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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Скакун П.П.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Улічев О.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

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

Скакуну Павлу Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу*

2. Керівник роботи *Улічев Олександр Сергійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 від 04.08.2023 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Скакун П.П. Дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Метою розробки є дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Об'єктом дослідження є процес створення архівів у файловій системі NTFS з розмежуванням доступу.

Предметом дослідження є методи створення архівів у файловій системі NTFS з розмежуванням доступу.

Методи дослідження базуються на методах теорії інформації та кодування, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

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.

Ключові слова: комп'ютерні науки, NTFS, розмежуванням доступу

ABSTRACT

Skakun P.P. Research and software implementation of the system for creating archives in the NTFS file system with access demarcation. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of creating archives in the NTFS file system with access demarcation.

The purpose of the development is the research and software implementation of the system for creating archives in the NTFS file system with access delimitation.

The object of research is the process of creating archives in the NTFS file system with limited access.

The subject of the study is the methods of creating archives in the NTFS file system with limited access.

Research methods are based on methods of information theory and coding, methods of mathematical statistics, software development methods.

The result of the work is the software implementation of the system for creating archives in the NTFS file system with access demarcation.

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 environment.

Key words: computer science, NTFS, access partitioning

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. Програми для стиску даних почали розроблятися одночасно зі створенням перших персональних комп'ютерів, адже ще тоді постійно відчувалася недостача вільного місця на жорстких дисках. Тим більше, що переносних носіїв з великим обсягом збереженої інформації носіїв ще не було, а щоб умістити на дискету навіть невеликий текстовий документ, доводилося придумувати різні хитрування. Багато кращих архіваторів тих часів і сьогодні радують користувачів новими версіями, продовжуючи допомагати в питаннях переносу, а також тривалого зберігання даних. Як правило, користувачі займаються стиском текстових документів і книг, рідше – фотографій і відеоданих, тому що в останньому випадку виграш у вільному місці виявляється зовсім невеликим. Процес створення архіву називається архівацією або упакуванням, а добування файлів з архіву – розпакуванням або екстракцією.

Те, що є можливість архівувати інформацію – це добре. Але при перенесенні її на з'ємних носіях, або при передачі по мережі, гостро постають питання безпеки цієї заархівованої інформації. Причини можуть бути різні. Існує величезна кількість програм розмежування доступу, з ціллю збереження конфіденційності, але всі вони (принаймні більшість) мають один істотний недолік. Існує потенційна небезпека розкриття конфіденційних даних. Справа в тому, що програма розмежування доступу, з ціллю збереження конфіденційності, повинна зберігати незашифровану копію даних на диску (якщо звичайно операції читання/запису не виконуються разом із шифруванням/дешифруванням, що віднімає багато ресурсів) для користувача й тому існує потенційна можливість роздобути цю інформацію.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

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

– Огляд існуючих систем створення архівів у файловій системі NTFS з розмежуванням доступу.

– Дослідження системи створення архівів у файловій системі NTFS з розмежуванням доступу.

– Програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Об'єктом дослідження є процес створення архівів у файловій системі NTFS з розмежуванням доступу.

Предметом дослідження є методи створення архівів у файловій системі NTFS з розмежуванням доступу.

Методи дослідження базуються на методах теорії інформації та кодування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод створення архівів у файловій системі NTFS з розмежуванням доступу.

– Розроблено вітчизняний продукт створення архівів у файловій системі NTFS з розмежуванням доступу, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі створення архівів у файловій системі NTFS з розмежуванням доступу.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу, 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

Система, яка розробляється у ході виконання магістерського проектування призначена для системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Розмежування доступу, з ціллю збереження конфіденційності, на рівні файлів, має більшу гнучкість для користувача, ніж розмежування доступу, з ціллю збереження конфіденційності, на рівні диска. Набагато зручніше вибрати на диску декілька файлів або каталогів і оголосити їх шифрованими, ніж возитися зі створенням віртуальних дисків або перерозмічати жорсткий диск, щоб створити на ньому окремий шифрований розділ. Однак така гнучкість таїть у собі підводні камені. По-перше, виникають складності, коли необхідно, щоб до якихось файлів був забезпечений доступ більше одного користувача. По-друге, існує потенційна небезпека витоку даних при створенні тимчасових файлів за межами зашифрованих каталогів, це може робити будь-яка програма, не доводити до відома користувача. Нарешті, не треба забувати про людського фактора: користувач може забути про те, що необхідно зберігати всі файли строго в шифруємому каталозі, і конфіденційна інформація виявиться на диску у відкритому вигляді

Якщо говорити про розмежування доступу, з ціллю збереження конфіденційності, на рівні диска, то його теж можна реалізовувати по-різному.

Перший спосіб – файл-контейнер. Суть у тому, що на жорсткому диску комп'ютера створюється файл великого розміру, що спеціальним драйвером «проектується» у системі як ще один диск. Усе, що записується на цей віртуальний диск, насправді зашифровується й записується у файл. Зворотна операція відбувається при читанні. Всі програми, системні утиліти й користувачі

бачать цей віртуальний диск так, ніби він був звичайним жорстким диском. Однак системи типу «файл-контейнер» мають один досить істотний недолік, що полягає в набагато більшому падінні продуктивності при читанні-запису, у порівнянні із системами блокового розмежування доступу, з ціллю збереження конфіденційності, роздягнула, і теоретично в меншій надійності таких систем.

Другий спосіб – блокове розмежування доступу, з ціллю збереження конфіденційності, розділу жорсткого диска. При цьому ніяких файлів-контейнерів і додаткових віртуальних дисків не створюється, а шифрується один або кілька існуючих розділів жорсткого диска. Ця функціональність забезпечується спеціальним драйвером-фільтром диска, інформація при цьому шифрується посекторно, у процесі читання-запису.

1.2 Область застосування

Областю застосування системи, яка розробляється у ході виконання магістерського проектування є створення архівів у файловій системі NTFS з розмежуванням доступу, з ціллю збереження конфіденційності, файлів. Більшість систем не захищені, хоча такі відомі засоби "безпеки", як паролі Windows, паролі ZIP-файлів, паролі BIOS і FTP/Web мають на увазі захист. Насправді правда от у чому: усе, що обробляється або зберігається у відкритому виді (як у випадку з тільки що перерахованими прикладами), можна обійти. Паролі Windows зберігаються в системній пам'яті й забезпечують захист, тільки якщо не доступні інші шляхи доступу, наприклад, через мережу або завантаження з USB-накопичувача. Якщо запастися терпінням, то можна розкрити ZIP-файли шляхом підбора пароля методом грубої сили (brute-force), а багато хто web-служби взагалі не використовують ніякого розмежування доступу, з ціллю збереження конфіденційності, при обробці даних для входу. Справжня безпека можлива тільки в тому випадку, якщо дані й канали захищені за допомогою сучасних

алгоритмів розмежування доступу, з ціллю збереження конфіденційності, при використанні надійних паролів.

Коли ми задумалися про засоби захисту інформації, нам згадалися такі технології, як модуль Trusted Platform Module, реалізований у вигляді чипа на материнській платі, що перевіряє стан системи при включенні комп'ютера, валідує програмне забезпечення або користувачів. Є компоненти з убудованим апаратним прискоренням розмежування доступу, з ціллю збереження конфіденційності, й дешифрування; свіжий приклад тому процесор VIA Nano. Є навіть компоненти, які мають убудоване розмежування доступу, з ціллю збереження конфіденційності,: зараз популярні тверді диски, що самошифруються, а дорогі версії Windows 10/11 Ultimate і Enterprise підтримують функцію апаратного розмежування доступу, з ціллю збереження конфіденційності, диска за назвою "Bit Locker" (при наявності чипа TPM).

Однак більшість рішень містять "підводні камені". Вони або вимагають, щоб ви купили відповідні програми або компоненти, або вам доведеться змінити спосіб роботи зі своєю системою. Крім того, жодне з рішень не забезпечує справжньої безпеки, оскільки для них є обхідні шляхи, які наражають на небезпеку ваші дані. Зовнішні жорсткі диски з убудованим шифруванням іноді містять навмисні або ненавмисні бекдори; інші приклади ми назвали вище.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу, 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

Для початку розберемося, що таке архіватор, і навіщо він потрібний. Архіватор – це така програма, що може впакувати Ваші файли, наприклад фото з відпочинку, які потрібно відправити поштою, в один або кілька архівів. Це робиться для зручності зберігання даних і їхньої передачі. Також ще однією особливістю архіваторів є стиск даних, це робиться для економії місця на носії. Це дуже актуально.

Інтерфейс і оформлення програм

7-Zip

Першим виступить архіватор 7-Zip.

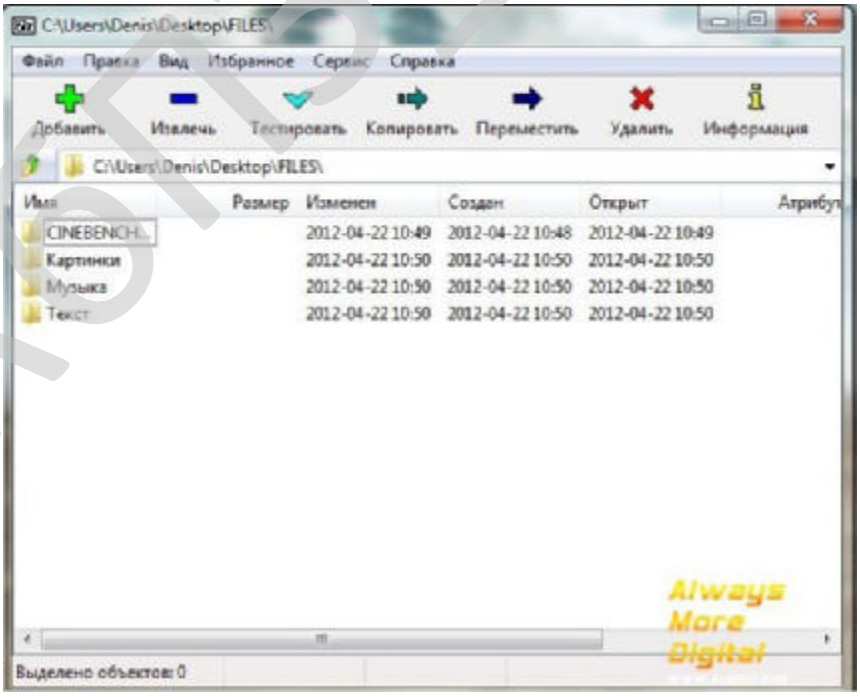

Рисунок 2.1 – Інтерфейс користувача 7-Zip

Почнемо з того, що цей архіватор абсолютно безкоштовний. Це безсумнівно сподобається аматорам ліцензійного ПЗ, совість яких не дає використовувати "піратські" програми. Інтерфейс програми простий і дуже зручний. Основним кольором оформлення виступає білий і лише іконки, які можна зробити більше раза у два, кольорові.

Просто, зручно й без надмірностей – це про інтерфейс архіватора 7-Zip.

PowerArchiver

Наступний претендент на кращий архіватор від розроблювачів ConeXware- PowerArchiver. По зручності він нічим не уступає архіватору 7-Zip, і навіть щось у нього залучає.

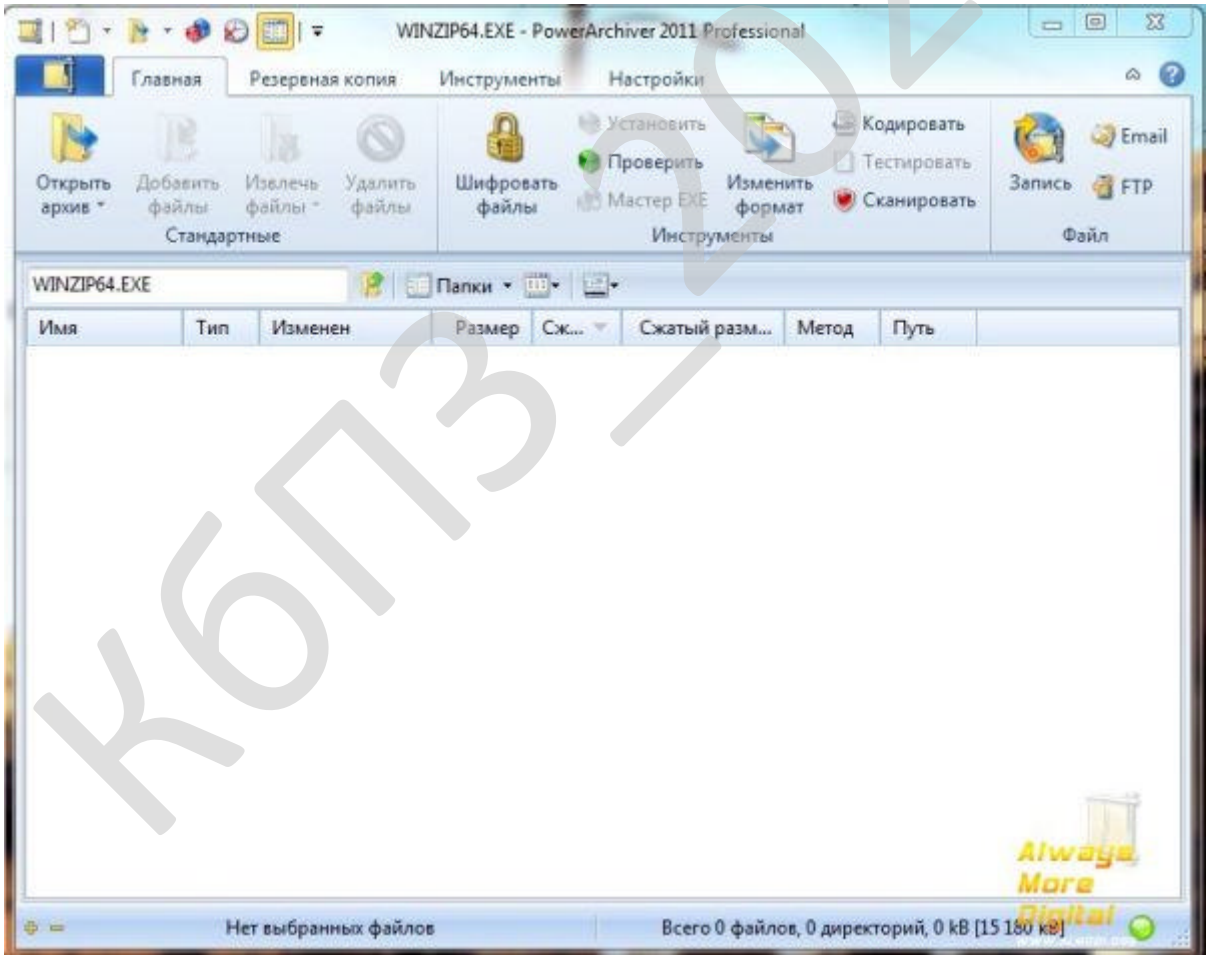

Рисунок 2.2 – Інтерфейс користувача PowerArchiver

Зовнішній вигляд можна настроїти на свій смак. У нас на вибір вісім тем оформлення, правда тут був замічений недолік з перекладом назв тим: перші чотири переведені правильно, а от наступні три теми, називаються, м'яко говорячи, дивно.

WinRAR

Думаю, наш третій учасник не має потреби в поданні. Це архіватор WinRAR**.**

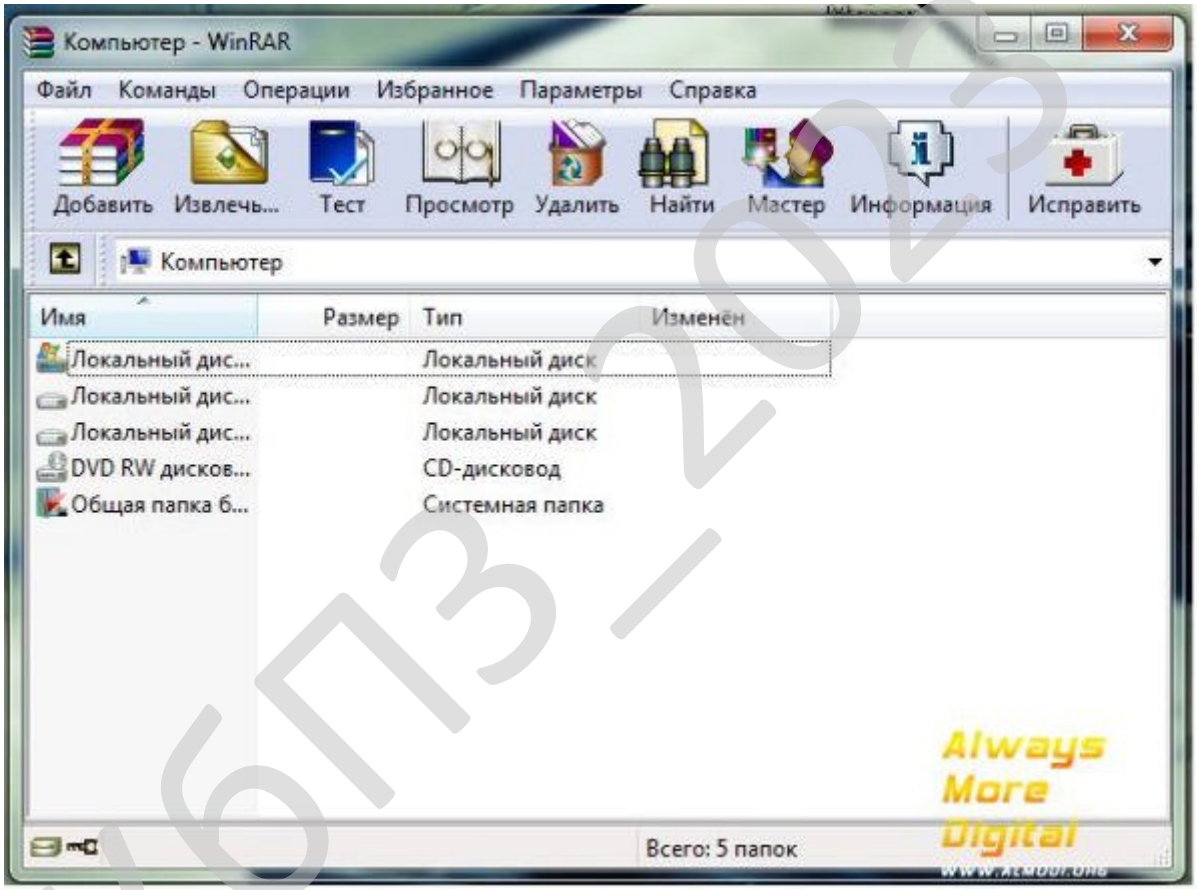

Рисунок 2.3 – Інтерфейс користувача WinRAR

Оформлення даної програми особливо нічим не виділяється, але адже Вам не для краси архіватор, вірно? А якщо Ви виключення, то на офіційному сайті є безліч різноманітних стилів оформлення, які можна скачати перейшовши по посиланню http://www.rarlab.com/themes.htm. Що стосується самого інтерфейсу,

те можна відразу сказати, що знайти програму-архіватор з великим набором функцій затяжко – отут можна навіть попередньо перевірити на віруси архів, і потім тільки приступати з розпакуванню, або, наприклад протестувати своє "залізо" на стабільність!

WinZip

І нарешті, останній учасник – WinZip від корпорації Corel, що, у свою чергу, купила в далекому 2006 році компанію WinZip Computing. Тут також мене відвідало дежавю, тому що теж простежується подібність оформлення з Microsoft Office Word 2007. Ribbon або Microsoft Fluent Interface – так називається зовнішнє оформлення, що саме й використовують PowerArchiver і WinZip, а також деякі інші програми середовища Windows. Тому й простежується схожість в оформленні. Ну й правильно! Навіщо міняти те, що добре працює?

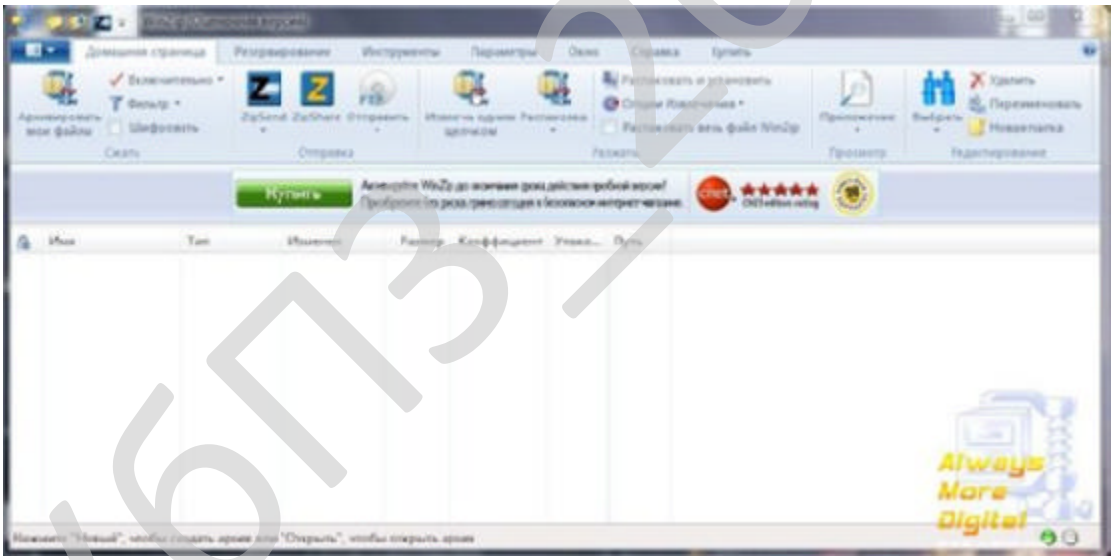

Рисунок 2.4 – Інтерфейс користувача WinZip

А якщо Ви хочете іншого оформлення, то на офіційному сайті є небагато тем. Тут присутні всі ті ж функції, що й у попередніх архіваторах, плюс убудована функція відправлення архіву по електронній пошті, підтримка протоколу FTP. Як і у випадку з PowerArchiver, та й з іншими архіваторами, інтерфейс зручний і простий в освоєнні. В WinZip є дві версії: Standard

(стандартна) і Professional (Професійна), а розрізняючи полягають у наявності резервного копіювання:

– Резервне копіювання важливих даних натисканням однієї кнопки.

– Планування резервного копіювання для автоматичного запуску без участі користувача.

– Запис резервних копій на компакт-диск або DVD або їхнє відправлення по електронній пошті.

– Передача резервних копій на зовнішній сервер за допомогою убудованого FTP-клієнта.

Ще варто сказати що всі програми мають російську локалізацію, так що працювати легше й приємніше.

Перейдемо до конкретних утиліт для розмежування доступу, з ціллю збереження конфіденційності даних.

Veritas Enterprise Vault

Veritas Enterprise Vault – це комплексне рішення для архівування інформації, яке допомагає організаціям керувати своїми даними та захищати їх. Він забезпечує безперебійне архівування електронних листів, файлів та інших неструктурованих даних, забезпечуючи дотримання нормативних вимог. Завдяки розширеним можливостям пошуку та пошуку він дозволяє користувачам швидко отримувати доступ до архівної інформації.

ZL Unified Archive

Детальна підтримка судових процесів для кожної частини еталонної моделі Electronic Discovery у проактивному та реактивному електронному виявленні.

MessageSolution Enterprise Archiver

Архівація всіх файлів MessageSolution Enterprise File Archive – це автоматизована система архівування на основі програмного забезпечення, яка працює всередині корпоративної мережі та підключається до наявних файлових серверів. Enterprise File Archive знаходить, вивантажує, очищає та зберігає файли, які відповідають політикам архівування організації. Керовані політикою функції

Enterprise File Archive дозволяють адміністратору файлового архіву легко налаштовувати параметри архівування та зберігання відповідно до унікальних цілей організації. Адміністратор файлового архіву може вибрати, які файли архівувати, як файли зберігаються, де зберігаються файли та скільки часу файли зберігаються.

IBM Archive and Essence Manager (AREMA)

IBM Archive and Essence Manager (AREMA) – це повністю інтегроване багатофункціональне рішення зі штучним інтелектом (ШІ), яке покращує гнучкість робочих процесів виробництва та розповсюдження мультимедійних файлів. AREMA організовує будь-який робочий процес виробництва трансляцій на основі файлів, забезпечуючи повний контроль над процесами виробництва мовника.

Atempo Digital Archive

ADA (Atempo-Digital Archive) – це програмне рішення для резервного копіювання, переміщення, міграції, синхронізації та архівування файлів, яке забезпечує постійну доступність і безпеку великих обсягів даних. ADA інтегрується в усі робочі процеси, які потребують продуктивності та відкритих форматів, звільняючи вас від блокування пам’яті.

C2C ArchiveOne

Керуйте електронними листами та архівуйте важливу кореспонденцію для тривалого зберігання та контролю інформації.

Digital Safe

Autonomy Zantaz – це високомасштабована та гнучка платформа архівування, яка дозволяє вашій організації отримувати та зберігати повідомлення електронної пошти, вкладення та файли таким чином, щоб оптимізувати зберігання та надати негайний доступ для виявлення та перегляду судами, регуляторними органами, адміністраторами та кінцевими клієнтами. користувачів. На базі IDOL EAS автоматизує обробку величезних обсягів інформації, динамічно виявляючи шаблони, зв’язки та значення всередині.

Veritas Enterprise Vault.cloud

Veritas Enterprise Vault.cloud – це хмарна служба архівування, яка допомагає регульованим організаціям ефективніше відповідати вимогам управління даними. і ефективніше здійснювати ділові або юридичні eDiscovery. Тисячі клієнтів використовували Enterprise Vault.cloud, щоб забезпечити успішну та сумісну міграцію на хмарну платформу додатків, таку як Microsoft Office 365 або Google G Suite Enterprise.

Metalogix Archive Manager for Files

Менеджер архівів – це програмне рішення для прозорого архівування файлів із збереженням існуючої структури дисків, папок і файлів. Він забезпечує повне керування версіями з можливістю отримати будь-яку з версій. Архівовані файли замінюються на файловому сервері ярликом, який займає набагато менше місця, але вони доступні для кінцевих користувачів, які можуть працювати з ними як зазвичай. Крім того, наша повнотекстова пошукова система шукає в архівах, застосовуючи різні критерії пошуку.

Proofpoint Intelligent Compliance Platform

Непросто розробити стратегію, щоб забезпечити дотримання корпоративних або нормативних вимог. Зростання даних турбує IT-відділи. І юридичні команди та відділи дотримання нормативних вимог намагаються зберегти, виявити та контролювати вміст у всіх каналах. Proofpoint пропонує сімейство продуктів і рішень Intelligent Compliance, які полегшують вам прийняття більш обґрунтованих рішень щодо відповідності, управління інформаційними ризиками та підвищення готовності до розслідування.

OpenText Content Manager

OpenText Content Manager, раніше Records Manager, – це корпоративна система керування корпоративним вмістом, розроблена для допомоги державним установам, регульованим галузям і глобальним організаціям керувати своїм бізнес-контентом від створення до вилучення. Незалежно від того, як ви створюєте свій вміст і співпрацюєте над ним, Content Manager дає вам

можливість використовувати точну, контекстну та повну інформацію протягом усього життєвого циклу. Управління вмістом у такий спосіб допоможе вам значно покращити послуги для ваших клієнтів, бізнес-рішення та розуміння майбутньої стратегії.

OpenText InfoArchive

OpenText InfoArchive is a modern archive solution and cloud-based service for compliant archiving of both structured and unstructured information highly-accessible, scalable and economical.

Microsoft Exchange

Створений для забезпечення безпеки та надійності корпоративного рівня, які потрібні компаніям, Microsoft Exchange забезпечує електронну пошту, календар і контакти на вашому ПК, телефоні та веб-браузері.

Commvault Complete Data Protection

Commvault Complete Data Protection допомагає забезпечити доступність даних і безперервність бізнесу в локальних і хмарних середовищах за допомогою єдиної розширюваної платформи. Він забезпечує просте, масштабоване, комплексне резервне копіювання, реплікацію та координацію аварійного відновлення для всіх сучасних робочих навантажень.

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 створення архівів у файловій системі NTFS з розмежуванням доступу.

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

Що таке архівування файлів? Архівування файлів – це своєчасний систематичний процес збору, зіставлення та переміщення застарілих, застарілих або рідко використовуваних даних із сервера чи операційної системи в окреме місце зберігання, сервер, стрічку чи навіть хмару. Усі заархівовані дані, як правило, стискаються за допомогою подібних або координуючих файлів і програм, які раніше працювали разом і спільно використовували історію. Крім того, до цього файлу буде включено будь-які записи та документацію щодо цих даних. Архівування файлів, як правило, виконується, щоб звільнити більше місця та зменшити «навантаження» на головний сервер або операційні системи організації та забезпечити швидку, безперебійну та ефективну роботу цих систем або зберегти важливі дані в безпеці для різних цілей зберігання. Архівування не слід плутати з резервним копіюванням файлів, коли дані копіюються з метою відновлення.

Призначення програмного забезпечення, розробленого в ході виконання магістерського проектування:

– Створення архівів у файловій системі NTFS з розмежуванням доступу.

– Захист від несанкціонованого доступу й розкриття конфіденційності архівованої інформації, що зберігається й обробляється на персональному комп'ютері або ноутбуці.

– Захист заархівованої інформації при переносі й зберіганні на знімних носіях.

– Розмежування прав користувачів на доступ до захищеної інформації з використанням надійної двофакторної автентифікації (володіння електронним ключем eToken і знання пароля).

Необхідність програмного забезпечення, розробленого в ході виконання магістерського проектування:

– При роботі на ноутбуці. Втрата або крадіжка ноутбука, несанкціоноване використання сторонніми особами.

– При роботі на персональному комп'ютері в офісі. Несанкціонований доступ до заархівованих даних по локальній мережі або неправомірне використання сторонніми особами під час відсутності користувача на робочому місці.

– Комп'ютер передається на сервісне обслуговування. Несанкціонований доступ до заархівованих даних під час проведення ремонтних і сервісних робіт внутрішньої IT-службою або зовнішньою сервісною компанією.

– Заархівована інформація конфіденційного характеру переноситься або пересилається на знімних носіях. Втрата або крадіжка носіїв.

– Необхідно забезпечити виконання вимог закону про персональні дані. Порушення конфіденційності персональних даних, які зберігаються й обробляються на персональних комп'ютерах в організації.

Відмінні риси програмного забезпечення, розробленого в ході виконання магістерського проектування

Створення архівів у файловій системі NTFS з розмежуванням доступу

Стиск даних (графічних зображень, відеозображень і звуку) – процедура їхнього перекодування, вироблена з метою зменшення їхнього обсягу. Застосовується для більше раціонального використання пристроїв зберігання й передачі даних.

Стиск заснований на усуненні надмірності інформації, що втримується у вихідних даних.

Надійний захист даних

Розмежування доступу, з ціллю збереження конфіденційності, розділів на жорстких дисках, томів на динамічних дисках, віртуальних дисків і знімних носіїв.

Захист системного розділу жорсткого диска

Системний розділ жорсткого диска містить дані, що представляють особливий інтерес для хакерів, конкурентів або інсайдерів. Наприклад, у системному розділі зберігаються облікові записи користувачів, логіни й паролі до різних інформаційних ресурсів, електронна пошта, ліцензійна інформація використовуваних програм і т.д. Зловмисники можуть одержати всі ці дані, аналізуючи тимчасові файли ОС, файли підкачування, файли-журнали додатків, дампи пам'яті, а також образ, що зберігається на диск при переході системи в «сплячий» режим.

Програмне забезпечення, розроблене в ході виконання магістерського проектування, на відміну від багатьох конкурентів, дозволяє захистити системний розділ, а також, що зберігається на ньому інформацію.

Завантаження операційної системи по електронному ключі eToken

Одержавши доступ до персонального комп'ютера, зловмисник або несумлінний співробітник може використовувати його для одержання доступу до закритих ресурсів (наприклад, до корпоративних серверів або платіжної системи користувача). Стандартні засоби авторизації операційної системи Microsoft Windows не можуть надійно обмежити завантаження й роботу в операційній системі. Використання електронних USB-ключів і смарт-карт eToken для автентифікації користувачів до завантаження ОС гарантує доступ до комп'ютера тільки довірених осіб.

Програмне забезпечення, розроблене в ході виконання магістерського проектування надає найбільш безпечну й надійну на сьогоднішній день процедуру підтвердження прав користувача – двофакторну автентифікацію – для доступу до даних необхідно не тільки наявність USB-токена, але й знання пароля до нього.

Прозоре розмежування доступу, з ціллю збереження конфіденційності

Операції початкового зашифрування або повного перешифрування для сучасних дисків великого обсягу можуть зажадати значного часу, що може створити певні незручності для користувача.

У програмному забезпеченні, розробленому в ході виконання магістерського проектування всі операції зашифрування, перешифрування й розшифрування проводяться у фоновому режимі. Під час виконання цих операцій диск повністю доступний для роботи, що дає можливість використовувати комп'ютер, не чекаючи закінчення процесу розмежування доступу, з ціллю збереження конфіденційності.

Відновлення доступу до зашифрованих дисків

Якщо Ваш електронний ключ, персональний комп'ютер або окремий диск із даними потрапили в чужі руки, Ви можете бути спокійні за схоронність і неприступність Ваших даних – ніхто, крім Вас не зможе одержати доступ до них в обхід розробленої системи.

У випадку втрати або поломки USB-ключа або смарт-карти в програмному забезпеченні, розробленому в ході виконання магістерського проектування, передбачена можливість резервного відновлення доступу до даних.

Заборона доступу по мережі до зашифрованих даних

Дані, що зберігаються на зашифрованих дисках персонального комп'ютера, доступні тільки адміністраторові безпеки й користувачам, що володіють електронними ключами eToken і зареєстрованими в програмному забезпеченні, розробленому в ході виконання магістерського проектування. Інші користувачі, включаючи системного адміністратора, не можуть одержати доступ до зашифрованих даних.

Необоротне видалення даних

У програмному забезпеченні, розробленому в ході виконання магістерського проектування реалізовані дві функції безпеки даних:

– необоротне видалення даних;

– переміщення файлу без можливості відновлення по вихідному шляху.

Додаткові особливості

– Захист даних від збоїв під час операцій розмежування доступу, з ціллю збереження конфіденційності, включаючи перебої електроживлення.

– Підтримка режиму енергозбереження для ноутбуків.

– Динамічний розподіл швидкості розмежування доступу, з ціллю збереження конфіденційності.

3.2 Розробка структурної схеми

Структурна схема наведена на рисунку 3.1 З неї ми бачимо, що розроблена система складається з наступних структурних блоків.

1. Файли, які необхідно зашифрувати.
2. Блок архівування/розархівування.
3. Блок розмежування доступу, з ціллю збереження конфіденційності, за допомогою алгоритму AES.
4. Блок розшифрування за допомогою алгоритму AES.
5. Розшифровані файли.
6. Блок вибору ключів розмежування доступу, з ціллю збереження конфіденційності.
7. Генератор ключів розмежування доступу, з ціллю збереження конфіденційності.
8. Генератор псевдовипадкових послідовностей.

Основним блоками системи є:

– Блок архівування/розархівування.

– Блок розмежування доступу, з ціллю збереження конфіденційності, з використанням алгоритму AES.

Розглянемо їх більш детальніше.

Блок архівування/розархівування

Цей блок складається з наступних підблоків:

– Блок вибору алгоритмів архівування.

– Блок вибору формату файлу архівування.

– Блок вибору формату файлу розархівування.

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Розмежування доступу, з ціллю збереження конфіденційності

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.

Визначення й допоміжні процедури

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

– 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 розмежування доступу, з ціллю збереження конфіденційності.

– 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 розмежування доступу, з ціллю збереження конфіденційності, 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.

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.

На початку розмежування доступу, з ціллю збереження конфіденційності, input копіюється в масив State за правилом:

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

для $0 \leq r < 4$ й $0 \leq c < Nb$.

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 $0 \leq r < 4$ й $0 \leq c < Nb$.

Операція розмежування доступу, з ціллю збереження конфіденційності, описана в псевдокоді. Окремі трансформації SubBytes(), ShiftRows(), MixColumns(), і AddRoundKey() – обробляють State. Масив $w[]$ – містить key schedule.

Функція розмежування доступу, з ціллю збереження конфіденційності, 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()

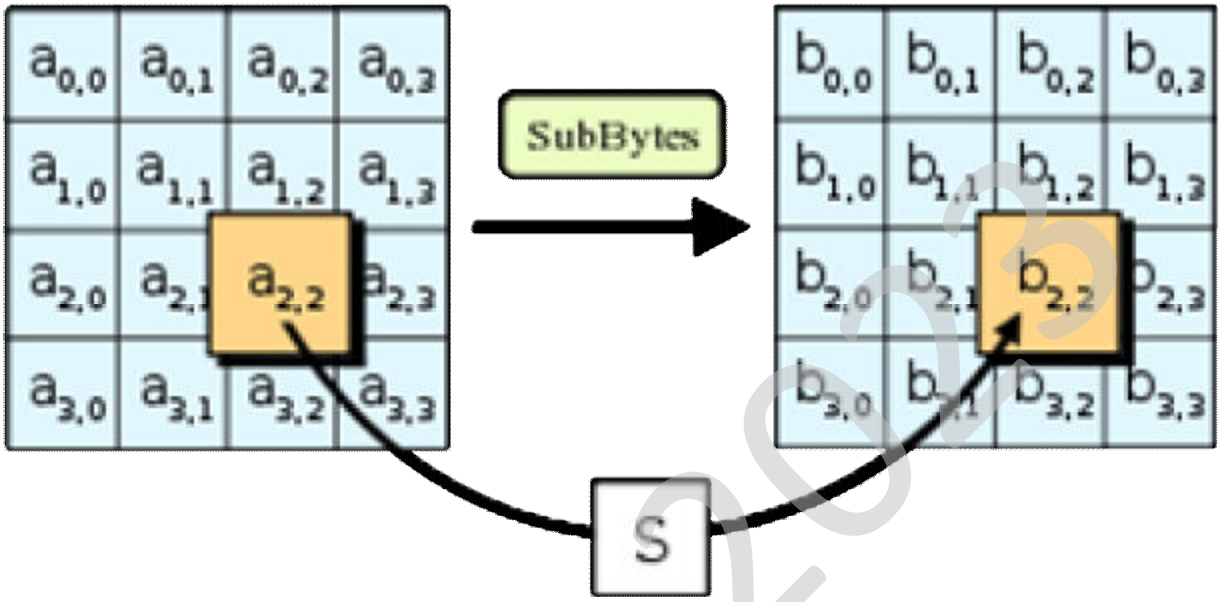

Рисунок 3.2 – Опис процедури SubBytes()

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 розмежування доступу, з ціллю збереження конфіденційності. 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()

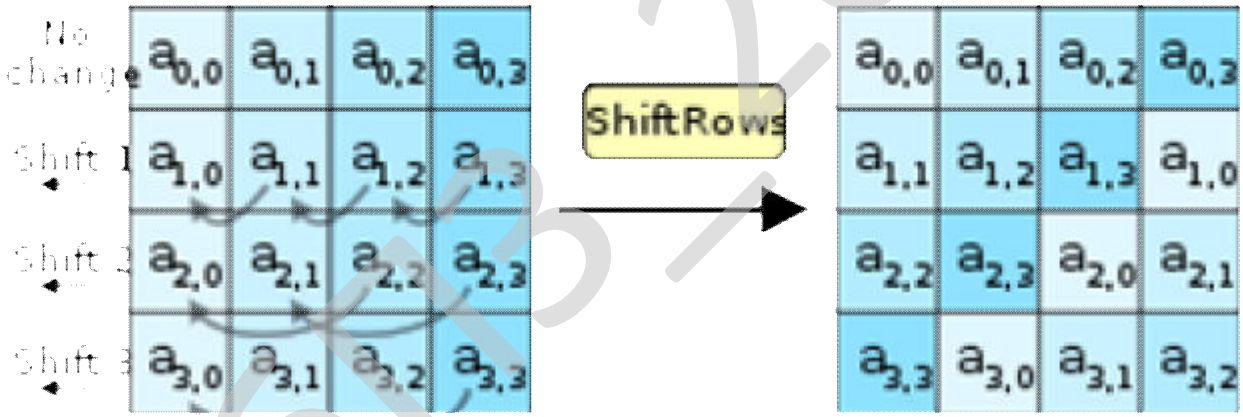

Рисунок 3.3 – Опис процедури ShiftRows()

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()

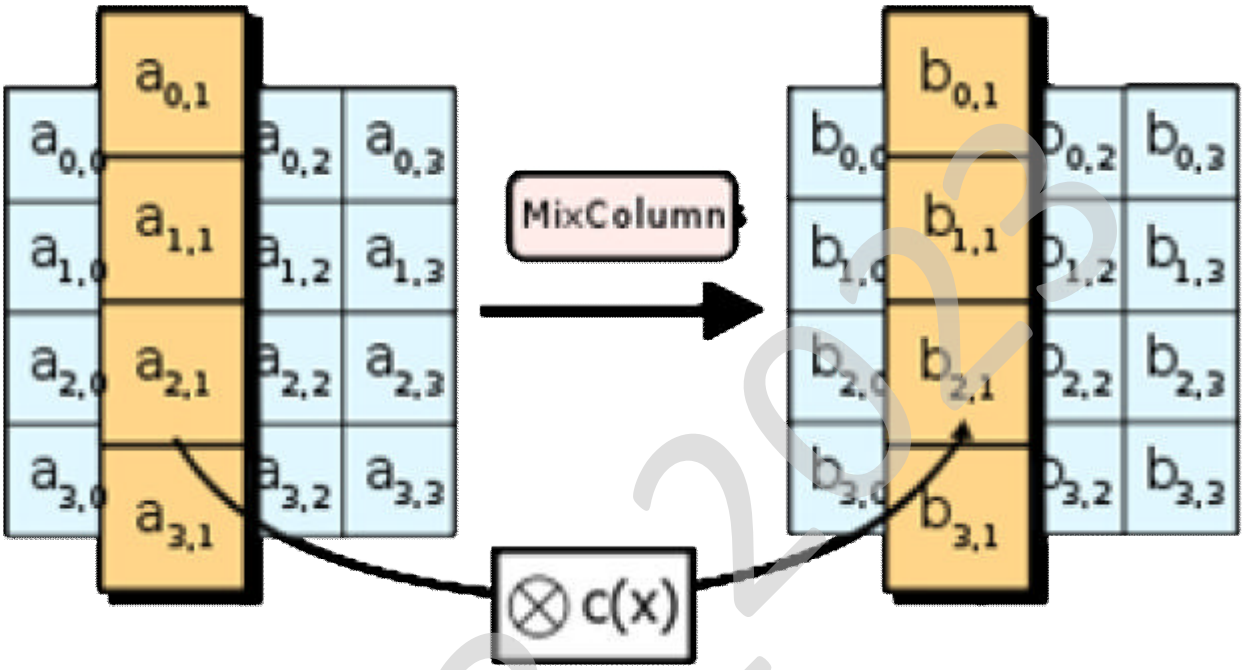

Рисунок 3.4 – Опис процедури MixColumns()

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 (⊕).

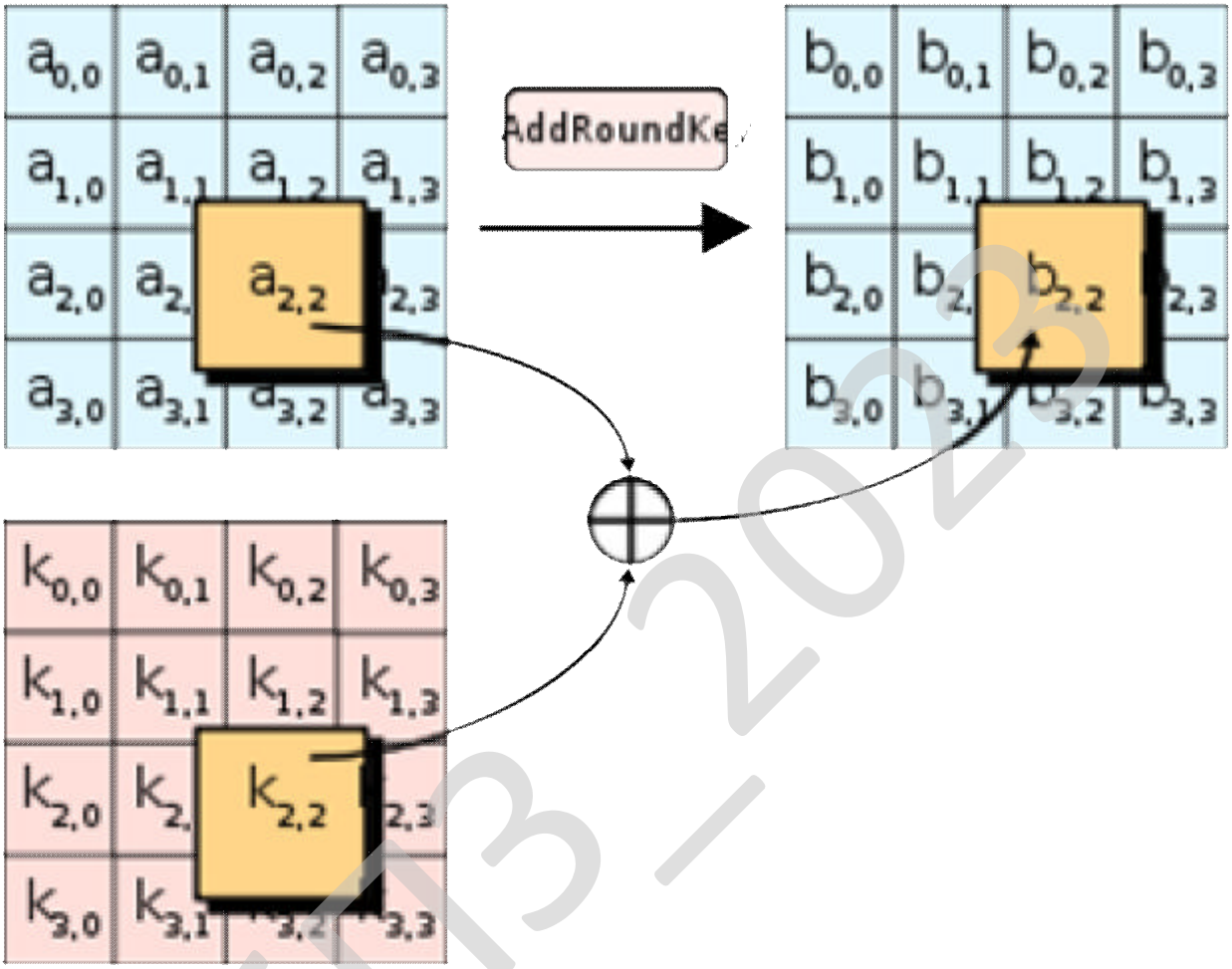

Рисунок 3.5 – Опис процедури AddRoundKey()

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 .

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
```

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 архівування/розархівування.
3. Блок роботи генератора псевдовипадкових чисел.
4. Блок формування ключа AES.
5. Блок зчитування та перевірки на легітимність паролю.
6. Блок підрахунку спроб введення некоректного паролю.
7. Блок розмежування доступу, з ціллю збереження конфіденційності даних.

8. Блок дешифрування даних.
9. Блок гарантованого знищення інформації.
10. Блок допомоги та інформації про програму.

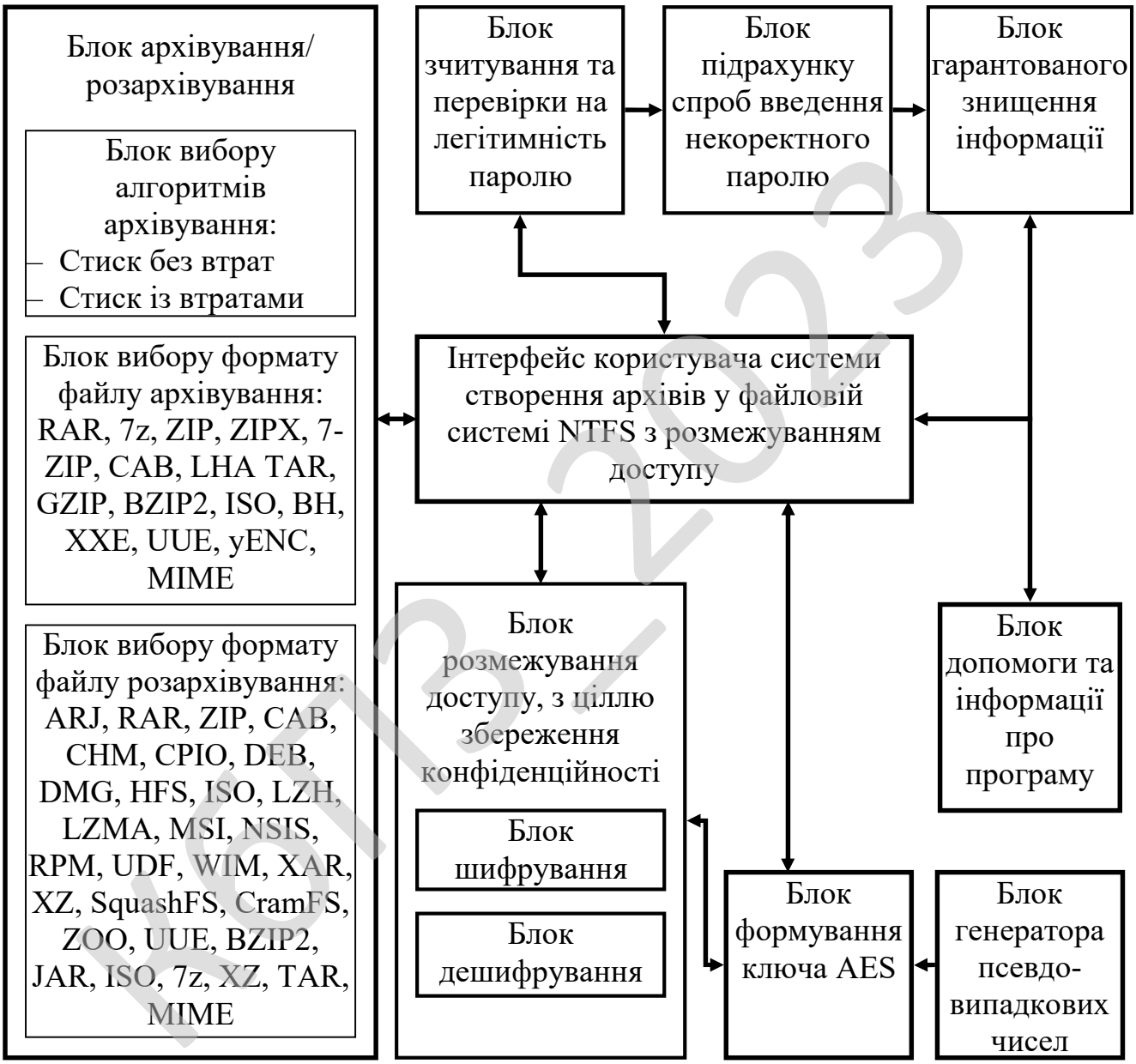

Рисунок 3.6 – Функціональна схема програмного забезпечення

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 архівування/розархівування

Він складається з наступних підблоків:

– Блок вибору алгоритмів архівування.

– Блок вибору формату файлу архівування.

– Блок вибору формату файлу розархівування.

Блок вибору алгоритмів архівування включає в себе:

– Стиск без втрат: перетворення Барроуза-Уілера; перетворення Шиндлера; алгоритм DEFLATE; дельта-кодування; ентропійне кодування; інкрементне кодування; алгоритми Лемпеля-Зіва; LZ77; LZ 77-PM; LZFG; LZFG-PM; LZP; LZBW; LZSS; LZB; LZH; LZRW1; LZ78; LZW; LZW-PM; LZMW; LZMA; LZO; PPM; RLE; SEQUITUR; вейвлет; алгоритм Шеннона-Фано; алгоритм Хаффмана; адаптивне кодування Хаффмана; усічене двійкове кодування; арифметичне кодування; адаптивне арифметичне кодування; кодування відстаней; ентропійне кодування; унарне кодування; кодування Фібоначчі; кодування Голомба; кодування Райса; кодування Еліаса.

– Стиск із втратами: JPEG; лінійне кодування, що пророкує; а-закон; мю-закон; фрактальний стиск; кодування, що трансформує; векторна квантизація; вейвлетний стиск.

Блок вибору формату файлу архівування включає в себе наступні формати: RAR, 7z, ZIP, ZIPX, 7-ZIP, CAB, LHA TAR, GZIP, BZIP2, ISO, BH, XXE, UUE, yENC, MIME

Блок вибору формату файлу розархівування включає в себе наступні формати: ARJ, RAR, ZIP, CAB, CHM, CPIO, DEB, DMG, HFS, ISO, LZH, LZMA, MSI, NSIS, RPM, UDF, WIM, XAR, XZ, SquashFS, CramFS, ZOO, UUE, BZIP2, JAR, ISO, 7z, XZ, TAR, MIME

Блок роботи генератора псевдовипадкових чисел

Один з найбільш сильних генераторів псевдовипадкових чисел описаний в ANSI X9.17. Серед додатків, що використовують цю технологію, є додатки фінансової безпеки й PGP.

Для розмежування доступу, з ціллю збереження конфіденційності, тут використовується потрійний DES. Генератор ANSI X9.17 складається з наступних частин:

1. **Вхід:** генератором управляють два псевдовипадкових входи. Один є 64-бітним поданням поточної дати й часу, які змінюються щоразу при створенні числа. Інший – 64-бітне початкове значення, що ініціюється деяким довільним значенням і змінюється в ході генерації послідовності псевдовипадкових чисел.

2. **Ключі:** генератор використовує три модулі потрійного DES. Всі три використовують ту саму пару 56-бітних ключів, що повинні триматися в секреті й застосовуватися тільки для генерації псевдовипадкового числа.

3. **Вихід:** вихід складається з 64-бітного псевдовипадкового числа й 64-бітного значення, що буде використовуватися як початкове значення при створенні наступного числа.

Якщо:

DT_i – значення дати й часу на початок і-тої стадії генерації.

V_i – початкове значення для і-тої стадії генерації.

R_i – псевдовипадкове число, створене на і-тій стадії генерації.

K_1, K_2 – ключі, що використовуються на кожній стадії.

Тоді:

$$R_i = EDE_{K1,K2} [EDE_{K1,K2} [DT_i] \oplus V_i]$$

$$V_{i+1} = EDE_{K1,K2} [EDE_{K1,K2} [DT_i] R_i]$$

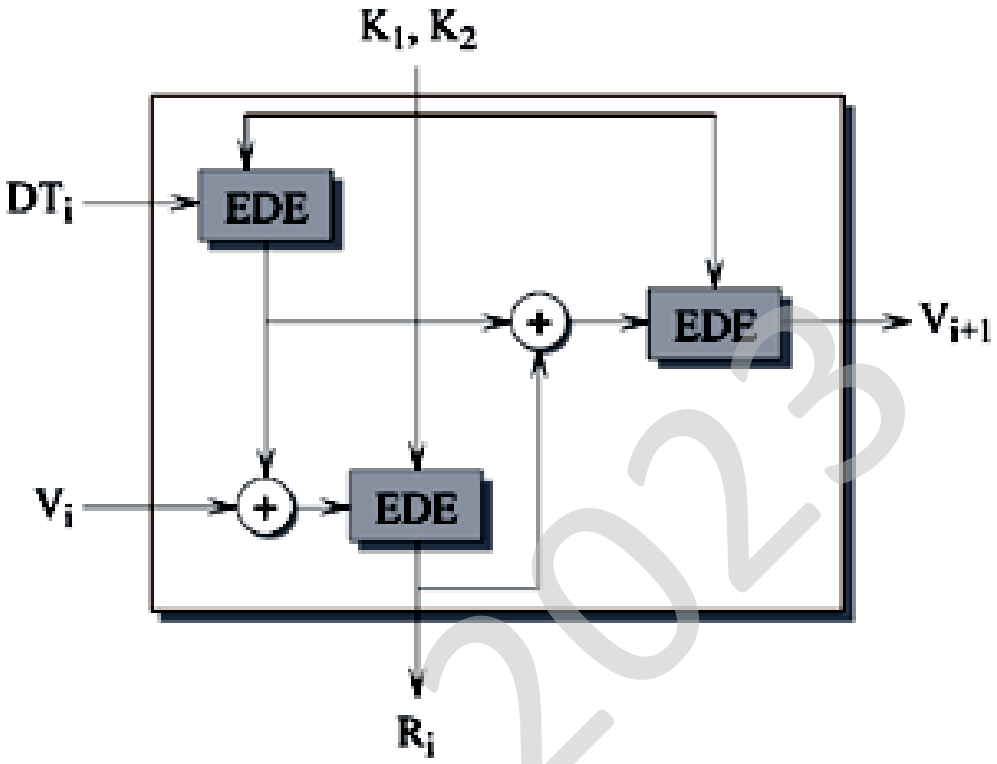

Рисунок 3.7 – Генератор псевдовипадкових чисел ANSI X9.17

Схема включає використання 112-бітного ключа й трьох EDE-шифрувань. На вхід подаються два псевдовипадкових значення: значення дати і часу та початкове значення чергової ітерації, на виході створюються початкове значення для наступної ітерації й чергове псевдовипадкове значення. Навіть якщо псевдовипадкове число R_i буде скомпрометовано, обчислити V_{i+1} з R_i неможливо і, отже, неможливо обчислити й псевдовипадкове значення R_{i+1}, тому що для одержання V_{i+1} додатково виконуються три операції EDE.

Блок формування ключа AES

У цьому боці генеруються ключі розмежування доступу, з ціллю збереження конфіденційності, та сеансові ключі.

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 розмежування доступу, з ціллю збереження конфіденційності даних

Цей блок шифрує дані використовуючи алгоритм AES. Детальна робота цього алгоритму розписана у пункті 3.2.

Блок дешифрування даних

Цей блок розшифрує дані використовуючи алгоритм AES. Детальна робота цього алгоритму розписана у пункті 3.2.

Блок гарантованого знищення інформації

Цей блок призначений для гарантованого знищення інформації, при неправильному введені паролю. За основу був вибраний алгоритм Гутмана.

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.

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 3.8. Першим процесом, який запускається у розробленій, під час виконання магістерського проектування, системі є процес виведення головного вікна. Він взаємодіє з наступними процесами:

– Процес генерації ключів шифрування, який, у свою чергу, взаємодіє з процесом збереження ключів шифрування.

– Процес вибору параметрів.

– Процес виведення списку файлів та папок, який, у свою чергу, взаємодіє з процесом вибору файлів.

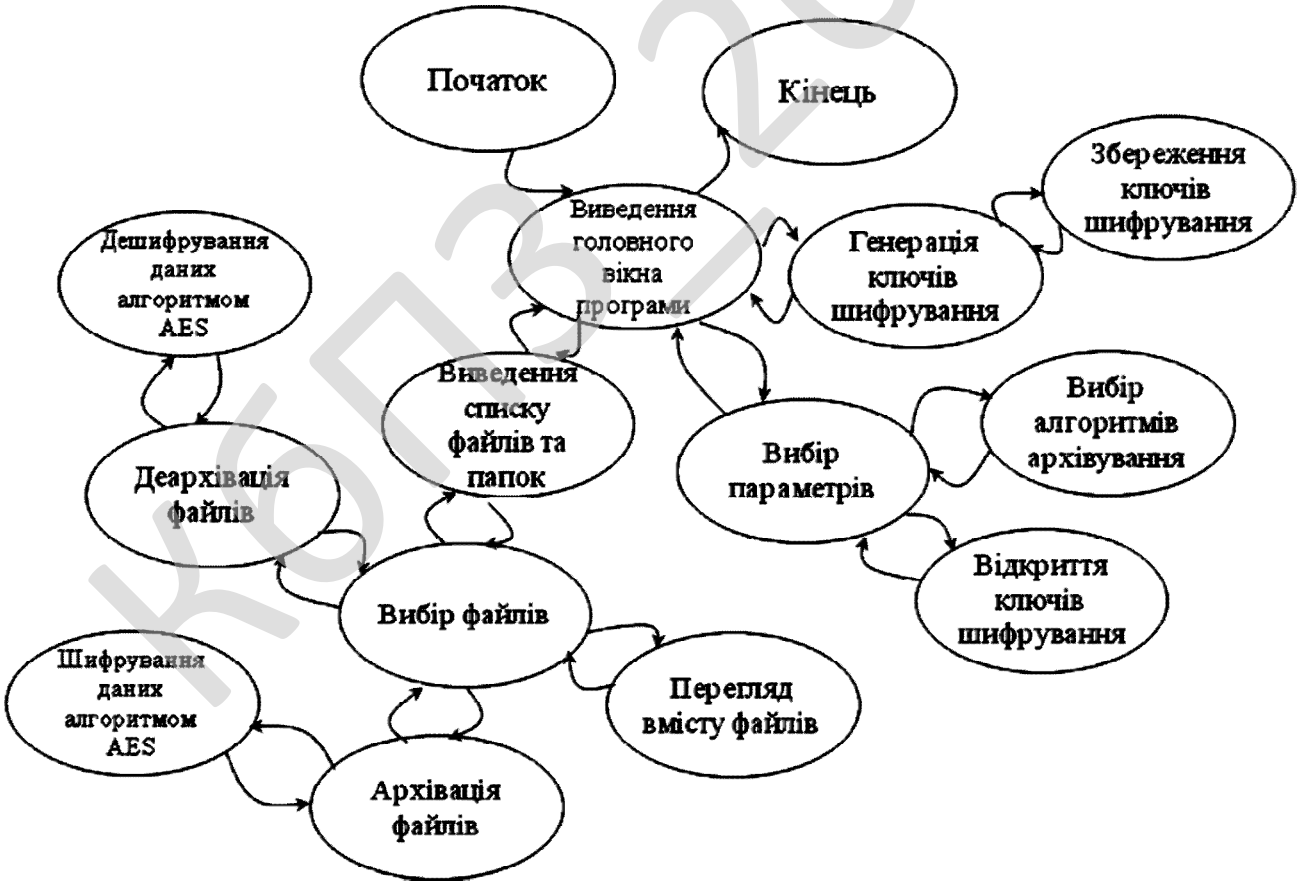

Рисунок 3.8 – Діаграма взаємодії процесів

Процес вибору параметрів взаємодіє з наступними процесами:

– Процес вибору алгоритму архівування.

– Процес відкриття ключів шифрування.

Процес вибору файлів взаємодіє з наступними процесами:

– Процес перегляду вмісту файлу.

– Процес архівування файлів, який, у свою чергу, взаємодіє з процесом шифрування даних алгоритмом AES.

– Процес деархівування файлів, який, у свою чергу, взаємодіє з процесом дешифрування даних алгоритмом AES.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо покроково алгоритм роботи основної програми, блок-схема якої наведена на рисунку 4.1.

Крок 1. Виведення основного вікна програми.

Крок 2. Виведення файлів та папок.

Крок 3. Вибір та виділення файлів.

Крок 4. Якщо потрібно переглянути файли – крок 5, інакше – крок 6.

Крок 5. Перегляд вмісту файлів.

Крок 6. Якщо потрібно створити ключі шифрування – крок 7, інакше – крок 9.

Крок 7. Генерація ключів AES.

Крок 8. Збереження згенерованих ключів.

Крок 9. Якщо потрібно вибрати алгоритм архівації – крок 10, інакше – крок 11.

Крок 10. Вибір алгоритму архівації зі списку.

Крок 11. Запит на архівацію. Якщо потрібно архівувати – крок 12, інакше – крок 13.

Крок 12. Архівування вибраних файлів.

Крок 13. Запит на деархівацію. Якщо потрібно деархівувати – крок 14, інакше – крок 15.

Крок 14. Деархівація вибраних файлів.

Крок 15. Запит на вихід із програми. Якщо «так» – завершення роботи програми, інакше – крок 2.

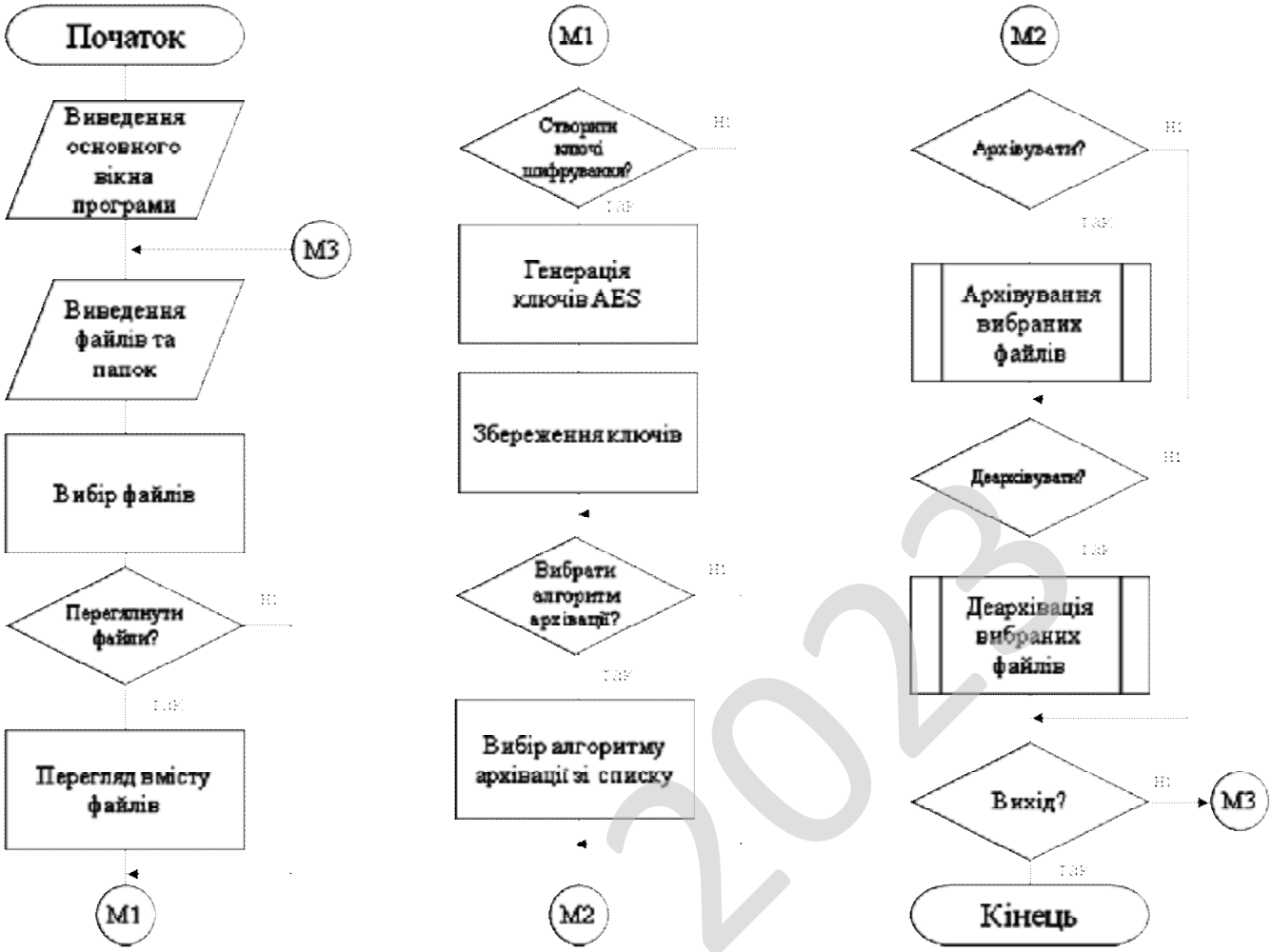

Рисунок 4.1 – Блок-схема основної програми

Наведемо частину коду основної програми, призначеного для 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;
```

Для дешифрування використовується наступний код:

```
  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;
```

Розглянемо більш детально процеси архівації та деархівації.

На рисунку 4.2 наведено блок-схему підпрограми архівації у файловій системі NTFS з розмежуванням доступу. Розглянемо покроково її роботу.

Крок 1. Введення ключа шифрування для розмежування доступу.

Крок 2. Пошук надмірностей в даних.

Крок 3. Формування словника для архіватора.

Крок 4. Підстановка даних по словнику.

Крок 5. Формування архіву.

Крок 6. Розбиття архіву на блоки для шифрування.

Крок 7. Розрахунок раундових ключів з ключа шифрування AES.

Крок 8. Ініціалізація S-box'ів.

Крок 9. Вибір раундового ключа.

Крок 10. Трансформація раундового ключа.

Крок 11. Поки не оброблені всі блоки, виконувати кроки 12-16. Потім – крок 17.

Крок 12. Підстановка байтів.

Крок 13. Циклічний зсув рядків.

Крок 14. Змішування даних у стовпцях.

Крок 15. Вибір раундового ключа.

Крок 16. Трансформація раундового ключа.

Крок 17. Об'єднання блоків та запис на диск зашифрованого архіву.

Крок 18. Завершення роботи підпрограми.

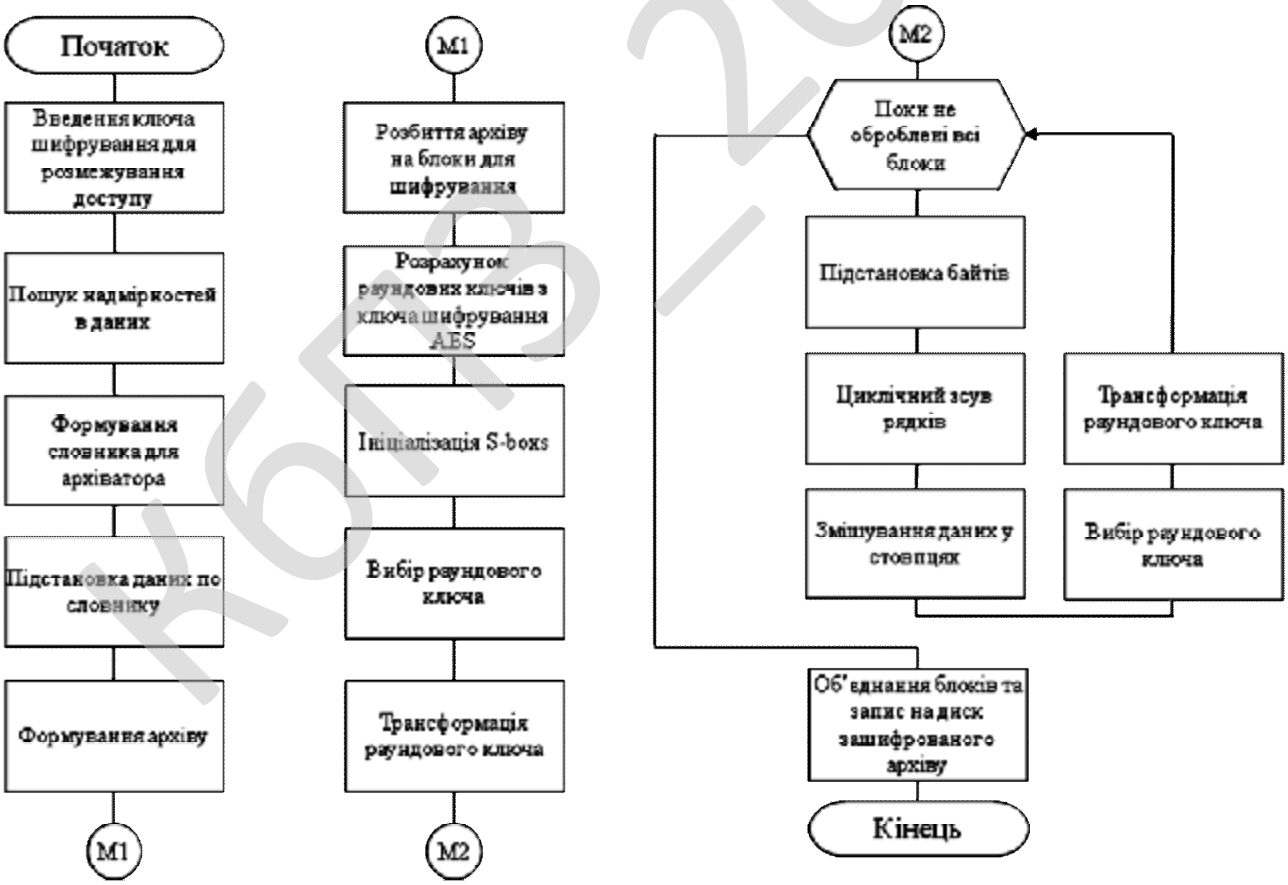

Рисунок 4.2 – Блок-схема підпрограми архівації у файловій системі NTFS з розмежуванням доступу

Наведемо частину коду, яка використовується для обмеження прав доступу шляхом шифрування файлів.

Ключі AES TAESKey128, TAESKey192 або TAESKey256 являють собою масиви із 15-31 змінних типу byte; розширені ключі TAESExpandedKey128, TAESExpandedKey192 або TAESExpandedKey256 – масиви довжиною 43-63 із змінних типу longword.

Розширення ключів за допомогою процедури `ExpandAESKeyForEncryption` відбувається наступним чином:

```
  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;
```

Далі, після ініціаліалізації

```
  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];
```

відбуваються раунди шифрування за допомогою процедури `EncryptAESStreamECB` (наведено приклад одного із раундів):

```
  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];
```

Останній раунд перетворень має вигляд:

```
  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];
```

Завершується процедура шифрування для обмеження прав доступу наступним чином:

```
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
```

На рисунку 4.3 наведена блок-схема підпрограми деархівації у файловій системі NTFS з розмежуванням доступу. Розглянемо покроково її роботу.

Крок 1. Введення ключа дешифрування для розмежування доступу.

Крок 2. Розбиття архіву для розшифровування на блоки.

Крок 3. Розрахунок раундових ключів з ключа шифрування AES.

Крок 4. Ініціалізація S-box'ів.

Крок 5. Вибір раундового ключа.

Крок 6. Трансформація раундового ключа.

Крок 7. Поки не оброблені всі блоки, виконувати кроки 8-12, потім – крок 13.

Крок 8. Зворотна підстановка байтів.

Крок 9. Зворотний циклічний зсув рядків.

Крок 10. Зворотне перемішування даних у стовпцях.

Крок 11. Вибір раундового ключа.

Крок 12. Трансформація раундового ключа.

Крок 13. Об’єднання блоків та формування розшифрованих даних.

Крок 14. Читання словника архіватора.

Крок 15. Підстановка даних по словнику.

Крок 16. Формування звіту деархівації.

Крок 17. Запис на диск деархівованих файлів.

Крок 18. Виведення звіту деархівації на екран.

Крок 19. Завершення роботи підпрограми.

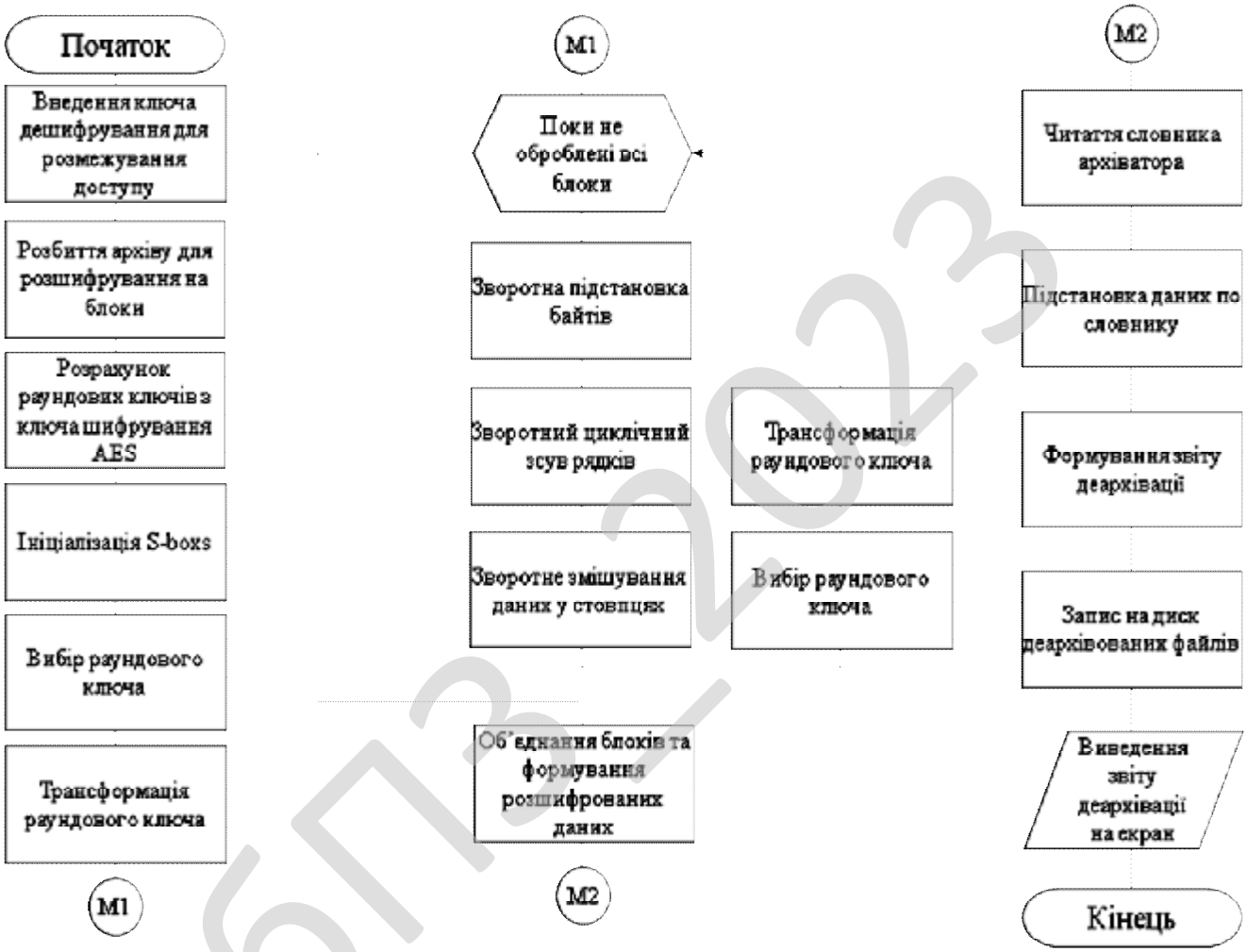

Рисунок 4.3 – Блок-схема підпрограми деархівації у файловій системі NTFS з розмежуванням доступу

Наведемо частину коду, яка використовується для доступу до зашифрованих файлів.

Розширення ключів для деархівації:

```
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;
```

Один із раундів дешифрування:

```
  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];
```

Останній із раундів дешифрування також відрізняється від основних незначним чином.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі

універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

$$Tag=H_{K1}(M)\oplus F_{K2}(Nonce)$$

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

Вибір блокового шифру

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16(тому що AES працює з 16-байтними блоками).

KDF – функція створення ключа

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

PDF: функція створення псевдовипадкового числа

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

Генерація UMAC-тегів

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

UMAC-32 UMAC-64 UMAC-96 UMAC-128

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 (K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 (K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 (K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

Універсальна функція гешування(UHASH)

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і, нарешті, до результату застосовується L3-HASH . Якщо при цьому довжина вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

Універсальна функція

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї

функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

L1-hash – перший етап

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

L2-hash – другий етап

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

L3-hash – третій етап

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо інтерфейс програми. На рисунку 5.1 зображено головне вікно програми. У вікні виводиться список файлів для архвівації/деархівації із обмеженням прав доступу. У верхній частині розташований рядок вводу адреси.

Керування роботою програми реалізується за допомогою кнопок на бічній панелі. За їх допомогою можна визначити місце збереження чи відкриття файлів, розпочати архвівацію/деархівацію, провести генерацію ключів шифрування, встановити параметри архівації та шифрування.

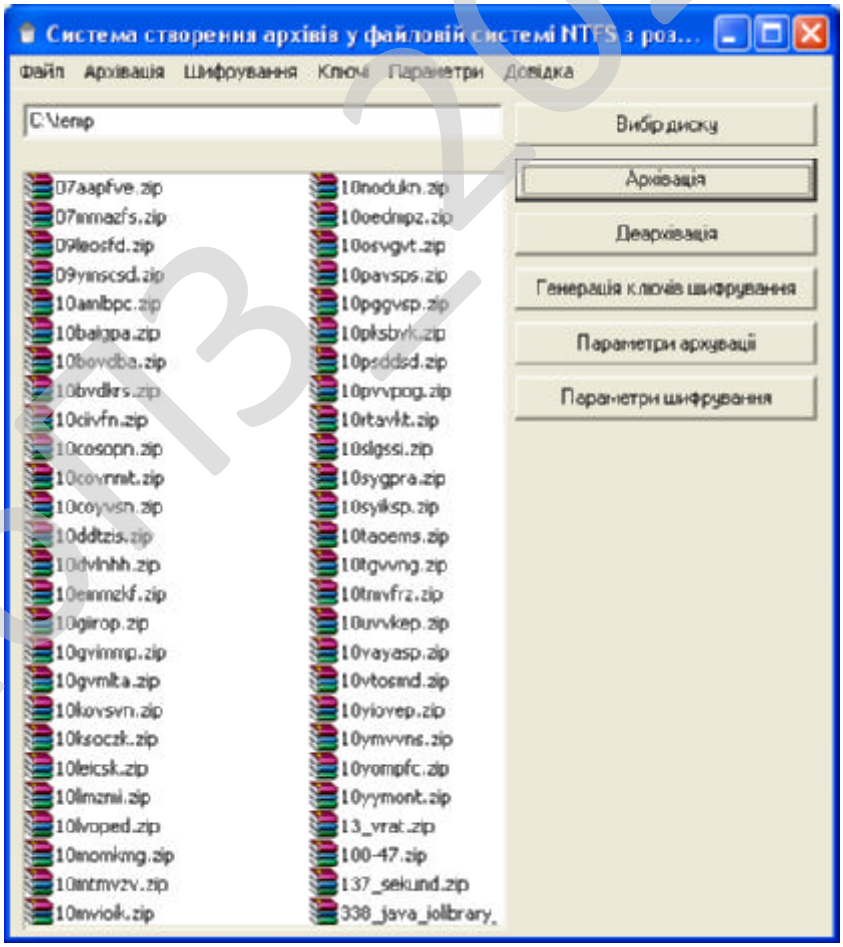

Рисунок 5.1 – Головне вікно програми

Доступ до основних можливостей програми також можливий через пункти меню:

– Файл.

– Архівація.

– Шифрування.

– Ключі.

– Параметри.

– Довідка.

В меню Файл вибирається шлях до файлів, які підлягають обробці.

За допомогою меню Архівація також відбувається вибір формату файлу для архівування/розархівування.

Меню Ключі крім генерації ключів шифрування дає можливість збереження цих ключів.

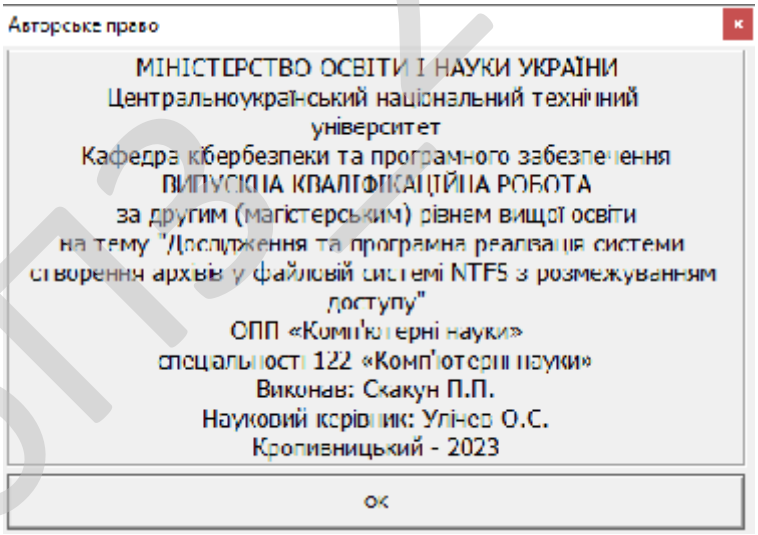

Рисунок 5.2 – Вікно довідки

На рисунку 5.2 зображено вікно довідки про програму, яке містить інформацію щодо теми магістерського проекту, керівника та розробника, дату та місце виконання.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Метою розробки є дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Об'єктом дослідження є процес створення архівів у файловій системі NTFS з розмежуванням доступу.

Предметом дослідження є методи створення архівів у файловій системі NTFS з розмежуванням доступу.

Методи дослідження базуються на методах теорії інформації та кодування, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод створення архівів у файловій системі NTFS з розмежуванням доступу.

– Розроблено вітчизняний продукт створення архівів у файловій системі NTFS з розмежуванням доступу, 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 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу. 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	58
3. Запланований термін розробки, днів	Fpq	48 (2 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
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
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	55
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 А – коефіцієнт Боема, А=2,45;

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

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

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

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

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

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

$$T_{уточн}=T_{ном}\, ПV_j, \tag{7.3}$$

де $ПV$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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:

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

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot55=93 \text{ люд/день}$$

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	93	Ф 7.1-7.4
Впровадження	13	Д13
Всього	134	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{134 \cdot 1}{48\text{-}5} = 3{,}1 \text{ ставки}$$

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	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	44,58

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{45 \cdot 2}{1,2} = 75 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=75/(48·8)=0,2 ставки

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	23000	46000
Продакт-менеджер	0,5	16900	16900
Інженер-програміст	3,1	20500	127100
Інженер-електронщик	0,2	15000	6000
Інженер-системотехнік	0,2	17000	6800
Адміністратор мережі	0,2	18000	7200
Системний програміст	0,2	17000	6800
Дизайнер WEB	0,2	18000	7200
Інженер-верстальник	0,2	15000	6000
Бухгалтер-економіст	0,2	19000	7600
Всього за період розробки	$R_{сп}$=6	-	$Ф_{роб}$=237600

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рq}}, \tag{7.8}$$

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

$$З_{сд} = \frac{237600}{6 \cdot 48} = 825 \; грн.$$

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

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

де Цм – ціна меблів для одного робочого місця, грн.

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за комерційною пропозицією Інтернет магазину Компбест за 14.10.23 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	Intel Core i5-8500T (6 ядер по 2.1 – 3.5 GHz) 9 MB Smart Cache	-
Системна плата	Menlo 843F, чипсет Intel H370, 2 x DDR DIMM, M.2 socket 1, 2230 type for WLAN M.2 socket 3, 2280 type for SSD, 4x USB 3.1 4x USB 2.0, 4x Audio, 1x LAN (RJ-45), 1 COM-порт, 1x HDMI, 1x VGA	-
Жорсткий диск	SSD M.2 2280 240GB Apacer	-
Оперативна пам’ять	DDR4 8GB 2400 MHz	-
Відеокарта	Intel UHD Graphics 630	-
DVD-привод	DVD-RW	-
Корпус	HP Slim 290-p0001ng Desktop	-
інше	Клавіатура, мишка	-
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

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

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608

Продовження таблиці 7.8

1	2	3	4
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля взята по даним з автосалону «Авто-РІА», джерело https://auto.ria.com/uk/auto_ford_focus_ 33565425.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

де $N_е$ – Кількість екземплярів програм, шт.

$$З_о = 825 \cdot 134/58 = 1906 \text{ грн}$$

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

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

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 1906 \cdot 10 \cdot 0{,}01 = 191 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

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

де H_c – відрахування на соціальні потреби, %

$$C_{оц} = 0{,}01 \cdot 22(1906+191) = 461 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

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

де $H_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1906 \cdot 15 \cdot 0{,}01 = 286 \text{ грн}$$

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

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

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 20):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,7 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 20 \cdot 23{,}7 = 474 \text{ грн.}$$

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

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

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.

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

$$З_{М} = (80 + 474 + 1702)/58=39 \text{ грн.}$$

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, %

$$О_п = 1906 \cdot 15 \cdot 0{,}01=286 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 58 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.

$$А_м = 174246 \cdot 2/(58 \cdot 12)=501 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$С_п = 1906+191+461+286+39+286+501=3670 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

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

де P_c – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 3670 = 2018{,}5 \text{ грн.}$$

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	$З_о$	1906
2. Додаткова зарплата виконавців	$З_д$	191
3. Відрахування на соціальні потреби	$С_{оц}$	461
4. Загальногосподарські витрати	$Г_{осп}$	286
5. Витрати на матеріали	$З_М$	39
6. Освоєння нових операційних систем, мов програмування	$О_п$	286
7. Амортизація основних фондів	$А_м$	501
8. Повна собівартість програмного забезпечення	$С_п$	3670
9. Плановий прибуток	$П_р$	2018,5
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5688,5
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1137,7
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	6826,2

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	–	6826
Всього капітальних витрат	–	6826

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	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	36905	23619
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	3413
Всього витрат за рік	$І$	36905	27032

Витрати на технічне обслуговування:

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

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*,

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 250 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 250 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 36905 \text{ грн.}$$

до

$$З_{р\,нов} = 160 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 23619 \text{ грн.}$$

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

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

Витрати на електроенергію при впровадженні нової системи не змінюються.

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	50	–	6826	–	3413
Всього відрахувань	-	–	6826	–	3413

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.

$$Ев = (5688 - 3670) \cdot 58 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 88003 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{1769406}{(5688 - 3670) \cdot 58 \cdot 12 / 2} = 2,52 \; років$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

$$E_{сп} = (36905\text{-}27032)\text{-}0,5 \cdot 6826 = 6460 \text{ грн.}$$

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

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

$$T_{сп} = \frac{6826}{36905 - 27032} = 0,7 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	58
2. Повна собівартість розробленої програми	Грн.	3670
3. Ціна розробленої програми	Грн.	5688
4. Плановий прибуток від реалізації розробленої програми	Грн.	2018
5. Рентабельність програмної продукції	%	55
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	117044
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	88003
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,52
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6826
11.Величина економічного ефекту у користувача програмної продукції	Грн.	6464
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,7

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

Характерною ознакою сучасного науково-технічного прогресу практично у всіх сферах діяльності людини є широке застосування комп'ютерних технологій, заснованих на використанні електронно-обчислювальних машин (ЕОМ). Сьогодення, а тим більше, майбутнє, вже важко уявити без комп'ютерів та іншої електронної техніки. Адже саме завдяки їм стала можливою швидка переробка величезних обсягів інформації, проведення необхідних розрахунків, виконання різних видів робіт, пов'язаних обробкою текстових та ілюстраційних зображень, організація оперативного отримання та передачі інформації, збереження її значних обсягів електронним способом.

Стрімке впровадження комп'ютерів не тільки в сфері управління виробництвом, в банківській систему, бізнесі, системі освіти, але також на транспорті, сфері обслуговування призвело до того, що десятки мільйонів людей у всьому світі виявились втягнутими у взаємодію людини з комп'ютером. Природно виникає запитання: настільки безпечною є ця взаємодія для людини? Адже відома аксіома про те, що будь-яка взаємодія людини та засобів праці двостороння.

Техніка безпеки – це система правил і заходів, які допомагають запобігти травмам, хворобам і аваріям на робочому місці або в повсякденному житті. Знання техніки безпеки дуже важливе, бо воно рятує життя і здоров’я людей.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [4], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час

роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

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

Ці шкідливі фактори можуть привести до професійних захворювань.

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

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

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

8.2. Шкідливі і небезпечні фактори при роботі з комп’ютером

Можна виділити наступні основні фактори, що впливають на стан здоров'я людей, які працюють за комп'ютером:

– сидяче положення на протязі тривалого періоду;

– вплив електромагнітного випромінювання монітора;

– втома очей, навантаження на зір;

– перевантаження суглобів кистей;

– стрес при втраті інформації.

У кожному з цих випадків ступінь ризику прямо пропорційний часу, що проводиться за комп’ютером і поблизу нього.В сучасних умовах взаємодія людини з технікою значно ускладнилась, що вимагає комплексного підходу, який передбачає розгляд людини, технічних засобів праці та виробничого середовища, як взаємозв'язаних елементів єдиної системи. Все вищесказане в повній мірі відноситься й до системи «людина–комп'ютер–середовище».

Вагомий вплив на працездатність та здоров’я користувачів комп’ютерів здійснює виробниче середовище. Це середовище у виробничих приміщеннях (офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму, випромінювання.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам та запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини необхідно здійснити санітарно-гігієнічну характеристику умов працівника, який працює з програмним продуктом.

Таким чином, робота з комп’ютером не така безпечна, як може здатися на перший погляд. Тому дуже важливо дотримуватися правил охорони праці і гігієни при роботі з комп’ютером.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

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

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	6,2
Довжина	8,8
Висота	3,4

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

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,1
Обсяг, V	$м^3$	не менше 20.0	20,6

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

У зазначеному приміщенні працє 9 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа приміщення та а об'єм у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 кКал. у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

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

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

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	21,4-23	44-60	0,12
Тепла	23-25	50-70	0,1	23-25	52-70	0,1

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

В літню пору застосовується кондиціонер.

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

У приміщенні знаходяться наступні джерела шуму: принтер Pantum P2207, електродвигуни вентиляторів ЕОМ.

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

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень.

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

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

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 6,2 м.; довжина –8,8 м.; висота – 3,4 м.

Розрахунок штучного освітлення проводемо за методом коефіцієнта використання світлового потоку.

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

$$F=ESKZ/n,$$

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

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

S – площа освітлюваного приміщення (у нашому випадку *S=6,2 х 8,8 = 54,56* м^2);

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

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

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

Обчислимо індекс приміщення за формулою:

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

де: *S* – площа приміщення, *S = 54,56* м^2;

h – розрахункова висота підвісу, *h = 3,4* м;

A – ширина приміщення, А = 6,2 м;

B – довжина приміщення, В = *8,8* м.

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

Знаючи індекс приміщення, за знаходимо *n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу).* Підставимо всі значення у формулу, визначемо світловий потік: *F=58711 Лм.*

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

Число світильників визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 58711/ 8000=7,3 \text{ шт.}$$

Приймаємо необхідну кількість світильників 8 шт.

Висновки до розділу

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

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

Список використаних джерел інформації

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

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

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

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

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

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

7. Методичні рекомендації до виконання розділу «Заходи з охорони праці та техніки безпеки» у магістерській дисертації / Л.Д. Третякова; М-во освіти і науки України, Національний технічний університет україни «Київський політехнічний інститут» – Київ, КПІ, 2014. – 26 с. – Режим доступу до ресурсу: http://surl.li/dhulo (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи створення архівів у файловій системі NTFS з розмежуванням доступу.

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 створення архівів у файловій системі NTFS з розмежуванням доступу.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем створення архівів у файловій системі NTFS з розмежуванням доступу.

– Досліджена система створення архівів у файловій системі NTFS з розмежуванням доступу.

– На основі отриманих результатів досліджень створена програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання створення архівів у файловій системі NTFS з розмежуванням доступу.

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. 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 UMAC.

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 6464 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,7 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скакун П.П. Дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Гаража В.О. Особливості програмної реалізації алгоритму AES / В.О. Гаража, О.П. Доренський. // Актуальні задачі сучасних технологій: збірник тез доповідей Міжнародної науково-технічної конференції молодих учених та студентів, 19–20 грудня 2012 р., м. Тернопіль – Тернопіль: Вид-во ТНТУ ім. Івана Пулюя, 2012. – С. 184-185.

3. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

4. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

5. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

6. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

7. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

8. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

9. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

10. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

11. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

12. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

13. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

14. 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,

15. 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.

16. 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.

17. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

18. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

19. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

20. 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.*

21. 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.

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

23. Smirnov O., 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.

24. 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.

25. 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.

26. 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.

27. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

28. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

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

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of

Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

36. 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.

37. 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.

38. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

39. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

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

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

42. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

43. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

44. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

45. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

46. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

47. 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.

48. 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.

49. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

50. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

51. 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.

52. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи створення архівів у файловій системі NTFS з розмежуванням доступу.

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. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу.

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 створення архівів у файловій системі NTFS з розмежуванням доступу;

– 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.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

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 – 103 аркуша.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Улічев О.С.

Дослідження та програмна реалізація системи створення архівів у файловій системі NTFS з розмежуванням доступу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 50

Літера: РП

Кропивницький – 2023 року

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 створення архівів
у файловій системі NTFS з розмежуванням доступу');
Memo1.Lines.Add(' ');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Керівник: Улічев О.С.');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Розробив: студент Скакун Павло Павлович');
Memo1.Lines.Add('                  гр. КН-22М-1');
Memo1.Lines.Add('');
Memo1.Lines.Add(' м. Кропивницький 2022');
Memo1.Lines.Add('');
end;

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

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('Розбіжностей не виявлено');
    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 – розмежування доступу за допомогою алгоритму шифрування/дешифрування 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 для дешифрування';
  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.
```