 + trans[ord(str256[i])];
End;

// Перетворення типів даних
Procedure ConvertBase64to2(str64: String; Var str2: String);
Var // Об’ява типів даних
i: longint;
Begin
str2:= '';
initialize6(trans);
For i:= 1 To length(str64) Do str2:= str2 + trans[ord(str64[i])];
End;

// Перетворення типів даних
Procedure ConvertBase2to256(str2: String; Var str256: String);
Var // Об’ява типів даних
i, len8: longint;
g: char;
x: String;
Begin
str256:= '';
While (length(str2) Mod 8) <> 0 Do str2:= '0' + str2;
len8:= length(str2) Div 8;
For i:= 1 To len8 Do
Begin
x:= copy(str2, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
delete(str2, 1, 8);
End;
End;

// Перетворення типів даних
Procedure ConvertBase2to64(str2: String; Var str64: String);
Var // Об’ява типів даних
i, len6: longint;
g: integer;
x: String;
Begin
str64:= '';
While (length(str2) Mod 6) <> 0 Do str2:= '0' + str2;
len6:= length(str2) Div 6;
For i:= 1 To len6 Do
Begin
x:= copy(str2, 1, 6);
zeronetochar6(g, x);
```

```
str64:= str64 + chr64[g];
delete(str2, 1, 6);
End;
End;

// Перетворення типів даних
Procedure ConvertBase256StringToHexString(Str256: String; Var HexStr: String);
Var // Об’ява типів даних
i: longint;
b: byte;
Begin
HexStr:= '';
For i:= 1 To length(str256) Do
Begin
b:= ord(str256[i]);
If (b Shr 4) < 10 Then HexStr:= HexStr + chr(48 + (b Shr 4))
Else HexStr:= HexStr + chr(55 + (b Shr 4));
If (b And 15) < 10 Then HexStr:= HexStr + chr(48 + (b And 15))
Else HexStr:= HexStr + chr(55 + (b And 15));
End;
End;

// Перетворення типів даних
Procedure ConvertHexStringToBase256String(HexStr: String; Var Str256: String);
Var // Об’ява типів даних
i: longint;
b, h1, h2: byte;
Begin
Str256:= '';
For i:= 1 To (length(Hexstr) Div 2) Do
Begin
h2:= ord(HexStr[2 * i]);
h1:= ord(HexStr[2 * i - 1]);
If h1 < 58 Then b:= ((h1 - 48) Shl 4) Else b:= ((h1 - 55) Shl 4);
If h2 < 58 Then b:= (b Or (h2 - 48)) Else b:= (b Or (h2 - 55));
Str256:= Str256 + chr(b);
End;
End;

Procedure ConvertBase256to64(Var str256, str64: String);
Var // Об’ява типів даних
temp, x, a: String;
i, len6: longint;
g: integer;
Begin
initialize8(trans);
temp:= '';
For i:= 1 To length(str256) Do temp:= temp + trans[ord(str256[i])];
If (length(temp) Mod 6) = 0 Then a:= '' Else
If (length(temp) Mod 6) = 4 Then
Begin
 temp:= temp + '00';
 a:= '=';
End
Else
Begin
temp:= temp + '0000';
```

```
a:= '==';
End;
str64:= '';
len6:= length(temp) Div 6;
For i:= 1 To len6 Do
Begin
x:= copy(temp, 1, 6);
zeronetochar6(g, x);
str64:= str64 + chr64[g];
delete(temp, 1, 6);
End;
str64:= str64 + a;
End;

// Перетворення типів даних
Procedure ConvertBase64to256(str64: String; Var str256: String);
Var // Об’ява типів даних
temp, x: String;
i, j, len8: longint;
g: char;
Begin
initialize6(trans);
temp:= '';
str256:= '';
If str64[length(str64) - 1] = '=' Then j:= 2 Else
If str64[length(str64)] = '=' Then j:= 1 Else j:= 0;
For i:= 1 To (length(str64) - j) Do temp:= temp + trans[ord(str64[i])];
If j <> 0 Then delete(temp, length(temp) - 2 * j + 1, 2 * j);
len8:= length(temp) Div 8;
For i:= 1 To len8 Do
Begin
x:= copy(temp, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
delete(temp, 1, 8);
End;
End;

Procedure ConvertBase64to2(str64: String; Var str2: String);
Var // Об’ява типів даних
i, j: longint;
Begin
str2:= '';
initialize6(trans);
If str64[length(str64) - 1] = '=' Then j:= 2 Else
If str64[length(str64)] = '=' Then j:= 1 Else j:= 0;
For i:= 1 To (length(str64) - j) Do str2:= str2 + trans[ord(str64[i])];
delete(str2, length(str2) - 2 * j + 1, 2 * j);
End;

Procedure HDD_DATAToBase2String(HDD_DATA: THDD_DATA; Var S: String);
Var // Об’ява типів даних
i: longint;
j: integer;
Begin
S:= '';
For i:= 1 To HDD_DATA.Number[0] Do
```

```
Begin
For j:= 0 To 14 Do
 If (1 And (HDD_DATA.Number[i] Shr j)) = 1 Then S:= '1' + S Else S:= '0' + S;
End;
While (length(S) > 1) And (S[1] = '0') Do delete(S, 1, 1);
If S = '' Then S:= '0';
End;

// Допоміжна функція
Procedure Base2StringToHDD_DATA(S: String; Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
i, j, size: longint;
Begin
size:= length(S) Div 15;
If (length(S) Mod 15) <> 0 Then size:= size + 1;
Getmem(HDD_DATA.Number, (size + 1) * 4);
HDD_DATA.Number[0]:= size;
j:= 1;
HDD_DATA.Number[j]:= 0;
i:= 0;
While length(S) > 0 Do
Begin
If S[length(S)] = '1' Then
 HDD_DATA.Number[j]:= HDD_DATA.Number[j] Or (1 Shl i);
i:= i + 1;
If i = 15 Then
Begin
 i:= 0;
 j:= j + 1;
 If j <= size Then HDD_DATA.Number[j]:= 0;
End;
delete(S, length(S), 1);
End;
HDD_DATA.Sign:= positive;
End;

// Допоміжна функція
Procedure HDD_DATAToBase256String(Const HDD_DATA: THDD_DATA; Var str256:
String);
Var // Об’ява типів даних
temp1, x: String;
i, len8: longint;
g: char;
Begin
HDD_DATAToBase2String(HDD_DATA, temp1);
While (length(temp1) Mod 8) <> 0 Do temp1:= '0' + temp1;
len8:= length(temp1) Div 8;
str256:= '';
For i:= 1 To len8 Do
Begin
x:= copy(temp1, 1, 8);
zeronetochar8(g, x);
str256:= str256 + g;
delete(temp1, 1, 8);
End;
End;
```

```
// Допоміжна функція
Procedure Base256StringToHDD_DATA(str256: String; Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
temp1: String;
i: longint;
Begin
temp1:= '';
initialize8(trans);
For i:= 1 To length(str256) Do temp1:= temp1 + trans[ord(str256[i])];
While (temp1[1] = '0') And (temp1 <> '0') Do delete(temp1, 1, 1);
Base2StringToHDD_DATA(temp1, HDD_DATA);
End;

// Допоміжна функція
Procedure MPIToHDD_DATA(MPI: String; Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
temp: String;
Begin
temp:= MPI;
delete(temp, 1, 2);
Base256StringToHDD_DATA(temp, HDD_DATA);
End;

Procedure HDD_DATAToMPI(HDD_DATA: THDD_DATA; Var MPI: String);
Var // Об’ява типів даних
len, i: word;
c: char;
b: byte;
Begin
HDD_DATAToBase256String(HDD_DATA, MPI);
len:= length(MPI) * 8;
c:= MPI[1];
For i:= 7 Downto 0 Do If (ord(c) Shr i) = 0 Then len:= len - 1 Else break;
b:= len Mod 256;
MPI:= chr(b) + MPI;
b:= len Div 256;
MPI:= chr(b) + MPI;
End;
Procedure Base10StringToHDD_DATA(Base10: String; Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
i, size: longint;
j: integer;
S, x: String;
sign: TSign;
Procedure GIntDivByIntBis1(Var GInt: THDD_DATA; by: longint; Var modres:
integer);
Var // Об’ява типів даних
i, size: longint;
rest: longint;
Begin
size:= GInt.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
 modres:= modres * 10000;
 rest:= modres + GInt.Number[i];
 GInt.Number[i]:= rest Div by;
```

```
 modres:= rest Mod by;
End;
While (GInt.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> GInt.Number[0] Then
Begin
 SetLength(GInt.Number, GInt.Number[0] + 1, size + 1);
 GInt.Number[0]:= size;
End;
End;
Begin
While (Not (Base10[1] In ['-', '0'..'9'])) And (length(Base10) > 1) Do
delete(Base10, 1, 1);
If copy(Base10, 1, 1) = '-' Then
Begin
Sign:= negative;
delete(Base10, 1, 1);
End
Else Sign:= positive;
While (length(Base10) > 1) And (copy(Base10, 1, 1) = '0') Do delete(Base10, 1,
1);
size:= length(Base10) Div 4;
If (length(Base10) Mod 4) <> 0 Then size:= size + 1;
Getmem(HDD_DATA.Number, 4 * (size + 1));
HDD_DATA.Number[0]:= size;
For i:= 1 To (size - 1) Do
Begin
x:= copy(Base10, length(Base10) - 3, 4);
HDD_DATA.Number[i]:= StrToInt4(x);
delete(Base10, length(Base10) - 3, 4);
End;
HDD_DATA.Number[size]:= StrToInt4(Base10);
S:= '';
While (HDD_DATA.Number[0] <> 1) Or (HDD_DATA.Number[1] <> 0) Do
Begin
GIntDivByIntBis1(HDD_DATA, 2, j);
S:= inttostr1(j) + S;
End;
S:= '0' + S;
HDD_DATADestroy(HDD_DATA);
Base2StringToHDD_DATA(S, HDD_DATA);
HDD_DATA.Sign:= sign;
End;

Procedure HDD_DATADivByIntBis(Var HDD_DATA: THDD_DATA; by: longint; Var modres:
longint);
Var // Об’ява типів даних
i, size, rest: longint;
Begin
size:= HDD_DATA.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres Or HDD_DATA.Number[i];
HDD_DATA.Number[i]:= rest Div by;
modres:= rest Mod by;
End;
```

```
While (HDD_DATA.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> HDD_DATA.Number[0] Then
Begin
SetLength(HDD_DATA.Number, HDD_DATA.Number[0] + 1, size + 1);
HDD_DATA.Number[0]:= size;
End;
End;

Procedure HDD_DATAToBase10String(Var HDD_DATA: THDD_DATA; Var Base10: String);
Var
S: String;
j: longint;
temp: THDD_DATA;
Begin
HDD_DATACopy(HDD_DATA, temp);
Base10:= '';
While (temp.Number[0] > 1) Or (temp.Number[1] > 0) Do
Begin
HDD_DATADivByIntBis(temp, 10000, j);
S:= IntToStr4(j);
While Length(S) < 4 Do S:= '0' + S;
Base10:= S + Base10;
End;
Base10:= '0' + Base10;
While (length(Base10) > 1) And (Base10[1] = '0') Do delete(Base10, 1, 1);
If HDD_DATA.Sign = negative Then Base10:= '-' + Base10;
End;
Function HDD_DATACompareAbs(Var HDD_DATA1, HDD_DATA2: THDD_DATA): TCompare;
Var // Об’ява типів даних
size1, size2, i: longint;
Begin
HDD_DATACompareAbs:= Er;
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
If size1 > size2 Then HDD_DATACompareAbs:= Lt Else
If size1 < size2 Then HDD_DATACompareAbs:= St Else
Begin
 i:= size2;
 While (HDD_DATA1.Number[i] = HDD_DATA2.Number[i]) And (i > 1) Do i:= i - 1;
 If HDD_DATA1.Number[i] = HDD_DATA2.Number[i] Then HDD_DATACompareAbs:= Eq Else
    If HDD_DATA1.Number[i] < HDD_DATA2.Number[i] Then HDD_DATACompareAbs:= St
Else
       If HDD_DATA1.Number[i] > HDD_DATA2.Number[i] Then HDD_DATACompareAbs:=
Lt;
End;
End;

Procedure HDD_DATAChangeSign(Var HDD_DATA: THDD_DATA);
Begin
If HDD_DATA.Sign = negative Then HDD_DATA.Sign:= positive Else HDD_DATA.Sign:=
negative;
End;
// Додавання даних до потоку
Procedure HDD_DATAAdd(Var HDD_DATA1, HDD_DATA2, Sum: THDD_DATA);
Var // Об’ява типів даних
i, size1, size2, size: longint;
rest: integer;
```

```
Trest: longint;
Begin
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
If size1 < size2 Then HDD_DATAAdd(HDD_DATA2, HDD_DATA1, Sum) Else
Begin
If HDD_DATA1.Sign = HDD_DATA2.Sign Then
Begin
 Sum.Sign:= HDD_DATA1.Sign;
 Getmem(Sum.Number, (size1 + 2) * 4);
 rest:= 0;
 For i:= 1 To size2 Do
 Begin
    Trest:= HDD_DATA1.Number[i] + HDD_DATA2.Number[i] + rest;
    Sum.Number[i]:= Trest And 32767;
    rest:= Trest Shr 15;
 End;
 For i:= (size2 + 1) To size1 Do
 Begin
    Trest:= HDD_DATA1.Number[i] + rest;
    Sum.Number[i]:= Trest And 32767;
    rest:= Trest Shr 15;
 End;
 size:= size1 + 1;
 Sum.Number[0]:= size;
 Sum.Number[size]:= rest;
 While (Sum.Number[size] = 0) And (size > 1) Do size:= size - 1;
 If Sum.Number[0] <> size Then SetLength(Sum.Number, Sum.number[0] + 1, size +
1);
 Sum.Number[0]:= size;
End
Else
Begin
 If HDD_DATACompareAbs(HDD_DATA2, HDD_DATA1) = Lt Then HDD_DATAAdd(HDD_DATA2,
HDD_DATA1, Sum)
 Else
 Begin
    Getmem(Sum.Number, 4 * (size1 + 1));
    rest:= 0;
    For i:= 1 To size2 Do
    Begin
   Trest:= 32768 + HDD_DATA1.Number[i] - HDD_DATA2.Number[i]+rest;
   Sum.Number[i]:= Trest And 32767;
   If (Trest > 32767) Then rest:= 0 Else rest:= -1;
    End;
    For i:= (size2 + 1) To size1 Do
    Begin
       Trest:= 32768 + HDD_DATA1.Number[i] + rest;
       Sum.Number[i]:= Trest And 32767;
       If (Trest > 32767) Then rest:= 0 Else rest:= -1;
    End;
    size:= size1;
    While (Sum.Number[size] = 0) And (size > 1) Do size:= size - 1;
    If size <> size1 Then SetLength(Sum.Number, size1 + 1, size + 1);
    Sum.Number[0]:= size;
    Sum.Sign:= HDD_DATA1.Sign;
 End;
```

```
End;
End;
End;
// Обробка даних
Procedure HDD_DATASub(Var HDD_DATA1, HDD_DATA2, dif: THDD_DATA);
Begin
HDD_DATAChangeSign(HDD_DATA2);
HDD_DATAAdd(HDD_DATA1, HDD_DATA2, dif);
HDD_DATAChangeSign(HDD_DATA2);
End;

Procedure HDD_DATAMulByInt(Var HDD_DATA, res: THDD_DATA; by: longint);
Var // Об’ява типів даних
i, size: longint;
Trest, rest: longint;
Begin
size:= HDD_DATA.Number[0];
Getmem(res.Number, 4 * (size + 2));
rest:= 0;
For i:= 1 To size Do
Begin
Trest:= HDD_DATA.Number[i] * by + rest;
res.Number[i]:= Trest And 32767;
rest:= Trest Shr 15;
End;
If rest <> 0 Then
Begin
size:= size + 1;
Res.Number[size]:= rest;
End
Else SetLength(Res.Number, size + 2, size + 1);
Res.Number[0]:= size;
Res.Sign:= HDD_DATA.Sign;
End;
// конвертація
Procedure HDD_DATAMulByIntbis(Var HDD_DATA: THDD_DATA; by: longint);
Var // Об’ява типів даних
i, size: longint;
Trest, rest: longint;
Begin
size:= HDD_DATA.Number[0];
Setlength(HDD_DATA.Number, size + 1, size + 2);
rest:= 0;
For i:= 1 To size Do
Begin
Trest:= HDD_DATA.Number[i] * by + rest;
HDD_DATA.Number[i]:= Trest And 32767;
rest:= Trest Shr 15;
End;
If rest <> 0 Then
Begin
size:= size + 1;
HDD_DATA.Number[size]:= rest;
End
Else SetLength(HDD_DATA.Number, size + 2, size + 1);
HDD_DATA.Number[0]:= size;
End;
```

```
Procedure HDD_DATADivByInt(Var HDD_DATA, res: THDD_DATA; by: longint; Var
modres: longint);
Var // Об’ява типів даних
i, size: longint;
rest: longint;
Begin
size:= HDD_DATA.Number[0];
Getmem(res.Number, 4 * (size + 1));
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres Or HDD_DATA.Number[i];
res.Number[i]:= rest Div by;
modres:= rest Mod by;
End;
While (res.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> HDD_DATA.Number[0] Then SetLength(res.Number, HDD_DATA.Number[0] + 1,
size + 1);
res.Number[0]:= size;
Res.Sign:= HDD_DATA.Sign;
End;
Procedure HDD_DATAModByInt(Var HDD_DATA: THDD_DATA; by: longint; Var modres:
longint);
Var // Об’ява типів даних
i, size, rest: longint;
Begin
size:= HDD_DATA.Number[0];
modres:= 0;
For i:= size Downto 1 Do
Begin
modres:= modres Shl 15;
rest:= modres + HDD_DATA.Number[i];
modres:= rest Mod by;
End;
End;

Procedure HDD_DATAAbs(Var HDD_DATA: THDD_DATA);
Begin
HDD_DATA.Sign:= positive;
End;

Procedure HDD_DATAShiftLeft(Var HDD_DATA: THDD_DATA);
Var
l, m, i, size: longint;
Begin
size:= HDD_DATA.Number[0];
l:= 0;
For i:= 1 To Size Do
Begin
m:= HDD_DATA.Number[i] Shr 14;
HDD_DATA.Number[i]:= ((HDD_DATA.Number[i] Shl 1) Or l) And 32767;
l:= m;
End;
If l <> 0 Then
Begin
setlength(HDD_DATA.Number, size + 1, size + 2);
```

```
HDD_DATA.Number[size + 1]:= l;
HDD_DATA.Number[0]:= size + 1;
End;
End;

Procedure HDD_DATAShiftRight(Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
l, m, i, size: longint;
Begin
size:= HDD_DATA.Number[0];
l:= 0;
For i:= size Downto 1 Do
Begin
m:= HDD_DATA.Number[i] And 1;
HDD_DATA.Number[i]:= (HDD_DATA.Number[i] Shr 1) Or l;
l:= m Shl 14;
End;
If (HDD_DATA.Number[size] = 0) And (size > 1) Then
Begin
setlength(HDD_DATA.Number, size + 1, size);
HDD_DATA.Number[0]:= size - 1;
End;
End;

Procedure HDD_DATAShiftLeftBy15(Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
f1, f2, i, size: longint;
Begin
size:= HDD_DATA.Number[0];
SetLength(HDD_DATA.Number, size + 1, size + 2);
f1:= 0;
For i:= 1 To size Do
Begin
f2:= HDD_DATA.Number[i];
HDD_DATA.Number[i]:= f1;
f1:= f2;
End;
HDD_DATA.Number[size + 1]:= f1;
HDD_DATA.Number[0]:= size + 1;
End;

Procedure HDD_DATAShiftRightBy15(Var HDD_DATA: THDD_DATA);
Var // Об’ява типів даних
size, i: longint;
Begin
size:= HDD_DATA.Number[0];
If size > 1 Then
Begin
For i:= 1 To size - 1 Do
Begin
 HDD_DATA.Number[i]:= HDD_DATA.Number[i + 1];
End;
SetLength(HDD_DATA.Number, size + 1, Size);
HDD_DATA.Number[0]:= size - 1;
End
Else HDD_DATA.Number[1]:= 0;
End;
```

```
Procedure HDD_DATAAddBis(Var HDD_DATA1, HDD_DATA2: THDD_DATA);
Var // Об’ява типів даних
i, size1, size2, Trest: longint;
rest: integer;
Begin
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
rest:= 0;
For i:= 1 To size2 Do
Begin
Trest:= HDD_DATA1.Number[i] + HDD_DATA2.Number[i] + rest;
rest:= Trest Shr 15;
HDD_DATA1.Number[i]:= Trest And 32767;
End;
For i:= size2 + 1 To size1 Do
Begin
Trest:= HDD_DATA1.Number[i] + rest;
rest:= Trest Shr 15;
HDD_DATA1.Number[i]:= Trest And 32767;
End;
If rest <> 0 Then
Begin
SetLength(HDD_DATA1.Number, size1 + 1, size1 + 2);
HDD_DATA1.Number[0]:= size1 + 1;
HDD_DATA1.Number[size1 + 1]:= rest;
End;
End;

Procedure HDD_DATASubBis(Var HDD_DATA1, HDD_DATA2: THDD_DATA);
Var // Об’ява типів даних
i, size1, size2: longint;
rest: integer;
Trest: longint;
Begin
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
rest:= 0;
For i:= 1 To size2 Do
Begin
Trest:= 32768 + HDD_DATA1.Number[i] - HDD_DATA2.Number[i] + rest;
If (Trest > 32767) Then rest:= 0 Else rest:= -1;
HDD_DATA1.Number[i]:= Trest And 32767;
End;
For i:= size2 + 1 To size1 Do
Begin
Trest:= 32768 + HDD_DATA1.Number[i] + rest;
If (Trest > 32767) Then rest:= 0 Else rest:= -1;
HDD_DATA1.Number[i]:= Trest And 32767;
End;
i:= size1;
While (HDD_DATA1.Number[i] = 0) And (i > 1) Do i:= i - 1;
If i <> size1 Then
Begin
SetLength(HDD_DATA1.Number, size1 + 1, i + 1);
HDD_DATA1.Number[0]:= i;
End;
End;
```

```
Procedure HDD_DATAMul(Var HDD_DATA1, HDD_DATA2, Prod: THDD_DATA);
Var // Об’ява типів даних
i, j, size, size1, size2: longint;
rest, Trest: longint;
Begin
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
size:= size1 + size2;
Getmem(Prod.Number, 4 * (size + 1));
For i:= 1 To size Do Prod.Number[i]:= 0;
For i:= 1 To size2 Do
Begin
rest:= 0;
For j:= 1 To size1 Do
Begin
 Trest:= Prod.Number[j + i - 1] + HDD_DATA1.Number[j] * HDD_DATA2.Number[i] +
rest;
 Prod.Number[j + i - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
End;
Prod.Number[i + size1]:= rest;
End;
Prod.Number[0]:= size;
While (Prod.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> Prod.Number[0] Then
Begin
SetLength(Prod.Number, prod.Number[0]+1, size+1);
Prod.Number[0]:= size;
End;
If HDD_DATA1.Sign = HDD_DATA2.Sign Then Prod.Sign:= Positive Else prod.Sign:=
negative;
End;

Procedure HDD_DATASquare(Var HDD_DATA, Square: THDD_DATA);
Var // Об’ява типів даних
size, size1, i, j: longint;
rest, Trest: longint;
Begin
size1:= HDD_DATA.Number[0];
size:= 2 * size1;
Getmem(Square.Number, 4 * (size + 1));
Square.Number[0]:= size;
For i:= 1 To size Do Square.Number[i]:= 0;
For i:= 1 To size1 Do
Begin
Trest:= Square.Number[2 * i - 1] + HDD_DATA.Number[i] * HDD_DATA.Number[i];
Square.Number[2 * i - 1]:= Trest And 32767;
rest:= Trest Shr 15;
For j:=i+1 To size1 Do
Begin
 Trest:=Square.Number[i+j-1]+2*HDD_DATA.Number[i]*HDD_DATA.Number[j]+rest;
 Square.Number[i + j - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
End;
Square.Number[i + size1]:= rest;
End;
```

```
Square.Sign:= positive;
While (Square.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size <> (2 * size1) Then
Begin
SetLength(Square.Number, 2 * size1 + 1, size + 1);
Square.Number[0]:= size;
End;
End;
// Обробка даних
Procedure HDD_DATAExp(Var HDD_DATA, exp, res: THDD_DATA);
Var // Об’ява типів даних
temp2, temp3: THDD_DATA;
S: String;
i: longint;
Begin
HDD_DATAToBase2String(exp, S);
If S[length(S)] = '0' Then Base10StringToHDD_DATA('1', res) Else
HDD_DATACopy(HDD_DATA, res);
HDD_DATACopy(HDD_DATA, temp2);
If length(S) > 1 Then
For i:= (length(S) - 1) Downto 1 Do
Begin
 HDD_DATASquare(temp2, temp3);
 HDD_DATADestroy(temp2);
 HDD_DATACopy(temp3, temp2);
 HDD_DATADestroy(temp3);
 If S[i] = '1' Then
 Begin
    HDD_DATAMul(res, temp2, temp3);
    HDD_DATADestroy(res);
    HDD_DATACopy(temp3, res);
    HDD_DATADestroy(temp3);
 End;
End;
HDD_DATADestroy(temp2);
End;
Procedure HDD_DATAFac(Var HDD_DATA, res: THDD_DATA);
Var // Об’ява типів даних
one, temp, temp1: THDD_DATA;
Begin
HDD_DATACopy(HDD_DATA, temp);
Base10StringToHDD_DATA('1', res);
Base10StringToHDD_DATA('1', one);
While Not (HDD_DATACompareAbs(temp, one) = Eq) Do
Begin
HDD_DATAMul(temp, res, temp1);
HDD_DATACopy(temp1, res);
HDD_DATADestroy(temp1);
HDD_DATASubBis(temp, one);
End;
HDD_DATADestroy(one);
HDD_DATADestroy(temp);
End;
// робота з числовими даними
Procedure HDD_DATADivMod(Var HDD_DATA1, HDD_DATA2, QHDD_DATA, MHDD_DATA:
THDD_DATA);
Var // Об’ява типів даних
```

```
one, zero, temp1, temp2: THDD_DATA;
s1, s2: TSign;
i, j, s, t: longint;
Begin
s1:= HDD_DATA1.Sign;
s2:= HDD_DATA2.Sign;
HDD_DATAAbs(HDD_DATA1);
HDD_DATAAbs(HDD_DATA2);
HDD_DATACopy(HDD_DATA1, MHDD_DATA);
HDD_DATACopy(HDD_DATA2, temp1);
If HDD_DATACompareAbs(HDD_DATA1, HDD_DATA2) <> St Then
Begin
s:= HDD_DATA1.Number[0] - HDD_DATA2.Number[0];
Getmem(QHDD_DATA.Number, 4 * (s + 2));
QHDD_DATA.Number[0]:= s + 1;
For t:= 1 To s Do
Begin
 HDD_DATAShiftLeftBy15(temp1);
 QHDD_DATA.Number[t]:= 0;
End;
j:= s + 1;
QHDD_DATA.Number[j]:= 0;
While HDD_DATACompareAbs(MHDD_DATA, HDD_DATA2) <> St Do
Begin
 While HDD_DATACompareAbs(MHDD_DATA, temp1) <> St Do
 Begin
    If MHDD_DATA.Number[0] > temp1.Number[0] Then
       i:= (32768 * MHDD_DATA.Number[MHDD_DATA.Number[0]] +
MHDD_DATA.Number[MHDD_DATA.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] +
1)
    Else i:= MHDD_DATA.Number[MHDD_DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       HDD_DATACopy(temp1, temp2);
       HDD_DATAMulByIntBis(temp2, i);
       HDD_DATASubBis(MHDD_DATA, temp2);
       QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + i;
       If HDD_DATACompareAbs(MHDD_DATA, temp2) <> St Then
       Begin
          QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + i;
          HDD_DATASubBis(MHDD_DATA, temp2);
       End;
       HDD_DATADestroy(temp2);
    End Else
    Begin
       QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + 1;
       HDD_DATASubBis(MHDD_DATA, temp1);
    End;
 End;
 If MHDD_DATA.Number[0] <= temp1.Number[0] Then
    If HDD_DATACompareAbs(temp1, HDD_DATA2) <> Eq Then
    Begin
       HDD_DATAShiftRightBy15(temp1);
       j:= j - 1;
    End;
End;
```

```
End
Else Base10StringToHDD_DATA('0', QHDD_DATA);
s:= QHDD_DATA.Number[0];
While (s > 1) And (QHDD_DATA.Number[s] = 0) Do s:= s - 1;
If s < QHDD_DATA.Number[0] Then
Begin
setlength(QHDD_DATA.Number, QHDD_DATA.Number[0] + 1, s + 1);
QHDD_DATA.Number[0]:= s;
End;
QHDD_DATA.Sign:= positive;
HDD_DATADestroy(temp1);
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('1', one);
If s1 = negative Then
Begin
If HDD_DATACompareAbs(MHDD_DATA, zero) <> Eq Then
Begin
 HDD_DATAadd(QHDD_DATA, one, temp1);
 HDD_DATADestroy(QHDD_DATA);
 HDD_DATACopy(temp1, QHDD_DATA);
 HDD_DATADestroy(temp1);
 HDD_DATAsub(HDD_DATA2, MHDD_DATA, temp1);
 HDD_DATADestroy(MHDD_DATA);
 HDD_DATACopy(temp1, MHDD_DATA);
 HDD_DATADestroy(temp1);
End;
If s2 = positive Then QHDD_DATA.Sign:= negative;
End
Else QHDD_DATA.Sign:= s2;
HDD_DATADestroy(one);
HDD_DATADestroy(zero);
HDD_DATA1.Sign:= s1;
HDD_DATA2.Sign:= s2;
End;
// робота з числовими даними
Procedure HDD_DATADiv(Var HDD_DATA1, HDD_DATA2, QHDD_DATA: THDD_DATA);
Var // Об’ява типів даних
one, zero, temp1, temp2, MHDD_DATA: THDD_DATA;
s1, s2: TSign;
i, j, s, t: longint;
Begin
s1:= HDD_DATA1.Sign;
s2:= HDD_DATA2.Sign;
HDD_DATAAbs(HDD_DATA1);
HDD_DATAAbs(HDD_DATA2);
HDD_DATACopy(HDD_DATA1, MHDD_DATA);
HDD_DATACopy(HDD_DATA2, temp1);
If HDD_DATACompareAbs(HDD_DATA1, HDD_DATA2) <> St Then
Begin
s:= HDD_DATA1.Number[0] - HDD_DATA2.Number[0];
Getmem(QHDD_DATA.Number, 4 * (s + 2));
QHDD_DATA.Number[0]:= s + 1;
For t:= 1 To s Do
Begin
 HDD_DATAShiftLeftBy15(temp1);
 QHDD_DATA.Number[t]:= 0;
End;
```

```
j:= s + 1;
QHDD_DATA.Number[j]:= 0;
While HDD_DATACompareAbs(MHDD_DATA, HDD_DATA2) <> St Do
Begin
 While HDD_DATACompareAbs(MHDD_DATA, temp1) <> St Do
 Begin
    If MHDD_DATA.Number[0] > temp1.Number[0] Then
   i:= (32768 * MHDD_DATA.Number[MHDD_DATA.Number[0]] +
MHDD_DATA.Number[MHDD_DATA.Number[0] - 1])
      Div (temp1.Number[temp1.Number[0]] + 1)
    Else i:= MHDD_DATA.Number[MHDD_DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       HDD_DATACopy(temp1, temp2);
       HDD_DATAMulByIntBis(temp2, i);
       HDD_DATASubBis(MHDD_DATA, temp2);
       QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + i;
       If HDD_DATACompareAbs(MHDD_DATA, temp2) <> St Then
       Begin
          QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + i;
          HDD_DATASubBis(MHDD_DATA, temp2);
       End;
       HDD_DATADestroy(temp2);
    End Else
    Begin
       QHDD_DATA.Number[j]:= QHDD_DATA.Number[j] + 1;
       HDD_DATASubBis(MHDD_DATA, temp1);
    End;
 End;
 If MHDD_DATA.Number[0] <= temp1.Number[0] Then
    If HDD_DATACompareAbs(temp1, HDD_DATA2) <> Eq Then
    Begin
       HDD_DATAShiftRightBy15(temp1);
       j:= j - 1;
    End;
End;
End
Else Base10StringToHDD_DATA('0', QHDD_DATA);
s:= QHDD_DATA.Number[0];
While (s > 1) And (QHDD_DATA.Number[s] = 0) Do s:= s - 1;
If s < QHDD_DATA.Number[0] Then
Begin
setlength(QHDD_DATA.Number, QHDD_DATA.Number[0] + 1, s + 1);
QHDD_DATA.Number[0]:= s;
End;
QHDD_DATA.Sign:= positive;
HDD_DATADestroy(temp1);
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('1', one);
If s1 = negative Then
Begin
If HDD_DATACompareAbs(MHDD_DATA, zero) <> Eq Then
Begin
 HDD_DATAadd(QHDD_DATA, one, temp1);
 HDD_DATADestroy(QHDD_DATA);
 HDD_DATACopy(temp1, QHDD_DATA);
```

```
 HDD_DATADestroy(temp1);
 HDD_DATAsub(HDD_DATA2, MHDD_DATA, temp1);
 HDD_DATADestroy(MHDD_DATA);
 HDD_DATACopy(temp1, MHDD_DATA);
 HDD_DATADestroy(temp1);
End;
If s2 = positive Then QHDD_DATA.Sign:= negative;
End
Else QHDD_DATA.Sign:= s2;
HDD_DATADestroy(one);
HDD_DATADestroy(zero);
HDD_DATADestroy(MHDD_DATA);
HDD_DATA1.Sign:= s1;
HDD_DATA2.Sign:= s2;
End;
Procedure HDD_DATAMod(Var HDD_DATA1, HDD_DATA2, MHDD_DATA: THDD_DATA);
Var // Об’ява типів даних
one, zero, temp1, temp2: THDD_DATA;
s1, s2: TSign;
i, s, t: longint;
Begin
s1:= HDD_DATA1.Sign;
s2:= HDD_DATA2.Sign;
HDD_DATAAbs(HDD_DATA1);
HDD_DATAAbs(HDD_DATA2);
HDD_DATACopy(HDD_DATA1, MHDD_DATA);
HDD_DATACopy(HDD_DATA2, temp1);
If HDD_DATACompareAbs(HDD_DATA1, HDD_DATA2) <> St Then
Begin
s:= HDD_DATA1.Number[0] - HDD_DATA2.Number[0];
For t:= 1 To s Do HDD_DATAShiftLeftBy15(temp1);
While HDD_DATACompareAbs(MHDD_DATA, HDD_DATA2) <> St Do
Begin
 While HDD_DATACompareAbs(MHDD_DATA, temp1) <> St Do
 Begin
    If MHDD_DATA.Number[0] > temp1.Number[0] Then
    i:= (32768 * MHDD_DATA.Number[MHDD_DATA.Number[0]] +
MHDD_DATA.Number[MHDD_DATA.Number[0] - 1])
        Div (temp1.Number[temp1.Number[0]] + 1)
    Else i:= MHDD_DATA.Number[MHDD_DATA.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
    If (i <> 0) Then
    Begin
       HDD_DATACopy(temp1, temp2);
       HDD_DATAMulByIntBis(temp2, i);
       HDD_DATASubBis(MHDD_DATA, temp2);
       If HDD_DATACompareAbs(MHDD_DATA, temp2) <> St Then
HDD_DATASubBis(MHDD_DATA, temp2);
       HDD_DATADestroy(temp2);
    End Else HDD_DATASubBis(MHDD_DATA, temp1);
  If HDD_DATACompareAbs(MHDD_DATA, temp1) <> St Then
HDD_DATASubBis(MHDD_DATA,temp1);
 End;
 If MHDD_DATA.Number[0] <= temp1.Number[0] Then
    If HDD_DATACompareAbs(temp1, HDD_DATA2) <> Eq Then
HDD_DATAShiftRightBy15(temp1);
End;
End;
```

```
HDD_DATADestroy(temp1);
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('1', one);
If s1 = negative Then
Begin
If HDD_DATACompareAbs(MHDD_DATA, zero) <> Eq Then
Begin
 HDD_DATASub(HDD_DATA2, MHDD_DATA, temp1);
 HDD_DATADestroy(MHDD_DATA);
 HDD_DATACopy(temp1, MHDD_DATA);
 HDD_DATADestroy(temp1);
End;
End;
HDD_DATADestroy(one);
HDD_DATADestroy(zero);
HDD_DATA1.Sign:= s1;
HDD_DATA2.Sign:= s2;
End;

Procedure HDD_DATASquareMod(Var HDD_DATA, Modb, HDD_DATASM: THDD_DATA);
Var // Об’ява типів даних
temp: THDD_DATA;
Begin
HDD_DATASquare(HDD_DATA, temp);
HDD_DATAMod(temp, Modb, HDD_DATASM);
HDD_DATADestroy(temp);
End;

Procedure HDD_DATAAddMod(Var HDD_DATA1, HDD_DATA2, base, HDD_DATAres:
THDD_DATA);
Var // Об’ява типів даних
temp: THDD_DATA;
Begin
HDD_DATAadd(HDD_DATA1, HDD_DATA2, temp);
HDD_DATAMod(temp, base, HDD_DATAres);
HDD_DATADestroy(temp);
End;
Procedure HDD_DATAMulMod(Var HDD_DATA1, HDD_DATA2, base, HDD_DATAres:
THDD_DATA);
Var // Об’ява типів даних
temp: THDD_DATA;
Begin
HDD_DATAMul(HDD_DATA1, HDD_DATA2, temp);
HDD_DATAMod(temp, base, HDD_DATAres);
HDD_DATADestroy(temp);
End;

Procedure HDD_DATAGCD(Var HDD_DATA1, HDD_DATA2, GCD: THDD_DATA);
Var // Об’ява типів даних
k: TCompare;
temp1, temp2, temp3: THDD_DATA;
Begin
k:= HDD_DATACompareAbs(HDD_DATA1, HDD_DATA2);
If (k = Eq) Then HDD_DATACopy(HDD_DATA1, GCD) Else
If (k = St) Then HDD_DATAGCD(HDD_DATA2, HDD_DATA1, GCD) Else
Begin
 HDD_DATACopy(HDD_DATA1, temp1);
```

```
 HDD_DATACopy(HDD_DATA2, temp2);
 While (temp2.Number[0] > 1) Or (temp2.Number[1] > 0) Do
 Begin
    HDD_DATAMod(temp1, temp2, temp3);
    HDD_DATADestroy(temp1);
    HDD_DATACopy(temp2, temp1);
    HDD_DATADestroy(temp2);
    HDD_DATACopy(temp3, temp2);
    HDD_DATADestroy(temp3);
 End;
 HDD_DATACopy(temp1, GCD);
 HDD_DATADestroy(temp2);
 HDD_DATADestroy(temp1);
End; End;

Procedure HDD_DATALCM(Var HDD_DATA1, HDD_DATA2, LCM: THDD_DATA);
Var // Об’ява типів даних
temp1, temp2: THDD_DATA;
Begin
HDD_DATAGCD(HDD_DATA1, HDD_DATA2, temp1);
HDD_DATAmul(HDD_DATA1, HDD_DATA2, temp2);
HDD_DATAdiv(temp2, temp1, LCM);
HDD_DATADestroy(temp1);
HDD_DATADestroy(temp2);
End;

Procedure HDD_DATABezoutBachet(Var HDD_DATA1, HDD_DATA2, a, b: THDD_DATA);
Var // Об’ява типів даних
zero, r1, r2, r3, ta, gcd, temp, temp1, temp2: THDD_DATA;
stop: boolean;
Begin
If HDD_DATACompareAbs(HDD_DATA1, HDD_DATA2) <> St Then
Begin
HDD_DATAcopy(HDD_DATA1, r1); HDD_DATAcopy(HDD_DATA2, r2);
Base10StringToHDD_DATA('0', zero); Base10StringToHDD_DATA('1', a);
Base10StringToHDD_DATA('0', ta);
Repeat
 HDD_DATAdivmod(r1, r2, temp, r3); HDD_DATADestroy(r1);
 HDD_DATACopy(r2, r1); HDD_DATADestroy(r2);
 HDD_DATACopy(r3, r2);
 HDD_DATAmul(ta, temp, temp1);  HDD_DATAsub(a, temp1, temp2);
 HDD_DATACopy(ta, a);  HDD_DATACopy(temp2, ta);
 HDD_DATADestroy(temp1); HDD_DATADestroy(temp);
 If HDD_DATACompareAbs(r3, zero) = Eq Then stop:= true Else stop:= false;
 HDD_DATADestroy(r3);
Until stop;
HDD_DATAGCD(HDD_DATA1, HDD_DATA2, gcd); HDD_DATAmul(a, HDD_DATA1, temp1);
HDD_DATAsub(gcd, temp1, temp2); HDD_DATADestroy(temp1);
HDD_DATAdiv(temp2, HDD_DATA2, b); HDD_DATADestroy(temp2);
HDD_DATADestroy(ta); HDD_DATADestroy(r1);
HDD_DATADestroy(r2); HDD_DATADestroy(gcd);
End
Else HDD_DATABezoutBachet(HDD_DATA2, HDD_DATA1, b, a);
End;

Procedure HDD_DATAModInv(Var HDD_DATA1, base: THDD_DATA; Var Inverse:
THDD_DATA);
```

```
Var // Об’ява типів даних
zero, one, r1, r2, r3, tb, gcd, temp, temp1, temp2: THDD_DATA;
stop: boolean;
Begin
Base10StringToHDD_DATA('1', one);
HDD_DATAGCD(HDD_DATA1, base, gcd);
If HDD_DATACompareAbs(one, gcd) = Eq Then
Begin
HDD_DATAcopy(base, r1);
HDD_DATAcopy(HDD_DATA1, r2);
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('0', inverse);
Base10StringToHDD_DATA('1', tb);
Repeat
 HDD_DATAdivmod(r1, r2, temp, r3);
 HDD_DATADestroy(r1);
 HDD_DATACopy(r2, r1);
 HDD_DATADestroy(r2);
 HDD_DATACopy(r3, r2); HDD_DATAmul(tb, temp, temp1);
 HDD_DATAsub(inverse, temp1, temp2);
 HDD_DATADestroy(inverse);
 HDD_DATADestroy(temp1); HDD_DATACopy(tb, inverse);
 HDD_DATADestroy(tb); HDD_DATACopy(temp2, tb);
 HDD_DATADestroy(temp2); HDD_DATADestroy(temp);
 If HDD_DATACompareAbs(r3, zero) = Eq Then stop:= true Else stop:= false;
 HDD_DATADestroy(r3);
Until stop;
If inverse.Sign = negative Then
Begin
 HDD_DATAadd(base, inverse, temp);
 HDD_DATACopy(temp, inverse);
End;
HDD_DATADestroy(tb);
HDD_DATADestroy(r1);
HDD_DATADestroy(r2);
End;
HDD_DATADestroy(gcd);
HDD_DATADestroy(one);
End;

Procedure HDD_DATAModBis(Var HDD_DATA, HDD_DATAOut: THDD_DATA; b, head:
longint);
Var // Об’ява типів даних
i: longint;
Begin
If b <= HDD_DATA.Number[0] Then
Begin
Getmem(HDD_DATAOut.Number, 4 * (b + 1));
For i:= 0 To b Do HDD_DATAOut.Number[i]:= HDD_DATA.Number[i];
HDD_DATAOut.Number[b]:= HDD_DATAOut.Number[b] And head;
i:= b;
While (HDD_DATAOut.Number[i] = 0) And (i > 1) Do i:= i - 1;
If i < b Then SetLength(HDD_DATAOut.Number, b + 1, i + 1);
HDD_DATAOut.Number[0]:= i;
HDD_DATAOut.Sign:= positive;
End Else HDD_DATACopy(HDD_DATA, HDD_DATAOut);
End;
```

```
Procedure HDD_DATAMulModBis(Var HDD_DATA1, HDD_DATA2, Prod: THDD_DATA; b, head:
longint);
Var // Об’ява типів даних
i, j, size, size1, size2, t, rest, Trest: longint;
Begin
size1:= HDD_DATA1.Number[0];
size2:= HDD_DATA2.Number[0];
size:= min(b, size1 + size2);
Getmem(Prod.Number, 4 * (size + 1));
For i:= 1 To size Do Prod.Number[i]:= 0;
For i:= 1 To size2 Do
Begin
rest:= 0;
t:= min(size1, b - i + 1);
For j:= 1 To t Do
Begin
 Trest:= Prod.Number[j + i - 1] + HDD_DATA1.Number[j] * HDD_DATA2.Number[i] +
rest;
 Prod.Number[j + i - 1]:= Trest And 32767;
 rest:= Trest Shr 15;
End;
If (i + size1) <= b Then Prod.Number[i + size1]:= rest;
End;
Prod.Number[0]:= size;
If size = b Then Prod.Number[b]:= Prod.Number[b] And head;
While (Prod.Number[size] = 0) And (size > 1) Do size:= size - 1;
If size < Prod.Number[0] Then
Begin
SetLength(Prod.Number, prod.Number[0] + 1, size + 1);
Prod.Number[0]:= size;
End;
If HDD_DATA1.Sign = HDD_DATA2.Sign Then Prod.Sign:= Positive Else prod.Sign:=
negative;
End;

Procedure HDD_DATAMontgomeryMod(Var GInt, base, baseInv, MGInt: THDD_DATA; b,
head: longint);
Var // Об’ява типів даних
m, temp, temp1: THDD_DATA;
r, i: longint;
Begin
HDD_DATAModBis(GInt, temp, b, head);
HDD_DATAMulModBis(temp, baseInv, m, b, head);
HDD_DATAMul(m, base, temp1);
HDD_DATADestroy(temp);
HDD_DATAAdd(temp1, GInt, temp);
HDD_DATADestroy(temp1);
Getmem(MGInt.Number, 4 * (temp.Number[0] - b + 2));
For i:= 0 To temp.Number[0] - b + 1 Do MGInt.Number[i]:= temp.Number[b - 1 + i];
MGInt.Sign:= positive;
MGInt.Number[0]:= temp.Number[0] - b + 1;
HDD_DATADestroy(temp);
If (head Shr 14) = 0 Then HDD_DATADivByIntBis(MGInt, head + 1, r)
Else HDD_DATAShiftRightBy15(MGInt);
If HDD_DATACompareAbs(MGInt, base) <> St Then HDD_DATASubBis(MGInt, base);
HDD_DATADestroy(temp);
HDD_DATADestroy(m);
```

```
End;

// Допоміжна функція
Procedure HDD_DATAMontgomeryModExp(Var HDD_DATA, exp, modb, res: THDD_DATA);
Var // Об’ява типів даних
temp2, temp3, baseInv, r: THDD_DATA;
i, j, t, b: longint;
S: String;
head: longint;
Begin
HDD_DATAToBase2String(exp, S);
t:= modb.Number[0];
b:= t;
If (modb.Number[t] Shr 14) = 1 Then t:= t + 1;
Getmem(r.Number, 4 * (t + 1));
r.Number[0]:= t;
r.Sign:= positive;
For i:= 1 To t Do r.Number[i]:= 0;
If t = modb.Number[0] Then
Begin
head:= 32767;
For j:= 13 Downto 0 Do
Begin
 head:= head Shr 1;
 If (modb.Number[t] Shr j) = 1 Then
 Begin
    r.Number[t]:= 1 Shl (j + 1);
    break;
 End;
End;
End
Else
Begin
r.Number[t]:= 1;
head:= 32767;
End;
HDD_DATAModInv(modb, r, temp2);
If temp2.Sign = negative Then HDD_DATACopy(temp2, BaseInv)
Else
Begin
HDD_DATACopy(r, BaseInv);
HDD_DATASubBis(BaseInv, temp2);
End;
HDD_DATAAbs(BaseInv);
HDD_DATADestroy(temp2);
HDD_DATAMod(r, modb, res);
HDD_DATAMulMod(HDD_DATA, res, modb, temp2);
HDD_DATADestroy(r);
For i:= length(S) Downto 1 Do
Begin
If S[i] = '1' Then
Begin
 HDD_DATAmul(res, temp2, temp3);
 HDD_DATADestroy(res);
 HDD_DATAMontgomeryMod(temp3, modb, baseinv, res, b, head);
 HDD_DATADestroy(temp3);
End;
```

```
HDD_DATASquare(temp2, temp3);
HDD_DATADestroy(temp2);
HDD_DATAMontgomeryMod(temp3, modb, baseinv, temp2, b, head);
HDD_DATADestroy(temp3);
End;
HDD_DATADestroy(temp2);
HDD_DATAMontgomeryMod(res, modb, baseinv, temp3, b, head);
HDD_DATACopy(temp3, res);
HDD_DATADestroy(temp3);
HDD_DATADestroy(baseinv);
End;
// Обробка
Procedure HDD_DATAModExp(Var HDD_DATA, exp, modb, res: THDD_DATA);
Var // Об’ява типів даних
temp2, temp3: THDD_DATA;
i: longint;
S: String;
Begin
If (Modb.Number[1] Mod 2) = 1 Then
Begin
HDD_DATAMontgomeryModExp(HDD_DATA, exp, modb, res);
exit;
End;
HDD_DATAToBase2String(exp, S);
Base10StringToHDD_DATA('1', res);
HDD_DATAcopy(HDD_DATA, temp2);
For i:= length(S) Downto 1 Do
Begin
If S[i] = '1' Then
Begin
 HDD_DATAmulMod(res, temp2, modb, temp3);
 HDD_DATADestroy(res);
 HDD_DATACopy(temp3, res);
End;
HDD_DATASquareMod(temp2, Modb, temp3);
HDD_DATACopy(temp3, temp2);
End;
HDD_DATADestroy(temp2);
End;
// Допоміжна функція
Procedure HDD_DATATrialDiv9999(Var HDD_DATA: THDD_DATA; Var ok: boolean);
Var // Об’ява типів даних
j: longint;
i: integer;
Begin
If ((HDD_DATA.Number[1] Mod 2) = 0) Then ok:= false
Else
Begin
i:= 0;
ok:= true;
While ok And (i < 1228) Do
Begin
 i:= i + 1;
 HDD_DATAmodbyint(HDD_DATA, primes[i], j);
 If j = 0 Then ok:= false;
End;
End;
```

```
End;

Procedure HDD_DATARandom1(Var Seed, RandomHDD_DATA: THDD_DATA);
Var // Об’ява типів даних
temp, base: THDD_DATA;
Begin
Base10StringToHDD_DATA('281474976710656', base);
Base10StringToHDD_DATA('44485709377909', temp);
HDD_DATAMulMod(seed, temp, base, RandomHDD_DATA);
HDD_DATADestroy(temp);
HDD_DATADestroy(base);
End;

Procedure HDD_DATARabinMiller(Var HDD_DATAp: THDD_DATA; nrtest: integer; Var ok:
boolean);
Var // Об’ява типів даних
j, b, i: longint;
m, z, temp1, temp2, temp3, zero, one, two, pmin1: THDD_DATA;
ok1, ok2: boolean;
Begin
randomize;
j:= 0;
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('1', one);
Base10StringToHDD_DATA('2', two);
HDD_DATAsub(HDD_DATAp, one, temp1);
HDD_DATAsub(HDD_DATAp, one, pmin1);
b:= 0;
While (temp1.Number[1] Mod 2) = 0 Do
Begin
b:= b + 1;
HDD_DATAShiftRight(temp1);
End;
m:= temp1;
i:= 0;
ok:= true;
Randomize;
While (i < nrtest) And ok Do
Begin
i:= i + 1;
Base10StringToHDD_DATA(inttostr4(Primes[Random(1227) + 1]), temp2);
HDD_DATAMontGomeryModExp(temp2, m, HDD_DATAp, z);
HDD_DATADestroy(temp2);
ok1:= (HDD_DATACompareAbs(z, one) = Eq);
ok2:= (HDD_DATACompareAbs(z, pmin1) = Eq);
If Not (ok1 Or ok2) Then
Begin
 While (ok And (j < b)) Do
 Begin
    If (j > 0) And ok1 Then ok:= false
    Else
    Begin
       j:= j + 1;
       If (j < b) And (Not ok2) Then
       Begin
          HDD_DATASquaremod(z, HDD_DATAp, temp3);
          HDD_DATADestroy(z);
```

```
          HDD_DATACopy(temp3, z);
          ok1:= (HDD_DATACompareAbs(z, one) = Eq);
          ok2:= (HDD_DATACompareAbs(z, pmin1) = Eq);
          If ok2 Then j:= b;
       End
       Else If (Not ok2) And (j >= b) Then ok:= false;
    End;
 End;
End
End;
HDD_DATADestroy(zero);
HDD_DATADestroy(one);
HDD_DATADestroy(two);
HDD_DATADestroy(m);
HDD_DATADestroy(z);
HDD_DATADestroy(pmin1);
End;

// Тестування
Procedure HDD_DATAPrimetest(Var HDD_DATAp: THDD_DATA; nrRMtests: integer; Var
ok: boolean);
Begin
HDD_DATATrialdiv9999(HDD_DATAp, ok);
If ok Then HDD_DATARabinMiller(HDD_DATAp, nrRMtests, ok);
End;

Procedure HDD_DATALegendreSymbol(Var a, p: THDD_DATA; Var L: integer);
Var // Об’ява типів даних
temp1, temp2, temp3, temp4, temp5, zero, one: THDD_DATA;
i: longint;
ok1, ok2: boolean;
Begin
Base10StringToHDD_DATA('0', zero);
Base10StringToHDD_DATA('1', one);
HDD_DATAMod(a, p, temp1);
If HDD_DATACompareAbs(zero, temp1) = Eq Then
Begin
HDD_DATADestroy(temp1);
L:= 0;
End
Else
Begin
HDD_DATADestroy(temp1);
HDD_DATACopy(p, temp1);
HDD_DATACopy(a, temp2);
L:= 1;
While HDD_DATACompareAbs(temp2, one) <> Eq Do
Begin
 If (temp2.Number[1] Mod 2) = 0 Then
 Begin
    HDD_DATASquare(temp1, temp3);
    HDD_DATASub(temp3, one, temp4);
    HDD_DATADestroy(temp3);
    HDD_DATADivByInt(temp4, temp3, 8, i);
    If (temp3.Number[1] Mod 2) = 0 Then ok1:= false Else ok1:= true;
    HDD_DATADestroy(temp3);
    HDD_DATADestroy(temp4);
```

```
    If ok1 = true Then L:= L * (-1);
    HDD_DATADivByIntBis(temp2, 2, i);
 End
 Else
 Begin
    HDD_DATASub(temp1, one, temp3);
    HDD_DATASub(temp2, one, temp4);
    HDD_DATAMul(temp3, temp4, temp5);
    HDD_DATADestroy(temp3);
    HDD_DATADestroy(temp4);
    HDD_DATADivByInt(temp5, temp3, 4, i);
    If (temp3.Number[1] Mod 2) = 0 Then ok2:= false Else ok2:= true;
    HDD_DATADestroy(temp5);
    HDD_DATADestroy(temp3);
    If ok2 = true Then L:= L * (-1);
    HDD_DATAMod(temp1, temp2, temp3);
    HDD_DATADestroy(temp1);
    HDD_DATACopy(temp2, temp1);
    HDD_DATADestroy(temp2);
    HDD_DATACopy(temp3, temp2);
 End;
End;
HDD_DATADestroy(temp1);
HDD_DATADestroy(temp2);
End;
HDD_DATADestroy(zero);
HDD_DATADestroy(one);
End;

Procedure HDD_DATASquareRootModP(Square, Prime: THDD_DATA; Var SquareRoot:
THDD_DATA);
Var // Об’ява типів даних
one, n, b, s, r, temp, temp1, temp2, temp3: THDD_DATA;
a, i, j: longint;
L: integer;
Begin
Base2StringToHDD_DATA('1', one);
Base2StringToHDD_DATA('2', n);
a:= 0;
HDD_DATALegendreSymbol(n, Prime, L);
While L <> -1 Do
Begin
HDD_DATAAddBis(n, one);
HDD_DATALegendreSymbol(n, Prime, L);
End;
HDD_DATACopy(Prime, s);
s.Number[1]:= s.Number[1] - 1;
While (s.Number[1] Mod 2) = 0 Do
Begin
HDD_DATAShiftRight(s);
a:= a + 1;
End;
HDD_DATAMontgomeryModExp(n, s, Prime, b);
HDD_DATAAdd(s, one, temp);
HDD_DATAShiftRight(temp);
HDD_DATAMontgomeryModExp(Square, temp, Prime, r);
HDD_DATADestroy(temp);
```

```
HDD_DATAModInv(Square, Prime, temp1);
For i:= 0 To (a - 2) Do
Begin
HDD_DATASquareMod(r, Prime, temp2);
HDD_DATAMulMod(temp1, temp2, Prime, temp);
HDD_DATADestroy(temp2);
For j:= 1 To (a - i - 2) Do
Begin
 HDD_DATASquareMod(temp, Prime, temp2);
 HDD_DATADestroy(temp);
 HDD_DATACopy(temp2, temp);
 HDD_DATADestroy(temp2);
End;
If HDD_DATACompareAbs(temp, one) <> Eq Then
Begin
 HDD_DATAMulMod(r, b, Prime, temp3);
 HDD_DATADestroy(r);
 HDD_DATACopy(temp3, r);
 HDD_DATADestroy(temp3);
End;
HDD_DATADestroy(temp);
HDD_DATADestroy(temp2);
If i = (a - 2) Then break;
HDD_DATASquareMod(b, Prime, temp3);
HDD_DATADestroy(b);
HDD_DATACopy(temp3, b);
HDD_DATADestroy(temp3);
End;
// Знищення
HDD_DATACopy(r, SquareRoot);
HDD_DATADestroy(r);
HDD_DATADestroy(s);
HDD_DATADestroy(b);
HDD_DATADestroy(temp1);
HDD_DATADestroy(one);
HDD_DATADestroy(n);
End;
End.
```

ПОТОКОВИЙ АЛГОРИТМ УПРАВЛІННЯ ДАНИМИ З ЦІЛЛЮ ЗБЕРЕЖЕННЯ КОНФІДЕНЦІЙНОСТІ – STREAM.PAS

```
unit Stream;// назва модулю
{
Золотухін Богдан Євгенійович, 2023
}

interface //інтерфейс

uses // Підключаємо бібліотеки
  Windows, Sysutils;

type // Об'ява власних структур
  THDD_Data= record
    Key: array[0..255] of byte;
    OrgKey: array[0..255] of byte;
  end;


var
   s: array [0..255] of Byte;
   i,j: Byte;

implementation // реалізація

procedure InitHDD_Cipher(key: ShortString);
var
     k: array [0..255] of Byte;
     t: Byte;
     l: Cardinal;
 i0,j0: Byte;
begin
  for i0:= 0 to 255 do s[i0]:= i0;
  j0:= 1; l:= Length(key);
  for i0:= 0 to 255 do
    begin
      k[i0]:= Ord(key[j0]);
      if j0 = l then j0:= 0;
      Inc(j0);
    end;
  for i0:= 0 to 255 do
    begin
      j0:= (j0 + k[i0] + s[i0]) mod 256;
      t:= s[i0];
      s[i0]:= s[j0];
      s[j0]:= t;
    end;
  i:= 0;
  j:= 0;
end;
// Ініціалізация
procedure HDD_Init;
var
  xKey: array[0..255] of byte;
  i, j: integer;
  t: byte;
begin
  if (Len<= 0) or (Len> 256) then
    raise Exception.Create('HDD_: неправильная длинна ключа');
  for i:= 0 to 255 do
  begin
    Data.Key[i]:= i;
    xKey[i]:= PByte(integer(Key)+(i mod Len))^;
  end;
  j:= 0;
  for i:= 0 to 255 do
  begin
```

```
    j:= (j+Data.Key[i]+xKey[i]) and $FF;
    t:= Data.Key[i];
    Data.Key[i]:= Data.Key[j];
    Data.Key[j]:= t;
  end;
  Move(Data.Key,Data.OrgKey,256);
end;
// Перевірка
procedure HDD_Burn;
begin
  FillChar(Data,Sizeof(Data),$FF);
end;
// Тестування
function HDD_SelfTest;
const
  InBlock: array[0..4] of byte= ($dc,$ee,$4c,$f9,$2c);
  OutBlock: array[0..4] of byte= ($f1,$38,$29,$c9,$de);
  Key: array[0..4] of byte= ($61,$8a,$63,$d2,$fb);
var
  Block: array[0..4] of byte;
  Data: THDD_Data;
begin
  HDD_Init(Data,@Key,5);
  HDD_Crypt(Data,@InBlock,@Block,5);
  Result:= CompareMem(@Block,@OutBlock,5);
  HDD_Reset(Data);
  HDD_Crypt(Data,@Block,@Block,5);
  Result:= Result and CompareMem(@Block,@InBlock,5);
  HDD_Burn(Data);
end;

function GetHDD_ByteCiphered(bt: Byte): Byte;
var
  t: Byte;

begin
  i:= (i + 1)    mod 256;
  j:= (j + s[i]) mod 256;
  t:= s[i];
  s[i]:= s[j];
  s[j]:= t;
  t:= (s[i] + s[j]) mod 256;
  Result:= bt XOR s[t];
end;


// Застосувати до потоку даних
function ApplyHDD_ToData(Data: TStream; var Buffer: TStream; key: ShortString):
Boolean; stdcall;
var
   i: Cardinal;
   d: Byte;
 pos: Cardinal;
begin
  if (key = '')OR(Buffer = Data)OR(Buffer = nil)OR(Data =
      nil)OR(Data.Size = 0)OR(Buffer.Size <> 0) then
    begin
      Result:= false;
      Exit;
    end;
  pos:= Data.Position;
  Data.Position:= 0;
  Buffer.CopyFrom(Data,Data.Size);
  Buffer.Position:= 0;
  Data.Position:= 0;
  try
    InitHDD_Cipher(key);
    for i:= 0 to Buffer.Size-1 do
```

```
      begin
        Data.ReadBuffer(d,1);
        d:= GetHDD_ByteCiphered(d);
        Buffer.WriteBuffer(d,1);
      end;
  except
    Result:= false;
    Exit;
  end;
  Data.Position:= pos;
  Buffer.Position:= 0;
  Result:= true;
end;
End.
```