**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Центральноукраїнський національний технічний університет**

**Кафедра кібербезпеки та програмного забезпечення**

На правах рукопису

Ангел В’ячеслав Ігорович

# Програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science

Спеціальність: 123 «Комп’ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

**Коваленко Анна Степанівна** _____

(підпис) (дата)

кандидат технічних наук

**ДОПУЩЕНО ДО ЗАХИСТУ**

**Завідувач кафедри**

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.



Кропивницький – 2021

**Міністерство освіти і науки України**
**Центральноукраїнський національний технічний університет**

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
**д.т.н., проф.**
**_____О.А.Смірнов**
« 11 » січня 2021 року

# З А В Д А Н Н Я
## НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

*Ангелу В'ячеславу Ігоровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science*

керівник роботи *Коваленко Анна Степанівна, канд. техн. наук*
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 186-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science*

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

*1. Призначення та область використання.*
*2.Перегляд аналогічних існуючих систем.*
*3. Опис і обґрунтування проектних рішень.*
*4. Етапи програмування системи.*
*5. Впровадження системи в промислову експлуатацію.*
*6. Висновки*

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

| | |
|---|---|
| *Структурна схема системи* | *1 аркуш* |
| *Функціональна схема системи* | *1 аркуш* |
| *Діаграма процесів* | *1 аркуш* |
| *Блок-схема алгоритму роботи додатку* | *2 аркуша* |

6. Дата видачі завдання « 11 » січня 2021 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної бакалаврської роботи | Строк виконання етапів кваліфікаційної бакалаврської роботи | Примітка |
|---|---|---|---|
| 1. | Аналіз існуючих систем | 10.03.2021 р. | |
| 2. | Постановка задачі, оформлення ТЗ | 15.03.2021 р. | |
| 3. | Розробка моделі компонента | 20.03.2021 р. | |
| 4. | Розробка структур даних | 25.03.2021 р. | |
| 5. | Розробка алгоритмів зв’язку та відображення | 30.03.2021 р. | |
| 6. | Програмування алгоритмів | 10.04.2021 р. | |
| 7. | Оформлення ПЗ | 17.04.2021 р. | |
| 8. | Попередній захист роботи | 14.05.2021 р. | |
| | | | |
| | | | |
| | | | |
| | | | |

**Студент** _____ _____
( підпис ) (прізвище та ініціали)

**Керівник роботи** _____ _____
( підпис ) (прізвище та ініціали)

# АНОТАЦІЯ

**Ангел В.І. Програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.**

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи автоматизації бізнес-процесів з використанням Data Science.

Метою розробки є програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science.

Результат роботи – програмна реалізація системи автоматизації бізнес-процесів з використанням Data Science.

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 XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.4 Sydney.

**Ключові слова**: комп'ютерна інженерія, автоматизація бізнес-процесів

# ABSTRACT

**Anhel V.I. Business process automation software using Data Science. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021**

In this bachelor's qualification the software which is intended for system of automation of business processes with use of Data Science is developed.

The purpose of the development is the software of the business process automation system using Data Science.

The result is a software implementation of a business process automation system using Data Science.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment of Delphi 10.4 Sydney.

**Keywords:** computer engineering, business process automation

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| АБС | – | автоматизована банківська система |
| АІС | – | автоматизованих інформаційних системах |
| БЗ | – | база знань |
| ЕС | – | експертна система |
| ЗК | – | загальні критерії |
| ІБ | – | інформаційна безпека |
| ІС | – | інформаційна система |
| КБ | – | комерційний банк |
| КС | – | конкурентоспроможність |
| НЛ | – | системи нечіткої логіки |
| НМ | – | нейронні мережі |
| ПЗ | – | профіль захисту |
| ПК | – | програмний комплекс |
| СЗІ | – | система захисту інформації |
| ARIS | – | (Architecture of Integrated Information Systems, засобів моделювання |
| BPMN | – | Business Process Management Notation – це система умовних позначок для побудови схеми протікання бізнес-процесів |
| ERP | – | Enterprise Resource Planning, планування ресурсів підприємства |

# ВСТУП

**Актуальність теми**. Багато рітейлерів просунулися убік автоматизації бізнес-процесів. Наступним на порядку денному стоїть питання інтелектуальної автоматизації або автоматизації прийняття рішень – впровадження Data Science і технологій машинного навчання. У зв'язку із цим з'явилися нові походи: алгоритмічний маркетинг, логістика, планування замовлень і поставок, так як в їхній основі – алгоритми й моделі, на які переносять рутинні завдання фахівців. Останні, у свою чергу, починають займатися більш інтелектуальними речами.

Кожний з фахівців з маркетингу, по взаємодії із клієнтами знайде серед цих завдань свої. У поточних умовах вони все стосуються супроводу життєвого циклу клієнта: визначення етапу, наповнення вирви, розуміння, активний клієнт чи ні, які ознаки сьогодні говорять, що ми ризикуємо його втратити, прийняття рішень про повернення споживача або максимізація його досвіду.

Це важливий момент із погляду економіки: фокус не просто на збагачення клієнтського досвіду, але й обопільне одержання економічного ефекту.

Завдання використання даних – зрозуміти, на якому етапі вирви й життєвого циклу перебуває клієнт. Здається, що це тривіальне завдання. Для його рішення часто використовують евристичний підхід: якщо клієнт зробив одну покупку або дві – він є новачком, якщо не купував 30 днів – він у відтоку, якщо не купував 15 днів – у зоні ризику.

Однак евристичні правила можуть помилятися. Наприклад, клієнти з коротким міжтранзакційним інтервалом уже через три-чотири пропуски інтервалів переходять у зону ризику. Якщо ми до них вертаємося через 1,5-2 місяця, то збільшуємо витрати не на втримання, а на повернення клієнтів. Це дорожче.

У категорії новачків ми спостерігаємо багато « one-таймерів». Це клієнти, які один раз спробували й пішли у відтік по різних причинах. Наприклад, частина

клієнтів стають «one-таймерами», так як не супроводжуються наступним зворотним зв'язком або пропозиціями.

Завдання на цьому етапі – вибудувати ритм комунікації з новим споживачем і генерувати пропозиції, які дозволять розвивавати його досвід з компанією. Клієнт одержує можливість налагодити ритм взаємодії й перетворитися в лояльного.

**Мета й завдання дослідження.** Метою роботи є програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science.

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

– Огляд існуючих систем автоматизації бізнес-процесів з використанням Data Science.

– Дослідження системи автоматизації бізнес-процесів з використанням Data Science.

– Програмна реалізація системи автоматизації бізнес-процесів з використанням Data Science.

**Практична цінність отриманих результатів** полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматизації бізнес-процесів з використанням Data Science.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

# 1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

## 1.1 Призначення системи

Екосистема таргетування сьогодні представлено багатьма елементами й підходами. Ціль таргетування – визначити:

– Кому пропонувати.

– Який стимул використовувати (товар, механіка промо, інформація, контент).

– У який час (і враховувати дані для поповнення товарних позицій).

– Як реагувати на конкурентну інформацію й інформацію про тренди.

– По якому каналу (директ-маркетинг, онлайн-канали, push-повідомлення, живе спілкування з консультантом).

Один з підходів – формування фіксованої промо-матриці. Він має на увазі готові товарні позиції, погоджені комерційні умови, плани дисконтів, акційні плани й погоджені під них механіки. Залишається зрозуміти, як, через які канали й до яких клієнтів донести промо-матрицю, спланувати й ухвалити рішення щодо її доставки за допомогою маркетингу. Компанії, які працюють по такому принципу, використовують product-driven або discount-driven стратегію – задають матрицю й під неї шукають клієнтів.

Частина компаній переходить до customer-driven підходу. Вони відслідковують усі події, пов'язані з життєвим циклом клієнта, із тригерами, зовнішніми або такими подіями, які характеризують клієнта, і починають формувати гнучку промо-матрицю.

## 1.2 Область застосування

Найбільша частина роботи сьогодні ведеться з активними клієнтами, вони – «ядро бізнесу».

Найбільш важливі завдання у взаємодії з ними – це:

– Зрозуміти, які максимізаційні стратегії можуть застосовуватися до клієнтів: продовження життєвого циклу, розширення асортиментів, підвищення еластичності чека;

– Вчасно розпізнавати й грамотно взаємодіяти із клієнтами в зоні ризику.

На основі даних ми можемо підготувати ряд моделей, які дозволять ідентифікувати мінімальні відхилення в поведінці клієнта. Ланцюжок покупок клієнта, зміна якого-небудь товару в його кошику, чека, відгуку на комунікацію або використання накопичених балів – усе це може говорити про коливання клієнта.

Варто пам'ятати, що існує органічний відтік, наприклад, молоді батьки. У міру дорослішання дитини вони перемикають свій попит. На основі транзакційних даних можна ідентифікувати цих клієнтів, вчасно задіяти конкурентне перемикання й удержати їх.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

# 2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

## 2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Створення практично будь-якого застосунку для бізнесу, включаючи CRM, не обходиться без створення бізнес-процесів.

Побудова бізнес-процесу – не завжди проста справа, так як потрібно не тільки накреслити якусь схему, але й продумати, як практично буде відбуватися той або інший процес, регламентувати його, донести цей регламент до кожного співробітника, протестувати, як усе відбувається на практиці, подумати над недоліками й варіантами їх виправлення.

Для моделювання бізнес-процесу відповідно до нотації BPMN найчастіше використовується спеціальне ПЗ для моделювання бізнес-процесів, здатні не тільки становити схеми, але й забезпечувати реальну роботу бізнесу по цих схемах.

BPMN – Business Process Management Notation – це система умовних позначок для побудови схеми протікання бізнес-процесів (моделювання бізнес-процесів).

В основі будь-якої схеми такого типу лежить подія. Приміром, до подій можуть відноситися:

– Одержання заявки від клієнта (початкова подія).

– Створення документа.

– Узгодження документа.

– Відправлення товару клієнтові/надання послуги клієнтові.

– Одержання відгуку від клієнта (кінцева подія, якщо не потрібно обробити відгуку).

– Інші події.

Крім подій, існують шлюзи. Саме вони дозволяють одержати конкретну схему бізнес-процесу, так як саме шлюзи регламентують рух процесу. Це численні зв'язування-переходи з логічними значеннями «далі», «якщо», «і». Вони дозволяють бізнес-процесам гілкуватися: від одного події (наприклад, вступ заявки) відходять кілька шлюзів: фіксація заявки, обробка заявки, збір контактних даних від клієнта, і так далі.

9 програм для моделювання бізнес процесів:

**1. Bizagi Process Modeler**

Безкоштовне програмне забезпечення для створення діаграм процесів і документації в нотації стандарту BPMN. Відмінний інструмент побудови бізнес-процесів. Допомагає не тільки створити, але й опублікувати результати роботи в різних форматах, включаючи MS Word і інтерактивний HTML.

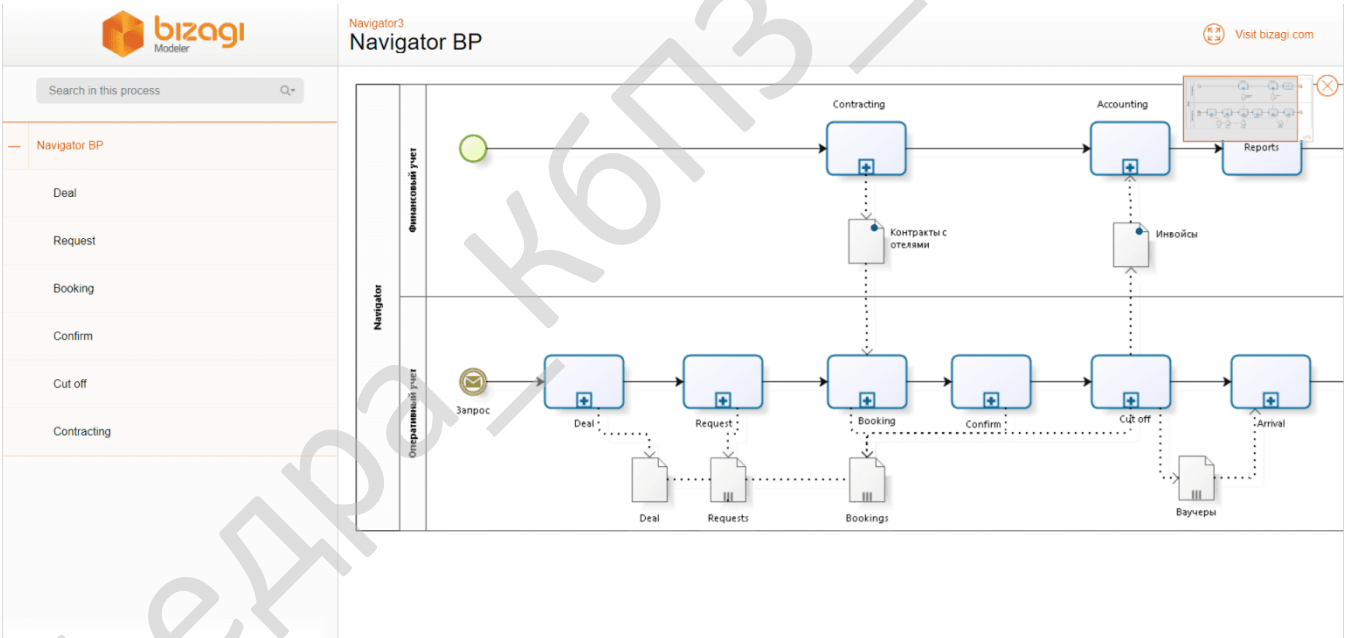


Рисунок 2.1 – Інтерфейс користувача Bizagi Process Modeler

**2. Intalio BPMS**

Безкоштовна. Open Source Business Management System. Програма для побудови й аналізу бізнес процесів.

**3. ARIS Express**

Досить простій в установці й використанні інструмент, так що його можуть застосовувати й починаючі користувачі. ARIS Express належить до сімейства засобів моделювання ARIS (Architecture of Integrated Information Systems) і включає не тільки інструменти моделювання бізнес-процесів і публікації моделей, але засоби, що й інтегруються між собою, розробки системи збалансованих показників, оцінки й оптимізації вартості бізнес-процесів, інструменти, що спрощують впровадження ERP-систем, а також інструменти контролю над виконанням бізнес-процесів.

**4. Camunda**

Це BPM-движок для автоматизації бізнес-процесів.

– Відкритий вихідний код дозволяють однозначно розуміти як працює софт, а відмінна документація дозволяє дуже швидко розібратися, як інтегрувати движок у свою інфраструктуру.

– Camunda підтримує останню версію Java, або взагалі будь-який JVM-мову.

– Відмінна архітектура усередині – движок робить те, що від нього очікується самим очевидним і очікуваним способом. Немає ніяких зайвих абстракцій, які необхідно вивчати.

– Зручність розробки, тестування й вбудовування в CI\CD за рахунок того, що Camunda можна використовувати просто як бібліотеку в Java-додатку. Camunda не обмежує розроблювача якимись своїми умовами. Використовуйте будь-які зручні інструменти – статистичні аналізатори, тестові фреймворки, засоби складання, засоби контролю версій.

Camunda – це також набір застосунків Modeler, Task List, BPMN Engine, DMN Engine, Cockpit, Admin,Optimize.

Modeler – це застосунок для створення моделей BPMN процесів. Ці моделі потрібні для інших частин системи.

Task list – це веб-застосунок, у якому виконавці виконують завдання, поставлені на них бізнес-процесом.

BPMN Engine – це безпосередньо движок, які відповідає за інтерпретацію BPMN в об'єкти JAVA, збереження об'єктів у базі й реалізацію інших речей (типу листенерів активностей), які звиваються навколо процесів.

DMN Engine – аналогічно BPMN Engine, тільки для DMN (Decision Model and Notation)

Cockpit – це веб-застосунок для перегляду стану процесів. У безкоштовній версії він сильно обрізаний по функціоналу.

Admin – це веб-застосунок для керування правами користувачів і користувачами.

Optimize – це веб-застосунок для аналізу бізнес-процесів. Він платний.

**5. Allfusion Process Modeler**

Дозволяє проводити опис, аналіз і моделювання моделі даних, побудівник позначка-моделей даних. Займає одне з лідируючих місць у своєму сегменті ринку.

Включає три стандартні методології: IDEF0 (функціональне моделювання), DFD (моделювання потоків даних) і IDEF3 (моделювання потоків робіт).

**6. IBM Websphere Business Modeler**

IBM Websphere Business Modeler є програмним засобом, націленим на моделювання, імітацію й аналіз бізнес-процесів.

Ключові характеристики IBM Websphere Business Modeler такі:

– дозволяє сформувати перелік показників KPI, прив'язати їх до елементів бізнес-процесу й шляхом імітації моделі спрогнозувати їхні значення. Таким чином, відслідковується досягнення стратегічних і тактичних цілей компанії.

– дозволяє описувати бізнес-процеси за допомогою діаграм стандарту BPMN. Інформація про організацію може накопичуватися у вигляді

структурованих довідників, між довідниками можуть установлюватися взаємозв'язки.

– за допомогою інструментарію Crystal Report у системі можуть створюватися будь-які види звітності по об'єктах моделі й регламентної звітності, які можуть бути вивантажені в Word, Excel, pdf та інші формати.

– система підтримує більш 40 видів аналізу як статичного (аналізується структура моделі), так і динамічного (аналізується модель під час і після імітації).

– можливості збору й контролю значень показників дозволяють використовувати систему не тільки як систему проектування, але і як систему виконання.

– моделі можуть бути опубліковані так, що стануть доступні команді розроблювачів для ознайомлення й аналізу.

– система легко інтегрується з іншими продуктами розробки IBM.

**7. ELMA**

Є безкоштовна версія. Система керування бізнес-процесами заснована на простій ідеї: іде побудова моделі бізнес-процесів вашої компанії за допомогою наочних діаграм (нотація BPMN), завантажуєте ці описи в комп'ютерну систему ELMA, і програма дозволяє відстежити виконання процесів у реальній практиці роботи підприємства.

Ключові характеристики:

– крім керування послідовними завданнями, які вигідно автоматизувати, є модуль керування проектами.

– існуюча система контролю (у тому числі через модуль керування KPI) і звітів створює оптимальний умови для роботи в команді, у тому числі віддалено (особливо коштовне для філій).

– електронний документообіг зв'язаний з усіма модулями системи й забезпечує зберігання, класифікацію документів. Це значно заощаджує час і зводить концепцію «незамінного працівника» до мінімуму.

– облік клієнтів і прав доступу вирішений у модулі CRM: з'явилася функція інтеграції з call центрами. Для рядового користувача ELMA може бути корисна як альтернатива внутришньокорпоративній пошті й інструмент керування завданнями.

**8. Fox Manager Бізнес Процеси**

Безкоштовна. Використовувана нотація близька до Basic Flow Chart, яка багатьом добре знайома своєю простотою побудови бізнес процесів. Програма автоматично будує процеси верхнього рівня, відображаючи взаємодії категорій і бізнес процесів у вигляді наочної діаграми. Аналітичні функції програми дозволяють вчасно помітити й усунути помилки, допущені при побудові побудованої процесної моделі, а саме виділити процеси, за які ніхто не відповідає, знайти посилання на неіснуючі документи, посади, постачальників або бізнес процеси.

**9. Comindware Business Application Platform**

Low-code платформа для моделювання й керування BPMN-процесами й цифрової трансформації підприємства.

Платформа від Comindware прекрасно підходить для спрощення й поглиблення автоматизації бізнес-процесів у рамках систем електронного документообігу. Затвердження й підписання договору – найбільш типовий процес у рамках документообігу будь-якої компанії. За допомогою користувацького інструмента від Comindware, що входить у функціонал платформи й доступного з будь-якого веб-браузера, з'являється можливість без зайвих складностей зібрати такий процес відповідно до BPMN.

## 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 внутришньоій 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 кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи автоматизації бізнес-процесів з використанням Data Science.

В процесі розробки кваліфікаційної бакалаврської роботи 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

Machine Learning і AI – це, у першу чергу, про дані, їхня якість і постійну роботу над їхнім поліпшенням. На підставі даних можна одержати інформацію про ефекти, відокремивши споживачів з базовою реакцією на промо від тих, хто був мотивований shape-активностями. Важливо звертати увагу не тільки на конверсію і якість клієнтів, але й на інформацію про збільшення чека – який економічний ефект принесла кампанія.

Економічний ефект відноситься багатьма командами в главу кута. Він може бути розбитий на дві частини: позитивний ефект від кожної окремої акції й від довгострокової взаємодії. Важливо не перевантажити споживача «рекламним шоком» і одержати не замученого, а задоволеного своєчасністю й релевантною інформацією клієнта. Це не завжди промо, це можуть бути інформаційні повідомлення або корисний контент, щоб клієнт почував зв'язок з компанією.

Згідно з дослідженнями Gartner, багато компаній, які вирішують свої завдання з командами Data Science, схиляються убік консалтингу, сервісів і готових застосунків. Проте, готової «коробочки», яка замінить гарних фахівців з маркетингу або сервісу немає. Алгоритми вирішують просте завдання: на основі накопичених даних визначити вирішальні фактори. Серед них – що мотивує новачків рухатися далі, активних клієнтів – збільшувати чек або залишати нас. На підставі факторів алгоритми допомагають вчасно визначати активності з найбільшим ефектом і впливати на клієнта оптимальним з погляду бюджету образом.

Компанії, зацікавлені в правильній роботі із клієнтами, усе більше розуміють необхідність у цільовій або контрольній групі.

Клієнт не може одночасно перебувати у двох станах – довідатися й не довідатися про знижку. Суть у тому, щоб визначити максимально релевантні групи по ряду поведінкових характеристик, які дозволять розділити клієнтів по типу відгуку.

Тип відгуку при таргетуванні значимий з погляду ефекту.

У нас можуть бути клієнти, які безнадійно не реагують, і ми тільки зливаємо бюджет на рекламу для таких клієнтів. А є клієнти, які вірять в нас, і в кожному разі будуть із нами працювати. Ми або ризикуємо їх нервувати, або безглуздо спалюємо маржу на промо.

Є sleeping dogs – клієнти, яких не варто торкати. Вони й так купують, і краще їх зайвий раз не смикати, щоб вони не задумалися про перемикання на конкурентів.

Клієнти, що піддаються переконанням. Здається, що таких клієнтів просто знайти. Але практика показує, що є дуже багато факторів для визначення таких клієнтів: стимул, канал, товар, момент життєвого циклу, тип дисконту, кількість балів.

Команди маркетингу стають командами експериментаторів, готовлять і проводять експерименти, збирають інформацію, яка допомагає ділити й знаходити клієнтів.

Якщо предвідносити 100% клієнтів, які піддаються таргетуванню, можна одержати типовий зріз:

– Частина клієнтів з якогось моменту починає демонструвати позитивний ефект від відгуку;

– Частина має нульовий ефект. Багато компаній, не маючи інформації про А/Б тестування, транслюють рекламні пропозиції на всіх клієнтів, і в зоні нульового приросту згоряють гроші;

– Існує негативна частина. Вона залежить або від продукту, або від типу взаємодії, і може спричинити серйозні витрати по відтокові клієнтів.

Ми із клієнтами визначаємо необхідні елементи застосунку – розвиваємо й тестуємо Decision Hub. Наприклад, А/Б тести – набір тестів по типу факторів. Це фактори, які:

1) впливають на відгук клієнта (акції, канали, тимчасові відрізки, тривалість);

2) впливають на витрати, збір інформації про клієнтську поведінку й роботу із цією інформацією за допомогою Ml-моделей.

Тут використовуються не тільки базові Ml-моделі, які виконують прогнози, але й моделі, які дозволяють відокремити або знайти причинно-наслідкові зв'язки. Такі моделі використовують дані про цільові групи. Вони дозволяють визначити, які фактори промо, зовнішнього середовища допомогли відгукнутися клієнтам, виділити їх в окрему групу й працювати з ними за допомогою тестованого промо.

Ключове завдання оптимізації промо й таргетування – максимізація клієнтської взаємодії, залучення й утримання. Це три ключові завдання, над якими ми працюємо з погляду персоналізації.

Customer Journey може бути виражений у послідовності дій.

Перехід клієнта від стану до стану супроводжується яким-небудь впливом: зовнішнього середовища, промо, ціноутворення, акцій або часу контакту. Грамотний вплив дозволяє підсилювати намір клієнта бути до нас лояльним або розширити досвід з нами, або ж бути незадоволеним і піти до конкурента.

Щоб зрозуміти, що створює потік новачків і звідки вони приходять, важливим інструментом є marketing mix modeling – моделювання міксу рекламних каналів, які генерують трафік і клієнтську базу. Ми постійно аналізуємо ці канали залежно від часу, інвестицій і визначення найкращих засобів з погляду кількості і якості клієнтів.

**Прогнозування попиту**

**Ключові товари**

Кожний товар, який попадає на ринок, це вже аналог.

Є моделі, підходи й теорії, які дозволяють спрогнозувати поведінка революційного товару, якого ще не було. Але більша частина товарів має той або інший аналог. Запускаючи продукт, важливо зрозуміти, яку нішу й у якій ціновій категорії він займе, і стежити за поведінкою продажів: чи став товар ключовим у кошику покупця?

Ключові товари стають конкурентною перевагою й фокусом уваги рітейлерів. І лояльні, і нові клієнти оцінюють, у кого ключовий товар, який вони планують придбати один раз або на постійній основі, кращий по співвідношенню « ціна-якість». Аналізуємо цінову еластичність, сезонність товару й канал просування – і визначаємо свій ключовий товар.

У середині життєвого циклу продукту все намагаються оптимізувати маржу й акційну політику, працюючи з товаром. В основі роботи – стратегії розширення ринку, розширення асортиментів і набору маржинальності.

Важливо відслідковувати інформацію про характеристики товару. Є кампанії, які, продаючи продукцію, неякісно її характеризують. Але ж це база для оцінки, продажу продукції й генерації попиту.

**Конкурентна відповідь**

Великий увага, особливо в fashion і електроніці, приділяється «конкурентній відповіді». У реальному часі бізнеси відслідковують цінники, промо й «живуть» на агрегаторах. Так формується динамічне ціноутворення. Щоб сформувати гідну конкурентну відповідь, у першу чергу, варто зрозуміти, це мій ключовий товар чи ні? Але компанії часто забувають про це й реалізують competitor-driven або market driven підхід у ситуації, коли досить просто йти урівень із конкурентом.

Приклад реалізації успішної стратегії, відстеження й прогнозування попиту: коли невелика різниця на частину товарних позицій не зменшує попит, не викликає відтоку й дозволяє накопичувати маржу.

Повернемося до життєвого циклу товару. Важливо вчасно сформулювати стратегію виходу з ринку: коли виводити товар, за якою ціною, з якою знижкою. Як правило, такі застосунки маркетологи ухвалюють інтуїтивно. Причина в тому, що накопичених даних дуже багато, і людей не може проаналізувати всі фактори, які дозволяють оптимально оцінити вихід, промо або ціну виходу. Завдання алгоритмів – допомогти маркетологові це зробити й визначити, де ця крапка.

У різних індустріях попит формується по-різному. Товари швидкого споживання й поповнення мають один тип попиту, smooth-товари, які драйвятся імпульсивно або тривалими циклами – іншої. Відповідно, моделі, які прогнозують попит, теж різні, і можуть використовувати сотні й тисячі факторів.

Але не всі фактори значимі. На попит на одні товари впливають цінові, акційні, конкурентні зміни, на інші – інформація про погоду або зовнішні події.

Із цього набору складається модель прогнозування, яка дозволяє зрозуміти фактори, якими ми управляємо – ціни й акції. А кореляція з іншими факторами дає розуміння: скільки не вкладай, на скільки не зменшуй маржу за рахунок акцій, одержиш однаковий попит.

Планування промо схоже на складання кубика Рубика, який складається з таких напрямків, як продукти, товарний асортименти, часовий контекст і клієнтська база. Сегментів дуже багато, і з кожного хочеться одержати максимальний ефект. Завдання алгоритму – спростити роботу фахівця у виробітку рекомендованих значень по промо-плануванню.

Чисті ефекти промо, по яких відслідковується ефективність стратегій і моделей, позитивно сприймаються в бізнесі. Але часто вони показують негативні результати, так як не враховують перемикання або запасання товаром.

Частина компаній намагаються спрогнозувати не тільки Post-ROI, не тільки вичленувати uplift, але й спрогнозувати й оптимізувати Pre-ROI: оцінити

попит, протестувати гіпотези й поекспериментувати із промо-офферами, механіками, товарним асортиментами, одержати набори ефектів і змоделювати завдання промо.

Ефекти можуть виражатися у вигляді маржинальности, кількості конверсій, покупок, market share. Важливо розуміти, яких моментальних цілей ми домагаємося в рамках промо на короткий період, і яких у рамках більш тривалого циклу.

У момент локдауну багато компаній зрозуміли, що потрібно активно утягуватися й брати планування й керування у свої руки. Завдання, яке надходило в цей момент моделям і даним, була перекручена, так як подібних подій в Україні не було. Довелося швидко вчиться, вивчати те, що зараз називають нова нормальність, з погляду даних. По деяких завданнях і прогнозам необхідно було відключати моделі, швидко накопичувати нові дані, пробувати на невеликих відрізках спланувати нову нормальність, новий попит. Частина компаній серйозно підійшла до цього питання. Вони включили фактори «закриття магазинів», «локдаун», «закриття метро», і на даний момент можуть із більшою впевненістю спрогнозувати попит з настанням червоної зони/ закриття суспільного транспорту/ масової роботи у вилученому режимі.

Ключові виводи:

1. «Без мізків і рук нікуди» – тільки ручне планування.

2. Пам'ятаємо, що це теж дані, які слід накопичувати й використовувати надалі у прогнозуванні.

Велике значення мають внутришньоасортиментні крос-залежності: дуже важливо розуміти, хто ще продається усередині моєї групи, у суміжних групах, чи існують залежності між сусідніми товарами на полку, не тільки конкуренти. Мої ж товари можуть виступати моїми конкурентами (ефект канібалізації). Ці елементи на очах, про них усі інтуїтивно знають, але питання як їх розраховувати і які метрики використовувати, щоб грамотно їх задіяти.

### 3.2 Розробка структурної схеми

Серед підприємців поширена думка, що автоматизація бізнес-процесів – доля великого бізнесу. А дрібному перед тем, як щось автоматизувати, необхідно масштабуватися. Але на практиці завжди їсти що автоматизувати, навіть якщо бізнес – це один ларьок преси або кіоск із кавою.

**Навіщо автоматизація потрібна малому бізнесу?**

Автоматизація бізнес-процесів дозволяє підприємствам скоротити грошові й тимчасові витрати, побільшати прибуток і підвищити конкурентоспроможність. Число організацій, що використовують у своїй роботі інформаційні технології, постійно росте з 2014 року. Особливо затребуване ПЗ для організаційних, управлінських і економічних завдань, а також для фінансових розрахунків.

**Що автоматизувати?**

Автоматизувати зараз можна майже будь-які процеси в бізнесі – продажу, маркетинг, керування, бухгалтерську й фінансову звітність.

Наприклад, у випадку з кіоском для продажу преси або іншими маленькими торговельними об'єктами, впровадження системи складського обліку дозволить підняти продажі. А «1С» дозволить організувати бухоблік.

Якщо штат підприємства – кілька сотень людей, то придадуться платформи для керування кадрами. Існують як спеціалізовані HR-платформи, так і бізнес-платформи, що поєднують функції кадрового, фінансового, податкового обліку.

За результатами опитування підприємців, 83% малих підприємств використовують програми бухобліку, 40% – програми для роботи із клієнтами, 61% – ПЗ для документообігу, 60% – для роботи з базами даних.

Хоча актуальність автоматизації бізнес-процесів дуже висока, специфіка малого бізнесу диктує свої правила впровадження It-технологій. Якщо великий бізнес може дозволити собі окремі застосунки для кожного напрямку – наймання, роботи із клієнтами, документообігу, бухгалтерського й податкового обліку,

керування кадрами, то малий бізнес найчастіше шукає єдиний застосунок для всіх завдань – багатофункціональні платформи або ж автоматизує актуальні для їхнього бізнесу процеси.

**Автоматизація служби підтримки**

Згідно з дослідженням Chatbotsmagazine, чат-боти – майбутнє торгівлі. 67% американців покоління Y згодні робити покупки з їхньою допомогою, а 40% використовують ботів для шоппінгу щодня. Чат-боти довелися по вдачі покупцям своєю миттєвою реакцією й доступністю в режимі 24/7.

У чат-ботів є 5 серйозних переваг для здійснення підтримки й продажів:

– **Доступність.** Чат-боти готові до роботи цілодобово, без вихідних, свят і обідів, сім днів у тиждень і 365 у році. Можливість у будь-який необхідний момент одержати інформацію або технічну підтримку – безумовно, привабливий сервіс в очах покупця й можливість бути ближче до свого клієнта.

– **Ефективність.** У будь-якій сфері бізнесу, будь те будівництво або надання послуг, є свій список «питань, що часто задаються». Відповіді на них займають величезну кількість часу й вимагають роботи цілої команди операторів. Брати на себе ці тяготи можуть чат-боти. Якісно сконструйований бот дозволить надавати всю необхідну інформацію про компанію або товар по всіх запитах одночасно.

– **Низька вартість.** Використання чат-ботів може суттєво знизити витрати компанії. Вони легко настроюються під різні вимоги, охоплюють широку аудиторію й працюють безперервно, виконуючи обов'язку цілого штату співробітників. А витрати на створення, підтримку й відновлення їх не настільки високі.

– **Універсальність.** Чат-боти легко інтегруються практично з усіма платформами й мають безліч варіантів настроювання. Завдяки цьому їх легко застосувати до будь-якої корпоративної системи (CRM, сайт).

– **Поінформованість.** У чат-ботів є можливість збору статистики і її наступного аналізу. Вони можуть записувати дані, які допоможуть мониторити і оптимізувати роботу.

Чат-ботів уже активно використовують у великих компаніях. Але вони підходять і під потреби невеликих підприємств. Сергій Кокорін керує компанією «100 земель» і вже спробував чат-бота для підтримки й продажів. За його словами, після впровадження виросла ефективність комунікації із клієнтами: чат-бот може правильно відповісти на будь-які питання про придбання земельної ділянки. Звичайно, на початку знадобилося зібрати багато інформації, що хвилює покупців, витратити час, щоб коректно підібрати й скласти питання й відповіді. Періодично бота потрібно «донавчати». Зараз він працює на лендинзі тільки одного проекту, але планується запустити його й на інших.

Основні проблеми, які малий бізнес вирішує через чат-ботів – це недороге залучення й утримання клієнтів, ріст повторних продажів, зниження витрат усередині бізнесу за рахунок оптимізації процесів:

**Автоматизація продажів**

**Автодозвон**

Обзвон клієнтів – часто рутинне заняття, що віднімає багато часу й потребуюче роботи декількох операторів. При цьому клієнти можуть не брати трубку годинником. Тому автоматизація бізнес-процесу тут як не можна до речі. Для того, щоб застосунати ці проблеми й знизити витрати, створені системи автодозвона. Звичайно, автодозвон працює в трьох режимах:

– прогресивний (progressive) – максимальна кількість каналів, підійде для автоінформування;

– предиктивний (predictive) – мінімальний час очікування оператором клієнта – як тільки абонент відповідає на дзвінок, його відразу з'єднують із оператором;

– прев'ю (preview) – оператор ініціює дозвон і не знімає трубку, поки клієнт не відповість. Оптимальний застосунок для колл-центрів.

**Чат-боти**

Чат-боти ефективні й у продажах.

**Онлайн-каси**

Завдання онлайн-кас – оперативне відправлення даних про всі транзакції налоговикам. Це спрощує податковий облік і робить його прозорим. Онлайн-каси потрібно впроваджувати всім, хто має відношення до торгівлі.

Сервіс дозволяє вести облік товарів, набудовувати автозакупку, надає доступ до повної інформації про структуру чеків, а також збирає й показує дані про швидкість продажів тих або інших товарів, величині й динаміці попиту, асортиментах і надає можливість працювати із програмами лояльності.

Сервіси автоматизації бізнес-процесів зараз перестали бути прерогативою великого бізнесу.

Хоча частка хмарного софта поки становить у нашій країні лише 5%, вона щорічно росте й, згідно з очікуваннями експертів, уже до 2025 року виросте до 80%.

**Автоматизація маркетингу**

Згідно з дослідженням Forrester Research, витрати на автоматизацію в 2020 році перевищили $11,4 млрд, а в 2023 складуть $25,1 млрд, забезпечивши індустрії 14% росту. Більш 45% b2b- і b2 c-компаній зі списку Fortune 500 уже автоматизували свої маркетингові системи.

Автоматизація маркетингу збільшує конверсію й продажу, знижує вартість залучення клієнта. Звільняючи маркетологів від рутинних завдань, автоматизація маркетингових бізнес-процесів підвищує їхня продуктивність і дозволяє мислити творчо. Особливо це актуально у випадку з малим бізнесом, коли у вас немає маркетингового відділу, а є всього 1-2 людину для цих завдань або ви займаєтеся маркетингом самостійно. Автоматизація впорається з рутиною й залишить час для вибору правильної стратегії просування.


Рисунок 3.1 – Структурна схема системи

**Автоматизація розсилання**

У нашій роботі, у виробництві контенту, складно впроваджувати автоматизацію. Особливість підходу просто не дозволяє робити щось по шаблонові. Але супутні процеси – це обов'язково.

Наприклад, у нас є платне розсилання. Без автоматизації там ніяк – треба збирати адреси, фіксувати їх, робити так, щоб листа починали автоматично відправлятися на нові адреси.

**Автоматизація керування репутацією**

Позитивна онлайн репутація – один з важливих аспектів. Наявність відкликань, робота зі згадуваннями в мережі й контент-маркетинг – напрямку, які можна автоматизувати без значних витрат.

**Що ще можна автоматизувати в маркетингу?**

– SMM – автопривітання передплатників, публікація нових постів, відповіді компанії на повідомлення в соціальних мережах.

– CRM – залучення клієнтів із сайту за допомогою чат-бота й автоматичної покупки без участі менеджера.

– Контент-менеджмент – впровадження зручних CMS для сайту й блога.

Наявність автоматизації не скасовує необхідності правильного аналізу запитів цільової аудиторії, грамотної роботи менеджерів, генерації якісного й цікавого аудиторії контенту. Але автоматизація помітно прискорює просування й підвищує продуктивність маркетингових кроків. Автоматизовані маркетингові кампанії дозволяють нагадати клієнтові про товар у підходящий момент – наприклад, до особистого свята.

**Автоматизація фінансово-управлінського обліку**

Управлінський облік – система для збору даних про бізнес для наступного аналізу. Облік – один з видів автоматизації бізнес-процесів, він допомагає одержувати докладні звіти про рухи грошей, прибутках і збитках і інших фінансових даних. Малий бізнес найчастіше веде такі звіти в Excel-таблицях, це ускладнює аналіз інформації й вимагає багато часу.

Ідеальний управлінський облік дозволяє керівникові в будь-який момент визначити, що відбувається з бізнесом, і вибудувати подальші дії. Бухоблік тут не допоможе – він не дозволяє оцінити прибутковість окремих сегментів і не дає досить інформації для грамотного фінансового планування. Бухоблік потрібний для звітності перед державою.

У невеликих компаній, на відміну від більш масштабних колег, немає ні окремого фінансового відділу, ні департаменту аналітики, ні стратегів-менеджерів, що займаються плануванням. Усі ці завдання часто лягають на самого керівника, тому важливо звільнити дорогоцінної час власника й автоматизувати облік грошей.

Для автоматизації обліку потрібно зібрати й структурувати інформацію, а також підібрати інструменти під найбільш складні й нужденні в контролі статті й контури обліку. Якщо великий і середній бізнес може замовити ERP (англ.

Enterprise Resource Planning, планування ресурсів підприємства) під власні потреби, то малому бізнесу прийде використовувати різні інструменти.

Комп'ютерні алгоритми успішно заміняють роботу фінансистів по усьому світу. Автоматизація фінансового обліку для малого бізнесу дозволяє вирішувати проблеми з мультивалютністю, розділяти проекти різних контрагентів і уникати втрати засобів через порушення в обліку дебіту-кредиту.

Підведемо підсумки:

– Автоматизація бізнес-процесів раніше була прерогативою великого бізнесу, але зараз її активно використовує й малий бізнес.

– Автоматизувати можна будь-які процеси – продажу, маркетинг, фінансовий облік, навіть спілкування з покупцями.

– Автоматизація дозволяє підвищити продуктивність, побільшати прибуток, масштабувати бізнес.

– Для малого бізнесу є як недорогі застосунки по конкретних процесах, так і комплексні застосунки, що поєднують відразу кілька напрямків.

– Автоматизовані застосунки – реальний спосіб для малого бізнесу підвищити ефективність роботи, не роздмухуючи бюджет. Автоматизовані платформи знімають із керівника й персоналу рутинні завдання й дозволяють працювати над такими складними й трудомісткими процесами як планування, комунікація із клієнтами, вибудовування стратегії розвитку й просування.

### 3.3 Розробка функціональної схеми

Розглянемо функції розробленої системи автоматизації бізнес-процесів з використанням Data Science.

**Створення бізнес-процесів**

BPM-системи допомагають бізнес-аналітикам проектувати діаграми процесів будь-якої складності з більшою кількістю розгалужень, умов і бізнес-правил. Завдяки цьому організація одержує чітко прописані регламенти, які

уніфікують діяльність компанії й кожного окремого співробітника для досягнення кращих результатів.

Для побудови бізнес-процесів існують різні інструменти, визнаним стандартом серед яких вважається нотація BPMN. Платформа BPMNDataScience, яка розроблена в даній роботі, підтримує цю методологію проектування бізнес-процесів і використовує інтуїтивно зрозумілі графічні елементи, кожний з яких визуалізує той або інший крок процесу. Ключова перевага BPMN полягає в тому, що спроектована з її допомогою діаграма процесу зрозуміла як бізнес-аналітикам, так і рядовим користувачам системи.

**Автоматизація бізнес-процесів**

Основне призначення BPM-системи – автоматизувати концептуальну схему процесу, перетворивши її в схему, що виконується. Наприклад, в BPMNDataScience, яка розроблена в даній роботі, можна настроїти необхідну послідовність дій, інтерфейси користувача, умови переходу, правила автоматичного виконання кроків або обробки даних. Low-code інструменти дозволяють бізнес-аналітикам робити це самостійно, залучаючи розроблювачів тільки для настроювання складних ділянок.

Але не завжди процес випливає по єдиному сценарію – хід його виконання може залежати від різних факторів, у тому числі від прийнятих співробітниками застосунків. За даними аналітичного агентства Gartner, частка таких неструктурованих процесів в організаціях досягає 80%.

Платформа BPMNDataScience, яка розроблена в даній роботі, дозволяє автоматизувати як структуровані, так і неструктуровані, динамічні процеси завдяки синергії концепцій керування бізнес-процесами (BPM) і адаптивного кейс-менеджменту (Dynamic Case Management, DCM). Це єдина система з великим набором інструментів, яка дозволяє покрити весь контур процесів компанії.

**Виконання бізнес-процесів**

За виконання процесів відповідає ядро системи, яке зчитує, що виконується схему й рухає кожний процес від його старту до завершення. Від потужності движка залежить швидкість виконання процесу й робота всієї системи. Так, платформа BPMNDataScience, яка розроблена в даній роботі, дозволяє виконувати одночасно мільйони операцій без втрат у продуктивності системи.

В BPMNDataScience, яка розроблена в даній роботі, є два способи запуску бізнес-процесу: вручну користувачем або автоматично по зазначеній події (тригеру). Після запуску система допомагає співробітникові рухатися по процесі з обліком раніше прописаних регламентів: у простому й зрозумілому інтерфейсі підказує необхідні дії, відображає список завдань, дає рекомендації. Крім того, деякі транзакції система проводить автоматично, якщо це передбачене бізнес-правилами, наприклад, виконує розрахунки, відправляє повідомлення або листи по заданому шаблонові й багато чого іншого.

**Моніторинг і оптимізація процесів**

Усі процеси підприємства вимагають періодичного перегляду й корекції, що неможливо без налагодженого моніторингу. Функціональні можливості BPMNDataScience, яка розроблена в даній роботі, значно спрощують аналіз і оптимізацію бізнес-процесів. Система фіксує кожний крок кожного виконуваного процесу, відслідковує різні метрики виконання процесу (наприклад, тривалість або середній час виконання), оформляє отримані дані в дашборди, а також дозволяє оперативно вносити необхідні зміни в схему процесу.

**Готові інструменти для вирішення бізнес-завдань**

На відміну від класичних BPM-застосунків, платформа BPMNDataScience, яка розроблена в даній роботі, не тільки надає основу для побудови й автоматизації процесів організації, але також дозволяє компаніям вирішувати різні бізнес-завдання за допомогою готових конфігурацій і застосунків.

Наприклад, CRM-лінійка BPMNDataScience, яка розроблена в даній роботі, включає цілий комплекс інструментів для керування процесами маркетингу, продажів і сервісу в єдиному інформаційному середовищі. Готові галузеві застосунки призначені для автоматизації процесів підприємств із різних індустрій, адже державним установам і рекламним агентствам необхідні різні інструменти для досягнення поставлений цілей. А безліч доповнень і вже настроєних шаблонів дозволяють компаніям швидко підбирати підходящі інструменти й розширювати систему новою функціональністю.

**Інтеграції з іншими системами**

Для максимально ефективного управління підприємством усі його дані й процеси повинні функціонувати в єдиному цифровому середовищі. BPMNDataScience, яка розроблена в даній роботі, надає таку можливість: система зв'язує різні елементи IT-інфраструктури підприємства за допомогою різних інтеграційних застосунків. При цьому в коробкову версію BPM-платформи вже включені деякі інтеграції, наприклад, з поштовими сервісами, соціальними мережами й IP-телефонією для організації внутришньоіх комунікацій.

Крім того, десятки застосунків для інтеграції пропонує онлайн-майданчик Маркетплейс – каталог готових застосунків і коннекторів для BPMNDataScience, яка розроблена в даній роботі. Ці інструменти дозволяють синхронізувати платформу з обліковими системами, аналітичними платформами й іншими корпоративними програмами, при цьому для більшої частини інтеграцій не потрібне залучення розроблювача.

**Керування даними й документами**

Кожна компанія оперує величезними обсягами інформації: дані про клієнтів і співробітників, великий продуктовий каталог, різні документи, від управлінських до фінансових, і багато чого іншого. Бізнес-процеси використовують цю інформацію в тому або іншому виді, тому потрібно мати постійний, швидкий і зручний доступ до оперативних і накопичених даних організації.

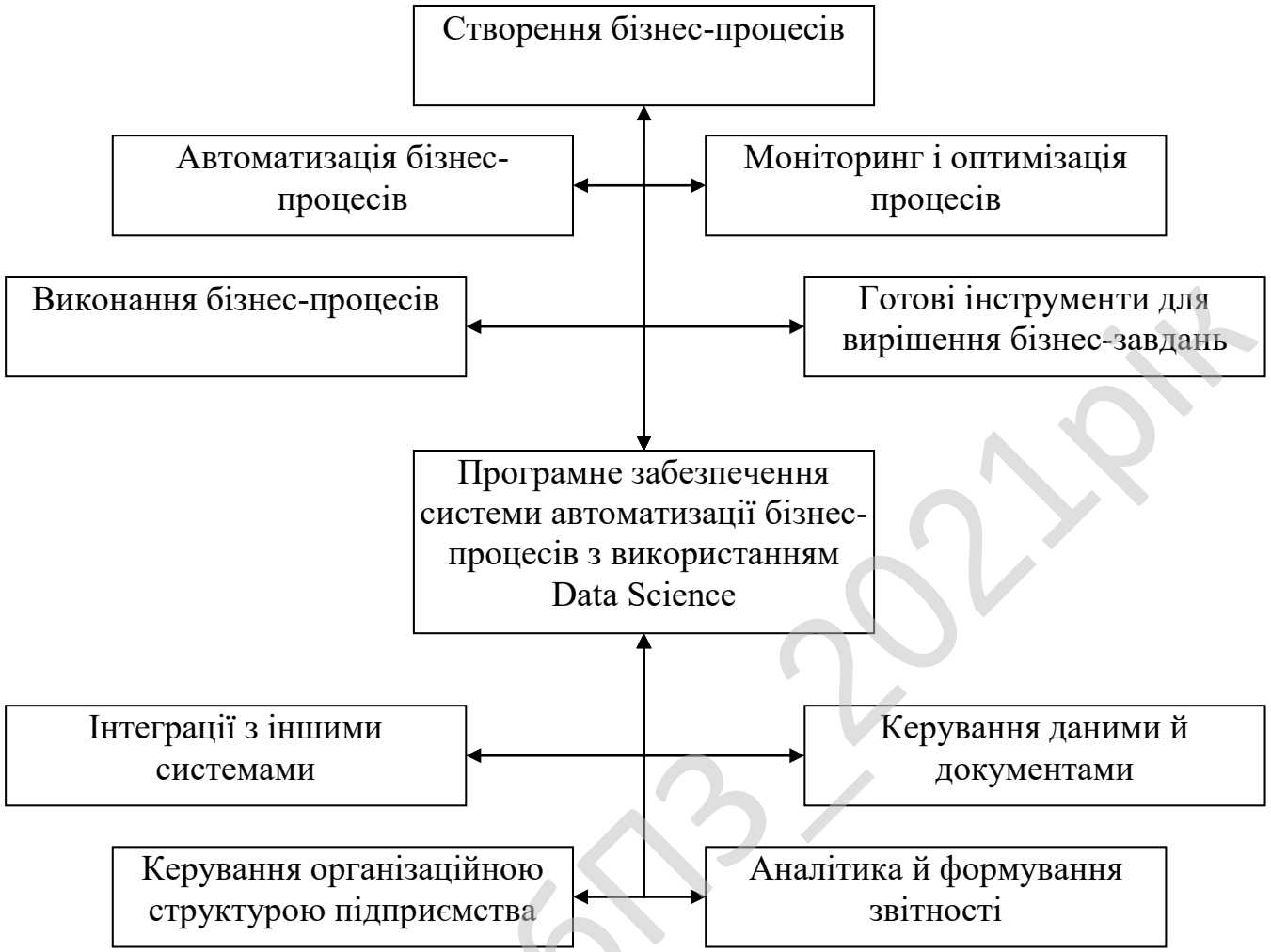


Рисунок 3.2 – Функціональна схема системи

BPM-системи надають широкий спектр можливостей для керування інформацією підприємства: з їхньою допомогою можна структурувати дані, організувати зручну навігацію по існуючих записах, настроїти автоматичне формування документів. Більше того, інтелектуальні технології обробки даних в BPMNDataScience, яка розроблена в даній роботі, дозволяють автоматизувати рутинні операції, допомагають ухвалювати правильні застосунки й швидше досягати запланованих результатів.

**Керування організаційною структурою підприємства**

Ключову роль у досягненні мети бізнес-процесу відіграє чітка й злагоджена робота всіх її учасників.

Концепція BPMNDataScience, яка розроблена в даній роботі, розглядає співробітників компанії у двох площинах – як багаторівневу організаційну структуру підрозділів і як розгалужену структуру функціональних ролей. Така багатошарова рольова модель дозволяє зафіксувати зони відповідальності кожного співробітника організації, а також дуже тонко розподілити рівні доступу користувачів до даних і операцій у системі на різних етапах виконання процесу.

**Аналітика й формування звітності**

BPM-система дозволяє працювати зі статистичними даними компанії, при цьому інформація доступна не тільки у вигляді цифр, але й у вигляді інформативних графіків і дашбордів для кращого сприйняття. Завдяки аналітичним можливостям системи відповідальні співробітники, наприклад, директори по розвиткові або керівники напрямків, можуть відслідковувати показники окремих співробітників і всієї команди для того, щоб визначати вузькі місця, знаходити шляхи оптимізації й підвищувати ефективність.

### 3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників.

Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

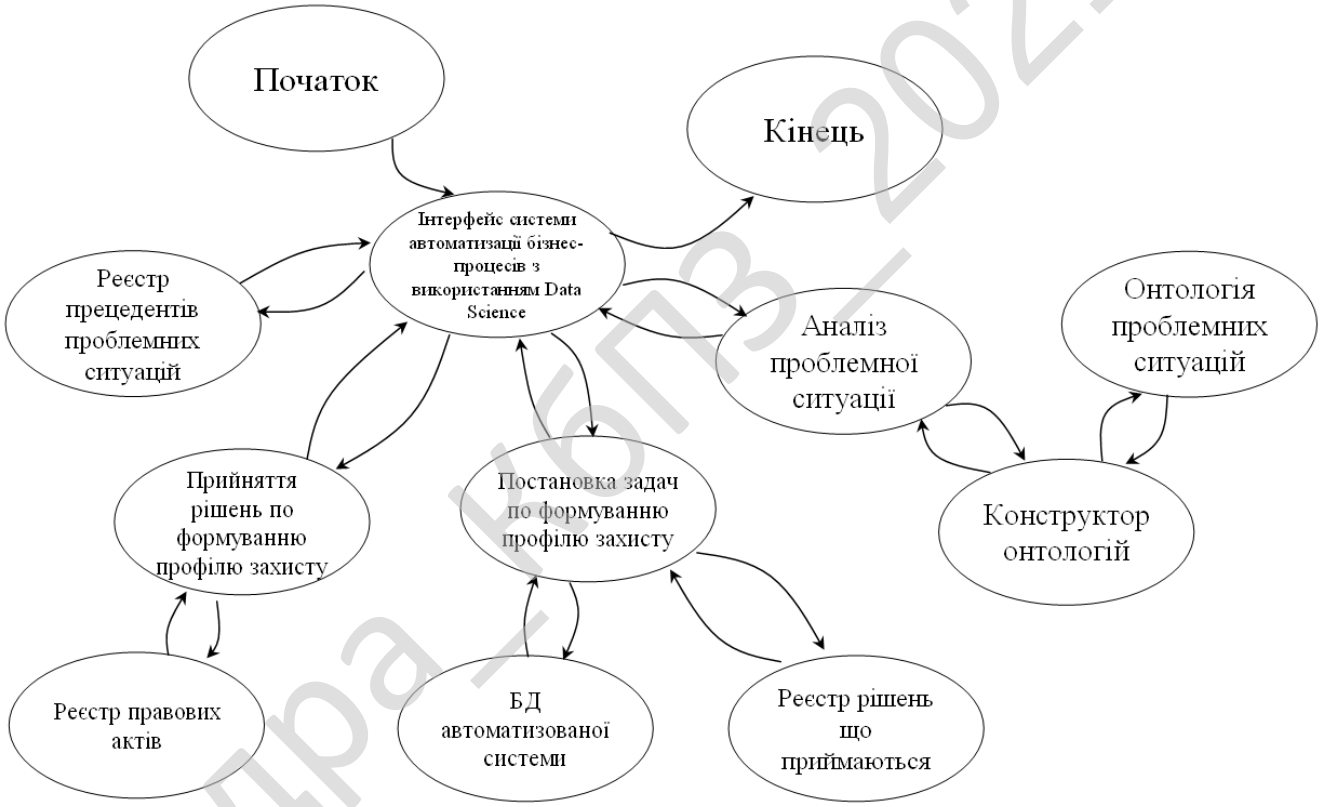


Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

## 4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо послідовність дій та викликів підпрограм в загальному алгоритмі роботи основної програми що зображено на рисунку 4.1. у вигляді блок-схеми:

– Виведення вікна автоматизації бізнес-процесів з використанням Data Science.

– Виклик підпрограми моніторингу та навчання що зображено на рисунку 4.2.

– Запит завантаження вагового файлу з наступною реалізацією.

– Запит завантаження криптотексту з наступною реалізацією.

– Виведення криптотексту на екран.

– Запит перевірки стійкості системи.

– Формування пакету даних та проведення криптоаналізу даних.

– Запит перевірки знайдення вразливості.

– Виведення на екран розшифрованого тексту.

– Обчислення показників стійкості застосованого алгоритму шифрування.

– Виведення на екран значень показників стійкості.

– Виведення рекомендацій по забезпеченню стійкості системи.

– Запит завершення роботи (цикл системи).

На рисунку 4.2 зображено роботу підпрограми з реалізацією наступних дій:

– Завантажити дані системи автоматизації бізнес-процесів .

– Цикл поки можлива мінімізація цільової функції.

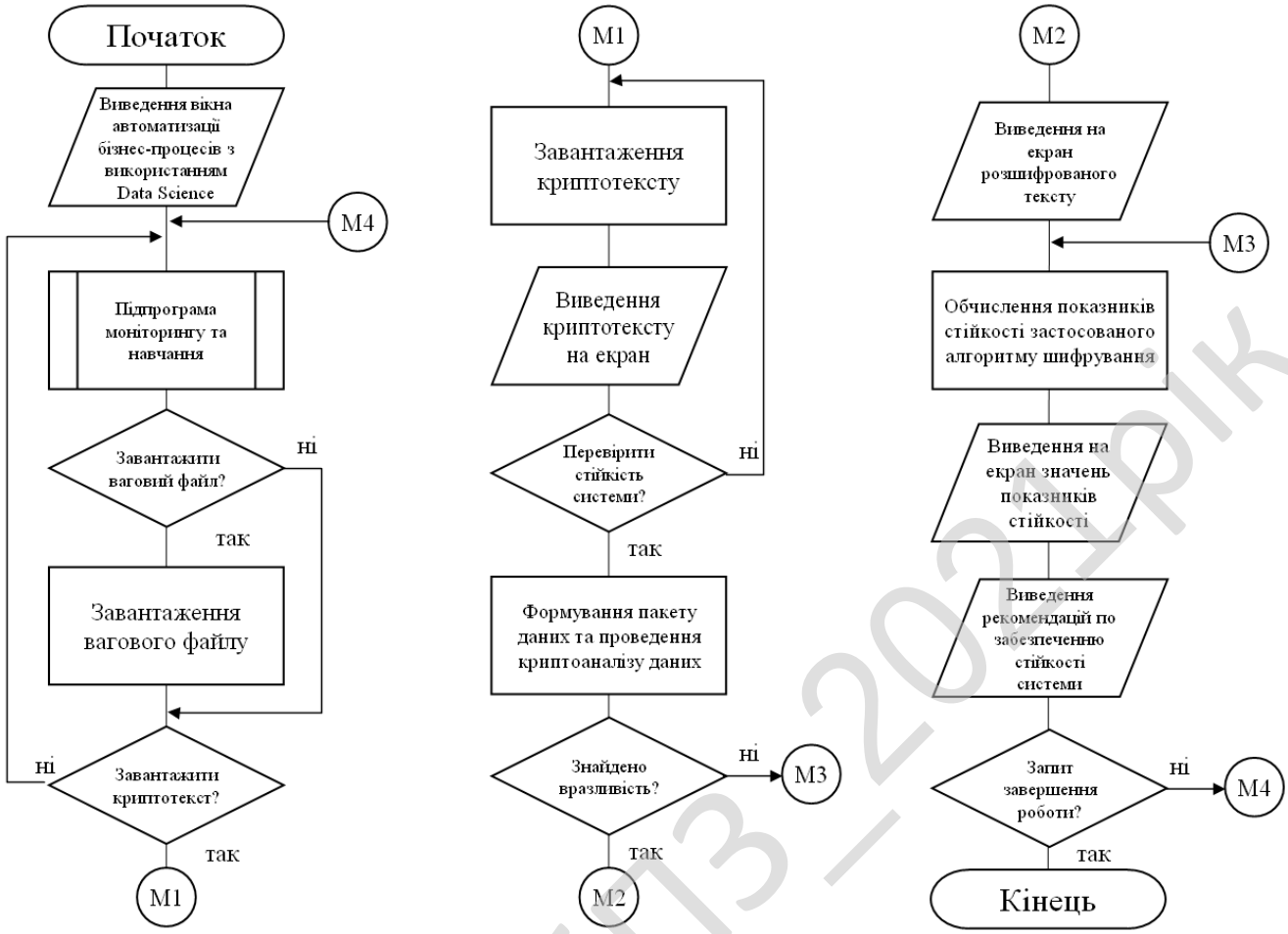


Рисунок 4.1 – Блок-схема основної програми

– Вибрати вагу випадковим чином.

– Підкоректувати вагу на невелике випадкове значення.

– Пред'явити множину входів X.

– Обчислити виходи Y.

– Обчислити величину різниці між фактичними та бажаними виходами.

– Знаходження цільової функції зведенням різниць у квадрат.

– Вибрати вагу випадковим чином.

– Підкоректувати вагу на невелике випадкове значення.

– Запит корекція зменшує цільову функцію.

– Підкоректувати вагу на невелике випадкове значення.

– Збереження значення цільової функції.

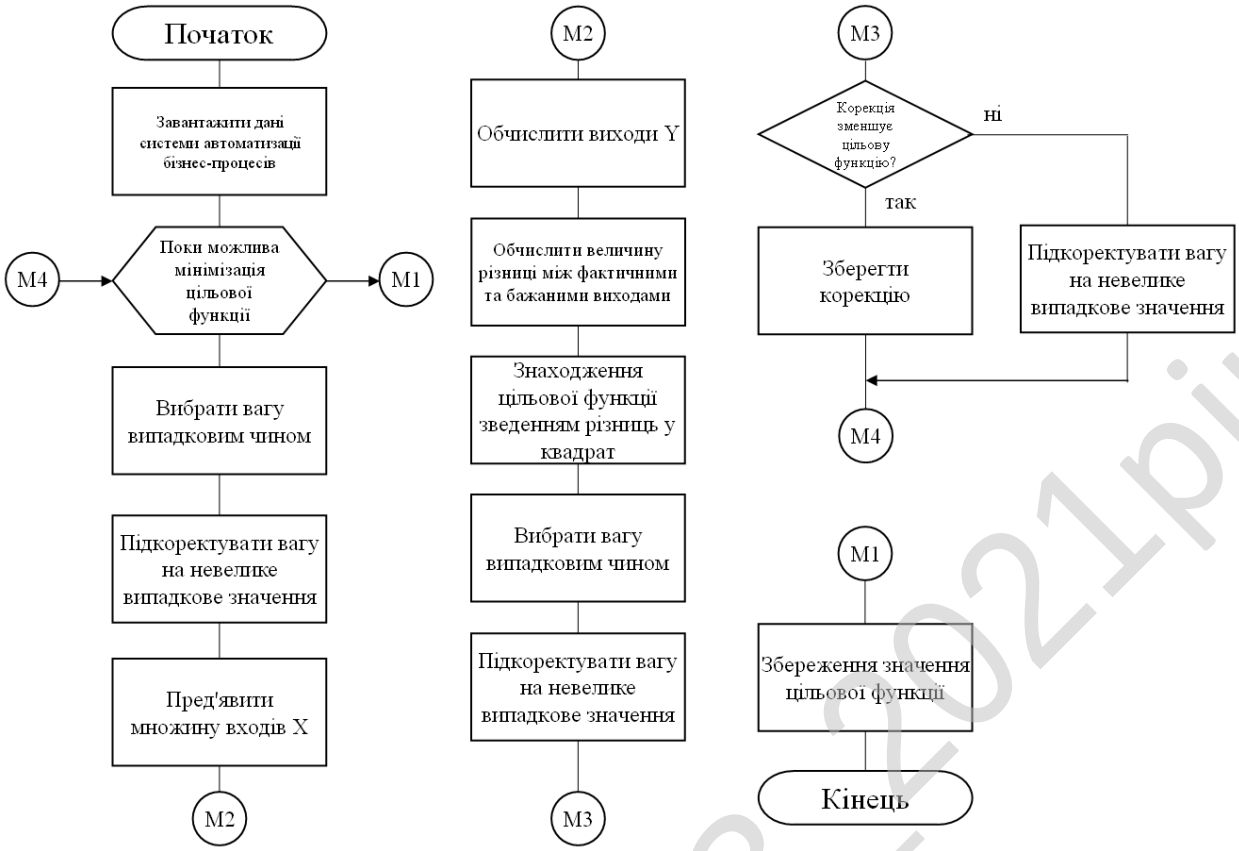


Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо модуль роботи з профілем захисту у файловому контексті у вигляді вихідного коду програмного забезпечення:

```
unit FileData; // назва модулю

interface //інтерфейс на частина модулю FileData

uses // виклик бібліотек
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ComCtrls, ShellCtrls, XPMan, Buttons, FileCtrl,
ExtCtrls;

Type // об’ява типів
  TForm1 = class(TForm)
ShellTreeView1: TShellTreeView;
Label1: TLabel;
ShellTreeView2: TShellTreeView;
Label2: TLabel;
XPManifest1: TXPManifest;
```

```
    GroupBox1: TGroupBox;
    ComboBox1: TComboBox;
    Label3: TLabel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    DriveComboBox1: TDriveComboBox;
    DriveComboBox2: TDriveComboBox;
    CheckBox1: TCheckBox;
    Bevel1: TBevel;
    SpeedButton1: TSpeedButton;
    Label4: TLabel;
    Edit1: TEdit;
    RadioButton3: TRadioButton;
    Edit2: TEdit;
    Animate1: TAnimate;
    RadioButton4: TRadioButton;
    CheckBox2: TCheckBox;
    procedure DriveComboBox1Change(Sender: TObject);
    procedure DriveComboBox2Change(Sender: TObject);
    procedure FindFile(Dir:String);
    procedure SpeedButton1Click(Sender: TObject);
    procedure ShellTreeView1Change(Sender: TObject; Node: TTreeNode);
    procedure ShellTreeView2Change(Sender: TObject; Node: TTreeNode);
    procedure RadioButton4Click(Sender: TObject);
    procedure RadioButton3Click(Sender: TObject);
    procedure RadioButton2Click(Sender: TObject);
    procedure RadioButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

    var
    // об'ява змінних
      Form1: TForm1;
      SR:TSearchRec;
    Implementation
    // частина реалізації

    uses Unit2;
```

```
// підключення модулів внутрішніх
{$R *.dfm}
// ресурс
procedure TForm1.DriveComboBox1Change(Sender: TObject);
begin
shelltreeview1.Root:=DriveComboBox1.Drive+':\';
end;

procedure TForm1.DriveComboBox2Change(Sender: TObject);
begin
shelltreeview2.Root:=DriveComboBox2.Drive+':\';
end;
// пошук
procedure TForm1.FindFile(Dir:String);
var
 i:integer;
 SR:TSearchRec;
 FindRes:Integer;
 Dir1,Dir2:string;
begin
 i:=0;
 FindRes:=FindFirst(Dir+'*.*',faAnyFile,SR);
 while FindRes=0 do
  begin
   if ((SR.Attr and faDirectory)=faDirectory) and
   ((SR.Name='.')or(SR.Name='..')) then
     begin
      FindRes:=FindNext(SR);
      Continue;
     end;
// якщо знайдений каталог
   if checkbox2.Checked=true then begin
   if ((SR.Attr and faDirectory)=faDirectory) then
    begin
// входимо в процедуру пошуку з параметрами поточного каталогу
     FindFile(Dir+SR.Name+'\');
     label4.Caption:='Поиск: '+dir+sr.Name;
     FindRes:=FindNext(SR);
// після огляду вкладеного каталогу ми продовжуємо пошук
// в цьому каталозі
     Continue;
// продовжити цикл
```

```
    end;
    end;
if checkbox2.Checked=false then begin
   if ((SR.Attr and faDirectory)=faDirectory) then
    begin
     FindRes:=FindNext(SR);
     Continue;
    end;
    end;
   if checkbox1.Checked=true then begin
if (combobox1.Text<>'*.*') and (copy(sr.Name,length(sr.name)-
   length(combobox1.Text)+1,length(combobox1.Text))=
   combobox1.Text) then begin
  form2.ListBox1.Items.Add(dir+SR.Name);
  form2.ListBox1.ItemIndex:=form2.ListBox1.Count-1;
  form2.ListBox3.Items.Add(SR.Name);
  form2.ListBox3.ItemIndex:=form2.ListBox1.ItemIndex;
  form2.Caption:=form2.ListBox3.Items[form2.ListBox3.itemindex];
  form2.ListBox2.Items.Add(inttostr(sr.size));
form2.Label4.Caption:=inttostr(form2.ListBox1.Count);
form2.Label6.Caption:=floattostr(strtofloat(form2.Label6.Caption)
+strtofloat(form2.ListBox2.Items[form2.ListBox2.Count-1]));
end;
end;
if Application.Terminated then Break;
Application.ProcessMessages;
FindRes:=FindNext(SR);
end;
FindClose(SR);
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
i, MaxWidth: integer;
begin
animate1.Active:=true;
speedbutton1.Caption:='Работаю...';
if checkbox1.Checked=true then begin
form2.Caption:='Идет поиск...';
form2.Show;
end;
```

```
form2.ListBox1.Clear;
// очищення списку файлів
form2.ListBox2.Clear;
// очищення списку розмірів файлів
form2.ListBox3.Clear;
// очищення списку імен файлів
form2.Label6.Caption:='0';
// обнуління загального розміру
FindFile(Edit1.Text);
// Пошук
animate1.Active:=false;
label4.Caption:='Поиск:';
form2.Label4.Caption:=inttostr(form2.ListBox1.Count);
// Кількість знайдених файлів
form2.ListBox1.ItemIndex:=0;
form2.progressbar1.Max:=form2.ListBox1.Count;
// Додавання горизонтальної прокрутки в form2.listbox1
MaxWidth := 0;
for i := 0 to form2.ListBox1.Items.Count - 1 do
if MaxWidth < form2.ListBox1.Canvas.TextWidth(
               form2.ListBox1.Items.Strings[i]) then
MaxWidth := form2.ListBox1.Canvas.TextWidth(
               form2.ListBox1.Items.Strings[i]);
SendMessage(form2.ListBox1.Handle, LB_SETHORIZONTALEXTENT, MaxWidth+2, 0);
if form2.ListBox1.Items.Text<>'' then begin
if radiobutton1.Checked=true then begin
copyfile(PAnsiChar(form2.listbox1.Items[0]),
PAnsiChar(Form1.ShellTreeView2.path+'\'+
form2.listbox3.Items[0]),longbool(0));
end;
if radiobutton2.Checked=true then begin
copyfile(PAnsiChar(form2.listbox1.Items[0]),
PAnsiChar(Form1.ShellTreeView2.path+'\'+
form2.listbox3.Items[0]),longbool(0));
deletefile(form2.listbox1.Items[0]);
end;
if radiobutton4.Checked=true then begin
deletefile(form2.listbox1.Items[0]);
end;
end;
if radiobutton3.Checked=false then begin
form2.Timer1.Enabled:=true;
```

```
// включення таймеру
form2.SpeedButton2.Enabled:=true;
form2.ListBox1.Enabled:=false;

end;
if radiobutton3.Checked=true then begin
form2.Caption:='Ок';
form1.speedbutton1.Caption:='Почати пошук';
end;
end;

procedure TForm1.ShellTreeView1Change(Sender: TObject; Node: TTreeNode);
begin
edit1.Text:=shelltreeview1.Path+'\';
end;

procedure TForm1.ShellTreeView2Change(Sender: TObject; Node: TTreeNode);
begin
 edit2.Text:=shelltreeview2.Path+'\';
end;

procedure TForm1.RadioButton4Click(Sender: TObject);
begin
shelltreeview2.Enabled:=false;
end;

procedure TForm1.RadioButton3Click(Sender: TObject);
begin
shelltreeview2.Enabled:=false;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
shelltreeview2.Enabled:=true;
end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
shelltreeview2.Enabled:=true;
end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
shelltreeview2.Enabled:=false;
end;
end.
```

Було використано MySQL - вільна система керування реляційними базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, яка отримала права на торговельну марку разом з поглиненої Sun Microsystems, яка раніше придбала шведську компанію MySQL AB.

Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за замовленням ліцензійних користувачів. Саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP, VertrigoServ. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СКБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СКБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СКБД MySQL постійно з'являються нові типи таблиць.

26 лютого 2008 року Sun Microsystems придбала MySQL AB за 1 млрд доларів, 27 січня 2010 року Oracle придбала Sun Microsystems за 7,4 млрд доларів і включила MySQL в свою лінійку СКБД.

Спільнотою розробників MySQL створені різні відгалуження коду, такі як Drizzle (англ.), OurDelta, Percona Server і MariaDB. Всі ці відгалуження вже існували на момент поглинання компанії Sun корпорацією Oracle.

MySQL має подвійне ліцензування. MySQL може розповсюджуватися відповідно до умов ліцензії GPL. Але за умовами GPL, якщо якась програма використовує бібліотеки MySQL, то вона теж повинна розповсюджуватися за ліцензією GPL. Проте це може розходитися з планами розробників, які не бажають відкривати сирцеві тексти своїх програм. Для таких випадків передбачена комерційна ліцензія компанії Oracle, яка також забезпечує якісну сервісну підтримку.

В разі використання та розповсюдження програмного забезпечення з іншими вільними ліцензіями, такими як BSD, Apache, MIT та інші, MySQL дозволяє використання бібліотек MySQL за ліцензією GPL.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM – підпрограми низького рівня для індексного доступу до даних. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» – достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze.

В січні-лютому 2008 Sun Microsystems придбала розробника системи керування базами данних MySQL за $1 млрд. Після поглинання у 2009 році Sun Microsystems компанією Oracle Corporation MySQL стала власністю Oracle.

За час розвитку під орудою Oracle дедалі більше відокремлює MySQL від спільноти і робить процес розробки все менш прозорим. Наприклад, повернута практика поставки власницьких розширених функцій в Enterprise-версії MySQL, спостерігається приховування інформації про вразливості, зі складу виключений тестовий набір, закритий доступ до більшої частини системи відстеження

помилок та припинено публікація згрупованого логу змін, що дозволяє судити про прив'язку патчів до конкретних змін.

**Можливості**

MySQL – компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

1. Простота у встановленні та використанні.

2. Підтримується необмежена кількість користувачів, що одночасно працюють із БД.

3. Кількість рядків у таблицях може досягати 50 млн.

4. Висока швидкість виконання команд.

5. Наявність простої і ефективної системи безпеки.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15–30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog

items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner 'а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

**Планування спринта (Sprint Planning Meeting)**

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда.

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах.

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі.

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт.

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо.

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog.

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

**Щоденна нарада (Daily Scrum Meeting)**

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?;

– Що буде зроблено з моменту поточної наради до наступної?;

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1–3ма годинами.

### 4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення по модулю $2^{32}$-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту ($x_0$, $x_1$, $x_2$, $x_3$) і 32-бітовими підблоками ключу ($k_0$, $k_1$, $k_2$, $k_3$). Це спрощує реалізацію алгоритму на сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються по модулю 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція *f* виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються по модулю 4. Операція множення на кроці 1 виконується по модулі $2^{32}$-1. Спеціальний випадок для даного алгоритму: якщо другий операнд дорівнює $2^{32}$-1, результат теж дорівнює $2^{32}$-1. В алгоритмі використовуються наступні константи:

$C$ = 2aaaaaaa, $c_0$ = 025f1cdb, $c_1 = 2*c_0$, $c_2=2^3 *c_0$, $c_3=2^7 *c_0$.

Константа $C$ – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи $c_0$ є інші особливі характеристики. Константи $c_1$, $c_2$ і $c_3$ – зрушені версії $c_0$, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість $c_i$ використовується $c_i^{-1}$. Значення $c_0^{-1}$ = 0dad4694.

# 5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функціональних кнопок ПЗ: Обробка документів; Організаційна структура; Моніторинг; Бізнес-процес; Налаштування; Авторство.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші: Журнал роботи.

– Верхнього меню: Файл; Експертна система; Аналіз; Довідка.

– Розділу обрання групи.

– Розділу виведення результату роботи системи.

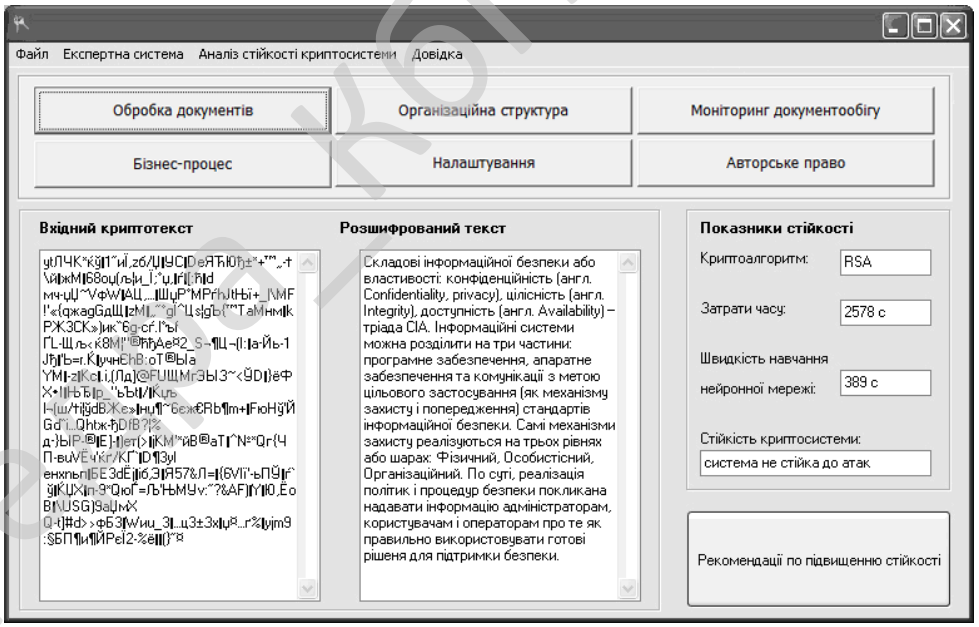


Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з

користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

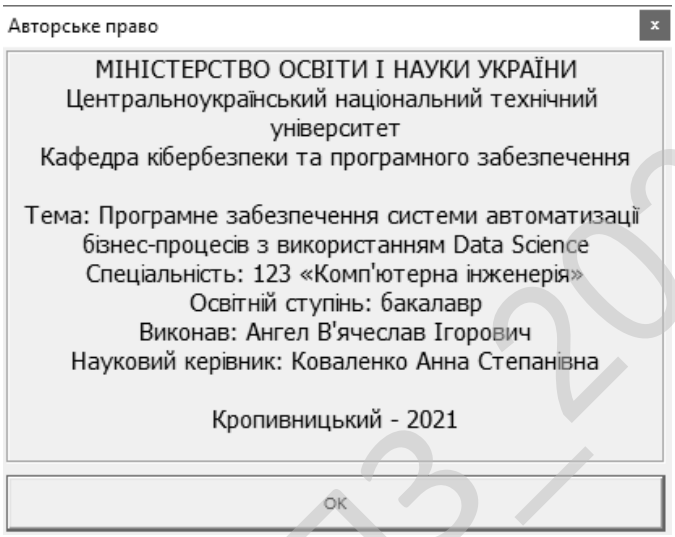


Рисунок 5.2 – Авторське право

Процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

# 6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи автоматизації бізнес-процесів з використанням Data Science.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем автоматизації бізнес-процесів з використанням Data Science.

– Досліджена система автоматизації бізнес-процесів з використанням Data Science.

– На основі отриманих результатів досліджень створена програмна реалізація системи автоматизації бізнес-процесів з використанням Data Science.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання автоматизації бізнес-процесів з використанням Data Science.

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 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи автоматизації бізнес-процесів з використанням Data Science. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення

ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ММВ.

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.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Искусственный интеллект. Справочник / Под ред. Д.А. Поспелова. – М.: Радио и связь, 1990. – 348 с.

2. Інформаційні технології. Взаємозв'язок відкритих систем. Базова еталонна модель. Частина 1. Базова модель (ISO/IEC 7498-1:1994, IDT):ДСТУ ISO/IEC 7498-1:2004 – [Чинний від 2006–01–01]. – Київ: Держспоживстандарт України, 2007. – 67 с. – (Національний стандарт України).

3. Карманов И.Н. Измерения, испытания, контроль. Метрология и метрологическое обеспечение: учеб. пособ. / И.Н. Карманов, Н.А. Мещеряков, О.К. Ушаков. – Новосибирск: СГГА, 2006. – 184 с.

4. Каспина Т.В. Экономика и управление приборостроительным произвоздством: учебн. пособ. для высших учебных заведений / Т.В. Каспина, Н.Н. Лямина. – М.: ИЦ "Академия", 2008. – 240 с.

5. Клюев В.В. Неразрушающий контроль и диагностика. Справочник, 2-е изд., перераб. и доп. / В.В. Клюев – М: Машиностроение, 2003. – 656 с.

6. Клюня В.Л. Основы экономической теории / В.Л. Клюня, Н.В. Черченко. – Минск: Минск, 2006. – 238 с.

7. Коваленко А.С. Разработка структуры базы данных интегрированной информационной системы / А.С. Коваленко, А.В. Коваленко // Информационные технологии и защита информации в информационно-коммуникационных системах: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – С. 54-64.

8. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

9. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

10. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

11. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

12. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

13. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

14. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

15. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

16. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

17. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

18. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

19. Коваленко А.С. Удосконалення методу технічного обслуговування об’єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

20. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

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

22. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація

озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

23. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

24. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

25. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

26. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

27. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28-31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

28. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

29. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

30. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

31. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

32. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: І всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

33. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (IT & I): II між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

34. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы /

А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

35. Королюк Н.А. Оценка временных интервалов работы лица, принимающего решение, на автоматизированном командном пункте / Н.А. Королюк, А.И. Тимочко // Системи обробки інформації. – Х.: ХУПС, 2005. – Вип. 8 (48). – С. 51-54.

36. Костерев В.В. Надёжность технических систем и управление риском: учебн. пособ. / В.В. Костерев. – М.: МИФИ, 2008. – 280 с.

37. Костюков А.В. Підвищення операційної ефективності підприємств на основі моніторингу в реальному часі. / А.В. Костюков, В.М. Костюков. – М.: Машинобудування, 2009. – 192 с.

38. Лазарев А.А. Выбор показателя затрат для анализа сравнительной экономической эффективности техники конечного потребления / А.А. Лазарев, М.В. Бейлин // Сборник научных трудов ХГПУ.– Х.: ХГПУ, 1999. – Вып. 74. – С. 27-29.

39. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 1. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 308 с.

40. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 2. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 208 с.

41. Лапсарь А.П. Метод оценки состояния сложных технических объектов для синтеза быстродействующих прогнозирующих систем / А.П. Лапсарь // Измерительная техника. – 2004. – № 2. – С. 7-10.

42. Линейные задачи оптимизации: Учеб. пособие / С.В. Лутманов. – Пермь: ЛИТЕР-А, 2004. – Ч.1. – Линейное программирование. – 128 с.

43. Литвак Б.Г. Экспертные технологии в управлении. Учебное пособие / Б.Г. Литвак. – М.: Дело, 2014. – 318 с.

44. Локазюк В.М. Надійність, контроль, діагностика і модернізація ПК: Посібн. / В.М. Локазюк, Ю.Г. Савченко. – К.: Видавничий центр «Академія», 2004. – 376 с.

45. Лопатников Л. И. Экономико-математический словарь: Словарь современной экономической науки / Л.И. Лопатников. – М.: Дело, 2003. – 520 с.

46. Манухина С.Ю. Инженерная психология и эргономика: хрестоматия / С.Ю Манухина. – М.: Изд. центр ЕАОИ, 2009. –224 с.

47. Мартыненко М.В. Человекомашинные процедуры поддержки организационно–управленческих решений: учеб. пособие СПбГЭТУ / М.В. Мартыненко, О.И. Шеховцов. – СПб, 2012. – 250 с.

48. Мунипов О.В. Эргономика: человекоориентированное проектирование техники, программных средств и среды: Учебник / О.В. Мунипов, В.П. Зинченко. – М.: Логос, 2001. – 356 с.

49. Надеев А.И. Математическая модель эксплуатационной надежности интеллектуальных датчиков / А.И. Надеев, Р.А. Юсупов, Ю.К. Свечников, Д.Р. Юсупов // Измерительная техника. – М: Стандартинформ, 2004. – № 1. – С. 8-11.

50. Надійність техніки. Аналіз надійності. Основні положення: ДСТУ 2861-94 – [Чинний від 1997–01–01]. – Київ: Держстандарт України, 1995. – 33 с. – (Національний стандарт України).

51. Надійність техніки. Терміни та визначення: ДСТУ 2860-94 – [Чинний від 1996–01–01]. – Київ: Держстандарт України, 1994. – 36 с. – (Національний стандарт України).

52. Нейлор К. Как построить свою экспертную систему / К. Нейлор. – М.: Энегроатомиздат, 2007. – 242 с.

Додаток А
(обов'язковий)

**Технічне завдання**

**Зміст**

## 1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи автоматизації бізнес-процесів з використанням Data Science.

## 2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 186-02 від 28.12.2020 року).

## 3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи автоматизації бізнес-процесів з використанням Data Science.

## 4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

### 5.2 Показники призначення

Система повинна забезпечувати:

– системи автоматизації бізнес-процесів з використанням Data Science;

– 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 XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

### 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 XP/Vista/7/8/10.

#### 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 Sydney.

### 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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 67 аркушів.

## 8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

## 11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 10.06.2021 р.

Додаток Б

(обов’язковий)

**Міністерство освіти і науки України**

**Центральноукраїнський національний технічний університет**

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Коваленко А.С.

***Програмне забезпечення системи автоматизації бізнес-процесів з використанням Data Science***

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 89

Літера: РП

Кропивницький – 2021 року

**Основна програма**

**Файл main.pas основної програми**

```
unit Main;

interface

{$I VER.INC}

// Підключення бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls, Buttons, HCMngr, ComCtrls, DECUtil, RNG;

// Опис змінних

type
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    MItemFile: TMenuItem;
    MItemStats: TMenuItem;
    MItemHashMemory: TMenuItem;
    MItemHashFile: TMenuItem;
    MItemExit: TMenuItem;
    P1: TPanel;
    Bevel1: TBevel;
    LHash: TLabel;
    EHashFile: TEdit;
    LInputfile: TLabel;
    BtnHashFile: TBitBtn;
    CBHash: TComboBox;
    LHAlgorithm: TLabel;
    LHashInfo: TLabel;
    LBase16: TLabel;
    EBase16: TEdit;
    LBase64: TLabel;
    EBase64: TEdit;
    BtnCalcHash: TBitBtn;
    OpenDialog: TOpenDialog;
    LHashTimes: TLabel;
    LHashTime: TLabel;
    LCipher: TLabel;
    Bevel2: TBevel;
    LCInput: TLabel;
    LCAlgorithm: TLabel;
    LCipherInfo: TLabel;
    LCKey: TLabel;
    LHashKey: TLabel;
    LCTimes: TLabel;
    LEncodeTime: TLabel;
    ECipherFile: TEdit;
    BtnInputFile: TBitBtn;
    CBCipher: TComboBox;
    EKey: TEdit;
    EHashKey: TEdit;
    BtnCipher: TBitBtn;
    CBCipherMode: TComboBox;
    LCMode: TLabel;
    LModeInfo: TLabel;
    LCipherHint: TLabel;
    BtnViewHashFile: TBitBtn;
    BtnViewCipherFiles: TBitBtn;
```

```
    LDTimes: TLabel;
    LDecodeTime: TLabel;
    LHashInput: TLabel;
    EHashInput: TEdit;
    EHashDEC: TEdit;
    LHashDEC: TLabel;
    EHashENC: TEdit;
    LHashENC: TLabel;
    N2: TMenuItem;
    MItemTestFile: TMenuItem;
    MItemTestRes: TMenuItem;
    N1: TMenuItem;
    MItemCipherMemory: TMenuItem;
    MItemCipherFile: TMenuItem;
    MItemMemCBC: TMenuItem;
    MItemMemCTS: TMenuItem;
    MItemMemCFB: TMenuItem;
    MItemMemOFB: TMenuItem;
    MItemMemECB: TMenuItem;
    MItemFileCTS: TMenuItem;
    MItemFileCBC: TMenuItem;
    MItemFileCFB: TMenuItem;
    MItemFileOFB: TMenuItem;
    MItemFileECB: TMenuItem;
    MItemHashVector: TMenuItem;
    MItemCipherVector: TMenuItem;
    Progress: TProgressBar;
    MItemExamples: TMenuItem;
    MItemPart: TMenuItem;
    MItemStrings: TMenuItem;
    MItemIV: TMenuItem;
    CipherManager: TCipherManager;
    HashManager: THashManager;
    OneTimePassword1: TMenuItem;
    HowuseTProtectionClasses1: TMenuItem;
    procedure MItemExitClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure CBHashClick(Sender: TObject);
    procedure BtnCalcHashClick(Sender: TObject);
    procedure BtnHashFileClick(Sender: TObject);
    procedure CBCipherClick(Sender: TObject);
    procedure CBCipherModeClick(Sender: TObject);
    procedure BtnViewHashFileClick(Sender: TObject);
    procedure EHashFileChange(Sender: TObject);
    procedure BtnInputFileClick(Sender: TObject);
    procedure BtnViewCipherFilesClick(Sender: TObject);
    procedure ECipherFileChange(Sender: TObject);
    procedure BtnCipherClick(Sender: TObject);
    procedure EKeyChange(Sender: TObject);
    procedure MItemTestFileClick(Sender: TObject);
    procedure MItemTestResClick(Sender: TObject);
    procedure MItemHashSpeedClick(Sender: TObject);
    procedure MItemCipherMemSpeedClick(Sender: TObject);
    procedure MItemCipherFileSpeedClick(Sender: TObject);
    procedure MItemHashVectorClick(Sender: TObject);
    procedure MItemCipherVectorClick(Sender: TObject);
    procedure ManagerProgress(Sender: TObject; Current, Maximal: Integer);
    procedure FormActivate(Sender: TObject);
    procedure MItemPartClick(Sender: TObject);
    procedure MItemStringsClick(Sender: TObject);
    procedure MItemIVClick(Sender: TObject);
    procedure OneTimePassword1Click(Sender: TObject);
    procedure HowuseTProtectionClasses1Click(Sender: TObject);
  private
    FTime: Comp;
    FViewer: String;
    FENCFile: String;
    FDECFile: String;
    FCreated: Boolean;
```

```
    function SelectFile(Edit: TEdit): Boolean;
    procedure ExecuteView(const FileName: String);
    function Counter(Size: Integer): String;
  public
  end;

var
  MainForm: TMainForm;

implementation

// Початок опису реалізацій функцій нейронного аналізу

uses ShellAPI, Hash, Cipher, ResFrm, MemSpd, ClipBrd, PCrypt, Strdemo,
     IVDemo, RFC2289, OTPDemo, GenForm;

{$R *.DFM}

// визначення розміру файлу, який дається на аналіз

function GetFileSize(const Filename: String): Integer;
var
  SR: TSearchRec;
begin
  if FindFirst(Filename, faAnyFile, SR) = 0 then Result := SR.Size
    else Result := -1;
  FindClose(SR);
end;

procedure TMainForm.ExecuteView(const FileName: String);
begin
  if FileExists(FileName) then
    ShellExecute(Handle, nil, PChar(FViewer), PChar(Filename), nil,
sw_ShowNormal);
end;

// Вибір файлу

function TMainForm.SelectFile(Edit: TEdit): Boolean;
begin
  OpenDialog.InitialDir := ExtractFilePath(Edit.Text);
  if OpenDialog.InitialDir = '' then OpenDialog.InitialDir :=
ExtractFilepath(ParamStr(0));
  Result := OpenDialog.Execute;
  if Result then Edit.Text := OpenDialog.FileName;
end;

function TMainForm.Counter(Size: Integer): String;
var
  S: Double;
begin
  if Size >= 0 then
  begin
    FTime := PerfCounter - FTime;
    S := FTime / PerfFreq;
    Result := FormatFloat('#,###0.0000 Секунд,  ', S) +
              FormatFloat('#0 ms,   ', S * 1000) +
              Format('%d байт, %s Kb розмір даних  ', [Size,
FormatFloat('#,##0.00', Розмір / 1024)]) +
              FormatFloat('са #,##0.00 Мб/с', (1 / S) * (Розмір / (1024 *
1024)));
  end else
  begin
    Result := '';
    FTime := PerfCounter;
  end;
end;

// обробка нажаття клавіші
```

```
procedure TMainForm.MItemExitClick(Sender: TObject);
begin
  Close;
end;

//Створення головної форми

procedure TMainForm.FormCreate(Sender: TObject);
begin
  if not DECUtil.InitTestIsOk then Caption := 'Init Test failed';
  FViewer := 'notepad.exe';
  FENCFile := ChangeFileExt(ParamStr(0), '.enc');
  FDECFile := ChangeFileExt(ParamStr(0), '.dec');
  EHashFile.Text := ParamStr(0);
  ECipherFile.Text := EHashFile.Text;

  HashNames(CBHash.Items);
  CipherNames(CBCipher.Items);
end;

procedure TMainForm.FormActivate(Sender: TObject);
begin

  if not FCreated then
  begin
    Update;
    CBHash.ItemIndex := 0;
    CBCipherMode.ItemIndex := 0;
    CBCipherModeClick(CBCipherMode);
    CBCipher.ItemIndex := 0;
    FCreated := True;
    CBHashClick(CBHash);
    CBCipherClick(CBCipher);
  end;
end;

procedure TMainForm.CBHashClick(Sender: TObject);
begin
  CBHash.ItemIndex := CBHash.ItemIndex;

{1. Варіант вибору алгоритму хеш функції для аналізу}
  HashManager.Algorithm := CBHash.Text;
  LHashInfo.Caption := HashManager.Description + ', ' +
                       HashManager.HashClass.ClassName;

{2.Варіант }
//  HashManager.HashClass := THashClass(CBHash.Items.Objects[CBHash.ItemIndex]);
//  LHashInfo.Caption := Format('%s, %d bit Digestsize',
//    [HashManager.HashClass.ClassName, HashManager.HashClass.DigestKeySize *
8]);

// Тестування хеш-функції на корректість
  try
    if not HashManager.HashClass.SelfTest then
      MessageBox(Handle, 'Самотестування не пройдене', 'Самотестування хеш-
функції', mb_Ok);
  except
    Application.HandleException(Self);
  end;
  BtnCalcHashClick(nil);
end;

procedure TMainForm.BtnCalcHashClick(Sender: TObject);
var
  FileSize: Integer;
//  Hash: THash;
//  HashClass: THashClass;
//  Digest: String;
```

```
//  Stream: TFileStream;
//  Buf: array[0..7] of Integer;
//  Len: Integer;
begin
  EKeyChange(nil);
  EBase16.Text := '';
  EBase64.Text := '';
  FileSize := GetFileSize(EHashFile.Text);
  if (FileSize >= 0) and FCreated then
  try
      Screen.Cursor := crHourglass;
      Application.ProcessMessages;
      Counter(-1);

    HashManager.CalcFile(EHashFile.Text);

      LHashTime.Caption := Counter(FileSize);
      EBase16.Text := HashManager.DigestString[fmtHEX];
      EBase64.Text := HashManager.DigestString[fmtMIME64];

  finally
    Screen.Cursor := crDefault;
  end;
end;

procedure TMainForm.BtnHashFileClick(Sender: TObject);
begin
  if SelectFile(EHashFile) then BtnCalcHashClick(nil);
end;

procedure TMainForm.BtnViewHashFileClick(Sender: TObject);
begin
  ExecuteView(EHashFile.Text);
end;

procedure TMainForm.EHashFileChange(Sender: TObject);
begin
  BtnViewHashFile.Enabled := FileExists(EHashFile.Text);
  BtnCalcHash.Enabled := FileExists(EHashFile.Text);
end;

procedure TMainForm.CBCipherClick(Sender: TObject);
begin
{скоректуємо вибір Display з Combobox де вибір з VK_UP або VK_DOWN}
  CBCipher.ItemIndex := CBCipher.ItemIndex;


{1. Варіант визначення шифру}
  CipherManager.Algorithm := CBCipher.Text;

  LCipherInfo.Caption := CipherManager.Description + ', ' +
                         CipherManager.CipherClass.ClassName;

{2.Варіант }
//  CipherManager.CipherClass :=
TCipherClass(CBCipher.Items.Objects[CBCipher.ItemIndex]);

//  LCipherInfo.Caption := Format('%s, %d bit MaxKeysize',
//    [CipherManager.CipherClass.ClassName, CipherManager.CipherClass.KeySize *
8]);


// Тестування шифру на коректність результату
  try
    if not CipherManager.CipherClass.SelfTest then
      MessageBox(Handle, 'Самотестування не пройдене', 'Самотестування алгоритму
шифрування', mb_Ok);
  except
// Abstract Error when TCipher.TestVector not анульовано
```

```
    Application.HandleException(Self);
  end;
  BtnCipherClick(nil);
end;

procedure TMainForm.CBCipherModeClick(Sender: TObject);
const
  sMode : array[TCipherMode] of String =
    ('Шифрування тексту', 'Шифрування ланцюжка блоків ', 'Шифрування зі
зворотнім зв’язком ',
     'Зворотній зв’язк по виходу ', 'Електроний кодовий блокнот', 'CBC MAC',
'CTS MAC', 'CFB MAC');
begin
  CipherManager.Mode := TCipherMode(CBCipherMode.ItemIndex);
  LModeInfo.Caption := sMode[CipherManager.Mode];
  BtnCipherClick(nil);
end;

procedure TMainForm.BtnInputFileClick(Sender: TObject);
begin
  if SelectFile(ECipherFile) then BtnCipherClick(nil);
end;

procedure TMainForm.BtnViewCipherFilesClick(Sender: TObject);
begin
  ExecuteView(ECipherFile.Text);
  ExecuteView(FENCFile);
  ExecuteView(FDECFile);
end;

procedure TMainForm.ECipherFileChange(Sender: TObject);
begin
  BtnViewCipherFiles.Enabled := FileExists(ECipherFile.Text);
  BtnCipher.Enabled := FileExists(ECipherFile.Text);
end;

procedure TMainForm.EKeyChange(Sender: TObject);
begin
{Автоматичне коректування Display, значення хеш  Key}
{Використовуємо вибраний HashClass, це тотожне для вибору CipherManager для
кодування / декодування}
  EHashKey.Text := HashManager.HashClass.CalcString(EKey.Text, nil, fmtHEX);

{ Це показує закодований Hashvalue}
{
  CipherManager.InitKey(EKey.Text, nil);
  EHashKey.Text := CipherManager.Cipher.Hash.DigestBase16;
}
end;

procedure TMainForm.BtnCipherClick(Sender: TObject);
var
  FileSize: Integer;
begin
  EHashInput.Text := '';
  EHashENC.Text   := '';
  EHashDEC.Text   := '';
  FileSize := GetFileSize(ECipherFile.Text);
  if (FileSize > 0) and FCreated then
  try
      Screen.Cursor := crHourGlass;
      Application.ProcessMessages;

// ініціалізуємо ключ
    CipherManager.InitKey(EKey.Text, nil);
      Counter(-1);
// Декодуємо вхідний файл до файлу навчання Demo.enc
    CipherManager.EncodeFile(ECipherFile.Text, FENCFile);
      LEncodeTime.Caption := Counter(FileSize);
```

```
// ініціалізуємо ключ
    CipherManager.InitKey(EKey.Text, nil);
//  CipherManager.InitKey(EKey.Text + 'Bad Key', nil);

//  Замість CipherManager.InitKey потрібно
//  CipherManager.Cipher.Done;
      Counter(-1);
// Декодуємо Demo.enc до Demo.dec
    CipherManager.DecodeFile(FENCFile, FDECFile);

      LDecodeTime.Caption := Counter(FileSize);

// Перевіряємо який  процесс хешування використовується
      EHashInput.Text := THash_MD4.CalcFile(ECipherFile.Text, nil, fmtDEFAULT);
      EHashENC.Text   := THash_MD4.CalcFile(FENCFile, nil, fmtDEFAULT);
      EHashDEC.Text   := THash_MD4.CalcFile(FDECFile, nil, fmtDEFAULT);
      if EHashInput.Text <> EHashDEC.Text then EHashDEC.Color := clRed
        else EHashDEC.Color := clBtnHighlight;
  finally
    Screen.Cursor := crDefault;
  end;
end;

// Підпрограмма тестування файла навчання

procedure TMainForm.MItemTestFileClick(Sender: TObject);
const
  BufSize = 1024 * 4;
var
  P: PByteArray;
  Start,Stop: Comp;
begin
  EHashFile.Text := ChangeFileExt(ParamStr(0), '.tst');
  ECipherFile.Text := EHashFile.Text;
  GetMem(P, BufSize);
  try
    Screen.Cursor := crHourGlass;
    with TFileStream.Create(EHashFile.Text, fmCreate) do
    try
      RND.Protection := TCipher_SCOP.Create('Пароль', nil);
      RND.Seed('', -1);  // Повністю випадковий
      Start := PerfCounter;
      repeat
        RND.Buffer(P^, BufSize); // Запвнюємо буфер випадковими даними
        Write(P^, BufSize);
      until Position >= 1024 * 1024;
      Stop := PerfCounter;
    finally
      Free;
      RND.Protection := nil; // Звільняємо від захисту
    end;
  finally
    Screen.Cursor := crDefault;
    FreeMem(P, BufSize);
  end;
  Start := Stop - Start;
  Stop := PerfFreq;
  MessageDlg('1Mb in ' + FloatToStr(Start / Stop) + ' Сектор заповнений
зашифрованими даними.',
             mtInformation, [mbOk], 0);
  EHashFileChange(EHashFile);
  ECipherFileChange(ECipherFile);
  BtnCalcHashClick(nil);
  BtnCipherClick(nil);
end;

procedure TMainForm.MItemTestResClick(Sender: TObject);
begin
```

```
  with TCheckResForm.Create(Self) do
  try
    ShowModal;
  finally
    Free;
  end;
end;

// Підпрограма обробки швидкості хешування

procedure TMainForm.MItemHashSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(Sender = MItemHashMemory, True, cmECB);
end;

procedure TMainForm.MItemCipherMemSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(True, False, TCipherMode(TComponent(Sender).Tag));
end;

// Підпрограма обробки швидкості шифрування

procedure TMainForm.MItemCipherFileSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(False, False, TCipherMode(TComponent(Sender).Tag));
end;

// Визначаємо фрагмент якого коду обробляється

procedure MakeCodeFragment(const Data: String; Len: Integer);
var
  C: String;
  I: Integer;
begin
  C := '         MOV   EAX,OFFSET @Vector' + #13#10 +
       '         RET'                      + #13#10 +
       '@Vector: ';
  for I := 0 to Len -1 do
  begin
    if I mod 8 = 0 then
    begin
      if I > 0 then C := C + #13#10 + '         ';
      C := C + 'DB    ';
    end else C := C + ',';
    C := C + IntToHex(Byte(Data[I+1]), 3) + 'h';
  end;
  Clipboard.AsText := C;
end;

procedure TMainForm.MItemHashVectorClick(Sender: TObject);
{генеруємоTestVector для хеш та вставляємо Codefragment to Clipboard}
var
  Data,Caption: String;
begin
  with HashManager.HashClass do
  begin
    Data := CalcBuffer(GetTestVector^, 32, nil, fmtCOPY);
    MakeCodeFragment(Data, DigestKeySize);
    Caption := 'Тестовий вектор для ' + ClassName;
    Data    := StrToFormat(PChar(Data), DigestKeySize, fmtHEX);
    MessageBox(Handle, PChar(Data), PChar(Caption), mb_Ok);
  end;
end;

procedure TMainForm.MItemCipherVectorClick(Sender: TObject);
{ генеруємо TestVector для шифру та вставляємо Codefragment до буферу обміну}
```

```
var
  Data, Caption: String;
begin
  with CipherManager.CipherClass.Create('', nil) do
  try
    Data := ClassName;
    Mode := cmCTS;
    Init(PChar(Data)^, Length(Data), nil);
    SetLength(Data, 32);
    EncodeBuffer(GetTestVector^, PChar(Data)^, 32);
    MakeCodeFragment(Data, 32);
    Caption := 'Тестовий вектор для ' + ClassName;
    Data    := StrToFormat(PChar(Data), 32, fmtHEX);
    MessageBox(Handle, PChar(Data), PChar(Caption), mb_Ok);
  finally
    Free;
  end;
end;

procedure TMainForm.ManagerProgress(Sender: TObject; Current, Maximal: Integer);
begin
{Визначаємо шифр або хеш
  TCipher_xxx.En/DecodeFile(), TCipher_xxx.En/DecodeStream()
  THash_xxx.CalcStream(), THash_xxx.CalcFile()}
{$IFDEF VER_D3H}
  Progress.Max := Maximal;
  Progress.Position := Current;
{$ELSE}
  if Maximal <= 0 then Progress.Position := 0
    else Progress.Position := Trunc(Progress.Max / Maximal * Current)
{$ENDIF}
{finished is by Current = 0 and Maximal = 0}
end;

// Процедури обробки натискання клавіш

procedure TMainForm.MItemPartClick(Sender: TObject);
begin
  with TPartForm.Create(Self) do Show;
end;

procedure TMainForm.MItemStringsClick(Sender: TObject);
begin
  with TStringForm.Create(Self) do Show;
end;

procedure TMainForm.MItemIVClick(Sender: TObject);
begin
  with TIVForm.Create(Self) do Show;
end;

procedure TMainForm.OneTimePassword1Click(Sender: TObject);
begin
  with TOTPForm.Create(Self) do Show;
end;
procedure TMainForm.HowuseTProtectionClasses1Click(Sender: TObject);
begin
  with TGForm.Create(Self) do Show;
end;
end.
```

**GenForm.pas – Побудова форм та основних обробників клавіш**

```
unit GenForm;

interface

// Підключення бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Menus, ComCtrls, DECUtil, Hash, Cipher, RNG, RFC2289, ShellAPI,
  Sample, Cipher1;

// Опис головного об'єкту

type
  TGForm = class(TForm)
    MainMenu: TMainMenu;
    HashMAC: TMenuItem;
    M: TRichEdit;
    THashXXX: TMenuItem;
    MACwithRFC1: TMenuItem;
    ViewRFC2202html1: TMenuItem;
    N1: TMenuItem;
    N2: TMenuItem;
    UsingfromHashs1: TMenuItem;
    File1: TMenuItem;
    Exit1: TMenuItem;
    N3: TMenuItem;
    MItemFormats: TMenuItem;
    TCipherXXX: TMenuItem;
    TRandomXXX: TMenuItem;
    UsingfromCiphers1: TMenuItem;
    N4: TMenuItem;
    CipherMAC: TMenuItem;
    TransactionNumbersTANs1: TMenuItem;
    UsingfromRandoms1: TMenuItem;
    procedure HashMACClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure MACwithRFC2104Click(Sender: TObject);
    procedure ViewRFC2202html1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure UsingfromHashs1Click(Sender: TObject);
    procedure UsingfromCiphers1Click(Sender: TObject);
    procedure TransactionNumbersTANs1Click(Sender: TObject);
    procedure CipherMACClick(Sender: TObject);
    procedure UsingfromRandoms1Click(Sender: TObject);
  private
    Format: Integer; // використовує формат рядка
    procedure DoInfo(const Value: String; Color: TColor);
    procedure FormatClick(Sender: TObject);
  public
  end;

var
  GForm: TGForm;

implementation

{$R *.DFM}
const
  sSelfTest : array[Boolean] of String = ('failed', 'success');

//Початок роботи підпрограми

procedure TGForm.DoInfo(const Value: String; Color: TColor);
begin // Показуємо Value в Color в Richedit
  M.SelStart := MaxInt div 16;
```

```
  M.SelLength := 0;
  M.SelAttributes.Color := Color;
  M.Lines.Add(Value);
  M.SelAttributes.Color := clWindowText;
  M.Perform(em_ScrollCaret, 0, 0);
  M.Update;
end;

// Обробник закриття форм

procedure TGForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TGForm.FormatClick(Sender: TObject);
begin
  with TMenuItem(Sender) do
  begin
    Checked := True;
    Format := Tag;
    DoInfo('Строка Displayformat змінена на: ' + Caption, clRed);
  end;
end;

procedure TGForm.FormCreate(Sender: TObject);
var
  I: Integer;
  S: TStringList;
  FMT: TStringFormatClass;
  MI: TMenuItem;
begin
// Встановлюємо внутрішній стан
  Format := fmtHEXVIEW;

  M.HandleNeeded;
  M.Paragraph.Tab[0] := 90;

  S := TStringList.Create;
  try
    GetStringFormats(S);
    for I := 0 to S.Count-1 do
    begin
      FMT := TStringFormatClass(S.Objects[I]);
      if (FMT.Format = fmtCOPY) or
         (FMT.Format = fmtSAMPLE) then Continue;
      MI := TMenuItem.Create(MItemFormats);
      MI.Caption := FMT.Name;
      MI.Tag := FMT.Format;
      MI.OnClick := FormatClick;
      MI.RadioItem := True;
      MI.Checked := FMT.Format = Format;
      MItemFormats.Add(MI);
    end;
  finally
    S.Free;
  end;
end;

procedure TGForm.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TGForm.UsingfromHashs1Click(Sender: TObject);
var
  HUser: THashClass;
  S: String;
  Buffer: array[0..127] of Byte;
```

```
  I: Integer;
  Stream: TStream;
  Cipher: TCipher;
begin
  M.Clear;
  HUser := THash_RipeMD128; // змінюємо для інших хеш-функцій

  for I := Low(Buffer) to High(Buffer) do Buffer[I] := I; // встановлюємо Buffer

  DoInfo('Використовується хеш', clRed);
  DoInfo('Користувач визнчів хеш: ' + GetHashName(HUser), clMaroon);
  DoInfo('Початок криптоаналізу хешу користувача', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//---------------------------------------------------------------------------
  DoInfo('1. Індивідуальні дані з ParamStr(0), DEMO.EXE', clBlue);


  S := THash_MD5.CalcFile(ParamStr(0), nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcFile(ParamStr(0), nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('2. Індивідуальні дані з строки, "Test"', clBlue);

  S := THash_MD5.CalcString('Тест', nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcString('Тест', nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('3. Індивідуальні дані з буферу', clBlue);

  S := THash_MD5.CalcBuffer(Buffer, SizeOf(Buffer), nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcBuffer(Buffer, SizeOf(Buffer), nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('4. Індивідуальні дані з TStream, 1024 Bytes from DEMO.EXE at Position
123', clBlue);

  Stream := TFileStream.Create(ParamStr(0), fmOpenRead or fmShareDenyNone);
  try
    Stream.Position := 123;
    S := THash_MD5.CalcStream(Stream, 1024, nil, Format);
    DoInfo('MD5'#9+S, clWindowText);

    Stream.Position := 123;
    S := HUser.CalcStream(Stream, 1024, nil, Format);
    DoInfo('HUser'#9+S, clWindowText);

  finally
    Stream.Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('5. Використовувати любу хеш-функцію', clBlue);

  with THash_MD5.Create(nil) do
  try
    Init;
// встановлюємо Initial Digest, XOR's a Passphrase з Digest, повинно бути після
виклику Init
```

```
    S := 'Пароль';
    for I := 0 to Length(S)-1 do
      PByteArray(DigestKey)[I mod DigestKeySize] := PByteArray(DigestKey)[I mod
DigestKeySize] xor Byte(S[I+1]);

    Calc(Buffer[ 0], 16);   // розраховуємо перші 16 байтів з Buffer
    Calc(Buffer[33], 15);   // розраховуємо з Buffer[33] 15 байт
    for I := 1 to 11 do     // розраховуємо 11 станів для "Проміжний пароль"
      Calc('DEC Частина I', 10);
    Calc(Buffer[99],  20);
    Done;

    S := DigestStr(Format);
    DoInfo('MD5'#9+S, clWindowText);

  finally
    Free;
  end;
// для добавлення хеш функцій на криптаналіз
  with HUser.Create(nil) do
  try
    Init;
// встановлюємо Initial Digest, XOR's a Passphrase з Digest, повинно бути після
Init
    S := 'Пароль';
    for I := 0 to Length(S)-1 do
      PByteArray(DigestKey)[I mod DigestKeySize] := PByteArray(DigestKey)[I mod
DigestKeySize] xor Byte(S[I+1]);

    Calc(Buffer[ 0], 16);   // розраховуємо перші 16 Байт з буфера
    Calc(Buffer[33], 15);   // розраховуємо з  Buffer[33] 15 байт
    for I := 1 to 11 do
      Calc('DEC Частина I', 10);
    Calc(Buffer[99],  3);
    Done;

    S := DigestStr(Format);
    DoInfo('HUser'#9+S, clWindowText);

  finally
    Free;
  end;
//---------------------------------------------------------------------------
  DoInfo('6. використовуємо TProtection метод для хеш', clBlue);

  with THash_MD5.Create(nil) do
  try
//---------------------------------------------------------------------------
// використовуємо MD5 кодування декодування:
// розраховуємо перші, Initialseed S0 (Password), рахуємо S0->S1->S2 та так
далі,//
    DoInfo('MD5 зашифровано, PlainText: "ваш текст наступний», Пароль «DEC"',
clGreen);
    Protection := TMAC_RFC2104.Create('DEC', nil); // Ваш пароль

    S := CodeString('Ваш текст наступний', paEncode, Format);
    DoInfo('encoded'#9+S, clWindowText);

    S := CodeString(S, paDecode, Format);
    DoInfo('decoded'#9+S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('MD5 зашифрований, PlainText: "Ваш текст наступний», Пароль «DED"',
clGreen);
    Protection := TMAC_RFC2104.Create('DED', nil); // ваш пароль
//    Protection := TCipher_Blowfish.Create('DED', nil);

    S := CodeString('Ваш текст наступний', paEncode, Format);
    DoInfo('1. encoded'#9+S, clWindowText);
```

```
    S := CodeString(S, paDecode, Format);
    DoInfo('1. decoded'#9+S, clWindowText);

// обратно зашифрований paEncode/paDecode обміном,
// при заміні захисту на Blowfish ви побачите цей результат
    S := CodeString('Ваш текст наступний', paDecode, fmtCOPY); // Формат повинен
бути fmtCOPY
    DoInfo('2. encoded'#9+StrToFormat(PChar(S), Length(S), Format),
clWindowText);

    S := CodeString(S, paEncode, fmtCopy);
    DoInfo('2. decoded'#9+S, clWindowText);

//---------------------------------------------------------------------------
// paScramble, це одношляхова функція
    DoInfo('MD5 Scramble, Data: "Проміжні дані"', clGreen);
    Protection := TMAC.Create('DEC Scramble', nil); // ваш пароль

    S := CodeString('Проміжні дані', paScramble, Format);
    DoInfo('1. scramble'#9+S, clWindowText);
    S := CodeString('Проміжні дані', paScramble, Format);
    DoInfo('2. scramble'#9+S, clWindowText);

//---------------------------------------------------------------------------
// paWipe, - один з видів Function (paScramble) для видачі усіх інших
результатів
// Захист не потрібен при використанні корректних CodeBuffer, CodeFile,
CodeStream
// Для того, щоб убрати слабкі параметри використовується CodeString()

    DoInfo('MD5 Взломано, Data: "Дані взломані"', clGreen);
    Protection := nil;

    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('1. wiped'#9+S, clWindowText);
    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('2. wiped'#9+S, clWindowText);
    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('3. wiped'#9+S, clWindowText);

//---------------------------------------------------------------------------
// розраховуємо MD5 Fingerprint над Blowfish зашифрований DEMO.EXE
// THash_MD5.CalcFile() с  Blowfish шифром розраховується
// MD5 автентифікатор.
// Цей метод Взламуванняує DEMO.EXE, розраховуючи MD5 та Взламуванняує MD5 Final
Digest
    DoInfo('MD5 розраховується над Blowfish Взламуванняючи DEMO.EXE', clGreen);

    Protection := TCipher_Blowfish.Create('DEC', nil);
    CodeFile(ParamStr(0), '', paCalc);

    DoInfo('MD5-Digest'#9+DigestStr(Format), clWindowText);

//---------------------------------------------------------------------------
// розрахуємо MD5-HMAC над Blowfish Взламуванняаним рядком
    DoInfo('MD5 розрахований над Blowfish Взламуванняаним рядком ', clGreen);
// використовуємо TProtection методи для побудови ланцюжка
    Protection := TCipher_Blowfish.Create('DEC Частина I', nil);
    CodeString('Teststring', paCalc, fmtNONE); // fmtNONE = no Stringconvert

    DoInfo('CodeString()'#9+DigestStr(Format), clWindowText);
//---------------------------------------------------------------------------
    Protection := nil;
    Cipher := TCipher_Blowfish.Create('', nil);
    try
      // ініціалізуємо шифр та Взламуванняуємо рядок
      Cipher.InitKey('DEC Частина I', nil);
      S := Cipher.EncodeString('Teststring');
```

```
      Cipher.Done;
      // розраховуємо MD5 на зашифрованим рядком
      Init;                        // ініціалізуємо MD5
      Calc(PChar(S)^, Length(S)); // розраховуємо MD5
      Done;                        // MD5
      // Взламуванняуємо MD5 повідомлення
      Cipher.EncodeBuffer(DigestKey^, DigestKey^, DigestKeySize);

      DoInfo('conventional'#9+DigestStr(Format), clWindowText);
    finally
      Cipher.Free;
    end;

// CodeBuffer(), CodeStream() та CodeFile()
    Free; // знищуємо MD5
  end;

end;

procedure TGForm.HashMACClick(Sender: TObject);
var
  S,FileName: String;
  MAC: TMAC;
  Protection: TProtection;
  HUser: THashClass;
begin
  M.Clear;

  HUser := THash_Haval192; // вибираємо хеш функцію для Взламуванняу
  FileName := ParamStr(0); // вибираємо файл для Взламуванняу

  DoInfo('Хеш повідомлення автентифікаційного коду', clRed);
  DoInfo('Користувач визначає код як: ' + GetHashName(HUser), clMaroon);
  DoInfo('Простий тест на злам функції', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('SHA1:'#9 + sSelfTest[THash_SHA1.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('1. Generic THash_MD5(TMAC) -> MAC-MD5', clBlue);

  S := THash_MD5.CalcFile(FileName, TMAC.Create('DEC Частина I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC Частина I"'#9+S, clWindowText);

  S := THash_MD5.CalcFile(FileName, TMAC.Create('DEC Частина I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC Частина I"'#9+S, clWindowText);


//-----------------------------------------------------------------------------

  MAC := TMAC.Create('DEC', nil);
  try
    MAC.AddRef;  //

  //-----------------------------------------------------------------------------
    DoInfo('2. Загальний TMAC -> використовує TMAC Instance,
THash_XXX(TMAC("DEC"))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
```

```
    DoInfo('3. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", TCipher_Blowfish(«Зашифрований текст»)))', clBlue);
// визначіть  Blowfish Protection, це кодує фінальний Hash.DigestKey
    MAC.Protection := TCipher_Blowfish.Create('Зашифрований текст', nil);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  Визвольте Blowfish MAC Protection й визначіть  TRandom_LFSR захищений
//  TRandom_LFSR з періодом 2^400-1, дивіться RNG.pas якщо потрібна додаткова
інформація
    MAC.Protection := TRandom_LFSR.Create('Зашифрований текст', 400, False,
nil);

    DoInfo('4. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", TRandom_LFSR(«Зашифрований текст»)))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  визвольте LFSR MAC Protection и визначіть  THash_MD4(TMAC) захист
// a Double HMAC -> HMAC-MD5-HMAC-MD4
    MAC.Protection := THash_MD4.Create(TMAC.Create('Зашифрований текст', nil));
// Ланцюжок: THash_XXX -> TMAC('DEC') -> THash_MD4 -> TMAC('Зашифрований текст')
    DoInfo('5. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_MD4(TMAC(«Зашифрований текст»))))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  Change Password
    MAC.Protection.Protection := TMAC.Create('Зашифрований текст', nil);

    DoInfo('6. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_MD4(TMAC(«Зашифрований текст»))))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
// встановіть  MAC.Protection to THash_SHA1(TMAC(«Зашифрований текст»))
// a HMAC-MD5-HMAC-SHA1
```

```
    MAC.Protection := THash_SHA1.Create(TMAC.Create('Зашифрований текст', nil));

    DoInfo('7. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_SHA1(TMAC(«Зашифрований текст»)))))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

  finally
// реліз  MAC Instance та відлінкуйте Protections (THash_MD4 -> TMAC);
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('8. polymorph MAC -> THash_XXX(TCipher_Blowfish(«Зашифрований
текст»))', clBlue);

  Protection := TCipher_Blowfish.Create('Зашифрований текст', nil);
  try
    Protection.AddRef; // це загальний ресурс
    Protection.AddRef;
    // MD5-Blowfish-CTS-MAC
    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);
    // SHA1-Blowfish-CTS-MAC
    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release; // double AddRef -> double Release
    Protection.Release; // визвольте Cipher
  end;

//---------------------------------------------------------------------------
  DoInfo('9. поліморфичний MAC -> THash_XXX(TRandom_LFSR(«Зашифрований
текст»))', clBlue);

  Protection := TRandom_LFSR.Create('Зашифрований текст', 2032, False, nil); //
Period 2^2032-1 see RNG.pas for Details
  try
    Protection.AddRef; // це загальний ресурс

    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release; // визвольте Random
  end;

//---------------------------------------------------------------------------
  DoInfo('10. поліморфичний MAC -> THash_XXX(TMAC("DEC"))', clBlue);
  DoInfo('Результати повинні бути такі ж як Step 2.', clMaroon);

  Protection := TMAC.Create('DEC', nil);
  try
    Protection.AddRef; // це загальний ресурс
```

```
    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release;
  end;

end;

procedure TGForm.MACwithRFC2104Click(Sender: TObject);

  function RepKey(Value, Count: Integer): String;
  begin
    SetLength(Result, Count);
    FillChar(PChar(Result)^, Count, Value);
  end;

var
  HUser: THashClass;
  MAC: TMAC;
  Data: array[1..50] of Byte;
  S: String;
  I: Integer;
  Stream: TMemoryStream;
begin
  M.Clear;

  HUser := DefaultHashClass;

  DoInfo('RFC2104 стандарт HMAC', clRed);
  DoInfo('Користувач визначає код як: ' + GetHashName(HUser), clMaroon);
  DoInfo(' MACs використовує Testcases з RFC2202, дивись Docus\RFC2202.html',
clMaroon);
  DoInfo('Це сірі значення з RFC2202.html', clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Hashs', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('SHA1:'#9 + sSelfTest[THash_SHA1.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('Testcase No. 1', clBlue); // Test, використовує інший Key's для
кожного Thing

  S := THash_MD5.CalcString('Тут ', TMAC_RFC2104.Create(RepKey($0B, 16), nil),
fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'9294727a3638bb1c13f48ef8158bfc9d', clGrayText);

  S := THash_SHA1.CalcString('Тут ', TMAC_RFC2104.Create(RepKey($0B, 20), nil),
fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
  DoInfo(#9'b617318655057264e28bc0b6fb378c8ef146be00', clGrayText);

  S := HUser.CalcString('Тут ', TMAC_RFC2104.Create(RepKey($0B, 20), nil),
fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('Testcase No. 2', clBlue);

  MAC := TMAC_RFC2104.Create('Jefe', nil);
  try
    MAC.AddRef;
```

```
    S := THash_MD5.CalcString('нічого не робити?', MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'750c783e6ab0b503eaa86e310a5db738', clGrayText);

    S := THash_SHA1.CalcString('нічого не робити?', MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'effcdf6ae5eb2fa2d27416d5f184df9c259a7c79', clGrayText);

    S := HUser.CalcString('нічого не робити?', MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 3', clBlue);

  FillChar(Data, SizeOf(Data), $DD);

  S := THash_MD5.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA,
16), nil), fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'56be34521d144c88dbb8c733f0e8b3f6', clGrayText);

  S := THash_SHA1.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA,
20), nil), fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
  DoInfo(#9'125d7342b9ac11cd91a39af48aa17b4f63f175d3', clGrayText);

  S := HUser.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA, 20),
nil), fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 4', clBlue);

  FillChar(Data, SizeOf(Data), $CD);
  SetLength(S, 25);
  for I := 1 to 25 do Byte(S[I]) := I;

  MAC := TMAC_RFC2104.Create(S, nil);
  try
    MAC.AddRef;

    S := THash_MD5.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'697eaf0aca3a3aea3a75164746ffaa79', clGrayText);

    S := THash_SHA1.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'4c9007f4026250c6bc8414f9bf50c86c2d7235da', clGrayText);

    S := HUser.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 5', clBlue);

  S := THash_MD5.CalcString('Test With Truncation',
TMAC_RFC2104.Create(RepKey($0C, 16), nil), fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'56461ef2342edc00f9bab995690efd4c', clGrayText);
  SetLength(S, 96 div 8 * 2);  // 96 Біт розділених по 8 біт * 2 Chars для байта
```

```
  DoInfo('HMAC-MD5-96'#9+S, clWindowText);
  DoInfo(#9'56461ef2342edc00f9bab995', clGrayText);

  S := THash_SHA1.CalcString(«Тест з зкругленням»,
TMAC_RFC2104.Create(RepKey($0C, 20), nil), fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
  DoInfo(#9'4c1a03424b55e07fe7f27be1d58bb9324a9a5a04', clGrayText);
  SetLength(S, 96 div 8 * 2);
  DoInfo('HMAC-SHA1-96'#9+S, clWindowText);
  DoInfo(#9'4c1a03424b55e07fe7f27be1', clGrayText);

  S := HUser.CalcString(«Тест з зкругленням», TMAC_RFC2104.Create(RepKey($0C,
20), nil), fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);
  SetLength(S, 96 div 8 * 2);
  DoInfo('HMAC-HUser-96'#9+S, clWindowText);

//  Tests використовує Stream
  Stream := TMemoryStream.Create;
  MAC    := TMAC_RFC2104.Create(RepKey($AA, 80), nil);
  try
    MAC.AddRef;

//-----------------------------------------------------------------------------
    DoInfo('Пакет тестування нейромережею № 6', clBlue);

    Stream.Write('Test Using Larger Than Block-Size Key - Hash Key First', 54);
// не повинно використовуватися Stream.Position, використовується StreamSize = -
1, THash_XXX manage the Seeking
    S := THash_MD5.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'6b1ab7fe4bd7bf8f0b62e6ce61b9d0cd', clGrayText);

    S := THash_SHA1.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'aa4ae5e15272d00e95705637ce8a3b55ed402112', clGrayText);

    S := HUser.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
    DoInfo('Пакет тестування нейромережею № 7', clBlue);

    Stream.Size := 0;
    Stream.Write('Тест нейронною мережю використовує Larger Than Block-Size Key
and Larger Than One Block-Size Data', 73);
// Нейронною мережею встановлюється Stream.Position, we використовує a
StreamSize = Stream.Size
    Stream.Position := 0;
    S := THash_MD5.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'6f630fad67cda0ee1fb1f562db3aa53e', clGrayText);

    Stream.Position := 0;
    S := THash_SHA1.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'e8e99d0f45237d786d6bbaa7965c7808bbff1a91', clGrayText);

    Stream.Position := 0;
    S := HUser.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
    Stream.Free;
  end;

end;
```

```
procedure TGForm.ViewRFC2202html1Click(Sender: TObject);
var
  S: String;
begin
  S := ExtractFilePath(ParamStr(0));
  SetLength(S, Length(S)-1);
  S := ExtractFilePath(S) + 'Docus\RFC2202.html';
  ShellExecute(Handle, nil, PChar(S), nil, nil, sw_ShowNormal);
end;

procedure TGForm.TransactionNumbersTANs1Click(Sender: TObject);
const
  HashTAN : THashClass = THash_SHA1;
  maxTANEntries = 10; // TAN List have 10 Numbers

// робить короткий рядок
  function FoldStr(const Value: String): String;
  const
    maxLen = 8; // робить не більш коротке чим 6, 6 байт – це тільки 2^48
комбінацій !
  var
    I,Len: Integer;
  begin
    Result := Value;
    Len := Length(Result);
    for I := 1 to Len do
      Byte(Result[I]) := Byte(Result[I]) xor Byte(Result[(I + maxLen) mod Len]);
    SetLength(Result, maxLen);
  end;

// Складає Ліст шифрів для клієнта
  function CreateTANList(const SeedTANList, SeedTAN, Name: String; ID: Integer;
var LastTAN: String): TStringList;
  type
    PClient = ^TClient;
    TClient = packed record
                Name: array[0..80] of Char; // Імя клієнта
                ID: Integer;                // ID клієнта
                Seed: array[0..64] of Char;
                TANCount: Integer;          // лічильник TAN lists
              end;
   var
     Client: TClient;
     S: String;
     I: Integer;
   begin
// складаємо лист шифрів
     Result := TStringList.Create;
// устанавлюємо Client Infos
     FillChar(Client, Sizeof(Client), 0);
     StrPLCopy(Client.Name, AnsiUpperCase(Trim(Name)), SizeOf(Client.Name));
     Client.ID := ID;
     Client.TANCount := 1;

 // Розраховуються безпечні параметри клієнта з параметрів серверу S0
     S := FormatToStr(PChar(SeedTANList), -1, Format);
     I := ID;
     repeat
       S := HashTAN.CalcString(S, nil, fmtCOPY);
       Dec(I);
     until I <= 0;
     StrPLCopy(Client.Seed, S, SizeOf(Client.Seed));
     S := HashTAN.CalcBuffer(Client, SizeOf(Client), nil, fmtCOPY);
     I := maxTANEntries;
     repeat
       S := HashTAN.CalcString(S, nil, fmtCOPY);
       S := FoldStr(S);
       Result.Insert(0, StrToFormat(PChar(S), Length(S), Format));
       Dec(I);
```

```
    until I <= 0;
    S := HashTAN.CalcString(S, nil, fmtCOPY);
    S := FoldStr(S);
    S := S + FormatToStr(PChar(SeedTAN), -1, Format);
    LastTAN := HashTAN.CalcString(S, nil, Format);
  end;

// перевіряємо CurrentTAN з LastTAN/SeedTAN та записуємо наступний LastTAN
  function CheckTAN(const SeedTAN, LastTAN: String; var CurrentTAN: String):
Boolean;
  var
    C,L,S: String;
    I: Integer;
  begin
    try
      S := FormatToStr(PChar(SeedTAN), -1, Format);
      C := FormatToStr(PChar(CurrentTAN), -1, Format);
      L := FormatToStr(PChar(LastTAN), -1, Format);
      I := maxTANEntries; // max. TAN List Count
      repeat
        C := HashTAN.CalcString(C, nil, fmtCOPY);
        C := FoldStr(C);
// розраховуємо коректний TAN
        Result := HashTAN.CalcString(C + S, nil, fmtCOPY) = L;
        Dec(I);
      until Result or (I <= 0);
      C := FormatToStr(PChar(CurrentTAN), -1, Format) + S;
      CurrentTAN := HashTAN.CalcString(C, nil, Format);
    except
      Result := False;
      Application.HandleException(nil);
    end;
  end;

var
  SeedTANList, SeedTAN: String;
  TANList: TStringList;
  I: Integer;
  LastTAN: String;
  TAN: String;
begin
  M.Clear;
  DoInfo('Кількість транзакцій для визначення паролю ', clRed);
  DoInfo('Hash алгоритм це: ' + GetHashName(HashTAN), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Hashs', clBlue);
  DoInfo('HashTAN:'#9 + sSelfTest[HashTAN.SelfTest], clWindowText);
  DoInfo('будуємо сервер S0', clBlue);

  SeedTANList := HashTAN.CalcString('Пароль серверу "Sample BANK of Ukraine"',
nil, Format);
  SeedTAN     := SeedTANList; //

  DoInfo('S0:'#9+SeedTANList, clWindowText);

  DoInfo('будуємо TAN list для "Matvienko Tatiyana"', clBlue);

  TANList := CreateTANList(SeedTANList, SeedTAN, 'Matvienko Tatiyana', 54,
LastTAN);

  try
// На сервері нейронної мережі побудована база даних з полями:
// ClientID та Last використовують TAN
// запам’ятовуємо перший TAN (LastTAN) в базі даних нейронної мережі

// Для Клієнта
    DoInfo('TAN список клієнта:', clWindowText);
    for I := 0 to TANList.Count-1 do
      DoInfo(IntToStr(I)+ ':'#9+TANList[I], clWindowText);
```

```
    DoInfo('Нейронна мережа зробила транзакцію', clBlue);

// 1. TA ---------------------------------------------------------------------
    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('Current TAN:'#9 + TAN, clWindowText);

// Clients записується  TAN та Client ID надається в сервер нейронної мережі
// Server шукає в Database Client ID, достає LastTAN та
// перевіряє TAN
    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN добрий ', clGreen);
// зберігаємо поточний TAN в Database та останній клієнтський TAN
      LastTAN := TAN;
      DoInfo('останній TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;
    DoInfo('', clWindowText);

// 2. TA ---------------------------------------------------------------------
    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('поточний TAN:'#9 + TAN, clWindowText);
    Delete(TAN, 1, 4);
    DoInfo('Поганий TAN:'#9 + TAN, clMaroon);
// on the Server, check the TAN
    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN нормальний', clGreen);
      LastTAN := TAN;
      DoInfo('Останній TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;
    DoInfo('', clWindowText);

// 3. TA ---------------------------------------------------------------------
    TANList.Delete(0); TANList.Delete(0);

    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('Current TAN:'#9 + TAN, clWindowText);

/    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN добрий ', clGreen);
// Зберігаємо поточний TAN в Database використовуємо останній TAN
      LastTAN := TAN;
      DoInfo('Last TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;

  finally
    TANList.Free;
  end;
end;

procedure TGForm.UsingfromCiphers1Click(Sender: TObject);
const
  sMode: array[TCipherMode] of String = ('cmCTS', 'cmCBC', 'cmCFB', 'cmOFB',
                                         'cmECB', 'cmCTSMAC', 'cmCBCMAC',
'cmCFBMAC');
var
  CUser: TCipherClass;
  Buffer: array[0..15] of Byte;
  I: Integer;
```

```
  S: String;
  Stream: TMemoryStream;
  K: TCipherMode;
begin
  M.Clear;
  CUser := TCipher_Blowfish; // вибираємо шифр для Взламуванняу

  for I := Low(Buffer) to High(Buffer) do Buffer[I] := I + 32; // setup Buffer

  DoInfo('Шифр обрано', clRed);
  DoInfo('Користувач визначив шифр як : ' + GetCipherName(CUser), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Ciphers', clBlue);
  DoInfo('CUser:'#9 + sSelfTest[CUser.SelfTest], clWindowText);
  DoInfo('Blowfish:'#9 + sSelfTest[TCipher_Blowfish.SelfTest], clWindowText);
  DoInfo('IDEA:'#9 + sSelfTest[TCipher_IDEA.SelfTest], clWindowText);
  DoInfo('GOST:'#9 + sSelfTest[TCipher_GOST.SelfTest], clWindowText);


  with CUser.Create('', nil) do
  try
//-----------------------------------------------------------------------------
    DoInfo('1. Шифрування/дешифрування файлу, ParamStr(0), DEMO.EXE', clBlue);

    InitKey('DEC', nil);
    EncodeFile(ParamStr(0), ChangeFileExt(ParamStr(0), '.ENC'));
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Done;
//    InitKey('DEC', nil);

    DecodeFile(ChangeFileExt(ParamStr(0), '.ENC'), ChangeFileExt(ParamStr(0),
'.DEC'));
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect; //

//-----------------------------------------------------------------------------
    DoInfo('2. Шифрування/дешифрування рядка, "Тест шифрування рядка"', clBlue);
    InitKey('DEC Частина I', nil);

    S := EncodeString('Тест дешифрування рядка');
    DoInfo('Encryted:'#9+ StrToFormat(PChar(S), Length(S), Format),
clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
    Done;

    S := DecodeString(S);

    DoInfo('Decryted:'#9+ S, clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect;

//-----------------------------------------------------------------------------
    DoInfo('3. Шифрування/дешифрування буфера, "' +StrToFormat(@Buffer,
Sizeof(Buffer), Format) + '"', clBlue);
    InitKey('DEC Частина I', nil);

    EncodeBuffer(Buffer, Buffer, SizeOf(Buffer));
    DoInfo('Шифрування:'#9+ StrToFormat(@Buffer, Sizeof(Buffer), Format),
clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
    Done;

    DecodeBuffer(Buffer, Buffer, SizeOf(Buffer));

    DoInfo('Взламування нейронною мережею:'#9+ StrToFormat(@Buffer,
Sizeof(Buffer), Format), clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
```

```
    Protect;

//---------------------------------------------------------------------------
    DoInfo('4. Шифрування/дешифрування потоку, "Partial Stream En/Decryption"',
clBlue);
    InitKey('DEC Частина I', nil);

    Stream := TMemoryStream.Create;
    try
      S := 'Partial Stream En/Decryption';
      Stream.Write(PChar(S)^, Length(S));

      Stream.Position := 8;
      EncodeStream(Stream, Stream, 6);

      DoInfo('Шифрування:'#9+ StrToFormat(Stream.Memory, Stream.Size, Format),
clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
      Done;

      Stream.Position := 8;
      DecodeStream(Stream, Stream, 6);

      DoInfo('Взламування нейронною мережею:'#9+ StrToFormat(Stream.Memory,
Stream.Size, Format), clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    finally
      Stream.Free;
    end;

//---------------------------------------------------------------------------
    DoInfo('5. Різні режими шифрування', clBlue);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      InitKey('DEC', nil);
      S := EncodeString('Тест нейронною мережею зашифрованого рядка');
      DoInfo('Шифрування:'#9+ StrToFormat(PChar(S), Length(S), Format),
clWindowText);

      Done;

      S := DecodeString(S);
      DoInfo('Взламування нейронною мережею:'#9+ StrToFormat(PChar(S),
Length(S), Format), clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('6. Використовувати TProtection-Method CodeString() with any
Protection', clBlue);

  with CUser.Create('', nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('Без шифрування', clBlue);

    InitKey('DEC', nil);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;
```

```
      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взламування нейронною мережею:'#9+ S, clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено  TRandom_LFSR("DEC")', clBlue);
    Protection := TRandom_LFSR.Create('DEC', 400, False, nil);
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взламування нейронною мережею:'#9+ S, clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено THash_MD5(TMAC_RFC2104("DEC"))', clBlue);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('DEC', nil));
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взламування нейронною мережею:'#9+ S, clWindowText);
    end;

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('7. Використовуємо TProtections Methods', clBlue);

  with CUser.Create('', nil) do
  try
    HashClass := THash_MD5;  // установлюємо інші HashClass for the InitKey()
    InitKey('DEC', nil);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paEncode/paDecode)', clGreen);

    S := CodeString('CodeString()', paEncode, Format);
    DoInfo('Шифрування:'#9+ S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Взламування нейронною мережею:'#9+ S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paScramble)', clGreen);

    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);
```

```
    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);

//--------------------------------------------------------------------------
    DoInfo('CodeString(paWipe)', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);

  finally
    Free;
  end;
//--------------------------------------------------------------------------
  DoInfo('8. A secure зашифрований ', clBlue);
//  demonstrate a Multi-En/Decryption that використовує 3 Ciphers in a Chain.

  with TCipher_Blowfish.Create('DEC',
         TCipher_IDEA.Create('DEC',
           TCipher_GOST.Create('DEC',
             TRandom_LFSR.Create('Scramble', 128, False, nil)))) do
  try
    S := CodeString('Добрий DEC Частина I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

procedure TGForm.CipherMACClick(Sender: TObject);
const
  sMode: array[TCipherMode] of String = ('CTS-MAC', 'CBC-MAC', 'CFB-MAC',
                                         'invalid', 'invalid',
                                         'CTS-MAC', 'CBC-MAC', 'CFB-MAC');
var
  CUser: TCipherClass;
  K: TCipherMode;
  I: Integer;
begin
  M.Clear;
  CUser := TCipher_Blowfish; // вибираємо шифр для Взламуванняу

  DoInfo('Посилається автентифікаційний код з шифром', clRed);
  DoInfo('Користувач визначив шифр як : ' + GetCipherName(CUser), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Ciphers', clBlue);
  DoInfo('CUser:'#9 + sSelfTest[CUser.SelfTest], clWindowText);
  DoInfo('SCOP:'#9 + sSelfTest[TCipher_SCOP.SelfTest], clWindowText);

//--------------------------------------------------------------------------
  DoInfo('1. MAC для ParamStr(0), DEMO.EXE', clBlue);
  with CUser.Create('DEC', nil) do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() у MAC режимі: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() у MAC режимі: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
```

```
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('2. MAC для ParamStr(0), DEMO.EXE Захищено Blowfish(«Зашифрований
текст»)', clBlue);
  with CUser.Create('DEC', TCipher_Blowfish.Create('Зашифрований текст', nil))
do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() у MAC режимі: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() у MAC режимі: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('3. MAC''s with TProtection Method CodeString', clBlue);

  with CUser.Create('DEC', nil) do
  try
// визначіть  HMAC-MD5-LFSR128-SCOP protection :-)
//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                      TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для MD5', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                       TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для LFSR', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
```

```
                        TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//------------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                       TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;
  finally
    Free;
  end;
 CodeBuffer(), CodeString()
//   CodeStream(), CodeFile()
// - with Action = paCalc it's the Result from CodeString()
end;

procedure TGForm.UsingfromRandoms1Click(Sender: TObject);
var
  I: Integer;
  S,SaveState: String;
  Buf: array[0..15] of Byte;
begin
  M.Clear;

  DoInfo('Random using', clRed);
//------------------------------------------------------------------------------
  DoInfo('1. TRandom_LFSR Instance з періодом 2^400-1', clBlue);

  with TRandom_LFSR.Create('', 400, False, nil) do
  try
//------------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC Частина I"', clBlue);
    Seed('DEC Частина I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//------------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC Частина I"', clBlue);
    Seed('DEC Частина I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//------------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, default Seed', clBlue);
    Seed('', 0);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);
```

```
//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1000 to 1000, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('Change Period to 2^2032-1', clGreen);
    Size := 2032;

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1 to 1, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed', clBlue);
    Seed('', 0);

    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed зі збереженням у State', clBlue);
    DoInfo('Ключ на період  2^128-1', clGreen);
    Size := 128;
    Protection := TCipher_Blowfish.Create('DEC', nil);
    Seed('', -1);          SaveState := State;
    DoInfo('SaveState:' + InsertBlocks(SaveState, #9, #10, 64), clGreen);
// випадковий Buffer
    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, SizeOf(Buf), Format), clWindowText);

    Seed('', -1); //
    DoInfo('Стан відновлено з SaveState', clGreen);

    State := SaveState;
    SetLength(S, Sizeof(Buf));
    Buffer(PChar(S)^, Length(S));

    DoInfo(StrToFormat(PChar(S), Length(S), Format), clWindowText);

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('2. Глобальна випадкова змінна "RND"', clBlue);
// RND is per default TRandom_LFSR('', 128);
  RND.Seed('', 0);
```

```
  RND.Buffer(Buf, SizeOf(Buf));
  DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
  DoInfo('3. TProtection методи з TRandom_LFSR', clBlue);
  with TRandom_LFSR.Create('DEC', 128, False, nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('Кодування / декодування нейронною мережею з TRandom', clGreen);

    S := CodeString('Добрий DEC Частина I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('Утаємничення з TRandom', clGreen);

    S := CodeString('Добрий DEC Частина I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

    DoInfo('BasicSeed changed from: '+ SysUtils.Format('$%0.8x to $%0.8x',
[BasicSeed, BasicSeed +1]), clGreen);
    BasicSeed := BasicSeed +1; // використовує інший BasicSeed
    S := CodeString('Добрий DEC Частина I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('Взломано з TRandom', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
    S := CodeString('Добрий DEC Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

end.
```

**Chiper.pas – файл реалізації алгоритмів для автоматизації бізнес-процесів з використанням Data Science нейронною мережею**

```
unit Cipher;

interface

{$I VER.INC}

uses SysUtils, Classes, DECUtil, Hash;

const {Коди помилок}
  errGeneric        = 0;  {Помилка генерації}
  errInvalidKey     = 1;  {Ключ декодування некоректний}
  errInvalidKeySize = 2;  {Розмір ключа дуже великий}
  errNotInitialized = 3;  {Методи Init() або InitKey() не викликаються}
  errInvalidMACMode = 4;  {CalcMAC не повертає  cmECB, cmOFB}
  errCantCalc       = 5;

type
  ECipherException = class(Exception)
  public
    ErrorCode: Integer;
  end;

{Перелік алгоритмів для крипто аналізу у вигляді классів}
  TCipher_Gost         = class;
  TCipher_Blowfish     = class;
  TCipher_IDEA         = class;
  TCipher_SAFER        = class;
  TCipher_SAFER_K40    = class;
  TCipher_SAFER_SK40   = class;
  TCipher_SAFER_K64    = class;
  TCipher_SAFER_SK64   = class;
  TCipher_SAFER_K128   = class;
  TCipher_SAFER_SK128  = class;
  TCipher_TEA          = class;
  TCipher_TEAN         = class;
  TCipher_SCOP         = class;
  TCipher_Q128         = class;
  TCipher_3Way         = class;
  TCipher_Twofish      = class;
  TCipher_Shark        = class;
  TCipher_Square       = class;

  TCipherMode = (cmCTS, cmCBC, cmCFB, cmOFB, cmECB, cmCTSMAC, cmCBCMAC,
cmCFBMAC);
{ Режими шифрування:
  cmCTS     Шифрування тексту
  cmCBC     Шифрування ланцюжка блоків
  cmCFB     K-bit Шифрування зі зворотнім зв'язком
  cmOFB     K-bit Зворотній зв'язк по виходу
  cmECB *   Electronic Codebook

  cmCTSMAC  Message Authentication Code в режимі cmCTS
  cmCBCMAC – CBC-MAC
  cmCFBMAC –  CFB-MAC
}

  TCipherClass = class of TCipher;

  TCipher = class(TProtection)
  private
    FMode: TCipherMode;
    FHash: THash;
    FHashClass: THashClass;
    FKeySize: Integer;
    FBufSize: Integer;
```

```
    FUserSize: Integer;
    FBuffer: Pointer;
    FVector: Pointer;
    FFeedback: Pointer;
    FUser: Pointer;
    FFlags: Integer;
    function GetHash: THash;
    procedure SetHashClass(Value: THashClass);
    procedure InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
    procedure InternalCodeFile(const Source, Dest: String; Encode: Boolean);
  protected
    function GetFlag(Index: Integer): Boolean;
    procedure SetFlag(Index: Integer; Value: Boolean); virtual;
{використовуються в методі Init()}
    procedure InitBegin(var Size: Integer);
    procedure InitEnd(IVector: Pointer); virtual;
{ анульовано}
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
virtual;
    class function TestVector: Pointer; virtual;
{анульовано TProtection Methods}
    procedure CodeInit(Action: TPAction); invalidated;
    procedure CodeDone(Action: TPAction); invalidated;
    procedure CodeBuf(var Buffer; const BufferSize: Integer; Action: TPAction);
invalidated;
{ анульовано}
    procedure Encode(Data: Pointer); virtual;
{після декодування функції анульовано}
    procedure Decode(Data: Pointer); virtual;
    property User: Pointer read FUser;
    property Buffer: Pointer read FBuffer;
    property UserSize: Integer read FUserSize;
  public
    constructor Create(const Password: String; AProtection: TProtection);
    destructor Destroy; invalidated;
    class function MaxKeySize: Integer;
{тест на коректність роботи}
    class function SelfTest: Boolean;
{ініціалізація форм для шифрування}
    procedure Init(const Key; Size: Integer; IVector: Pointer); virtual;
    procedure InitKey(const Key: String; IVector: Pointer);
    procedure Done; virtual;
    procedure Protect; virtual;

    procedure EncodeBuffer(const Source; var Dest; DataSize: Integer);
    procedure DecodeBuffer(const Source; var Dest; DataSize: Integer);
    function  EncodeString(const Source: String): String;
    function  DecodeString(const Source: String): String;
    procedure EncodeFile(const Source, Dest: String);
    procedure DecodeFile(const Source, Dest: String);
    procedure EncodeStream(const Source, Dest: TStream; DataSize: Integer);
    procedure DecodeStream(const Source, Dest: TStream; DataSize: Integer);

    function CalcMAC(Format: Integer): String;

{Cipher Mode = cmXXX}
    property Mode: TCipherMode read FMode write FMode;
{ поточний Hash-Object, будує Digest з InitKey()}
    property Hash: THash read GetHash;
{ Class Hash-Object}
    property HashClass: THashClass read FHashClass write SetHashClass;
{максимальний розмір ключа та буфера }
    property KeySize: Integer read FKeySize;
    property BufSize: Integer read FBufSize;

{Init() повинно визиватися}
    property Initialized: Boolean index 1 read GetFlag write SetFlag;
    property Vector: Pointer read FVector;
```

```
    property Feedback: Pointer read FFeedback;
    property HasHashKey: Boolean index 0 read GetFlag;
  end;

// Опис шифрів

  TCipher_Gost = class(TCipher) {російський шифр}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Blowfish = class(TCipher)
  private
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
    procedure Encode386(Data: Pointer);
    procedure Decode386(Data: Pointer);
  {$ENDIF}
{$ENDIF}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_IDEA = class(TCipher) {International Data Encryption Algorithm }
  private
    procedure Cipher(Data, Key: PWordArray);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TSAFERMode = (smDefault, smK40, smK64, smK128, smStrong, smSK40, smSK64,
smSK128);

  TCipher_SAFER = class(TCipher)
  private
    FRounds: Integer;
    FSAFERMode: TSAFERMode;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure InitNew(const Key; Size: Integer; IVector: Pointer; SAFERMode:
TSAFERMode);
    property Rounds: Integer read FRounds write SetRounds;
  end;
```

```
  TCipher_SAFER_K40 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK40 = class(TCipher_SAFER_K40)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K64 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK64 = class(TCipher_SAFER_K64)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K128 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK128 = class(TCipher_SAFER_K128)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_TEA = class(TCipher) {Tiny Encryption Algorithm}
  private
    FRounds: Integer;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    property Rounds: Integer read FRounds write SetRounds;
  end;

  TCipher_TEAN = class(TCipher_TEA) {Tiny Encryption Algorithm, extended
Version}
  protected
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
```

```
  end;

  TCipher_SCOP = class(TCipher) {Stream Cipher in Blockmode}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure Done; invalidated;
  end;

  TCipher_Q128 = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_3Way = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Twofish = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Shark = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Square = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;
```

```
function DefaultCipherClass: TCipherClass;
procedure SetDefaultCipherClass(CipherClass: TCipherClass);
procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
function UnregisterCipher(const ACipher: TCipherClass): Boolean;
function CipherList: TStrings;
procedure CipherNames(List: TStrings);
function GetCipherClass(const Name: String): TCipherClass;
function GetCipherName(CipherClass: TCipherClass): String;

const
  CheckCipherKeySize: Boolean = False;

implementation

uses DECConst, Windows;

{$I *.inc}
{$I Square.inc}

const
  FDefaultCipherClass : TCipherClass = TCipher_Blowfish;
  FCipherList         : TStringList  = nil;

function DefaultCipherClass: TCipherClass;
begin
  Result := FDefaultCipherClass;
end;

procedure SetDefaultCipherClass(CipherClass: TCipherClass);
begin
  if CipherClass = nil then FDefaultCipherClass := TCipher_Blowfish
    else FDefaultCipherClass := CipherClass;
end;

procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
var
  E: ECipherException;
begin
  E := ECipherException.Create(Msg);
  E.ErrorCode := ErrorCode;
  raise E;
end;

function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
var
  I: Integer;
  S: String;
begin
  Result := False;
  if ACipher = nil then Exit;
  S := Trim(AName);
  if S = '' then
  begin
    S := ACipher.ClassName;
    if S[1] = 'T' then Delete(S, 1, 1);
    I := Pos('_', S);
    if I > 0 then Delete(S, 1, I);
  end;
  S := S + '=' + ADescription;
  I := CipherList.IndexOfObject(Pointer(ACipher));
  if I < 0 then CipherList.AddObject(S, Pointer(ACipher))
    else CipherList[I] := S;
  Result := True;
end;

function UnregisterCipher(const ACipher: TCipherClass): Boolean;
```

```
var
  I: Integer;
begin
  Result := False;
  repeat
    I := CipherList.IndexOfObject(Pointer(ACipher));
    if I < 0 then Break;
    Result := True;
    CipherList.Delete(I);
  until False;
end;

function CipherList: TStrings;
begin
  if not IsObject(FCipherList, TStringList) then FCipherList :=
TStringList.Create;
  Result := FCipherList;
end;

procedure CipherNames(List: TStrings);
var
  I: Integer;
begin
  if not IsObject(List, TStrings) then Exit;
  for I := 0 to CipherList.Count-1 do
    List.AddObject(FCipherList.Names[I], FCipherList.Objects[I]);
end;

function GetCipherClass(const Name: String): TCipherClass;
var
  I: Integer;
  N: String;
begin
  Result := nil;
  N := Name;
  I := Pos('_', N);
  if I > 0 then Delete(N, 1, I);
  for I := 0 to CipherList.Count-1 do
    if AnsiCompareText(N, GetShortClassName(TClass(FCipherList.Objects[I]))) = 0
then
    begin
      Result := TCipherClass(FCipherList.Objects[I]);
      Exit;
    end;
  I := FCipherList.IndexOfName(N);
  if I >= 0 then Result := TCipherClass(FCipherList.Objects[I]);
end;

function GetCipherName(CipherClass: TCipherClass): String;
var
  I: Integer;
begin
  I := CipherList.IndexOfObject(Pointer(CipherClass));
  if I >= 0 then Result := FCipherList.Names[I]
    else Result := GetShortClassName(CipherClass);
end;

function TCipher.GetFlag(Index: Integer): Boolean;
begin
  Result := FFlags and (1 shl Index) <> 0;
end;

procedure TCipher.SetFlag(Index: Integer; Value: Boolean);
begin
  Index := 1 shl Index;
  if Value then FFlags := FFlags or Index
    else FFlags := FFlags and not Index;
end;
```

```
procedure TCipher.InitBegin(var Size: Integer);
begin
  Initialized := False;
  Protect;
  if Size < 0 then Size := 0;
  if Size > KeySize then
    if not CheckCipherKeySize then Size := KeySize
      else RaiseCipherException(errInvalidKeySize, Format(sInvalidKeySize,
[ClassName, 0, KeySize]));
end;

procedure TCipher.InitEnd(IVector: Pointer);
begin
  if IVector = nil then Encode(Vector)
    else Move(IVector^, Vector^, BufSize);
  Move(Vector^, Feedback^, BufSize);
  Initialized := True;
end;

class procedure TCipher.GetContext(var ABufSize, AKeySize, AUserSize: Integer);
begin
  ABufSize := 0;
  AKeySize := 0;
  AUserSize := 0;
end;

class function TCipher.TestVector: Pointer;
begin
  Result := GetTestVector;
end;

procedure TCipher.Encode(Data: Pointer);
begin
end;

procedure TCipher.Decode(Data: Pointer);
begin
end;

constructor TCipher.Create(const Password: String; AProtection: TProtection);
begin
  inherited Create(AProtection);
  FHashClass := DefaultHashClass;
  GetContext(FBufSize, FKeySize, FUserSize);
  GetMem(FVector, FBufSize);
  GetMem(FFeedback, FBufSize);
  GetMem(FBuffer, FBufSize);
  GetMem(FUser, FUserSize);
  Protect;
  if Password <> '' then InitKey(Password, nil);
end;

destructor TCipher.Destroy;
begin
  Protect;
  ReallocMem(FVector, 0);
  ReallocMem(FFeedback, 0);
  ReallocMem(FBuffer, 0);
  ReallocMem(FUser, 0);
  FHash.Release;
  FHash := nil;
  inherited Destroy;
end;

class function TCipher.MaxKeySize: Integer;
var
  Dummy: Integer;
begin
  GetContext(Dummy, Result, Dummy);
```

```
end;

class function TCipher.SelfTest: Boolean;
var
  Data: array[0..63] of Char;
  Key: String;
  SaveKeyCheck: Boolean;
begin
  Result       := InitTestIsOk;
  Key          := ClassName;
  SaveKeyCheck := CheckCipherKeySize;
  with Self.Create('', nil) do
  try
    CheckCipherKeySize := False;
    Mode := cmCTS;
    Init(PChar(Key)^, Length(Key), nil);
    EncodeBuffer(GetTestVector^, Data, 32);
    Result := Result and (MemCompare(TestVector, @Data, 32) = 0);
    Done;
    DecodeBuffer(Data, Data, 32);
    Result := Result and (MemCompare(GetTestVector, @Data, 32) = 0);
  finally
    CheckCipherKeySize := SaveKeyCheck;
    Free;
  end;
  FillChar(Data, SizeOf(Data), 0);
end;

procedure TCipher.Init(const Key; Size: Integer; IVector: Pointer);
begin
end;

procedure TCipher.InitKey(const Key: String; IVector: Pointer);
var
  I: Integer;
begin
  Hash.Init;
  Hash.Calc(PChar(Key)^, Length(Key));
  Hash.Done;
  I := Hash.DigestKeySize;
  if I > FKeySize then I := FKeySize;
  Init(Hash.DigestKey^, I, IVector);
  EncodeBuffer(Hash.DigestKey^, Hash.DigestKey^, Hash.DigestKeySize);
  Done;
  SetFlag(0, True);
end;

procedure TCipher.Done;
begin
  if MemCompare(FVector, FFeedback, FBufSize) = 0 then Exit;
  Move(FFeedback^, FBuffer^, FBufSize);
  Move(FVector^, FFeedback^, FBufSize);
end;

procedure TCipher.Protect;
begin
  SetFlag(0, False);
  Initialized := False;
//!!
  FillChar(FVector^, FBufSize, $AA);
  FillChar(FFeedback^, FBufSize, $AA);
  FillChar(FBuffer^, FBufSize, $AA);
  FillChar(FUser^, FUserSize, $AA);

  FillChar(FVector^, FBufSize, $55);
  FillChar(FFeedback^, FBufSize, $55);
  FillChar(FBuffer^, FBufSize, $55);
  FillChar(FUser^, FUserSize, $55);
```

```
  FillChar(FVector^, FBufSize, $FF);
  FillChar(FFeedback^, FBufSize, $FF);
  FillChar(FBuffer^, FBufSize, 0);
  FillChar(FUser^, FUserSize, 0);
end;

function TCipher.GetHash: THash;
begin
  if not IsObject(FHash, THash) then
  begin
    if FHashClass = nil then FHashClass := DefaultHashClass;
    FHash := FHashClass.Create(nil);
    FHash.AddRef;
  end;
  Result := FHash;
end;

procedure TCipher.SetHashClass(Value: THashClass);
begin
  if Value <> FHashClass then
  begin
    FHash.Release;
    FHash := nil;
    FHashClass := Value;
    if FHashClass = nil then FHashClass := DefaultHashClass;
  end;
end;

procedure TCipher.InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
const
  maxBufSize = 1024 * 4;
var
  Buf: PChar;
  SPos: Integer;
  DPos: Integer;
  Len: Integer;
  Proc: procedure(const Source; var Dest; DataSize: Integer) of object;
  Size: Integer;
begin
  if Source = nil then Exit;
  if Encode or (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) then Proc :=
EncodeBuffer
    else Proc := DecodeBuffer;
  if Dest = nil then Dest := Source;
  if DataSize < 0 then
  begin
    DataSize := Source.Size;
    Source.Position := 0;
  end;
  Buf := nil;
  Size := DataSize;
  DoProgress(Self, 0, Size);
  try
    Buf    := AllocMem(maxBufSize);
    DPos   := Dest.Position;
    SPos   := Source.Position;
    if Mode in [cmCTSMAC, cmCBCMAC, cmCFBMAC] then
    begin
      while DataSize > 0 do
      begin
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
```

```
    end else
      while DataSize > 0 do
      begin
        Source.Position := SPos;
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        SPos := Source.Position;
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
        Dest.Position := DPos;
        Dest.Write(Buf^, Len);
        DPos := Dest.Position;
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
  finally
    DoProgress(Self, 0, 0);
    ReallocMem(Buf, 0);
  end;
end;

procedure TCipher.InternalCodeFile(const Source, Dest: String; Encode: Boolean);
var
  S,D: TFileStream;
begin
  S := nil;
  D := nil;
  try
    if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then
    begin
      S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
      D := S;
    end else
      if (AnsiCompareText(Source, Dest) <> 0) and (Trim(Dest) <> '') then
      begin
        S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
        D := TFileStream.Create(Dest, fmCreate);
      end else
      begin
        S := TFileStream.Create(Source, fmOpenReadWrite);
        D := S;
      end;
    InternalCodeStream(S, D, -1, Encode);
  finally
    S.Free;
    if S <> D then
    begin
{$IFDEF VER_D3H}
      D.Size := D.Position;
{$ENDIF}
      D.Free;
    end;
  end;
end;

procedure TCipher.EncodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, True);
end;

procedure TCipher.DecodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, False);
end;

procedure TCipher.EncodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, True);
```

```
end;

procedure TCipher.DecodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, False);
end;

function TCipher.EncodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  EncodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

function TCipher.DecodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  DecodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

procedure TCipher.EncodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Encode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        while DataSize >= FBufSize do
        begin
          XORBuffers(S, FFeedback, FBufSize, D);
          Encode(D);
          XORBuffers(D, FFeedback, FBufSize, FFeedback);
          Inc(S, FBufSize);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(S, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCBC:
      begin
        F := FFeedback;
```

```
      while DataSize >= FBufSize do
      begin
        XORBuffers(S, F, FBufSize, D);
        Encode(D);
        F := D;
        Inc(S, FBufSize);
        Inc(D, FBufSize);
        Dec(DataSize, FBufSize);
      end;
      Move(F^, FFeedback^, FBufSize);
      if DataSize > 0 then
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        XORBuffers(S, FBuffer, DataSize, D);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
      end;
    end;
  cmCFB:
    while DataSize > 0 do
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      D^ := S^ xor PByte(FBuffer)^;
      Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
      PByteArray(FFeedback)[FBufSize-1] := D^;
      Inc(D);
      Inc(S);
      Dec(DataSize);
    end;
  cmOFB:
    while DataSize > 0 do
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      D^ := S^ xor PByte(FBuffer)^;
      Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
      PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
      Inc(D);
      Inc(S);
      Dec(DataSize);
    end;
  cmCTSMAC:
    begin
      while DataSize >= FBufSize do
      begin
        XORBuffers(S, FFeedback, FBufSize, FBuffer);
        Encode(FBuffer);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        Inc(S, FBufSize);
        Dec(DataSize, FBufSize);
      end;
      if DataSize > 0 then
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
      end;
    end;
  cmCBCMAC:
    begin
      while DataSize >= FBufSize do
      begin
        XORBuffers(S, FFeedback, FBufSize, FBuffer);
        Encode(FBuffer);
        Move(FBuffer^, FFeedback^, FBufSize);
        Inc(S, FBufSize);
        Dec(DataSize, FBufSize);
      end;
```

```
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFBMAC:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^ xor PByte(FBuffer)^;
        Inc(S);
        Dec(DataSize);
      end;
  end;
end;

procedure TCipher.DecodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F,B: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Decode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          XORBuffers(D, F, FBufSize, B);
          Decode(D);
          XORBuffers(D, F, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(FBuffer, D, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
```

```
        end;
      end;
    cmCBC:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          Move(D^, B^, FBufSize);
          Decode(D);
          XORBuffers(F, D, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(D, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^;
        D^ := S^ xor PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmOFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        D^ := S^ xor PByte(FBuffer)^;
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmCTSMAC, cmCBCMAC, cmCFBMAC:
      begin
        EncodeBuffer(Source, Dest, DataSize);
        Exit;
      end;
  end;
end;

procedure TCipher.CodeInit(Action: TPAction);
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
{  if (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) <> (Action = paCalc) then
    RaiseCipherException(errCantCalc, Format(sCantCalc, [ClassName]));}
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
```

```
  inherited CodeInit(Action);
end;

procedure TCipher.CodeDone(Action: TPAction);
begin
  inherited CodeDone(Action);
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
end;

procedure TCipher.CodeBuf(var Buffer; const BufferSize: Integer; Action:
TPAction);
begin
  if Action = paDecode then
  begin
    if Action in Actions then
      DecodeBuffer(Buffer, Buffer, BufferSize);
    inherited CodeBuf(Buffer, BufferSize, Action);
  end else
  begin
    inherited CodeBuf(Buffer, BufferSize, Action);
    if Action in Actions then
      EncodeBuffer(Buffer, Buffer, BufferSize);
  end;
end;

function TCipher.CalcMAC(Format: Integer): String;
var
  B: PByteArray;
begin
  if Mode in [cmECB, cmOFB] then
    RaiseCipherException(errInvalidMACMode, sInvalidMACMode);
  Done;
  B := AllocMem(FBufSize);
  try
    Move(FBuffer^, B^, FBufSize);
    EncodeBuffer(B^, B^, FBufSize);
    SetLength(Result, FBufSize);
    Move(FFeedback^, PChar(Result)^, FBufSize);
    if Protection <> nil then Result := Protection.CodeString(Result,
paScramble, Format)
      else Result := StrToFormat(PChar(Result), Length(Result), Format);
  finally
    ReallocMem(B, 0);
    Done;
  end;
end;

class procedure TCipher_Gost.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 32;
  AUserSize := 32;
end;

class function TCipher_Gost.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B3h,003h,0A0h,03Fh,0B5h,07Bh,091h,04Dh
         DB    097h,051h,024h,040h,0BDh,0CFh,025h,015h
         DB    034h,005h,09Ch,0F8h,0ABh,010h,086h,09Fh
         DB    0F2h,080h,047h,084h,047h,09Bh,01Ah,0D1h
end;

type
  PCipherRec = ^TCipherRec;
```

```
  TCipherRec = packed record
                 case Integer of
                   0: (X: array[0..7] of Byte);
                   1: (A, B: LongWord);
               end;

procedure TCipher_Gost.Encode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  K := User;
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := User;
    T := A + K[0];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[1];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Inc(PInteger(K), 2);
  end;
  K := @PIntArray(User)[6];
  for I := 0 to 3 do
  begin
    T := A + K[1];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[0];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Decode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  K := User;
  for I := 0 to 3 do
  begin
    T := A + K[0];
    B := B xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    T := B + K[1];
    A := A xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    Inc(PInteger(K), 2);
```

```
  end;
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := @PIntArray(User)[6];
    T := A + K[1];
    B := B xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    T := B + K[0];
    A := A xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  InitEnd(IVector);
end;

class procedure TCipher_Blowfish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 56;
  AUserSize := SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key);
end;

class function TCipher_Blowfish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    019h,071h,0CAh,0CDh,02Bh,09Ch,085h,029h
         DB    0DAh,081h,047h,0B7h,0EBh,0CEh,016h,0C6h
         DB    091h,00Eh,01Dh,0C8h,040h,012h,03Eh,035h
         DB    070h,0EDh,0BCh,096h,04Ch,013h,0D0h,0B8h
end;

type
  PBlowfish = ^TBlowfish;
  TBlowfish = array[0..3, 0..255] of LongWord;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
procedure TCipher_Blowfish.Encode386(Data: Pointer);
asm  // спеціально для CPU < 486
        PUSH   EDI
        PUSH   ESI
        PUSH   EBX
        PUSH   EBP
        PUSH   EDX

        MOV    ESI,[EAX].TCipher_Blowfish.FUser

        MOV    EBX,[EDX]         // A
        MOV    EDX,[EDX + 4]     // B

        XCHG   BL,BH       // Це BSWAP EBX,EDX
        XCHG   DL,DH
        ROL    EBX,16
        ROL    EDX,16
        XCHG   BL,BH
```

```
        XCHG    DL,DH

        XOR     EBX,[ESI + 4 * 256 * 4]
        XOR     EDI,EDI

@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP
        INC     EDI
        TEST    EDI,010h
        JZ      @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4 + 17 * 4]

        XCHG    BL,BH         // це BSWAP EBX,EDX
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;

procedure TCipher_Blowfish.Decode386(Data: Pointer);
asm // спеціально для CPU < 486
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser

        MOV     EBX,[EDX]          // A
        MOV     EDX,[EDX + 4]      // B

        XCHG    BL,BH
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        XOR     EBX,[ESI + 4 * 256 * 4 + 17 * 4]

        MOV     EDI,16
```

```
@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP

        DEC     EDI
        JNZ     @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4]

        XCHG    BL,BH         // BSWAP
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;
  {$ENDIF} //486GE
{$ENDIF}

procedure TCipher_Blowfish.Encode(Data: Pointer);
{$IFDEF UseASM}  // спеціально для CPU >= 486
asm
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser
        MOV     EBX,[EDX]          // A
        MOV     EBP,[EDX + 4]      // B

        BSWAP   EBX                // CPU >= 486
        BSWAP   EBP

        XOR     EDI,EDI
        XOR     EBX,[ESI + 4 * 256 * 4]
//      XOR     ECX,ECX
@@1:

        MOV     EAX,EBX
        SHR     EBX,16
        MOVZX   ECX,BH     // це прискорюється для AMD Chips,
//        MOV     CL,BH    // це прискорюється для PII's
        MOV     EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
```

```
//        MOV    CL,BL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX  ECX,AH
//        MOV    CL,AH
        XOR    EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX  ECX,AL
//        MOV    CL,AL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR    EBP,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        INC    EDI
        XOR    EDX,EBP
        TEST   EDI,010h
        MOV    EBP,EAX
        MOV    EBX,EDX
        JZ     @@1

        POP    EAX
        XOR    EBP,[ESI + 4 * 256 * 4 + 17 * 4]

        BSWAP  EBX
        BSWAP  EBP

        MOV    [EAX],EBP
        MOV    [EAX + 4],EBX

        POP    EBP
        POP    EBX
        POP    ESI
        POP    EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  A := SwapInteger(PCipherRec(Data).A) xor P[0]; Inc(PInteger(P));
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    B := B xor P[0] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
                         D[3, A        and $FF]);

    A := A xor P[1] xor (D[0, B shr 24        ] +
                         D[1, B shr 16 and $FF] xor
                         D[2, B shr  8 and $FF] +
                         D[3, B        and $FF]);
    Inc(PInteger(P), 2);
  end;
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
        PUSH   EDI
        PUSH   ESI
        PUSH   EBX
        PUSH   EBP
        PUSH   EDX
```

```
        MOV    ESI,[EAX].TCipher_Blowfish.FUser
        MOV    EBX,[EDX]          // A
        MOV    EBP,[EDX + 4]      // B

        BSWAP  EBX
        BSWAP  EBP

        XOR    EBX,[ESI + 4 * 256 * 4 + 17 * 4]
        MOV    EDI,16
//        XOR    ECX,ECX

@@1:    MOV    EAX,EBX
        SHR    EBX,16

        MOVZX  ECX,BH
//        MOV    CL,BH
        MOV    EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX  ECX,BL
//        MOV    CL,BL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX  ECX,AH
//        MOV    CL,AH
        XOR    EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX  ECX,AL
//        MOV    CL,AL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR    EBP,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR    EDX,EBP
        DEC    EDI
        MOV    EBP,EAX
        MOV    EBX,EDX
        JNZ    @@1

        POP    EAX
        XOR    EBP,[ESI + 4 * 256 * 4]

        BSWAP  EBX
        BSWAP  EBP

        MOV    [EAX],EBP
        MOV    [EAX + 4],EBX

        POP    EBP
        POP    EBX
        POP    ESI
        POP    EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key) -
SizeOf(Integer));
  A := SwapInteger(PCipherRec(Data).A) xor P[0];
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    Dec(PInteger(P), 2);
    B := B xor P[1] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
                         D[3, A        and $FF]);
    A := A xor P[0] xor (D[0, B shr 24        ] +
                         D[1, B shr 16 and $FF] xor
```

```
                        D[2, B shr  8 and $FF] +
                        D[3, B        and $FF]);
  end;
  Dec(PInteger(P));
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Init(const Key; Size: Integer; IVector: Pointer);
var
  I,J: Integer;
  B: array[0..7] of Byte;
  K: PByteArray;
  P: PIntArray;
  S: PBlowfish;
begin
  InitBegin(Size);
  K := @Key;
  S := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  Move(Blowfish_Data, S^, SizeOf(Blowfish_Data));
  Move(Blowfish_Key, P^, Sizeof(Blowfish_Key));
  J := 0;
  for I := 0 to 17 do
  begin
    P[I] := P[I] xor (K[(J + 0) mod Size] shl 24 +
                      K[(J + 1) mod Size] shl 16 +
                      K[(J + 2) mod Size] shl  8 +
                      K[(J + 3) mod Size]);
    J := (J + 4) mod Size;
  end;
  FillChar(B, SizeOf(B), 0);
  for I := 0 to 8 do
  begin
    Encode(@B);
    P[I * 2]     := SwapInteger(PCipherRec(@B).A);
    P[I * 2 + 1] := SwapInteger(PCipherRec(@B).B);
  end;
  for I := 0 to 3 do
    for J := 0 to 127 do
    begin
      Encode(@B);
      S[I, J * 2]     := SwapInteger(PCipherRec(@B).A);
      S[I, J * 2 +1] := SwapInteger(PCipherRec(@B).B);
    end;

  FillChar(B, SizeOf(B), 0);
  InitEnd(IVector);
end;

class procedure TCipher_IDEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 208;
end;

class function TCipher_IDEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,065h,0CAh,0D8h,043h,0E7h,099h,093h
         DB    0EDh,041h,0EAh,048h,0FDh,066h,050h,094h
         DB    0A2h,025h,06Dh,0D7h,0B1h,0D0h,09Ah,023h
         DB    03Dh,0D2h,0E8h,0ECh,0C9h,045h,07Fh,07Eh
end;
```

```
function IDEAMul(X, Y: LongWord): LongWord; assembler; register;
asm
     AND    EAX,0FFFFh
     JZ     @@1
     AND    EDX,0FFFFh
     JZ     @@1
     MUL    EDX
     MOV    ECX,EAX
     MOV    EDX,EAX
     SHR    EDX,16
     SUB    EAX,EDX
     CMP    AX,CX
     JNA    @@2
     INC    EAX
@@2: RET
@@1: MOV    ECX,1
     SUB    ECX,EAX
     SUB    ECX,EDX
     MOV    EAX,ECX
end;

procedure TCipher_IDEA.Cipher(Data, Key: PWordArray);
var
  I: LongWord;
  X,Y,A,B,C,D: LongWord;
begin
  I := SwapInteger(PIntArray(Data)[0]);
  A := LongRec(I).Hi;
  B := LongRec(I).Lo;
  I := SwapInteger(PIntArray(Data)[1]);
  C := LongRec(I).Hi;
  D := LongRec(I).Lo;
  for I := 0 to 7 do
  begin
    A := IDEAMul(A, Key[0]);
    Inc(B, Key[1]);
    Inc(C, Key[2]);
    D := IDEAMul(D, Key[3]);
    Y := C xor A;
    Y := IDEAMul(Y, Key[4]);
    X := B xor D + Y;
    X := IDEAMul(X, Key[5]);
    Inc(Y, X);
    A := A xor X;
    D := D xor Y;
    Y := B xor Y;
    B := C xor X;
    C := Y;
    Inc(PWord(Key), 6);
  end;
  LongRec(I).Hi := IDEAMul(A, Key[0]);
  LongRec(I).Lo := C + Key[1];
  PIntArray(Data)[0] := SwapInteger(I);
  LongRec(I).Hi := B + Key[2];
  LongRec(I).Lo := IDEAMul(D, Key[3]);
  PIntArray(Data)[1] := SwapInteger(I);
end;

procedure TCipher_IDEA.Encode(Data: Pointer);
begin
  Cipher(Data, User);
end;

procedure TCipher_IDEA.Decode(Data: Pointer);
begin
  Cipher(Data, @PIntArray(User)[26]);
end;

procedure TCipher_IDEA.Init(const Key; Size: Integer; IVector: Pointer);
```

```
  function IDEAInv(X: Word): Word;
  var
    A, B, C, D: Word;
  begin
    if X <= 1 then
    begin
      Result := X;
      Exit;
    end;
    A := 1;
    B := $10001 div X;
    C := $10001 mod X;
    while C <> 1 do
    begin
      D := X div C;
      X := X mod C;
      Inc(A, B * D);
      if X = 1 then
      begin
        Result := A;
        Exit;
      end;
      D := C div X;
      C := C mod X;
      Inc(B, A * D);
    end;
    Result := 1 - B;
  end;

var
  I: Integer;
  E: PWordArray;
  A,B,C: Word;
  K,D: PWordArray;
begin
  InitBegin(Size);
  E := User;
  Move(Key, E^, Size);
  for I := 0 to 7 do E[I] := Swap(E[I]);
  for I := 0 to 39 do
    E[I + 8] := E[I and not 7 + (I + 1) and 7] shl 9 or
                E[I and not 7 + (I + 2) and 7] shr 7;
  for I := 41 to 44 do
    E[I + 7] := E[I] shl 9 or E[I + 1] shr 7;
  K  := E;
  D  := @E[100];
  A  := IDEAInv(K[0]);
  B  := 0 - K[1];
  C  := 0 - K[2];
  D[3] := IDEAInv(K[3]);
  D[2] := C;
  D[1] := B;
  D[0] := A;
  Inc(PWord(K), 4);
  for I := 1 to 8 do
  begin
    Dec(PWord(D), 6);
    A    := K[0];
    D[5] := K[1];
    D[4] := A;
    A    := IDEAInv(K[2]);
    B    := 0 - K[3];
    C    := 0 - K[4];
    D[3] := IDEAInv(K[5]);
    D[2] := B;
    D[1] := C;
    D[0] := A;
    Inc(PWord(K), 6);
```

```
  end;
  A := D[2]; D[2] := D[1]; D[1] := A;
  InitEnd(IVector);
end;

type
  PSAFERRec = ^TSAFERRec;
  TSAFERRec = packed record
                case Integer of
                  0: (A,B,C,D,E,F,G,H: Byte);
                  1: (X,Y: Integer);
              end;

procedure TCipher_SAFER.SetRounds(Value: Integer);
begin
  if (Value < 4) or (Value > 13) then
    case FSaferMode of
      smK40, smSK40: Value := 5;
      smK64, smSK64: Value := 6;
      smK128, smSK128: Value := 10;
    else
      Value := 8;
    end;
  FRounds := Value;
end;

class procedure TCipher_SAFER.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 768;
end;

class function TCipher_SAFER.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    000h,03Dh,049h,020h,073h,063h,085h,0AAh
         DB    0D9h,0C2h,00Ah,0DEh,07Eh,09Eh,0E9h,0ABh
         DB    024h,0D0h,074h,034h,047h,07Eh,021h,01Dh
         DB    055h,0F9h,035h,028h,098h,084h,0A8h,075h
end;

procedure TCipher_SAFER.Encode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 512);
  with PSAFERRec(Data)^ do
  begin
    for I := 1 to FRounds do
    begin
      A := A xor Key[0];
      B := B  +  Key[1];
      C := C  +  Key[2];
      D := D xor Key[3];
      E := E xor Key[4];
      F := F  +  Key[5];
      G := G  +  Key[6];
      H := H xor Key[7];

      A := Exp[A]  +  Key[8];
      B := Log[B] xor Key[9];
      C := Log[C] xor Key[10];
```

```
      D := Exp[D]  +  Key[11];
      E := Exp[E]  +  Key[12];
      F := Log[F] xor Key[13];
      G := Log[G] xor Key[14];
      H := Exp[H]  +  Key[15];

      Inc(B, A); Inc(A, B);
      Inc(D, C); Inc(C, D);
      Inc(F, E); Inc(E, F);
      Inc(H, G); Inc(G, H);

      Inc(C, A); Inc(A, C);
      Inc(G, E); Inc(E, G);
      Inc(D, B); Inc(B, D);
      Inc(H, F); Inc(F, H);

      Inc(E, A); Inc(A, E);
      Inc(F, B); Inc(B, F);
      Inc(G, C); Inc(C, G);
      Inc(H, D); Inc(D, H);

      T := B; B := E; E := C; C := T;
      T := D; D := F; F := G; G := T;

      Inc(PByte(Key), 16);
    end;
    A := A xor Key[0];
    B := B  +  Key[1];
    C := C  +  Key[2];
    D := D xor Key[3];
    E := E xor Key[4];
    F := F  +  Key[5];
    G := G  +  Key[6];
    H := H xor Key[7];
  end;
end;

procedure TCipher_SAFER.Decode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 504 + 8 * (FRounds * 2 + 1));
  with PSAFERRec(Data)^ do
  begin
    H := H xor Key[7];
    G := G  -  Key[6];
    F := F  -  Key[5];
    E := E xor Key[4];
    D := D xor Key[3];
    C := C  -  Key[2];
    B := B  -  Key[1];
    A := A xor Key[0];

    for I := 1 to FRounds do
    begin
      Dec(PByte(Key), 16);
      T := E; E := B; B := C; C := T;
      T := F; F := D; D := G; G := T;

      Dec(A, E); Dec(E, A);
      Dec(B, F); Dec(F, B);
      Dec(C, G); Dec(G, C);
      Dec(D, H); Dec(H, D);

      Dec(A, C); Dec(C, A);
```

```
      Dec(E, G); Dec(G, E);
      Dec(B, D); Dec(D, B);
      Dec(F, H); Dec(H, F);

      Dec(A, B); Dec(B, A);
      Dec(C, D); Dec(D, C);
      Dec(E, F); Dec(F, E);
      Dec(G, H); Dec(H, G);

      H := H  -  Key[15];
      G := G xor Key[14];
      F := F xor Key[13];
      E := E  -  Key[12];
      D := D  -  Key[11];
      C := C xor Key[10];
      B := B xor Key[9];
      A := A  -  Key[8];

      H := Log[H] xor Key[7];
      G := Exp[G]  -  Key[6];
      F := Exp[F]  -  Key[5];
      E := Log[E] xor Key[4];
      D := Log[D] xor Key[3];
      C := Exp[C]  -  Key[2];
      B := Exp[B]  -  Key[1];
      A := Log[A] xor Key[0];
    end;
  end;
end;

procedure TCipher_SAFER.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smStrong);
end;

procedure TCipher_SAFER.InitNew(const Key; Size: Integer; IVector: Pointer;
SAFERMode: TSAFERMode);

  procedure InitTab;
  var
    I,E: Integer;
    Exp: PByte;
    Log: PByteArray;
  begin
    Exp := User;
    Log := Pointer(PChar(User) + 256);
    E   := 1;
    for I := 0 to 255 do
    begin
      Exp^ := E and $FF;
      Log[E and $FF] := I;
      E := (E * 45) mod 257;
      Inc(Exp);
    end;
  end;

  procedure InitKey;

    function ROR3(Value: Byte): Byte; assembler;
    asm
      ROR AL,3
    end;

    function ROL6(Value: Byte): Byte; assembler;
    asm
      ROL AL,6
    end;

  var
```

```
    D: PByte;
    Exp: PByteArray;
    Strong: Boolean;
    K: array[Boolean, 0..8] of Byte;
    I,J: Integer;
  begin
    Strong := FSAFERMode in [smStrong, smSK40, smSK64, smSK128];
    Exp := User;
    D := User;
    Inc(D, 512);
    FillChar(K, SizeOf(K), 0);
{Встановлюється ключ A}
    I := Size;
    if I > 8 then I := 8;
    Move(Key, K[False], I);

    if FSAFERMode in [smK40, smSK40] then
    begin
      K[False, 5] := K[False, 0] xor K[False, 2] xor 129;
      K[False, 6] := K[False, 0] xor K[False, 3] xor K[False, 4] xor 66;
      K[False, 7] := K[False, 1] xor K[False, 2] xor K[False, 4] xor 36;
      K[False, 8] := K[False, 1] xor K[False, 3] xor 24;
      Move(K[False], K[True], SizeOf(K[False]));
    end else
    begin
      if Size > 8 then
      begin
        I := Size - 8;
        if I > 8 then I := 8;
        Move(TByteArray(Key)[8], K[True], I);
      end else Move(K[False], K[True], 9);
      for I := 0 to 7 do
      begin
        K[False, 8] := K[False, 8] xor K[False, I];
        K[True, 8]  := K[True, 8]  xor K[True, I];
      end;
    end;
{Встановлюються дані ключа}
    Move(K[True], D^, 8);
    Inc(D, 8);

    for I := 0 to 8 do K[False, I] := ROR3(K[False, I]);

    for I := 1 to FRounds do
    begin
      for J := 0 to 8 do
      begin
        K[False, J] := ROL6(K[False, J]);
        K[True, J] := ROL6(K[True, J]);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[False, (J + I * 2 -1) mod 9] + Exp[Exp[18 * I + J
+1]]
          else D^ := K[False, J] + Exp[Exp[18 * I + J +1]];
        Inc(D);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[True, (J + I * 2) mod 9] + Exp[Exp[18 * I + J
+10]]
          else D^ := K[True, J] + Exp[Exp[18 * I + J +10]];
        Inc(D);
      end;
    end;
    FillChar(K, SizeOf(K), 0);
  end;

begin
```

```
  InitBegin(Size);
  FSAFERMode := SAFERMode;
  if SAFERMode = smDefault then
    if Size <= 5 then FSAFERMode := smK40 else
      if Size <= 8 then FSAFERMode := smK64 else FSAFERMode := smK128
  else
    if SAFERMode = smStrong then
      if Size <= 5 then FSAFERMode := smSK40 else
        if Size <= 8 then FSAFERMode := smSK64 else FSAFERMode := smSK128;
  SetRounds(FRounds);
  InitTab;
  InitKey;
  InitEnd(IVector);
end;

class procedure TCipher_SAFER_K40.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 5;
end;

class function TCipher_SAFER_K40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    005h,0B4h,019h,057h,026h,05Ch,013h,060h
         DB    0A0h,082h,094h,045h,0D6h,0A5h,046h,0D8h
         DB    073h,050h,096h,080h,04Fh,06Dh,0F7h,0E5h
         DB    0C8h,01Ah,0EFh,044h,04Ch,0B4h,059h,013h
end;

procedure TCipher_SAFER_K40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK40);
end;

class function TCipher_SAFER_SK40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0D9h,003h,003h,06Dh,018h,038h,0D1h,0C1h
         DB    089h,0E8h,038h,012h,07Fh,028h,0FCh,0C7h
         DB    0C5h,00Bh,0B7h,0C4h,0DBh,021h,0A4h,031h
         DB    020h,008h,08Ah,077h,0F7h,0DFh,026h,0FFh
end;

procedure TCipher_SAFER_SK40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK40);
end;

class procedure TCipher_SAFER_K64.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 8;
end;

class function TCipher_SAFER_K64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,0B2h,032h,0F0h,00Eh,0C2h,0DAh,0CBh
         DB    039h,008h,02Dh,05Ch,093h,0FFh,0CEh,0F3h
         DB    08Fh,01Fh,0B7h,02Ch,0C5h,0C7h,0A7h,0E9h
         DB    089h,0BEh,061h,08Bh,000h,0E6h,09Fh,00Eh
end;
```

```
procedure TCipher_SAFER_K64.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK64);
end;

class function TCipher_SAFER_SK64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0DDh,09Ch,01Ah,0D6h,029h,00Ch,0EEh,04Fh
         DB    0E5h,04Bh,0C0h,055h,0BFh,022h,00Eh,0BCh
         DB    019h,041h,078h,0CFh,094h,0DBh,02Fh,039h
         DB    06Bh,01Eh,0A7h,0CAh,04Bh,05Fh,077h,0E0h
end;

procedure TCipher_SAFER_SK64.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK64);
end;

class procedure TCipher_SAFER_K128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 16;
end;

class function TCipher_SAFER_K128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    00Ch,0A9h,070h,0B9h,0F3h,014h,087h,0D9h
         DB    09Eh,05Eh,078h,031h,074h,0DFh,0A8h,0BBh
         DB    03Dh,040h,0A5h,0D9h,08Ch,07Ch,004h,0B7h
         DB    09Ch,001h,0DAh,063h,0ABh,026h,035h,0BCh
end;

procedure TCipher_SAFER_K128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK128);
end;

class function TCipher_SAFER_SK128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0C8h,0A6h,070h,033h,029h,038h,038h,02Bh
         DB    069h,0ACh,061h,072h,08Fh,0DCh,09Fh,0A4h
         DB    09Eh,06Fh,0C4h,053h,0D8h,089h,0FFh,042h
         DB    072h,009h,07Dh,0CDh,0D0h,0EAh,07Eh,028h
end;

procedure TCipher_SAFER_SK128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK128);
end;

type
  PTEARec = ^TTEARec;
  TTEARec = packed record
              A,B,C,D: LongWord;
            end;

const
  TEA_Delta = $9E3779B9;

procedure TCipher_TEA.SetRounds(Value: Integer);
begin
  FRounds := Value;
```

```
  if FRounds < 16 then FRounds := 16 else
    if FRounds > 32 then FRounds := 32;
end;

class procedure TCipher_TEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 32;
end;

class function TCipher_TEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B7h,0B8h,0AAh,0BBh,026h,04Bh,006h,0F9h
         DB    070h,086h,0B0h,0E4h,056h,004h,029h,0CCh
         DB    0BFh,055h,0EAh,04Eh,0EFh,059h,026h,018h
         DB    019h,0B0h,003h,07Ch,029h,08Ch,0E2h,077h
end;

procedure TCipher_TEA.Encode(Data: Pointer);
{$IFDEF UseASM}
asm
         PUSH  EDI
         PUSH  ESI
         PUSH  EBX
         PUSH  EBP
         PUSH  EDX

         MOV   EBX,[EDX]       // X
         MOV   EDX,[EDX + 4]   // Y
         XOR   EDI,EDI         // Sum

         MOV   ESI,[EAX].TCipher_TEA.FUser  // Користувач
         MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди

@@1:     ADD   EDI,TEA_Delta

         MOV   EAX,EDX
         MOV   EBP,EDX
         SHL   EAX,4
         SHR   EBP,5
         ADD   EAX,[ESI]
         ADD   EBP,[ESI +  4]
         XOR   EAX,EDX
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EAX,EBX
         MOV   EBX,EAX
         SHL   EAX,4
         MOV   EBP,EBX
         SHR   EBP,5
         ADD   EAX,[ESI +  8]
         XOR   EAX,EBX
         ADD   EBP,[ESI + 12]
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EDX,EAX

         DEC   ECX
         JNZ   @@1

         POP   EAX
         MOV   [EAX],EBX
         MOV   [EAX + 4],EDX
```

```
          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Inc(Sum, TEA_Delta);
      Inc(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Inc(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
          PUSH  EDI
          PUSH  ESI
          PUSH  EBX
          PUSH  EBP
          PUSH  EDX

          MOV   EBX,[EDX]       // X
          MOV   EDX,[EDX + 4]   // Y

          MOV   ESI,[EAX].TCipher_TEA.FUser  // Користувач
          MOV   EDI,TEA_Delta
          MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди
          IMUL  EDI,ECX

@@1:      MOV   EAX,EBX
          MOV   EBP,EBX
          SHL   EAX,4
          SHR   EBP,5
          ADD   EAX,[ESI +  8]
          ADD   EBP,[ESI + 12]
          XOR   EAX,EBX
          ADD   EAX,EDI
          XOR   EAX,EBP
          SUB   EDX,EAX
          MOV   EAX,EDX
          SHL   EAX,4
          MOV   EBP,EDX
          SHR   EBP,5
          ADD   EAX,[ESI]
          XOR   EAX,EDX
          ADD   EBP,[ESI + 4]
          ADD   EAX,EDI

          XOR   EAX,EBP
          SUB   EDI,TEA_Delta
          SUB   EBX,EAX

          DEC   ECX
          JNZ   @@1

          POP   EAX
```

```
          MOV   [EAX],EBX
          MOV   [EAX + 4],EDX

          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
      Dec(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Dec(Sum, TEA_Delta);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  SetRounds(FRounds);
  InitEnd(IVector);
end;

class function TCipher_TEAN.TestVector: Pointer;
asm
          MOV   EAX,OFFSET @Vector
          RET
@Vector:  DB    0CDh,07Eh,0BBh,0A2h,092h,01Ah,04Bh,03Bh
          DB    0E2h,09Eh,062h,0CFh,0F7h,01Dh,0A5h,0DFh
          DB    063h,033h,094h,029h,0E2h,036h,07Ch,066h
          DB    03Fh,0F8h,01Ah,0F9h,002h,078h,0BFh,0A1h
end;

procedure TCipher_TEAN.Encode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  for I := 1 to FRounds do
  begin
    Inc(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    Inc(Sum, TEA_Delta);
    Inc(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
  end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

procedure TCipher_TEAN.Decode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
```

```
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
      Dec(Sum, TEA_Delta);
      Dec(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

const
  SCOP_SIZE = 32; {це максімум}

class procedure TCipher_SCOP.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := SCOP_SIZE * SizeOf(Integer);
  AKeySize := 48;
  AUserSize := (384 * 4 + 4 * SizeOf(Integer)) * 2;
end;

class function TCipher_SCOP.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    014h,0C0h,009h,0E8h,073h,0B6h,053h,092h
         DB    08Bh,013h,069h,0A9h,0F2h,099h,0FEh,05Eh
         DB    0EEh,03Bh,0FDh,0C1h,050h,059h,00Eh,094h
         DB    062h,017h,008h,01Eh,0A4h,01Ah,04Dh,08Fh
end;

procedure TCipher_SCOP.Encode(Data: Pointer);
var
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
  B: PInteger;
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Inc(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Decode(Data: Pointer);
var
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
```

```
  B: PInteger;
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Dec(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Init(const Key; Size: Integer; IVector: Pointer);
var
  Init_State: packed record
                Coef: array[0..7, 0..3] of Byte;
                X: array[0..3] of LongWord;
              end;

  procedure ExpandKey;
  var
    P: PByteArray;
    I,C: Integer;
  begin
    C := 1;
    P := @Init_State;
    Move(Key, P^, Size);
    for I := Size to 47 do P[I] := P[I - Size] + P[I - Size +1];
    for I := 0 to 31 do
      if P[I] = 0 then
      begin
        P[I] := C;
        Inc(C);
      end;
  end;

  procedure GP8(Data: PIntArray);
  var
    I,I2: Integer;
    NewX: array[0..3] of LongWord;
    X1,X2,X3,X4: LongWord;
    Y1,Y2: LongWord;
  begin
    I := 0;
    while I < 8 do
    begin
      I2 := I shr 1;
      X1 := Init_State.X[I2] shr 16;
      X2 := X1 * X1;
      X3 := X2 * X1;
      X4 := X3 * X1;
      Y1 := Init_State.Coef[I][0] * X4 +
            Init_State.Coef[I][1] * X3 +
            Init_State.Coef[I][2] * X2 +
            Init_State.Coef[I][3] * X1 + 1;
      X1 := Init_State.X[I2] and $FFFF;
      X2 := X1 * X1;
      X3 := X2 * X1;
```

```
      X4 := X3 * X1;
      Y2 := Init_State.Coef[I +1][0] * X4 +
            Init_State.Coef[I +2][1] * X3 +
            Init_State.Coef[I +3][2] * X2 +
            Init_State.Coef[I +4][3] * X1 + 1;
      Data[I2] := Y1 shl 16 or Y2 and $FFFF;
      NewX[I2] := Y1 and $FFFF0000 or Y2 shr 16;
      Inc(I, 2);
    end;
    Init_State.X[0] := NewX[0] shr 16 or NewX[3] shl 16;
    Init_State.X[1] := NewX[0] shl 16 or NewX[1] shr 16;
    Init_State.X[2] := NewX[1] shl 16 or NewX[2] shr 16;
    Init_State.X[3] := NewX[2] shl 16 or NewX[3] shr 16;
  end;

var
  I,J: Integer;
  T: array[0..3] of Integer;
  P: PIntArray;
begin
  InitBegin(Size);
  FillChar(Init_State, SizeOf(Init_State), 0);
  FillChar(T, SizeOf(T), 0);
  P := Pointer(PChar(User) + 12);
  ExpandKey;
  for I := 0 to 7 do GP8(@T);
  for I := 0 to 11 do
  begin
    for J := 0 to 7 do GP8(@P[I * 32 + J * 4]);
    GP8(@T);
  end;
  GP8(@T);
  I := T[3] and $7F;
  P[I] := P[I] or 1;
  P := User;
  P[0] := T[3] shr 24;
  P[1] := T[3] shr 16;
  P[2] := T[3] shr  8;
  FillChar(Init_State, SizeOf(Init_State), 0);
  InitEnd(IVector);
  P := Pointer(PChar(User) + FUserSize shr 1);
  Move(User^, P^, FUserSize shr 1);
end;

procedure TCipher_SCOP.Done;
begin
  inherited Done;
  Move(PByteArray(User)[FUserSize shr 1], User^, FUserSize shr 1);
end;

class procedure TCipher_Q128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 256;
end;

class function TCipher_Q128.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    099h,0AAh,0D0h,03Dh,0CAh,014h,04Eh,02Ah
        DB    0F8h,01Eh,001h,0A0h,0EAh,0ABh,09Fh,048h
        DB    023h,02Dh,059h,054h,054h,07Eh,02Bh,012h
        DB    086h,080h,0E8h,033h,0EBh,0E1h,05Eh,0AEh
end;

procedure TCipher_Q128.Encode(Data: Pointer);
```

```
{$IFDEF UseASM}
asm
        PUSH    ESI
        PUSH    EDI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     EDI,[EAX].TCipher_Q128.FUser

        MOV     EAX,[EDX]       // B0
        MOV     EBX,[EDX +  4]  // B1
        MOV     ECX,[EDX +  8]  // B2
        MOV     EDX,[EDX + 12]  // B3

        MOV     EBP,16

@@1:    MOV     ESI,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     ESI,10
        ADD     EAX,[EDI]
        XOR     EAX,EBX

        MOV     EBX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     EBX,10
        ADD     EAX,[EDI + 4]
        XOR     EAX,ECX

        MOV     ECX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     ECX,10
        ADD     EAX,[EDI + 8]
        XOR     EAX,EDX

        MOV     EDX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     EDX,10
        ADD     EAX,[EDI + 12]
        XOR     EAX,ESI

        ADD     EDI,16

        DEC     EBP
        JNZ     @@1

        POP     ESI

        MOV     [ESI],EAX       // B0
        MOV     [ESI +  4],EBX  // B1
        MOV     [ESI +  8],ECX  // B2
        MOV     [ESI + 12],EDX  // B3


        POP     EBP
        POP     EBX
        POP     EDI
        POP     ESI
end;
{$ELSE}
var
  D: PInteger;
  B0, B1, B2, B3, I: LongWord;
begin
  D  := User;
```

```
  B0 := PIntArray(Data)[0];
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B1 := B1 xor (Q128_Data[B0 and $03FF] + D^); Inc(D); B0 := B0 shl 10 or B0
shr 22;
    B2 := B2 xor (Q128_Data[B1 and $03FF] + D^); Inc(D); B1 := B1 shl 10 or B1
shr 22;
    B3 := B3 xor (Q128_Data[B2 and $03FF] + D^); Inc(D); B2 := B2 shl 10 or B2
shr 22;
    B0 := B0 xor (Q128_Data[B3 and $03FF] + D^); Inc(D); B3 := B3 shl 10 or B3
shr 22;
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}
procedure TCipher_Q128.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
       PUSH   ESI
       PUSH   EDI
       PUSH   EBX
       PUSH   EBP
       PUSH   EDX

       MOV    EDI,[EAX].TCipher_Q128.FUser
       LEA    EDI,[EDI + 64 * 4]

       MOV    ESI,[EDX]       // B0
       MOV    EBX,[EDX +  4]  // B1
       MOV    ECX,[EDX +  8]  // B2
       MOV    EDX,[EDX + 12]  // B3

       MOV    EBP,16

@@1:   SUB    EDI,16

       ROR    EDX,10
       MOV    EAX,EDX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI + 12]
       XOR    ESI,EAX

       ROR    ECX,10
       MOV    EAX,ECX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  8]
       XOR    EDX,EAX

       ROR    EBX,10
       MOV    EAX,EBX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  4]
       XOR    ECX,EAX

       ROR    ESI,10
       MOV    EAX,ESI
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI]
       XOR    EBX,EAX
```

```
        DEC    EBP
        JNZ    @@1

        POP    EAX

        MOV    [EAX],ESI       // B0
        MOV    [EAX +  4],EBX  // B1
        MOV    [EAX +  8],ECX  // B2
        MOV    [EAX + 12],EDX  // B3


        POP    EBP
        POP    EBX
        POP    EDI
        POP    ESI
end;
{$ELSE}
var
  D: PInteger;
  B0, B1, B2, B3, I: LongWord;
begin
  D  := @PIntArray(User)[63];
  B0 := PIntArray(Data)[0];
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B3 := B3 shr 10 or B3 shl 22; B0 := B0 xor (Q128_Data[B3 and $03FF] + D^);
Dec(D);
    B2 := B2 shr 10 or B2 shl 22; B3 := B3 xor (Q128_Data[B2 and $03FF] + D^);
Dec(D);
    B1 := B1 shr 10 or B1 shl 22; B2 := B2 xor (Q128_Data[B1 and $03FF] + D^);
Dec(D);
    B0 := B0 shr 10 or B0 shl 22; B1 := B1 xor (Q128_Data[B0 and $03FF] + D^);
Dec(D);
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}

procedure TCipher_Q128.Init(const Key; Size: Integer; IVector: Pointer);
var
  K: array[0..3] of LongWord;
  I: Integer;
  D: PInteger;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  D := User;
  for I := 19 downto 1 do
  begin
    K[1] := K[1] xor Q128_Data[K[0] and $03FF]; K[0] := K[0] shr 10 or K[0] shl
22;
    K[2] := K[2] xor Q128_Data[K[1] and $03FF]; K[1] := K[1] shr 10 or K[1] shl
22;
    K[3] := K[3] xor Q128_Data[K[2] and $03FF]; K[2] := K[2] shr 10 or K[2] shl
22;
    K[0] := K[0] xor Q128_Data[K[3] and $03FF]; K[3] := K[3] shr 10 or K[3] shl
22;
    if I <= 16 then
    begin
      D^ := K[0]; Inc(D);
      D^ := K[1]; Inc(D);
```

```
      D^ := K[2]; Inc(D);
      D^ := K[3]; Inc(D);
    end;
  end;
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

type
  P3Way_Key = ^T3Way_Key;
  T3Way_Key = packed record
                E_Key: array[0..2] of Integer;
                E_Data: array[0..11] of Integer;
                D_Key: array[0..2] of Integer;
                D_Data: array[0..11] of Integer;
              end;

class procedure TCipher_3Way.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 12;
  AKeySize := 12;
  AUserSize := SizeOf(T3Way_Key);
end;

class function TCipher_3Way.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    077h,0FCh,077h,094h,07Ch,08Fh,0DEh,021h
         DB    0E9h,081h,0DFh,02Ah,0B1h,0BCh,07Eh,0F8h
         DB    0A3h,0B6h,044h,04Bh,0B6h,0FCh,079h,0C4h
         DB    09Bh,068h,04Fh,009h,0C7h,0BFh,00Eh,005h
end;

procedure TCipher_3Way.Encode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := E_Key[0];
    K1 := E_Key[1];
    K2 := E_Key[2];
    E  := @E_Data;
  end;
  A0 := PIntArray(Data)[0];
  A1 := PIntArray(Data)[1];
  A2 := PIntArray(Data)[2];
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
```

```
    asm
      ROR  B0,10
      ROL  B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  PIntArray(Data)[0] := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl
16 xor
                              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl
24 xor
                              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl
8;
  PIntArray(Data)[1] := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl
16 xor
                              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl
24 xor
                              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl
8;
  PIntArray(Data)[2] := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl
16 xor
                              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl
24 xor
                              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl
8;
end;

procedure TCipher_3Way.Decode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := D_Key[0];
    K1 := D_Key[1];
    K2 := D_Key[2];
    E  := @D_Data;
  end;
  A0 := SwapBits(PIntArray(Data)[2]);
  A1 := SwapBits(PIntArray(Data)[1]);
  A2 := SwapBits(PIntArray(Data)[0]);
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
```

```
    asm
      ROR B0,10
      ROL B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
  B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
  B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;

  PIntArray(Data)[2] := SwapBits(B0);
  PIntArray(Data)[1] := SwapBits(B1);
  PIntArray(Data)[0] := SwapBits(B2);
end;

procedure TCipher_3Way.Init(const Key; Size: Integer; IVector: Pointer);

  procedure RANDGenerate(Start: Integer; var P: Array of Integer);
  var
    I: Integer;
  begin
    for I := 0 to 11 do
    begin
      P[I] := Start;
      Start := Start shl 1;
      if Start and $10000 <> 0 then Start := Start xor $11011;
    end;
  end;

var
  A0, A1, A2: Integer;
  B0, B1, B2: Integer;
begin
  InitBegin(Size);
  with P3Way_Key(User)^ do
  begin
    Move(Key, E_Key, Size);
    Move(Key, D_Key, Size);
    RANDGenerate($0B0B, E_Data);
    RANDGenerate($B1B1, D_Data);

    A0 := D_Key[0]; A1 := D_Key[1]; A2 := D_Key[2];
    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    D_Key[2] := SwapBits(B0); D_Key[1] := SwapBits(B1); D_Key[0] :=
SwapBits(B2);
  end;
```

```
  InitEnd(IVector);
end;

class procedure TCipher_Twofish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 32;
  AUserSize := 4256;
end;

class function TCipher_Twofish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0A5h,053h,057h,003h,0EFh,033h,048h,079h
         DB    09Fh,022h,0B4h,054h,097h,005h,084h,019h
         DB    087h,0BDh,083h,01Ch,04Dh,0AEh,012h,013h
         DB    060h,07Ch,07Ch,0D1h,098h,045h,002h,019h
end;

type
  PTwofishBox = ^TTwofishBox;
  TTwofishBox = array[0..3, 0..255] of Longword;

  TLongRec = record
               case Integer of
                 0: (L: Longword);
                 1: (A,B,C,D: Byte);
             end;

procedure TCipher_Twofish.Encode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S   := User;
  A.L := PIntArray(Data)[0] xor S[0];
  B.L := PIntArray(Data)[1] xor S[1];
  C.L := PIntArray(Data)[2] xor S[2];
  D.L := PIntArray(Data)[3] xor S[3];

  S   := @PIntArray(User)[8];
  Box := @PIntArray(User)[40];
  for I := 0 to 7 do
  begin
    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C] xor Box[0, B.D];
    asm ROL  D.L,1 end;
    C.L := C.L xor (X + Y       + S[0]);
    D.L := D.L xor (X + Y shl 1 + S[1]);
    asm ROR  C.L,1 end;

    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C] xor Box[0, D.D];
    asm ROL  B.L,1 end;
    A.L := A.L xor (X + Y       + S[2]);
    B.L := B.L xor (X + Y shl 1 + S[3]);
    asm ROR  A.L,1 end;
    Inc(PInteger(S), 4);
  end;
  S := User;
  PIntArray(Data)[0] := C.L xor S[4];
  PIntArray(Data)[1] := D.L xor S[5];
  PIntArray(Data)[2] := A.L xor S[6];
  PIntArray(Data)[3] := B.L xor S[7];
end;
```

```
procedure TCipher_Twofish.Decode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S := User;
  Box := @PIntArray(User)[40];
  C.L := PIntArray(Data)[0] xor S[4];
  D.L := PIntArray(Data)[1] xor S[5];
  A.L := PIntArray(Data)[2] xor S[6];
  B.L := PIntArray(Data)[3] xor S[7];
  S := @PIntArray(User)[36];
  for I := 0 to 7 do
  begin
    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[0, D.D] xor Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C];
    asm ROL  A.L,1 end;
    B.L := B.L xor (X + Y shl 1 + S[3]);
    A.L := A.L xor (X + Y       + S[2]);
    asm ROR  B.L,1 end;

    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[0, B.D] xor Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C];
    asm ROL  C.L,1 end;
    D.L := D.L xor (X + Y shl 1 + S[1]);
    C.L := C.L xor (X + Y       + S[0]);
    asm ROR  D.L,1 end;
    Dec(PByte(S),16);
  end;
  S := User;
  PIntArray(Data)[0] := A.L xor S[0];
  PIntArray(Data)[1] := B.L xor S[1];
  PIntArray(Data)[2] := C.L xor S[2];
  PIntArray(Data)[3] := D.L xor S[3];
end;

procedure TCipher_Twofish.Init(const Key; Size: Integer; IVector: Pointer);
var
  BoxKey: array[0..3] of TLongRec;
  SubKey: PIntArray;
  Box: PTwofishBox;

  procedure SetupKey;

    function Encode(K0, K1: Integer): Integer;
    var
      R, I, J, G2, G3: Integer;
      B: byte;
    begin
      R := 0;
      for I := 0 to 1 do
      begin
        if I <> 0 then R := R xor K0 else R := R xor K1;
        for J := 0 to 3 do
        begin
          B := R shr 24;
          if B and $80 <> 0 then G2 := (B shl 1 xor $014D) and $FF
            else G2 := B shl 1 and $FF;
          if B and 1 <> 0 then G3 := (B shr 1 and $7F) xor $014D shr 1 xor G2
            else G3 := (B shr 1 and $7F) xor G2;
          R := R shl 8 xor G3 shl 24 xor G2 shl 16 xor G3 shl 8 xor B;
        end;
      end;
      Result := R;
    end;
```

```
  function F32(X: Integer; K: array of Integer): Integer;
  var
    A, B, C, D: Integer;
  begin
    A := X and $FF;
    B := X shr  8 and $FF;
    C := X shr 16 and $FF;
    D := X shr 24;
    if Size = 32 then
    begin
      A := Twofish_8x8[1, A] xor K[3] and $FF;
      B := Twofish_8x8[0, B] xor K[3] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[3] shr 16 and $FF;
      D := Twofish_8x8[1, D] xor K[3] shr 24;
    end;
    if Size >= 24 then
    begin
      A := Twofish_8x8[1, A] xor K[2] and $FF;
      B := Twofish_8x8[1, B] xor K[2] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[2] shr 16 and $FF;
      D := Twofish_8x8[0, D] xor K[2] shr 24;
    end;
    A := Twofish_8x8[0, A] xor K[1] and $FF;
    B := Twofish_8x8[1, B] xor K[1] shr  8 and $FF;
    C := Twofish_8x8[0, C] xor K[1] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[1] shr 24;

    A := Twofish_8x8[0, A] xor K[0] and $FF;
    B := Twofish_8x8[0, B] xor K[0] shr  8 and $FF;
    C := Twofish_8x8[1, C] xor K[0] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[0] shr 24;

    Result := Twofish_Data[0, A] xor Twofish_Data[1, B] xor
              Twofish_Data[2, C] xor Twofish_Data[3, D];
  end;

var
  I,J,A,B: Integer;
  E,O: array[0..3] of Integer;
  K: array[0..7] of Integer;
begin
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  if Size <= 16 then Size := 16 else
    if Size <= 24 then Size := 24
      else Size := 32;
  J := Size shr 3 - 1;
  for I := 0 to J do
  begin
    E[I] := K[I shl 1];
    O[I] := K[I shl 1 + 1];
    BoxKey[J].L := Encode(E[I], O[I]);
    Dec(J);
  end;
  J := 0;
  for I := 0 to 19 do
  begin
    A := F32(J, E);
    B := ROL(F32(J + $01010101, O), 8);
    SubKey[I shl 1] := A + B;
    B := A + B shr 1;
    SubKey[I shl 1 + 1] := ROL(B, 9);
    Inc(J, $02020202);
  end;
end;

procedure DoXOR(D, S: PIntArray; Value: LongWord);
var
  I: LongWord;
```

```
  begin
    Value := (Value and $FF) * $01010101;
    for I := 0 to 63 do D[I] := S[I] xor Value;
  end;

  procedure SetupBox128;
  var
    L: array[0..255] of Byte;
    A,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L);
    A := BoxKey[0].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 8);
    A := BoxKey[0].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L shr 16);
    A := BoxKey[0].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 24);
    A := BoxKey[0].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, L[I]] xor A];
  end;

  procedure SetupBox192;
  var
    L: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L);
    A := BoxKey[0].A;
    B := BoxKey[1].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L ,@Twofish_8x8[0], BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

  procedure SetupBox256;
  var
    L: array[0..255] of Byte;
    K: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L);
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L);
    A := BoxKey[0].A;
```

```
    B := BoxKey[1].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0], BoxKey[3].L shr 8);
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0],BoxKey[3].L shr 16);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L shr 24);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

begin
  InitBegin(Size);
  SubKey := User;
  Box    := @SubKey[40];
  SetupKey;
  if Size = 16 then SetupBox128 else
    if Size = 24 then SetupBox192
      else SetupBox256;
  InitEnd(IVector);
end;

class procedure TCipher_Shark.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 112;
end;

class function TCipher_Shark.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    0D9h,065h,021h,0AAh,0C0h,0C3h,084h,060h
        DB    09Dh,0CEh,01Fh,08Bh,0FBh,0ABh,018h,03Fh
        DB    0A1h,021h,0ACh,0F8h,053h,049h,0C0h,06Fh
        DB    027h,03Ah,089h,015h,0D3h,07Ah,0E9h,00Bh
end;

{$IFDEF VER_D4H} // >= D4
  {.$DEFINE Shark64}
{$ENDIF}

type
  PInt64        = ^TInt64;
{$IFDEF Shark64}
  TInt64        = Int64;
{$ELSE}
  TInt64        = packed record
```

```
                    L,R: Integer;
                  end;
{$ENDIF}

  PInt64Array = ^TInt64Array;
  TInt64Array = array[0..1023] of TInt64;

{$IFDEF Shark64}
  TShark_Data = array[0..7, 0..255] of Int64;
{$ENDIF}

procedure TCipher_Shark.Encode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  L,R: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
         TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CE)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SE[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    T := Shark_CE[0, R shr 23 and $1FE] xor
         Shark_CE[1, R shr 15 and $1FE] xor
         Shark_CE[2, R shr  7 and $1FE] xor
         Shark_CE[3, R shl  1 and $1FE] xor
         Shark_CE[4, L shr 23 and $1FE] xor
         Shark_CE[5, L shr 15 and $1FE] xor
         Shark_CE[6, L shr  7 and $1FE] xor
         Shark_CE[7, L shl  1 and $1FE];
    R := Shark_CE[0, R shr 23 and $1FE or 1] xor
         Shark_CE[1, R shr 15 and $1FE or 1] xor
         Shark_CE[2, R shr  7 and $1FE or 1] xor
         Shark_CE[3, R shl  1 and $1FE or 1] xor
         Shark_CE[4, L shr 23 and $1FE or 1] xor
         Shark_CE[5, L shr 15 and $1FE or 1] xor
```

```
          Shark_CE[6, L shr  7 and $1FE or 1] xor
          Shark_CE[7, L shl  1 and $1FE or 1];
    L := T;
  end;
  L := L xor K[0];
  R := R xor K[1];
  Inc(PInteger(K), 2);
  L := LongWord(Shark_SE[L shr 24        ]) shl 24 xor
       LongWord(Shark_SE[L shr 16 and $FF]) shl 16 xor
       LongWord(Shark_SE[L shr  8 and $FF]) shl  8 xor
       LongWord(Shark_SE[L        and $FF]);
  R := LongWord(Shark_SE[R shr 24        ]) shl 24 xor
       LongWord(Shark_SE[R shr 16 and $FF]) shl 16 xor
       LongWord(Shark_SE[R shr  8 and $FF]) shl  8 xor
       LongWord(Shark_SE[R        and $FF]);
  PInt64(Data).L := L xor K[0];
  PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Decode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  R,L: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  Inc(K, 7);
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CD)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CD)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CD)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CD)[3, D shr 32 and $FF] xor

         TShark_Data(Shark_CD)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CD)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CD)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CD)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SD[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SD[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SD[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SD[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SD[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SD[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SD[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SD[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  Inc(PInteger(K), 14);
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    T := Shark_CD[0, R shr 23 and $1FE] xor
         Shark_CD[1, R shr 15 and $1FE] xor
```

```
          Shark_CD[2, R shr  7 and $1FE] xor
          Shark_CD[3, R shl  1 and $1FE] xor
          Shark_CD[4, L shr 23 and $1FE] xor
          Shark_CD[5, L shr 15 and $1FE] xor
          Shark_CD[6, L shr  7 and $1FE] xor
          Shark_CD[7, L shl  1 and $1FE];
    R := Shark_CD[0, R shr 23 and $1FE or 1] xor
          Shark_CD[1, R shr 15 and $1FE or 1] xor
          Shark_CD[2, R shr  7 and $1FE or 1] xor
          Shark_CD[3, R shl  1 and $1FE or 1] xor
          Shark_CD[4, L shr 23 and $1FE or 1] xor
          Shark_CD[5, L shr 15 and $1FE or 1] xor
          Shark_CD[6, L shr  7 and $1FE or 1] xor
          Shark_CD[7, L shl  1 and $1FE or 1];
    L := T;
  end;
  L := L xor K[0];
  R := R xor K[1];
  Inc(PInteger(K), 2);
  L := Integer(Shark_SD[L shr 24        ]) shl 24 xor
       Integer(Shark_SD[L shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[L shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[L        and $FF]);
  R := Integer(Shark_SD[R shr 24        ]) shl 24 xor
       Integer(Shark_SD[R shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[R shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[R        and $FF]);
  PInt64(Data).L := L xor K[0];
  PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Init(const Key; Size: Integer; IVector: Pointer);
var
  Log, ALog: array[0..255] of Byte;

  procedure InitLog;
  var
    I, J: Word;
  begin
    ALog[0] := 1;
    for I := 1 to 255 do
    begin
      J := ALog[I-1] shl 1;
      if J and $100 <> 0 then J := J xor $01F5;
      ALog[I] := J;
    end;
    for I := 1 to 254 do Log[ALog[I]] := I;
  end;

  function Transform(A: TInt64): TInt64;
  type
    TInt64Rec = packed record
                  Lo, Hi: Integer;
                end;

    function Mul(A, B: Integer): Byte;
    begin
      Result := ALog[(Log[A] + Log[B]) mod 255];
    end;

  var
    I,J: Byte;
    K,T: array[0..7] of Byte;
  begin
{$IFDEF Shark64}
    Move(TInt64Rec(A).Hi, K[0], 4);
    Move(TInt64Rec(A).Lo, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
```

```
{$ELSE}
    Move(A.R, K[0], 4);
    Move(A.L, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
{$ENDIF}
    for I := 0 to 7 do
    begin
      T[I] := Mul(Shark_I[I, 0], K[0]);
      for J := 1 to 7 do T[I] := T[I] xor Mul(Shark_I[I, J], K[J]);
    end;
{$IFDEF Shark64}
    Result := T[0];
    for I := 1 to 7 do Result := Result shl 8 xor T[I];
{$ELSE}
    Result.L := T[0];
    Result.R := 0;
    for I := 1 to 7 do
    begin
      Result.R := Result.R shl 8 or Result.L shr 24;
      Result.L := Result.L shl 8 xor T[I];
    end;
{$ENDIF}
  end;

  function Shark(D: TInt64; K: PInt64): TInt64;
  var
    R,T: Integer;
  begin
{$IFDEF Shark64}
    for R := 0 to 4 do
    begin
      D := D xor K^; Inc(K);
      D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
           TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
           TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
           TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
           TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
           TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
           TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
           TShark_Data(Shark_CE)[7, D        and $FF];
    end;
    D := D xor K^; Inc(K);
    D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
         (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
         (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
         (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
         (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
         (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
         (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
         (Int64(Shark_SE[D        and $FF]));
    Result := D xor K^;
{$ELSE}
    for R := 0 to 4 do
    begin
      D.L := D.L xor K.L;
      D.R := D.R xor K.R;
      Inc(K);
      T   := Shark_CE[0, D.R shr 23 and $1FE] xor
             Shark_CE[1, D.R shr 15 and $1FE] xor
             Shark_CE[2, D.R shr  7 and $1FE] xor
             Shark_CE[3, D.R shl  1 and $1FE] xor
             Shark_CE[4, D.L shr 23 and $1FE] xor
             Shark_CE[5, D.L shr 15 and $1FE] xor
             Shark_CE[6, D.L shr  7 and $1FE] xor
             Shark_CE[7, D.L shl  1 and $1FE];

      D.R := Shark_CE[0, D.R shr 23 and $1FE or 1] xor
             Shark_CE[1, D.R shr 15 and $1FE or 1] xor
             Shark_CE[2, D.R shr  7 and $1FE or 1] xor
```

```
              Shark_CE[3, D.R shl  1 and $1FE or 1] xor
              Shark_CE[4, D.L shr 23 and $1FE or 1] xor
              Shark_CE[5, D.L shr 15 and $1FE or 1] xor
              Shark_CE[6, D.L shr  7 and $1FE or 1] xor
              Shark_CE[7, D.L shl  1 and $1FE or 1];
      D.L := T;
    end;
    D.L := D.L xor K.L;
    D.R := D.R xor K.R;
    Inc(K);
    D.L := Integer(Shark_SE[D.L shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.L shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.L shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.L        and $FF]);
    D.R := Integer(Shark_SE[D.R shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.R shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.R shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.R        and $FF]);
    Result.L := D.L xor K.L;
    Result.R := D.R xor K.R;
{$ENDIF}
  end;

var
  T: array[0..6] of TInt64;
  A: array[0..6] of TInt64;
  K: array[0..15] of Byte;
  I,J,R: Byte;
  E,D: PInt64Array;
  L: TInt64;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  InitLog;
  E := User;
  D := @E[7];
  Move(Shark_CE[0], T, SizeOf(T));
  T[6] := Transform(T[6]);
  I := 0;
{$IFDEF Shark64}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R] := K[I and $F];
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R] := A[R] shl 8 or K[I and $F];
    end;
  end;
  E[0] := A[0] xor Shark(0, @T);
  for R := 1 to 6 do E[R] := A[R] xor Shark(E[R - 1], @T);
{$ELSE}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R].L := K[I and $F];
    A[R].R := 0;
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R].R := A[R].R shl 8 or A[R].L shr 24;
      A[R].L := A[R].L shl 8 or K[I and $F];
    end;
  end;
  L.L := 0;
  L.R := 0;
  L := Shark(L, @T);
```

```
  E[0].L := A[0].L xor L.L;
  E[0].R := A[0].R xor L.R;
  for R := 1 to 6 do
  begin
    L := Shark(E[R - 1], @T);
    E[R].L := A[R].L xor L.L;
    E[R].R := A[R].R xor L.R;
  end;
{$ENDIF}

  E[6] := Transform(E[6]);
  D[0] := E[6];
  D[6] := E[0];
  for R := 1 to 5 do D[R] := Transform(E[6-R]);

  FillChar(Log, SizeOf(Log), 0);
  FillChar(ALog, SizeOf(ALog), 0);
  FillChar(T, SizeOf(T), 0);
  FillChar(A, SizeOf(A), 0);
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

class procedure TCipher_Square.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 9 * 4 * 2 * SizeOf(LongWord);
end;

class function TCipher_Square.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    043h,09Ch,0A6h,0C4h,067h,0E8h,02Eh,047h
         DB    022h,095h,066h,085h,006h,039h,06Ah,0C9h
         DB    018h,021h,020h,0F7h,044h,036h,0F1h,061h
         DB    07Dh,014h,090h,0B1h,0A9h,068h,056h,0C7h
end;

procedure TCipher_Square.Encode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := User;
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);
  for I := 0 to 6 do
  begin
    AA := Square_TE[0, A        and $FF] xor
          Square_TE[1, B        and $FF] xor
          Square_TE[2, C        and $FF] xor
          Square_TE[3, D        and $FF] xor Key[0];
    BB := Square_TE[0, A shr  8 and $FF] xor
          Square_TE[1, B shr  8 and $FF] xor
          Square_TE[2, C shr  8 and $FF] xor
          Square_TE[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TE[0, A shr 16 and $FF] xor
          Square_TE[1, B shr 16 and $FF] xor
          Square_TE[2, C shr 16 and $FF] xor
          Square_TE[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TE[0, A shr 24        ] xor
```

```
          Square_TE[1, B shr 24        ] xor
          Square_TE[2, C shr 24        ] xor
          Square_TE[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);

    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SE[A        and $FF])        xor
                        LongWord(Square_SE[B        and $FF]) shl  8 xor
                        LongWord(Square_SE[C        and $FF]) shl 16 xor
                        LongWord(Square_SE[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SE[A shr  8 and $FF])        xor
                        LongWord(Square_SE[B shr  8 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SE[A shr 16 and $FF])        xor
                        LongWord(Square_SE[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SE[A shr 24       ])        xor
                        LongWord(Square_SE[B shr 24       ]) shl  8 xor
                        LongWord(Square_SE[C shr 24       ]) shl 16 xor
                        LongWord(Square_SE[D shr 24       ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Decode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := @PIntArray(User)[9 * 4];
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);

  for I := 0 to 6 do
  begin
    AA := Square_TD[0, A        and $FF] xor
          Square_TD[1, B        and $FF] xor
          Square_TD[2, C        and $FF] xor
          Square_TD[3, D        and $FF] xor Key[0];
    BB := Square_TD[0, A shr  8 and $FF] xor
          Square_TD[1, B shr  8 and $FF] xor
          Square_TD[2, C shr  8 and $FF] xor
          Square_TD[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TD[0, A shr 16 and $FF] xor
          Square_TD[1, B shr 16 and $FF] xor
          Square_TD[2, C shr 16 and $FF] xor
          Square_TD[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TD[0, A shr 24        ] xor
          Square_TD[1, B shr 24        ] xor
          Square_TD[2, C shr 24        ] xor
          Square_TD[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);
    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SD[A        and $FF])        xor
                        LongWord(Square_SD[B        and $FF]) shl  8 xor
                        LongWord(Square_SD[C        and $FF]) shl 16 xor
                        LongWord(Square_SD[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SD[A shr  8 and $FF])        xor
```

```
                        LongWord(Square_SD[B shr  8 and $FF]) shl  8 xor
                        LongWord(Square_SD[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SD[A shr 16 and $FF])        xor
                        LongWord(Square_SD[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SD[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SD[A shr 24        ])        xor
                        LongWord(Square_SD[B shr 24        ]) shl  8 xor
                        LongWord(Square_SD[C shr 24        ]) shl 16 xor
                        LongWord(Square_SD[D shr 24        ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Init(const Key; Size: Integer; IVector: Pointer);
type
  PSquare_Key = ^TSquare_Key;
  TSquare_Key = array[0..8, 0..3] of LongWord;
var
  E,D: PSquare_Key;
  T,I: Integer;
begin
  InitBegin(Size);
  E := User;
  D := User; Inc(D);
  Move(Key, E^, Size);
  for T := 1 to 8 do
  begin
    E[T, 0] := E[T -1, 0] xor ROR(E[T -1, 3], 8) xor 1 shl (T - 1); D[8 -T, 0]
:= E[T, 0];
    E[T, 1] := E[T -1, 1] xor E[T, 0];                               D[8 -T, 1]
:= E[T, 1];
    E[T, 2] := E[T -1, 2] xor E[T, 1];                               D[8 -T, 2]
:= E[T, 2];
    E[T, 3] := E[T -1, 3] xor E[T, 2];                               D[8 -T, 3]
:= E[T, 3];
    for I := 0 to 3 do
      E[T -1, I] :=     Square_PHI[E[T -1, I]        and $FF]      xor
                    ROL(Square_PHI[E[T -1, I] shr  8 and $FF],  8) xor
                    ROL(Square_PHI[E[T -1, I] shr 16 and $FF], 16) xor
                    ROL(Square_PHI[E[T -1, I] shr 24        ], 24);
  end;
  D[8] := E[0];
  InitEnd(IVector);
end;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386


procedure FindVirtualMethodAndChange(AClass: TClass; MethodAddr, NewAddress:
Pointer);
type
  PPointer = ^Pointer;
const
  PageSize = SizeOf(Pointer);
var
  Table: PPointer;
  SaveFlag: DWORD;
begin
  Table := PPointer(AClass);
  while Table^ <> MethodAddr do Inc(Table);
  if VirtualProtect(Table, PageSize, PAGE_EXECUTE_READWRITE, @SaveFlag) then
  try
    Table^ := NewAddress;
  finally
    VirtualProtect(Table, PageSize, SaveFlag, @SaveFlag);
  end;
end;
  {$ENDIF}
```

```
{$ENDIF}

{$IFDEF VER_D3H}
procedure ModuleUnload(Module: Integer);
var
  I: Integer;
begin
  if IsObject(FCipherList, TStringList) then
    for I := FCipherList.Count-1 downto 0 do
      if FindClassHInstance(TClass(FCipherList.Objects[I])) = Module then
        FCipherList.Delete(I);
end;
{$ENDIF}

initialization
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
  if CPUType <= 3 then  // CPU <= 386
  begin
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Encode,
                                                 @TCipher_Blowfish.Encode386);
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Decode,
                                                 @TCipher_Blowfish.Decode386);
  end;
  {$ENDIF}
{$ENDIF}
{$IFDEF VER_D3H}
  AddModuleUnloadProc(ModuleUnload);
{$ENDIF}
{$IFNDEF ManualRegisterClasses}
  RegisterCipher(TCipher_3Way, '', '');
  RegisterCipher(TCipher_Blowfish, '', '');
  RegisterCipher(TCipher_Gost, '', '');
  RegisterCipher(TCipher_IDEA, '', 'не комерційний');
  RegisterCipher(TCipher_Q128, '', '');
  RegisterCipher(TCipher_SAFER_K40, 'SAFER-K40', '');
  RegisterCipher(TCipher_SAFER_SK40, 'SAFER-SK40', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K64, 'SAFER-K64', '');
  RegisterCipher(TCipher_SAFER_SK64, 'SAFER-SK64', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K128, 'SAFER-K128', '');
  RegisterCipher(TCipher_SAFER_SK128, 'SAFER-SK128', 'Keyscheduling');
  RegisterCipher(TCipher_SCOP, '', '');
  RegisterCipher(TCipher_Shark, '', '');
  RegisterCipher(TCipher_Square, '', '');
  RegisterCipher(TCipher_TEA, 'TEA', '');
  RegisterCipher(TCipher_TEAN, 'TEA розширений', '');
  RegisterCipher(TCipher_Twofish, '', '');
{$ENDIF}
finalization
{$IFDEF VER_D3H}
  RemoveModuleUnloadProc(ModuleUnload);
{$ENDIF}
  FCipherList.Free;
  FCipherList := nil;
end.
```