МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Система візуального програмування за криптографічним алгоритмом»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Лаврик Т.В.**

Студентка групи КБ – 61 — **Мучарова Є.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ

до випускної роботи

Студентки четвертого курсу, групи КБ-61 спеціальності "Кібербезпека" денної форми навчання Мучарової Євгенії Олександрівни.

Тема: "**Система візуального програмування за криптографічним алгоритмом**"

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: **:** 1) аналіз предметної області; 2) огляд існуючих програмних рішень; 3) постановка задачі; 4) опис основних положень і математичних моделей криптографічних алгоритмів, що використовуються; 5) реалізація програмного рішення.

Дата видачі завдання "_____"_____2020 р.

Керівник випускної роботи _____ Лаврик Т.В.

Завдання приняв до виконання _____ Мучарова Є.О.

РЕФЕРАТ

Записка: 96 стор., 39 рис., 11 табл., 2 додатки, 17 джерел.

Об’єкт дослідження – системи візуального програмування за криптографічними алгоритмами.

Мета роботи — розробка системи візуального програмування за криптографічними алгоритмами, яка дозволяє на основі побудованої блок-схеми відобразити відповідний програмний код і здійснити шифрування даних.

Методи дослідження — метод аналітичного огляду, метод порівняння та аналогій, метод моделювання, методи розробки, що базуються на мові програмування C#.

Результати — проведено інформаційний огляд, обрані методи реалізації, виконано розробку системи, що має назву “CodAlgotithm”. Додаток надає змогу зашифрувати дані трьома різними криптографічними алгоритмами. Але це можливо тільки після того, як користувач правильно побудує блок-схеми та перегляне рядки коду, що відображають дії елементів цих схем. “CodAlgotithm” готовий для використання та інтегрування в інші системи.

ІНФОРМАЦІЙНА БЕЗПЕКА, КРИПТОГРАФІЧНИЙ АЛГОРИТМ, ШИФРУВАННЯ ДАНИХ, БЛОК-СХЕМА, ВІЗУАЛЬНЕ ПРОГРАМУВАННЯ

ЗМІСТ

ВСТУП

Важко уявити сучасний світ без інформаційних технологій. Вдома та на роботі, мобільні телефони, електронні листи та комп’ютери стали частиною повсякденного життя. ІТ – це загальне найменування, яке надається всім удосконаленням, які відбуваються в нашому світі завдяки взаємопов'язаному просуванню технологій, навчання та інформації. Три основні компоненти інформаційних технологій – це апаратне забезпечення, програмне забезпечення та люди. Люди використовують апаратне та програмне забезпечення для зберігання, отримання, захисту та перетворення інформації. Апаратне забезпечення – це фізична складова комп'ютерної системи, яка забезпечує носій для введення даних та отримання необхідного результату. Інформаційні технології в основному зосереджені на використанні попередньо існуючих систем та їх включенні в нову систему з невеликими змінами залежно від потреб користувача.

Однак, з розвитком ІТ важливим аспектом у кожній організації стала інформаційна безпека. Компанії та підприємства володіють великими обсягами інформації, яка є цінним їх ресурсом. Злом важливих даних, відключення мережі, комп'ютерні віруси та інші кіберзагрози впливають на наше життя, яке варіюється від незначних незручностей до серйозних інцидентів.

Збиток від "комп'ютерних" злочинів щорічно оцінюється десятками мільярдів в грошових одиницях найбільш розвинених в застосуванні інформаційних технологій країн. Зростання загроз для інформації визначає необхідність розвитку захисту інформації. Згідно «Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» захисту в системі підлягає:

- відкрита інформація, яка належить до державних інформаційних ресурсів, а також відкрита інформація про діяльність суб'єктів владних повноважень, військових формувань;
- конфіденційна інформація;

- службова інформація;
- інформація, яка становить державну або іншу передбачену законом таємницю;
- інформація, вимога захисту якої встановлена законом [1].

Актуальним і важливим є вивчення криптографічних алгоритмів майбутніми програмістами і фахівцями з інформаційної безпеки. Вирішення задач шифрування чи розшифрування повідомлень за допомогою криптографічних алгоритмів можливо здійснювати вручну за допомогою середовищ MSWord, MSExcel та інших, або ж за допомогою створеного програмного коду.

1 АНАЛІЗ СИСТЕМИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ ЗА КРИПТОГРАФІЧНИМИ АЛГОРИТМАМИ

1.1 Загальна характеристика систем візуалізації програм

На сьогоднішній день спостерігається зростання ролі візуалізації в використовуваних технологіях програмування. Проаналізувавши концепцію візуального програмування більш уважно, можна зрозуміти, що вона базується на парадигмі програмування потоку даних (dataflow programming). Цей підхід був придуманий ще в 70-х роках минулого століття. Він полягає в тому, що будь-яку програму можна представити у вигляді орграфа, який відображає потік даних між компонентами програми (по суті, це та ж блок-схема).

Прагнення до візуалізації алгоритмів у людини зародилося практично одночасно з появою самого поняття «алгоритм». Воно походить від природного бажання точніше визначити і позначити свої цілі та дії. Крім того, візуалізація допомагає краще осягати задуману ідею і розвивати її. Наприклад, ієрогліфічний запис в піраміді «Муміфікувати слід ось так ...» цілком може вважатися візуалізацією алгоритму.

Якщо ж говорити про програмування, то перше, що спадає на думку – це блок-схема, найочевидніший приклад візуального програмування. Блок-схема – це схематичне / графічне подання послідовності кроків для вирішення проблеми.

Переваги блок-схеми:

- блок-схема є відмінним способом передачі логіки програми;
- простий та ефективний аналіз проблеми за допомогою блок-схеми;
- під час циклу розробки програми блок-схема відіграє роль креслення, що полегшує процес розробки програми;
- блок-схема полегшує обслуговування програми або системи;
- легко перетворюється в будь-який код мови програмування [2].

Новий графічний підхід до вирішення проблеми автоматизації розробки, котрий базується на ідеї залучення візуальних форм представлення програм,

більшою мірою відповідає образному способу мислення людини. Застосування графічних методів обіцяє кардинально підвищити продуктивність праці програміста. Візуальне програмування, безперечно, має гідність наочного подання інформації та набагато краще відповідає природі людського сприйняття, ніж методи традиційного, текстового програмування. Крім того, графічна форма запису в порівнянні з текстовим поданням програм забезпечує більш високий рівень структуризації, дотримання технологічної культури програмування, пропонує більш надійний стиль програмування.

Так як в будь-якій програмі головне це алгоритм, то, записавши його в наглядній графічній формі, далі вже не принципово операторами якої мови програмування будуть навантажені дуги цього базису. Тому вивчення алгоритмізації та програмування в системі освіти починати доцільно з візуальних мов. Критерій високого рівня розуміння вимагає, щоб форма запису алгоритму була максимально зручною і зрозумілою. Чим зрозуміліше запис алгоритму, тим легше знайти дефекти в ньому. Адже чим більше помилок буде виявлено при візуальному аналізі алгоритму, тим більша ймовірність, що в програмі їх залишиться менше. Крім того, зрозуміла візуальна форма запису алгоритму простіше для пояснення суті алгоритму іншій зацікавленій особі й більш зручна для вивчення.

До того ж всі без винятку відомі мови програмування занадто складні для звичайної людини (непрофесійного програміста), щоб записувати на них алгоритми. Тому базисом може бути візуальна графічна мова. Будь-який фахівець зможе виконувати алгоритмізацію своїх професійних знань самостійно, без допомоги програмістів.

Для порівняння розглянемо дві візуальні мови. Перша – це мова Дракон-схем, описана В. Пароннджановим, друга – мова Р-схем, запропонована І. В. Вельбицький [3].

Дракон-схеми є не що інше, як правильно складені блок-схеми. Графічна мова Р-схем сконструйована спеціально, щоб здійснити плавний перехід від

алгоритму, представленого у вигляді двовимірної картинки, до двовимірної програми. При цьому базис алгоритму і програми не змінюється.

Розглянемо приклад, в якому реалізується послідовність дій, що чергується перевірками успішності їх виконання. Результатом такої перевірки можуть бути два результати — успіх і відмова. Ситуація типова для багатьох керуючих алгоритмів. Спочатку наведемо фрагмент запису алгоритму на візуальній мові Дракон (рис. 1.1), а потім приведемо запис цього ж алгоритму у вигляді Р-схеми (рис. 1.2), де для "структуризації" майбутньої програми явно дублюється відповідна дія «відмова від запуску», а на рис. 1.3 показаний перехід в кінець алгоритму за допомогою return.

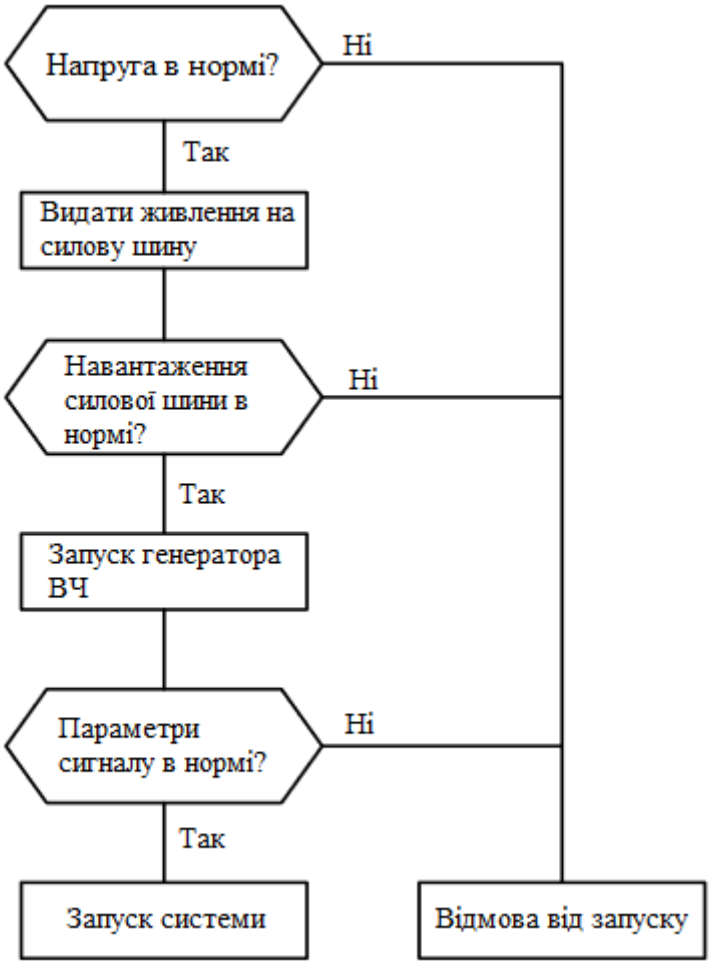

Рисунок 1.1 — Дракон-схема

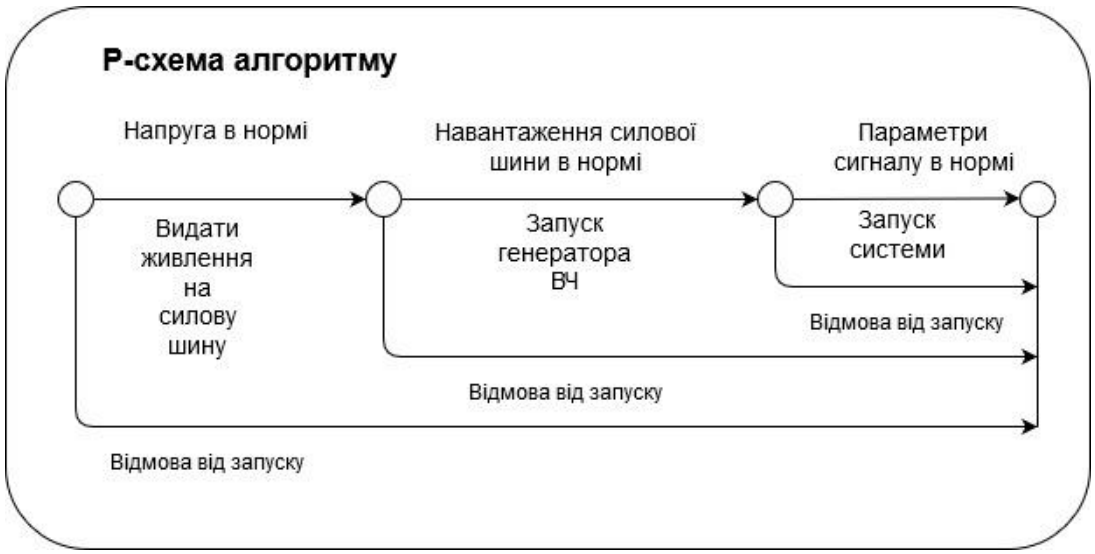

Рисунок 1.2 — Р-схема алгоритму

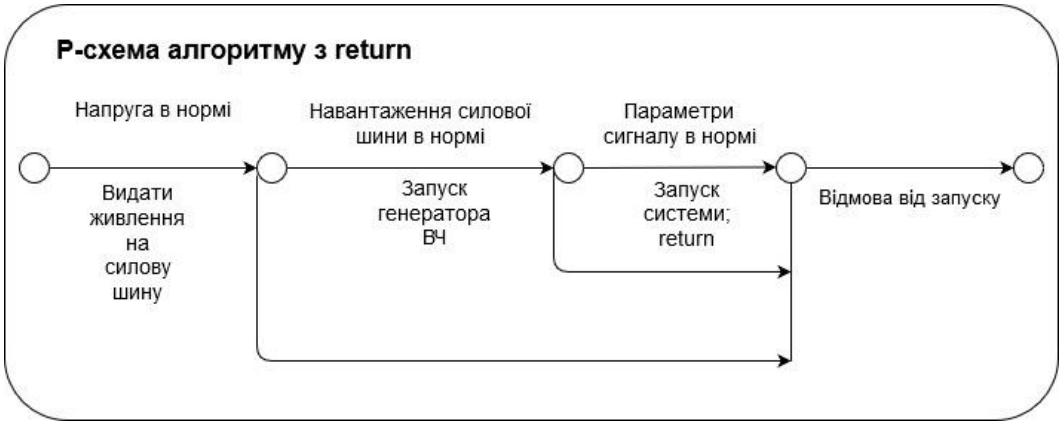

Рисунок 1.3 — Р-схема алгоритму з використанням return

В даний час відома велика кількість вдалих інструментальних засобів візуалізації програмування. Перш за все, до них відносяться візуальні засоби розробки програмного забезпечення (ПЗ): Visual PROLOG, Visual Studio, Visual BASIC тощо (рис. 1.4). У перерахованих вище засобах процедурі візуалізації піддається головним чином призначений для користувача інтерфейс, заснований на віконній технології організації взаємодії людини із середовищем програмування. Тут часто використовуються різноманітні виразні засоби виділення кольором різних компонент програми (операторів, змінних, функцій і т.п.), табличні форми подання інформації, діаграми, графіки тощо.

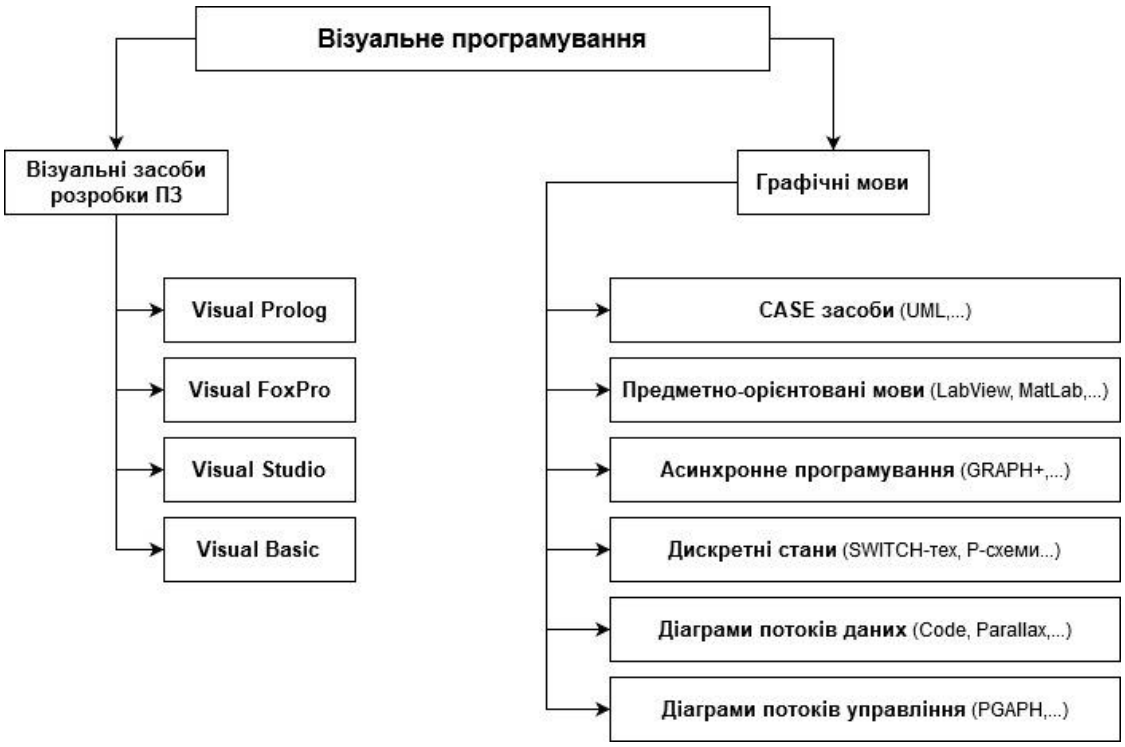

Рисунок 1.4 – Класифікація засобів візуального програмування

Подібного роду засоби візуального програмування зазвичай вирішують завдання побудови призначеного для користувача інтерфейсу і спрощують розробку додатків шляхом заміни методу написання програми на метод її конструювання. Однак візуальне уявлення алгоритмів програм вони не зачіпають.

Більш високий рівень візуалізації досягається в предметно-орієнтованих середовищах програмування. У таких середовищах візуальні образи повністю відповідають реальним об'єктам предметної області, а технологія програмування близька до традицій, що були прийняті в відповідних предметних галузях [4].

1.2 Огляд існуючих програмних рішень

Серед сучасних програмних рішень існує велика кількість різноманітних продуктів, що вирішують задачі візуалізації. Розглянемо приклади деяких з них.

Code2Flow

Code2Flow (**https://code2flow.com/**) – це інноваційний розробник діаграм, який використовує просту структуру програмування для побудови блок-схем найбільш інтуїтивно зрозумілим способом. Code2flow є легкодоступним інструментом з можливістю його редагування, що забезпечує миттєвий зворотній зв'язок на вхідний код. Програма надає доступ до багатофункціональної платформи, що допомагає розробити свої блок-схеми протягом декількох хвилин.

Code2Flow розроблявся з необхідності швидко документувати різні процеси та робочі процеси в медичному програмному забезпеченні, де прототипування та аналіз відіграє величезну роль у забезпеченні безпечної роботи програми та належної перевірки ISO.

Code2Flow є експериментальним, він проглядає вихідний код вашого проекту, шукає визначення функцій, потім виконує інший аналіз, шукаючи, куди викликаються ці функції. Він з'єднує точки і представляє блок-схему, що оцінює функціональну структуру програми. Це особливо корисно для розкручування спагетті-коду (погано спроектований та складний для розуміння програмний код) та швидкого отримання нових розробників. В процесі програмування, програмісти можуть переглядати свої змінені структури коду в протягом усього часу. Для того, щоб користувачі розуміли вихідний код набагато простіше, програма підтримує підсвічування синтаксису для вихідного коду, це означає, що при натисканні на будь-яке логічне вікно на блок-схемі, відповідний вихідний код буде виділений, це допоможе чітко подивитися програму і знайти помилки. Code2Flow має на меті забезпечити приблизний огляд структури простих проектів. Вихідний код, який є трохи езотеричним, порушить

це. Навіть при звичайному коді існує багато відомих обмежень і багато відвертих помилок.

Приклад роботи Code2Flow (рис. 1.5):

Рисунок 1.5 — Інтерфейс Code2Flow

Переваги:

- не потрібна взаємодія миші, що значно швидше створює блок-схему;
- ніякої мови для вивчення, прості правила синтаксису;
- вбудована довідкова система з прикладами;
- завантажує вихід у форматі PDF, SVG або PNG.

Недоліки:

- інтерфейс англійською мовою;
- безкоштовно можна будувати лише до 3-х блок-схем;
- мінімальний функціонал (масиви функцій не обробляються; функції, не задекларовані у початкових визначеннях класу/об'єкта (наприклад, додані пізніше), здебільшого не обробляються; у python функції, успадковані від батьківського класу, не обробляються; у python імпорт ... as ... не обробляється правильно; у javascript прототипи приведуть до непередбачуваних результатів).

В основному, Code2Flow може не схематизувати ваш вихідний код точно так, як ви могли його очікувати.

Labview

LabVIEW (англ. Laboratory Virtual Instrumentation Engineering Workbench) – це середовище розробки і платформа для виконання програм, створених на графічній мові програмування «G» фірми National Instruments (США). Перша версія LabVIEW була випущена в 1986 році для Apple Macintosh. В даний час існують версії для Unix, Linux, Mac OS і Microsoft Windows.

Програма LabVIEW є віртуальним приладом (англ. Virtual Instrument) і складається з двох частин:

- блокової діаграми, яка описує логіку роботи віртуального приладу;
- лицьовій панелі, яка описує зовнішній інтерфейс віртуального приладу [5].

LabVIEW – мова графічного програмування, в якому для створення додатків використовуються графічні образи (іконки) замість традиційного текстового коду. Від користувача пакета не потрібно знань мов програмування, але поняття про алгоритм, цикли, вихід за умовою та інші основні базові функції звичайно мати потрібно. Всі дії зводяться до простої побудови структурної схеми додатків в інтерактивній графічній системі з набором всіх необхідних бібліотечних образів, з яких збираються об'єкти, що називають віртуальними інструментами (VI).

Будь-яка LabVIEW програма містить як мінімум один VI. У термінах мови С можна досить сміливо провести аналогію з функцією з тією лише різницею, що в LabVIEW одна функція міститься в одному файлі (можна також створювати бібліотеки інструментів). Само собою зрозуміло, один VI може бути викликаний з іншого VI. В принципі кожен VI складається з двох частин

- Блок-Діаграма (Block Diagram) і Передня Панель (Front Panel). Блок-діаграма – це програмний код (точніше візуальне графічне представлення коду), а Передня панель - це інтерфейс [6].

Класичний приклад Hello, World ! (рис. 1.6):

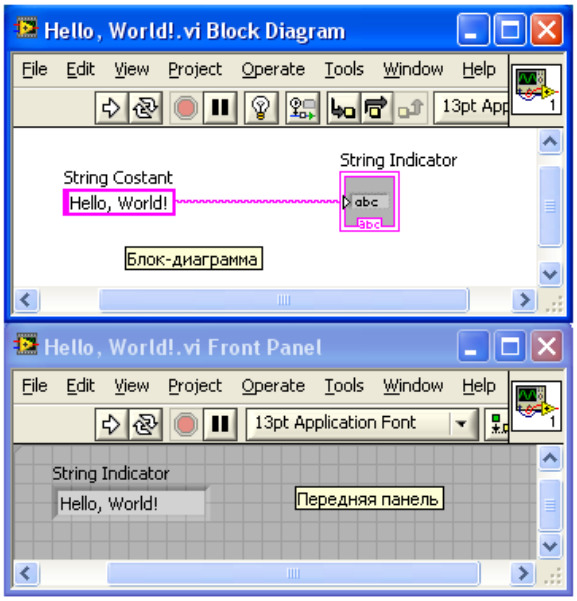

Рисунок 1.6 — Інтерфейс LabVIEW

Ось трохи складніший приклад: додавання і множення двох чисел.

```
int a, b, sum, mul;
// ...
sum = a + b;
mul = a * b;
```

Як це виглядає в LabVIEW (рис. 1.7):

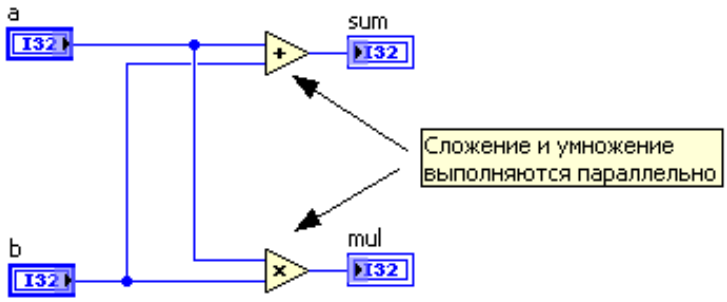

Рисунок 1.7 — Приклад додавання і множення в LabVIEW

Переваги LabVIEW:

- повноцінна мова програмування;
- широкі можливості збору, обробки та аналізу даних, управління приладами, генерації звітів і обміну даних через мережеві інтерфейси;
- драйверна підтримка понад 2000 приладів.

Недоліки:

- складна для освоєння;
- обмежена підтримка не-Windows платформ (MAC, Linux): немає драйверів, немає спеціальних toolkit-програм;
- LabVIEW – це продукт з закритим вихідним кодом.

Visustin

Це автоматизована програма створення блок-схем для розробників програм і авторів документів. Visustin виконує зворотний інжиніринг вашого вихідного коду для його розбиття на блок-схеми або діаграми діяльності UML (Activity Diagram). Visustin зчитує оператори if і else, оператори циклу і оператори переходу і створює блок-схему - в повністю автоматичному режим. Користувачу не потрібно нічого малювати вручну.

Visustin створює блок-схеми з коду на багатьох мовах програмування (ABAP, ActionScript, Ada, ASP, several assembly languages, AutoIt, BASIC, .bat files, C, C++, C#, Clipper, COBOL, ColdFusion, Delphi, Fortran, GW-BASIC, HTML, Java, JavaScript, JCL (MVS), JavaServer Pages, LotusScript, MATLAB, MXML, Pascal, Perl, PHP, PL/I, PL/SQL, PowerBASIC, PowerBuilder PowerScript, PureBasic, Python, QuickBASIC, REALbasic, Rexx, RPG, Ruby, SAS, Tcl, TSQL, Unix shell script (bash, csh, tcsh, ksh, sh), VB, VBA, VBScript, VB.Net, Visual FoxPro, XHTML, XML and XSLT).

Visustin автоматично перетворює вихідний код в блок-схеми. Автоматична компоновка забезпечує візуально оптимальні результати. Просто натисніть клавішу - і все готово. Блок-схеми візуалізують код і, за бажанням, коментарі.

Visustin підтримує як блок-схеми, так і діаграми дій UML. Можна зберігати блок-схеми в форматі PDF, у вигляді файлів зображень або веб-сторінок [7].

Приклад роботи Visustin (рис. 1.8):

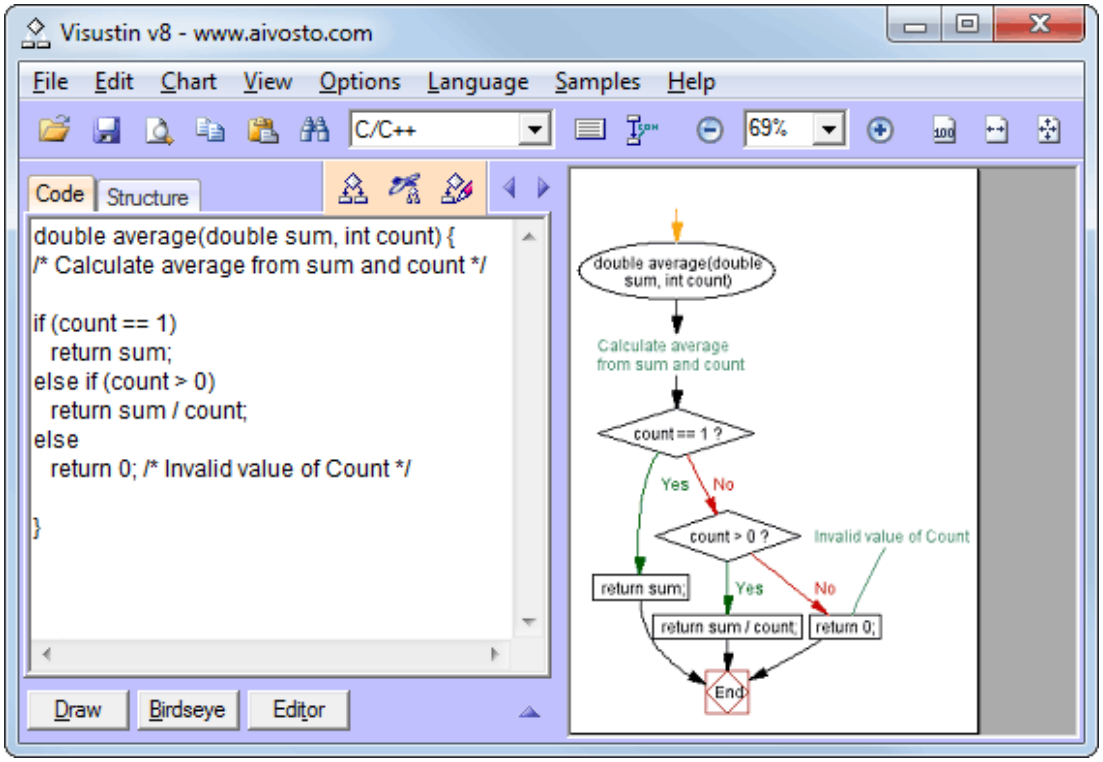

Рисунок 1.8 — Інтерфейс Visustin

Недоліки:

- інтерфейс англійською мовою;
- є платною програмою;

Crystal Flow

CRYSTAL FLOW – це корисна програма, яка дозволяє створювати схеми потоків із вихідного коду одним клацанням миші. Дозволяє використовувати блок-схеми для огляду рефактора. Коментарі до блок-схеми допомагають легше зрозуміти код, перевірити правильність логіки функції та виявити помилки. Можна експортувати блок-схеми у .bmp чи .jpg файли, а також у вигляді Visio XML-файлів. В налаштуваннях є режими блок-схеми «Тільки для коду», «Лише для коментарів» та «Код + коментарі». Ця програма має широкий функціонал.

Рівень ароматів декількох блок-діаграм оптимальний, цикл та умови, залежні від площі.

Відбувається автоматична синхронізація блок-схеми коду під час перегляду.

Програму можна встановити на WinNT 4.x, Windows2000, Windows2003, Windows Vista Starter, Windows Vista Home Basic, Windows Vista Home Premium, Windows Vista Business, Windows Vista Enterprise, Windows Vista Ultimate, Windows Vista Home Basic x64, Windows Vista Home Premium x64, Windows Vista Business x64, Windows Vista Enterprise x64, Windows Vista Ultimate x64, Unix, Linux, консоль Linux, Linux Gnome, Linux GPL, Linux Open Source, Mac OS 9, Mac OS X, Mac OS X 10.1, Mac OS X 10.2, Mac OS X 10.3, Mac OS X 10.4, Mac OS X 10.5 [8].

Приклад роботи Crystal Flow (рис. 1.9):

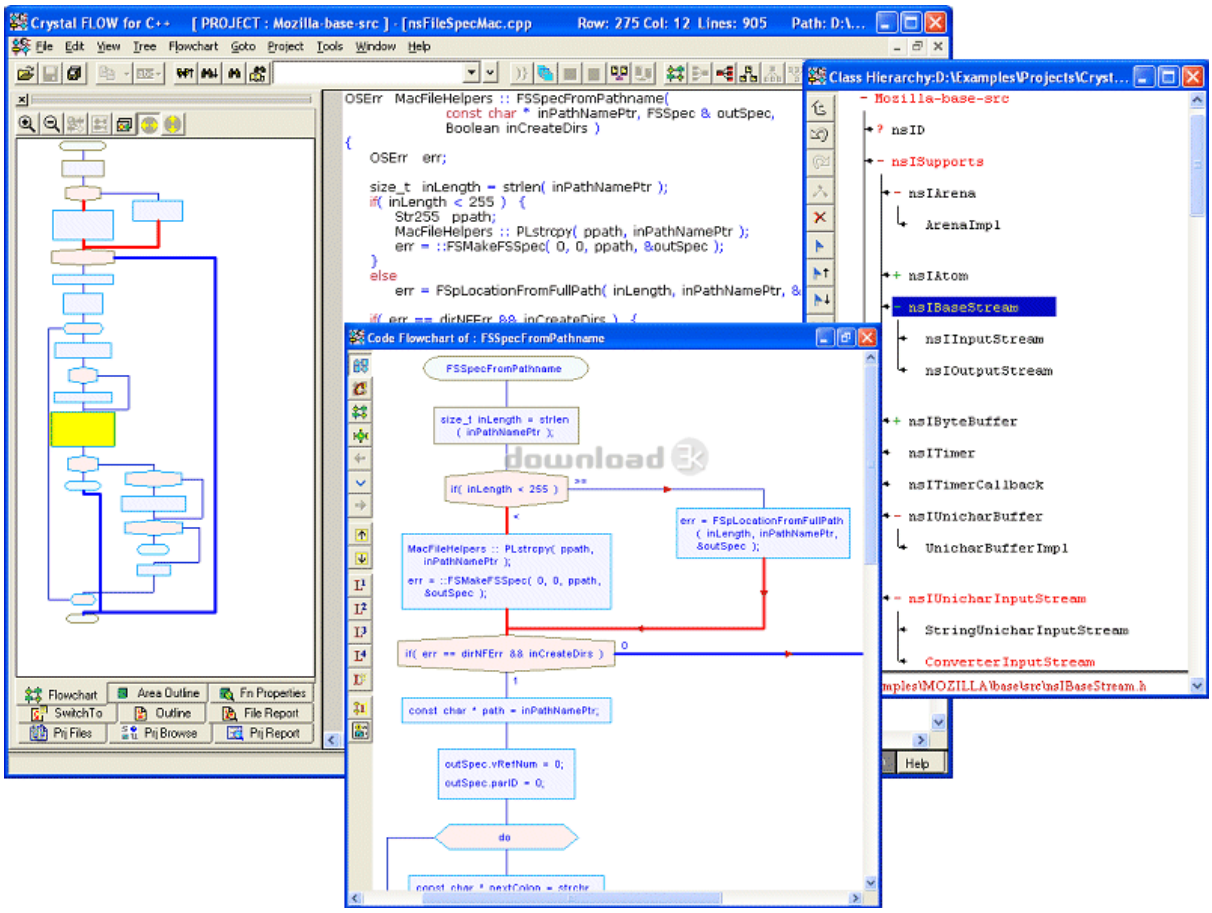

Рисунок 1.9 – Інтерфейс Crystal Flow

Недоліки:

- англомовний інтерфейс;
- є складною у використанні;
- лише 3-х денне випробування в незареєстрованій версії;
- лише для C/C++.

AthTek Flowchart to Code

AthTek Flowchart to Code розроблений як супровідний інструмент програмування до конвертера AthTek Code to FlowChart. Це може допомогти інженерам-програмістам без зусиль перетворювати блок-схему в код. Користувачу більше не потрібно вручну вводити рядок вихідного коду за рядком. За допомогою AthTek Flowchart to Code, єдине, що потрібно зробити, – це побудувати блок-схему програми, і тоді ви отримаєте вихідний код автоматично. Це може абсолютно прискорити цикли розвитку і звільнити інженерів програмного забезпечення від повторних і механічних робіт.

AthTek Flowchart to Code підтримує генерування вихідного коду на декількох мовах, включаючи C, C ++, C #, Java, JavaScript та Delphi. Він також може експортувати блок-схему в MS Word / Visio / SVG / BMP та роздрукувати. Вихідний код буде згенерований лише одним натисканням.

AthTek Flowchart to Code – це професійний інструмент управління проектами для інженерів програмного забезпечення. Він має найелегантніші діаграми серед усіх аналогічних інструментів для блок-діаграм. Це програма №1 для конвертора блок-схеми в Пошуку Google [9, 10].

Приклад роботи AthTek Flowchart to Code (рис. 1.10):

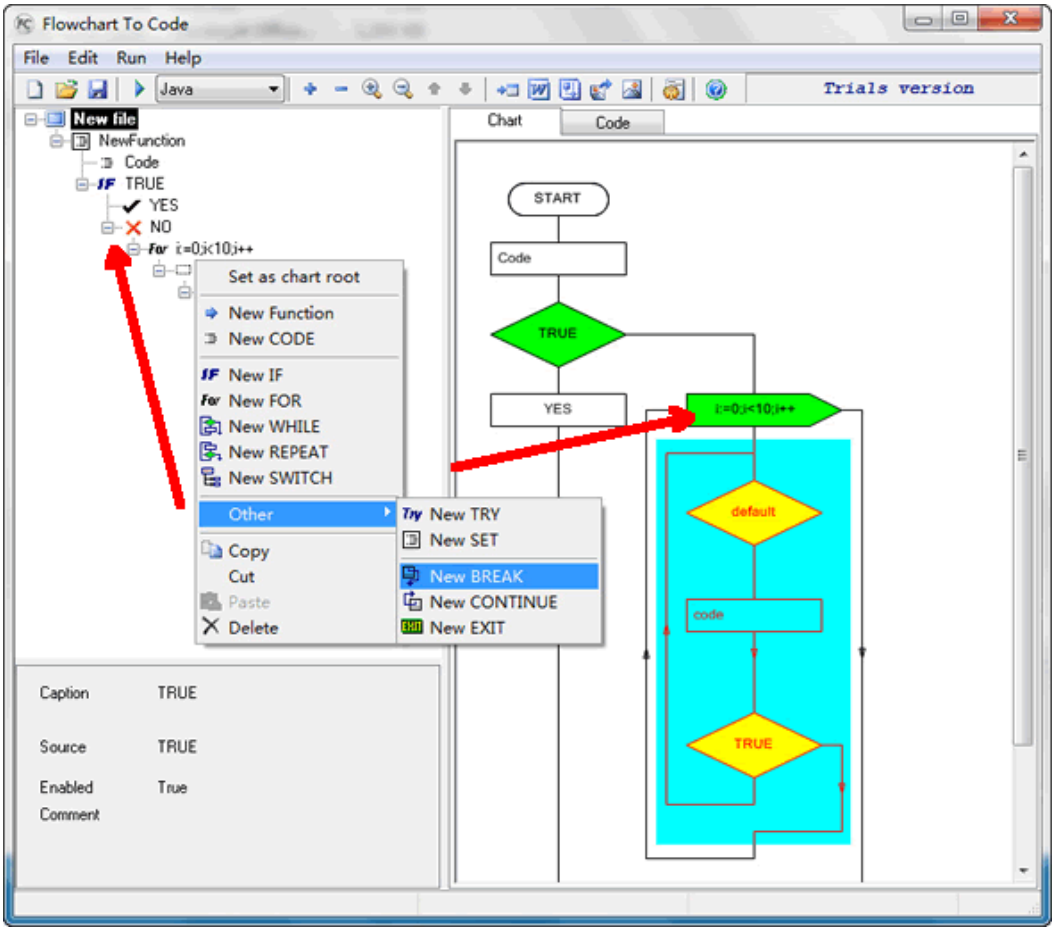

Рисунок 1.10 — Інтерфейс AthTek Flowchart to Code

Недоліки:

- англомовний інтерфейс;
- складна у використанні.

Flowgorithm

Flowgorithm – це графічний інструмент, який дозволяє користувачам писати та виконувати програми за допомогою блок-схем. Підхід покликаний підкреслювати алгоритм, а не синтаксис конкретної мови програмування. Діаграма може бути перетворена на кілька основних мов програмування. Flowgorithm був створений в Державному університеті Сакраменто.

Flowgorithm може інтерактивно переводити програми блок-схеми у вихідний код, написаний іншими мовами програмування. Коли користувач переходить через свою блок-схему, відповідний код у перекладеній програмі автоматично виділяється.

Підтримуються наступні мови програмування: C++, C#, Delphi, ·Java, JavaScript, Lua, Perl, PHP, Python, QBasic, Ruby, Swift 2 & 3, Visual Basic for Applications, Visual Basic .NET

Окрім англійської, Flowgorithm підтримує інші мови інтерфесу: арабська, китайська, чеська, голландська, фарси, французька, галицька, німецька, угорська, індонезійська, італійська, японська, монгольська, польська, португальська, російський, словенська, іспанська, тайська, турецька.

Класичний метод взаємодії з комп’ютером – це використання "консолі". Використовуючи такий підхід, програма відображає вихід на текстовому екрані, а користувач вводить дані за допомогою клавіатури. Іноді консоль дозволяє тексту змінювати кольори, але, здебільшого, це білий текст на чорному фоні.

Консоль працює, але її простий інтерфейс може ускладнити розмежування вводу користувача та виводу програми. Отже, замість використання текстового екрана, Flowgorithm намагається зробити його схожим на типове вікно швидкого обміну миттєвими повідомленнями. З точки зору програмістів, схоже, вони надсилають текстові повідомлення з комп'ютером.

"Chat bubbles" кодуються кольором, щоб відповідати формам введення та виводу, що використовуються на блок-схемі. На скріншоті праворуч введення користувача відображається синім кольором, тоді як вихід програми відображається зеленим кольором.

Якщо користувач не бажає використовувати режим "Chat bubbles", він має можливість перемикатися між цими режимами, в тому числі, переходити до класичного звичайного тексту [11].

Приклад роботи Flowgorithm (рис. 1.11):

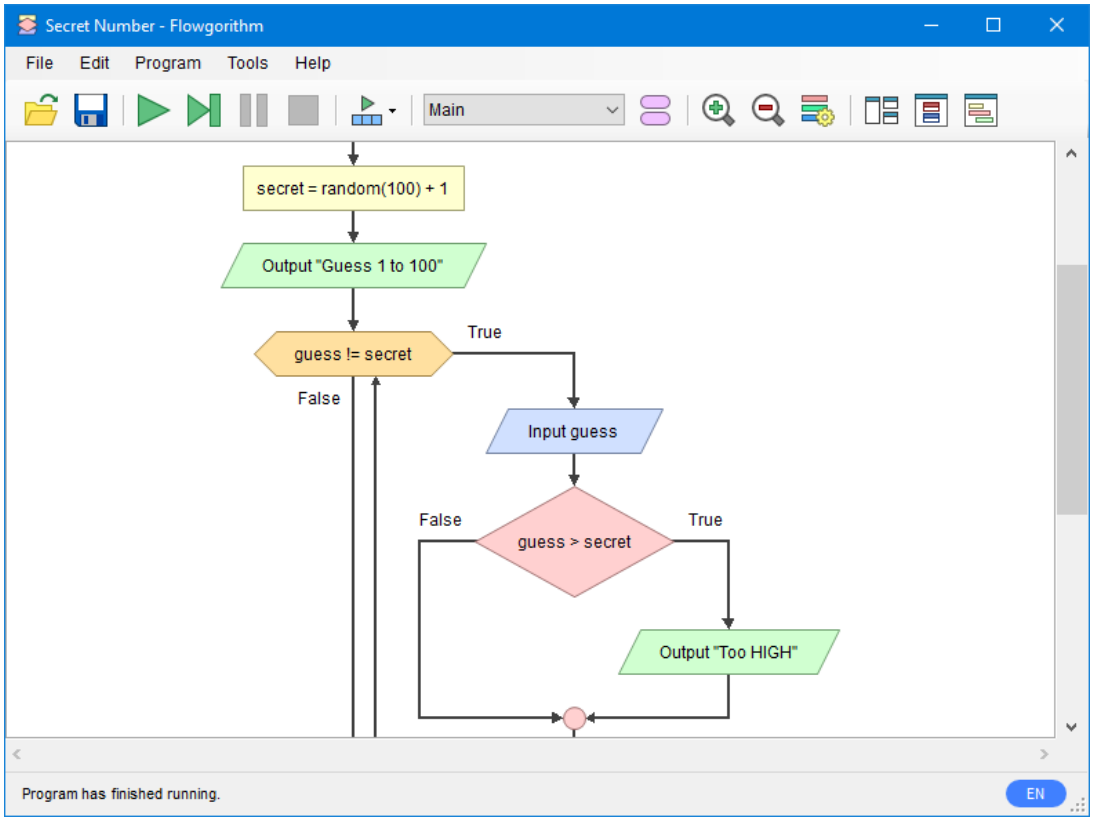

Рисунок 1.11 – Інтерфейс Flowgorithm

Недоліки:

- у чат вікні зникають власні відповіді;
- бракує break і continue;
- бракує глобальних і / або статичних змінних;
- необхідно поправляти роботу конвертера вручну;
- функція не може віддати масив (за допомогою return).

1.3 Постановка задачі

Проаналізувавши існуючі програмні рішення було виявлено ряд недоліків. На основі цього було вирішено розробити систему візуального програмування за криптографічним алгоритмом, яка дозволяє на основі побудованої блок-схеми відобразити відповідний програмний код і здійснити шифрування даних, з метою дослідження криптографічних алгоритмів.

Метою роботи є створення додатку, що має розв’язувати такі задачі:

- побудова блок-схеми криптографічного алгоритму для програмного коду;
- перевірка цієї блок-схеми;
- відображення програмного коду криптоалгоритму мовою C#;
- реалізація шифрування повідомлень за допомогою визначених трьох криптографічних алгоритмів;
- програма повинна працювати як окремий незалежний додаток.

Таким чином, при розробці даної програми буде досягнуто скорочення часу за рахунок спрощення процедури написання програмного коду, який виконується вручну.

2 КРИПТОГРАФІЧНІ АЛГОРИТМИ ТА ЇХ МАТЕМАТИЧНІ МОДЕЛІ

2.1 Види шифрів

З розвитком розгалужених інформаційних систем та мереж актуальним стає питання забезпечення цілісності, конфіденційності та надійності інформаційних ресурсів за допомогою використання новітніх методів криптографії. Сьогодні, будуючи криптографічні системи захисту інформації, потрібно звертати увагу на середовище їх функціонування, можливість реалізації захищених каналів передачі інформації, доцільність використання певних алгоритмів шифрування, здатність системи протистояти різним видам загроз. Основна вимога при впровадженні криптосистеми – використання шифрування за допомогою криптоалгоритму, оскільки це є гарантією стабільності системи до різних несанкціонованих дій зловмисника.

Розглянемо короткий блок про симетричні та асиметричні криптоалгоритми (їх коротка характеристика і найпоширеніші сучасні шифри):

1. Тайнопис. Відправник і одержувач роблять над повідомленням перетворення, відомі лише їм двом. Стороннім особам невідомий сам алгоритм шифрування. Деякі фахівці вважають, що тайнопис не є криптографією взагалі.

2. Криптографія з ключем. Алгоритм впливу на передані дані відомий усім стороннім особам, але він залежить від деякого параметра – "ключа", яким володіють лише відправник і одержувач.

- симетричні криптоалгоритми. Для зашифрування і розшифрування повідомлення використовується один і той же блок інформації (ключ).
- асиметричні криптоалгоритми. Алгоритм такий, що для зашифрування повідомлення використовується "відкритий" ключ, відомий усім бажаючим, а для розшифрування –"закритий" ключ, яким володіє тільки одержувач.

В залежності від кількості ключів, які застосовуються у конкретному алгоритмі, їх поділяють на:

- безключові криптографічні алгоритми (КА) – не використовують в обчисленнях ніяких ключів;
- одноключові КА – працюють з одним додатковим ключовим параметром (якимсь таємним ключем);
- двухключові КА – на різних стадіях роботи в них застосовуються два ключових параметри: секретний та відкритий ключі.

В залежності від характеру впливів, що здійснюються над даними, алгоритми поділяються на:

- перестановочні – блоки інформації (байти, біти, більші одиниці) не змінюються самі по собі, але змінюється їх порядок проходження, що робить інформацію недоступною сторонньому спостерігачеві.
- підстановочні – самі блоки інформації змінюються за законами криптоалгоритму. Переважна більшість сучасних алгоритмів належить цій групі.

Залежно від розміру блоку інформації криптоалгоритми поділяються на:

- потокові шифри – одиницею шифрування є один біт. Це шифр, в якому кожен символ відкритого тексту перетворюється в символ шифрованого тексту в залежності не тільки від використовуваного ключа, а й від його розташування в потоці відкритого тексту. Схема застосовується в системах передачі потоків інформації, тобто в тих випадках, коли передача інформації починається і закінчується в довільні моменти часу і може випадково перериватися. Найбільш поширеними представниками потокових шифрів являються скремблери.
- блочні шифри – одиницею шифрування є блок з декількох байтів (в даний час 4-32). Результат шифрування залежить від усіх вихідних байтів цього блоку. Схема застосовується при пакетній передачі інформації та кодуванні файлів [12].

Симетричні криптографічні системи

До симетричних систем шифрування відносяться такі системи, в яких для шифрування і для розшифрування використовується один і той же ключ.

Тому такі системи називають також одноключовими.

Залежно від типу перетворення, виконуваного над відкритим текстом при шифруванні, симетричні системи шифрування базуються на наступних базових класах.

Моно- і багатоалфавітні підстановки.

Моноалфавітні підстановки – це найпростіший вид перетворень, що полягає в заміні символів вихідного тексту. У випадку моноалфавітної підстановки кожен символ вихідного тексту перетворюється в символ шифрованого тексту по одному і тому ж закону. При багатоалфавітній підстановці закон перетворення змінюється від символу до символу. Для забезпечення високої криптостійкості потрібно використовувати більші ключі. До цього класу належить так звана криптосистема з одноразовим ключем, що володіє абсолютною теоретичної стійкістю, але, на жаль, незручна для практичного застосування.

Перестановки.

Так само нескладний метод криптографічного перетворення, що полягає в перестановці місцями символів вихідного тексту за деяким правилом. Шифри перестановок в даний час не використовуються в чистому вигляді, так як їх криптостійкость недостатня.

Блочні шифри.

Такі шифри являють собою сімейство обернених перетворень блоків (частин фіксованої довжини) вихідного тексту. Фактично блоковий шифр – це система підстановки блоків. В даний час блокові шифри найбільш поширені на практиці.

Гамування.

Являє собою перетворення вихідного тексту, при якому символи вихідного тексту додаються (за модулем, рівним потужності алфавіту) з символами псевдовипадкової послідовності, що генерується за деяким правилом. Власне кажучи, гамування не можна цілком виділити в окремий клас криптографічних

перетворень, так як ця псевдовипадкова послідовність може вироблятися, наприклад, за допомогою блочного шифру. У разі, якщо послідовність є істинно випадковою і кожен її фрагмент використовується тільки один раз, ми отримуємо криптосистему з одноразовим ключем.

Кількість відомих на сьогодні симетричних криптосистем дуже велика, серед яких DES, AES, RC2, RC5, Blowfish, IDEA та ін [13].

Асиметричні криптографічні системи

Однією з головних проблем практичного використання розглянутих систем шифрування є проблема розподілу секретних ключів між абонентами та проблема зберігання цих ключів. Якщо в системі є N абонентів, то для забезпечення обміну інформацією між будь-якими двома абонентами потрібно буде генерувати та поширювати секретні ключі N (N-1) / 2, і кожен абонент буде змушений зберігати (N-1) секретний ключ для обміну інформацією з іншими абонентами. Вирішити проблему розподілу ключів допомагає використання асиметричних криптографічних систем. У цих системах для обміну даними використовуються два ключі, один з яких є секретним, а інший – відкритим, тобто загальнодоступним. З цієї причини асиметричні системи називаються також двохключовими.

Всі асиметричні криптографічні системи засновані на використанні односторонніх функцій.

Функція $F: X \rightarrow Y$ називається односторонньою, якщо виконуються наступні дві умови:

1) існує ефективний алгоритм, що обчислює $F(x)$ для будь-якого $x \in X$;

2) не існує ефективного алгоритму інвертування функції F, тобто алгоритму, що дозволяє визначити значення x за значенням $F(x)$.

У даному визначенні «ефективним» називається поліноміальний алгоритм, тобто алгоритм, який для отримання результату для входу довжини n витрачає не більше $P(n)$ кроків, де P – деякий поліном.

У даний час теорія алгоритмів не дозволяє довести, що не існує ефективних алгоритмів вирішення того чи іншого завдання. Тому, строго кажучи, нам невідома жодна одностороння функція. Однак запропоновано декілька функцій, які можуть виявитися односторонніми – для цих функцій в даний час, незважаючи на інтенсивні дослідження, ще невідомі ефективні алгоритми інвертування.

Найбільш часто використовуються «односторонні» функції, запозичені з теорії чисел:

- функція F (a,b) = ab, тобто добуток двох чисел. Якщо a і b - прості числа, то за відомим c = ab можна однозначно визначити a і b. Ця задача називається задачею факторизації числа. Досі невідомий жоден поліноміальний алгоритм для вирішення завдання факторизації, хоча для обчислення добутку чисел (тобто самої функції F) такі алгоритми відомі.
- функція F (a, n) = $a^n(mod\ M)$, де a, n і M - цілі числа. При відомих a, n і M значення $a^n(mod\ M)$, може бути обчислено за поліноміальну кількість кроків. Однак для оберненої задачі – визначити n за відомим a, b і M (задача дискретного логарифмування) – поліноміальні алгоритми, в загальному випадку, поки невідомі.

Не будь-яка одностороння функція може бути використана для шифрування. Дійсно, якщо перетворити відкритий текст t за допомогою односторонньої функції: c = F(t), то розшифрувати отриманий текст, тобто за *c* відновити *t*, не зможе вже ніхто, у тому числі і законний одержувач. Для використання в криптографії необхідно, щоб задача інвертування шифрувального перетворення (тобто обчислення t за F (t)) мала розв'язок за прийнятний час, але зробити це може тільки той, хто знає секретний ключ.

Процедура відкритого розподілу ключа

Асиметричні криптографічні системи дозволяють розподіляти секретні ключі через відкриті канали, тобто канали, які потенційно можуть бути прослухані противником. Така процедура відкритого розподілу ключів була

вперше опублікована в 1976 році в роботі У. Діффі та М.Е.Хеллмана «Нові напрямки в криптографії».

В основі процедури Діффі-Хеллмана лежить використання односторонньої функції дискретного піднесення в степінь: $F(x) = g^x(mod\ p)$, де x - ціле число ($1 \leq x \leq p\text{-}1$), p - просте число, g - первісний корінь за модулем p.

Первісним коренем за модулем p називається таке ціле число g ($g < p$), для якого:

- всі степені $g^1(mod\ p)$, $g^2(mod\ p)$, ..., $g^{p-1}(mod\ p)$ різні;
- для будь-якого цілого числа a, такого що $1 \leq a \leq p\text{-}1$, знайдеться n, при якому $a = g^n(mod\ p)$.

Зводячи число g в степені 1, 2, ..., p-1 (по модулю p) ми отримаємо всі числа від 1 до p-1, що утворюють Zp * (мультиплікативну групу кільця Zp). Тому таке число g називається також генератором групи Zp *.

Процедура Діффі-Хеллмана для відкритого розподілу ключів полягає в наступному. Для початку вибирається велике просте число p і число g - первісний корінь за модулем p. Для забезпечення стійкості число p повинне мати довжину, більшу або рівну 512 біт (тут і далі довжиною цілого числа будемо називати кількість біт в двійковому записі цього числа), і розкладання числа p-1 на множники повинно містити хоча б один великий простий множник (наприклад, $p\text{-}1 = 2q$, де q - просте число). При такому виборі числа p в даний час не існує ефективного алгоритму для вирішення завдання інвертування функції.

Кожен абонент в якості свого секретного ключа вибирає деяке випадкове число, за яким обчислює свій відкритий ключ: $A = g^a(mod\ p)$, $B = g^b(mod\ p)$. Всі абоненти розміщують свої відкриті ключі в загальнодоступний довідник. Після цього два абонента A і B беруть із загальнодоступного довідника відкриті ключі один одного і обчислюють загальний таємний ключ:

1) абонент A обчислює:

$z_A = B^a(mod\ p) = g^{ab}(mod\ p)$;

2) абонент В обчислює :

$$z_B = A^b (mod\ p) = g^{ab} (mod\ p).$$

Таким чином, після виконання описаної процедури у абонентів А і В є загальне число $z_A = z_B$. Це число вони при обміні повідомленнями можуть використовувати як ключ для шифрування (наприклад, методом гамування). Для того, щоб зловмисник визначив секретний ключ, йому необхідно вирішити задачу дискретного логарифмування.

Відомими криптосистемами з відкритим ключем є RSA, система Ель-Гамаля, система Рабіна, система Мак-Еліса тощо [14].

2.2 Моделі симетричних шифрів

Розглянемо афінний шифр та шифр DES, що є моделями симетричних шифрів.

Афінний шифр

Традиційні шифри з симетричним ключем можна розділити на дві великі категорії: шифри підстановки та шифри перестановки. У шифрі підстановки замінюється один символ у зашифрованому тексті на інший символ; у шифрі перестановки — міняються місцями позиції символів у початковому тексті.

Шифр підстановки замінює один символ іншим. Якщо символи в початковому тексті — символи алфавіту, то одна літера замінюється іншою. Наприклад, можна замінити літеру *A* літерою *D,* використовуючи англійський алфавіт, а літеру *T* — літерою *Z*. Якщо символи — цифри (від *0* до *9*), то можна, наприклад, замінити *3* на *7*, а *2* на *6*. Шифри підстановки можуть бути розбиті на дві категорії: *моноалфавітної* (*одноалфавітної*) й *багатоалфавітної* підстановки. Для шифрів підстановки довжина алфавіту відкритого тексту і довжина алфавіту зашифрованого тексту однакова.

При використанні цих шифрів літери алфавіту зручно ототожнювати з їх порядковими номерами [15].

Афінна система шифрування відноситься до класу шифрів, заснованих на аналітичних перетвореннях даних.

Розглядаючи безліч цілих чисел Z_m з двома операціями додавання і множення за модулем m можна отримати систему підстановок, яку називають афінною системою шифрування Цезаря або аффінним шифром. Афінний шифр є узагальненням системи Цезаря.

Визначимо таке перетворення $E_{a,b} : Z_m \rightarrow Z_m$:

$E_{a,b}(x) = (ax+b) \bmod m$,

де в якості ключа k використовується пара цілих чисел (a, b), які відповідають умовам

$0 £ a,b < m$, НСД$(a,m)=1$.

У даному перетворенні літера, що відповідає числу x у відкритому тексті, замінюється на літеру шифротекста, відповідну числовому значенню $y = (ax+b) \bmod m$.

Слід зауважити, що перетворення $E_{a,b}(x)$ є взаємно однозначним відображенням на множині Z_m тільки в тому випадку, якщо НСД (a, m) = 1, тобто a й m повинні бути взаємно простими числами.

Ця умова взаємної простоти необхідна для забезпечення ін'єкційних відображень $E_{a,b}(x) = ax+b \bmod m$. Якщо воно не виконується, можлива ситуація, коли дві різні букви відображаються в одну (виникає неоднозначність розшифрування), а деякі букви відсутні в шифротекста, так як ніякі букви в них не відображаються.

Перевагою афінної системи є зручне управління ключами – ключі шифрування і розшифрування представляються в компактній формі у вигляді пари чисел (a, b). У порівнянні з простою системою шифрування Цезаря, кількість можливих ключів значно більше і алфавітний порядок слів при шифруванні не зберігається.

Афінна система використовувалася на практиці кілька століть назад, а сьогодні її застосування обмежується здебільшого ілюстраціями основних криптологічних положень.

Процес шифрування описується формулою:

$Y = (ax + b) \bmod n$,

де x – номер символу відкритого тексту в алфавіті,

Y – номер символу шифрованого тексту в алфавіті,

n – потужність алфавіту (кількість символів),

(a, b) – ключ шифрування. Значення ***a*** і ***b*** обираються із множини $Z_n=\{0,1,2, ..., n-1\}$, при чому ***a*** і ***n*** взаємно прості числа, тобто **НСД(a, n)=1**.

Для українського алфавіту таблиця відповідності кожній літері з порядковим номером (табл. 2.1):

Таблиця 2.1 Відповідність літер укр. алфавіту порядковим номерам

А	**Б**	**В**	**Г**	**Д**	**Е**	**Є**	**Ж**	**З**	**И**	**І**	**Ї**	**Й**	**К**	**Л**	**М**	**Н**	**О**	**П**	**Р**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
С	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ь**	**Ю**	**Я**								
20	21	22	23	24	25	26	27	28	29	30	31								

Приклад (табл. 2.2)**:** Зашифрувати текст "**Криптографія**".

Ключ: a=3; b= 5 .

Таблиця 2.2 Поетапне кодування повідомлення афінним шифром

Текст	**К**	**Р**	**И**	**П**	**Т**	**О**	**Г**	**Р**	**А**	**Ф**	**І**	**Я**
x	13	19	9	18	21	17	3	19	0	23	10	31
3x+5	44	62	32	59	68	56	14	62	5	74	35	98
(3x+5)mod32	12	30	0	27	4	23	14	30	5	10	3	2
Шифротекст	Й	Ю	Ф	Ш	Д	Х	Л	Ю	Е	І	Г	В

Зашифрований текст "**йюфшдхлюеігв**".

Шифр DES

DES (Data Encryption Standard) – алгоритм блокового шифрування з довжиною блоку 64 біти і симетричним ключем довжиною 56 біт. На практиці

ключ має довжину 64 біти, з яких кожний восьмий використовується для контролю парності байту.

Шифр DES ґрунтується на схемі Фейстеля з такими параметрами:

- довжина блоку – 64 біти,
- кількість раундів – 16,
- розмір ключа – 56 бітів,
- розмір кожного з підключів K_1, K_2, …, K_{16} – 48 бітів.

Структура алгоритму DES показана на рис. 2.1.

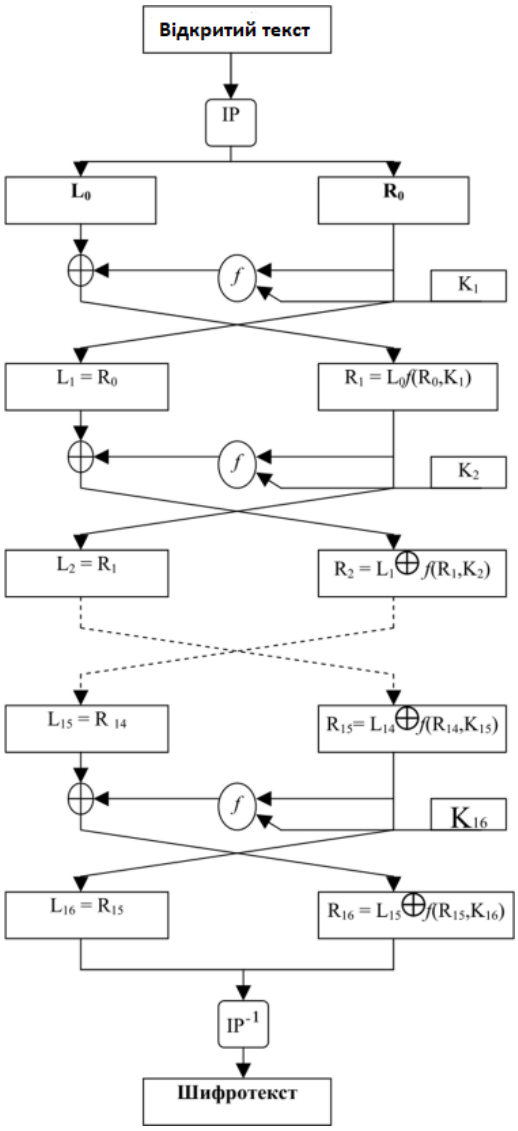

Рисунок 2.1 — Структура алгоритму DES

Алгоритм DES описується за допомогою трьох етапів:

1. До вхідного блоку довжиною 64 біти застосовується фіксована початкова перестановка IP, яка описується виразом (L0,R0) ← IP (Вхідний блок).

Тут L0 та R0 називаються лівою і правою половинами блоку, кожна з яких має довжину 32 біти.

Перестановка IP застосовується для того, щоб здійснити початкове розсіювання статистичної структури повідомлення. Вона є фіксованою і відкритою (табл. 2.3).

Таблиця 2.3 Матриця початкової перестановки IP

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	36	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

2. Виконуються 16 раундів з таких операцій:

- права половина блоку перетворюється функцією f з використанням поточної ключової послідовності довжиною 48 біт, яка знімається з виходу блоку вироблення ключової послідовності.
- результат перетворення правої половини складається за модулем 2 з лівою частиною, а сума записується у вихідний регістр, при цьому вихідна права половина за допомогою операції зсуву записується на місце вихідної лівої половини.

Їх можна описати рівняннями:

$L_i = R_i$, $R_i = L_{i-1} f(R_{i-1}, K_i)$, де K_i– ключ раунду, який складається з 48-бітового підрядка 56-бітного вихідного ключа, f – функція шифрування, яка представляє собою перестановочний шифр. Ці операції перестановки забезпечують значний рівень «дифузії повідомлення».

3. До результату 16-го раунду (L_{16}, R_{16}) застосовується завершуюча перестановка, яка є оберненою до початкової перестановки (табл. 2.4).

В кожному раунді перетворення F складається з таких п’яти кроків:

1. **Перестановка з розширенням**. Права половина з 32 бітів розширюється до 48 бітів і перемішується. Ця операція передбачає дописування у

Таблиця 2.4 Матриця завершуючої перестановки IP

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Структура функції *f* (рис. 2.2)

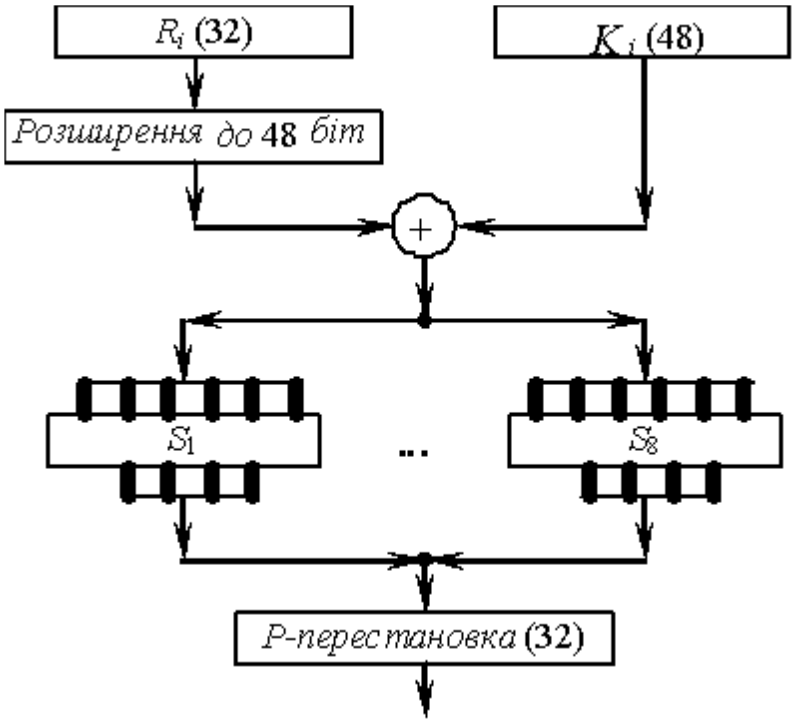

Рисунок 2.2 — Структура функції *f*

вихідну послідовність окремих бітів у відповідності з підстановкою розширення (табл. 2.5).

Таблиця 2.5 Матриця підстановки розширення

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Підстановка розширення забезпечує, що один вхідний біт впливає на дві заміни через S-блоки, що створює «лавиноподібний» ефект – мала відмінність між двома наборами вхідних даних перетворюється у велику на виході.

2. **Додавання з підключем.** До отриманого після перестановки з розширенням рядка з 48 бітів і підключа довжиною також 48 бітів застосовується операція виключаю АБО, тобто кожна пара відповідних бітів складається за модулем 2.

48-бітний підключ отримується з 56-бітного ключа у такий спосіб. Біти ключа записуються у два 28-бітні циклічних регістрів зсуву, які переміщують вміст в кожному такті на кількість бітів, що залежить від номера раунду (табл. 2.6).

Таблиця 2.6 Значення циклічного зсуву в 16-ти раундах шифрування

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Результуюча послідовність отримується шляхом вибірки 48 бітів з вмісту регістрів (табл. 2.7).

Таблиця 2.7 Матриця вибірки з вмісту регістрів

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

3. **Розщеплення.** Результат додавання з підключем розбивається на 6 частин по 8 бітів у кожному, кожна з яких передається в один з восьми S-блоків.

4. **S-блок.** Перетворює набір з 6 бітів в набір з 4 бітів.

S-блоки є нелінійними компонентами алгоритму DES, які і забезпечують криптостійкість шрифту. Кожний S-блок представляє собою пошукову таблицю з чотирьох рядків і шістнадцяти стовпців (табл. 2.8). Шість бітів, що входять до блоку, визначають який рядок і стовпець необхідно використовувати для заміни. Перший і шостий біт задають номер рядка, а інші – номер стовпця. Вихід S-блоку – бітове представлення числа відповідної комірки таблиці.

Таблиця 2.8 Підстановки в S-блоках

S1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	6	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	12
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

5. **Р-блок**. Вихід S-блоків з восьми 4-бітових елементів надходить до Р-блоку, в якому відбувається перестановка (табл. 2.9).

Таблиця 2.9 Матриця Р-перестановки

16	7	20	21	29	12	28	17	1	15	23	2	65	18	31	16
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Розшифровування в алгоритмі DES відбувається аналогічно шифруванню з тією різницею, що вибірка ключової послідовності в раундах розшифровування буде оберненою, тобто К16, К15 … К1.

Схема формування ключів у DES

Ключ алгоритму DES звичайно представляють 64-бітовим числом, але кожний 8-й біт не використовується і є просто перевіркою на парність. На кожній ітерації з 56-бітового ключа обчислюється 48-бітовий раундовий ключ. Схема формування ключів показана на рис. 2.3. Спочатку ключ розбивається на 2 блоки по 28 біт, до кожного з яких застосовується циклічний зсув на C_i позицій, а потім - перестановка із стисненням, тобто з 28 бітів вибираються 24. Величини C_i, i=0,…,15 задані. Два одержаних 24-бітових вектори шляхом конкатенації і утворюють черговий цикловий ключ K_i. 28- бітові вектори, що утворились після циклічного зсуву, йдуть на вхід наступного циклу циклу формування ключів.

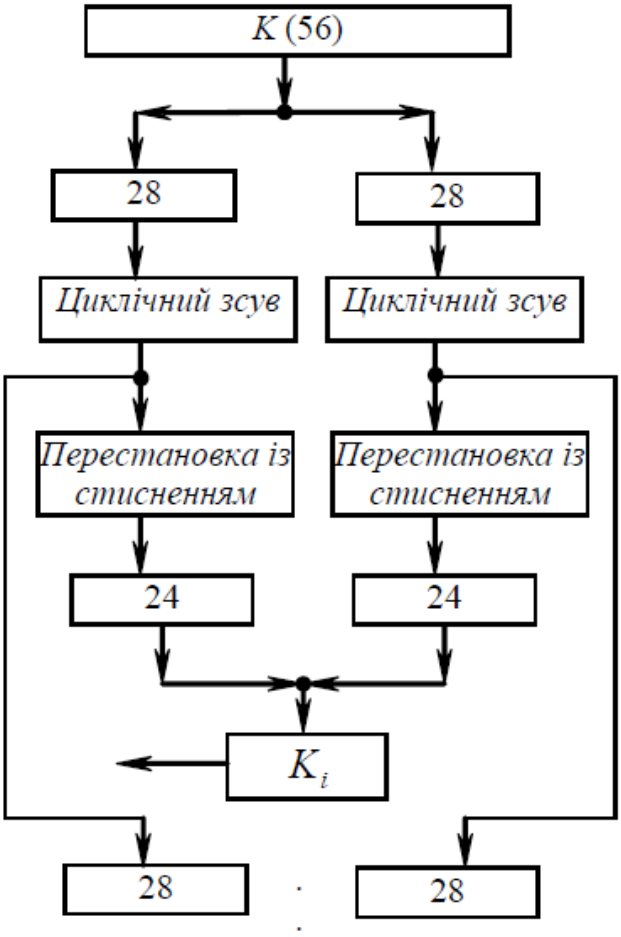

Рисунок 2.3 — Схема формування раундових ключів у DES

Стійкість DES

Основними недоліками DES, що суттєво зменшують рівень його безпеки, є такі:

- **наявність слабких ключів.** Їх виникнення пов’язане з тим, що в кожному раунді при генерації підключа використовуються два регістри зсуву. При цьому може виникати ситуація, коли для кожного підключа буде генеруватись одна, або лише дві, чотири ключових послідовності. Такі ключі називаються слабкими. Приклд слабкого ключа: 1F1F1F1F 0E0E0E0E (з урахуванням бітів парності); однакові підключі будуть генеруватись у всіх 16 раундах.

- **невелика довжина ключа**. На сучасному рівні розвитку мікропроцесорних засобів довжини ключа 56 біт (або 64 біти з котролем парності) не забезпечує достатнього рівня захисту.

-**надлишковість ключа**. Наявність контролю парності кожного байту ключа, що і призводить до надлишковості, дозволяє відновлювати ключ при виникненні помилок в пам'яті. **Використання статичних підстановок в S-блоках** [14].

2.3 Моделі асиметричних шифрів

Як приклад асиметричної системи шифрування розглянемо шифр Ель-Гамаля.

Шифр, запропонований Ель-Гамалем (Taher ElGamal), вирішує задачу передачі між абонентами зашифрованих повідомлень незахищеними каналами, але використовує при цьому лише одну пересилку повідомлення. Фактично тут використовується протокол Діффі-Хеллмана для формування секретного ключа і далі повідомлення шифрується шляхом множення його на цей ключ.

Для групи абонентів A,B,C,… вибирається велике просте число p і деяке число g, 1< g < p -1, таке, що всі числа з множини {1,2,…, p-1} можуть бути представлені як різні степені числа $g \ mod \ p$. Числа p і g передаються всім абонентам у відкритому вигляді.

Потім кожний абонент вибирає своє секретне число x, 1< x < p і обчислює відповідне йому число $y = g^x (mod\ p)$.

Процес пересилання повідомлення M від абонента A до абонента B (рис. 2.4):

1. Абонент A вибирає випадкове число k, $1 \le k \le p-1$, обчислює числа $a = g^k (mod\ p)$, $b = M \times y^k (mod\ p)$ і передає пару чисел (a,b) абоненту B.

2. Абонент B, отримавши (a,b), обчислює:

$$M' = b \times (a^x)^{-1} (mod\ p) = b \times a^{(p-1-x)} (mod\ p).$$

За властивістю криптографічної системи Ель-Гамаля:

1) абонент B отримав повідомлення, тобто M' = M;

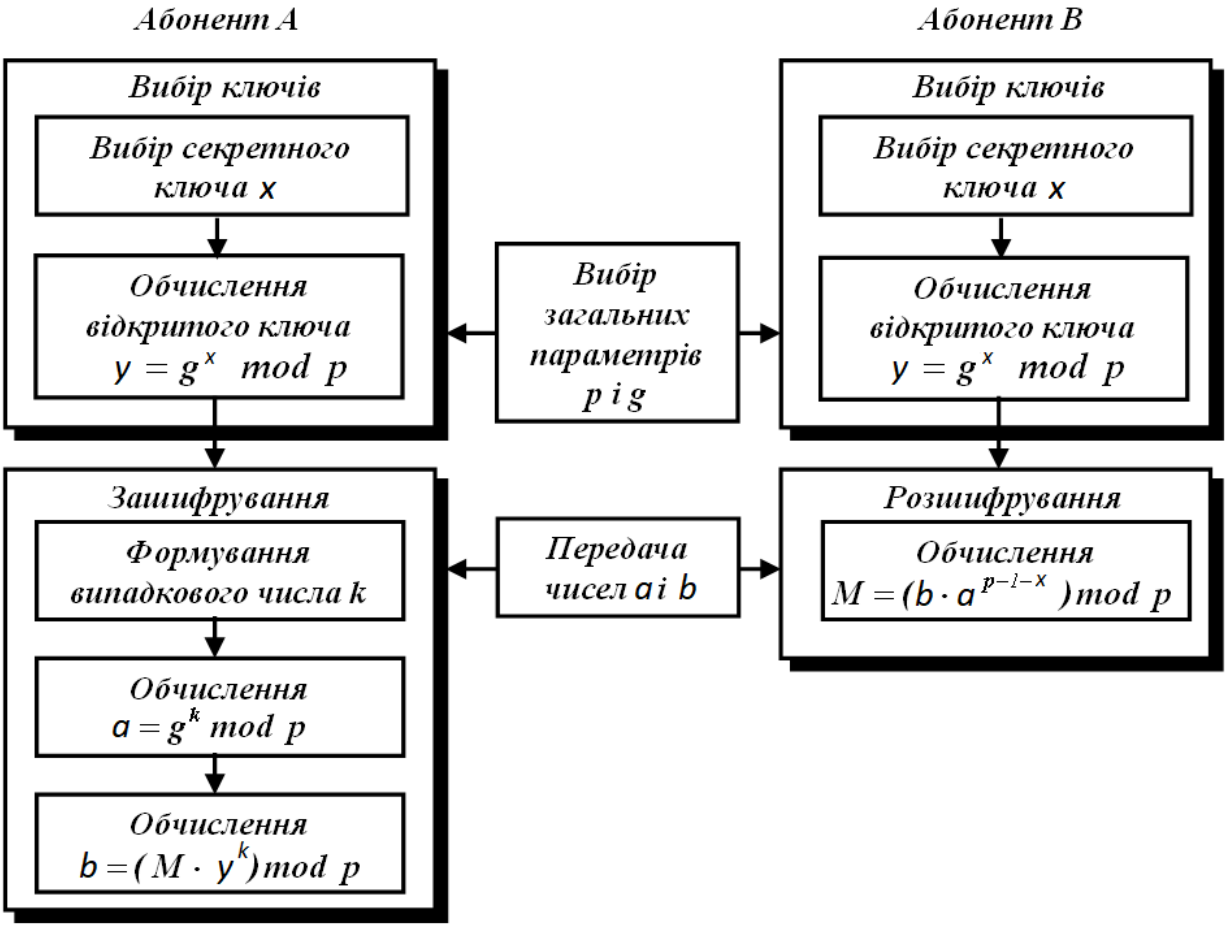

Рисунок 2.4 — Схема криптографічної системи Ель-Гамаля

2) зловмисник, знаючи p, g, y, a і b, не може обчислити М.

Доведення:

$M' = b \times a^{(p-1-x)}(mod\ p) = M \times y^k \times g^{k^{(p-1-x)}}(mod\ p) = M \times g^{xk} \times g^{k^{(p-1-x)}}(mod\ p) = M \times\ g^{xk+kp-k-kx})(mod\ p) = M \times\ g^{k(p-1)}(mod\ p)$

За теоремою Ферма:

$g^{k(p-1)}(mod\ p) = 1^k(mod\ p) = 1$, звідси отримуємо першу частину твердження.

Для доведення другої частини зауважимо, що зловмисник не може обчислити k у рівності, оскільки це завдання дискретного логарифмування. Отже, він не може обчислити M, оскільки M було помножено на невідоме йому число. Зловмисник також не може відтворити дії законного одержувача повідомлення (абонента B), оскільки йому невідоме секретне число b (обчислення b — також задача дискретного логарифмування) [13].

Розглянемо на прикладі шифр Ель-Гамаля. Нехай M=5. Візьмемо p=11, g=2. Нехай абонент B обрав для себе секретне число x=8 і обчислив значення $y = 2^8 (mod\ 11) = 3$.

Абонент A обирає випадкове k=9 і обчислює $a = 2^9 (mod\ 11) = 6$, $b = 5 \times 3^9 (mod\ 11) = 9$.

Абонент A надсилає абоненту B пару чисел (6,9). Абонент B знаходить

$$M' = 9 \times (6^8)^{-1} (mod\ 11) = 5 .$$

Обсяг зашифрованого повідомлення в 2 рази перевищує обсяг повідомлення, але вимагається лише одна передача даних [16].

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ

3.1 Вибір програмних засобів і опис програмної реалізації

Програму розроблено на мові програмування C#, в середовищі Microsoft Visual Studio на платформі OS Windows.

У Microsoft VS створюється Windows Forms Application (рис. 3.1).

У Windows Forms форма - це візуальна поверхня, на якій виводиться інформація для користувача. Додаток Windows Forms побудовано шляхом приміщення елементів управління на форму і написання коду для реагування на дії користувача, такі як клацання миші або натискання клавіш. Елемент управління - це окремий елемент призначеного для користувача інтерфейсу для відображення або введення даних.

Windows Forms включає широкий набір елементів управління, які можна додавати на форми: текстові поля, кнопки, списки, що розкриваються, перемикачі та навіть веб-сторінки [17].

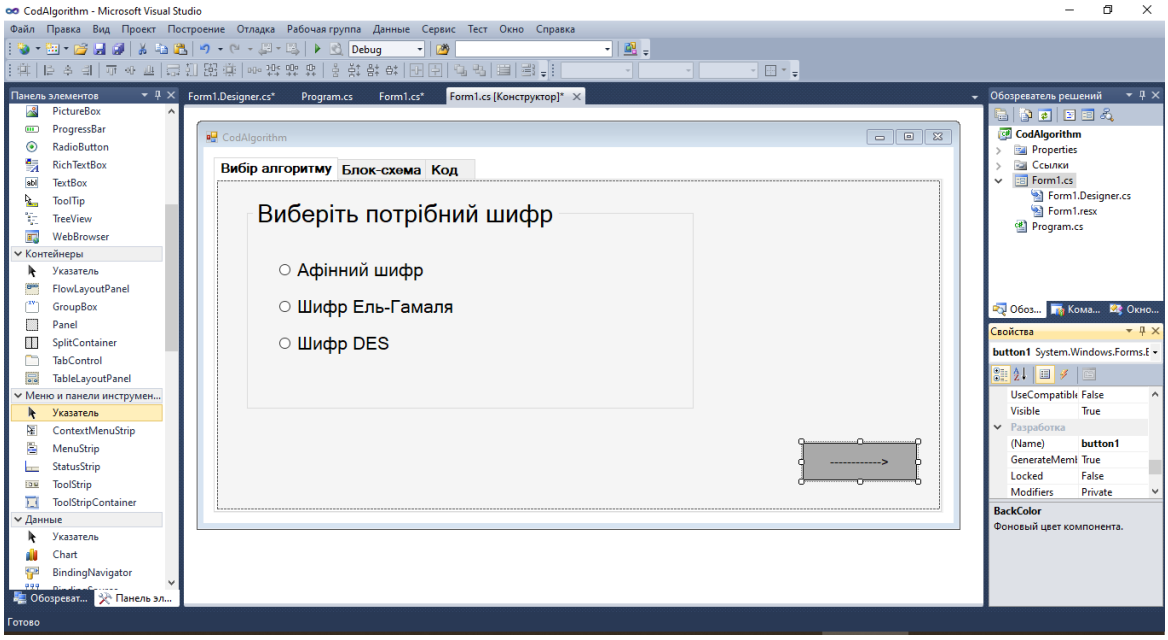

Рисунок 3.1 – Процес створення додатку

Елементи управління Windows Forms, що використані у даній програмі наведено у табл. 3.1.

Таблиця 3.1 — Елементи управління Windows Forms

TabControl (контейнер)	Дозволяє створити елемент управління з декількома вкладками. Кожна вкладка (TabPage)буде зберігати деякий набір інших елементів управління, як кнопки, текстові поля і ін. TabPage1 — “ Вибір алгоритму” TabPage2 —“Блок-схема” TabPage3 — “ Код”
GroupBox (контейнер)	Групує набір елементів управління (перемикачів) та має необов’язковий заголовок. GroupBox1 — “Виберіть потрібний шифр” GroupBox2 — “Умовні позначення – Афінний шифр Цезаря” GroupBox3 — “ Умовні позначення – шифр Ель-Гамаля ” GroupBox4 — “Умовні позначення – DES”
RadioButton (стандартний елемент управління)	Відображає кнопки-перемикачі. Створена група перемикачів поміщена в контейнер GroupBox1. Включення одного перемикача означає відключення всіх інших. RadioButton1 —“Афінний шифр” RadioButton2—“Шифр Ель-Гамаля” RadioButton3—“Шифр DES”
Button (стандартний елемент управління)	Кнопка, що запускає, зупиняє або перериває процес.
PictureBox (стандартний елемент управління)	Відображає графічні файли в кадрі. PictureBox1 — поле для будування блок-схеми на другій вкладці “Блок-схема”.
TextBox (стандартний елемент управління)	Поле для відображення тексту. textBoxHelp — поле-підказка для побудування одного з трьох алгоритмів.
RichTextBox (стандартний елемент управління)	Поле для відображення тексту. На відміну від TextBox може відображати шрифти, кольори і посилання. RichTextBox1 — поле для виведення рядків коду на третій вкладці “Код”.

StatusStrip	Відображає відомості про поточний стан програми за допомогою області в рамці, зазвичай в нижній частині батьківської форми.

Опис і функціональне призначення методів наведено у табл. 3.2.

Таблиця 3.2 — Опис методів

Ім'я	**Модифікатор доступу**	**Тип**	**Аргументи**	**Призначення**
Шифр DES				
StringToRightLength	private	string	string input	Метод, який доводить рядок до розміру, щоб він ділився на sizeOfBlock
CutStringIntoBlocks	private	void	string input	Метод, який розбиває рядок в звичайному (символьному) форматі на блоки.
CutBinaryStringIntoBlocks	private	void	string input	Метод, який розбиває рядок в двійковому форматі на блоки.
StringToBinaryFormat	private	string	string input	Метод, що переводить рядок в двійковий формат.
CorrectKeyWord	private	string	string input, int lengthKey	Метод, який доводить довжину ключа до потрібної довжини.
EncodeDES_One_Round	private	string	string input, string key	Один раунд шифрування алгоритмом DES.
XOR	private	string	string s1, string s2	XOR двох рядків з двійковими даними.
KeyToNextRound	private	string	string key	Обчислення ключа для наступного раунду шифрування DES.

StringFromBinaryToNormalFormat	private	string	string input	Метод, що переводить рядок з двійковими даними в символьний формат.
Button1Click	—	void	object sender, EventArgs e	Реалізація кнопки "Зашифрувати".
Шифр Ель-Гамаля				
controlP	—	bool	int pW	Перевірка на простоту числа p.
Button1Click	—	void	object sender, EventArgs e	Обробка кнопки «Перевірити», що p>g.
Button2Click	—	void	object sender, EventArgs e	Обробка кнопки «Призначити» величин p I g.
Button3Click	—	void	object sender, EventArgs e	Обробка кнопки «Задати y I k», які обчислюються та генеруються автоматично.
Button4Click	—	void	object sender, EventArgs e	Обчислення шифротекстів a I b.
Афінний шифр				
toCodeAffine	—	string	int a,int b,string startString	Реалізація шифру.
Button1Click	—	void	object sender, EventArgs e	Обробка кнопки «Зашифрувати».

Система візуального програмування розрахована на три різні криптоалгоритми (програмний код у додатку А) і запускається з основної програми, код якої наведено у додатку Б.

3.2 Результати роботи програми

На першій сторінці додатку користувачу запропоновано обрати один з трьох криптографічних алгоритмів (рис. 3.2).

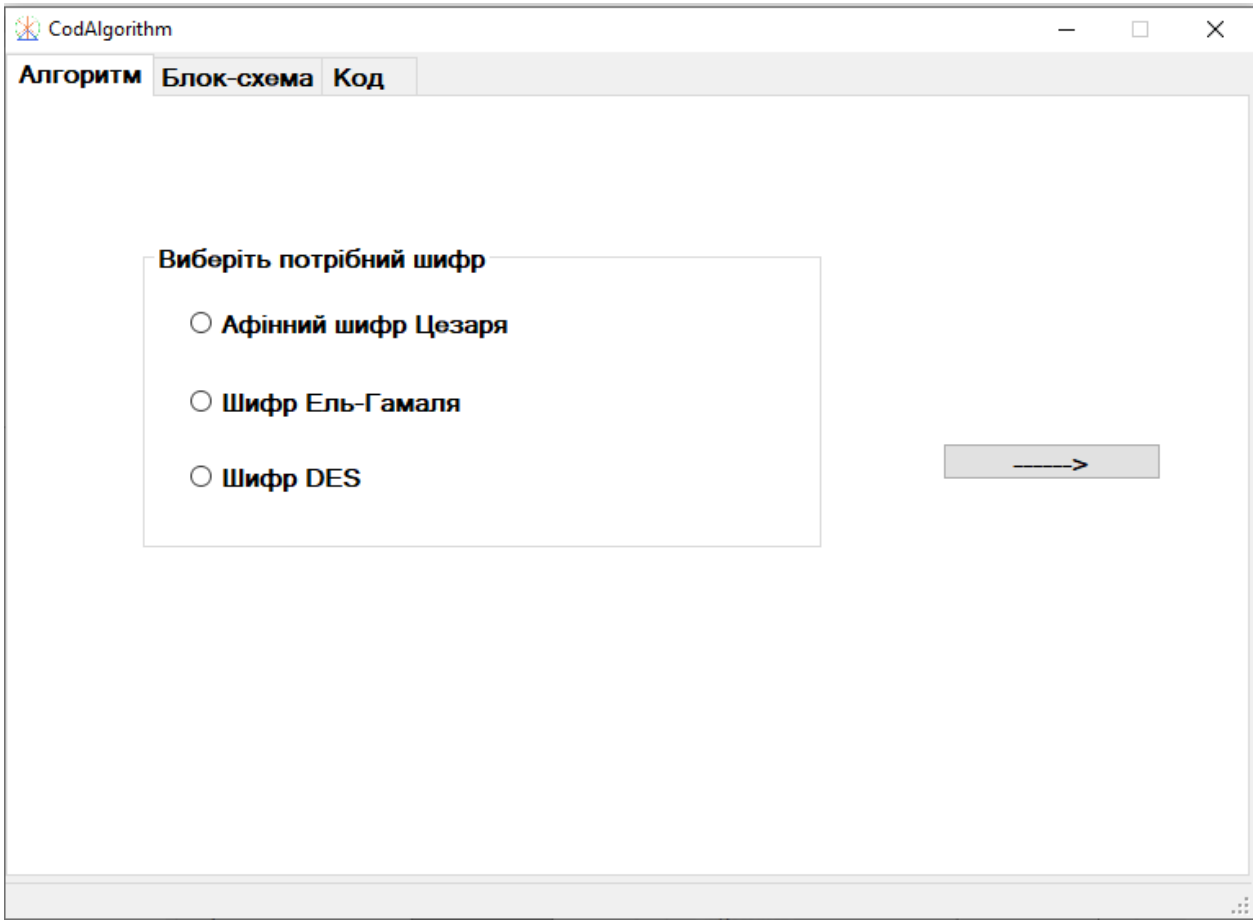

Рисунок 3.2 — Стартова вкладка системи

Після того, як алгоритм обрано, з’являється можливість переходу до наступного кроку (рис 3.3).

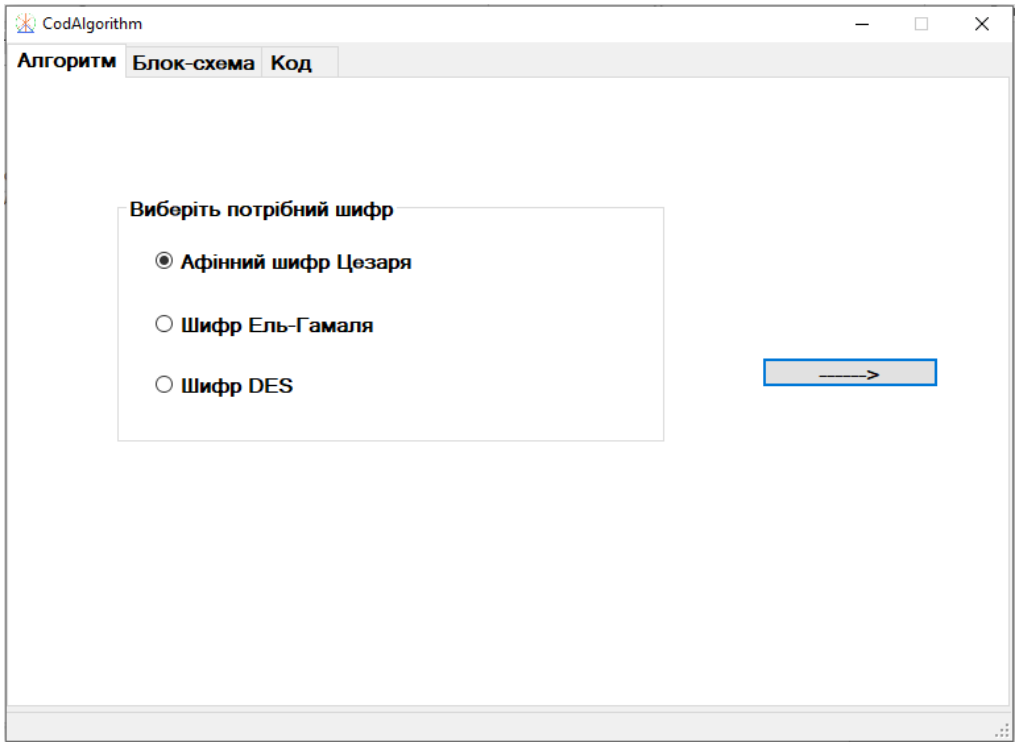

Рисунок 3.3 — Вигляд початкової сторінки з можливістю переходу до наступної дії

На наступній сторінці з’являються елементи схеми (рис. 3.4, рис. 3.9, рис. 3.13), які повинні бути розташовані у відповідності до необхідного алгоритму шифрування (рис. 3.5, рис. 3.10, рис. 3.14), щоб з’явилася можливість побачити програмний код криптоалгоритму (рис. 3.7 – 3.8, рис. 3.11 – 3.12, рис. 3.15 – 3.16).

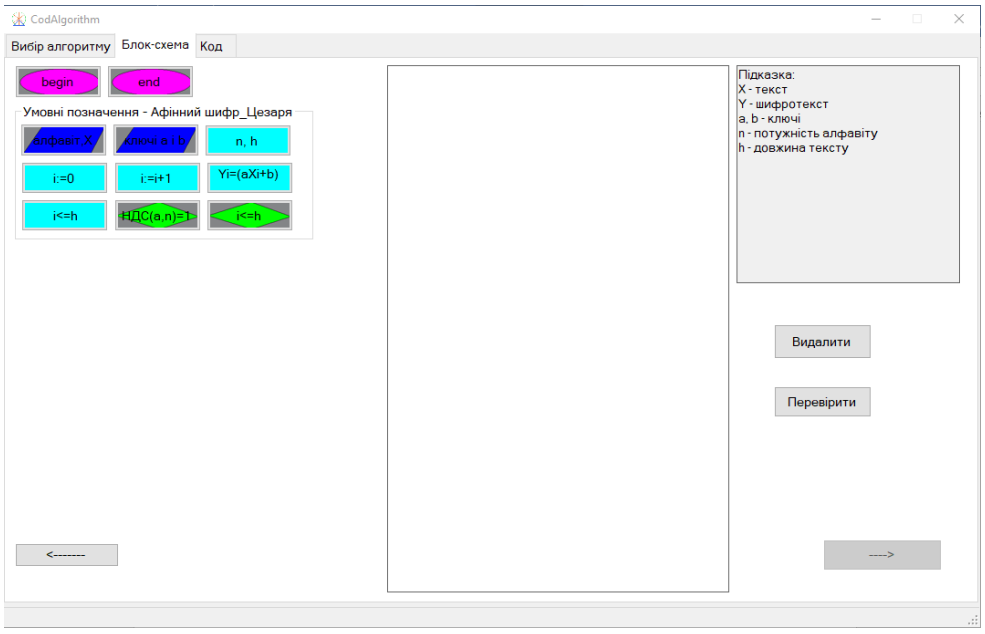

Рисунок 3.4 — Поле для побудови схеми афінного шифру

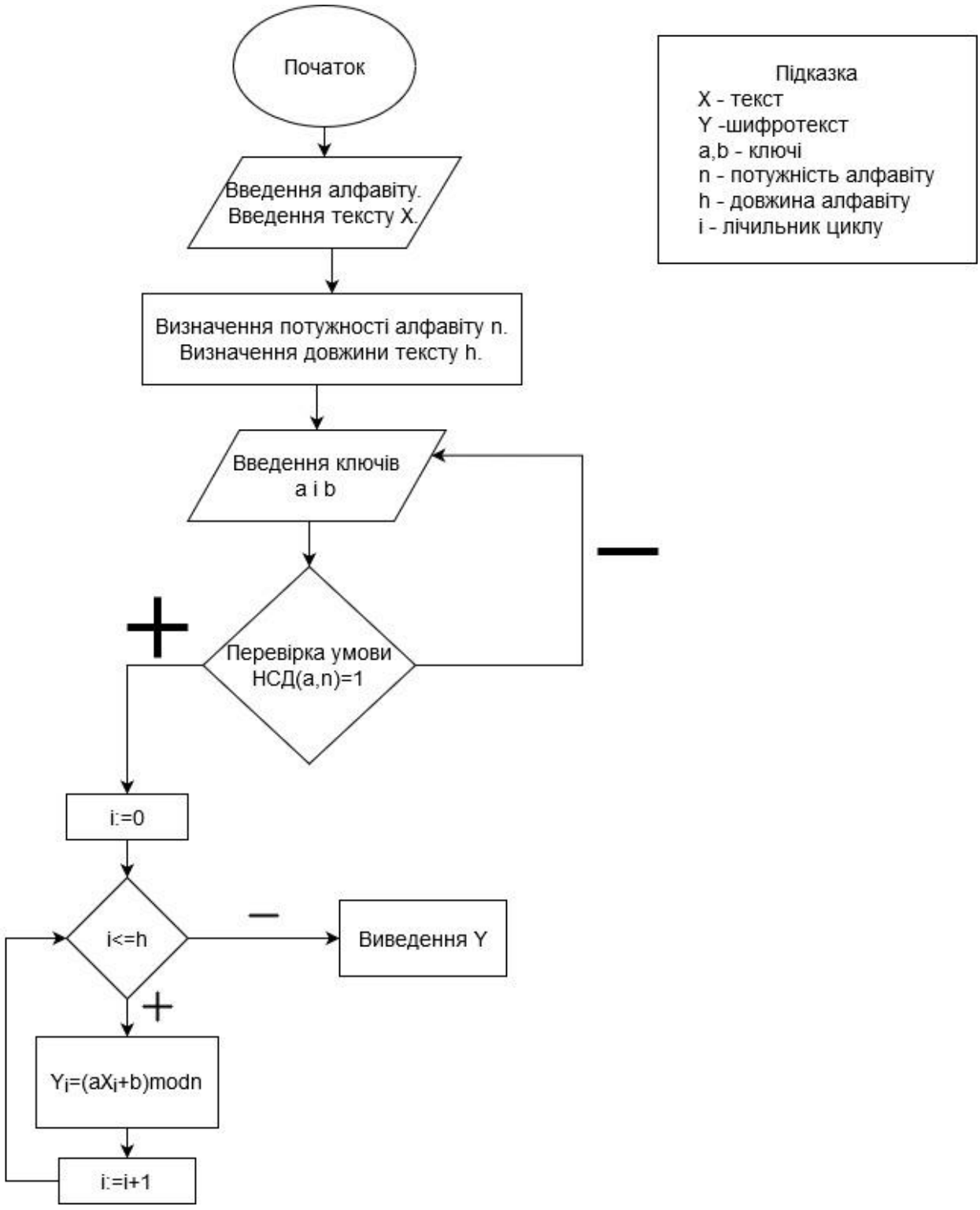

Рисунок 3.5 — Схема афінного шифру

Якщо схема побудована неправильно, то з’являється відповідне повідомлення (рис. 3.6).

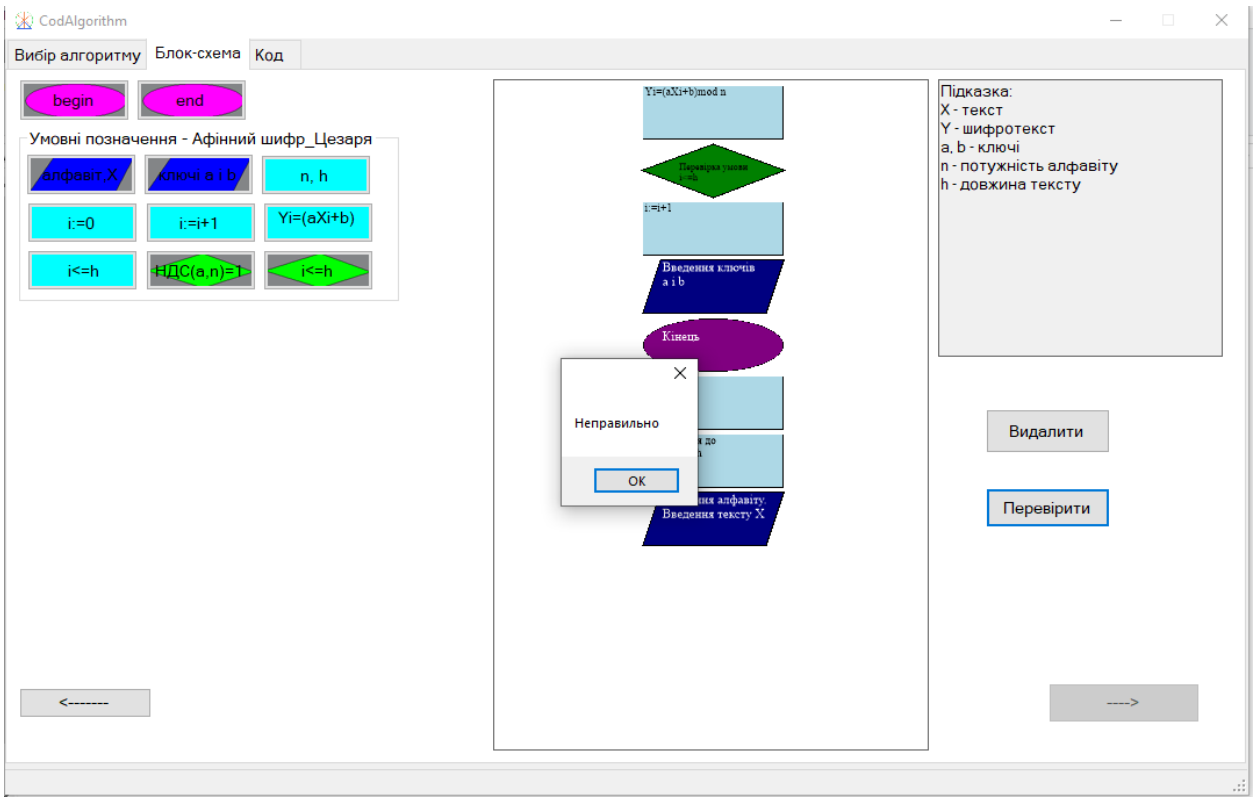

Рисунок 3.6 — Повідомлення про неправильність схеми

Якщо схема побудована правильно, то з’являється можливість перейти на сторінку, де показана частина коду, що містить основну ідею шифру.

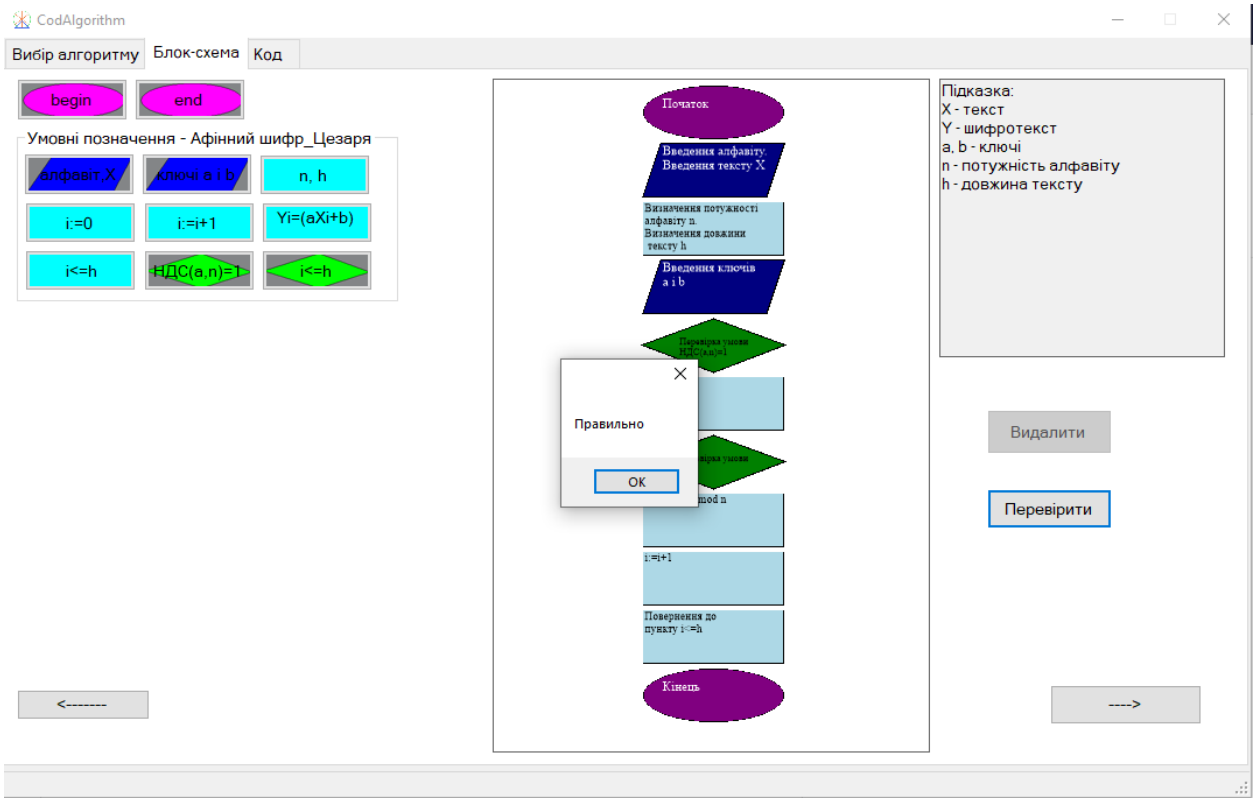

Рис. 3.7 — Повідомлення про правильно побудовану схему афінного шифру

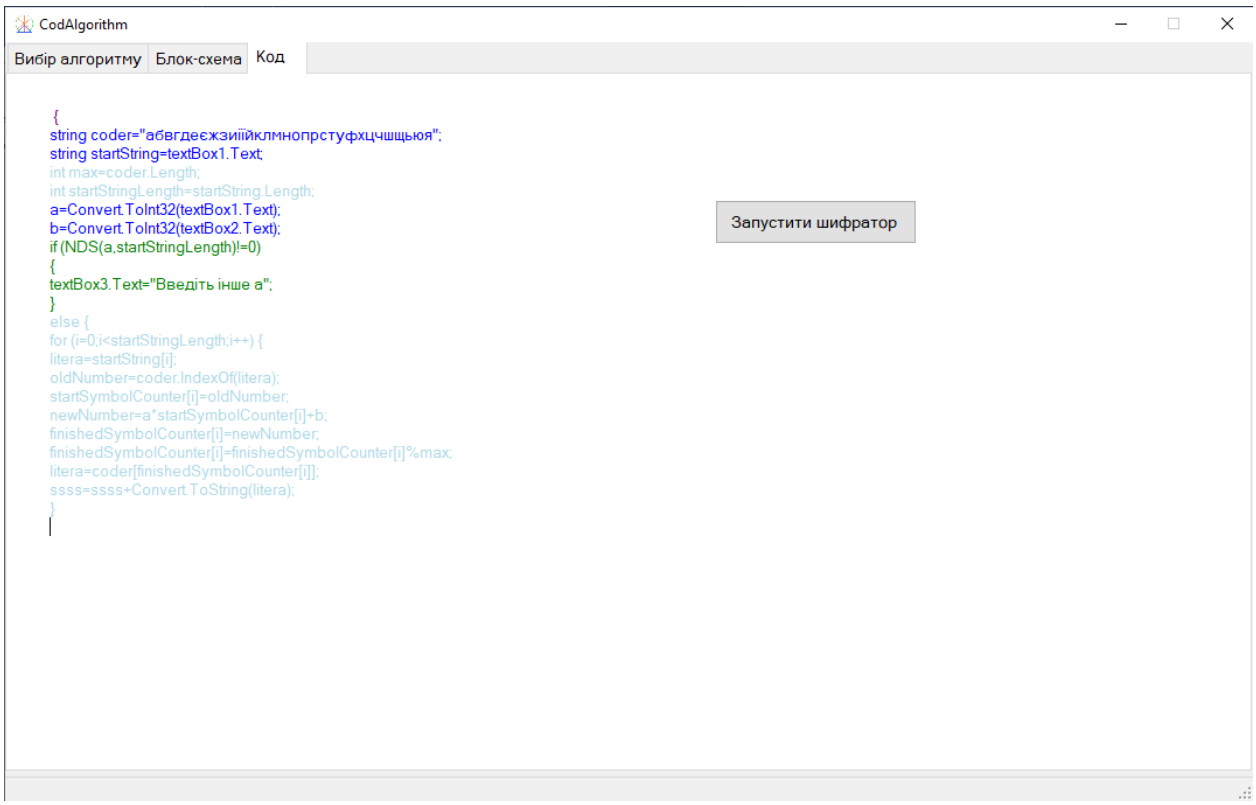

Рисунок 3.8 – Частина коду афінного шифру

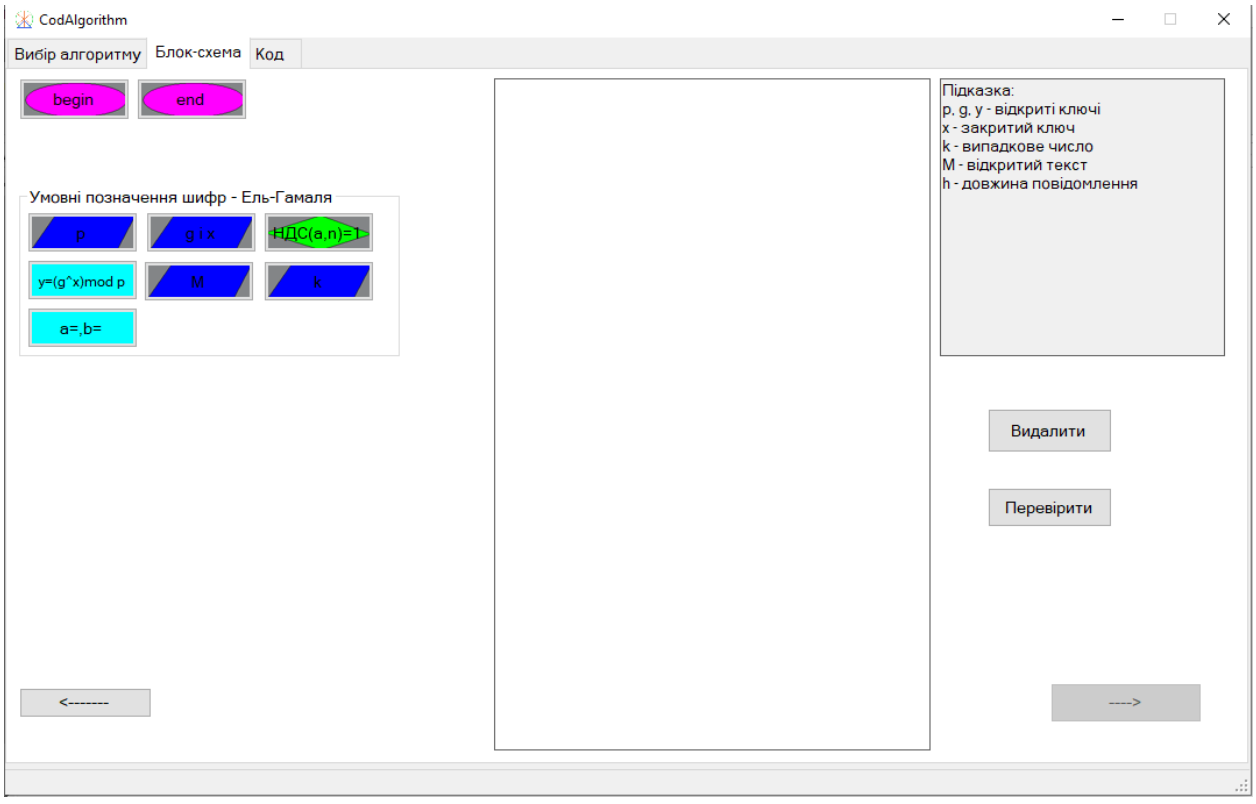

Рисунок 3.9 — Поле для введення схеми шифру Ель-Гамаля

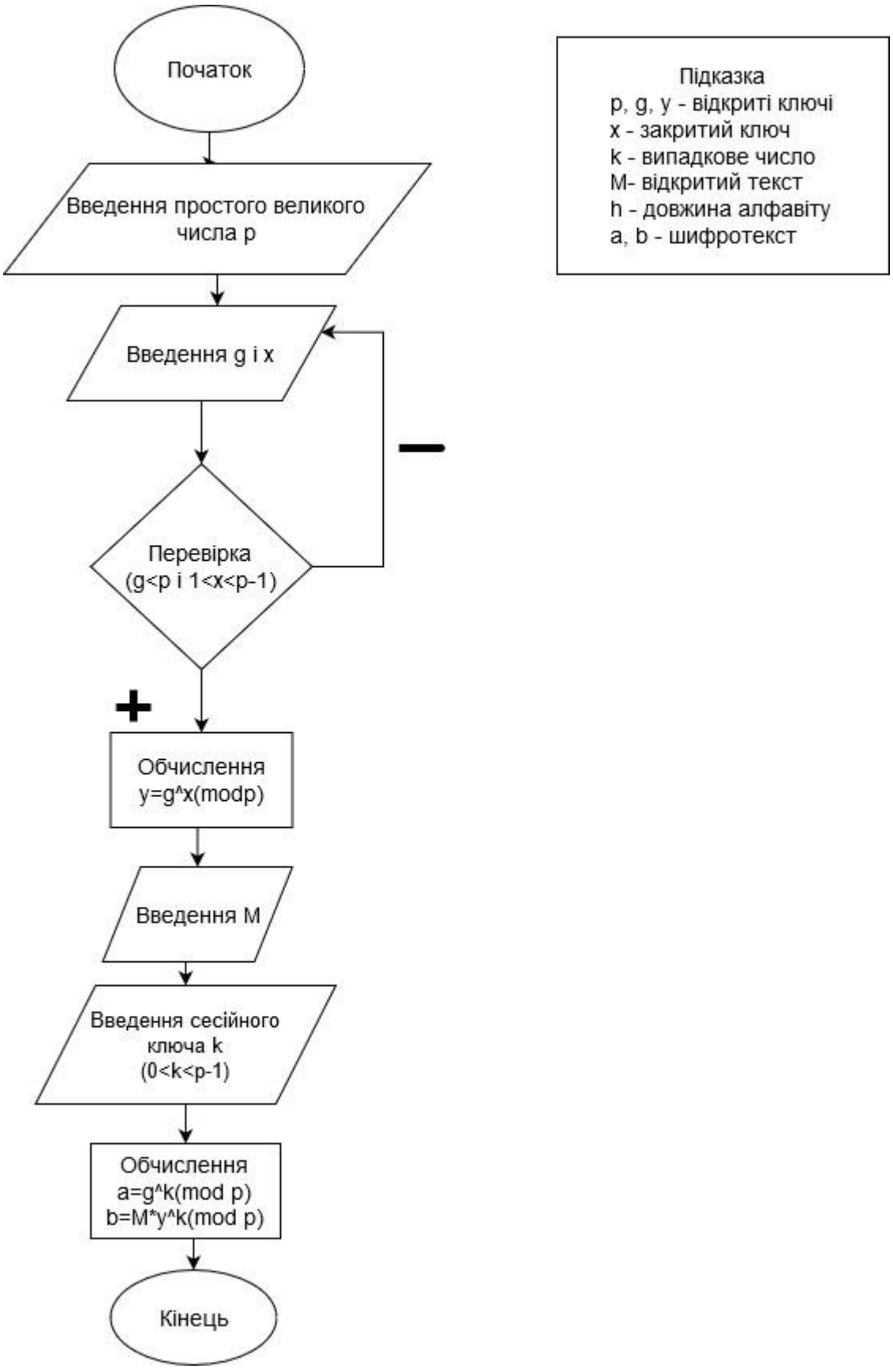

Рисунок 3.10 — Схема шифру Ель-Гамаля

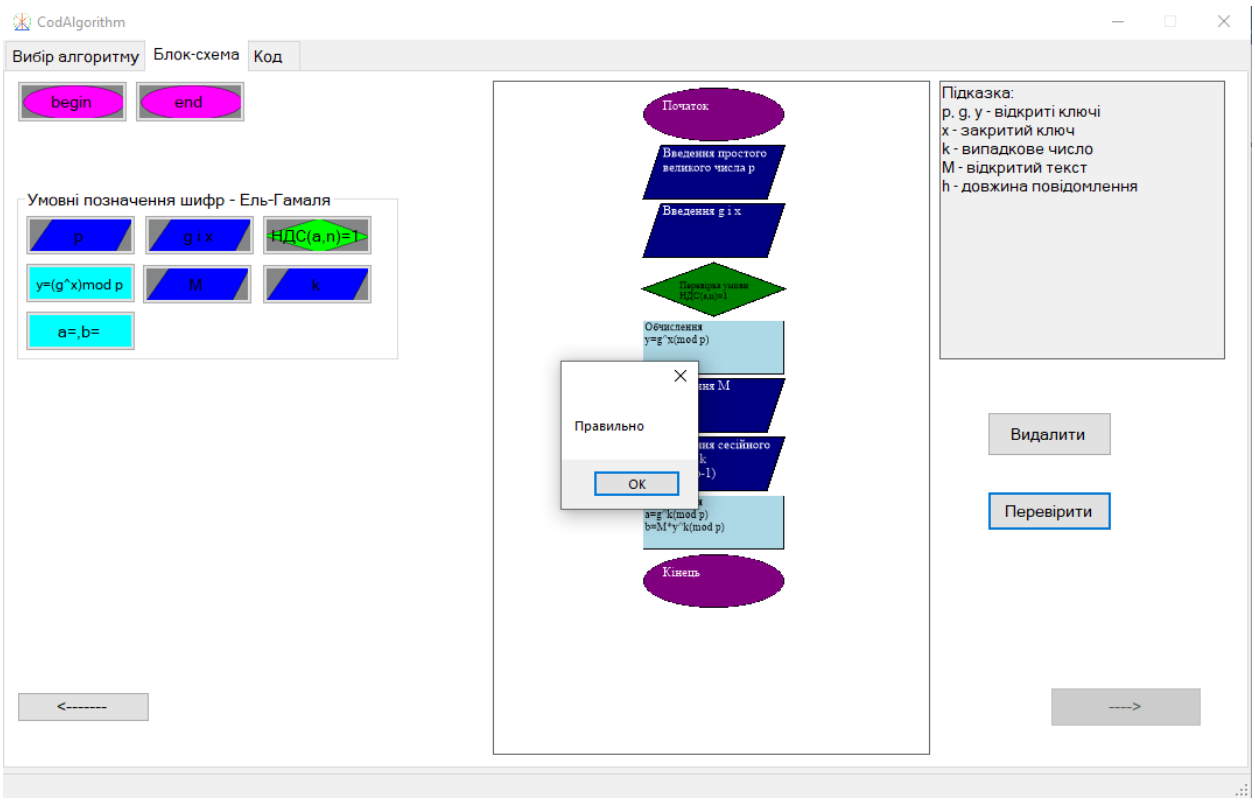

Рисунок 3.11 — Повідомлення про правильно побудовану схему шифру Ель-Гамаля

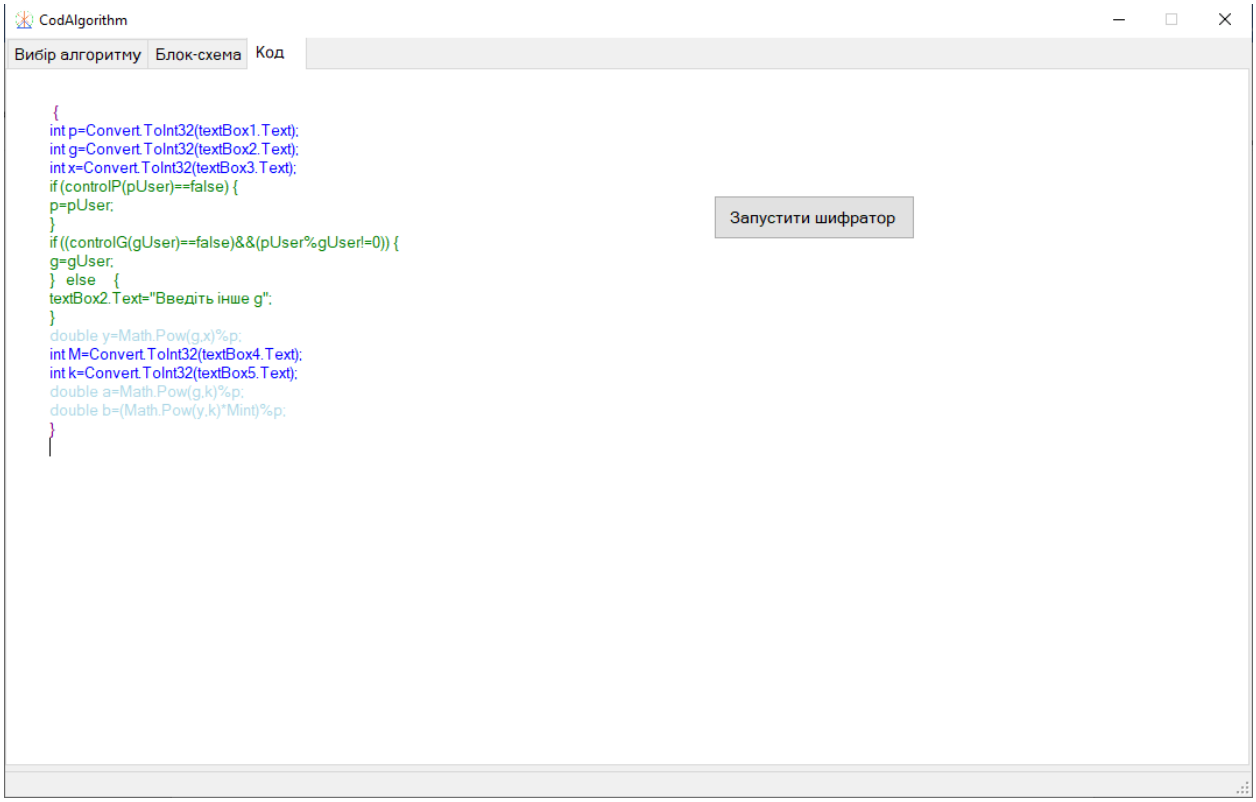

Рисунок 3.12 — Частина коду шифру Ель-Гамаля

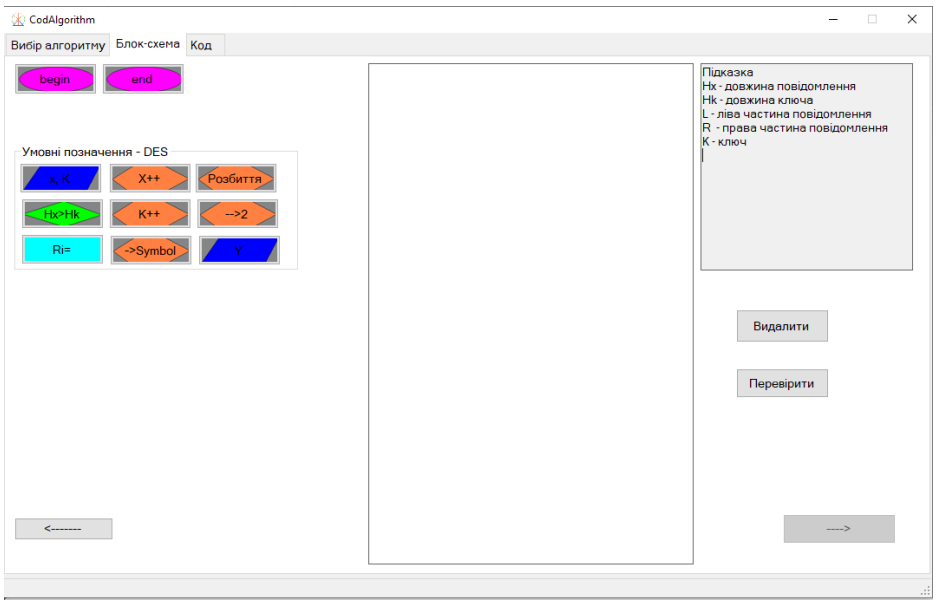

Рисунок 3.13 — Поле для введення схеми шифру DES

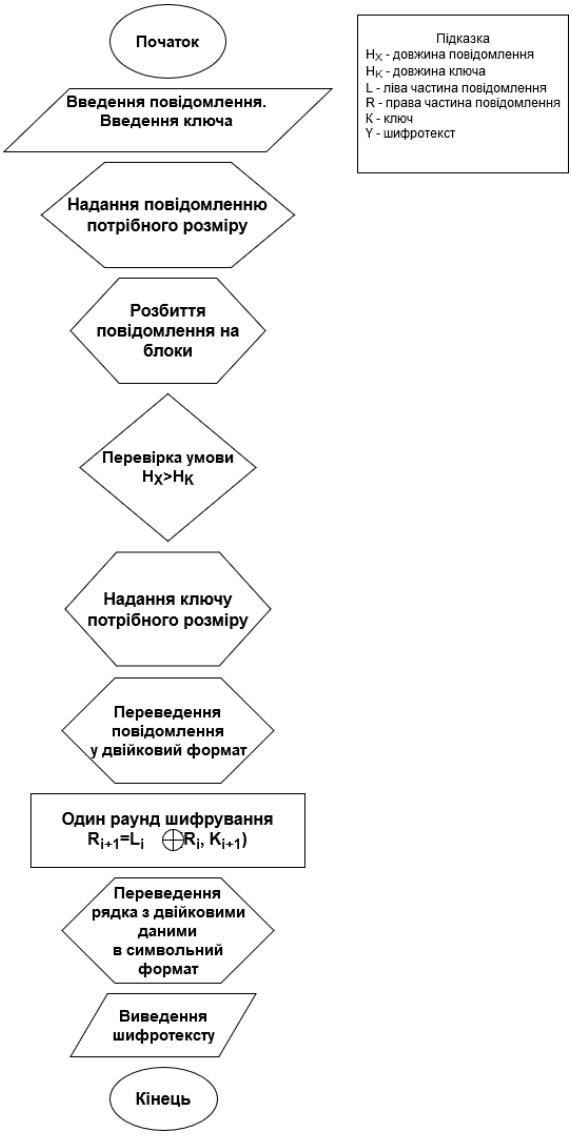

Рисунок 3.14 — Схема шифру DES

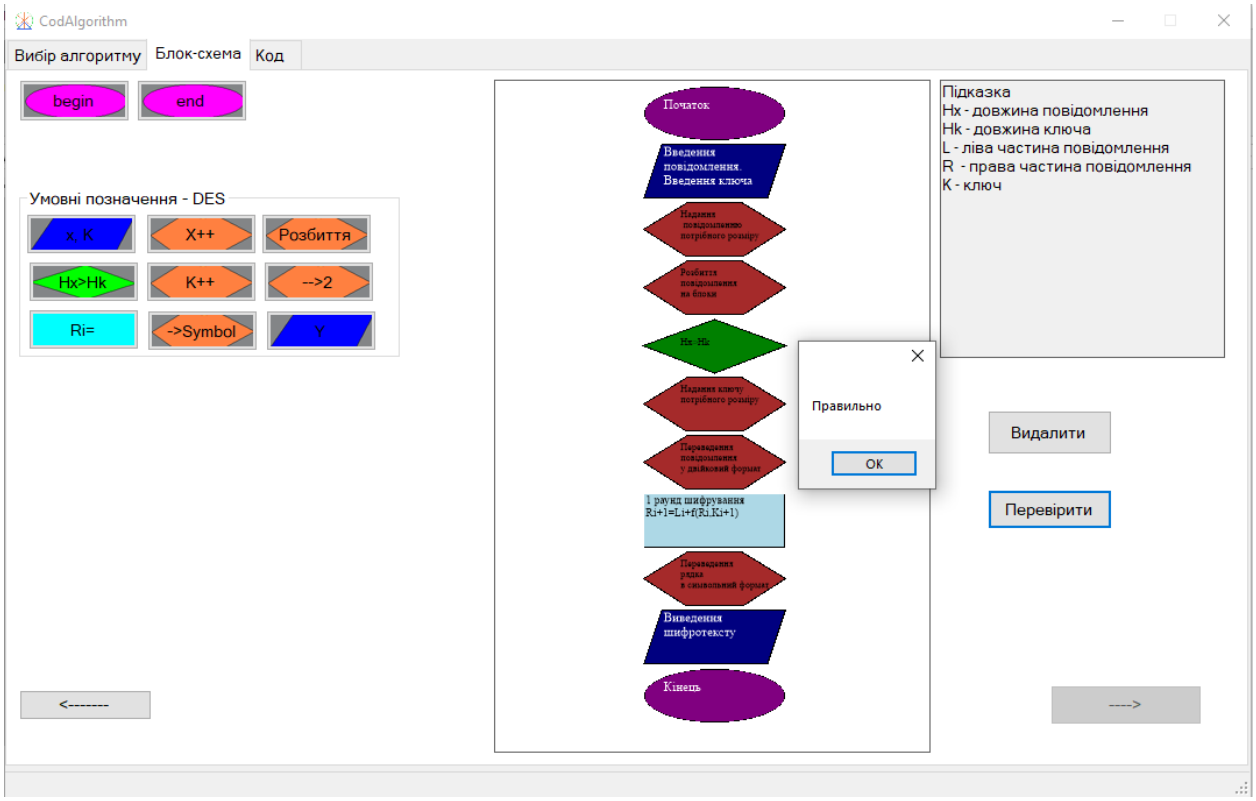

Рисунок 3.15 — Повідомлення про правильно побудовану схему шифру DES

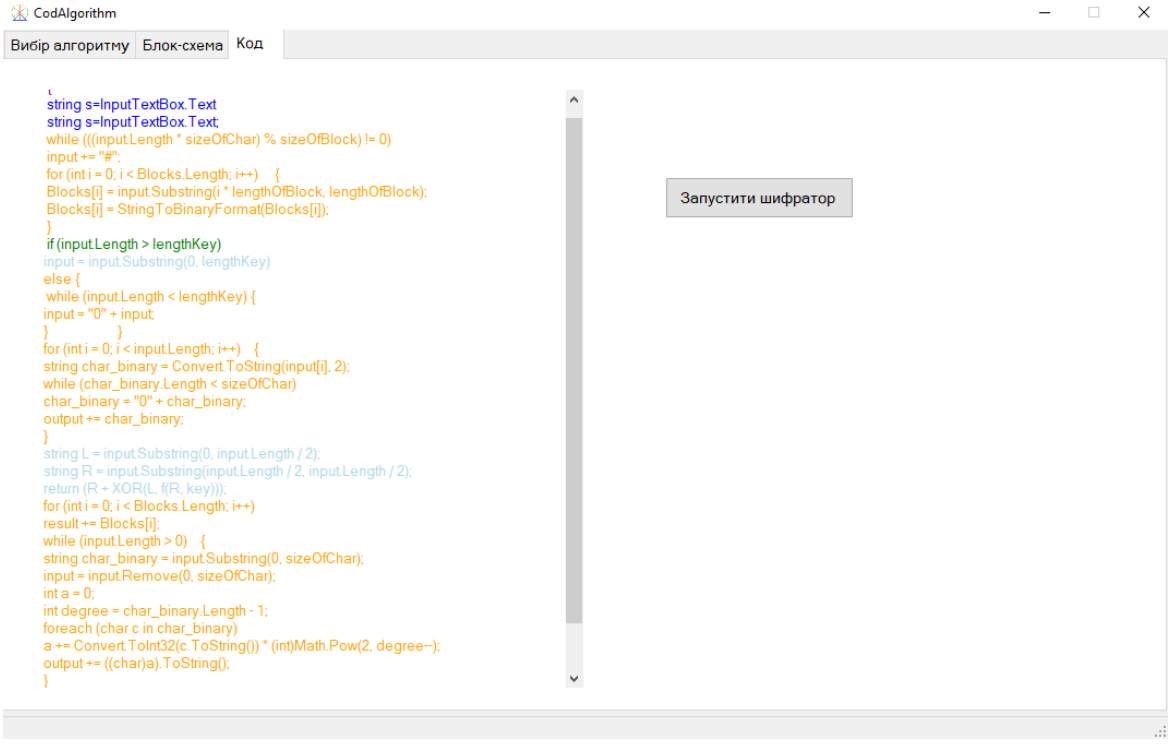

Рисунок 3.16 — Частина коду шифру DES

Після того, як програмний код було проаналізовано, можемо зашифровувати дані (рис. 3.17 - рис. 3.24).

Афінний шифр

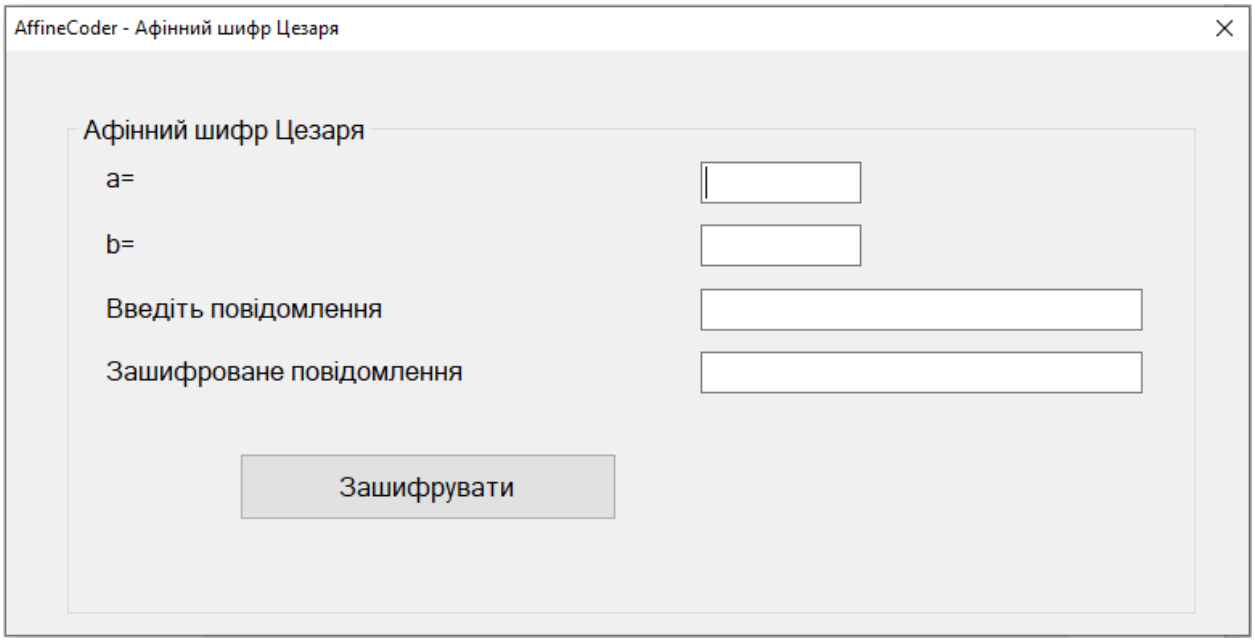

Рисунок 3.17 — Афінний шифр до введення даних

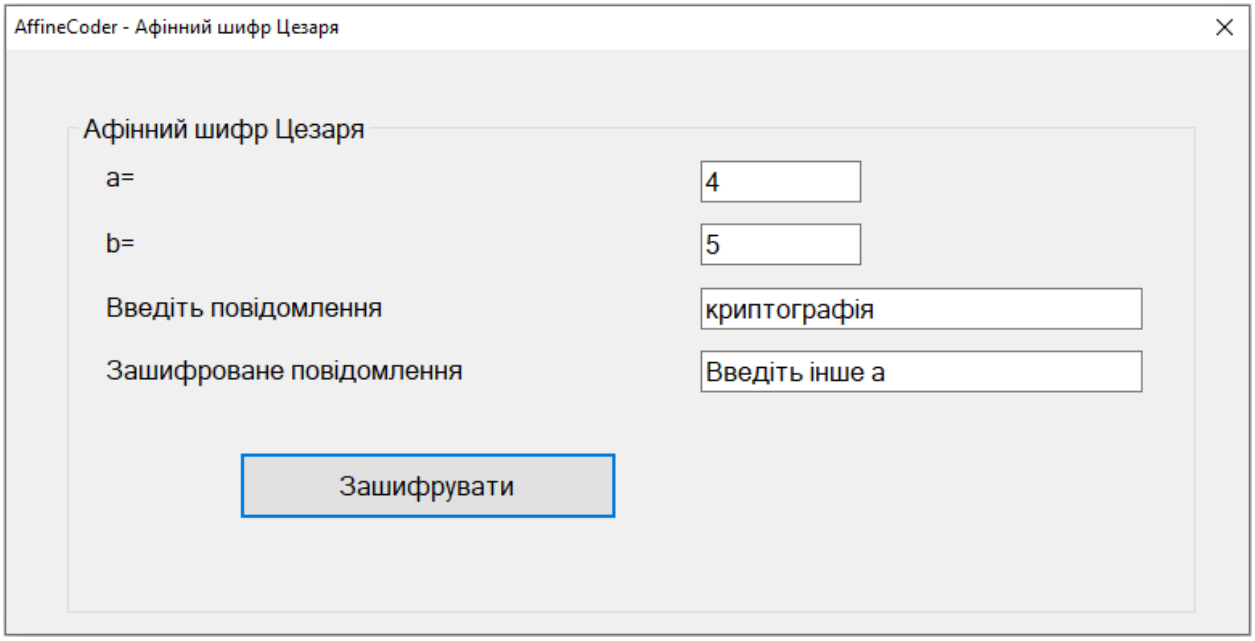

Рисунок 3.18 — Попередження про неправильні дані при шифруванні афінним шифром

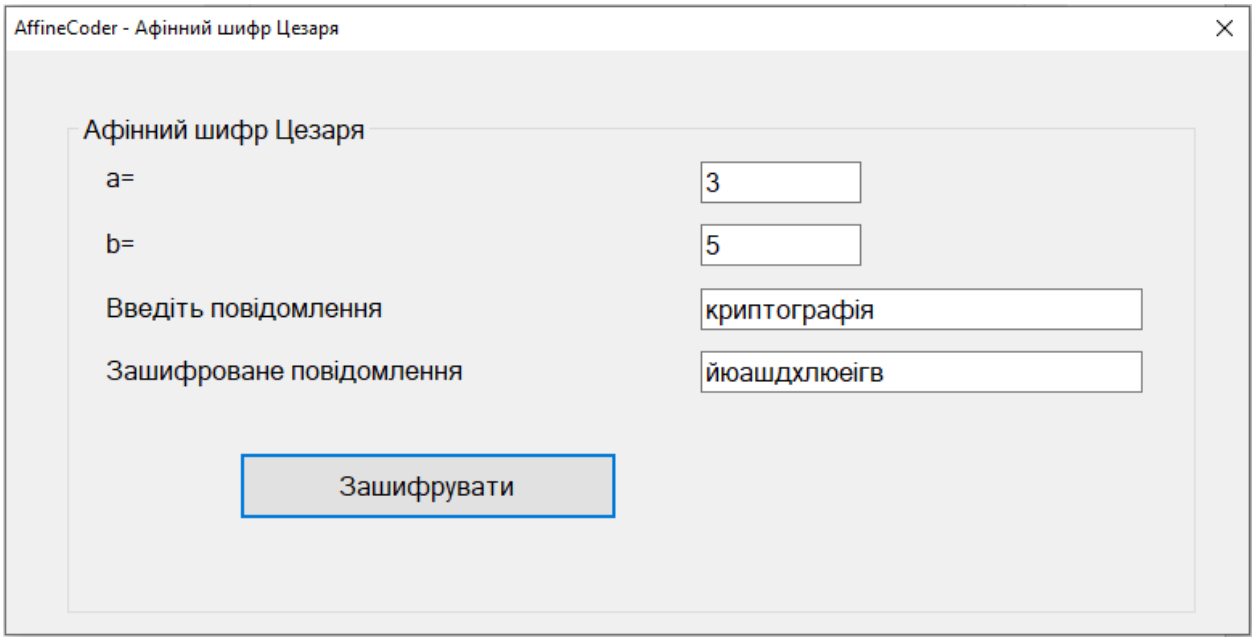

Рисунок 3.19 — Дані після шифрування афінним шифром

Шифр Ель-Гамаля

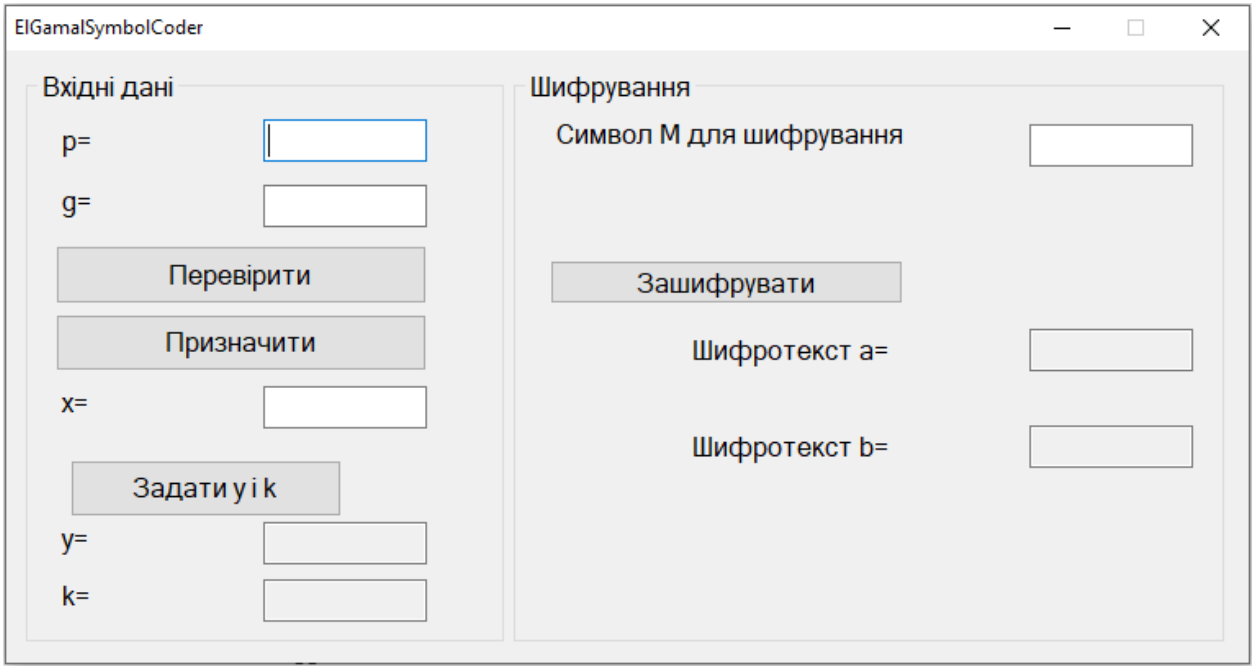

Рисунок 3.20 — Шифр Ель-Гамаля до введення даних

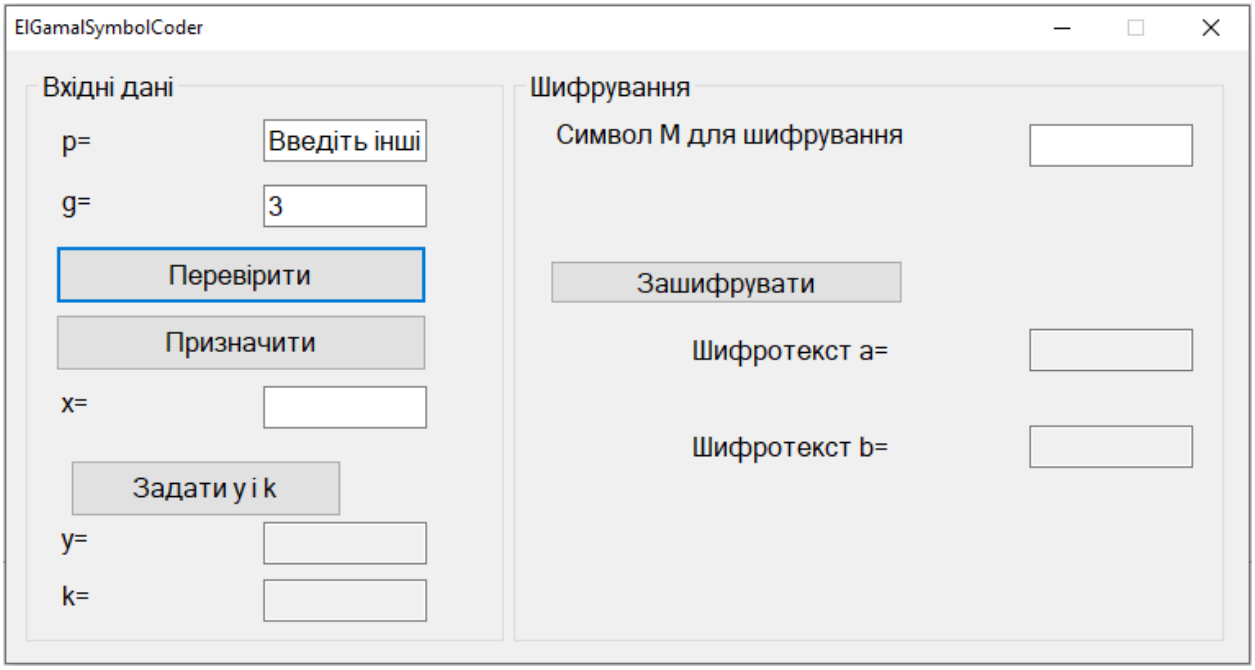

Рисунок 3.21 — Попередження про неправильні дані при шифруванні Ель-Гамаля

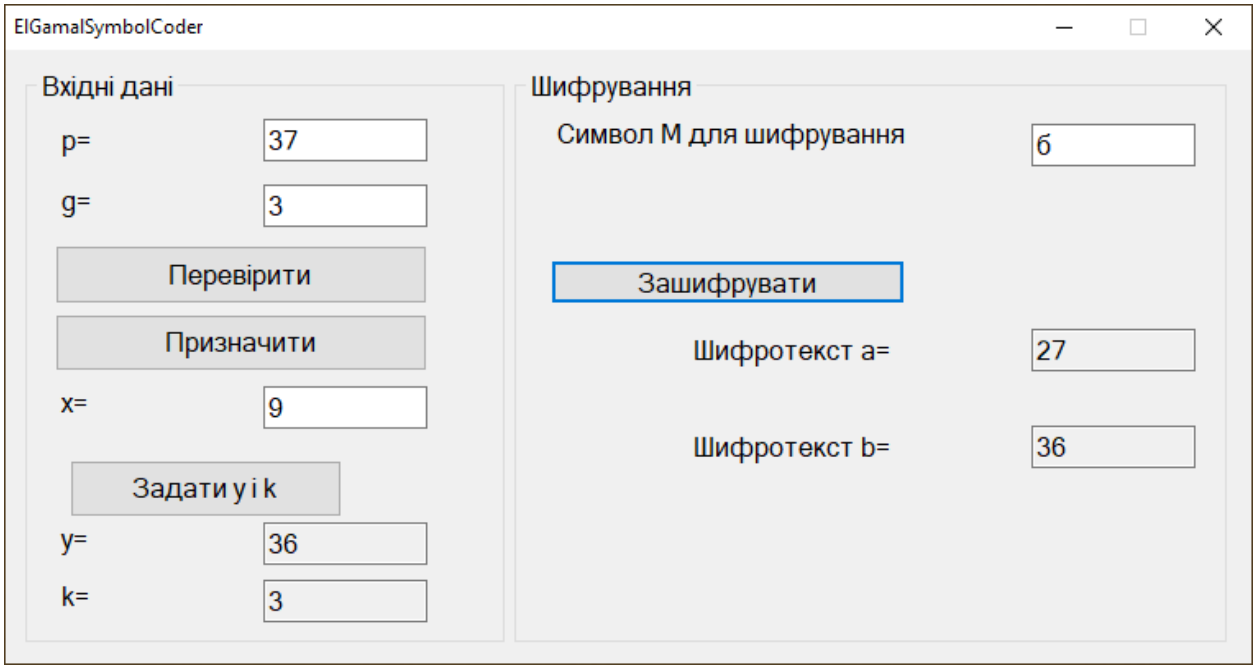

Рисунок 3.22 — Дані після шифрування Ель-Гамаля

Шифр DES

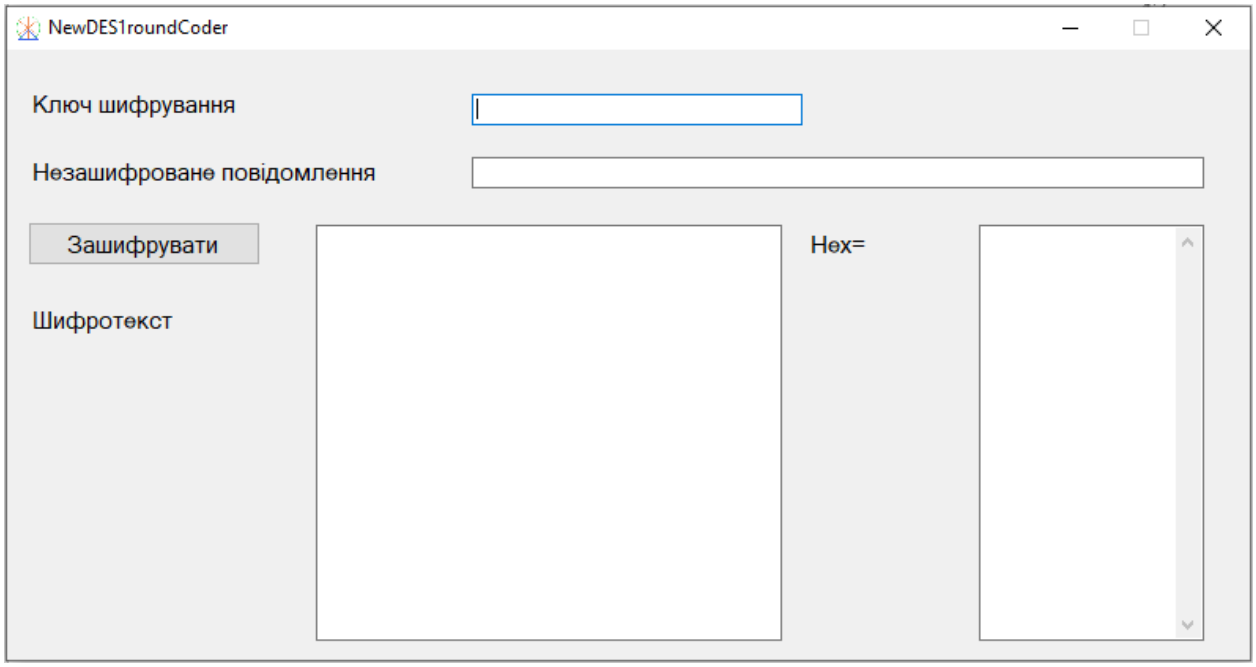

Рисунок 3.23 — Шифр DES до введення даних

Зашифрований текст показано як в двійковій системі числення, так і в шістнадцятковій.

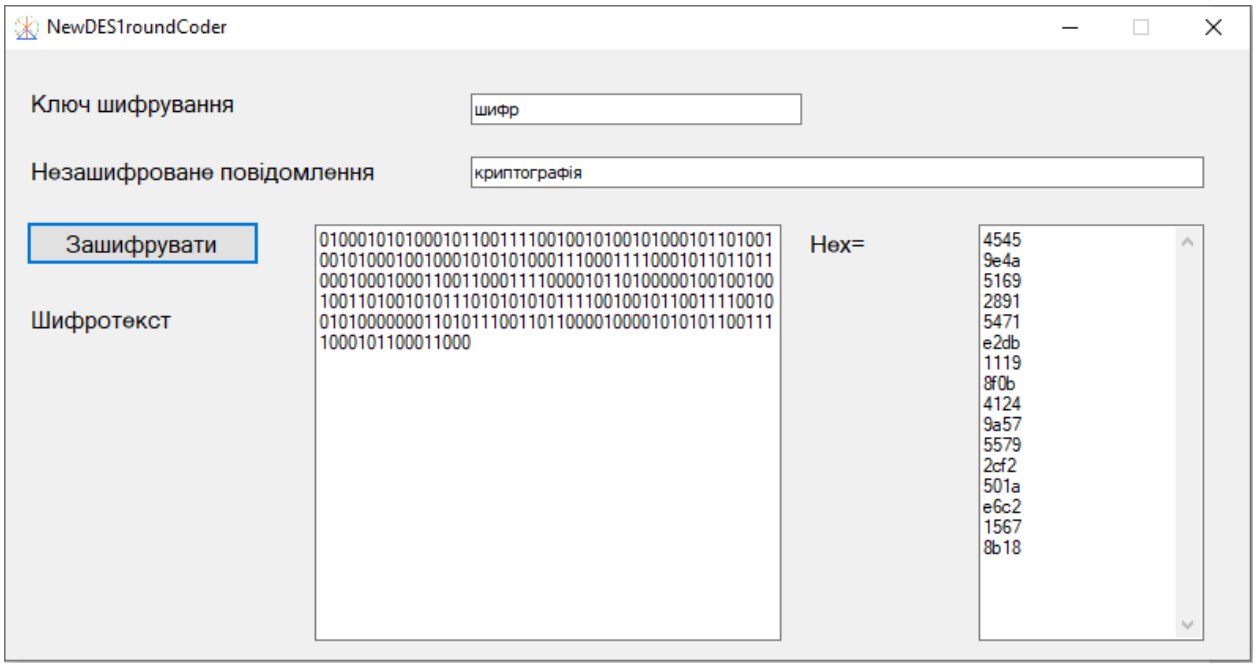

Рисунок 3.24 — Дані після одного раунду шифрування DES

ВИСНОВКИ

Дана робота присвячена проблемі розуміння коду криптографічних шифрів за допомогою принципу візуалізації, наочності та алгоритмізації. Іноді програмісту, що працює з програмою, буває корисно побачити її у вигляді алгоритму, щоб більш наочно зрозуміти її логіку.

Розв’язання даної проблеми можливе із застосуванням візуального програмування. Проведений літературний огляд та інформаційний аналіз існуючих програмних рішень, дозволив констатувати наявність нагальної потреби створенні такого продукту, що буде відображати програмний код на основі створеної візуальної блок-схеми для криптографічного алгоритму.

Було розроблено додаток під назвою “CodAlgorithm”, що надає змогу переглянути програмний код “Афінного шифру”, “Шифру Ель-Гамаля” та “Шифру DES” після того, як буде побудовано їхні блок-схеми, що перевіряються програмою на правильність. Від користувача потрібно не знання мов програмування, а поняття про алгоритм, тому для кращого розуміння вихідного коду його рядки показані різними кольорами, що відповідають елементам схеми. Додаток, що дозволяє шифрувати повідомлення трьома різними криптоалгоритмами, призначения акцентувати увагу на самому алгоритмі, а не на синтаксисі конкретної мови програмування.

“CodAlgorithm” може бути використаний як самостійний продукт, а також може бути інтегрований в інші інформаційні системи.

СПИСОК ЛІТЕРАТУРИ

1. Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах: Постанова Кабінету Міністрів України від 29 березня 2006 р. №373. – Режим доступу: https://zakon.rada.gov.ua/laws/show/373-2006-%D0%BF

2. Algorithm & Flowchart Manual [Електронний ресурс] – Режим доступу: http://www.yspuniversity.ac.in/cic/algorithm-manual.pdf.

3. Дробушевич Л.Ф. Способы визуализации алгоритмов и программ / Л.Ф.Дробушевич, В.В. Конах [Електронний ресурс] // Материалы международного научного конгресса “Информационные системы и технологии”. Минск: БГУ, 2011. – С. 345-351. – Режим доступу: http://elib.bsu.by/bitstream/123456789/10196/1/Konakh64.pdf.

4. Коварцев А.Н. Методы и технологии визуального программирования: Учебное пособие / А.Н. Коварцев, В.В. Жидченко, Д.А. Попова-Коварцева – Самара: ООО «Офорт», 2017. – 197с.

5. LabVIEW [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/LabVIEW.

6. LabVIEW — первое знакомство [Електронний ресурс] – Режим доступу: https://habr.com/ru/post/57859/.

7. Visustin — генератор блок-схем [Електронний ресурс] – Режим доступу: https://www.aivosto.com/visustin-ru.html.

8. Crystal FLOW for C++ [Електронний ресурс] – Режим доступу: https://crystal-flow-for-c-plus-plus.soft112.com/.

9. AthTek Flowchart to Code [Електронний ресурс] – Режим доступу: http://www.athtek.com/flowchart-to-code.html.

10. AthTek Flowchart to Code [Електронний ресурс] – Режим доступу: https://athtek-flowchart-to-code.soft112.com/.

11. Flowgorithm [Електронний ресурс]. – Режим доступу: http://www.flowgorithm.org/.

12. Класифікація криптоалгоритмів [Електронний ресурс] – Режим доступу: https://wiki.tntu.edu.ua/Класифікація_криптоалгоритмів.

13. Корченко О. Г. Прикладна криптологія: системи шифрування: підручник / О. Г. Корченко, В. П. Сіденко, Ю. О. Дрейс. – К.: ДУТ, 2014. – 448 с.

14. Тарнавський Ю.А. Технології захисту інформації: методичні вказівки до виконання лабораторних робіт для студентів напряму підготовки 6.050101 «Комп'ютерні науки» програм професійного спрямування «Інформаційні технології проектування» та «Комп'ютерний еколого-економічний моніторинг» / Ю.А. Тарнавський – К.: НТУУ «КПІ», 2014. – 31 с. – Режим доступу: http://ela.kpi.ua/handle/123456789/7434.

15. Лисенко І.А. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни "Основи криптографічного захисту інформації" / І.А.Лисенко – Кропивницький: ЦНТУ, 2018. – 64с. – Режим доступу: http://dspace.kntu.kr.ua/jspui/bitstream/123456789/8548/1/Osn_kript_zax_inf_lab.pdf.

16. Схема Эль-Гамаля [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Схема_Эль-Гамаля.

17. Обзор Windows Forms [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/framework/winforms/windows-forms-overview.

ДОДАТОК А

Афінний шифр:

```
string toCodeAffine(int a,int b,string startString) {
int [] startSymbolCounter=new Int32[255];
int [] finishedSymbolCounter=new Int32[255];
string coder="абвгдеєжзиіїйклмнопрстуфхцчшщьюя";
int max=coder.Length;//max=32
int i,oldNumber=0,newNumber=0;
char litera;
int startStringLength=startString.Length;
string finishString;
string ssss="";
if (((max%a)!=0)&&((a%2)!=0)) {
for (i=0;i<startStringLength;i++) {
    litera=startString[i];
    oldNumber=coder.IndexOf(litera);
    startSymbolCounter[i]=oldNumber;
}
for (i=0;i<startStringLength;i++) {
    newNumber=a*startSymbolCounter[i]+b;
    finishedSymbolCounter[i]=newNumber;
}
for (i=0;i<startStringLength;i++) {
    finishedSymbolCounter[i]=finishedSymbolCounter[i]%max;
    }
for (i=0;i<startStringLength;i++) {
    litera=coder[finishedSymbolCounter[i]];
    ssss=ssss+Convert.ToString(litera);
}
 finishString=ssss;
return(finishString);              }
else {
    ssss="Введість інше a";
    return(ssss);
}        }
bool verifyA(int aaa,int mmm) {
    bool rez=false;
```

```
    if (aaa>1) {
        for (int j=2;j<aaa;j++) {
            if ((aaa%j==0)&&(mmm%j==0)) {
                rez=true;
            }   }   }
    return rez;       }
void MainFormLoad(object sender, EventArgs e)
{   }
int aa,bb;
int m=32;
string startMessage,codedMessage;
string sss;
void Button1Click(object sender, EventArgs e)
{
    if (textBox1.Text!=""){
    sss=textBox1.Text;
    aa=Convert.ToInt32(sss);
    }
    if (textBox2.Text!=""){
    sss=textBox2.Text;
    bb=Convert.ToInt32(sss);
    }
    if (textBox3.Text!=""){
    startMessage=textBox3.Text;
    }
    if (verifyA(aa,m)==false) {

    codedMessage=toCodeAffine(aa,bb,startMessage);
    textBox4.Text=codedMessage;
    }
    else {
        textBox4.Text="Введіть інше a";
    }   }   }   }
```

Шифр Ель-Гамаля:

```
int p,g;
int pUser,gUser;
string s1,s2,s3;
```

```
bool validation1=false;
int x;
double y;
int k=0;
string coder="абвгдеєжзиіїйклмнопрстуфхцчшщьюя";
char Mchar;
int Mint;
        bool controlP(int pW) {
    bool rez=false;
    for (int i=2;i<pW;i++) {
        if ((pW%i)==0) {
            rez=true;
        }         }
    return (rez);
}
        bool controlG(int gW) {
    bool rez=false;
    for (int i=2;i<gW;i++) {
        if (gW%i==0) {
           rez=true;
           }      }
    return (rez);
}
        void MainFormLoad(object sender, EventArgs e)
{    }
        void Button1Click(object sender, EventArgs e)
{
    s1=textBox1.Text;
    pUser=Convert.ToInt32(s1);
    s2=textBox2.Text;
    gUser=Convert.ToInt32(s2);
    if ((pUser>gUser)&&(pUser>36)) {
    if (controlP(pUser)==false) {
        p=pUser;
    }
    else {
        textBox1.Text="Введіть інше p";
```

```
    }
    if ((controlG(gUser)==false)&&(pUser%gUser!=0)) {
        g=gUser;
    }
    else {
        textBox2.Text="Введіть інше g";
    }
        validation1=true;
    }
    else {
        textBox1.Text="Введіть інші числа p і g";
    }   }
void Button2Click(object sender, EventArgs e)
{
    if (validation1==true) {
        p=pUser;
        g=gUser;
    }   }
void Button3Click(object sender, EventArgs e)
{
    s3=textBox3.Text;
    x=Convert.ToInt32(s3);
    if ((x>1)&&(x<p)) {
        textBox4.Enabled=true;
        textBox5.Enabled=true;
        y=Math.Pow(g,x)%p;
        s1=Convert.ToString(y);
        textBox4.Text=s1;
        Random RND=new Random();
        int k=RND.Next(1,p-1);
        s2=Convert.ToString(k);
        textBox5.Text=s2;
    }
    else {
        textBox4.Enabled=false;
        textBox5.Enabled=false;
        textBox4.Text="Ведіть інше x";
```

```
    }    }
void Button4Click(object sender, EventArgs e)
{
    if (textBox6.Text!="") {
    s1=textBox6.Text;
    s1=s1.ToLower();
    Mchar=Convert.ToChar(s1);
    Mint=coder.IndexOf(Mchar);
    x=Convert.ToInt32(textBox3.Text);
    y=Convert.ToDouble(textBox4.Text);
    k=Convert.ToInt32(textBox5.Text);
    double a=Math.Pow(g,k)%p;
    double b=(Math.Pow(y,k)*Mint)%p;
    s1=Convert.ToString(a);
    textBox7.Text=s1;
    s2=Convert.ToString(b);
    textBox8.Text=s2;
        }    }    }    }
```

Шифр 1 раунду DES:

```
private const int sizeOfBlock = 128;
private const int sizeOfChar = 16;
private const int shiftKey = 2;
private const int quantityOfRounds = 16;
string[] Blocks;
string s = "";
string result="";
string []binBlocks=new string[255];
string [] hexBlocks=new string[255];
private string StringToRightLength(string input)
    {
    while (((input.Length * sizeOfChar) % sizeOfBlock) != 0)
        input += "#";
         return input;
    }
         private void CutStringIntoBlocks(string input)
    {
          Blocks = new string[(input.Length * sizeOfChar) / sizeOfBlock];
```

```
            int lengthOfBlock = input.Length / Blocks.Length;
            for (int i = 0; i < Blocks.Length; i++)
                {
                    Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
                    Blocks[i] = StringToBinaryFormat(Blocks[i]);
                }   }
            private void CutBinaryStringIntoBlocks(string input)
{
            Blocks = new string[input.Length / sizeOfBlock];
            int lengthOfBlock = input.Length / Blocks.Length;
            for (int i = 0; i < Blocks.Length; i++)
                Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
        }
            private string StringToBinaryFormat(string input)
                {
                    string output = "";
                    for (int i = 0; i < input.Length; i++)
            {
                string char_binary = Convert.ToString(input[i], 2);
                while (char_binary.Length < sizeOfChar)
                    char_binary = "0" + char_binary;
                output += char_binary;
                }
        return output;
            }
    private string CorrectKeyWord(string input, int lengthKey)
            {
                if (input.Length > lengthKey)
                input = input.Substring(0, lengthKey);
                        else
                            while (input.Length < lengthKey)
                input = "0" + input;
                return input;
            }
            private string EncodeDES_One_Round(string input, string key)
            {
                string L = input.Substring(0, input.Length / 2);
```

```
            string R = input.Substring(input.Length / 2, input.Length / 2);
            return (R + XOR(L, f(R, key)));
        }
        private string XOR(string s1, string s2)
        {
            string result = "";
            for (int i = 0; i < s1.Length; i++)
    {
            bool a = Convert.ToBoolean(Convert.ToInt32(s1[i].ToString()));
            bool b = Convert.ToBoolean(Convert.ToInt32(s2[i].ToString()));
            if (a ^ b)
                    result += "1";
                else
                result += "0";
    }
        return result;
        }
        private string f(string s1, string s2)
        {
                return XOR(s1, s2);
        }
        private string KeyToNextRound(string key)
        {
                for (int i = 0; i < shiftKey; i++)
                        {
                            key = key[key.Length - 1] + key;
                            key = key.Remove(key.Length - 1);
                        }
                return key;
        }
private string KeyToPrevRound(string key)
                {
                 for (int i = 0; i < shiftKey; i++)
                        {
                            key = key + key[0];
                            key = key.Remove(0, 1);
                        }
```

```
            return key;
            }
        private string StringFromBinaryToNormalFormat(string input)
            {
            string output = "";
            while (input.Length > 0)
                {
                string char_binary = input.Substring(0, sizeOfChar);
                input = input.Remove(0, sizeOfChar);
                int a = 0;
                int degree = char_binary.Length - 1;
                foreach (char c in char_binary)
                a += Convert.ToInt32(c.ToString()) * (int)Math.Pow(2, degree--);
                output += ((char)a).ToString();
                }
                return output;
                }
void MainFormLoad(object sender, EventArgs e)
{       }
    void Button1Click(object sender, EventArgs e)
{
    if ((textBoxEncodeKeyWord.Text.Length > 0)&&(InputTextBox.Text!=""))
    {
  s = "";
  string key = textBoxEncodeKeyWord.Text;
 s=InputTextBox.Text;
        s = StringToRightLength(s);
        CutStringIntoBlocks(s);
        key = CorrectKeyWord(key, s.Length / (2 * Blocks.Length));
        textBoxEncodeKeyWord.Text = key;
        key = StringToBinaryFormat(key);
        for (int j = 0; j < quantityOfRounds; j++)
                        {
                            for (int i = 0; i < Blocks.Length; i++)
                        Blocks[i] = EncodeDES_One_Round(Blocks[i], key);
                            key = KeyToNextRound(key);
                                }
```

```
                key = KeyToPrevRound(key);
                result = "";
                for (int i = 0; i < Blocks.Length; i++)
            result += Blocks[i];
        CutStringS(result);
                            }
        else {
                            MessageBox.Show("Введите ключевое слово!");
                                }    }
void CutStringS(string input) {
        int StrLen=input.Length;
        int maxIterations=Convert.ToInt32(StrLen/sizeOfChar);
        int pos=0;
        for (int j=0;j<maxIterations;j++) {
            pos=j*sizeOfChar;
            binBlocks[j]=input.Substring(pos,sizeOfChar);
            textBox4.AppendText(binBlocks[j]+Environment.NewLine);
            int GGW=Convert.ToInt32(binBlocks[j],2);
            hexBlocks[j]=Convert.ToString(GGW,16);
            textBox1.AppendText(hexBlocks[j]+Environment.NewLine);
        }          }
void Button2Click(object sender, EventArgs e)
{
        CutStringS(result);
}
}}
```

ДОДАТОК Б

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;
using System.Data;
using System.Drawing.Drawing2D;
using System.Text;
using System.Text.RegularExpressions;
using System.IO;
using System.Management;
using System.Diagnostics;
namespace CodAlgorithm
{
    /// <summary>
    /// Description of MainForm.
    /// </summary>
    public partial class MainForm : Form
    {
        public MainForm()
        {
                InitializeComponent();
        }
        int xBase,yBase;
        int qSelector=0;
        int xCurrent=30, yCurrent=0;
        int a=120;
        int b=45;
        int numberFigures=0;
        string indikOperation="";
        string userOperations="";
        int stepY=50;
        Point [] points =new Point[8];
        Bitmap myBitmap;
        int classOfCode=0;
        string coderName="";
```

```
        string []codeLines=new string[255];
        string [] affineCodeLines=new string[255];
        string []ElGanalCodeLines=new string[255];
        string []DEScodeLines=new string[255];
        int maxInput;
        int dlina;
        int startPos=0;
        string []commandsAffine=new string[20];
        string []helpAffine=new string[9];
        string[]commandsGamal=new string[20];
        string []helpGamal=new string[7];
        string sss="";
        string []commandsDES=new string[12];
        string []helpDES=new string[8];
// Ініціалізація змінних, які виводять код програми на третій вкладці:
void initCoderLines() {
            affineCodeLines[0]=" {";
            affineCodeLines[1]="string coder=\"абвгдеєжзиіїйклмнопрстуфхцчшщьюя\";";
            affineCodeLines[2]="string startString=textBox1.Text;";
            affineCodeLines[3]="int max=coder.Length;";
            affineCodeLines[4]="int startStringLength=startString.Length;";
            affineCodeLines[5]="a=Convert.ToInt32(textBox1.Text);";
            affineCodeLines[6]="b=Convert.ToInt32(textBox2.Text);";
            affineCodeLines[7]="if (NDS(a,startStringLength)!=0) ";
            affineCodeLines[8]="{";
            affineCodeLines[9]="textBox3.Text=\"Введіть інше a\";";
            affineCodeLines[10]="}";
            affineCodeLines[11]="else {";
            affineCodeLines[12]="for (i=0;i<startStringLength;i++) {";
            affineCodeLines[13]="litera=startString[i];";
            affineCodeLines[14]="oldNumber=coder.IndexOf(litera);";
            affineCodeLines[15]="startSymbolCounter[i]=oldNumber;";
            affineCodeLines[16]="newNumber=a*startSymbolCounter[i]+b;";
            affineCodeLines[17]="finishedSymbolCounter[i]=newNumber;";
            affineCodeLines[18]="finishedSymbolCounter[i]=finishedSymbolCounter[i]%max;";
            affineCodeLines[19]="litera=coder[finishedSymbolCounter[i]];";
            affineCodeLines[20]="ssss=ssss+Convert.ToString(litera);";
```

```
affineCodeLines[21]="}";

ElGanalCodeLines[0]=" {";
ElGanalCodeLines[1]="int p=Convert.ToInt32(textBox1.Text);";
ElGanalCodeLines[2]="int g=Convert.ToInt32(textBox2.Text);";
ElGanalCodeLines[3]="int x=Convert.ToInt32(textBox3.Text);";
ElGanalCodeLines[4]="if (controlP(pUser)==false) {";
ElGanalCodeLines[5]="p=pUser;";
ElGanalCodeLines[6]="}";
ElGanalCodeLines[7]="if ((controlG(gUser)==false)&&(pUser%gUser!=0)) {";
ElGanalCodeLines[8]="g=gUser;";
ElGanalCodeLines[9]="}   else     {";
ElGanalCodeLines[10]="textBox2.Text=\"Введіть інше g\";";
ElGanalCodeLines[11]="}";
ElGanalCodeLines[12]="double y=Math.Pow(g,x)%p;";
ElGanalCodeLines[13]="int M=Convert.ToInt32(textBox4.Text);";
ElGanalCodeLines[14]="int k=Convert.ToInt32(textBox5.Text);";
ElGanalCodeLines[15]="double a=Math.Pow(g,k)%p;";
ElGanalCodeLines[16]="double b=(Math.Pow(y,k)*Mint)%p;";
ElGanalCodeLines[17]="}";
DEScodeLines [0]=" {";
DEScodeLines [1]=" string s=InputTextBox.Text";
DEScodeLines [2]=" string s=InputTextBox.Text;";
DEScodeLines [3]=" while (((input.Length * sizeOfChar) % sizeOfBlock) != 0)";
DEScodeLines [4]=" input += \"#\";";
DEScodeLines [5]=" for (int i = 0; i < Blocks.Length; i++)     {";
DEScodeLines [6]=" Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);";
DEScodeLines [7]=" Blocks[i] = StringToBinaryFormat(Blocks[i]);";
DEScodeLines [8]=" }";
DEScodeLines [9]=" if (input.Length > lengthKey)";
DEScodeLines [10]="input = input.Substring(0, lengthKey)";
DEScodeLines [11]="else {";
DEScodeLines [12]=" while (input.Length < lengthKey) {";
DEScodeLines [13]="input = \"0\" + input;";
DEScodeLines [14]="}                    }";
DEScodeLines [15]="for (int i = 0; i < input.Length; i++)    {";
DEScodeLines [16]="string char_binary = Convert.ToString(input[i], 2);";
```

```
DEScodeLines [17]="while (char_binary.Length < sizeOfChar)";
DEScodeLines [18]="char_binary = \"0\" + char_binary;";
DEScodeLines [19]="output += char_binary;";
DEScodeLines [20]="}";
DEScodeLines [21]="string L = input.Substring(0, input.Length / 2);";
DEScodeLines [22]="string R = input.Substring(input.Length / 2, input.Length / 2);";
DEScodeLines [23]="return (R + XOR(L, f(R, key)));";
DEScodeLines [24]="for (int i = 0; i < Blocks.Length; i++)";
DEScodeLines [25]="result += Blocks[i];";
DEScodeLines [26]="while (input.Length > 0)    {";
DEScodeLines [27]="string char_binary = input.Substring(0, sizeOfChar);";
DEScodeLines [28]="input = input.Remove(0, sizeOfChar);";
DEScodeLines [29]="int a = 0;";
DEScodeLines [30]="int degree = char_binary.Length - 1;";
DEScodeLines [31]="foreach (char c in char_binary)";
DEScodeLines [32]="a += Convert.ToInt32(c.ToString()) * (int)Math.Pow(2, degree--
);";
DEScodeLines [33]="output += ((char)a).ToString();";
DEScodeLines [34]="}";
DEScodeLines [35]="textBox3.Text=result;";
DEScodeLines [36]="}";        }
```

//Ініціалізація змінних, що виводять написи на елементах блок-схеми і підказку в текстовому полі

```
void initBlockScheme(){
        commandsAffine[0]="Початок";
        commandsAffine[1]="Введення алфавіту."+Environment.NewLine+"Введення тексту X";
        commandsAffine[2]="Визначення потужності "+Environment.NewLine+"алфавіту n."+Environment.NewLine+"Визначення довжини тексту h";
        commandsAffine[3]="Введення ключів a і b";
        commandsAffine[4]="Перевірка умови"+Environment.NewLine+"НДС(a,n)=1.";
        commandsAffine[5]="i:=0";
        commandsAffine[6]="i<=h";//+Environment.NewLine+"(якщо умова не"+Environment.NewLine+"виконується, то"+Environment.NewLine+"виводимо"+Environment.NewLine+"шифротекст";
        commandsAffine[7]="Yi=(aXi+b) mod n";
        commandsAffine[8]="i:=i+1";
```

```
commandsAffine[9]="Повернення до"+Environment.NewLine+"пункту i<=h";
commandsGamal[0]="Початок";
commandsGamal[1]="Введення простого"+Environment.NewLine+"великого числа p";
commandsGamal[2]="Введення g і x";
commandsGamal[3]="Перевірка умови"+Environment.NewLine+"НДС(a,n)=1";//+Environment.NewLine+"(якщо умова не виконується,"+Environment.NewLine+"то виконуємо попередній"+Environment.NewLine+"пункт ще раз)";
commandsGamal[4]="Обчислення"+Environment.NewLine+"y=g^x(mod p)";
commandsGamal[5]="Введення M";
commandsGamal[6]="Введення сесійного"+Environment.NewLine+"ключа k"+Environment.NewLine+"(0<k<p-1)";
commandsGamal[7]="Обчислення"+Environment.NewLine+"a=g^k(mod p)"+Environment.NewLine+"b=M*y^k(mod p)";
commandsGamal[8]="a,b,Y";
commandsGamal[9]="Кінець";
helpAffine[0]="Підказка:";
helpAffine[1]="X - текст";
helpAffine[2]="Y - шифротекст";
helpAffine[3]="a, b - ключі";
helpAffine[4]="n - потужність алфавіту";
helpAffine[5]="h - довжина тексту";
helpAffine[6]="i - лічильник циклу";
helpGamal[0]="Підказка:";
helpGamal[1]="p, g, y - відкриті ключі";
helpGamal[2]="x - закритий ключ";
helpGamal[3]="k - випадкове число";
helpGamal[4]="M - відкритий текст";
helpGamal[5]="h - довжина повідомлення";
helpGamal[6]="n - потужність алфавіту";
commandsDES[0]="Початок";
commandsDES[1]="Введення"+Environment.NewLine+"повідомлення."+Environment.NewLine+"Введення ключа";
commandsDES[2]="Надання"+Environment.NewLine+" повідомленню"+Environment.NewLine+"потрібного розміру";
commandsDES[3]="Розбиття "+Environment.NewLine+"повідомлення"+Environment.NewLine+"на блоки";
```

```
                commandsDES[4]="Hx>Hk";
                commandsDES[5]="Надання ключу"+Environment.NewLine+"потрібного розміру";
                commandsDES[6]="Переведення "+Environment.NewLine+"повідомлення"+Environment.NewLine+"у двійковий формат";
                commandsDES[7]="1 раунд шифрування"+Environment.NewLine+"Ri+1=Li+f(Ri,Ki+1)";
                commandsDES[8]="Переведення "+Environment.NewLine+"рядка"+Environment.NewLine+"в символьний формат";
                commandsDES[9]="Виведення "+Environment.NewLine+"шифротексту";
                commandsDES[10]="Кінець";
                helpDES[0]="Підказка";
                helpDES[1]="Hx - довжина повідомлення";
                helpDES[2]="Hk - довжина ключа";
                helpDES[3]="L - ліва частина повідомлення";
                helpDES[4]="R  - права частина повідомлення";
                helpDES[5]="K - ключ";        }
    //Створення об’єкту Bitmap для графічного зображення блок-схеми:
        void MainFormLoad(object sender, EventArgs e)
            {
                initBlockScheme();
                initCoderLines() ;
                myBitmap=new Bitmap(pictureBox1.Width,pictureBox1.Height);
                xBase=Convert.ToInt32(0.5*(pictureBox1.Width-a));
            }
        //Можливість переходу на другу вкладку після вибору однієї з радіо кнопок, що відображають криптоалгоритм:
        void RadioButton1CheckedChanged(object sender, EventArgs e)
            {           button1.Enabled=true;      }
        void RadioButton2CheckedChanged(object sender, EventArgs e)
            {           button1.Enabled=true;      }
        void RadioButton3CheckedChanged(object sender, EventArgs e)
            {           button1.Enabled=true;      }
        //Обробка натискання на кнопку переходу на першій вкладці. Виводиться підказка і один з трьох видів блок-схем:
        void Button1Click(object sender, EventArgs e)
            {
```

```
            textBoxHelp.Text="";
            if (radioButton1.Checked==true) {
                tabControl1.SelectedTab = tabPage2;
                classOfCode=1;//шифр цезаря
                formulaYes="binavohyprf";
                groupBox2.Visible=true;
                groupBox3.Visible=false;
                groupBox4.Visible=false;
                for (int jj=0;jj<6;jj++) {
                    textBoxHelp.AppendText(helpAffine[jj]+Environment.NewLine);
} }
            else {
                if (radioButton2.Checked==true) {
                    classOfCode=2;//шифр ель гамаля
                    tabControl1.SelectedTab = tabPage2;
                    formulaYes="bpgnyMkaf";
                    groupBox2.Visible=false;
                groupBox3.Top=100;
                groupBox3.Visible=true;
                groupBox4.Visible=false;
                    for (int jj=0;jj<6;jj++) {
                        textBoxHelp.AppendText(helpGamal[jj]+Environment.NewLine); } }
                else {
                    if (radioButton3.Checked==true) {
                        classOfCode=3;//шифр DES
                    tabControl1.SelectedTab = tabPage2;
                    formulaYes="bixdketrsof";
                    groupBox2.Visible=false;
                groupBox3.Visible=false;
                groupBox4.Top=100;
                groupBox4.Visible=true;
                    for (int jj=0;jj<6;jj++) {
                    textBoxHelp.AppendText(helpDES[jj]+Environment.NewLine);}}}  }
            xCurrent=xBase;
            yCurrent=yBase+5;
            tabPage2.Enabled=true;     }
        //Обробка натискання на кнопку повернення на другій вкладці. Обнуляються початкові дані:
```

```
void Button13Click(object sender, EventArgs e)
        {
            classOfCode=0;
            tabControl1.SelectedTab = tabPage1;
            radioButton1.Checked=false;
            radioButton2.Checked=false;
            radioButton3.Checked=false;
            button1.Enabled=false;
            pictureBox1.BackColor=Color.Black;
            pictureBox1.BackColor=Color.Transparent;
        }
public void drawEll(int x0, int y0, int aa, int bb,string ss) {
            //графічні примітиви
            points[0].X=x0;
            points[0].Y=y0;
            points[1].X=x0+aa;
            points[1].Y=y0+Convert.ToInt32(bb*0.7);
            pictureBox1.Image = myBitmap;
        Graphics    GG1 = Graphics.FromImage (myBitmap);
            Pen PP1=new Pen(Color.Black);
            SolidBrush SB1=new SolidBrush(Color.Purple);
            GG1.DrawEllipse(PP1,x0,y0,aa,bb);
            GG1.FillEllipse(SB1,x0,y0,aa,bb);
            pictureBox1.Image = myBitmap;
            Font Font1= new System.Drawing.Font("Times New Roman", 8, FontStyle.Regular);
            String Text = String.Format("{0}", ss);
            Brush SSB = new SolidBrush(Color.White);
            GG1 = Graphics.FromImage (myBitmap);
            GG1.TextRenderingHint = System.Drawing.Text.TextRenderingHint.AntiAlias;
            GG1.DrawString(Text, Font1, SSB, x0+15, y0+10);
        }
        // Початок і Кінець алгоритму
        void Button2Click(object sender, EventArgs e)
        {
            sss="Початок";
            if (yCurrent<(pictureBox1.Height-stepY)) {
            qSelector=1;
```

```
    indikOperation="b";
    userOperations=userOperations+indikOperation;
    drawEll(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }   }
void Button3Click(object sender, EventArgs e)
{
    sss="Кінець";
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=1;
    indikOperation="f";
    userOperations=userOperations+indikOperation;
    drawEll(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }
}
    public void drawParal(int x0, int y0, int aa, int bb,string ss) {
    pictureBox1.Image = myBitmap;
    Graphics    GG1 = Graphics.FromImage (myBitmap);
    Pen PP1=new Pen(Color.Black,2);
    SolidBrush SB1=new SolidBrush(Color.Navy);
    Point point1 = new Point(xCurrent+15,  yCurrent);
    Point point2 = new Point(xCurrent,  yCurrent+b);
    Point point3 = new Point(xCurrent+a-15,  yCurrent+b);
    Point point4 = new Point(xCurrent+a,  yCurrent);
    Point[] curvePoints =  { point1, point2,point3,  point4};
    GG1.DrawPolygon(PP1,curvePoints);
    GG1.FillPolygon(SB1,curvePoints);
    }
```

```
void Button4Click(object sender, EventArgs e)
{
    sss="Введення алфавіту."+Environment.NewLine+"Введення тексту X";
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="i";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}           else {
        MessageBox.Show("Схема вийшла за межі екрану!");       }    }
void Button5Click(object sender, EventArgs e)
{
    sss="Введення ключів"+Environment.NewLine+"a i b";
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="a";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}           else {
        MessageBox.Show("Схема вийшла за межі екрану!");   }    }
//Малювання прямокутника з заданими координатами і текстом всередині:
public void drawRRR(int x0, int y0, int aa, int bb, string ss) {
    pictureBox1.Image = myBitmap;
Graphics    GG1 = Graphics.FromImage (myBitmap);
    Pen PP1=new Pen(Color.Black);
    SolidBrush SB1=new SolidBrush(Color.LightBlue);
    GG1.DrawRectangle(PP1,x0,y0,aa,bb);
    GG1.FillRectangle(SB1,x0,y0,aa,bb);
void Button6Click(object sender, EventArgs e)
{
    sss="Визначення потужності "+Environment.NewLine+"алфавіту
n."+Environment.NewLine+"Визначення довжини"+Environment.NewLine+" тексту h";
    if (yCurrent<(pictureBox1.Height-stepY)) {
```

```
    qSelector=2;
    indikOperation="n";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");} }
void Button7Click(object sender, EventArgs e)
{
    sss="i:=0 ";
        if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="o";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");      }       }
void Button8Click(object sender, EventArgs e)
{
    sss="Yi=(aXi+b)mod n";
        if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="y";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }   }
void Button9Click(object sender, EventArgs e)
{
    sss="i:=i+1";
```

```
        if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="p";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);

    numberFigures++;
    yCurrent=yCurrent+stepY;
    }
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }   }

void Button10Click(object sender, EventArgs e)
{
        sss="Повернення до "+Environment.NewLine+"пункту i<=h";
        if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="r";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);

    numberFigures++;
    yCurrent=yCurrent+stepY;
    }
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }   }

public void drawRomb(int x0, int y0, int aa, int bb, string ss) {
    pictureBox1.Image = myBitmap;
Graphics    GG1 = Graphics.FromImage (myBitmap);
    Pen PP1=new Pen(Color.Black,2);
        SolidBrush SB1=new SolidBrush(Color.Green);
        Point point1 = new Point(xCurrent+Convert.ToInt32(0.5*a),  yCurrent);
        Point point2 = new Point(xCurrent,  yCurrent+Convert.ToInt32(0.5*b));
    Point point3 = new Point(xCurrent+Convert.ToInt32(0.5*a),  yCurrent+b);
```

```
    Point point4 = new Point(xCurrent+a,  yCurrent+Convert.ToInt32(0.5*b));
        Point[] curvePoints =  { point1, point2,point3,  point4};

        GG1.DrawPolygon(PP1,curvePoints);
        GG1.FillPolygon(SB1,curvePoints);
    }
void Button11Click(object sender, EventArgs e)
{
    sss="Перевірка умови"+Environment.NewLine+"НДС(a,n)=1";
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=4;
    indikOperation="v";
    userOperations=userOperations+indikOperation;
    drawRomb(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }       }
void Button12Click(object sender, EventArgs e)
{
    sss="Перевірка умови"+Environment.NewLine+"i<=h";
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=4;
    indikOperation="h";
    userOperations=userOperations+indikOperation;
    drawRomb(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}
    else {
        MessageBox.Show("Схема вийшла за межі екрану!");
    }       }
//Підпрограма витирання останньої фігури на блок-схемі:
private Bitmap MakeImageWithoutArea(Bitmap source_bm,    List<Point> points)
      {
```

```
                Bitmap bm = new Bitmap(source_bm);
                  using (Graphics gr = Graphics.FromImage(bm))
                                {
                               GraphicsPath path = new GraphicsPath();
                                path.AddPolygon(points.ToArray());
                                gr.SetClip(path);
                                  gr.Clear(Color.Transparent);
                                   gr.ResetClip();
                                     }
                                     return bm;          }
void Button14Click(object sender, EventArgs e)
{
    yCurrent=yCurrent-stepY;
    if (yCurrent>0) {
    int PPOS=userOperations.Length;
    string LastOperation=Convert.ToString(userOperations[PPOS-1]);
    userOperations=userOperations.Substring(0,PPOS-1);
    if (LastOperation=="m") {
         points [0]= new Point(xCurrent+a-2,  yCurrent-2);
    points [1] = new Point(xCurrent+3*a+2,  yCurrent-2);//замість 3а треба 2а
    points [2]=new Point(xCurrent+3*a+2,yCurrent+b+2);//замість 3а треба 2а
    points [3]=new Point(xCurrent+a-2,yCurrent+b+2);
    points [4]= new Point(xCurrent+a-2,  yCurrent-2);
    }
    else {
    points [0]=new Point(xCurrent-2,yCurrent-2);
    points [1]=new Point(xCurrent-2,yCurrent+b+2);
    points [2]=new Point(xCurrent+3*a+2,yCurrent+b+2);//замість 3а треба а
    points [3]=new Point(xCurrent+3*a+2,yCurrent-2);//замість 3а треба а
    points [4]=new Point(xCurrent-2,yCurrent-2);
    }
    pictureBox1.Image = myBitmap;
Graphics    GG7 = Graphics.FromImage (myBitmap);
GraphicsPath path = new GraphicsPath();
path.AddPolygon(points);
GG7.SetClip(path);
GG7.Clear(Color.Transparent)          ;
```

```
GG7.ResetClip();
numberFigures--;
    }    }

//шифр ЕльГамаля
void Button15Click(object sender, EventArgs e)
{
    sss=commandsGamal[1];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="p";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}              else {
        MessageBox.Show("Схема вийшла за межі екрану!");        }          }
void Button16Click(object sender, EventArgs e)
{
    sss=commandsGamal[2];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="g";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}              else {
        MessageBox.Show("Схема вийшла за межі екрану!");              }          }
void Button17Click(object sender, EventArgs e)
{
    sss=commandsGamal[3];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=4;
    indikOperation="n";
    userOperations=userOperations+indikOperation;
    drawRomb(xCurrent, yCurrent,a,b,sss);
```

```
    numberFigures++;
    yCurrent=yCurrent+stepY;
}           else {
        MessageBox.Show("Схема вийшла за межі екрану!");        }       }
void Button18Click(object sender, EventArgs e)
{
    sss=commandsGamal[4];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="y";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");        }       }
void Button19Click(object sender, EventArgs e)
{
    sss=commandsGamal[5];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="M";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}           else {
        MessageBox.Show("Схема вийшла за межі екрану!");        }       }
void Button20Click(object sender, EventArgs e)
{
    sss=commandsGamal[6];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="k";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
```

```
    yCurrent=yCurrent+stepY;
}               else {
        MessageBox.Show("Схема вийшла за межі екрану!");            }       }
void Button21Click(object sender, EventArgs e)
{
    sss=commandsGamal[7];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="a";
    userOperations=userOperations+indikOperation;
    drawRRR(xCurrent,yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
    }               else {
        MessageBox.Show("Схема вийшла за межі екрану!");            }       }
//шифр DES
void Button22Click(object sender, EventArgs e)
{
    sss=commandsDES[1];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=3;
    indikOperation="i";
    userOperations=userOperations+indikOperation;
    drawParal(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}               else {
        MessageBox.Show("Схема вийшла за межі екрану!");            }       }
public void drawHexagon(int x0, int y0, int aa, int bb, string ss) {
pictureBox1.Image = myBitmap;
Graphics    GG1 = Graphics.FromImage (myBitmap);
    Pen PP1=new Pen(Color.Black,2);
    SolidBrush SB1=new SolidBrush(Color.Brown);
    Point point1 = new Point(xCurrent+Convert.ToInt32(0.3*a),  yCurrent);
    Point point2 = new Point(xCurrent,  yCurrent+Convert.ToInt32(0.5*b));
    Point point3 = new Point(xCurrent+Convert.ToInt32(0.3*a),  yCurrent+b);
    Point point4 = new Point(xCurrent+Convert.ToInt32(0.7*a),  yCurrent+b);
```

```
    Point point5 = new Point(xCurrent+a,  yCurrent+Convert.ToInt32(0.5*b));
    Point point6 = new Point(xCurrent+Convert.ToInt32(0.7*a),  yCurrent);
        Point[] curvePoints =  { point1, point2,point3,  point4, point5,point6};
        GG1.DrawPolygon(PP1,curvePoints);
        GG1.FillPolygon(SB1,curvePoints);

}
void Button23Click(object sender, EventArgs e)
{
    sss=commandsDES[2];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=5;
    indikOperation="x";
    userOperations=userOperations+indikOperation;
    drawHexagon(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
                }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");          }       }
void Button24Click(object sender, EventArgs e)
{
    sss=commandsDES[3];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=5;
    indikOperation="d";
    userOperations=userOperations+indikOperation;
    drawHexagon(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
                }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");          }       }
void Button25Click(object sender, EventArgs e)
{
    sss=commandsDES[4];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=4;
    indikOperation="k";
```

```
    userOperations=userOperations+indikOperation;
    drawRomb(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
}           else {
        MessageBox.Show("Схема вийшла за межі екрану!");          }       }
void Button26Click(object sender, EventArgs e)
{
    sss=commandsDES[5];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=5;
    indikOperation="e";
    userOperations=userOperations+indikOperation;
    drawHexagon(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
            }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");          }       }
void Button27Click(object sender, EventArgs e)
{
    sss=commandsDES[6];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=5;
    indikOperation="t";
    userOperations=userOperations+indikOperation;
    drawHexagon(xCurrent, yCurrent,a,b,sss);
    numberFigures++;
    yCurrent=yCurrent+stepY;
                }           else {
        MessageBox.Show("Схема вийшла за межі екрану!");          }       }
void Button28Click(object sender, EventArgs e)
{
    sss=commandsDES[7];
    if (yCurrent<(pictureBox1.Height-stepY)) {
    qSelector=2;
    indikOperation="r";
    userOperations=userOperations+indikOperation;
```

```
            drawRRR(xCurrent,yCurrent,a,b,sss);
            numberFigures++;
            yCurrent=yCurrent+stepY;
            }               else {
                MessageBox.Show("Схема вийшла за межі екрану!");            }       }
        void Button29Click(object sender, EventArgs e)
        {
            sss=commandsDES[8];
                    if (yCurrent<(pictureBox1.Height-stepY)) {
            qSelector=5;
            indikOperation="s";
            userOperations=userOperations+indikOperation;
            drawHexagon(xCurrent, yCurrent,a,b,sss);
            numberFigures++;
            yCurrent=yCurrent+stepY;
                        }               else {
                MessageBox.Show("Схема вийшла за межі екрану!");            }       }
        void Button30Click(object sender, EventArgs e)
        {
            sss=commandsDES[9];
            if (yCurrent<(pictureBox1.Height-stepY)) {
            qSelector=3;
            indikOperation="o";
            userOperations=userOperations+indikOperation;
            drawParal(xCurrent, yCurrent,a,b,sss);
            numberFigures++;
            yCurrent=yCurrent+stepY;
        }               else {
                MessageBox.Show("Схема вийшла за межі екрану!");            }       }
//натискання на кнопку «Перевірити»:
        void Button31Click(object sender, EventArgs e)
        {
            MessageBox.Show(userOperations);
            MessageBox.Show(formulaYes);
            int RES = String.Compare(userOperations, formulaYes);
            if (RES==0) {
                button13.Enabled=true;
```

```
            MessageBox.Show("Правильно");
            button32.Enabled=true;
            button14.Enabled=false;
        }           else {
            MessageBox.Show("Неправильно");
            button32.Enabled=false;          }       }
    // Знищення графічного об’єкту у пам’яті під час закриття форми:
    void MainFormFormClosing(object sender, FormClosingEventArgs e)
    {
        myBitmap.Dispose();
    }
//Перехід на третю вкладку, виведення у RichTextBox коду алгоритму різним кольором:
    void Button32Click(object sender, EventArgs e)
    {
        richTextBox1.Text="";
        tabControl1.SelectedTab = tabPage3;
        if (classOfCode==1) {
            codeLines=affineCodeLines;
            maxInput=22;            }
        else {if (classOfCode==2) {
            codeLines=ElGanalCodeLines;
            maxInput=18;          }
        else {
            if (classOfCode==3) {
                codeLines=DEScodeLines;
                maxInput=37;
            }   }   }
        for (int i=0;i<maxInput;i++){
            richTextBox1.AppendText(codeLines[i]+Environment.NewLine);
        }
        dlina=0;
        if (classOfCode==1) {
            startPos=0;
            dlina=affineCodeLines[0].Length;
            richTextBox1.Select(startPos,dlina);
            richTextBox1.SelectionColor=Color.Purple;
            startPos=startPos+dlina;
```

```
                    dlina=affineCodeLines[1].Length+affineCodeLines[2].Length+2;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Blue;
                    startPos=startPos+dlina;
                    dlina=affineCodeLines[3].Length+affineCodeLines[4].Length+2;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.LightBlue;
                    startPos=startPos+dlina;
                    dlina=affineCodeLines[5].Length+affineCodeLines[6].Length+2;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Blue;
                    startPos=startPos+dlina;
    dlina=affineCodeLines[7].Length+affineCodeLines[8].Length+affineCodeLines[9].Length+affineCo
deLines[10].Length+4;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Green;
                    startPos=startPos+dlina;
                    dlina=0;
                    for (int j=11;j<22;j++) {
                    dlina=dlina+affineCodeLines[j].Length;
                    }
                    dlina=dlina+12;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.LightBlue;
                }
                else {
                    if (classOfCode==2) {
                    startPos=0;
                    dlina=ElGanalCodeLines[0].Length;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Purple;
                    startPos=startPos+dlina;
        dlina=ElGanalCodeLines[1].Length+ElGanalCodeLines[2].Length+ElGanalCodeLines[3].Length
+3;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Blue;
                    startPos=startPos+dlina;
```

```
dlina=0;
for (int j=4;j<12;j++) {
dlina=dlina+ElGanalCodeLines[j].Length;
}
dlina=dlina+9;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Green;
startPos=startPos+dlina;
dlina=ElGanalCodeLines[12].Length;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.LightBlue;
startPos=startPos+dlina;
dlina=ElGanalCodeLines[13].Length+ElGanalCodeLines[14].Length+2;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Blue;
startPos=startPos+dlina;
dlina=ElGanalCodeLines[15].Length+ElGanalCodeLines[16].Length+2;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.LightBlue;
startPos=startPos+dlina;
//тут додати виведення даних
dlina=ElGanalCodeLines[17].Length+1;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Purple;
} else {
    if (classOfCode==3) {
        startPos=0;
        dlina=DEScodeLines[0].Length+1;
      richTextBox1.Select(startPos,dlina);
      richTextBox1.SelectionColor=Color.Purple;
      startPos=startPos+dlina;
      dlina=DEScodeLines[1].Length+DEScodeLines[2].Length+2;
      richTextBox1.Select(startPos,dlina);
      richTextBox1.SelectionColor=Color.Blue;
      startPos=startPos+dlina;
      dlina=DEScodeLines[3].Length+DEScodeLines[4].Length+2;
      richTextBox1.Select(startPos,dlina);
```

```
richTextBox1.SelectionColor=Color.Orange;
startPos=startPos+dlina;
dlina=0;
for (int j=5;j<9;j++)
{
   dlina=dlina+DEScodeLines[j].Length;
}
dlina=dlina+3;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Orange;
startPos=startPos+dlina;
dlina=DEScodeLines[9].Length+1;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Green;
startPos=startPos+dlina;
dlina=DEScodeLines[10].Length+1;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.LightBlue;
startPos=startPos+dlina;
dlina=0;
for (int j=11;j<21;j++) {
dlina=dlina+DEScodeLines[j].Length+1;
}
dlina=dlina+0;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.Orange;
   startPos=startPos+dlina;
dlina=0;
for (int j=21;j<24;j++) {
dlina=dlina+DEScodeLines[j].Length+1;
}
dlina=dlina+0;
richTextBox1.Select(startPos,dlina);
richTextBox1.SelectionColor=Color.LightBlue;
startPos=startPos+dlina;
dlina=DEScodeLines[24].Length+DEScodeLines[25].Length+3;
richTextBox1.Select(startPos,dlina);
```

```
                    richTextBox1.SelectionColor=Color.Orange;
                    startPos=startPos+dlina;
                    dlina=0;
                    for (int j=26;j<35;j++) {
                    dlina=dlina+DEScodeLines[j].Length+1;
                    }
                    dlina=dlina+0;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Orange;
                       startPos=startPos+dlina;
                    dlina=DEScodeLines[35].Length+1;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Blue;
                    startPos=startPos+dlina;
                    dlina=DEScodeLines[36].Length+1;
                    richTextBox1.Select(startPos,dlina);
                    richTextBox1.SelectionColor=Color.Purple;
                   }   }   }   }
        // Запуск однієї з трьох додаткових програм-шифраторів:
        void Button33Click(object sender, EventArgs e)
        {
            switch (classOfCode) {
                case 1: coderName="AffineCoder.exe";break;
              case 2:coderName="ElGamalSymbolCoder.exe";break;
             case 3:coderName="NewDES1roundCoder.exe";break;
            }
            if (File.Exists(coderName)) {
            Process.Start (coderName);
            }
            else {
                MessageBox.Show("Файл-кодер відсутній");
            } }
// Повернення на першу сторінку при натисканні вкладок:
                void TabControl1MouseClick(object sender, MouseEventArgs e)
        {
            tabControl1.SelectedTab = tabPage1;
        } }  }
```