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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи Flash Drive”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Сільман Б.О.
«____» _____ 2022 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2022 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 2022 року

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

Сільману Богдану Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи Flash Drive*

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

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи Flash Drive*

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.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 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	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Сільман Б.О.
(прізвище та ініціали)

АНОТАЦІЯ

Сільман Б.О. Дослідження та програмна реалізація системи Flash Drive. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Flash Drive.

Метою розробки є дослідження та програмна реалізація системи Flash Drive.

Об'єктом дослідження є процес Flash Drive.

Предметом дослідження є методи Flash Drive.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи Flash Drive.

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 Delphi 10.4.1.

Ключові слова: комп'ютерна інженерія, захисту доступу, Flash Drive

ABSTRACT

Silman B.O. Research and software implementation of the Flash Drive system. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software designed for the Flash Drive system was developed.

The purpose of the development is research and software implementation of the Flash Drive system.

The object of study is the Flash Drive process.

The subject of research is Flash Drive methods.

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the Flash Drive system.

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

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

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

The program was developed in the Delphi 10.4.1 environment.

Keywords: computer engineering, access protection, Flash Drive

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення
AES	–	Advanced Encryption Standard
USB	–	universal serial bus

ВСТУП

Актуальність теми. Захист конфіденційної інформації сьогодні необхідний практично будь-якому власникові Flash Drive. На сьогоднішній день під засобами захисту інформації на Flash Drive розуміють сукупність різних технічних і програмних систем і пристроїв, використовуваних для рішення різних завдань по захисту інформації, у тому числі попередження витоку, захисту даних на флешці (Flash Drive) від запису й забезпечення повного комплексу мер для безпеки информации, що захищається.

Сучасні засоби захисту інформації покликані забезпечити безпеку даних на Flash Drive, як то:

– захист файлів на флешці від запису;

– захист файлів на флешці від копіювання;

– захист файлів на флешці від видалення;

– інші несанкціоновані дії.

Всі частіше покупці прагнуть придбати не просто флеш-накопичувач, а саме захищені флеш-накопичувачі, і їх прекрасно можна зрозуміти.

Захищені Flash Drive дозволили нам вступити в зовсім нову еру не просто швидкої, але й безпечної передачі й використання інформації. На флеш носіях, на даний час переноситься дуже багато конфіденційної інформації. І якщо при утраті флеш носія, ця інформація попаде у руки конкурентів то це приведе до значних економічних збитків, у кращому випадку. У гіршому випадку, наслідки можуть бути катастрофічними для підприємства або якоїсь установи.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи Flash Drive.

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

– Огляд існуючих систем Flash Drive.

– Дослідження системи Flash Drive.

– Програмна реалізація системи Flash Drive.

Об'єктом дослідження є процес Flash Drive.

Предметом дослідження є методи Flash Drive.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод Flash Drive.

– Розроблено вітчизняний продукт Flash Drive, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі Flash Drive.

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 LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи Flash Drive, 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

Програма призначена для захисту даних на Flash Drive (флешці) від запису, копіювання або іншого несанкціонованого використання. Отже програмне забезпечення, розроблене в ході виконання магістерського проектування, надасть вичерпні засоби захисту інформації, будь-яких файлів і масивів даних на флешці.

Модульність програмного забезпечення повинна дозволити запрограмувати систему, з огляду на найрізноманітніші побажання замовника.

USB-накопичувач (флешка) із захистом від запису, копіювання й несанкціонованого доступу до пристрою шляхом уведення пароля на клавіатурі дозволяє забезпечити повний захист інформаційного носія. При п'ятикратному неправильному уведенні пароля вся інформація знищується без можливості відновлення. При цьому пристрій залишається працездатним завдяки використанню унікальної технології засобів захисту інформації.

1.2 Область застосування

Областю застосування є системи захисту інформації на змінних носіях.

З необхідністю захисту інформації рано або пізно доводиться зіштовхуватися практично всім. Одна з животрепетних тем – боротьба з витоками. Все частіше й частіше зловмисники використовують змінні носії, щоб винести конфіденційну інформацію за межі корпоративного периметра. Як захиститися від цієї погрози?

Для того, щоб викрасти конфіденційну інформацію, у розпорядженні зловмисників є цілий ряд каналів передачі даних: поштові ресурси, вихід в

інтернет (веб-пошта, чат, форуми), звичайні порти робочих станцій (USB, COM, LPT), бездротові мережі (Wi-Fi, Bluetooth, IrDA) і т.д. Однак найнебезпечнішим каналом витоку вважаються комунікаційні можливості робочих станцій, до яких варто віднести стандартні порти, різні типи приводів (CD/DVD-RW, ZIP, Floppy, флеш-накопичувачі), бездротові мережі й будь-які інші засоби зняття даних з персонального комп'ютера без використання корпоративної пошти й інтернету. Стурбованість керівників саме цими каналами витоку продиктована, насамперед, зрослою популярністю мобільних накопичувачів, які протягом останнього року стали дешевше й, відповідно, одержали більше поширення.

Канали витоку даних

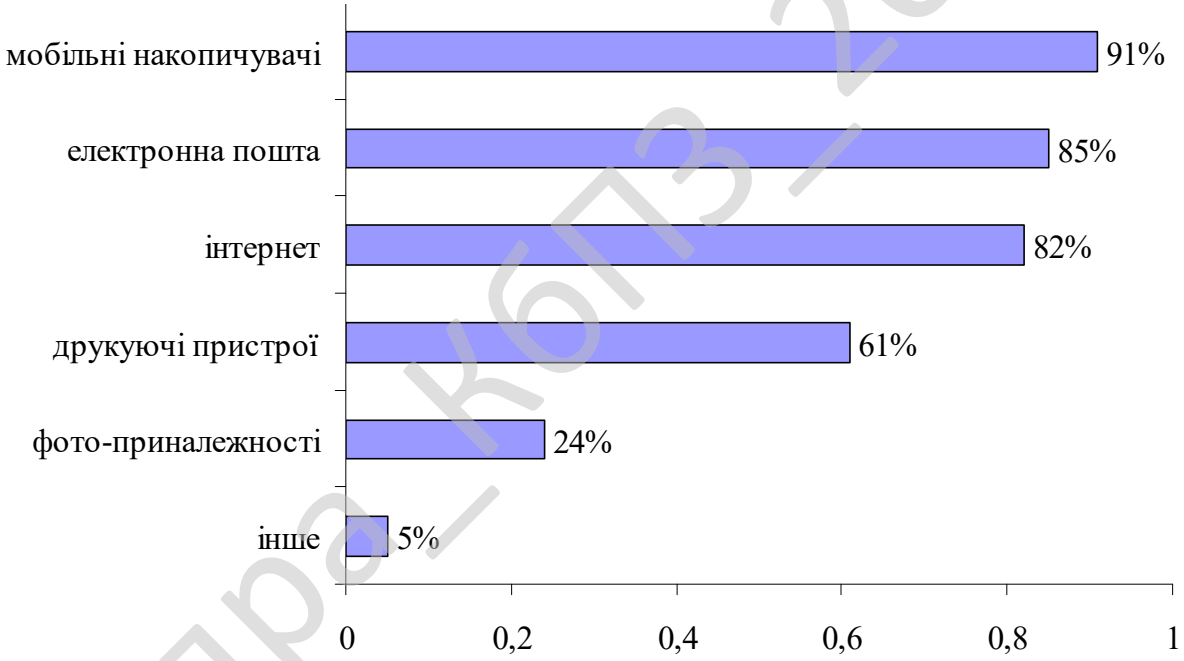

Рисунок 1.1 – Розподіл каналів витоку інформації

Слід зазначити, що чутливі відомості часто виявляються за межами мережного периметра не внаслідок навмисних дій нечистих на руку службовців, а в результаті простої безалаберності персоналу. Так, деякі співробітники воліють брати роботу додому або переписують класифіковані документи на портативний накопичувач, щоб вивчити їх на своєму ноутбуці у відрядженні. Інакше кажучи,

службовцями можуть рухати благі спонукання, які на практиці можуть привести до компрометації торгівельних або промислових секретів роботодавця.

Таким чином, саме проблема запобігання витоків на рівні робочих станцій є сьогодні однією із самих злободенних. Мінімізувати ці ризики можна або в рамках комплексного підходу, що припускає покриття всіх можливих каналів витоку, або шляхом реалізації автономного проекту, що дозволяє забезпечити контроль над обертом чутливих відомостей тільки на рівні робочих станцій.

Іншим шляхом вирішення проблеми є захист таких даних на змінних носіях інформації.

Серед усього спектра методів захисту даних від небажаного доступу особливе місце займають криптографічні методи. На відміну від інших методів, вони опираються лише на властивості самої інформації й не використовують властивості її матеріальних носіїв, особливості вузлів її обробки, передачі й зберігання.

Широке застосування комп'ютерних технологій і постійне збільшення обсягу інформаційних потоків викликає постійний ріст інтересу до криптографії. Останнім часом збільшується роль програмних засобів захисту інформації, просто модернізуємих, не потребуючих великих фінансових витрат у порівнянні з апаратними криптосистемами. Сучасні методи шифрування гарантують практично абсолютний захист даних, але завжди залишається проблема надійності їхньої реалізації.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи Flash Drive, 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

У цьому розділі розглянемо флеш пристрої в які реалізована система захисту інформації.

Silicon Power Unique 520

Втім, і в останній якості Unique 520 цікавіше багатьох флеш-накопичувачів за рахунок комплектного ПЗ. Крім засобів розширення функціональності, у нього, до речі, входить і утиліта низькорівневого форматування від виробника контролера (використовується Phison 2232), що дозволяє в деяких «важких випадках» відновити працездатність пристрою (природно, ціною втрати всіх даних).

Інші програми зібрані в типову кнопкову панельку, а додаток, її що демонструє, не вимагає інсталяції й може запускатися прямо із флешдрайва. Перша кнопка дозволяє скористатися парольним захистом.

Рисунок 2.1 – Інтерфейс парольного захисту Unique 520

Для початку варто розбити накопичувач на відкриту й захищену частини. При виконанні даної операції на першу буде поміщений і невеликий додаток для доступу до другої (хоча перемикатися можна й за допомогою «загальної» утиліти).

Потім задаємо пароль звичайним образом і все готово. Єдиний недолік цієї (уже звичної) схеми – оскільки обидві частини монтуються під одним ім'ям і просто перемикаються, одержати доступ відразу до обох неможливо.

Друга кнопка досить актуальна в наші часи тотальної відмови від флоппі-дисководів і вже почався від оптичних накопичувачів – просто робить флеш-накопичувач завантажувальним. Можна, звичайно, скористатися й сторонніми утилітами, що дає більшу гнучкість у плані вибору операційної системи, що завантажується, однак часто досить і банального MS DOS, що ми одержуємо при використанні даного додатка. Зате все дуже просто – досить лише пару раз нажати на кнопки, не забиваючи голову образами й іншим у повсякденному житті більшості користувачів непотрібним.

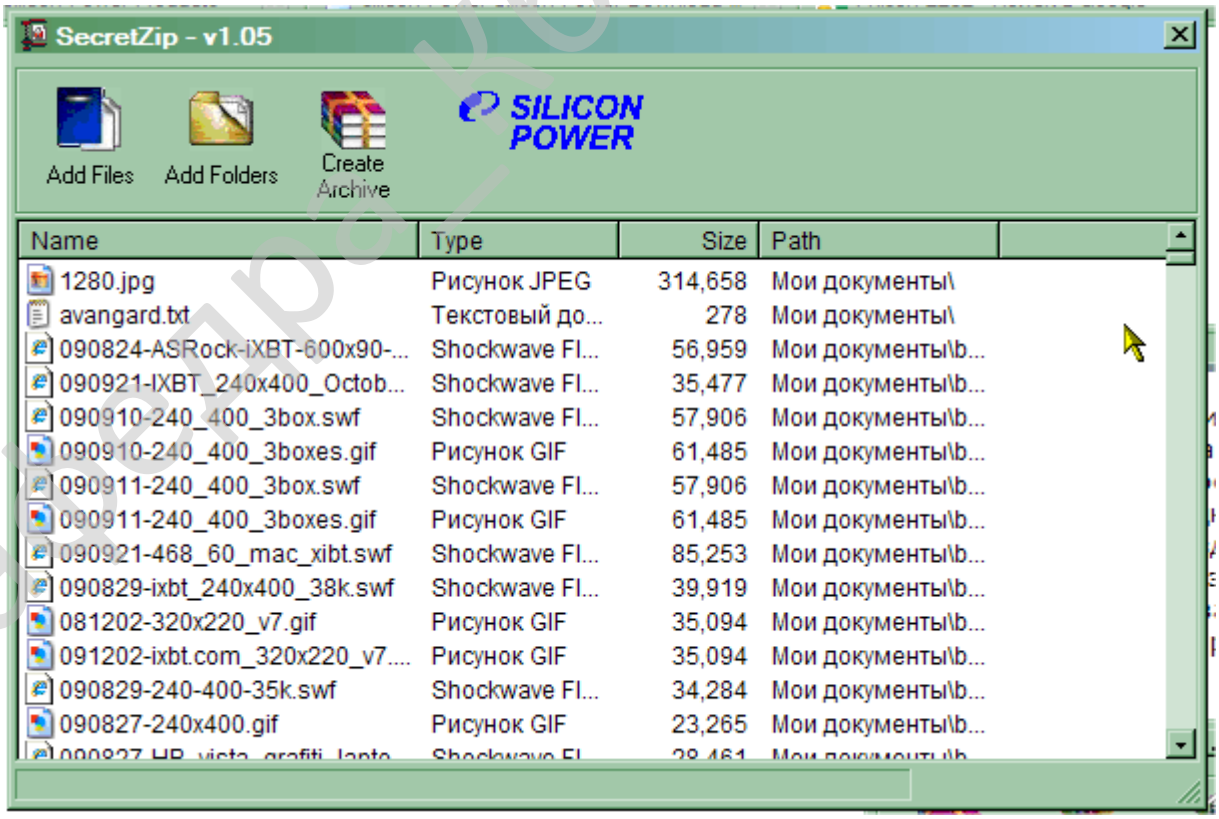

Рисунок 2.2 – Інтерфейс SecretZip

Третя кнопка забезпечує доступ до утиліти SecretZip, що дозволяє створювати на накопичувачі зашифровані архіви, а потім працювати з ними.

У порівнянні з парольним захистом (перший пункт) це більш надійно, нехай і менш зручно – губиться прозорість роботи із пристроєм. Зате це повноцінне шифрування, а не просте обмеження доступу за допомогою пароля.

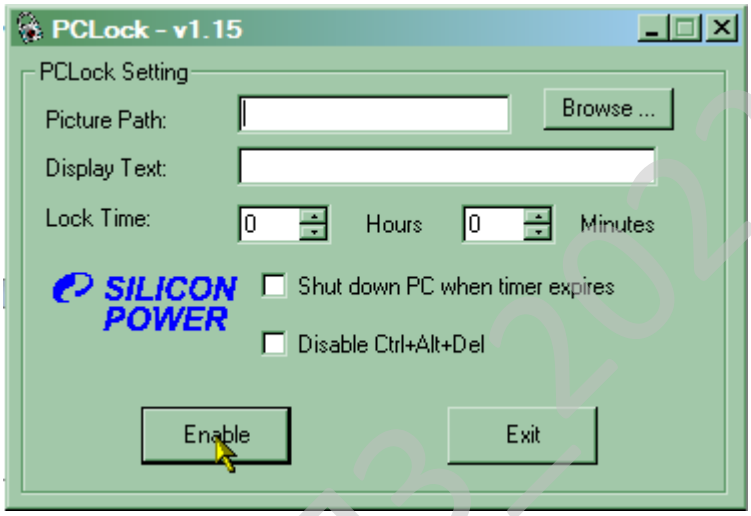

Рисунок 2.3 – Інтерфейс блокування комп'ютера за допомогою флеш-накопичувача

Ну й, нарешті, остання кнопка панелі призначена для настроювання блокування комп'ютера за допомогою флеш-накопичувача, що у цьому випадку перетворюється в ключ доступу. Дана функціональність зустрічається нам уже далеко не в перший раз, а всі її реалізації відрізняються друг від друга хіба що зовнішнім виглядом вікна настроювання, так що загострювати на ній увага не будемо.

Kingston DataTraveler Locker+

Флеш-накопичувач за назвою DataTraveler Locker+ з'явився в асортименті компанії Kingston Digital. Ключовою особливістю цього компактного й недорогого пристрою є високий ступінь захисту інформації, що перебуває усередині накопичувача, від сторонніх очей.

Дані захищені за допомогою апаратного шифрування по алгоритму AES з 256-розрядним ключем. Для одержання доступу необхідно ввести правильний пароль. Імовірність підбора пароля сильно знижена за рахунок того, що пристрій автоматично блокується й форматується після 10 невдалих спроб уведення пароля. Накопичувач сполучимо з ОС Windows 7, Vista (SP1, SP2), XP (SP1, SP2, SP3), 2000 (SP4) і Mac OS X v.10.4 – v.10.6.

Verbatim Combo SSD

Мініатюрний накопичувач Combo SSD обсягом 32 ГБ, оснащений двома інтерфейсами, з'явився в асортименті компанії Verbatim. При підключенні до порту eSATA досягається максимальна швидкість передачі даних. Наявність порту USB забезпечує накопичувачу сумісність із широким колом ПК.

По оцінці компанії, при підключенні по інтерфейсі eSATA максимальна швидкість читання дорівнює 60 МБ/с, запису -25 МБ/с. При підключенні по інтерфейсу USB 2.0 ці значення рівні 26 і 15 МБ/с відповідно.

Серед достоїнств накопичувача можна відзначити захист конфіденційної інформації, забезпечувану утилітою парольного захисту EasyLock і шифруванням по 256-бітному алгоритму AES. Це зводить до мінімуму ризик несанкціонованого доступу до даних у результаті втрати або крадіжки накопичувача.

TakeMS MEM-Drive Crypto AES з 256-бітним ключем

Компанія takeMS представила нову модель численної лінійки флеш-накопичувачів MEM-Drive. Головною особливістю новинки за назвою Crypto AES є 256-бітне апаратне шифрування. Це забезпечує надійний захист конфіденційних даних від несанкціонованого доступу.

Після обов'язкового уведення пароля, знімний носій буде разблокований, а дані розшифровані. Якщо невірний пароль буде уведений шість разів підряд, дані будуть вилучені з накопичувача без можливості відновлення.

Нова модель доступна в модифікаціях обсягом 2, 4, 8, 16 і 32 ГБ. Накопичувачі сумісні з інтерфейсами USB 2.0/1.1 і підтримують технологію Windows ReadyBoost. Заявлені швидкості читання й запису становлять 11 і 8

МБ/с відповідно. Поворотний дизайн пристроїв дозволяє власникам не турбуватися про загублені ковпачки.

Verbatim Store 'n' Go USB Executive Secure з апаратним шифруванням

Для тих, хто має потребу в захищеному мобільному сховищі даних, компанія Verbatim випустила нову лінійку флеш-накопичувачів Store 'n' Go USB Executive Secure. Безпека даних забезпечується апаратним 256-розрядним AES-Шифруванням, а також убудованим парольним захистом і алгоритмом хешування пароля. Після 20 невірних спроб уведення пароля дані на накопичувачі знищуються.

Флеш-накопичувач Verbatim TUFF-CLIP

Компанія Verbatim поповнила список портативних накопичувачів на базі флеш-пам'яті пристроєм з USB-підключенням TUFF-CLIP.

Модель TUFF-CLIP має інтерфейс USB 2.0/1.1 з висувним рознімання. Вона підтримує технологію Windows ReadyBoost і поставляється з попередньо встановленим ПЗ для парольного захисту даних. Міцний корпус із пластику ABS захищає TUFF-CLIP від механічних впливів, а карабін допоможе не втратити накопичувач.

Lavasoft Digital Lock

Рисунок 2.4 – Інтерфейс Lavasoft Digital Lock

Lavasoft Digital Lock – програма для шифрування важливої інформації й файлів, що використовує різні алгоритми (у тому числі новий стандарт AES 256). Утиліта дозволяє кодувати поштові повідомлення й файли, для перегляду яких вам не буде потрібно наявність аналогічної програми, а тільки вірний пароль. Інтерфейс Lavasoft Digital Lock простий і інтуїтивно зрозумілий.

Aladdin Secret Disk

Компанія Aladdin повідомляє про вихід нової версії системи захисту конфіденційної інформації й персональних даних Secret Disk.

Високий рівень системи захисту конфіденційної інформації й персональних даних в Secret Disk досягається шифруванням системного розділу, а також обов'язкової для всіх продуктів Aladdin процедурою строгої автентифікації користувача за допомогою апаратного USB-ключа eToken. У версії Secret Disk реалізована підтримка всього модельного ряду eToken, включаючи нову платформу eToken Java. Це дозволить виконати легку міграцію з версії «для новачків» – Secret Disk Lite – на повноцінну професійну версію Secret Disk.

Нова версія продукту дозволяє використовувати захист на основі Secret Disk для тих, хто вже працює з нової ОС Microsoft Windows.

Також розроблювачі Secret Disk доповнили функціонала продукту можливістю відновлення ключа шифрування системного розділу в eToken з копії до завантаження операційної системи. Тепер, навіть якщо ключ eToken відформатований, зламаний, загублений або заблокований, користувач однаково зможе завантажити системний диск за умови збереження копії ключа шифрування.

Стійкість використовуваного в Secret Disk PIN-кода істотно підвищена за рахунок можливості використання будь-яких мікс-комбінацій, що включають кириличні й латинські символи. Змішаний пароль створює додатковий захист від словникових атак, що в сполученні зі строгої автентифікацією дозволяє забезпечити високий рівень безпеки доступу до зашифрованих даних.

Зручний інтерфейс Secret Disk забезпечить ще більш комфортну роботу із продуктом і, завдяки оновленому меню, знизить імовірність ненавмисного блокування eToken. Крім того, нова редакція відрізняється вдосконаленою системою захисту від неправильного використання продукту, включаючи перевірку установки на локальний диск і заборона використання eToken, якщо він відформатований у режимі FIPS.

Trans-IT Edge 32 ГБ

Компанія TDK Life on Record (Imation) оголосила про надходження в продаж у Україні флеш-накопичувачів Trans-IT Edge. Лінійка представлена «флешками» у шліфованому металевому корпусі ємністю від 4 ГБ до 32 ГБ.

Завдяки технології Plug and Play, використовуваної у флеш-накопичувачах Trans-IT Edge, робота із записаною на них інформацією не вимагає попередньої установки програмного забезпечення. Несанкціонований доступ виключається за рахунок парольного захисту й розбивки на розділи.

Висновки

Як ми бачимо з аналізу сучасного стану ринку флеш накопичувачів з функцією захисту інформації, у якості алгоритму, який забезпечує цей захист виступає AES. Тому, надалі нашу увагу зосередимо саме на реалізації цього алгоритму.

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 Flash Drive.

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 алгоритму AES

Визначення й допоміжні процедури

Введемо ряд визначень і допоміжних процедур, які використовуються при реалізації алгоритму AES – Advanced Encryption Standard.

Таблиця 3.1 – Визначення

Block	послідовність біт, з яких складається input, output, State і Round Key. Також під Block можна розуміти послідовність байт
Cipher Key	секретний, криптографічний ключ, що використовується Key Expansion процедурою, щоб зробити набір ключів для раундів (Round Keys); може бути представлений як прямокутний масив байтів, що має чотири рядки й *Nk* колонок.
Ciphertext	вихідні дані алгоритму шифрування
Key Expansion	процедура використовувана для генерації Round Keys з Cipher Key
Round Key	Round Keys виходять із Cipher Key використовуючи процедуру Key Expansion. Вони застосовуються до State при шифруванні й розшифруванні
State	проміжний результат шифрування, що може бути представлений як прямокутний масив байтів що має 4 рядки й *Nb* колонок
S-box	нелінійна таблиця замін, що використовується в декількох трансформаціях заміни байт і в процедурі Key Expansion для взаємнооднозначної заміни значення байта
Nb	число стовпців(32-ух бітних слів), що становлять *State*. Для, AES *Nb* = 4
Nk	число 32-ух бітних слів, що становлять шифроключ. Для AES, *Nk* = 4,6, або 8
Nr	число раундів, що є функцією *Nk* і *Nb*. Для AES, *Nr* = 10, 12, 14
Rcon[]	масив, що складається з біт 32-х розрядного слова і є постійним для даного раунду

Таблиця 3.2 – Допоміжні процедури

AddRoundKey()	трансформація при шифруванні й зворотному шифруванні, при якуму Round Key XOR *c* State. Довжина RoundKey дорівнює розміру State (ті, якщо *Nb* = 4, то довжина RoundKey дорівнює 128 біт або 16 байт)
InvMixColumns()	трансформація при розшифруванні яка є зворотною стосовно MixColumns()
InvShiftRows()	трансформація при розшифруванні яка є зворотною стосовно ShiftRows()
InvSubBytes()	трансформація при розшифруванні яка є зворотною стосовно SubBytes()
MixColumns()	трансформація при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці
RotWord()	функція, що використовується в процедурі Key Expansion, що бере 4-х байтне слово й робить над ним циклічну перестановку
ShiftRows()	трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини
SubBytes()	трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State
SubWord()	функція, використовувана в процедурі Key Expansion, що бере на вході чотирьохбайтне слово й застосовуючи S-box до кожного із чотирьох байтів видає вихідне слово

Шифрування

AES є стандартом, заснованим на алгоритмі Rijndael. Для AES довжина input (блоку вхідних даних) і State (стану) постійна й дорівнює 128 біт, а довжина

шифроключа K становить 128, 192, або 256 біт. При цьому, вихідний алгоритм Rijndael допускає довжину ключа й розмір блоку від 128 до 256 біт із кроком в 32 біта. Для позначення обраних довжин input, State і Cipher Key у байтах використовується нотація $Nb = 4$ для input і State, $Nk = 4, 6, 8$ для Cipher Key відповідно для різних довжин ключів.

На початку шифрування input копіюється в масив State за правилом:

$$s[r,c] = in[r + 4c],$$

для $0 \le r < 4$ й $0 \le c < Nb$.

Після цього до State застосовується процедура AddRoundKey() і потім State проходить через процедуру трансформації (раунд) 10, 12, або 14 разів (залежно від довжини ключа), при цьому треба врахувати, що останній раунд трохи відрізняється від попередніх. У підсумку, після завершення останнього раунду трансформації, State копіюється в output за правилом:

$$out[r + 4c] = s[r,c],$$

для $0 \le r < 4$ й $0 \le c < Nb$.

Операція шифрування описана в псевдокоді. Окремі трансформації SubBytes(), ShiftRows(), MixColumns(), і AddRoundKey() – обробляють State. Масив w[] – містить key schedule.

Функція шифрування в псевдокоді

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey(state, w[0, Nb-1])
    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)* Nb-1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)* Nb-1])
    out = state
end
```

SubBytes()

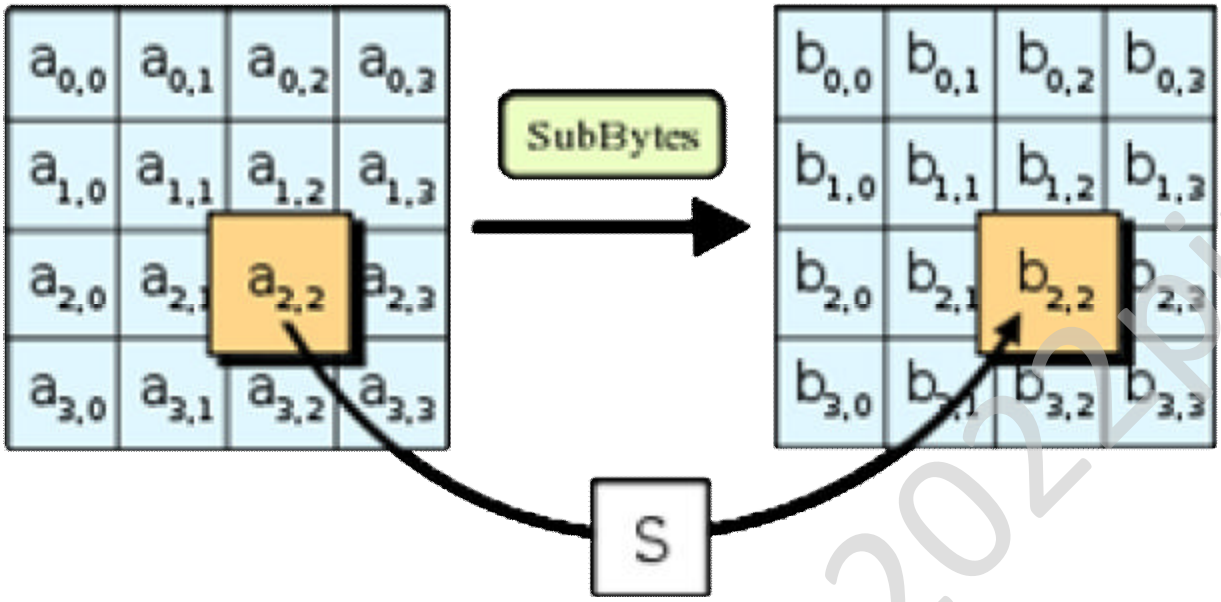

Рисунок 3.1 – Опис процедури SubBytes()

У процедурі SubBytes, кожний байт в state заміняється відповідним елементом у фіксованої 8-бітній таблиці пошуку, *S*; $b_{ij} = S(a_{ij})$.

Процедура SubBytes() обробляє кожний байт стану, незалежно роблячи нелінійну заміну байтів використовуючи таблицю замін (S-box). Така операція забезпечує нелінійність алгоритму шифрування. Побудова S-box складається із двох кроків. По-перше, виробляється узяття оберненого числа в GF{2^8}. По-друге, до кожного байта *b* з яких складається S-box застосовується наступна операція:

$$b_i' = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i,$$

де:

- $0 \leq i < 8$;
- b_i є *i-ий* біт *b*,
- c_i – *i-ий* біт *c* = {63} або {01100011}.

Таким чином, забезпечується захист від атак, заснованих на простих алгебраїчних властивостях.

$$\begin{Vmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{Vmatrix} = \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} * \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} + \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix}$$

ShiftRows()

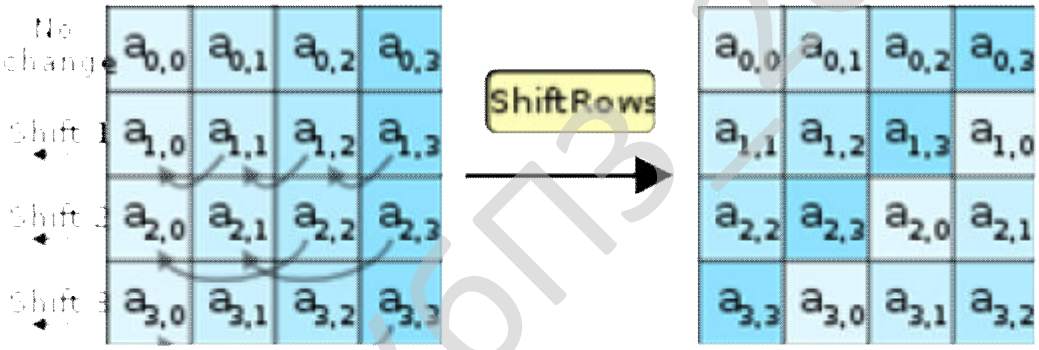

Рисунок 3.2 – Опис процедури ShiftRows()

У процедурі ShiftRows, байти в кожному рядку state циклічно зрушуються вліво. Розмір зсуву байтів кожного рядка залежить від її номера.

ShiftRows працює з рядками State. При цій трансформації рядка стани циклічно зрушуються на *r* байт по горизонталі, залежно від номера рядка. Для нульового рядка $r = 0$, для першого рядка $r = 1$ і т.д.. У такий спосіб кожна колонка вихідного стану після застосування процедури ShiftRows складається з байтів з кожної колонки початкового стану. Для алгоритму Rijndael паттерн зсуву рядків для 128 і 192-х бітних рядків однаковий. Однак для блоку розміром 256 біт відрізняється від попередніх тим, що 2, 3, і 4-і рядки зміщаються на 1, 3, і 4 байти відповідно.

MixColumns()

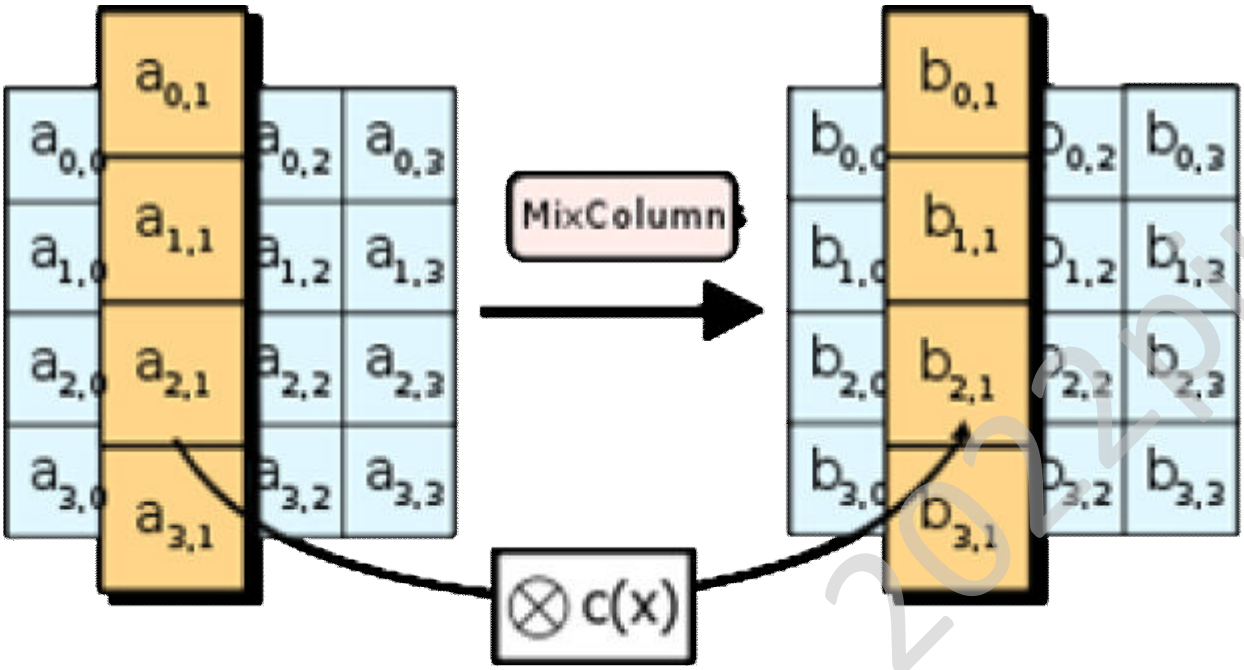

Рисунок 3.3 – Опис процедури MixColumns()

У процедурі MixColumns, кожна колонка стану перемножується з фіксованим багаточленом *c(x)*.

У процедурі MixColumns, чотири байти кожної колонки State змішуються використовуючи для цього оборотну лінійну трансформацію. MixColumns обробляє стану по колонках, трактуючи кожну з них як поліном четвертого ступеня. Над цими поліномами виробляється множення в $GF(28)$ по модулі $x^4 + 1$ на фіксований багаточлен $c(x) = 3x^3 + x^2 + x + 2$. Разом з ShiftRows, MixColumns вносить дифузію в шифр.

AddRoundKey()

У процедурі AddRoundKey, кожний байт стану поєднується з RoundKey використовуючи XOR operation (⊕). У процедурі AddRoundKey, RoundKey кожного раунду поєднується з State. Для кожного раунду Roundkey виходить із CipherKey використовуючи процедуру KeyExpansion; кожний RoundKey такого ж

розміру, що й State. Процедура робить побітовий XOR кожного байта State з кожним байтом RoundKey .

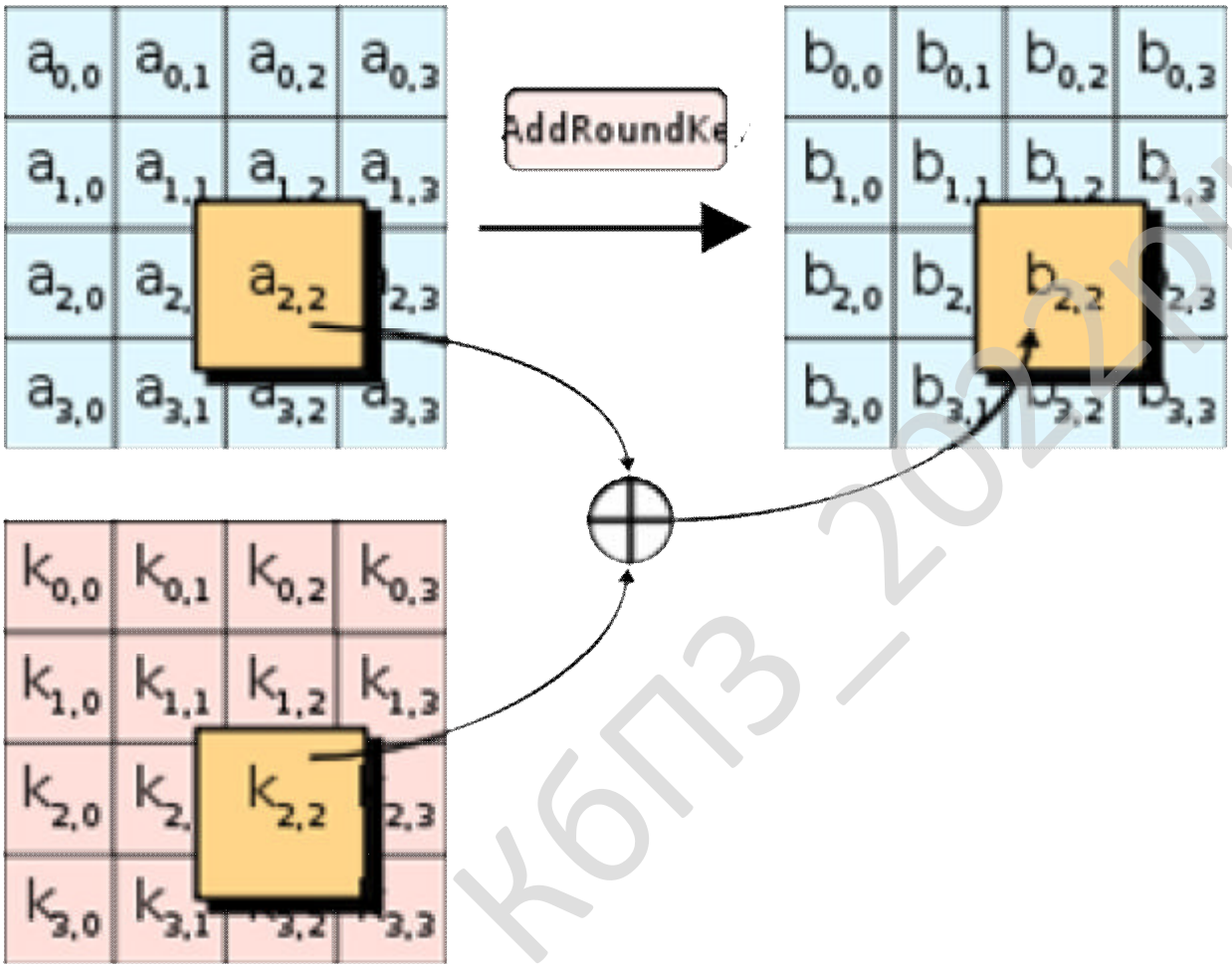

Рисунок 3.4 – Опис процедури AddRoundKey()

KeyExpansion()

AES алгоритм, використовуючи процедуру KeyExpansion() і подаючи в неї Cipher Key, K, одержує ключі для всіх раундів. Усього він одержує *Nb**(*Nr* + 1) слів: споконвічно для алгоритму потрібен набір з *Nb* слів, і кожному з *Nr* раундів потрібно *Nb* ключових набору даних. Отриманий масив ключів для раундів позначається як *w*[*i*], $0 \leq i < Nb * (Nr + 1)$.

Функція SubWord() бере чотирьохбайтове вхідне слово й застосовує S-box до кожного із чотирьох байтів те, що вийшло подається на вихід. На вхід

RotWord() подається слово $[a_0,a_1,a_2,a_3]$ яке вона циклічно переставляє й повертає $[a_1,a_2,a_3,a_0]$. Масив слів, слів постійний для даного раунду, Rcon[i], містить значення $[x^{i-1},00,00,00]$, де $x = \{02\}$, а x^{i-1} є ступенем x в GF$\{2^8\}$ (i починається з 1).

З рисунка можна побачити, що перші *Nk* слів розширеного ключа заповнені Cipher Key. У кожне наступне слово, *w*[*i*], кладе значення отримане при операції XOR *w*[*i-1*] і *w*[*i – Nk*], ті XOR'а попереднього й на *Nk* позицій раніше слів. Для слів, позиція яких кратна *Nk*, перед XOR'ом до *w*[*i-1*] застосовується трасформація, за якої треба XOR з константою раунду Rcon[*i*]. Зазначена вище трансформація складається із циклічного зрушення байтів у слові(RotWord()), за якої слідує процедура SubWord() – те ж саме, що й SubBytes(), тільки вхідні й вхідні дані будуть розміром у слово.

Важливо помітити, що процедура KeyExpansion() для 256 бітного Cipher Key небагато відрізняється від тих, які застосовуються для 128 і 192 бітних шифроключів. Якщо *Nk* = 8 і *i* – 4 кратно *Nk*, то SubWord() застосовується до *w*[*i-1*] до XOR'а.

Процедура одержання ключів для всіх раундів KeyExpansion() у псевдокоді

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp
    i = 0;
    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb * (Nr+1))
        temp = w[ i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
```

```
        w[i] = w[ i-Nk] xor temp
        i = i + 1
    end while
end
```

Функція розшифрування в псевдокоді

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]
    state = in
    AddRoundKey(state, w[Nr*Nb, (Nr+1)* Nb-1])
    for round = Nr-1 step -1 downto 1
        InvShiftRows(state)
        InvSubBytes(state)
        InvAddRoundKey(state, w[round*Nb, (round+1)* Nb-1])
        InvMixColumns(state)
    end for
    InvShiftRows(state)
    InvSubBytes(state)
    InvAddRoundKey(state, w[Nr*Nb, (Nr+1)* Nb-1])
    out = state
end
```

Варіанти алгоритму

На базі алгоритму Rijndael, що лежить в основі AES, реалізовані альтернативні криптоалгоритми. Серед найбільш відомих – учасники конкурсу Nessie: Anubis на інволюціях, автором якого є Вінсент Реймен і посилений варіант шифру – Grand Cru Йохана Борста.

3.2 Розробка структурної схеми

Структурна схема наведена на рисунку 3.5. З неї ми бачимо, що розроблена система складається з наступних структурних блоків.

1. Дані, які записуються на флешку.
2. Блок шифрування за допомогою алгоритму AES.
3. Блок розшифрування за допомогою алгоритму AES.
4. Флешка на яку записані зашифровані дані.

Основним блок системи є блок шифрування AES. Розглянемо його більш детальніше.

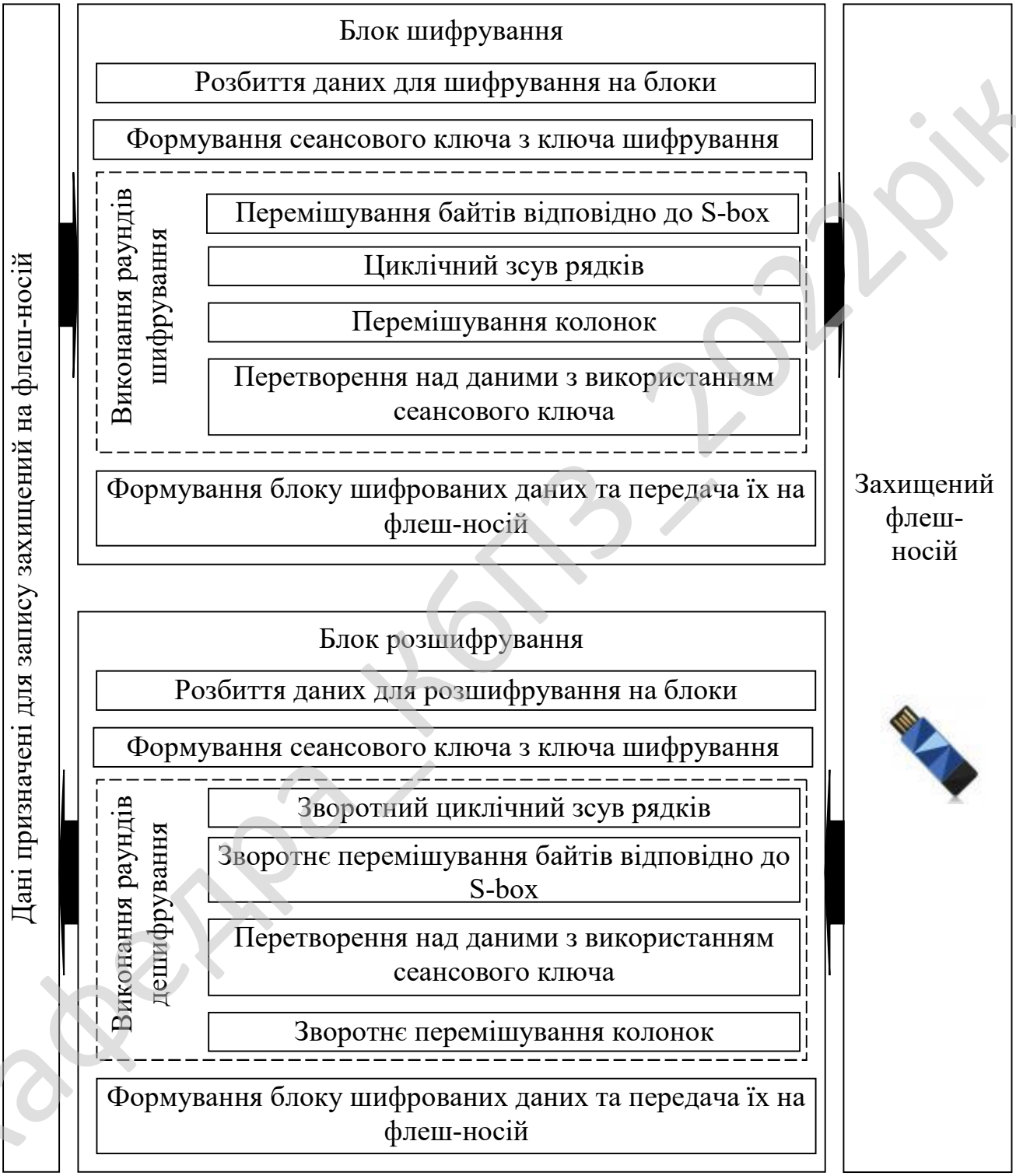

Рисунок 3.5 – Структурна схема системи

Алгоритм шифрування AES працює наступним чином:

1. Дані для шифрування input, розбивається на блоки та копіюються до установочного масиву State, згідно визначеного правилу.

2. Формується сеансовий ключ Round Key з ключа шифрування Cipher Key, за допомогою функції KeyExpansion().

3. Визначається число раундів в залежності від довжини ключа 10, 12, або 14 разів.

4. Виконання операції шифрування, тобто виконання раундів шифрування визначену в пункті 3 кількість раз:

– застосування SubBytes(), тобто трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State;

– застосування ShiftRows(), тобто трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини;

– застосування MixColumns(), тобто трансформація при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці;

– застосування AddRoundKey(), тобто трансформація при шифруванні, при якому Round Key XOR *c* State. Довжина RoundKey дорівнює розміру State (ті, якщо *Nb* = 4, то довжина RoundKey дорівнює 128 біт або 16 байт).

5. Формування блоку зашифрованих даних, для цього після завершення останнього раунду трансформації, State копіюється в output за визначеним правилом.

Алгоритм розшифрування AES працює наступним чином:

1. Дані для розшифрування input, розбивається на блоки та копіюються до установочного масиву State, згідно визначеного правилу.

2. Формується сеансовий ключ Round Key з ключа шифрування Cipher Key, за допомогою функції KeyExpansion().

3. Визначається число раундів в залежності від довжини ключа 10, 12, або 14 разів.

4. Виконання операції розшифрування, тобто виконання раундів шифрування визначену в пункті 3 кількість раз:

– застосування InvShiftRows(), яке призначене для трансформації при розшифруванні яка є зворотною стосовно ShiftRows(), тобто трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини;

– застосування InvSubBytes(), яке призначене для трансформації при розшифруванні яка є зворотною стосовно SubBytes(), тобто трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State;

– застосування InvAddRoundKey(), тобто трансформація при зворотному шифруванні, при якому Round Key XOR з State. Довжина RoundKey дорівнює розміру State (ті, якщо $Nb = 4$, то довжина RoundKey дорівнює 128 біт або 16 байт).

– застосування InvMixColumns(), яке призначене для трансформації при розшифруванні яка є зворотною стосовно MixColumns(), тобто трансформації при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці.

5. Формування блоку роз зашифрованих даних, для цього після завершення останнього раунду трансформації, State копіюється в output за визначеним правилом.

3.3 Розробка функціональної схеми

На рисунку 3.6 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

1. Головне вікно програми.
2. Блок розбиття флешки на незашифровану частину, та зашифровану частину.
3. Блок зчитування та перевірки на легітимність паролю.
4. Блок підрахунку спроб введення некоректного паролю.
5. Блок шифрування даних.
6. Блок дешифрування даних.
7. Блок гарантованого знищення інформації.
8. Блок блокування комп'ютера за допомогою флешки.
9. Блок допомоги та інформації про програму.

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

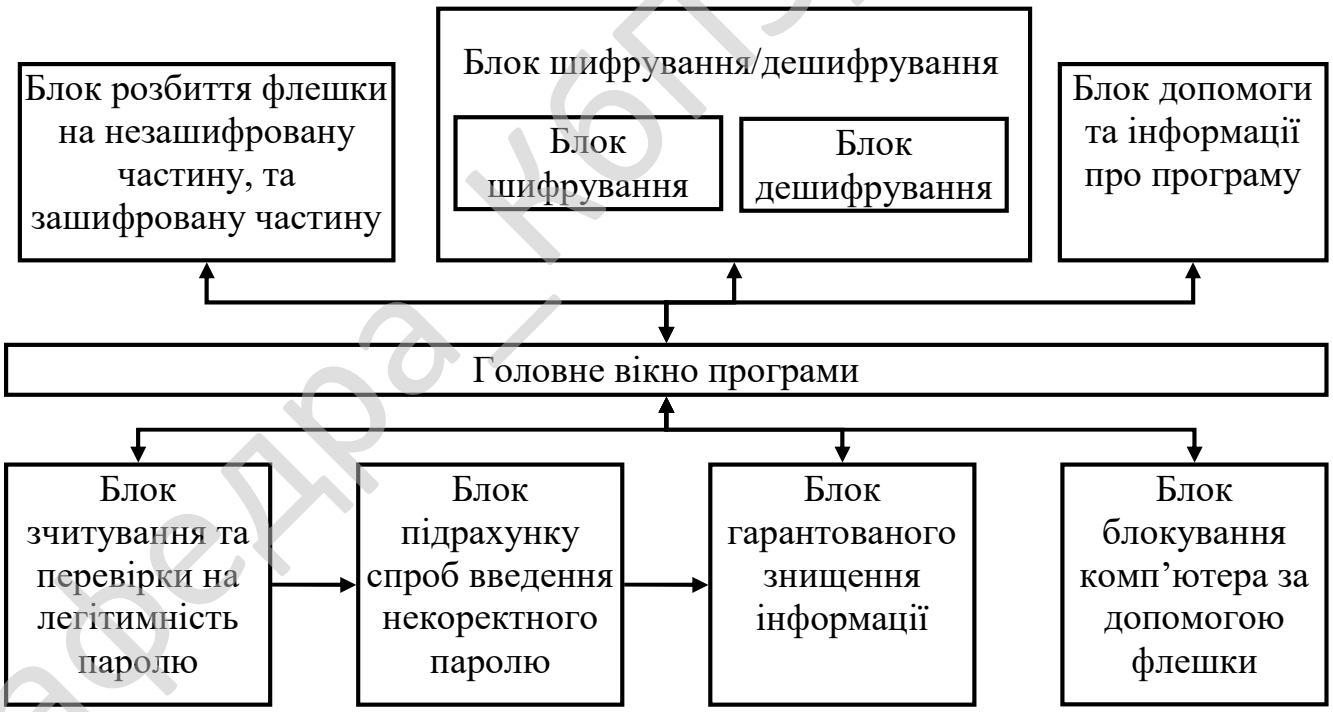

Рисунок 3.6 – Функціональна схема програмного забезпечення

Розглянемо більш детально функціональні блоки програмного забезпечення.

Головне вікно програми

Головне вікно додатка призначене для доступу до усіх функцій програми й містить в собі:

- назву програмного модуля;
- рядок головного меню;
- панель інструментів;
- робочу область;
- статусний рядок стану.

Блок розбиття флешки на незашифровану частину, та зашифровану частину

Дозволяє розбити накопичувач на відкриту й захищену частини. При виконанні даної операції на першу буде поміщений і невеликий додаток для доступу до другої (хоча перемикатися можна й за допомогою «загальної» утиліти).

Потім задаємо пароль звичайним чином і все готово. Єдиний недолік цієї схеми – оскільки обидві частини монтуються під одним ім'ям і просто перемикаються, одержати доступ відразу до обох неможливо.

Блок зчитування та перевірки на легітимність паролю

Блок зчитування та перевірки на легітимність паролю дозволяє зчитати пароль та порівняти його з тим, який збережений у програмі. Також є можливість заміни паролю, засобом введення старого паролю, та нового паролю з підтвердженням.

Блок підрахунку спроб введення некоректного паролю

Призначення цього блоку заключається у тому, що пристрій автоматично блокується й гарантовано видаляється інформація після 10 невдалих спроб уведення пароля.

Блок шифрування даних

Цей блок шифрує дані використовуючи алгоритм AES. Детальна робота цього алгоритму розписана у пунктах 3.1, та 3.2.

Блок дешифрування даних

Цей блок розшифрує дані використовуючи алгоритм AES. Детальна робота цього алгоритму розписана у пунктах 3.1, та 3.2.

Блок гарантованого знищення інформації

Цей блок призначений для гарантованого знищення інформації, при неправильному введені паролю. З основу був вибраний алгоритм Гутмана, виходячи з наступних міркувань.

Всі програмні реалізації алгоритмів знищення інформації засновані на найпростіших операціях запису, тим самим відбувається багаторазовий перезапис інформації в секторах диска помилковими даними. Залежно від алгоритму це може бути випадкове число генератора псевдовипадкових чисел або фіксоване значення. Як правило, кожний алгоритм передбачає запис восьми бітових одиниць (#FF) і нуля (#00). В існуючих алгоритмах перезапис може виробляється від одного до 35 і більше раз. Існують реалізації з можливістю довільного вибору числа циклів перезапису.

Теоретично, найпростішим методом знищенні вихідного файлу є його повний перезапис байтом #FF, тобто бітовою маскою з восьми логічних одиниць (11111111), нулів або довільних чисел, тим самим виключивши його програмне відновлення стандартними засобами, доступними користувачеві. Однак з використанням спеціалізованих апаратних засобів, що аналізують поверхню магнітних носіїв і дозволяють відновити вихідну інформацію виходячи з показників залишкової намагніченості, існує ймовірність, що найпростіший перезапис не гарантує повноцінне знищення.

З метою виключення можливості відновлення й розроблені існуючі алгоритми знищення інформації:

– Найбільш відомий і розповсюджений алгоритм, застосовуваний в американському національному стандарті Міністерства оборони Do 5220.22-M. Варіант E відповідно до даного стандарту передбачає два цикли запису

псевдовипадкових чисел і один – фіксованих значень, залежних від значень першого циклу, четвертий цикл – верифікація записів. У варіанті ECE перезапис даних виробляється 7 разів – 3 рази байтом #FF, три #00 і один #F6.

– В алгоритмі Брюса Шнайра в першому циклі записується #FF, у другому – #00 і в п'яти циклах – псевдовипадкові числа. Уважається одним з найбільш ефективних.

– У найбільш повільному, але, на думку безлічі експертів, найбільш ефективному алгоритмі Питера Гутмана, існує 35 циклів, у яких записують усе найбільш ефективні бітові маски, даний алгоритм заснований на його теорії знищення інформації.

Таблиця 3.3 – Алгоритм Гутмана

Цикл	Дані	Цикл	Дані
1	Псевдовипадкові	19	#99
2	Псевдовипадкові	20	#AA
3	Псевдовипадкові	21	#BB
4	Псевдовипадкові	22	#CC
5	#55	23	#DD
6	#AA	24	#EE
7	#92 #49 #24	25	#FF
8	#49 #24 #92	26	#92 #49 #24
9	#24 #92 #49	27	#49 #24 #92
10	#00	28	#24 #92 #49
11	#11	29	#6D #B6 #DB
12	#22	30	#B6 #DB #6D
13	#33	31	#DB #6D #B6
14	#44	32	Псевдовипадкові
15	#55	33	Псевдовипадкові
16	#66	34	Псевдовипадкові
17	#77	35	Псевдовипадкові
18	#88		

– В алгоритмі, передбаченого американським національним стандартом NAVSO P-5239-26 для MFM-кодуємих пристроїв у першому циклі записується #01, у другому – #7FFFFF, у третьому – послідовність псевдовипадкових чисел, у четвертому проходить верифікація. У варіанті для RLL – кодуємих пристроїв даного алгоритму в другому циклі записується #27FFFFF.

– В алгоритмі, що описує німецький національний стандарт VSITR з першого по шостий цикл записуються послідовно байти #00 і #FF, у сьомому #AA.

– Існує думка про існування алгоритму, що передбачає запис #00 у кожний байт кожного сектора для систем з 4-6 класи захисту й запис псевдовипадкових чисел у кожний байт кожного сектора для систем 1-3 класу захисту. Однак даний стандарт містить лише формулювання «Очищення повинне вироблятися шляхом запису інформації, що маскує, до пам’яті при її звільненні перерозподілі», що не містить якої-небудь деталізації щодо порядку перезапису, кількості циклів і бітових масок. У той же час, існує діючий Керівний документ виданий в 1992 році й ряд, що передбачає, вимог до механізму знищення інформації для систем певних класів захищеності. Зокрема, для класів 3А и 2А «Очищення здійснюється дворазовим довільним записом у область пам'яті, що звільняється, раніше використану для зберігання захищаємих даних (файлів)», для класу 1Г передбачений однократний перезапис.

– В алгоритмі Парагона перший цикл полягає в перезаписі унікальними 512-бітними блоками, використовуючи криптографічно безпечний генератор випадкових чисел, потім, у другому циклі кожний перезаписуваний блок переписується своїм двійковим доповненням, третій цикл повторює перший цикл із новими унікальними випадковими блокам, у четвертому циклі відбувається перезапис байтом #AA. Завершується знищення інформації циклом верифікації.

Як правило, для утруднення програмного відновлення інформації, перезапис інформації в окремому файлі відповідно до алгоритму знищення супроводжується установкою нульового розміру файлу і його перейменуванням,

використовуючи довільний набір символів. Потім слідує видалення файлу з таблиці розміщення файлів.

Блок блокування комп'ютера за допомогою флешки

Цей блок дозволяє блокувати комп'ютер при необхідності й надавати доступ до даних, які зберігаються на ЕОМ, тільки при підключеній флешці.

Блок допомоги та інформації про програму

У цьому блоці знаходиться допомога по використанню програми, та інформацію про розробника, версію, та дату випуску програмного продукту.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.7.

З діаграми бачимо, що першим процесом, який запускається в системі, природно, є процес початку/кінця роботи програмного забезпечення.

Цей процес взаємодіє з наступними процесами:

– Процес підключення флеш-носія;

– Процес роботи з зашифрованою частиною флеш-носія.

Процес підключення флеш-носія, у свою чергу, взаємодіє з наступними процесами:

– Процес блокування комп'ютера за допомогою флешки;

– Процес розбиття флеш-носія на незашифровану й зашифровану частину.

Останній процес взаємодіє при розбитті на зашифровану частину, з процесом роботи з зашифрованою частиною флеш-носія.

При розбитті на незашифровану частину флешки, з процесом створення ключа шифрування.

Процес створення ключа шифрування взаємодіє з процесом формування сеансового ключа з ключа шифрування.

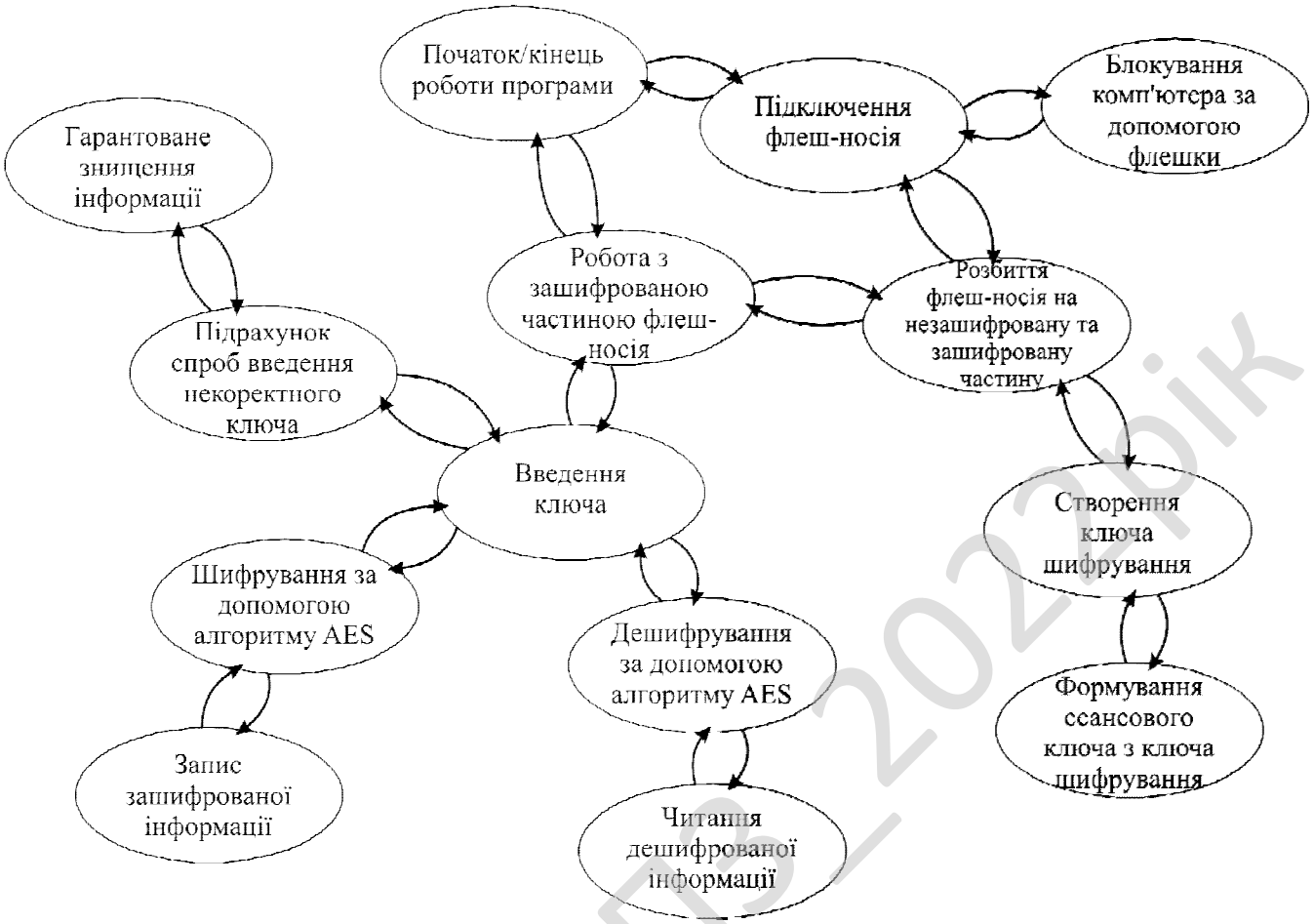

Рисунок 3.7 – Діаграма взаємодії процесів програмного забезпечення

Процес роботи з зашифрованою частиною флеш-носія взаємодіє з процесом введення ключа.

Цей процес взаємодіє з наступними процесами:

– Процес дешифрування за допомогою алгоритму AES;

– Процес шифрування за допомогою алгоритму AES;

– Процес підрахунку спроб введення некоректного ключа.

Процес дешифрування за допомогою алгоритму AES взаємодіє з процесом читання дешифрованої інформації.

Процес шифрування за допомогою алгоритму AES взаємодіє з процесом запису зашифрованої інформації.

Процес підрахунку спроб введення некоректного ключа взаємодіє з процесом гарантованого знищення інформації при кількості спроб, яке

перевищить задану у програмному продукті, кількість спроб знищення інформації.

Таким чином, розглянувши структурну та функціональну схему, а також діаграму взаємодії процесів, перейдемо до наступного розділу.

У цьому розділі опишемо блок-схеми основної програми та підпрограми шифрування/дешифрування інформації, яка знаходиться на флешці, за допомогою алгоритму AES. Також розглянемо алгоритми функціонування розробленої, у результаті виконання магістерського проектування, програми та відповідних підпрограм.

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 РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми.

Після цього відбувається пошук підключених флеш-дисків.

Коли закінчується пошук підключених флеш-дисків, то вони виводяться на екран.

Користувач вибирає той, флеш-диск, з яким він буде працювати.

Якщо треба створити зашифрований розділ, то відбувається створення окремого розділу для конфіденційної інформації.

У іншому випадку, користувач переходить до вибору процедури шифрування даних, які зберігаються на флеш-диску.

Якщо потрібно шифрувати, то користувач виконує наступні дії:

– Задає пароль доступу до зашифрованого розділу.

– Шифрує розділ алгоритмом AES.

Іншою дією, яку може проводити користувач, є процедура дешифрування інформації, яка зберігається на флеш-диску.

Для цього він реалізує наступні дії:

– Вводить пароль доступу до зашифрованого розділу.

– Дешифрує розділ, використовуючи алгоритм AES.

Якщо ж користувач обирає функцію блокування комп’ютера, то відбуваються наступні дії:

– Задається пароль доступу до комп’ютеру.

– Відбувається блокування комп’ютера.

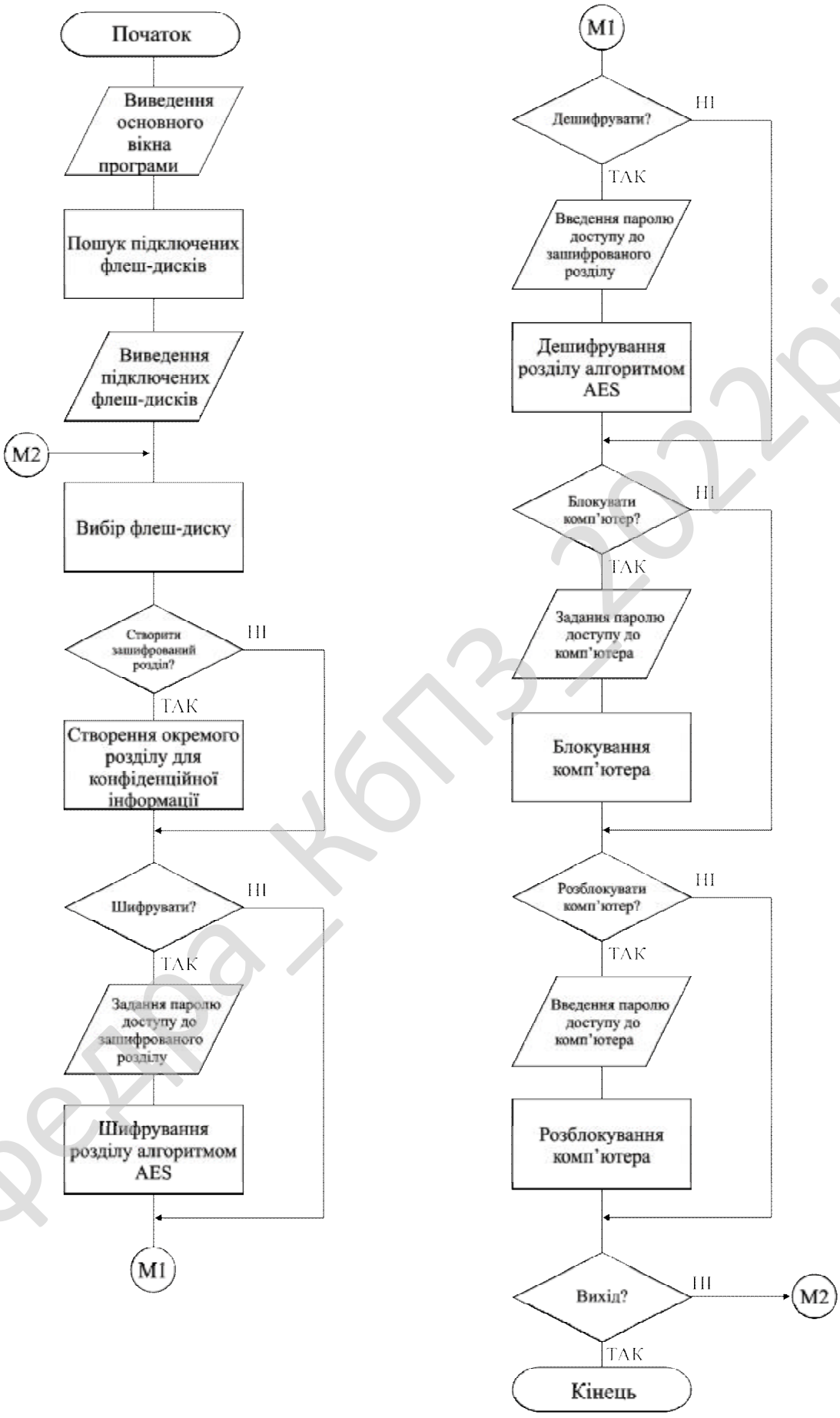

Рисунок 4.1 – Блок-схема основної програми

Якщо ж користувач хоче розблокувати комп’ютер, то він виконує наступні дії:

– Вводить пароль доступу до комп’ютеру.

– Відбувається розблокування комп’ютера.

Після проведення усіх вищеперерахованих дій, користувач обирає працювати йому далі з програмним продуктом, або вийти з програми.

На рисунку 4.2 зображена блок-схема підпрограми шифрування/дешифрування за допомогою алгоритму AES

Підпрограма складається з двох великих блоків:

– Шифрування даних на флеш-диску за допомогою алгоритму AES.

– Дешифрування даних на флеш-диску за допомогою алгоритму AES.

Блок шифрування даних на флеш-диску за допомогою алгоритму AES, складається з виконання наступних кроків:

1. Розбиття даних для шифрування на блоки.
2. Формування сеансового ключа з ключа шифрування.
3. Перемішування байтів відповідно до S-box.
4. Циклічний зсув рядків.
5. Перемішування колонок.
6. Перетворення над даними з використанням сеансового ключа.
7. Формування блоку шифрованих даних та передача їх на флеш-носій.

На цьому крокі підпрограма шифрування закінчує свою роботу.

Блок шифрування даних на флеш-диску за допомогою алгоритму AES, складається з виконання наступних кроків:

1. Розбиття даних для дешифрування на блоки.
2. Формування сеансового ключа з ключа шифрування.
3. Зворотній циклічний зсув рядків.
4. Зворотнє перемішування байтів відповідно до S-box.
5. Перетворення над даними з використанням сеансового ключа.
6. Зворотнє перемішування колонок.

7. Формування блоку дешифрованих даних та передача їх на флеш-носій.

На цьому кроці підпрограма дешифрування закінчує свою роботу.

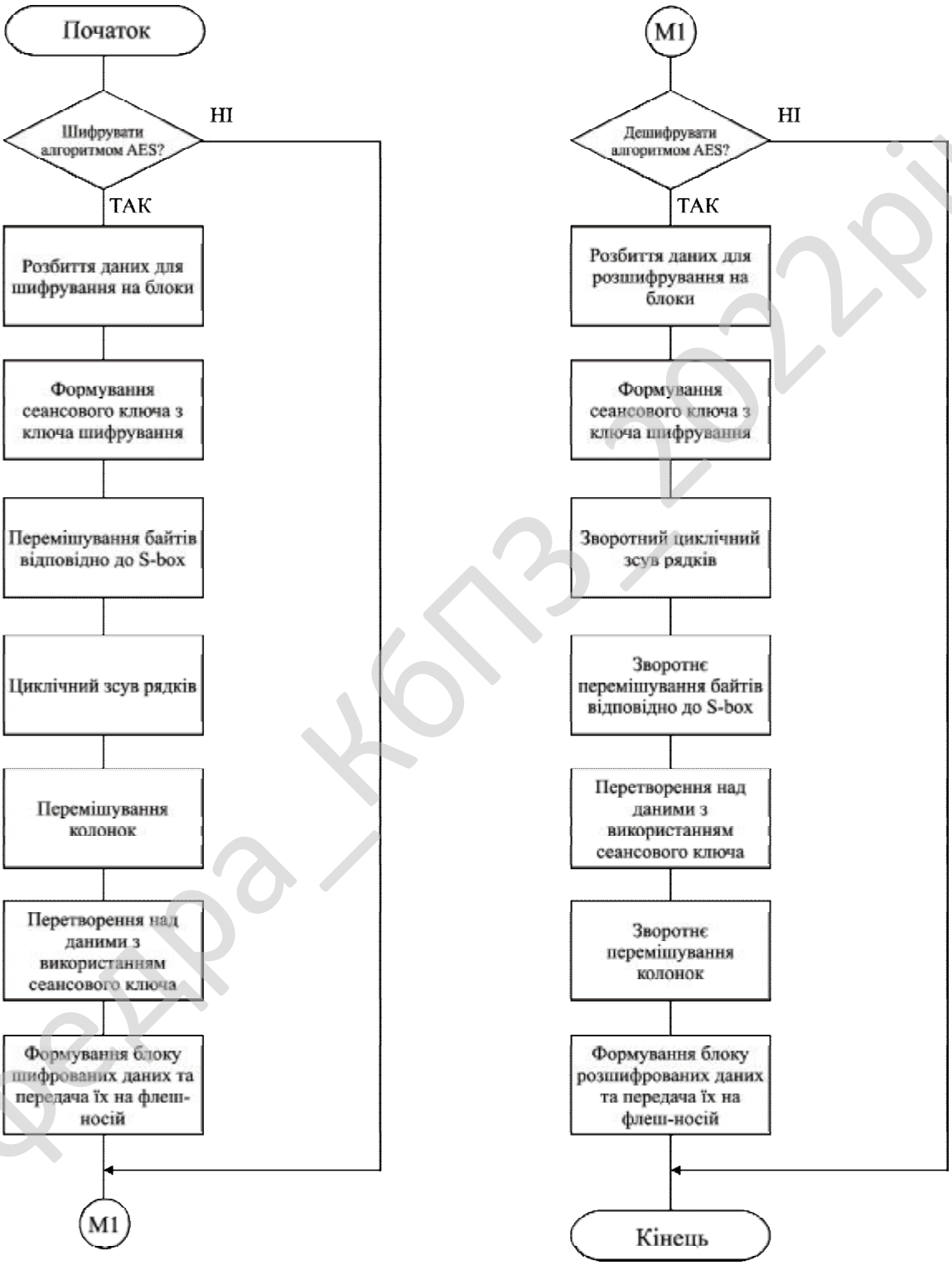

Рисунок 4.2 – Блок-схема підпрограми шифрування/дешифрування за допомогою алгоритму AES

Нижче приведемо частини коду програми, які відповідають за роботу програмного продукту.

Частина коду у якій описана процедура шифрування.

```
// Шифрування
Label_Status.Caption := 'Шифрування...';
Refresh;
Source := TStringStream.Create( Memo_PlainText.Text );
Dest   := TStringStream.Create( '' );
try
  // Зберігаємо дані у пам'яті...
  Size := Source.Size;
  Dest.WriteBuffer( Size, SizeOf(Size) );
  // Підготовлюємо ключ...
  FillChar( Key, SizeOf(Key), 0 );
  Move( PChar(Edit2.Text)^, Key, Min( SizeOf( Key ), Length( Edit2.Text )));
  // Початок Шифрування...
  Start := GetTickCount;
  EncryptAESStreamECB( Source, 0, Key, Dest );
  Stop := GetTickCount;
  Label_Time.Caption := IntToStr(Stop - Start) + ' ms';
  Refresh;
  // Виводимо зашифрований текст використовуючи шістнадцяткове подання...
Memo_CyperText.Lines.BeginUpdate;
  Memo_CyperText.Text := StringToHex( Dest.DataString );
Memo_CyperText.Lines.EndUpdate;
finally
  Source.Free;
  Dest.Free;
end;
```

Наведемо частину коду яка відповідає за дешифрування.

```
// Перетворюємо шістнадцяткову у рядок після дешифрування...
Source := TStringStream.Create( HexToString( Memo_CyperText.Text ));
Dest   := TStringStream.Create( '' );
try
  // Початок дешифрування...
  Size := Source.Size;
  Start := GetTickCount;
  Source.ReadBuffer(Size, SizeOf(Size));
  // Підготовлюємо ключ...
  FillChar(Key, SizeOf(Key), 0);
```

```
    Move(PChar(Edit2.Text)^, Key, Min(SizeOf(Key), Length(Edit2.Text)));
    // Дешифруємо...
    DecryptAESStreamECB(Source, Source.Size - Source.Position, Key, Dest);
    Stop := GetTickCount;
    Label8.Caption := IntToStr(Stop - Start) + ' ms';
    Refresh;
    // Виводимо дешифрований текст...
    Memo_UncipherText.Text := Dest.DataString;
  finally
    Source.Free;
    Dest.Free;
  end;
```

Наведемо один з раундів перетворення інформації, за допомогою сеансового ключа.

```
  // раунд 1
  //зсув рядків
  W0 := ForwardTable[Byte(T0[0])];
  W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)];
  W3 := ForwardTable[Byte(T0[3] shr 24)];
  // перетворення над даними з використанням сеансового ключа
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24))
           xor ((W2 shl 16) or (W2 shr 16))
           xor ((W3 shl 24) or (W3 shr 8)))
           xor Key[4];
  //зсув рядків
  W0 := ForwardTable[Byte(T0[1])];
  W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)];
  W3 := ForwardTable[Byte(T0[0] shr 24)];
  // перетворення над даними з використанням сеансового ключа
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24))
           xor ((W2 shl 16) or (W2 shr 16))
           xor ((W3 shl 24) or (W3 shr 8)))
           xor Key[5];
  //зсув рядків
  W0 := ForwardTable[Byte(T0[2])];
  W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)];
  W3 := ForwardTable[Byte(T0[1] shr 24)];
```

```
// перетворення над даними з використанням сеансового ключа
T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24))
         xor ((W2 shl 16) or (W2 shr 16))
         xor ((W3 shl 24) or (W3 shr 8)))
         xor Key[6];
//зсув рядків
W0 := ForwardTable[Byte(T0[3])];
W1 := ForwardTable[Byte(T0[0] shr 8)];
W2 := ForwardTable[Byte(T0[1] shr 16)];
W3 := ForwardTable[Byte(T0[2] shr 24)];
// перетворення над даними з використанням сеансового ключа
T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24))
          xor ((W2 shl 16) or (W2 shr 16))
          xor ((W3 shl 24) or (W3 shr 8)))
          xor Key[7];
```

Розглянувши блок-схему програми та підпрограми шифрування дешифрування, перейдемо до розділу захисту розробленого програмного забезпечення.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою E, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел $(x, y), x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \tag{4.3}$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина $J(E)$, що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \tag{4.4}$$

Коефіцієнти a, b еліптичної кривої E, по відомому інваріанту $J(E)$, визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \tag{4.5}$$

Пари (x,y), що задовольняють тотожності (4.1), називаються точками еліптичної кривої E, x та y – відповідно x– та y-координатами точки.

Точки еліптичної кривої позначатимемо $Q(x,y)$ або просто Q. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні x– і y-координати.

На безлічі всіх точок еліптичною кривою Е введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$ еліптичної кривої Е, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому

випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \quad (4.6)$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1 та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + ... + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс основного вікна програми.

З цього рисунка ми бачимо, що існують наступні основні елементи меню:

– Флеш диск, де обирається флеш диск з яким користувач буде працювати.

– Шифрування, де обирається який розділ користувач буде шифрувати.

– Блокування комп’ютера, де користувач може вибрати блокувати йому комп’ютер за допомогою флеш-диску, або ні.

– Довідка, у якій знаходяться данні про розробника програмного продукту, його керівника, та місце виконання проекту. Загальний вид вікна довідки наведений на рисунку 5.2.

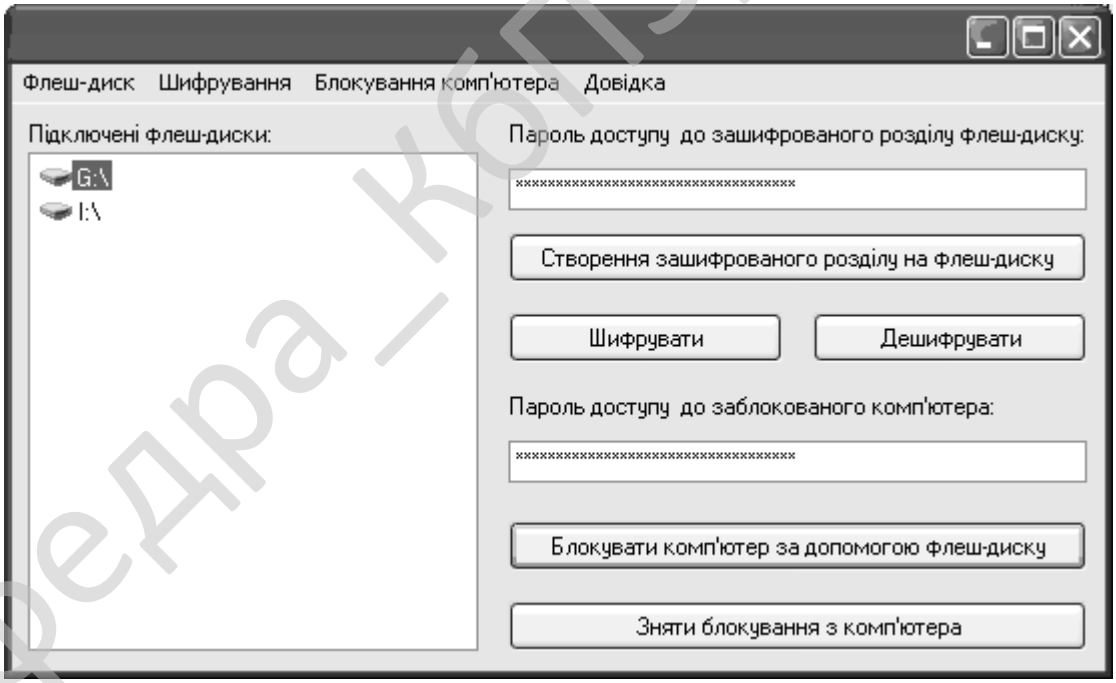

Рисунок 5.1 – Основне вікно програми

Крім того на рисунку 5.1, зображені наступні вікна:

– Вікно відображення підключених флеш-дисків.

– Вікно введення паролю доступу до зашифрованого розділу флеш диску.

– Вікно введення паролю доступу до заблокованого комп’ютера.

Крім того існує можливість створення ряду операцій при нажаті відповідних кнопок:

– Кнопка створення зашифрованого розділу на флеш-диску.

– Кнопка шифрування.

– Кнопка дешифрування.

– Кнопка блокування комп’ютера за допомогою флеш-дисків.

– Кнопка зняття блокування комп’ютера.

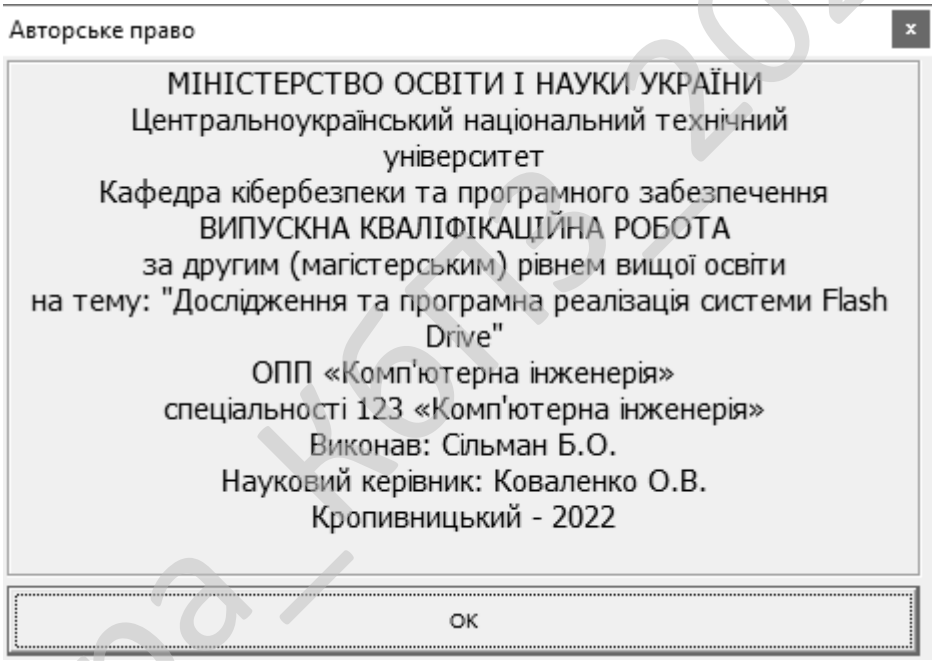

Рисунок 5.2 – Вікно довідки

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи Flash Drive.

Метою розробки є дослідження та програмна реалізація системи Flash Drive.

Об'єктом дослідження є процес Flash Drive.

Предметом дослідження є методи Flash Drive.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод Flash Drive.

– Розроблено вітчизняний продукт Flash Drive, 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 Flash Drive.

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	21
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	21000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

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

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

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

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

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

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

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

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

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

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

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	11	990	16,5
Монітор	60	11	660	11
Клавіатура	30	11	330	5,5
Маніпулятор «мишка»	30	11	330	5,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	2	40	0,67
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м. п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	58,42

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

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

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 145/(60 \cdot 8) = 0.3 \text{ ставки.}$$

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

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

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

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

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

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

Розрахуємо середньоденну зарплату одного виконавця:

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

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

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

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

Згідно даних 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:

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

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

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

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 HARD.kiev.ua за 5.11.22 – джерело https://hard.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		11186

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		6490
Процесор	Intel Xeon E5-2680 SR0KH 2.70GHz/20M LGA2011	1638
Системна плата	Intel X79 LGA2011 DDR3 USB 3.0 WiFi SL ATX	1050
Відеокарта	Видеокарта HP Quadro FX 580 450Mhz PCI E 2.0 1024Mb 1600Mhz 128bit DisplayPort	860
Жорсткий диск	SSD диск Micron RealSSD P400m 200G SAS 6G MLC 2.5 (MTFDEAK200MAS 2S1AA)	980
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC Registered (M393B1K70CH0-CH9Q5)	468
DVD-привод	Super Multi LG SATA DVD±RW R+22x 22x, RW+8x/-6x, DL+16x/-12x, RAM 12x SecurDisc, black (GH22NS40RBB)	432
Корпус	ASUS TA-861 500W FSP ATX-500V (Black/Silver panel) ATX (90-PL861AF5C4 53CZ)	822
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	240
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

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

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

Таблиця 7.7 – Балансова вартість обчислювальної техніки

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

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	214804	-	-
Всього по групі	214804	50	107402
Група 5, 6			
4. Вимірювальні пристрої	5190	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	28000	25	-
Всього по групі	33190	-	8297,5
Нематеріальні активи			
7. Нематеріальні активи	21000	10	2100
Разом	$К_р$ = 1676994		$А_р$ = 188199,5

Примітка: вартість автомобіля приймаємо рівною нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 600 \cdot 209/21 = 5971 \text{ грн.}$$

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

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

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

$$З_д = 5971 \cdot 10 \cdot 0{,}01 = 597 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(5971+597) = 1445 \text{ грн.}$$

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, %.

$$Г_{осп} = 5971 \cdot 15 \cdot 0{,}01 = 896 \text{ грн.}$$

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 0,75 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

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

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

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де Цд – вартість дисків CD: CDR TDK 700Mb, 80Min, Cake box,– 26,4 грн/шт.

$$З_{М2} = 26,4 \cdot 10 = 264 \text{ грн.}$$

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

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (150 + 264 + 1702)/21 = 101 \text{ грн.}$$

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

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

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

$$О_п = 5971 \cdot 15 \cdot 0,01 = 896 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 21 прим.):

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

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

$$А_м = 188200 \cdot 3/(21 \cdot 12) = 2240 \text{ грн.}$$

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

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

$$С_п = 5971+597+1445+896+101+896+2240 = 12146 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 12146 = 6073 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	5971
2. Додаткова зарплата виконавців	$З_д$	597
3. Відрахування на соціальні потреби	$С_{оц}$	1445
4. Загальногосподарські витрати	$Г_{осп}$	896
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	896

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	2240
8. Повна собівартість програмного забезпечення	$С_п$	12146
9. Плановий прибуток	$П_р$	6073
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	18219
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3643,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	21862,8

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	21863
Всього капітальних витрат	–	21863

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	64416	40260
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	5466
Всього витрат за рік	$І$	64416	45726

Витрати на обслуговування:

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

де: $Т_р$ – кількість годин, що витрачається на обслуговування на рік, *год.* (приймаємо 800 год.); $З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

$$З_{р\,баз} = 800 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 64416 \; грн,$$

до:

$$З_{р\,нов} = 500 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 = 40260 \; грн.$$

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

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

$$З_{ел\,баз} = З_{ел\,нов}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	21863	–	5465,75
Всього відрахувань	-	–	21863	–	5465,75

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (18218 - 12146) \cdot 21 - (0,05 \cdot 1408000 + 0,5 \cdot 214804 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 21000) \cdot 3/12 = 80462 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{268994}{(18218 - 12146) \cdot 21 \cdot 12 / 3} = 0{,}5 \text{ років}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	21
2. Повна собівартість розробленої програми	Грн.	12146
3. Ціна розробленої програми	Грн.	18218
4. Плановий прибуток від реалізації одної програми	Грн.	6072
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1676994
7. Загальний прибуток від реалізації програмної продукції	Грн.	127512
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	80462
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	21863
11.Величина економічного ефекту у користувача програмної продукції	Грн.	13224
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Року	1,2

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

$$E_{сп} = (64416\text{-}45726)\text{-}0{,}25 \cdot 21863 = 13224 \text{ грн.}$$

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

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

$$T_{сп} = \frac{21863}{64416 - 45726} = 1,2\,року\,.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

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

роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

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

Ці шкідливі фактори можуть привести до професійних захворювань.

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

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці.

Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [4]

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи

«оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які

сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *50·50·5 мм*., довжиною *L=3 м.,* та горизонтальний електрод – металева полоса з перетином *50·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проведемо 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=0,6 м*. Відстань між вертикальними заземлювачами (електродами) *A=3 м*. Глибина закладення горизонтального контура заземлення *t=0,75 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:

$$T=t+L/2=0,75+3/2=2,25 \text{ м.}$$

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

Еквівалентний діаметр вертикального електрода (кутка) [11]:

$$Dв=0,95\cdot K= 0,95\cdot 50 = 47,5\ мм. =0,0475м.$$

де $K= 50\ мм.$ – розмір металевого кутка (задан).

Відношення $A/L=3/3=1$.

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

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

$$=0,366(54,5/3)[(lg(2\cdot 3/0,0475)+(1/2\ )lg((4\cdot 2,25+3)/(4\cdot 2,25-3))]=14,9\ Ом.$$

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

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

$$N=R_O /(Кев\ R_{ЗН})= 14,9 / (0,53\cdot 4)= 7,05 \approx 7\ шт.$$

Визначаємо довжина з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot 3\cdot 7=18,5\approx 18\ м.$$

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

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

$$=0,366(40\cdot 5/18)\cdot lg((2\cdot 18^{\,2}\ )/(0,05\cdot 0,75\ ))=20,1\ Ом.$$

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

$B = 50\ мм. = 0,05\ м.$ - ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

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

$$=(14,9\cdot 20,1)/(14,9\cdot 0,55+ 7\cdot 20,1\cdot 0,53)=3,56\ Ом.$$

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

Умова $R \le R_{ЗН}$ виконується ($3,56 \le 4$).

Остаточно кількість вертикальних електродів дорівнює 7.

За потреби можна зменшити кількість електродів заземлювача, зменшивши загальний опір розтіканню електричного струму заземлювача методом зменшення питомого опору ґрунта, домішуючи у грунт безпосередньо

навколо електродів заземлювача розчини солей NaCl, CaCl, сажу, соду, шлак або спеціальні суміші.

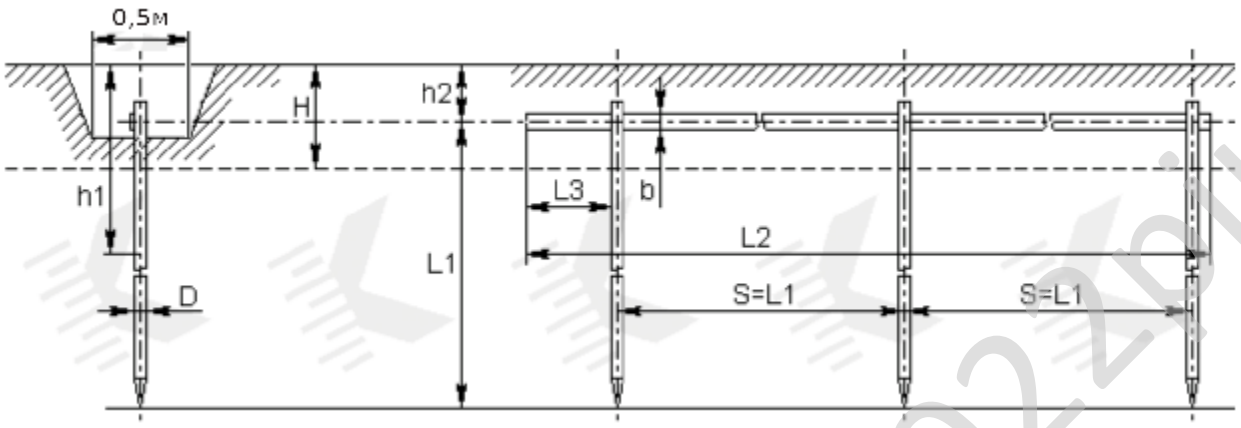

Рисунок 8.1 – Схема штучного заземлення

8.5 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

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

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи Flash Drive.

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 Flash Drive.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем Flash Drive.

– Досліджена система Flash Drive.

– На основі отриманих результатів досліджень створена програмна реалізація системи Flash Drive.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання Flash Drive.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4.1. 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 ДСТУ 4145-2002.

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 13224 грн. З урахуванням вартості розробки програми та обладнання, строк окупления становить 1,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сільман Б.О. Дослідження та програмна реалізація системи Flash Drive // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

3. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

4. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

5. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

6. 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 **(Scopus).**

7. 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 **(Scopus).**

8. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

9. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

10. 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. **(Scopus)**.

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

12. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

14. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

15. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions».

Lecture Notes in Networks and Systems, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

16. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**, «**Representation of Cascade Codes in the Frequency Domain**».** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

17. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus**).

18. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

19. 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. **(Scopus)**.

20. 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. **(Scopus)**.

21. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

22. 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. **(Scopus)**.

23. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

24. 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. **(Scopus)**.

25. 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. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

27. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

28. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. 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. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. 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).

30. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

31. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

32. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

33. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022.

35. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

36. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного

захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

38. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

39. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

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

41. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

42. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

43. 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.

44. 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.

45. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь – 2020. – № 3. – С. 50-61.

46. 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.

47. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

48. Смірнов О.А., Дрєєва Г.М., 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.

49. Смірнов О.А., Смірнова Т.В., Солових Є.К., 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.

50. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

51. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

52. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

53. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

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

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

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

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

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

59. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград:

КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

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

61. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0021.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Сільман Б.О.			Дослідження та програмна реалізація системи Flash Drive	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи Flash Drive.

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. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи Flash Drive.

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 Flash Drive;

– 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 Delphi 10.4.1.

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 2022 року.

8 Вимоги щодо охорони праці

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

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 – 95 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 25.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Коваленко О.В.

Дослідження та програмна реалізація системи Flash Drive

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 55

Літера: РП

Кропивницький – 2022 року

Файл Main.pas – основна програма

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ElAES, Math, Buttons, FleshData, about, FleshBl;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    OpenDialog1: TOpenDialog;
    Button1: TButton;
    Button2: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit2: TEdit;
    Label_Time: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label_Status: TLabel;
    Memo_PlainText: TMemo;
    Memo_CyperText: TMemo;
    Label11: TLabel;
    Label12: TLabel;
    Memo_UncipherText: TMemo;
    Label13: TLabel;
    BitBtn_Encrypt: TBitBtn;
    BitBtn_Decypt: TBitBtn;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure BitBtn_EncryptClick(Sender: TObject);
    procedure BitBtn_DecyptClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;


var
  Form1: TForm1;
  EncryptedText: string;

function StringToHex(S: string): string; forward;
function HexToString(S: string): string; forward;

implementation

{$R *.DFM}

function StringToHex(S: string): string;
var
      i: integer;

begin
```

```
  Result := '';

  // Беремо кожний символ, і конвертуємо його
  // у шістнадцятковий...
      for i := 1 to Length( S ) do
      Result := Result + IntToHex( Ord( S[i] ), 2 );
end;

function HexToString(S: string): string;
var
      i: integer;

begin
  Result := '';

  // Беремо кожний шістнадцятковий символ, та конвертуємо
  // його у ASCII символи...
  for i := 1 to Length( S ) do
  begin
    // Беремо блок з  2 рзрядних шістнадцяткових...
      if ((i mod 2) = 1) then
            Result := Result + Chr( StrToInt( '0x' + Copy( S, i, 2 )));
  end;
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
    Edit1.Text := OpenDialog1.FileName;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  Source, Dest: TFileStream;
  SrcFile, DestFile: string;
  Start, Stop: cardinal;
  Size: integer;
  Key: TAESKey128;
  SrcBuf, DstBuf: array [0..16383] of byte;
  SrcSize, DstSize: integer;
begin
  // Шифрування
  Label_Status.Caption := 'Шифрування...';
  Refresh;
  Source := TFileStream.Create(Edit1.Text, fmOpenRead);
  try
    Label4.Caption := IntToStr(Source.Size div 1024) + ' KB';
    Refresh;
    DestFile := ExtractFilePath(Application.ExeName) + 'aestemp.enc';
    Dest := TFileStream.Create(DestFile, fmCreate);
    try
      Size := Source.Size;
      Dest.WriteBuffer(Size, SizeOf(Size));

      FillChar(Key, SizeOf(Key), 0);
      Move(PChar(Edit2.Text)^, Key, Min(SizeOf(Key), Length(Edit2.Text)));

      Start := GetTickCount;
      EncryptAESStreamECB(Source, 0, Key, Dest);
      Stop := GetTickCount;
      Label_Time.Caption := IntToStr(Stop - Start) + ' ms';
      Refresh;
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
  end;
```

```
  // Дешифрування
  Label_Status.Caption := 'Дешифрування...';
  Refresh;
  Source := TFileStream.Create(DestFile, fmOpenRead);
  try
    Source.ReadBuffer(Size, SizeOf(Size));
    SrcFile := ExtractFilePath(Application.ExeName) + 'aestemp.dec';
    Dest := TFileStream.Create(SrcFile, fmCreate);
    try
      Start := GetTickCount;
      DecryptAESStreamECB(Source, Source.Size - Source.Position, Key, Dest);
      Dest.Size := Size;
      Stop := GetTickCount;
      Label8.Caption := IntToStr(Stop - Start) + ' ms';
      Refresh;
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
  end;
  // Порівняння
  Label_Status.Caption := 'Порівняння...';
  Refresh;
  Source := TFileStream.Create(Edit1.Text, fmOpenRead);
  SrcSize := Source.Size;
  try
    Dest := TFileStream.Create(SrcFile, fmOpenRead);
    DstSize := Dest.Size;
    try
      repeat
        Source.ReadBuffer(SrcBuf, Min(SizeOf(SrcBuf), SrcSize));
        Dest.ReadBuffer(DstBuf, Min(SizeOf(DstBuf), DstSize));
        if not CompareMem(@SrcBuf, @DstBuf, Max(Min(SizeOf(DstBuf), DstSize),
Min(SizeOf(SrcBuf), SrcSize))) then
        begin
          ShowMessage(Помилка!!!');
          DeleteFile(SrcFile);
          DeleteFile(DestFile);
          exit;
        end;
        Dec(SrcSize, Min(SizeOf(SrcBuf), SrcSize));
        Dec(DstSize, Min(SizeOf(DstBuf), DstSize));
      until (SrcSize = 0) or (DstSize = 0);
      ShowMessage('No differences found');
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
  end;
  Label_Status.Caption := 'Видалення...';
  Refresh;
  Label_Status.Caption := '';
end;


procedure TForm1.BitBtn_EncryptClick(Sender: TObject);
var
  Source: TStringStream;
  Dest: TStringStream;
  Start, Stop: cardinal;
  Size: integer;
  Key: TAESKey128;

begin
  // Шифрування
  Label_Status.Caption := 'Шифрування...';
  Refresh;
```

```
  Source := TStringStream.Create( Memo_PlainText.Text );
  Dest   := TStringStream.Create( '' );

  try
    // Зберігаємо дані у пам'яті...
    Size := Source.Size;
    Dest.WriteBuffer( Size, SizeOf(Size) );

    // Підготовлюємо ключ...
    FillChar( Key, SizeOf(Key), 0 );
    Move( PChar(Edit2.Text)^, Key, Min( SizeOf( Key ), Length( Edit2.Text )));

    // Початок Шифрування...
    Start := GetTickCount;
    EncryptAESStreamECB( Source, 0, Key, Dest );
    Stop := GetTickCount;
    Label_Time.Caption := IntToStr(Stop - Start) + ' ms';
    Refresh;

    // Виводимо зашифрований текст використовуючи шістнадцяткове подання...
        Memo_CyperText.Lines.BeginUpdate;
    Memo_CyperText.Text := StringToHex( Dest.DataString );
        Memo_CyperText.Lines.EndUpdate;

  finally
    Source.Free;
    Dest.Free;
  end;
end;

procedure TForm1.BitBtn_DecyptClick(Sender: TObject);
var
  Source: TStringStream;
  Dest: TStringStream;
  Start, Stop: cardinal;
  Size: integer;
  Key: TAESKey128;
  EncryptedText: TStrings;
  S: string;

begin
  // Перетворюємо шістнадцяткову у рядок після дешифрування...
  Source := TStringStream.Create( HexToString( Memo_CyperText.Text ));
  Dest   := TStringStream.Create( '' );

  try
    // Початок дешифрування...
    Size := Source.Size;
    Start := GetTickCount;
    Source.ReadBuffer(Size, SizeOf(Size));

    // Підготовлюємо ключ...
    FillChar(Key, SizeOf(Key), 0);
    Move(PChar(Edit2.Text)^, Key, Min(SizeOf(Key), Length(Edit2.Text)));

    // Дешифруємо...
    DecryptAESStreamECB(Source, Source.Size - Source.Position, Key, Dest);
    Stop := GetTickCount;
    Label8.Caption := IntToStr(Stop - Start) + ' ms';
    Refresh;

    // Виводимо дешифрований текст...
    Memo_UncipherText.Text := Dest.DataString;

  finally
    Source.Free;
    Dest.Free;
  end;
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin
FormAbout.Show;
end;

end.
```

Файл ElAES.pas – шифрування/дешифрування алгоритмом AES

```
unit ElAES;

interface

uses
  Classes, SysUtils;

type
  EAESError = class(Exception);

  PInteger  = ^Integer;

  TAESBuffer = array [0..15] of byte;
  TAESKey128 = array [0..15] of byte;
  TAESKey192 = array [0..23] of byte;
  TAESKey256 = array [0..31] of byte;
  TAESExpandedKey128 = array [0..43] of longword;
  TAESExpandedKey192 = array [0..53] of longword;
  TAESExpandedKey256 = array [0..63] of longword;

  PAESBuffer =^TAESBuffer;
  PAESKey128 =^TAESKey128;
  PAESKey192 =^TAESKey192;
  PAESKey256 =^TAESKey256;
  PAESExpandedKey128 =^TAESExpandedKey128;
  PAESExpandedKey192 =^TAESExpandedKey192;
  PAESExpandedKey256 =^TAESExpandedKey256;

// Розширення ключа для шифрування

procedure ExpandAESKeyForEncryption(const Key: TAESKey128;
  var ExpandedKey: TAESExpandedKey128); overload;
procedure ExpandAESKeyForEncryption(const Key: TAESKey192;
  var ExpandedKey: TAESExpandedKey192); overload;
procedure ExpandAESKeyForEncryption(const Key: TAESKey256;
  var ExpandedKey: TAESExpandedKey256); overload;

// Блок раундів шифрування

procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer); overload;
procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer); overload;
procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer); overload;

// Поток раундів Шифрування (ECB mode)

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream); overload;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream); overload;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream); overload;

// Поток раундів шифрування (CBC mode)
```

```
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

// Перетворення сеансового ключа для дешифрування

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey128);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey128;
  var ExpandedKey: TAESExpandedKey128); overload;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey192);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey192;
  var ExpandedKey: TAESExpandedKey192); overload;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey256);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey256;
  var ExpandedKey: TAESExpandedKey256); overload;

// Блок раундів дешифрування

procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer); overload;
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer); overload;
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer); overload;

// Поток раундів дешифрування (ECB mode)

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream); overload;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream); overload;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream); overload;

// Поток раундів дешифрування (CBC mode)

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream); overload;
```

```
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

resourcestring
  SInvalidInBufSize = 'Не хватає розміру буферу длядешифрування';
  SReadError = 'Stream read error';
  SWriteError = 'Stream write error';

implementation

type
  PLongWord = ^LongWord;

function Min(A, B: integer): integer;
begin
  if A < B then
    Result := A
  else
    Result := B;
end;

const
  Rcon: array [1..30] of longword = (
    $00000001, $00000002, $00000004, $00000008, $00000010, $00000020,
    $00000040, $00000080, $0000001B, $00000036, $0000006C, $000000D8,
    $000000AB, $0000004D, $0000009A, $0000002F, $0000005E, $000000BC,
    $00000063, $000000C6, $00000097, $00000035, $0000006A, $000000D4,
    $000000B3, $0000007D, $000000FA, $000000EF, $000000C5, $00000091
  );

  ForwardTable: array [0..255] of longword = (
    $A56363C6, $847C7CF8, $997777EE, $8D7B7BF6, $0DF2F2FF, $BD6B6BD6, $B16F6FDE,
$54C5C591,
    $50303060, $03010102, $A96767CE, $7D2B2B56, $19FEFEE7, $62D7D7B5, $E6ABAB4D,
$9A7676EC,
    $45CACA8F, $9D82821F, $40C9C989, $877D7DFA, $15FAFAEF, $EB5959B2, $C947478E,
$0BF0F0FB,
    $ECADAD41, $67D4D4B3, $FDA2A25F, $EAAFAF45, $BF9C9C23, $F7A4A453, $967272E4,
$5BC0C09B,
    $C2B7B775, $1CFDFDE1, $AE93933D, $6A26264C, $5A36366C, $413F3F7E, $02F7F7F5,
$4FCCCC83,
    $5C343468, $F4A5A551, $34E5E5D1, $08F1F1F9, $937171E2, $73D8D8AB, $53313162,
$3F15152A,
    $0C040408, $52C7C795, $65232346, $5EC3C39D, $28181830, $A1969637, $0F05050A,
$B59A9A2F,
    $0907070E, $36121224, $9B80801B, $3DE2E2DF, $26EBEBCD, $6927274E, $CDB2B27F,
$9F7575EA,
    $1B090912, $9E83831D, $742C2C58, $2E1A1A34, $2D1B1B36, $B26E6EDC, $EE5A5AB4,
$FBA0A05B,
    $F65252A4, $4D3B3B76, $61D6D6B7, $CEB3B37D, $7B292952, $3EE3E3DD, $712F2F5E,
$97848413,
    $F55353A6, $68D1D1B9, $00000000, $2CEDEDC1, $60202040, $1FFCFCE3, $C8B1B179,
$ED5B5BB6,
    $BE6A6AD4, $46CBCB8D, $D9BEBE67, $4B393972, $DE4A4A94, $D44C4C98, $E85858B0,
$4ACFCF85,
    $6BD0D0BB, $2AEFEFC5, $E5AAAA4F, $16FBFBED, $C5434386, $D74D4D9A, $55333366,
$94858511,
    $CF45458A, $10F9F9E9, $06020204, $817F7FFE, $F05050A0, $443C3C78, $BA9F9F25,
$E3A8A84B,
```

```
    $F35151A2, $FEA3A35D, $C0404080, $8A8F8F05, $AD92923F, $BC9D9D21, $48383870,
$04F5F5F1,
    $DFBCBC63, $C1B6B677, $75DADAAF, $63212142, $30101020, $1AFFFFE5, $0EF3F3FD,
$6DD2D2BF,
    $4CCDCD81, $140C0C18, $35131326, $2FECECC3, $E15F5FBE, $A2979735, $CC444488,
$3917172E,
    $57C4C493, $F2A7A755, $827E7EFC, $473D3D7A, $AC6464C8, $E75D5DBA, $2B191932,
$957373E6,
    $A06060C0, $98818119, $D14F4F9E, $7FDCDCA3, $66222244, $7E2A2A54, $AB90903B,
$8388880B,
    $CA46468C, $29EEEEC7, $D3B8B86B, $3C141428, $79DEDEA7, $E25E5EBC, $1D0B0B16,
$76DBDBAD,
    $3BE0E0DB, $56323264, $4E3A3A74, $1E0A0A14, $DB494992, $0A06060C, $6C242448,
$E45C5CB8,
    $5DC2C29F, $6ED3D3BD, $EFACAC43, $A66262C4, $A8919139, $A4959531, $37E4E4D3,
$8B7979F2,
    $32E7E7D5, $43C8C88B, $5937376E, $B76D6DDA, $8C8D8D01, $64D5D5B1, $D24E4E9C,
$E0A9A949,
    $B46C6CD8, $FA5656AC, $07F4F4F3, $25EAEACF, $AF6565CA, $8E7A7AF4, $E9AEAE47,
$18080810,
    $D5BABA6F, $887878F0, $6F25254A, $722E2E5C, $241C1C38, $F1A6A657, $C7B4B473,
$51C6C697,
    $23E8E8CB, $7CDDDDA1, $9C7474E8, $211F1F3E, $DD4B4B96, $DCBDBD61, $868B8B0D,
$858A8A0F,
    $907070E0, $423E3E7C, $C4B5B571, $AA6666CC, $D8484890, $05030306, $01F6F6F7,
$120E0E1C,
    $A36161C2, $5F35356A, $F95757AE, $D0B9B969, $91868617, $58C1C199, $271D1D3A,
$B99E9E27,
    $38E1E1D9, $13F8F8EB, $B398982B, $33111122, $BB6969D2, $70D9D9A9, $898E8E07,
$A7949433,
    $B69B9B2D, $221E1E3C, $92878715, $20E9E9C9, $49CECE87, $FF5555AA, $78282850,
$7ADFDFA5,
    $8F8C8C03, $F8A1A159, $80898909, $170D0D1A, $DABFBF65, $31E6E6D7, $C6424284,
$B86868D0,
    $C3414182, $B0999929, $772D2D5A, $110F0F1E, $CBB0B07B, $FC5454A8, $D6BBBB6D,
$3A16162C
  );

  LastForwardTable: array [0..255] of longword = (
    $00000063, $0000007C, $00000077, $0000007B, $000000F2, $0000006B, $0000006F,
$000000C5,
    $00000030, $00000001, $00000067, $0000002B, $000000FE, $000000D7, $000000AB,
$00000076,
    $000000CA, $00000082, $000000C9, $0000007D, $000000FA, $00000059, $00000047,
$000000F0,
    $000000AD, $000000D4, $000000A2, $000000AF, $0000009C, $000000A4, $00000072,
$000000C0,
    $000000B7, $000000FD, $00000093, $00000026, $00000036, $0000003F, $000000F7,
$000000CC,
    $00000034, $000000A5, $000000E5, $000000F1, $00000071, $000000D8, $00000031,
$00000015,
    $00000004, $000000C7, $00000023, $000000C3, $00000018, $00000096, $00000005,
$0000009A,
    $00000007, $00000012, $00000080, $000000E2, $000000EB, $00000027, $000000B2,
$00000075,
    $00000009, $00000083, $0000002C, $0000001A, $0000001B, $0000006E, $0000005A,
$000000A0,
    $00000052, $0000003B, $000000D6, $000000B3, $00000029, $000000E3, $0000002F,
$00000084,
    $00000053, $000000D1, $00000000, $000000ED, $00000020, $000000FC, $000000B1,
$0000005B,
    $0000006A, $000000CB, $000000BE, $00000039, $0000004A, $0000004C, $00000058,
$000000CF,
    $000000D0, $000000EF, $000000AA, $000000FB, $00000043, $0000004D, $00000033,
$00000085,
    $00000045, $000000F9, $00000002, $0000007F, $00000050, $0000003C, $0000009F,
$000000A8,
    $00000051, $000000A3, $00000040, $0000008F, $00000092, $0000009D, $00000038,
$000000F5,
```

```
    $000000BC, $000000B6, $000000DA, $00000021, $00000010, $000000FF, $000000F3,
$000000D2,
    $000000CD, $0000000C, $00000013, $000000EC, $0000005F, $00000097, $00000044,
$00000017,
    $000000C4, $000000A7, $0000007E, $0000003D, $00000064, $0000005D, $00000019,
$00000073,
    $00000060, $00000081, $0000004F, $000000DC, $00000022, $0000002A, $00000090,
$00000088,
    $00000046, $000000EE, $000000B8, $00000014, $000000DE, $0000005E, $0000000B,
$000000DB,
    $000000E0, $00000032, $0000003A, $0000000A, $00000049, $00000006, $00000024,
$0000005C,
    $000000C2, $000000D3, $000000AC, $00000062, $00000091, $00000095, $000000E4,
$00000079,
    $000000E7, $000000C8, $00000037, $0000006D, $0000008D, $000000D5, $0000004E,
$000000A9,
    $0000006C, $00000056, $000000F4, $000000EA, $00000065, $0000007A, $000000AE,
$00000008,
    $000000BA, $00000078, $00000025, $0000002E, $0000001C, $000000A6, $000000B4,
$000000C6,
    $000000E8, $000000DD, $00000074, $0000001F, $0000004B, $000000BD, $0000008B,
$0000008A,
    $00000070, $0000003E, $000000B5, $00000066, $00000048, $00000003, $000000F6,
$0000000E,
    $00000061, $00000035, $00000057, $000000B9, $00000086, $000000C1, $0000001D,
$0000009E,
    $000000E1, $000000F8, $00000098, $00000011, $00000069, $000000D9, $0000008E,
$00000094,
    $0000009B, $0000001E, $00000087, $000000E9, $000000CE, $00000055, $00000028,
$000000DF,
    $0000008C, $000000A1, $00000089, $0000000D, $000000BF, $000000E6, $00000042,
$00000068,
    $00000041, $00000099, $0000002D, $0000000F, $000000B0, $00000054, $000000BB,
$00000016
  );

  InverseTable: array [0..255] of longword = (
    $50A7F451, $5365417E, $C3A4171A, $965E273A, $CB6BAB3B, $F1459D1F, $AB58FAAC,
$9303E34B,
    $55FA3020, $F66D76AD, $9176CC88, $254C02F5, $FCD7E54F, $D7CB2AC5, $80443526,
$8FA362B5,
    $495AB1DE, $671BBA25, $980EEA45, $E1C0FE5D, $02752FC3, $12F04C81, $A397468D,
$C6F9D36B,
    $E75F8F03, $959C9215, $EB7A6DBF, $DA595295, $2D83BED4, $D3217458, $2969E049,
$44C8C98E,
    $6A89C275, $78798EF4, $6B3E5899, $DD71B927, $B64FE1BE, $17AD88F0, $66AC20C9,
$B43ACE7D,
    $184ADF63, $82311AE5, $60335197, $457F5362, $E07764B1, $84AE6BBB, $1CA081FE,
$942B08F9,
    $58684870, $19FD458F, $876CDE94, $B7F87B52, $23D373AB, $E2024B72, $578F1FE3,
$2AAB5566,
    $0728EBB2, $03C2B52F, $9A7BC586, $A50837D3, $F2872830, $B2A5BF23, $BA6A0302,
$5C8216ED,
    $2B1CCF8A, $92B479A7, $F0F207F3, $A1E2694E, $CDF4DA65, $D5BE0506, $1F6234D1,
$8AFEA6C4,
    $9D532E34, $A055F3A2, $32E18A05, $75EBF6A4, $39EC830B, $AAEF6040, $069F715E,
$51106EBD,
    $F98A213E, $3D06DD96, $AE053EDD, $46BDE64D, $B58D5491, $055DC471, $6FD40604,
$FF155060,
    $24FB9819, $97E9BDD6, $CC434089, $779ED967, $BD42E8B0, $888B8907, $385B19E7,
$DBEEC879,
    $470A7CA1, $E90F427C, $C91E84F8, $00000000, $83868009, $48ED2B32, $AC70111E,
$4E725A6C,
    $FBFF0EFD, $5638850F, $1ED5AE3D, $27392D36, $64D90F0A, $21A65C68, $D1545B9B,
$3A2E3624,
    $B1670A0C, $0FE75793, $D296EEB4, $9E919B1B, $4FC5C080, $A220DC61, $694B775A,
$161A121C,
    $0ABA93E2, $E52AA0C0, $43E0223C, $1D171B12, $0B0D090E, $ADC78BF2, $B9A8B62D,
$C8A91E14,
```

```
    $8519F157, $4C0775AF, $BBDD99EE, $FD607FA3, $9F2601F7, $BCF5725C, $C53B6644,
$347EFB5B,
    $7629438B, $DCC623CB, $68FCEDB6, $63F1E4B8, $CADC31D7, $10856342, $40229713,
$2011C684,
    $7D244A85, $F83DBBD2, $1132F9AE, $6DA129C7, $4B2F9E1D, $F330B2DC, $EC52860D,
$D0E3C177,
    $6C16B32B, $99B970A9, $FA489411, $2264E947, $C48CFCA8, $1A3FF0A0, $D82C7D56,
$EF903322,
    $C74E4987, $C1D138D9, $FEA2CA8C, $360BD498, $CF81F5A6, $28DE7AA5, $268EB7DA,
$A4BFAD3F,
    $E49D3A2C, $0D927850, $9BCC5F6A, $62467E54, $C2138DF6, $E8B8D890, $5EF7392E,
$F5AFC382,
    $BE805D9F, $7C93D069, $A92DD56F, $B31225CF, $3B99ACC8, $A77D1810, $6E639CE8,
$7BBB3BDB,
    $097826CD, $F418596E, $01B79AEC, $A89A4F83, $656E95E6, $7EE6FFAA, $08CFBC21,
$E6E815EF,
    $D99BE7BA, $CE366F4A, $D4099FEA, $D67CB029, $AFB2A431, $31233F2A, $3094A5C6,
$C066A235,
    $37BC4E74, $A6CA82FC, $B0D090E0, $15D8A733, $4A9804F1, $F7DAEC41, $0E50CD7F,
$2FF69117,
    $8DD64D76, $4DB0EF43, $544DAACC, $DF0496E4, $E3B5D19E, $1B886A4C, $B81F2CC1,
$7F516546,
    $04EA5E9D, $5D358C01, $737487FA, $2E410BFB, $5A1D67B3, $52D2DB92, $335610E9,
$1347D66D,
    $8C61D79A, $7A0CA137, $8E14F859, $893C13EB, $EE27A9CE, $35C961B7, $EDE51CE1,
$3CB1477A,
    $59DFD29C, $3F73F255, $79CE1418, $BF37C773, $EACDF753, $5BAAFD5F, $146F3DDF,
$86DB4478,
    $81F3AFCA, $3EC468B9, $2C342438, $5F40A3C2, $72C31D16, $0C25E2BC, $8B493C28,
$41950DFF,
    $7101A839, $DEB30C08, $9CE4B4D8, $90C15664, $6184CB7B, $70B632D5, $745C6C48,
$4257B8D0
  );

  LastInverseTable: array [0..255] of longword = (
    $00000052, $00000009, $0000006A, $000000D5, $00000030, $00000036, $000000A5,
$00000038,
    $000000BF, $00000040, $000000A3, $0000009E, $00000081, $000000F3, $000000D7,
$000000FB,
    $0000007C, $000000E3, $00000039, $00000082, $0000009B, $0000002F, $000000FF,
$00000087,
    $00000034, $0000008E, $00000043, $00000044, $000000C4, $000000DE, $000000E9,
$000000CB,
    $00000054, $0000007B, $00000094, $00000032, $000000A6, $000000C2, $00000023,
$0000003D,
    $000000EE, $0000004C, $00000095, $0000000B, $00000042, $000000FA, $000000C3,
$0000004E,
    $00000008, $0000002E, $000000A1, $00000066, $00000028, $000000D9, $00000024,
$000000B2,
    $00000076, $0000005B, $000000A2, $00000049, $0000006D, $0000008B, $000000D1,
$00000025,
    $00000072, $000000F8, $000000F6, $00000064, $00000086, $00000068, $00000098,
$00000016,
    $000000D4, $000000A4, $0000005C, $000000CC, $0000005D, $00000065, $000000B6,
$00000092,
    $0000006C, $00000070, $00000048, $00000050, $000000FD, $000000ED, $000000B9,
$000000DA,
    $0000005E, $00000015, $00000046, $00000057, $000000A7, $0000008D, $0000009D,
$00000084,
    $00000090, $000000D8, $000000AB, $00000000, $0000008C, $000000BC, $000000D3,
$0000000A,
    $000000F7, $000000E4, $00000058, $00000005, $000000B8, $000000B3, $00000045,
$00000006,
    $000000D0, $0000002C, $0000001E, $0000008F, $000000CA, $0000003F, $0000000F,
$00000002,
    $000000C1, $000000AF, $000000BD, $00000003, $00000001, $00000013, $0000008A,
$0000006B,
    $0000003A, $00000091, $00000011, $00000041, $0000004F, $00000067, $000000DC,
$000000EA,
```

```
    $00000097, $000000F2, $000000CF, $000000CE, $000000F0, $000000B4, $000000E6,
$00000073,
    $00000096, $000000AC, $00000074, $00000022, $000000E7, $000000AD, $00000035,
$00000085,
    $000000E2, $000000F9, $00000037, $000000E8, $0000001C, $00000075, $000000DF,
$0000006E,
    $00000047, $000000F1, $0000001A, $00000071, $0000001D, $00000029, $000000C5,
$00000089,
    $0000006F, $000000B7, $00000062, $0000000E, $000000AA, $00000018, $000000BE,
$0000001B,
    $000000FC, $00000056, $0000003E, $0000004B, $000000C6, $000000D2, $00000079,
$00000020,
    $0000009A, $000000DB, $000000C0, $000000FE, $00000078, $000000CD, $0000005A,
$000000F4,
    $0000001F, $000000DD, $000000A8, $00000033, $00000088, $00000007, $000000C7,
$00000031,
    $000000B1, $00000012, $00000010, $00000059, $00000027, $00000080, $000000EC,
$0000005F,
    $00000060, $00000051, $0000007F, $000000A9, $00000019, $000000B5, $0000004A,
$0000000D,
    $0000002D, $000000E5, $0000007A, $0000009F, $00000093, $000000C9, $0000009C,
$000000EF,
    $000000A0, $000000E0, $0000003B, $0000004D, $000000AE, $0000002A, $000000F5,
$000000B0,
    $000000C8, $000000EB, $000000BB, $0000003C, $00000083, $00000053, $00000099,
$00000061,
    $00000017, $0000002B, $00000004, $0000007E, $000000BA, $00000077, $000000D6,
$00000026,
    $000000E1, $00000069, $00000014, $00000063, $00000055, $00000021, $0000000C,
$0000007D
  );

procedure ExpandAESKeyForEncryption(const Key: TAESKey128; var ExpandedKey:
TAESExpandedKey128);
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 3] shl 24) or (ExpandedKey[I + 3] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 4] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
    Inc(J);
    ExpandedKey[I + 5] := ExpandedKey[I + 1] xor ExpandedKey[I + 4];
    ExpandedKey[I + 6] := ExpandedKey[I + 2] xor ExpandedKey[I + 5];
    ExpandedKey[I + 7] := ExpandedKey[I + 3] xor ExpandedKey[I + 6];
    Inc(I, 4);
  until I >= 40;
end;

procedure ExpandAESKeyForEncryption(const Key: TAESKey192; var ExpandedKey:
TAESExpandedKey192); overload;
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
```

```
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  ExpandedKey[4] := PLongWord(@Key[16])^;
  ExpandedKey[5] := PLongWord(@Key[20])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 5] shl 24) or (ExpandedKey[I + 5] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 6] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
    Inc(J);
    ExpandedKey[I + 7] := ExpandedKey[I + 1] xor ExpandedKey[I + 6];
    ExpandedKey[I + 8] := ExpandedKey[I + 2] xor ExpandedKey[I + 7];
    ExpandedKey[I + 9] := ExpandedKey[I + 3] xor ExpandedKey[I + 8];
    ExpandedKey[I + 10] := ExpandedKey[I + 4] xor ExpandedKey[I + 9];
    ExpandedKey[I + 11] := ExpandedKey[I + 5] xor ExpandedKey[I + 10];
    Inc(I, 6);
  until I >= 46;
end;

procedure ExpandAESKeyForEncryption(const Key: TAESKey256; var ExpandedKey:
TAESExpandedKey256); overload;
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  ExpandedKey[4] := PLongWord(@Key[16])^;
  ExpandedKey[5] := PLongWord(@Key[20])^;
  ExpandedKey[6] := PLongWord(@Key[24])^;
  ExpandedKey[7] := PLongWord(@Key[28])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 7] shl 24) or (ExpandedKey[I + 7] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 8] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
    Inc(J);
    ExpandedKey[I + 9] := ExpandedKey[I + 1] xor ExpandedKey[I + 8];
    ExpandedKey[I + 10] := ExpandedKey[I + 2] xor ExpandedKey[I + 9];
    ExpandedKey[I + 11] := ExpandedKey[I + 3] xor ExpandedKey[I + 10];
    W0 := LastForwardTable[Byte(ExpandedKey[I + 11])];
    W1 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 8)];
    W2 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 16)];
    W3 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 24)];
    ExpandedKey[I + 12] := ExpandedKey[I + 4] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8)));
    ExpandedKey[I + 13] := ExpandedKey[I + 5] xor ExpandedKey[I + 12];
    ExpandedKey[I + 14] := ExpandedKey[I + 6] xor ExpandedKey[I + 13];
    ExpandedKey[I + 15] := ExpandedKey[I + 7] xor ExpandedKey[I + 14];
    Inc(I, 8);
  until I >= 52;
end;

procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
```

```
  W0, W1, W2, W3: longword;
begin
  // Ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація - 9 раз
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
```

```
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
```

```
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
```

```
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація - 11 раз
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
```

```
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
```

```
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
```

```
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 10
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 11
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
```

```
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація 13 разів
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 10
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 11
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 12
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // раунд 13
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[52];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[53];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[54];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[55];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[56];
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[57];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[58];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[59];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey128);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 9 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
```

```
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure ExpandAESKeyForDecryption(const Key: TAESKey128; var ExpandedKey:
TAESExpandedKey128);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey192);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 11 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
```

```
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure ExpandAESKeyForDecryption(const Key: TAESKey192; var ExpandedKey:
TAESExpandedKey192);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey256);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 13 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
```

```
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure ExpandAESKeyForDecryption(const Key: TAESKey256; var ExpandedKey:
TAESExpandedKey256);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[40];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[41];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[42];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[43];
  // Попередня трансформація  9 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
```

```
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
```

```
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
```

```
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
```

```
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[48];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[49];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[50];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[51];
  // Попередня трансформація  11 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 10
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 11
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;
```

```
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[56];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[57];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[58];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[59];
  // Попередня трансформація  13 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[52];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[53];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[54];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[55];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
```

```
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
```

```
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 10
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
```

```
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 11
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 12
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 13
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
```

```
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

// Поток раундів шифрування (ECB mode)

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;
```

```
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
```

```
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

// Поток раундів дешифрування (ECB mode)

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
```

```
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
```

```
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

// Поток раундів шифрування (CBC mode)

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
```

```
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
```

```
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

// Поток раундів дешифрування (CBC mode)

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;
```

```
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do
```

```
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(TAESBuffer));
```

```
  end;
end;

end.
```

Файл FleshBl.pas – блокування комп'ютера за допомогою флеш-диску

```
unit FleshBl;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,tlhelp32,buttons,registry;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormPaint2(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  stop:boolean;
  pass:string;

procedure BlockInput; external user32;

implementation

uses Unit2;

{$R *.dfm}

function GetpidByname(sl:string):cardinal;
var
  h: HWND;
  snap: tprocessentry32;
begin
result:=0;
  h := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, cardinal(-1));
  snap.dwSize := SizeOf(ProcessEntry32);
  if process32first(h, snap) = true then
  repeat
            if lowercase(sl)=lowercase(snap.szExeFile)then
            begin
                 result:=snap.th32ProcessID;
                 break;
            end;
  until process32next(h,snap)<>true;
  closehandle(h);
end;

procedure Block(s:boolean);
var
p:cardinal;
begin
if s=true then
begin
asm
```

```
   push 1
   call blockinput;
end;
p:=getpidbyname('taskmgr.exe');
if p<>0 then
   begin
        p:=Openprocess(PROCESS_TERMINATE,true,p);
        terminateprocess(p,1);
        closehandle(p);
   end;
setwindowpos(form1.handle,HWND_TOPMOST,0,0,0,0,swp_nomove or swp_nosize);
end else
begin
asm
   push 0
   call blockinput;
end;
end;
end;

procedure Scan;
var
j:word;
a:array[1..512]of byte;
s:string;
readed:cardinal;
f:cardinal;
begin
     for j:=ord('A')  to ord('Z') do
     if getdrivetype(pchar(chr(j)+':\'))=DRIVE_REMOVABLE then
     begin

f:=createfile(pchar('\\.\'+chr(j)+':'),GENERIC_READ,FILE_SHARE_READ,nil,OPEN_EXI
STING,FILE_FLAG_RANDOM_ACCESS,0);
          if f<>INVALID_HANDLE_VALUE then
          begin
          setfilepointer(f,512*4,nil,0);
          readfile(f,a,sizeof(a),readed,nil);
          readed:=1;
          while (a[readed]<>0) do
          begin
               s:=s+chr(a[readed]);
               inc(readed);
          end;
          if s=pass then
             begin
                block(false);
                winexec(pchar('"'+paramstr(0)+'" -d '+chr(j)),sw_show);
                exitprocess(1);
             end;
          closehandle(f);
          end;
     end;
end;

procedure DoBlock;
begin
stop:=false;
while stop<>true do
      begin
           block(true);
           scan;
           application.ProcessMessages;
      end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
showcursor(false);
```

```
onpaint:=formpaint2;
left:=0;
top:=0;
button1.Hide;
button2.Hide;
button3.Hide;
showcursor(false);
color:=clBlack;
width:=screen.Width;
height:=screen.Height;
doblock;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
exitprocess(1);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
with TRegistry.Create(KEY_READ)do
begin
     rootkey:=HKEY_CURRENT_USER;
     if openkey('software\microsoft\windows\currentversion\Run',false)=true then
        form2.CheckBox1.Checked:=valueexists('fdcl');
     closekey;
     free;
end;
form2.Show;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
buttons.DrawButtonFace(canvas,clientrect,3,bsNew,true,false,false);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
left:=screen.Width-width;
top:=screen.WorkAreaHeight-height;
with TRegistry.Create(KEY_READ)do
begin
     rootkey:=HKEY_CURRENT_USER;
     if openkey('SOFTWARE',false)=true then
     if valueexists('fdcl')then
        pass:=readstring('fdcl');
     closekey;
     free;
end;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
showwindow(application.Handle,sw_hide);
if paramstr(1)='-lock'then button1.Click;
if paramstr(1)='-d'then
begin
   while windows.GetDriveType(pchar(paramstr(2)+':\'))=DRIVE_REMOVABLE do
         application.ProcessMessages;
   button1.Click;
end;
end;

procedure TForm1.FormPaint2(Sender: TObject);
begin
canvas.Font.Color:=clLime;
canvas.Font.Name:='Times New Roman';
canvas.font.Size:=30;
canvas.TextOut(0,0,'Unlock with Key-Flash');
```

```
end;

end.
```

Файл about.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm5 = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.dfm}

procedure TForm5.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add(' МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add(' на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи Flash Drive ');
Memo1.Lines.Add(' ');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Керівник: Коваленко О.В.');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Розробив: студент Сільман Богдан Олегович ');
Memo1.Lines.Add('                    гр. КІ-21М-1,4');
Memo1.Lines.Add('');
Memo1.Lines.Add(' м. Кропивницький 2022');
Memo1.Lines.Add('');
end;

procedure TForm5.Button1Click(Sender: TObject);
begin
Form5.Close;
end;
end.
```