Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: **Розроблення програмного забезпечення корекції помилок при завадостійкому кодуванні**

Виконав: студент VI курсу, групи КН-61м
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)
Лучкевич А.Т.
(прізвище та ініціали)
Керівник Шабатура Ю.В.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Львів – 2024 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Борецька І. Б.

"____"_____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Лучкевич Андрій Теодозійович
(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення програмного забезпечення корекції помилок при завадостійкому кодуванні**
керівник роботи, Шабатура Юрій Васильович, д.т.н, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 13.02.2023 року № С-49
2. Термін подання студентом роботи 05. 01. 2024 р.
3. Вихідні дані до роботи:
- провести огляд алгоритмів завадостійкого кодування;
- дослідити математичну модель завадостійкого кодування;
- розробити програмний продукт з інтуїтивним інтерфейсом;
- провести тестування роботи розробленого програмного продукту.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап-проєкту
Висновки
5. Перелік графічного матеріалу:
системний аналіз, розробка та відображення алгоритму роботи завадостійкого кодування, структура програмного рішення, експериментальна частина)
Додаток А. Вихідний код розробленого програмного забезпечення
6. Дата видачі завдання 15 лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання	15.02.2023	виконано
2	Огляд літературних джерел	30.04.2023	виконано
2	Розділ 1. Стан проблемної області	31.07.2023	виконано
3	Розділ 2. Інформаційне забезпечення	30.08..2023	виконано
4	Розділ 3. Математичне забезпечення	29.09.2023	виконано
5	Розділ 4. Програмне забезпечення	31.10.2023	виконано
7	Розділ 5. Розробка стартап-проєкту	30.11.2023	виконано
8	Оформлення пояснювальної записки	29.12.2023	виконано
9	Подання готової роботи	05.01.2024	виконано

Студент _____ Лучкевич А.Т.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шабатура Ю.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 62 сторінки пояснювальної записки, 13 рисунків, 6 таблиць, 1 додаток, 28 літературних джерел.

Розроблений метод спрощеного синтезу стійкої до помилок багатопозиційної кодової послідовності на основі ідеальних кільцевих в’язанок та створений ефективний алгоритм кодування та декодування інформації. Розроблений програмний додаток дозволяє проводити корекцію кодування з виправленням помилок на основі ідеальної кільцевої в’язанки (виправлення до 25% від довжини коду та виявлення до 50% від довжини коду).

Ключові слова: *завадостійке кодування, коригуючі коди, ідеальна кільцева в’язанка*.

ANNOTATION

The master's thesis contains 62 pages of explanatory note, 13 figures, 6 tables, 1 appendix, 28 literary sources.

A method of simplified synthesis of an error-resistant multi-position code sequence based on ideal circular knitting has been developed, and an effective algorithm for encoding and decoding information has been created. The developed software application allows error-correcting coding correction based on perfect circular knitting (correction up to 25% of the code length and detection up to 50% of the code length).

Keywords: *correction codes, ideal ring bundle, interference-resistant coding*.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно розробити програмне та алгоритмічне забезпечення завадостійкого перетворення інформації для корекції помилок, а саме:

1. провести попередній аналіз існуючих завадостійких кодів для корекції помилок;
2. визначити математичну модель завадостійких кодів для корекції помилок;
3. визначити кількість помилок що знаходяться та виправляються завадостійким кодом для корекції помилок;
4. розробити алгоритм синтезу завадостійкого коду для корекції помилок на вибраній математичній моделі;
5. провести інтерперетацію отриманих результатів;
6. розробити програмне забезпечення для представлення результатів роботи.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ОДИНИЦЬ І ТЕРМІНІВ

ІКВ	ідеальна кільцева в’язанка
ПЗ	програмне забезпечення
ЦК	циклічний код
BCH	Bose-Chowdhury-Hocquenghem
CRC	Cyclic Redundancy Check

ВСТУП

Людська діяльність пов’язана з обробкою та використанням матеріалів, енергії та інформації. Тому отримали розвиток науково-технічні дисципліни, що відображають питання техніки, енергетики та інформатики. ІТ – відносно нова галузь, яка досягла найбільшого розвитку на етапі розробки та використання електронно-обчислювальних машин та автоматизованих систем управління.

Інформатика в даний час використовується в різних областях теорії і практики. Центральною галуззю є теорія зв'язку, розроблена Шенноном на основі теорії ймовірностей.

Передача й обробка інформації пов’язана з роботою будь-якого автоматичного пристрою, поведінкою живої істоти, творчою діяльністю людини, розвитком науки й техніки, економічними й соціальними змінами в суспільстві й самому житті. Для більш ефективного використання інформації необхідний обмін інформацією, який неможливий без її передачі по каналах зв'язку.

При передачі інформації по каналах зв'язку може виникнути спотворення переданої інформації. Щоб запобігти втраті корисної інформації, існують різні методи захисту. Одним з них є кодування інформації за допомогою завадостійких кодів.

Двійковий код для всіх комбінацій не є завадостійким, оскільки його комбінації відрізняються один від одного лише однією цифрою, що не дозволяє виявити та виправити помилки на приймальному кінці. Тому існує потреба в створенні стійкого до помилок коду.

Завадостійкі коди - це коди, які дозволяють виявити та виправити помилки, тобто виправити отримані повідомлення. Щоб досягти завадостійкості,

надлишковість може бути введена шляхом додавання додаткових бітів керування.

Тому сьогодні завадостійке кодування перетворення інформації займає важливе місце в сучасних системах передачі даних зв'язку. В даний час існує багато завадостійких методів кодування, які активно використовуються в різних системах передачі даних і тому залишаються актуальними дослідженнями.

Головною проблемою сучасної теорії та техніки зв'язку є підвищення стійкості телекомунікаційних систем до перешкод, особливо в умовах дії як природних, так і штучних перешкод. Однією з основних концепцій підвищення завадостійкості, розроблених у магістерській роботі, є швидка зміна запущених збірок коду, тим самим підвищуючи завадостійкість, а також захист інформації від несанкціонованого доступу.

Метою магістерської роботи є дослідження одного з методів завадостійкого перетворення інформації на основі використання комбінації двійкових кодів на основі ідеальних кільцевих в’язанок.

Об’єктом дослідженя є методи та алгоритми завадостійкого кодування.

Предметом дослідження є завадостійке кодування на основі ідеальних кільцевих в’язанок.

Класичний опис циклічних кодів базується на таких поняттях, як кодова відстань і надмірність, що полегшує вибір кодів із заданими параметрами. Коди на основі ідеальних кільцевих в’язанок, які вивчаються в цій роботі, охоплюють широкий спектр, від «повільних» до «швидких» завадостійких кодів з мінімальними можливостями корекції. У цьому випадку при порівнянні особлива увага приділяється двійковим кодам BCH (Bose-Chowdhury-Hocquenghem).

Властивості кодів BCH детально обговорюються в роботах Peterson V., Weldon E., Kasam T., Clark J., Kane J., Berlekamp E. Про зростаючий інтерес до

кодів BCH також свідчить той факт, що розроблено значну кількість алгоритмів декодування таких кодів як за частотою, так і за часом. У працях професора Когновицького А.С. Розглянуто використання базису подвійного поля Галуа при декодуванні циклічних кодів з урахуванням їх властивостей повторюваності. Для раціонального вибору того чи іншого алгоритму необхідний порівняльний аналіз властивостей різних методів декодування корекційних кодів BLC, у тому числі методів декодування з використанням подвійної основи.

Досліджується ефективність декодування двійкових кодів на основі ідеальної кільцевої в’язанки та використання результатів декодування для оцінки якості каналу в процесі передачі даних. Під ефективністю декодування розуміємо можливості коду корекції, вважаючи критерієм ефективності ймовірність правильного декодування в залежності від ймовірності помилки в каналі.

Дані на стороні передачі кодуються за допомогою одного з добре відомих кодів корекції помилок. Відповідно, на приймальній стороні відбувається декодування отриманої інформації та виправлення виявлених помилок. Можливість виправлення помилок використовуваного коду залежить від кількості надлишкових бітів, згенерованих кодувальником. Якщо введена надлишковість невелика, тобто існує ризик того, що отримані дані будуть містити помилки, які не були знайдені, це може призвести до помилок у роботі процесу програми. Якщо використовується код із високою потужністю корекції, це приводить до низької швидкості передачі даних. Отже, знання теорії завадостійкого кодування дозволяє визначити оптимальні параметри завадостійкого коду залежно від виконуваного завдання.

У зв’язку з цим виділено два напрямки, що визначають постановку завдань роботи. Перший напрямок безпосередньо пов'язаний з дослідженням методів декодування коду на основі ідеальних кільцевих в’язанок, завданням якого є

порівняння з існуючими методами декодування, насамперед з точки зору можливості коригування.

Другий напрямок є логічним продовженням першого і спрямований на підвищення ефективності використання методу декодування на основі ідеальних кільцевих в'язанок шляхом реалізації механізму адаптації системи передачі даних до стану каналу.

Дослідження базується на теорії поля Галуа, теорії кодування, теорії рекурсивного регістра зсуву та теорії ймовірностей.

Наукова новизна проведених досліджень полягає в наступному:

- розроблено метод спрощеного синтезу стійкої до помилок багатопозиційної кодової послідовності на основі ідеальних кільцевих в'язанок;
- створений ефективний алгоритм кодування та декодування інформації.

Дослідження показують, що використання багатопозиційних кодових послідовностей на основі числових дефісів у задачах перетворення інформації забезпечує більшу ефективність порівняно з кодами BZH.

Практична цінність. Дослідження різних типів багатопозиційних кодів показує переваги тих, які синтезуються на основі ідеальних кільцевих в'язанок, що дозволяють досягти більшої криптографічної міцності та стійкості до завад під час перетворення інформації порівняно з класичними багатопозиційними кодовими послідовностями.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Загальні поняття завадостійких кодів

Коректність передачі інформації контролюється за допомогою завадодостійкого кодування. Є коди виявлення помилок і коди виправлення, які також виправляють помилки. Стійкість до перешкод досягається шляхом введення резервування і додаткових бітів. У симплексних каналах зв'язку помилки усуваються за допомогою коригувальних кодів. Для дуплексних систем достатньо використовувати коди виявлення помилок. Це основні методи, які використовуються в інформаційних мережах.

Найпростішими способами виявлення помилок є перевірка контрольної суми та парності. Однак вони недостатньо надійні, особливо якщо трапляється багато помилок. Тому циклічні коди використовуються як надійні коди виявлення помилок.

Завадодостійкість - здатність системи отримувати інформацію за наявності перешкод на лінії зв’язку та спотворень у внутрішніх апаратних трактах. Завадодостійкість забезпечує достовірність і достовірність інформації (даних), що передається. В основному будемо мати справу з двійковими кодами. Двійкові (кодові) дані передаються між обчислювальними терміналами, літаками, супутниками тощо.

Передача даних у комп’ютерних системах чутлива до невеликої частки помилок, оскільки одна помилка може значно порушити обчислювальний процес.

Найчастіше помилки виникають в пристроях введення/виведення, шинах і пристроях пам'яті. У багатокористувацьких системах із спільним часом повідомлення з тривалим часом виконання розбиваються на пакети.

Повідомлення, представлені довгими послідовностями бітів, зазвичай діляться на коротші послідовності бітів, які називаються пакетами. Пакети можуть бути надіслані через мережу як незалежні об’єкти та зібрані в повідомлення в пункті призначення. Пакет з іменем і керуючими бітами на початку і в кінці називається кадром. Управління лінією передачі даних здійснюється за допомогою спеціального алгоритму, який називається протоколом. Наявність перешкод висуває додаткові вимоги до методів кодування. Для захисту інформації від перешкод необхідно вводити надлишковість у тій чи іншій формі:

- підвищення потужності сигналу;
- повторювані повідомлення;
- збільшення довжини кодових комбінацій тощо.

Збільшення потужності сигналу призводить до більш складного та дорожчого обладнання, а деякі системи передачі інформації мають обмеження в потужності передачі, наприклад супутниковий зв’язок. Для повторної передачі повідомлень потрібні буфери для зберігання інформації та зворотного зв’язку для підтвердження точності переданої інформації. У цьому випадку швидкість передачі інформації істотно знижується, до того ж цей спосіб не завжди може спрацювати. Це використовується, наприклад, у системі реального часу.

Одним з найефективніших методів підвищення надійності та достовірності передачі даних є завадодостійке кодування, яке за рахунок введення додаткової надлишковості (збільшення мінімальної кодової відстані) у кодових комбінаціях повідомлень, що передаються, дозволяє виявляти та коригувати одиничні, множинні та групові помилки. Мінімальна кодова відстань характеризує стійкість до перешкод і надмірність повідомлень. Залежно від значення мінімальної кодової відстані розрізняють коди виявлення помилок і виправлення.

Весь набір кодів ділиться на дві групи: прості і завадодостійкі (коригуючі). Коли для передачі повідомлень використовуються прості коди, використовуються всі комбінації кодів, тому навіть одну помилку неможливо виправити, що автоматично призводить до пошкодження даних. Відомо, що простий код не має практичного застосування і не зацікавить нас.

Завадостійкі коди є одним із найефективніших способів забезпечення високої точності як зберігання, так і передачі дискретної інформації. Була створена спеціальна теорія завадостійкості кодування.

Завадодостійкі коди розуміються як коди, які можуть виявляти або виявляти та виправляти помилки, спричинені перешкодами. Завадостійкість кодування забезпечується введенням надлишковості кодових комбінацій, тобто за рахунок того, що не всі символи кодових комбінацій використовуються для передачі інформації.

Як блокові, так і безперервні коди залежно від способу введення надлишковості поділяються на окремі та нерозділені. В окремих кодах правильно визначена роль окремих символів. Деякі символи є інформативними, інші перевіряються і використовуються для виявлення та виправлення помилок.

Надмірність кодової комбінації - це відношення числа перевірочних елементів (r) до інформаційних елементів (k).

$$\eta=r/k \tag{1.1}$$

Вага кодового слова - це кількість ненульових елементів у кодовому слові.

Мінімальна кодова відстань у коді дорівнює найменшій з усіх відстаней Хеммінга між різними парами кодових слів ξ:

$$d^* = \min_{\substack{c_i, c_j \in \xi \\ i \neq j}} d\left(c_i, c_j\right). \tag{1.2}$$

Відстань Хеммінга між двома q-ми послідовностями x і y довжини n — це кількість позицій, у яких вони відрізняються.

Коректуючі властивості кодів:

– число помилок, що виявляються: $\sigma=d-1$;

– число помилок, що виправляються: $t=(d-1)/2$.

Принцип виправлення помилок заснований на принципі максимальної ймовірності отриманої кодової комбінації.

Кількість керуючих елементів визначається за такою формулою:

$$r \geq \log_2(1+\sum_{i=1}^{t} C_n^i). \qquad (1.3)$$

Аналітичного вирішення цього рівняння не існує, оскільки $n=k+r$, тому вирішується підбором.

1.2. Інформаційна модель каналу зв'язку із завадами

1.2.1. Основні поняття і визначення

Інформація передається через повідомлення від джерела до одержувача по каналу зв'язку. Каналами зв'язку можуть бути телефон, радіо та ін. Коли інформація проходить по каналу, в ньому виникають збої, в результаті одержувач може отримати невірну інформацію. Для усунення цього недоліку використовуються коригувальні коди (завадодостійкі, резервні), які з високою ймовірністю гарантують витяг достовірної інформації з приймача.

Розглянемо схему передачі інформації, дійсну для будь-яких каналів зв'язку. Інформація, надана з джерела, може бути зайвою. Для усунення цього недоліку і збільшення швидкості передачі даних використовується кодуючий пристрій, який стискає інформацію за допомогою оптимальних кодів. Для виявлення або усунення помилок, що виникають при передачі, в загальну схему включають канальний кодер і декодер. Виправлення помилок виконується за

допомогою канального кодера шляхом введення надлишковості, в яку кодер записує спеціальні дані, що дозволяє пристрою декодування виконувати корекцію перешкод. Потім пристрій декодування відновлює вихідну інформацію, якщо необхідно.

Принципи побудови коригувальних кодів. Теорія несприйнятливості кодування до перешкод базується на фундаментальній теоремі Шенона. Висновки з теореми Шенона:

- для будь-якої двійкової швидкості символу, меншої за пропускну здатність каналу, існує код, для якого ймовірність неправильного декодування нескінченно мала;
- достовірність помилок не може бути настільки низькою, як бажано, якщо швидкість передачі інформації перевищує пропускну здатність каналу.

Якщо послідовність дискретних повідомлень тривалістю T безперешкодно передається по каналу зв'язку, то швидкість передачі інформації по цьому каналу зв'язку:

$$(\text{біт/сек})=\lim_{T\to\infty}(I/T).$$

Швидкість передачі інформації в основному залежить від статичних параметрів повідомлень і характеристик каналу зв'язку.

Пропускна здатність – це максимальний обсяг даних, який можна надіслати через канал зв’язку за 1 секунду.

Для досягнення найбільшого ефекту необхідно, щоб швидкість передачі інформації була максимально наближена до пропускної здатності каналу, тобто джерело має бути сумісним з каналом. Координація здійснюється через кодування.

Код - це набір комбінацій певної кількості символів, що представляють інформацію. Кожна комбінація є кодовою комбінацією.

Загальна кількість кодових комбінацій у коді зазвичай менша за кількість усіх можливих комбінацій для даного набору символів. Коди можуть бути рівномірними або нерівномірними.

В уніфікованих кодах усі композиції містять однакову кількість символів (цифр). У нерівних кодах будь-яка композиція містить різну кількість бітів (наприклад, азбука Морзе). В обчислювальній техніці в основному використовуються уніфіковані коди.

Якщо двійкові числа записуються послідовно у вигляді коду, то такий код називається простим. Цей код не може виявити помилку, оскільки будь-яке пошкодження біта призведе до появи комбінації, що міститься в коді.

У комп'ютерній техніці використовуються надлишкові коди, тобто сам код містить лише частину всіх можливих комбінацій. Ці комбінації дозволені, інші заборонені. Якщо при передачі інформації кодове слово потрапляє в заборонену зону, фіксується наявність помилки. Звичайно, дозволені комбінації повинні відрізнятися одна від одної як мінімум двома цифрами.

У цьому контексті можна виділити наступні моменти:

- чим більше мінімальна відстань коду, тим краще його здатність виявляти помилки;
- слід розрізняти поняття «мінімальна кодова відстань для пари кодових комбінацій» і «мінімальна кодова відстань для коду».

Тому можна сказати, що у випадку n-розрядної кодової комбінації тільки m біт є інформативними і стоять останніми $k=n-m$ - надлишковими (контрольними). Тому з 2^n всіх можливих комбінацій допустимі лише 2^m , а інші (2^n-2^m) - заборонені.

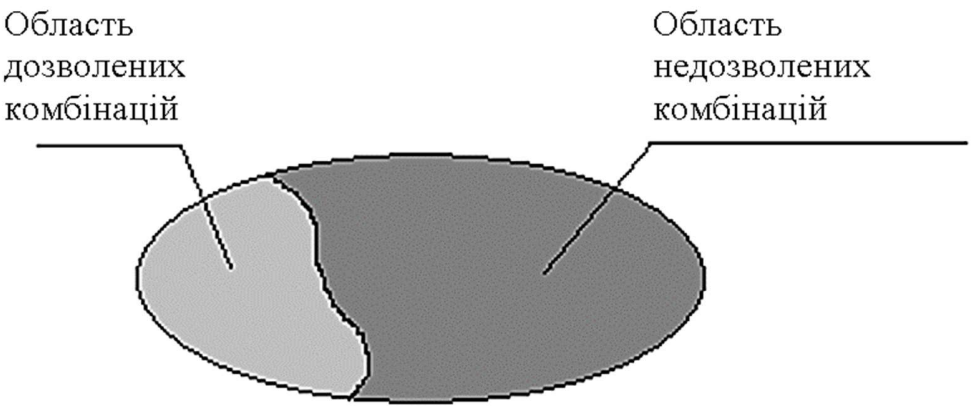


Рис. 1.1. Області дозволених та недозволених комбінацій

Якщо дозволені кодові комбінації відрізняються один від одного більш ніж на дві цифри, то в цьому випадку можна виправити більше помилок.

1.2.2. Класифікація кодів

Коригуючі коди можна розділити на 2 класи:

– блокові;

– безперервні.

У блочному кодуванні послідовність елементарних вихідних повідомлень ділиться на сегменти, і кожен сегмент пов’язується з певною послідовністю (блоком) кодових символів, яку зазвичай називають кодовою комбінацією. Сукупність усіх кодових комбінацій становить блоковий код. Кодування або декодування виконується окремо для кожної кодової комбінації. Довжина блоку може бути фіксованою або змінною, тому розрізняємо їх відповідно:

– рівномірні коди;

– нерівномірні коди.

Блокові коди бувають:

– систематичні (роздільні);

– несистематичні (нероздільні).

Систематичні коди - це коди, в яких біти можна розділити на інформаційні та керуючі, причому керуючі біти займають однакові позиції в кодових комбінаціях.

Вони позначені (n, m),

де n – довжина кодової комбінації,

m – інформаційна частина.

До несистематичних відносяться коди, категорії яких неможливо поділити на інформаційні та контрольні.

Систематичні коди поділяються на:

– лінійні;

– нелінійні.

Лінійні коди включають ті коди, в яких бітова сума за модулем 2 (у загальному випадку за модулем q) будь-яких двох кодових слів також є кодовим словом (дозволеним). Лінійні блокові коди часто називають груповими кодами.

Коди виправлення можна класифікувати за характером помилок, які вони мають виправити. Різні коди означають різні помилки або групи помилок. Всі розглянуті вище коди призначені для усунення помилок і передачі інформації.

1.2.3. Мінімальна кодова відстань

Кодова відстань d між двома комбінаціями - це кількість бітів, на які дві комбінації відрізняються одна від одної.

$$d(B_i, B_j)=\sum_{k=1}^{n}(b_{i,k}\oplus b_{j,k})$$

Під мінімальною кодовою відстанню d_{min} розуміється найменша кодова відстань що розглядається між всіма можливими парами в коді.

– $d_{min}>=t+1$ - для виявлення t помилок;

– $d_{min}>2t+1$ - для корекції t помилок;

– $d_{min}>=t+s+1$, $s<t$ - для виявлення t помилок і корекції s помилок.

Коригувальна здатність безпосередньо пов'язана з d_{min}. Це забезпечує введення надлишкових керуючих бітів, і очевидно, що їх кількість повинна бути мінімальною. Код, який має мінімальну кількість керуючих бітів, називається оптимальним.

У загальному випадку проблема пошуку оптимального коду в теорії кодування не вирішена; існують лише вирази оцінки, які можна використовувати для оцінки того, наскільки даний код близький до оптимального. Ці вирази розроблені для заданих m і n і називаються межами.

1. Границя Хеммінга

$$k>=\log_2(1+\sum_{i=1}^{d_{min}-1)/2} c^{i}_{n}),\ c^{m}_{n}=n!/m!(n-m!).$$

2. Границя Плоткіна

$$k>=2(d_{min}-1)-\log_2 d_{min.}$$

3. Границя Вартамова-Гільберта

$$k>=\log_2(1-\sum_{i=1}^{(d_{min}-1)/2} c^{i}_{n})$$

Границя Хеммінга близька до оптимальної границі для кодів з великим m/n ($m/n>=1/2$), тобто для високошвидкісних кодів, а границя Плоткіна для низькошвидкісних ($m/n<=1/3$).

Границя Вартамова-Гильберта вказує на існування лінійного коду з кодовою відстанню не менше d_{min} і з числом контрольних розрядів n, що не перевищує n-m.

Таким чином, границі 1 і 2 є необхідною умовою існування кодів, а 3 - достатньою. Рівність у виразі 3 справедливо для досконалих кодів. Ці коди виправляють всі помилки кратності $<=(d_{min}-1)/2$ і не виправляють жодної помилки більшої кратності.

1.2.4. Коригуючі коди

Циклічний код - це лінійний, систематичний, дискретний блоковий код корекції.

Коригувальний код - це код, який може виявляти або виявляти та виправляти помилки в окремих повідомленнях, які з'являються в зашумлених каналах.

Коди корекції поділяються на два класи - блокові та безперервні. До блокових кодів належать коди, в яких кожному символу алфавіту повідомлення відповідає блок (комбінація кодів), що складається з n(i) елементів, де i – номер повідомлення. Якщо $n(i)=n$, тобто довжина блоку постійна і не залежить від номера повідомлення, код називається рівномірним. Якщо довжина блоку залежить від номера повідомлення, то код блоку називається непарним. Безперервні коди являють собою безперервну послідовність бітів, і їх неможливо розділити на окремі блоки. У таких кодах елементи перевірки розташовуються в певному порядку серед елементів інформації.

Блокові коди, в свою чергу, поділяються на окремі та нероздільні. Окремі коди - це ті, в яких елементи розділені на інформаційні та перевірочні.

Окремі коди представляються як (n, k) - коди,

де n - довжина або кількість бітів коду;

k – кількість інформаційних бітів.

Серед окремих кодів розрізняють систематичні та несистематичні коди. Систематичними називають такі коди з окремими блоками, в яких керуючими бітами є лінійні комбінації інформації.

Циклічні коди забезпечують виявлення та виправлення як незалежних помилок (одиничних і множинних), так і серії помилок. Циклічні коди мають дві важливі властивості. По-перше, сума по модулю будь-яких двох дозволених

комбінацій циклічних кодів також є дозволеною комбінацією кодів. З цього випливає, що найменша кодова відстань у циклічному коді визначається найменшою вагою його комбінацій. По-друге, якщо циклічне перегрупування елементів цієї комбінації виконується в дозволеній кодовій комбінації, то результатом буде інша дозволена кодова комбінація, що належить цьому коду.

Циклічні коди належать до систематичних блокових кодів. Введемо такі позначення:

$G(x)$ - многочлен, який відображує комбінацію інформаційних елементів (степінь многочлена - k-1);

$R(x)$ - многочлен, що відображує комбінацію з r перевірочних елементів;

$F(x)$ - многочлен, що відображує кодову комбінацію в цілому (степінь многочлена - n-1=k+r-1):

$$F(x)=x^r G(x)+R(x).$$

Комбінація циклічного коду $F(x)$ створюється з вихідної комбінації інформаційних бітів $G(x)$ шляхом множення на поліном $P(x)$. Тому многочлен P називають твірним. Дозволені кодові комбінації циклічного коду характеризуються тим, що всі вони діляться без залишку на породжуючий поліном $P(x)$. Ця умова виконується, якщо $R(x)$ дорівнює залишку від ділення $x^r G(x)$ на $P(x)$.

Корекційні властивості циклічного коду полягають у розбитті дозволених кодових комбінацій $F(x)$ на поліном, що породжує $P(x)$ без залишку, і заборонених - із залишком.

Циклічні кодові комбінації створюються наступним чином:

– многочлен $G(x)$, що відображує інформаційну k-розрядну кодову комбінацію, умножається на x^r, тобто робимо зрушення k- розрядної кодової комбінації на r розрядів.

– добуток $x^rG(x)$ ділиться на утворюючий поліном $P(x)$, і отриманий залишок $R(x)$ визначає r перевірочних розрядів кодової комбінації;
– кодова комбінація $F(x)=x^rG(x)+R(x)$ є дозволеною, оскільки вона ділиться без залишку на $P(x)$.

Висновки до розділу 1

При передачі інформації по каналу зв'язку неминуче виникають помилки. Для боротьби з помилками використовується завадостійке кодування. Захист інформації від помилок досягається додаванням до інформаційних символів керуючих символів, тобто введенням надмірності. Необхідно знайти завадостійкий код з корекцією помилок, де найкраще співвідношення між його довжиною та кількістю виявляних та виправлених помилок.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Задачі створення завадостійкого перетворення інформації

Нам потрібно побудувати завадостійку систему, яка забезпечить захист у двох основних напрямках:

- захист від перешкод - система повинна мати максимально можливу здатність корекції, щоб мати можливість успішно передавати величезну кількість даних у мережі, де, як відомо, можуть виникати різні типи перешкод;
- висока надмірність - алгоритм, на основі якого будується ця система, повинен бути досить ефективним.

Вважатимемо, що будь-яка передана кодова комбінація B_i, i=1, 2, ..., N унаслідок дії перешкод може перейти в іншу кодову комбінацію B_j, j=1, 2, ..., N_0

Загальна кількість переходів з дозволеної області (від джерела) до приймача $Q=N*N$.

Серед цих переходів можна виділити 3 варіанти переходу:

- без спотворень (кількість таких переходів $N=2m$);
- кожна дозволена комбінація може перейти в інші комбінації, які дозволені $N*(N-1)=2^m(2^m-1)$. Ці помилки не можуть бути виявлені;
- дозволені комбінації переходять в заборонені і можуть бути виявлені $N*(N_0-N)$.

Отже, виявляюча здатність може бути визначена як:

$$N*(N_0-N)/N*N_0=1-N/N_0=1-2^m/2^n=1-1/2^K$$

Щоб виправити помилки, розділіть множину всіх заборонених комбінацій на M взаємно непересічних підмножин і призначте кожній такій підмножині одну

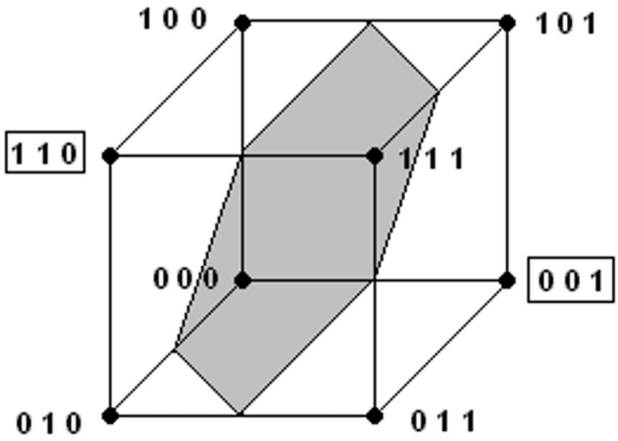
дозволену комбінацію B_i. Якщо при передачі інформації спотворене слово потрапляє в підмножину N_i, то правильним вважається слово B_i.

Регульована ємність дорівнює відношенню всіх комбінацій, які можна налаштувати, до всіх, які можна виявити:

$$(N_0\text{-}N)/N*(N_0\text{-}N)=1/N=1/2^m$$

Розглянемо наступний приклад, який покаже, як можна виправити одну помилку при передачі інформаційної частини числа, що складається з однієї цифри.

$m=a_1=\{0, 1\}$. Введемо додаткові розряди a_2 і a_3.



a_1	a_2	a_3	
1	1	0	дозвол.
0	1	0	недозвол.
1	0	0	
1	1	1	
0	0	1	дозвол.
1	0	1	недозвол.
0	1	1	
0	0	0	

Рис. 2.1. Виправлення одної помилки при передачі інформаційної частини числа

Як бачимо, спотворення однієї цифри призводить до того, що спотворене слово потрапляє в заборонену область, що відповідає заданому дозволеному числу, і його дозволена комбінація буде відновлена.

2.2. Синтез кодів з врахуванням завад

Введемо поняття вектора (послідовності) помилок.

1011010 - спотворене слово

⊕ 1001110 - неспотворене слово
0010100 - послідовність помилок

Кожне спотворене чисте слово теоретично можна представити як суму за модулем 2 двох послідовностей даних - неспотвореного слова та послідовності помилок. Послідовність помилок містить усі нулі, крім бітів, у яких виникає помилка, і не має значення, який це тип помилки (від 1 до 0 чи від 0 до 1).

Приклад: нехай код складається з дозволених комбінацій B_1=001, B_2=011, B_3=110.

Таблиця 2.1

Код з дозволених комбінацій B_1=001, B_2=011, B_3=110

Вектор помилок	Кодова комбінація			Кратність помилок Q
	B_1=001	B_2=011	B_3=110	
000	001	011	110	0
001 010 100	001 (011) 101	010 (001) 111	111 100 010	1
011 101 110	010 100 111	000 (110) 101	101 (011) 000	2
111	(110)	100	(001)	3

Як видно з табл. 2.1, з 21 можливої помилки 6 не виявлено (наведені в дужках).

Якщо передача повідомлень однакова, а помилки, які виникають, є незалежними, то поодинокі помилки набагато більш імовірні, ніж інші множинні помилки. Цей зв'язок зазвичай експоненціальний. Якщо немає спеціальних

інструкцій при побудові кодів, код спочатку створюється для виявлення та виправлення окремих помилок.

Розглянемо можливість побудови підмножин для виправлення однієї помилки:

1) $M_1=\{000, 101\}$ $M_2=\{010, 111\}$ $M_3=\{100\}$

2) $M_1=\{000, 101\}$ $M_2=\{010\}$ $M_3=\{100, 111\}$

3) $M_1=\{000, 101\}$ $M_2=\{111\}$ $M_3=\{100, 010\}$

2.3. Коригуючі властивості завадостійких кодів

Коригувальна здатність — здатність кодів виявляти та виправляти помилки.

Основні функції включають:

– n – довжина коду;

– m – інформаційна частина числа;

– k – контрольні розряди;

– M – підстава коду (для двійкових: 2);

– $N=2^m$ – потужність коду (кількість дозволених комбінацій);

– N_0 - загальна кількість всіх комбінацій;

– W – вага кодової комбінації;

– d_{min} – мінімальна кодова відстань;

– $\lambda=k/n$ – надмірність коду.

Коригуюча здатність коду залежить від надмірності та розташування керуючих бітів.

2.4. Інші класи кодів

Крім циклічних кодів на практиці використовуються інші типи кодів з різними властивостями.

Серед циклічних кодів особливе значення має клас кодів, запропонований Боузом і Рой-Чоудхурі та незалежно Хоквенгамом. Коди Бозе-Чоудхурі-Хокквенгема (скорочено BCH) відрізняються відносно простою процедурою декодування.

Процедура мажоритарного декодування відносно проста і застосовується до певного класу двійкових лінійних кодів, включаючи циклічні коди. Це пояснюється тим, що в цих кодах кожен інформаційний символ може бути виражений кількома способами за допомогою інших символів кодової комбінації.

Потужні коди (тобто коди з довгими блоками та великою кодовою відстанню d) з відносно простою процедурою декодування можуть бути створені шляхом комбінування кількох коротких кодів. Так будується, наприклад, ітеративний код з двох лінійних систематичних кодів (n_1, k_1) і (n_2, k_2). Мінімальна кодова відстань для двовимірного ітеративного коду $d=d_1d_2$, де d_1 и d_2 - відповідно мінімальні кодові відстані для кодів 1-го і 2-го рівнів.

Каскадний код схожий на ітеративний код, але між ними є істотна різниця. Перший рівень кодування в зв'язаному коді - це систематичний лінійний двійковий код (внутрішній код), кожна комбінація якого розглядається як один символ недвійкового коду другого рівня (зовнішній код). При отриманні спочатку декодуються всі рядки (блоки) внутрішнього коду (з виявленням або виправленням помилок), потім декодується m-й блок зовнішнього коду та виправляються помилки та стирання, що залишилися після декодування внутрішнього коду. Зовнішнім кодом зазвичай є m-й код Ріда-Соломона, який є

підкласом кодів BCH і забезпечує найбільший можливий d для даного n_2 і k_2, якщо $n_2<m$. Каскадні коди у багатьох випадках найбільш перспективні серед відомих блокових завадостійких кодів.

2.5. Метод перестановок

Для каналів із групуванням помилок часто використовується перестановка символів або декорування помилок. Це означає, що символи, що містяться в одній кодовій комбінації, не надсилаються безпосередньо один за одним, а перемежовуються символами інших кодових комбінацій.

Якщо проміжок між символами, включеними в одну комбінацію, перевищує максимально можливу довжину групи помилок, помилки не будуть згруповані в комбінації. Групу помилок буде розділено як окремі помилки в комбіновану групу. Поодинокі помилки будуть легко виявлені (виправлені) декодером.

Висновки до розділу 2

Необхідно використати таку математичну модель завадостійкого кодування з корекцією помилок, при якій виявляється та знаходиться найбільша кількість помилок в залежності від довжини кодової послідовності.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Завадостійке кодування на основі числових в’язанок

В'язанка дозволяє забезпечити досить високу корекційну здатність такої системи завдяки її унікальним комбінаторним властивостям, а несхожість процесу кодування-декодування порівняно з існуючими криптографічними методами з використанням властивостей поліномів робить захист ще сильнішим.

Як відомо, конструкція завадостійких кодів базується на принципі введення надлишковості, що дозволяє ідентифікувати та виправляти помилки шляхом пред'явлення додаткових вимог до сигналів, що передаються, з подальшою їх перевіркою. Серед великої різноманітності таких кодів особливе місце займають циклічні коди, широке використання яких на практиці зумовлено їх значною ефективністю у виявленні та виправленні помилок, а також відносно простою реалізацією пристроїв кодування та декодування із застосуванням зміщення регістри та напівсуматори.

3.2. Параметри коду на основі числових в’язанок

Кожна з S_n $(S_n - 1)/2$ різних пар кодових комбінацій містить точно R із n одиничних символів в розрядах, що є властивостею ідеальної кільцевої в’язанки (ІКВ). Іші $n - R$ символів відрізняються від символів, що містяться в однойменних розрядах. Тоді мінімальна кодова відстань для даного коду представляється як:

$$d_{\min} = 2(n - R) \tag{3.1}$$

Число помилок, що виявляються t_d, та число помилок, що виправляються t_c за допомогою цього коду, позв'язані з мінімальною кодовою відстанню і мають вигляд:

$$t_d \le d_{\min} - 1 \tag{3.2}$$

$$t_c \le \frac{d_{\min} - 1}{2} \tag{3.3}$$

Давайте розглянемо можливість встановлення зв'язку між n і R в якому розглянутий код отримує додаткові переваги. Як видно з формул (3.1), (3.2) і (3.3), корекційна здатність коду зростає зі збільшенням мінімальної кодової відстані, а фактично завдання полягає в забезпеченні максимально можливої кодової мінімальної відстані. $d_{\min}$ при найбільш меншому значенні S_n.

Коректувальна здатність коду зростає зі збільшенням співвідношення $\sigma = n - R$. Підставив $R = \sigma - n$ в формулу (3.1) отримуємо наступне рівняння відносно σ:

$$\sigma = n - \frac{n \times (n-1)}{S_n - 1} \tag{3.4}$$

Далі отримуємо:

$$S_n = 2n \tag{3.5}$$

Після підстановки та розв'язування цього рівняння в цілих значеннях знайдемо оптимальне співвідношення між n і R, при якому код виявляє та виправляє максимальну кількість помилок:

$$R = \begin{cases} \frac{n}{2}, \_для\_парних\_значень\_n \\ \frac{n-1}{2}, \_для\_непарних\_значень\_n \end{cases} \tag{3.6}$$

Знайдене співвідношення (3.6) надзвичайно важливе для розв'язання задачі побудови оптимального циклічного ІКВ-коду, який, по суті, зводиться до вибору ІКВ із необхідними параметрами.

Для ілюстрації розглянемо приклад розрахунку корекційної здатності завадостійких кодів, побудованих на основі ІКВ з параметрами:

1) $n = 6,\ R = 1$;

2) $n=15,\ R=(n-1)/2$;

3) $n=16,\ R=n/2$.

Довжина дозволених кодових комбінацій для всіх трьох кодів однакова: $S_n=31$. Максимальна кількість помилок, які можна виявити або виправити за допомогою першого з цих кодів, становить і, відповідно, з кожним із двох інших кодів, які дозволяють виявити та виправити помилки.

В принципі, будь-який ІКВ може стати основою для синтезу відмовостійкого коду. Однак слід зазначити, що використовуючи ІКВ, параметри яких пов'язані з цим співвідношенням, можна побудувати найбільш ефективні перешкодостійкі коди.

Для ІКВ (1, 1, 1, 2, 2, 1, 3, 4), визначимо кодову мінімальну відстань та відповідно коректувальну здатність коду:

$$d_{\min}=2\cdot(8-4)=8$$
$$t_d=8-1=7$$
$$t_c=\frac{8-1}{2}=3$$

3.3. Порівняння завадостійкого кодування на ІКВ та БЧХ

Завадостійке кодування на основі ІКВ дозволяє кодувати біти масиву даних у послідовність довжин і в той же час коригує положення послідовності, тоді як завадостійке кодування спирається на коди БЧХ тієї ж послідовності. довжина та кількість інформаційних бітів дозволяє редагувати позицію.

Таким чином, здатність корекції БЧХ демонструє найкращі результати, хоча загалом кількість бітів на блок довжини біта не є критичною, оскільки для більшості мережевих програм поріг здатності корекції становить. Представимо ці дані в табл. 3.1.

Таблиця 3.1

Порівняння завадостійкого кодування на ІКВ та БЧХ

Показник	Числове значення (бали)	
	Завадостійке кодування на основі ІКВ	Завадостійке кодування на основі БЧХ
Надлишковість коду	11	11
Коректувальна здатність	3	4
Потужність коду	16	16
Швидкість кодування даних, МБ/с	3,01	2,02
Швидкість декодування даних, МБ/с	0,56	1,21
Ресурсозатратність системи кодування, МБ	60	100

Виконується кодування на основі БЧХ під час кодування та декодування файлу розміром 10 МБ $100МБ$ пам'яті комп'ютера, в той час як на основі ІКВ – лише $60МБ$, що показує меншу ресурсозатратність завадостійкого кодування на основі ІКВ.

3.4. Кодові послідовності на числових в'язанках

Кодова послідовність - це набір нулів і одиниць (токенів), за допомогою яких кодується один символ (біт) для подальшої передачі. Сам код є послідовністю послідовностей коду. Основними властивостями кодової послідовності є дуже хороші властивості автокореляції та крос-кореляції. Стійкість системи до перешкод залежить від кодової послідовності. Враховуючи однакові довжини, властивості послідовності можуть різко відрізнятися. Завадодостійкість залежить від довжини кодових послідовностей і їх

характеристик, і перш за все від їх взаємних властивостей. Тому вибір оптимального набору кодів зводиться до знаходження такої структури кодових послідовностей.

Як наслідок, традиційні методи кодування інформації та обробки кодів, а також методи з використанням класичних комбінаторних конфігурацій не завжди дозволяють розкрити всі можливості систем кодування. Тому важливою та актуальною проблемою є дослідження та використання нових, ефективних моделей удосконалення систем кодування інформації за такими показниками, як швидкість передачі повідомлень, стійкість коду до перешкод, легкість виявлення та виправлення помилок. Такі моделі включають комбінаторні конструкції на основі ІКВ, елементи яких можна впорядкувати за цілими числами або цілими кортежами, і чиї комбінаторні властивості визначаються як значеннями їхніх елементів, так і значеннями сум будь-якої кількості цих елементів, приймаючи з урахуванням їх порядку.

При побудові кодування інформації на основі числових сполучників, як одного з видів комбінаторної конструкції, встановлюються певні співвідношення між елементами. Велику групу оптимальних комбінаторних кодів зі спеціальними властивостями складають надлишкові двійкові коди з високою стійкістю до різного роду перешкод, коди швидкого перетворення інформації, коди з можливістю реалізації секретного кодування та ін. Оптимальні комбінаторні системи створюють послідовності кодів зі спеціальними властивостями. , наприклад, послідовності на числових сполученнях з хорошими кореляційними властивостями.

Конструкція кодів, що використовуються для кодування, заснована на принципі обмеження або виключення декодування інформації.

Під кодами на числових послідовностях розуміють коди, дозволені комбінації яких створюють послідовності з різною кількістю нулів або одиниць, що зустрічаються певну кількість разів. Конфігурація таких послідовностей в принципі може бути будь-якою, наприклад вона може мати форму лінії, розгалуженої системи взаємопов'язаних ліній, кільця і т. д. Таким чином, можна говорити про лінію, кільце, дерево та ін. конфігурації кодових послідовностей на числових рядках.

Кодові послідовності в числових рядках мають багато переваг перед іншими послідовностями. Одним із них є легкість виявлення та виправлення помилок на приймальній стороні, оскільки поява принаймні одного символу «1» або символу «0», як втручання в отриману послідовність коду на числових рядках, сигналізує про помилку шляхом зміни кількість різних дозволених проб. Помилка не виявляється тільки в тих випадках, коли кількість помилкових спотворень дорівнює або перевищує кодову відстань. Якщо в кодовій послідовності на числових рядках з’являються хибні символи, вони відразу виявляються всі або частина, що забезпечує високу завадостійкість кодової послідовності на ідеальних кільцевих в’язанках.

3.5. Кодова послідовність на основі ідеальних кільцевих в’язанок

Проблема поліпшення якісних характеристик кодових послідовностей на числових послідовностях може бути вирішена за допомогою використання ідеальних кільцевих в’язанок (ІКВ).

Кодова послідовність на основі ІКВ є прикладом практичної реалізації однієї з головних переваг будь-якого рядка символів: генерування різних числових сум для наступних елементів даного рядка. Надалі ми будемо називати цю властивість сполучників і відповідних комбінаторних систем зі структурою принципом зв’язування.

У загальному випадку будь-який шаблон розподілу ваги бітів може відповідати різним позиціям у кодовій послідовності. Однак найбільший інтерес представляє реалізація кільцевих кодових послідовностей за допомогою ідеальних кільцевих в’язанок. Одновимірна ідеальна числова послідовність - це послідовність, в якій множина всіх породжених нею чисел вичерпує значення, пропорційні елементам натурального ряду при заданій кількості повторень кожного елемента. Так само по відношенню до одновимірної ідеальної послідовності визначається поняття багатовимірної числової послідовності, в якій множина всіх породжених нею векторів вичерпує зміст відповідних упорядкованих числових множин.

Концепція розшарування дозволяє розглядати проблеми різних постановок, пов'язані з побудовою оптимальних комбінаторних моделей. Такі моделі зручні для вивчення та вивчення властивостей і структурної організації об'єктів будь-якого типу. Числові послідовності особливо цікаві. Тому надалі ми зосередимося на найпростіших числових в’язанках – ідеальних кільцях.

Ідеальною кільцевою в’язанкою називається послідовність $K_N=(k_1,k_2,...,k_i,...,k_N)$ чисел, на якій всі можливі кільцеві суми вичерпують значення чисел натурального ряду 1,2,...,$S_N=N^2-(N-1)$.

Враховуючи визначення можна побудувати таблицю кільцевих сум ваг розрядів на ІКВ кодової послідовності після розміщення по її діагоналі чисел послідовності K_N.

Загальна кількість кільцевих сум ваг розрядів кодової послідовності на ІКВ, що мають різні значення:

$$S_N=N^2-(N-1). \qquad (3.7)$$

При $p_j=1,q_j=N$, а також при $p_j\neq1$, $q_j=p_j-1$ кільцева сума ваг розрядів кодової послідовності на основі ІКВ дорівнює S_N. Далі ми будемо називати кільцеві суми

бітових ваг, що відповідають таким кодовим послідовностям у ІКВ, залежними, а кільцеві суми, що відповідають решті кодів, незалежними.

Таблиця 3.2

Кільцеві суми кодової послідовності на основі ІКВ довжини SN

p_j	q_j					
	1	2	l-1	l	N-1	N
1	k_1	$\sum_{i=1}^{2} k_i$	$\sum_{i=1}^{l-1} k_i$	$\sum_{i=1}^{l} k_i$	$\sum_{i=1}^{N-1} k_i$	$\sum_{i=1}^{N} k_i$
2	$\sum_{i=1}^{N} k_i$	k_2	$\sum_{i=2}^{l-1} k_i$	$\sum_{i=2}^{l} k_i$	$\sum_{i=2}^{N-1} k_i$	$\sum_{i=2}^{N} k_i$
l-1	$\sum_{i=l-1}^{N} k_i + \sum_{i=1}^{1} k_i$	$\sum_{i=l-1}^{N} k_i + \sum_{i=1}^{2} k_i$	k_{l-1}	$\sum_{i=l-1}^{l} k_i$	$\sum_{i=l-1}^{N-1} k_i$	$\sum_{i=l-1}^{N} k_i$
l	$\sum_{i=1}^{N} k_i + \sum_{i=1}^{1} k_i$	$\sum_{i=1}^{N} k_i + \sum_{i=1}^{2} k_i$	$\sum_{i=1}^{N} k_i$	k_l	$\sum_{i=l}^{N-1} k_i$	$\sum_{i=l}^{N} k_i$
N-1	$\sum_{i=n-1}^{N} k_i + \sum_{i=1}^{1} k_i$	$\sum_{i=n-1}^{N} k_i + \sum_{i=1}^{2} k_i$	$\sum_{i=n-1}^{N} k_i + \sum_{i=1}^{l-1} k_i$	$\sum_{i=n-1}^{N} k_i + \sum_{i=1}^{l} k_i$	k_{N-1}	$\sum_{i=N-1}^{N} k_i$
N	$\sum_{i=n}^{N} k_i + \sum_{i=1}^{1} k_i$	$\sum_{i=n}^{N} k_i + \sum_{i=1}^{2} k_i$	$\sum_{i=n}^{N} k_i + \sum_{i=1}^{l-1} k_i$	$\sum_{i=n}^{N} k_i + \sum_{i=1}^{l} k_i$	$\sum_{i=1}^{N} k_i$	k_N

Число S_N^* незалежних кільцевих сум на n-послідовності ваг розрядів визначається залежністю:

$$S_N^* = N(N-1) \tag{3.8}$$

3.6. Завадостійка кодова послідовність на основі ідеальної кільцевої в'язанки

Розглянемо кодову послідовність, якому присвоєні ваги ІКВ 8-го ($N=8$) порядку кратності $(R=4)$: 1, 1, 1, 2, 2, 1, 3, 4. Оскільки значення ваг — це числа ІКВ 8-го порядку 4-ої кратності, кожне число натурального ряду від 1 до $N(N—1)/R$=15 можна отримати чотирьома способами, а число всіх способів дорівнює кількості одержаних чисел.

При заданій кількості N розрядів кодової послідовності система дає змогу кодувати будь-які числа від 1 до $S_N=N(N—1)/R+1$.

Кодові комбінації 10000000, 01000000, 00100000, 00000100 відповідають 4-м способам кодування числа 1. Кодуванню числа 2 відповідають 11000000, 01100000, 00010000, 00001000. Числу 3 відповідають 11100000, 00110000, 00001100, 00000010, числу 4 — 01110000, 00011000, 00000110, 00000001 і т. д., числу 14 — 01111111, 10111111, 11011111, 11111011, числу 15 — 11111111. Реалізовану кодову послідовність з вагами ІКВ ілюструє табл. 3.3.

Для кожної числової комбінації кодової послідовності існує набір пакетів одиниць і нулів, побудованих відповідно до вагових коефіцієнтів ІКВ за таким правилом: 1 — 1, 2 — 10, 3 — 100, 4 — 1000 і так далі.

Таблиця 3.3

Кодова послідовність з вагами ІКВ

8-го порядку 4-ої кратності: 1, 1, 1, 2, 2, 1, 3, 4

1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
0	1	1	1	1	0	1	0	1	1	0	0	1	0	0
0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
0	1	0	0	0	1	1	1	1	0	1	0	1	1	0
0	0	1	0	0	0	1	1	1	1	0	1	0	1	1
1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
1	1	0	0	1	0	0	0	1	1	1	1	0	1	0
0	1	1	0	0	1	0	0	0	1	1	1	1	0	1
1	0	1	1	0	0	1	0	0	0	1	1	1	1	0
0	1	0	1	1	0	0	1	0	0	0	1	1	1	1
1	0	1	0	1	1	0	0	1	0	0	0	1	1	1
1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
1	1	1	0	1	0	1	1	0	0	1	0	0	0	1

Кількість різних $S_N(S_N\text{-}1)/2$ кодові комбінації містять рівно R з N окремих символів у бітах з однаковим іменем, що випливає з властивості ІКВ. Решта $N\text{-}R$ символів однієї і тієї ж кількості інших кодових послідовностей на основі ІКВ відрізняються від символів, що містяться в однойменних бітах.

Тому мінімальна кодова відстань для будь-якої кодової послідовності на основі ІКВ визначається як наступне співвідношення між порядком і кратністю ІКВ:

$$d_{\min} = 2(N\text{-}R) \tag{3.9}$$

Щоб збільшити потужність кодової послідовності на основі ІКВ, ми побудуємо дзеркальну кодову послідовність із ваговими коефіцієнтами ІКВ, у якій ми міняємо місцями одиниці та нулі під час кодування.

Кодові комбінації 01111111, 10111111, 11011111, 11111011 відповідають 4-м способам кодування числа 1. Кодуванню числа 2 відповідають 00111111, 10011111, 11101111, 11110111. Числу 3 відповідають 00011111, 11001111, 11110011, 11111101, числу 4 — 10001111, 11100111, 11111001, 11111110 і т. д., числу 14 — 10000000, 01000000, 00100000, 00000100, числу 15 — 00000000.

Реалізовану дзеркальну кодову послідовність з вагами ІКВ ілюструє табл. 3.4.

Таблиця 3.4

Дзеркальна кодова послідовність з вагами ІКВ

8-го порядку 4-ої кратності: 1, 1, 1, 2, 2, 1, 3, 4

0	0	0	0	1	0	1	0	0	1	1	0	1	1	1
1	0	0	0	0	1	0	1	0	0	1	1	0	1	1
1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	1	1	0	0	0	0	1	0	1	0	0	1	1	0
0	1	1	1	0	0	0	0	1	0	1	0	0	1	1
1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
1	1	0	1	1	1	0	0	0	0	1	0	1	0	0
0	1	1	0	1	1	1	0	0	0	0	1	0	1	0
0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
1	0	0	1	1	0	1	1	1	0	0	0	0	1	0
0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
1	0	1	0	0	1	1	0	1	1	1	0	0	0	0
0	1	0	1	0	0	1	1	0	1	1	1	0	0	0
0	0	1	0	1	0	0	1	1	0	1	1	1	0	0
0	0	0	1	0	1	0	0	1	1	0	1	1	1	0

Потужність основної та дзеркальної кодових послідовностей на основі ІКВ:

$$P = 2S_N^R \tag{3.10}$$

Число помилок, які можна виявити t_1, за допомогою кодової послідовності на основі ІКВ, визначається мінімальною кодовою відстанню d_{min}:

$$t_1 \leq d_{min}-1. \tag{3.11}$$

Число помилок, що виправляються t_2 за допомогою кодової послідовності на основі ІКВ, визначається числом помилок, які можна виявити t_1:

$$t_2 \leq (t_1-1)/2 \tag{3.12}$$

Залежність для визначення кількості помилок, які можуть бути виявлені t_1 кодовою послідовністю на основі ІКВ:

$$t_1 \leq 2(N-R)-1. \tag{3.13}$$

Залежність для визначення кількості помилок, які можуть бути виправлені t_2 кодовою послідовністю на основі ІКВ:

$$t_2 \leq N\text{-}R\text{-}1 \tag{3.14}$$

Кодова відстань для кодової послідовності на основі ІКВ знаходиться як

$$d_{1,2} = S_N\text{-}2(N\text{-}R) \tag{3.15}$$

Формули для визначення числа помилок, які підлягають виявленню за допомогою кодової послідовності на основі ІКВ:

$$t_1 \leq 2(N-R)-1, \text{ якщо } S_N \geq 4(N\text{-}R) \tag{3.16};$$

$$t_1 \leq S_N - 2(N-R)-1, \text{ якщо } S_N < 4(N\text{-}R) \tag{3.17};$$

Формули для визначення числа помилок, які підлягають виправленню за допомогою кодової послідовності на основі ІКВ:

$$t_2 \leq N-R-1, \text{ якщо } S_N \geq 4(N\text{-}R) \tag{3.18};$$

$$t_2 \leq \frac{S_N - 2(N-R+1)}{2}, \text{ якщо } S_N < 4(N\text{-}R) \tag{3.19};$$

Розглянемо взаємозв'язок значень параметрів N і R. При яких співвідношеннях N і R існує оптимальна коригуюча здатність кодової послідовності на основі ІКВ. Завадостійкість цієї кодової послідовності зростає зі збільшенням різниці $L=N\text{-}R$. Максимальне значення L досягається за умови:

$$S_N = 2N. \tag{3.20}$$

Визначимо співвідношення між параметрами N і R, коли кодова послідовність на основі ІКВ набуває здатності виявляти та виправляти максимально можливу кількість помилок:

$$L = \begin{cases} N/2, & N - парне \\ (N-1)/2, & N - непарне \end{cases} \tag{3.21}$$

Синтезовані на основі ІКВ кодові послідовності дають змогу виявляти до $N\text{-}1$ або виправляти до $N/2\text{-}1$ помилок для парних, і виявляти до N або виправляти

до *(N-1)/2* помилок для непарних значень *N* при умові, що потужність кодових послідовностей на основі ІКВ зростає вдвічі за рахунок додаткового використання дзеркальної кодової послідовності на основі ІКВ.

3.7. Дослідження архівації завадостійкої послідовності на основі ІКВ

Завадостійке кодування на основі ІКВ демонструє хорошу ефективність корекції, але розмір закодованого файлу в 4 рази більший за вихідний розмір файлу. Це може бути проблемою для великих файлів, які потрібно передати через мережу з низькою пропускною здатністю.

Вивчаємо ефективність використання вбудованих рішень у Windows 10 архіватору на файлах малого розміру(до $1Мб$), середнього розміру ($2-15Мб$) та великого розміру (більше $15Мб$) шляхом архівація обох файлів та порівняння їхнього розміру.

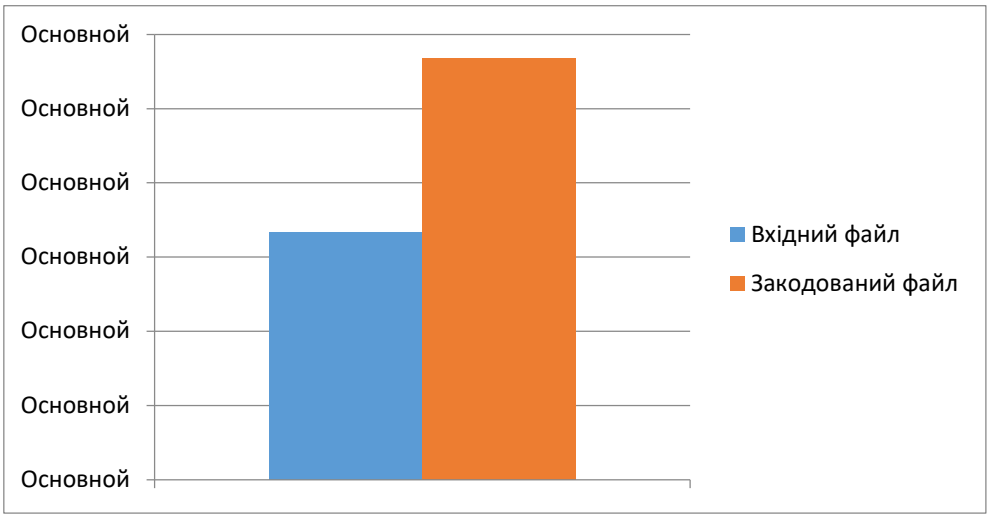


Рис. 4.1. Статистика для файлу малого розміру–$33,3Кб$, *.doc*

Таким чином, ми отримуємо, що розмір закодованого файлу більший за оригінальний файл на $\frac{56,8-33,3}{56.8}\times 100\% = 41,32\%$. Прямо скажемо, результат є незадовільним, але спробуймо провести експерименти і з іншими файлами.

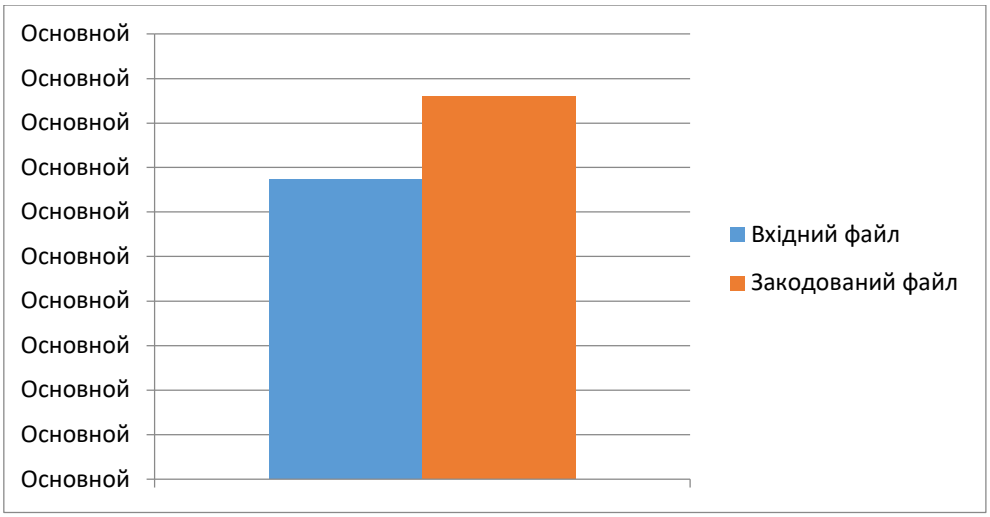


Рис. 4.2. Статистика для файлу середнього розміру– 3,37*Мб*, *.doc*

Вже візуально помітно, що цього разу отримали кращі результати в порівняні з попереднім дослідженням: $\frac{4,3-3,37}{4,3}\times 100\% = 21,63\%$.

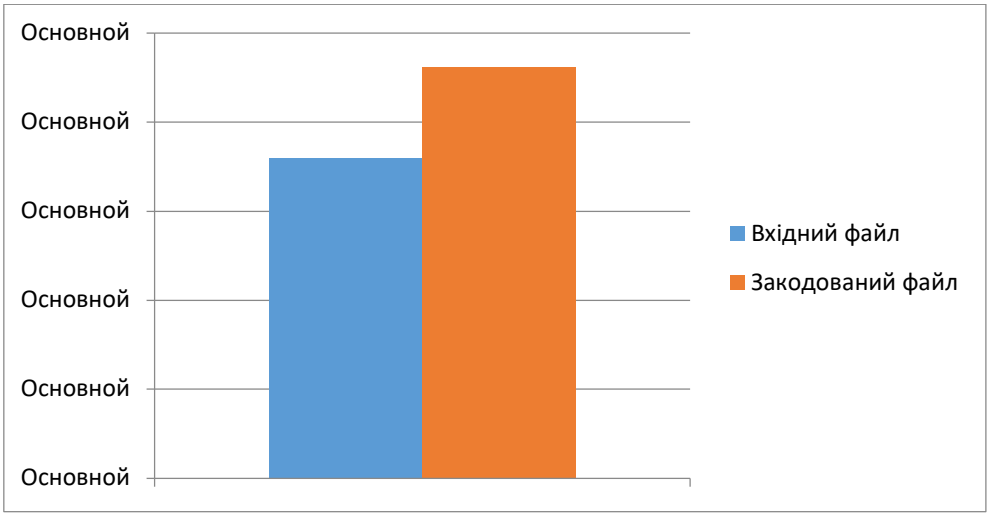


Рис. 4.3. Статистика для файлу великого розміру – 18,0*Мб*, *.doc*

Результати третього експерименту практично збігаються з попередніми, а саме: $\frac{23,1-18.0}{23,1}\times 100\% = 22,07\%$. Звідси напрошується висновок, що вбудований

архіватор добре стискає дані формату *.doc, за рахунок чого й отримали добре стиснення.

Давайте перевіримо, як це працює з іншими типами файлів на прикладі архівації для зашифрованого формату файлу *.bmp.

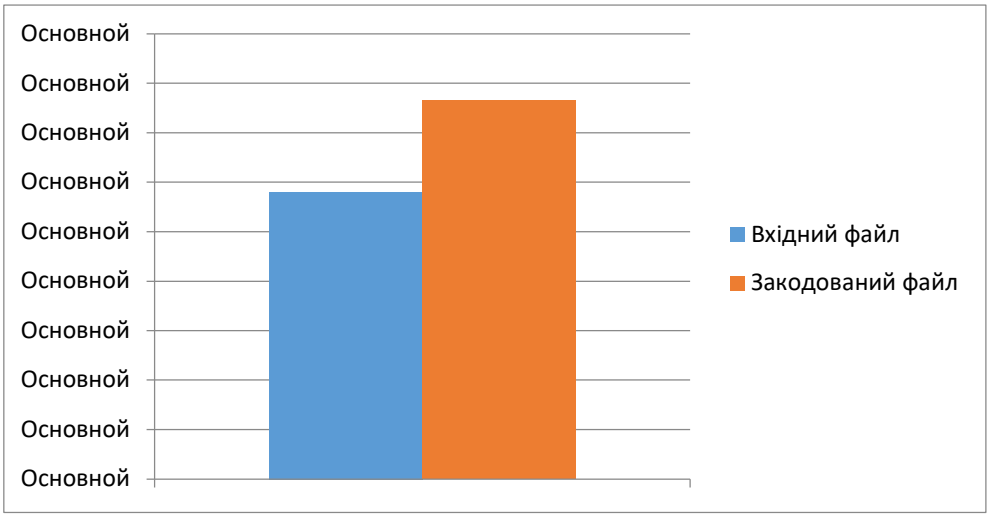


Рис. 4.4. Статистика для файлу малого розміру - 11,6*Кб*, *.bmp*

Отриманий результат фактично повторив та підтвердив результати трьох попередніх досліджень на рис. 4.1-4.3: $\frac{15,3-11.6}{15,3}\times 100\% = 24,18\%$

Через досить значну надмірність рекомендується стискати закодовані файли за допомогою вбудованого архіватора в Windows 10, тому що навіть якщо закодований файл більший за вхідний файл, то він здатний виправити кодову послідовність на основі ІКВ.

Висновки до розділу 3

На основі математичної моделі ідеальних кільцевих в’язанок побудований завадочтійкий код з корекцією помилок, що дозволяє знаходити до 50% помилок та виправляти 25% помилок від довжини кодової послідовності.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Обґрунтування вибору мови і системи програмування

Для виконання магістерської роботи була вибрана мова програмування Borland Delphi 7 яка не потребує потужного комп'ютера і має достатньо можливостей для вирішення поставленого завдання.

Так, як програма написана на Borland Delphi 7, а отже є повноцінною 32-х розрядною Windows програмою і коректно працює від Windows 7 до Windows 10, 11. Отже програма працює під найпоширеніші платформи для домашніх та офісних систем. На даний момент готова робоча версія, помилок під час тестування не виявлено.

Програма проста у використанні та інтуїтивно зрозуміла, особливо для користувачів, які мають хоча б мінімальні навички роботи з операційною системою Microsoft Windows будь-якої версії.

Основна група елементів форми:

- ідеальна кільцева в'язанка порядку N кратності R (рис. 4.1);
- кількість помилок, які необхідно буде виправити (не більше числа, вказаного у вікні Info після відкриття файлу (рис. 4.2 - 4.5);
- кнопка «Відкрити файл» дозволяє вибрати файл для кодування, генерувати випадкові помилки у вхідній інформації, шукати та виправляти помилки різного розміру у вхідній інформації на основі числових рядків.

Щоб почати процес кодування/декодування, натисніть кнопку <Відкрити файл>. Результати кодування та декодування з корекцією помилок кратності від однієї до п'яти за допомогою кодових послідовностей на основі ІКВ наведено на рис. 4.2 – рис. 4.5.

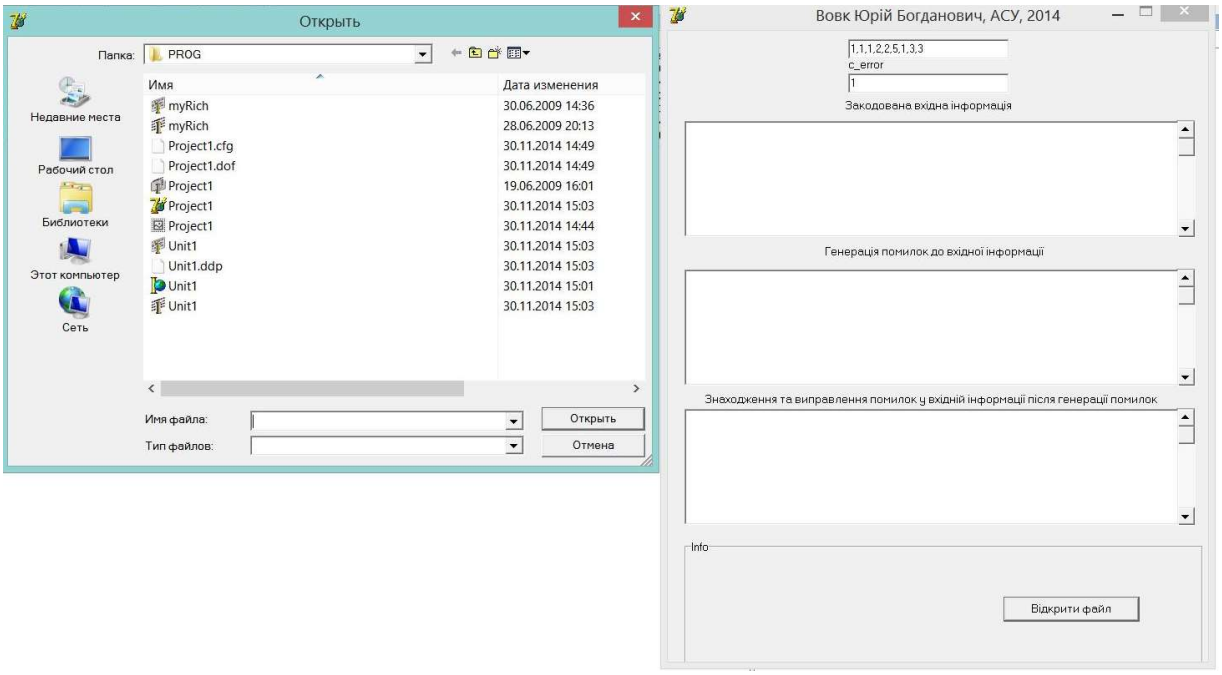


Рис. 4.1. Відкриття файлу для кодування, створення помилок у вихідних даних, пошук і виправлення кратних помилок у вхідних даних на основі ІКВ

Рис. 4.2. Кодування вхідної інформації, генерування помилок введення, пошук і виправлення помилок кратності одиниць у вхідній інформації

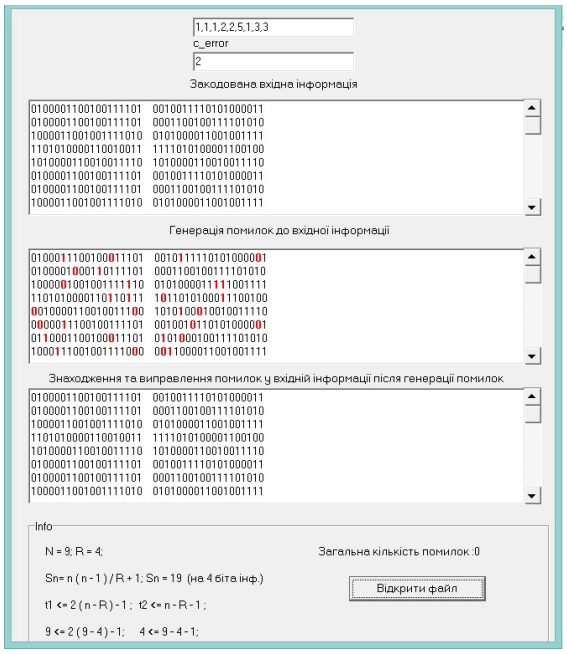


Рис. 4.3. Кодування вхідної інформації, генерування помилок введення, пошук і виправлення помилок кратності два у вхідній інформації

Рис. 4.4. Кодування вхідної інформації, генерація помилок у вхідній інформації, пошук і виправлення помилок кратності у вхідній інформації

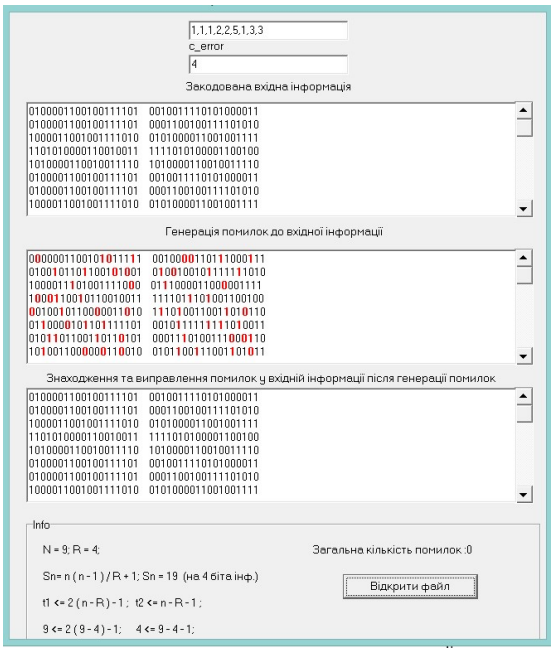


Рис. 4.5. Кодування вхідної інформації, генерація помилок у вхідній інформації, пошук і виправлення помилок кратності у вхідній інформації

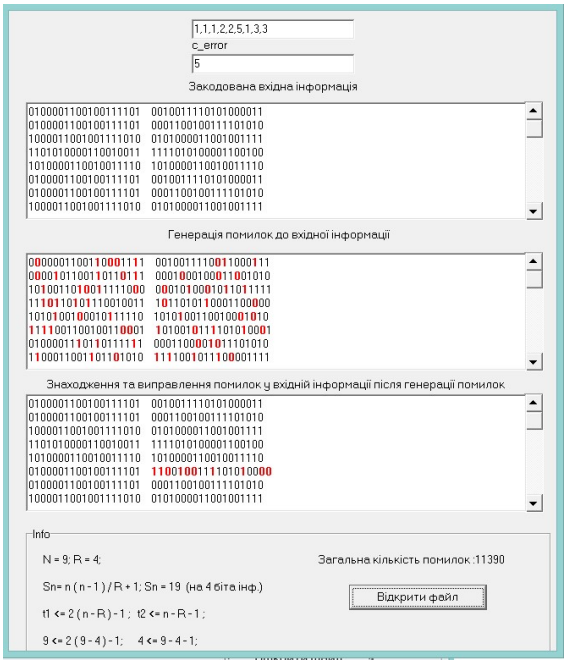


Рис. 4.6. Кодування вхідної інформації, генерація помилок у вхідній інформації, пошук і виправлення п'ятикратних помилок у вхідній інформації

Як видно на скріншотах вище, виправлення помилок виконується лише на рис. 4.2-4.5, що повністю підтверджує формули 3.12-3.13, тоді як на рис. 4.6 є 11390 невиправлених помилок у введеній інформації.

4.2. Вибір апаратних рішень

При побудові кодової послідовності на основі ІКВ найбільш критичним елементом комп'ютера є швидкість процесорів, тобто кількість операцій за одиницю часу; менш критичними елементами є кількість і швидкість оперативної пам'яті та частота системної шини.

Мінімальні вимоги до обладнання:

- ✓ процесор — Core 2 Duo 4.2 GHz;
- ✓ оперативна пам’ять — 8192 MB;
- ✓ вільної дискової пам’яті — 10 GB.

У табл. 4.1 наведено параметри швидкої роботи програмного продукту.

Таблиця 4.1

Параметри апаратної частини для роботи програмного продукту

Апаратна частина	**Опис апаратної частини**
Процесор	Core 2 Duo processor E8135 4,2 GHz
Материнська плата	Чіпсет NVidia nForce
Пам'ять	8 GB DDR3
Відеокарта	NVIIDIA GeForce 9900M, пам'ять 4096 MB
Жорсткий диск	SSD 1000 GB
Монітор	19" Full HD 1080P Acer 1920x1080 @ 85 Hz
Аудіоплата	Інтегроване аудіо 7.1
Корпус	Middle Tower ATX 500 Вт
Клавіатура	USB
Миша	USB

Висновки до розділу 4

Розроблений програмний продукт завадостійкого кодування з корекцією помилок дозволяє знаходити до 50% помилок та виправляти 25% помилок від довжини кодової послідовності.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Основною проблемою сучасної теорії та техніки зв'язку та радіоуправління є підвищення стійкості систем телекомунікацій, особливо ліній управління, як до природних, так і до штучних перешкод, створюваних противником. Однією з основних концепцій підвищення стійкості, розроблених у цьому проекті, є швидка зміна збірок робочого коду, що дозволяє підвищити стійкість, енергетичну та параметричну скритність системи зв'язку, а також захистити інформацію від несанкціонованого доступу.

З обраної проблеми та концепції підвищення стійкості сучасних телекомунікаційних систем випливає, що завдання синтезу повних класів систем лінійних та нелінійних шумоподібних завадостійких кодів є дуже важливим. Слід також зазначити, що ефективне вирішення проблеми стійкості може бути реалізовано за рахунок розвитку нових інформаційних технологій, що базуються на спільному застосуванні технології шумоподібного завадостійкого коду. Ці технології також мають підвищити ступінь захисту інформації від несанкціонованого доступу.

5.2. Розроблення ринкової стратегії

Метою проєкту було створення авадостійкого алгоритму кодування потоків числових даних на основі числових в’язанок та подальша реалізація програмного забезпечення однією з мов високого рівня.

Під час їх передачі може статися несанкціонований доступ до інформації. Щоб утруднити подальше використання захопленої інформації, перед передачею її можна закодувати різними кодами.

Слід зазначити, що запропонований метод кодування має низку переваг перед іншими кодами. Одним з них є простота виявлення та виправлення помилок на стороні, що приймає, оскільки поява в прийнятій кодовій комбінації хоча б одного символу «1» серед нулів або символу «0» серед одиниць свідчить про помилку. Помилка не виявляється лише при виникненні хибного сигналу в першому або останньому символі блоку (на межі блоків нулів та одиниць). З появою в коді невірних символів вони відразу виявляються все або частина, що забезпечує високу стійкість коду. Використання цієї програми продемонструвало її високу стійкість до навмисного злому інформації та може успішно використовуватись для захисту будь-якої інформації.

5.3. Розроблення маркетингової програми

Споживачами запропонованого програмного забезпечення можуть стати користувачі Інтернету, яким необхідна безпечна та надійна передача інформації.

До конкурентів продукту входить компанія SunSystem, що випускає аналог цієї програми Stego.

Основними недоліками аналога є:

- обмежений обсяг передачі;
- висока вартість програмного забезпечення;
- немає вихідного коду.

Основними перевагами аналогічного рішення у порівнянні з проектним рішенням є:

- високошвидкісний код;
- можливість надсилати інформацію у кількох файлах.

5.4. Вимоги до технічного та програмного забезпечення

Програма показує та реалізує можливість спрощеної побудови з використанням ідеальних числових в’язанок завадостійких моделей прямого та зворотного коду.

Дослідженні та проведенні програмні експерименти з виявлення та виправлення помилок з використанням завадостійких кодів на основі ідеальних числових в’язанок.

Використання завадостійкого коду, стійкого до прямих та зворотних перешкод, дозволяє підвищити захищеність при прийомі та передачі інформації.

Висновки до розділу 5

В даному проєкті з метою підвищення стійкості до відмови сучасних IT-систем був розроблений регулярний і конструктивний метод синтезу повних класів лінійних систем і нелінійних шумоподібних завадостійких кодів, стійких до перешкод, що дозволило істотно підвищити параметричні параметри - секретність і захист інформації від несанкціонованого доступу.

ВИСНОВКИ

1. Досліджено сучасні методи завадодостійкого перетворення інформації.
2. Визначено зв'язок між вибором оптимальних параметрів ідеальної кільцевої в’язанки та побудовою завадостійкого кодування з найкращими характеристиками.
3. Перевірено здатність корекції кодування з виправленням помилок на основі ідеальної кільцевої в’язанки (виправлення до 25% довжини коду та виявлення до 50% довжини коду).
4. Зроблено порівняння характеристик стійкого до помилок кодування на основі ідеальної кільцевої в’язанки з тими, що базуються на кодах BCH, і виявлено кращі характеристики корекції кодів ідеальної кільцевої в’язанки з довжиною блоку 63 біти або більше.
5. Досліджено можливість архівації завадостійкої системи на базі ідеальної кільцевої в’язанки. Після архівування розмір закодованого файлу приблизно на 20% більший за вхідний файл.

СПИСОК ЛІТЕРАТУРИ

1. В.Ємець, А.Мельник, Р.Попович Сучасна криптографія. Основні поняття. – Львів: Бак, 2022. – 144с.
2. Вернер М. Основи кодування. - К.: Техносфера, 2014.
3. Гордієнко Г.В. Входження України у всесвітню систему інформації. // Нова політика. - 2009 р. - №5 – С. 64-67.
4. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу. - НД ТЗІ 1.1-001-98, ДСТСЗІ СБ України, Київ, 2018.
5. Зюко А.Г., Кловський Д.Д., Назаров М.В., Фінк Л.М. Теорія передачі сигналів. К: Радіо і зв'язок, 2021 р . -368 С.
6. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу. -НД ТЗІ 2.2.-002 -98, ДСТСЗІ СБ України, Київ, 2018.
7. Кнут Дональд, Грехем Роналд, Паташнік Орен Конкретна математика. Підстава інформатики - К.: Світ; Біном. Лабораторія знань, 2016. - С. 703.
8. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. - НД ТЗІ 2.2-001-98, ДСТСЗІ СБ України, Київ, 2018.
9. Лідовскій В.І. Теорія інформації. - К., «Вища школа», 2022. - 120с.
10. Метрологія та радіовимірювань в телекомунікаційних системах. Підручник для ВУЗів. / В. І. Нефьодов, В.І. Халкин, Є.В. Федоров та ін - К.: Вища школа, 2021 р . - 383с.
11. Різник В.В. Синтез оптимальних комбінаторних систем. – Львів: Вища школа, 1989. – 168 с.

12. Різник О.Я., Парубчак В.О. Кодування інформації на основі Використання монолітних кодів. Праці 13-ої міжнародної конференції з автоматичного управління, м. Вінниця, 2016, т.1, с.30-32.
13. Рудаков О.М. Числа Фібоначчі та простота числа 2127 -1 / / Математичне Просвітництво, третя серія. - 2000. - Т. 4.
14. Стахов О.П. Коди золотої пропорції. -К.: Радіо та зв'язок, 2014.
15. A.Dollas, W.T.Rankin and D.McCracken published "A New Algorithm for Golomb Ruler Derivation and proof of the 19 Mark Ruler" in IEEE Transactions On Information Theory (January, 2013, Volume 44, Number 01).
16. BIB.COM.UA. Программирование. http://www.bib.com.ua.
17. distributed.net. Project OGR. http://www.distributed.net/ogr.
18. Gauley M. Direct product difference sets // J. of Combinatorial Theory. 2011, ser. A. Vol. 23. № 3. P. 321-332.
19. http://bugtraq.ua/dnet/faq/ogr/.
20. http://commsci.usc.edu/faculty/golomb.html.
21. http://commsci.usc.edu/faculty/golomb.html. Solomon W. Golomb. University Professor.
22. http://ftp.stel.ru/ogr/about.html.
23. http://mathworld.wolfram.com/GolombRuler.html.MathWorld's description of Golomb Rulers.
24. http://www.Iskhodniki.Ua. http://www.sources.ua.
25. http://www.jrsoftware.org/isinfo.php. Inno Setup.
26. http://www.relib.com. RELIB.COM. Технології програмування.
27. http://www.rsdn.ua.
28. http://www.soliday.com/stephen/papers/#ICGA95.

ДОДАТОК А. КОД ПРОГРАМИ ЗАВАДОСТІЙКОГО КОДУВАННЯ

```
unit myRich;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls;
type
  TmyRich = class(TrichEdit)
  private
    { Private declarations }
    FOnHScroll: TNotifyEvent;
    FOnVScroll: TNotifyEvent;
  protected
    { Protected declarations }
    procedure WMHScroll(var Message: TWMHScroll); message WM_HSCROLL;
    procedure WMVScroll(var Message: TWMVScroll); message WM_VSCROLL;
  public
    { Public declarations }
    constructor Create(AOwner: TComponent); override;
  published
    { Published declarations }
    property OnHScroll: TNotifyEvent read FOnHScroll write FOnHScroll;
    property OnVScroll: TNotifyEvent read FOnVScroll write FOnVScroll;
  end;
procedure Register;
implementation
constructor TmyRich.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FOnHScroll := nil;
  FOnVScroll := nil;
end;
procedure TmyRich.WMHScroll(var Message: TWMHScroll);
{ пам'ятайте що дане повідомлення виписується двічі }
begin
  if Assigned(FOnHScroll) then
    FOnHScroll(Self);
  DefaultHandler(Message);
end;
procedure TmyRich.WMVScroll(var Message: TWMHScroll);
{ пам'ятайте що дане повідомлення виписується двічі }
begin
  if Assigned(FOnVScroll) then
    FOnVScroll(Self);
  DefaultHandler(Message);
end;
procedure Register;
begin
  RegisterComponents('myRich', [TmyRich]);
end;
end.
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, ExtCtrls, myRich;
type
int = integer;
str = string;
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    LabeledEdit1: TLabeledEdit;
    ScrollBar1: TScrollBar;
    ScrollBar2: TScrollBar;
```

```
    ScrollBar3: TScrollBar;
    RichEdit1: TRichEdit;
    RichEdit2: TRichEdit;
    RichEdit3: TRichEdit;
    Label1: TLabel;
    Label2: TLabel;
    GroupBox1: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    Label3: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure ScrollBar3Scroll(Sender: TObject; ScrollCode: TScrollCode;
      var ScrollPos: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
function LinesVisible(Memo: TRichEdit): integer;
var
  Form1: TForm1;
  sn:int;
  mas:array of int;
  mas_s:array of str;
  c_lines:int;
  scroll_n:int;
  _in,_er,_out:TStringList;
implementation
{$R *.dfm}
procedure copy_mem(bin,p pointer;i:int);assembler;
asm
pushad
mov ecx,i
mov edi,p
mov esi,bin
rep movsb
popad
end;
procedure ror(p:pchar;i:byte;left:bool=false);
var ss:str;
    count:int;
begin
if i=0 then exit;
count:=length(p);
if not left then
  begin
    setlength(ss,count);
    copy_mem(p,pchar(ss),count);
    copy_mem(pchar(ss),p+i,count-i);
    copy_mem(pchar(ss)+count-i,p,i);
  end else
  begin
    setlength(ss,i);
    copy_mem(p,pchar(ss),i);
    copy_mem(p+i,p,count-i);
    copy_mem(pchar(ss),p+count-i,i);
  end;
end;
procedure Create_table(s:str);
  var p:pchar;
    i,j:int;
    r:int;
    count:int;
begin
    r:=0;
    s:=s+#0;
```

```
    i:=1;
    count:=0;
    p:=@s[1];
    repeat
      inc(i);
      if (s[i]=',')or (s[i]=#0)then
        begin
        s[i]:=#0;
        SetLength(mas,count+1);
        mas[count]:=strtoint(p);
        if mas[count] = 1 then inc(R);
        p:=@s[i+1];
        inc(i);
        inc(count);
        end;
     until s[i]=#0;
sn:=count*(count-1)div R+1;
SetLength(mas_s,sn);
SetLength(mas_s[0],sn);
p:=@mas_s[0][2];
(p-1)^:='1';
for j:=0 to count-2 do
 begin
  FillMemory(p,mas[j]-1,48);
  Inc(p,mas[j]);
  (p-1)^:='1';
 end;
 FillMemory(p,mas[j]-1,48);
for i:=1 to sn-1 do
  begin
    mas_s[i]:=Copy(mas_s[0],1,sn);
    ron(pchar(@mas_s[i][1]),i,true);
  end;
form1.Label4.Caption:='N = '+inttostr(count)+'; R = '+inttostr(R)+';'#13#10#13#10+
'Sn= n (n - 1 ) / R + 1; Sn = '+inttostr(count*(count-1)div R+1)+'  (яа 4 6ira inqb .)'+#13#10#13#10+
't1 <= 2 (n - R ) - 1 ;'#9't2 <= n - R - 1 ;'#13#10#13#10
+inttostr(2*(count-R)-1)+' <= 2 ( '+inttostr(count)+' - '+inttostr(R)+' ) - 1;'#9+
inttostr(count-R-1)+' <= '+inttostr(count)+' - '+inttostr(R)+' - 1; ';
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
Randomize();
c_lines:=LinesVisible(RichEdit1);
ScrollBar1.SmallChange:=c_lines-1;
ScrollBar2.SmallChange:=c_lines-1;
ScrollBar3.SmallChange:=c_lines-1;
end;
procedure AddColoredLine(p:TRichEdit;AText: string; AColor: TColor);
 begin
  with p do
  begin
    SelStart := Length(Text);
    SelAttributes.Color := AColor;
    SelText:=AText;
  end;
 end;
procedure m1;
var  op:TOpenDialog;
    f_in:TFileStream;
    s:array of byte;
    k,p,a:str;
    i,j,E,r,qf:int;
    o,l:pchar;
    qa,qs:TStringStream;
    dm:int;
    c_er:int;
    max,za:int;
begin
c_er:=0;
```

```
za:=0;
if scroll_n>_er.Count-c_lines then
max:=_er.Count-c_lines else
max:=scroll_n+c_lines;
Form1.RichEdit2.Lines.Clear;
 for i:=max-c_lines to max-1 do
   begin
    k:=_er.Strings[i];
    p:=_in.Strings[i];
    form1.RichEdit2.Lines.Add(k);
    q:=2*sn+4;
    if (k<>'')and(p<>'')then
    for j:=1 to q do
    if (k[j]<>' ')then
     if k[j]<>p[j] then
      with form1.RichEdit2 do
      begin
        inc(c_er);
        if c_er mod 20 = 0 then
        form1.Label1.Caption:=inttostr(c_er);
        SelStart:=Perform(EM_LINEINDEX,za,0)+j-1;
        SelLength:=1;
        SelAttributes.Color:=clRed;
        SelAttributes.Style:=[fsBold];
        SelText:=k[j];
      end;
     inc(za);
   end;
form1.Label1.Caption:=inttostr(max-c_lines)+'-'+inttostr(max-1)+#13#10'Err: '+inttostr(c_er);
end;
procedure m0;
var i,max:int;
begin
if scroll_n>_in.Count-c_lines then
max:=_in.Count-c_lines else
max:=scroll_n+c_lines;
Form1.RichEdit1.Lines.Clear;
 for i:=max-c_lines to max-1 do
   form1.RichEdit1.Lines.Add(_in.Strings[i]);
end;
procedure m2;
var  op:TOpenDialog;
   f_in:TFileStream;
   s:array of byte;
   k,p,a:str;
   i,j,E,r,q,f:int;
   o,l:pchar;
   qa,qs:TStringStream;
   dm:int;
   c_er,max,za:int;
begin
c_er:=0;
za:=0;
if scroll_n>_out.Count-c_lines then
max:=_out.Count-c_lines else
max:=scroll_n+c_lines;
q:=2*sn+4;
Form1.RichEdit3.Lines.Clear;
 for i:=max-c_lines to max-1 do
   begin
    k:=_out.Strings[i];
    p:=_in.Strings[i];
    form1.RichEdit3.Lines.Add(k);
    if (k<>'')and(p<>'')and(k<>p)then
    for j:=1 to q do
    if (k[j]<>' ')then
     if k[j]<>p[j] then
      with form1.RichEdit3 do
      begin
```

```
        inc(c_er);
        if c_er mod 20 = 0 then
        form1.Label2.Caption:=inttostr(c_er);
        SelStart:=Perform(EM_LINEINDEX,za,0)+j-1;
        SelLength:=1;
        SelAttributes.Color:=clRed;
        SelAttributes.Style:=[fsBold];
        SelText:=k[j];
      end;
      inc(za);
   end;
form1.Label2.Caption:=inttostr(max-c_lines)+'- '+inttostr(max-1)+#13#10'Err: '+inttostr(c_er);
end;
function find_d(s:str):str;
var s1,s2:str;
   c:int;
   d_tmp,dm,pdm:int;
   i,j:int;
begin
c:=pos(' ',s);
s[c]:=#0;
for c:=c+1 to length(s) do
 if s[c] in ['0','1'] then break;
s2:=pchar(@s[c]);
dm:=sn;
s1:=pchar(s);
for i:=0 to high(mas_s) do
 begin
  d_tmp:=0;
  for j:=1 to length(s1) do
    if (s1[j]<>mas_s[i][j])and((s1[j]='1')or(mas_s[i][j]='1'))then
      inc(d_tmp);
  if dm>d_tmp then
    begin
      dm:=d_tmp;
      pdm:=i;
    end;
 end;
Result:=mas_s[pdm]+'   ';
dm:=sn;
s1:=s2;
pdm:=0;
for i:=0 to high(mas_s) do
 begin
  d_tmp:=0;
  for j:=1 to length(s1) do
    if (s1[j]<>mas_s[i][j])and((s1[j]='1')or(mas_s[i][j]='1'))then
      inc(d_tmp);
  if dm>d_tmp then
    begin
      dm:=d_tmp;
      pdm:=i;
    end;
 end;
Result:=Result+mas_s[pdm];
end;
procedure TForm1.Button1Click(Sender: TObject);
var op:TOpenDialog;
   f_in:TFileStream;
   s:array of byte;
   k,p,a:str;
   i,j,E,r,q,f:int;
   o,l:pchar;
   qa,qs:TStringStream;
   dm:int;
   id:DWORD;
begin
op:=TOpenDialog.Create(nil);
op.InitialDir:=GetCurrentDir+'\';
```

```
if Assigned(_in)then
  _in.Free;
_in:=TStringList.Create;
if Assigned(_er)then
  _er.Free;
_er:=TStringList.Create;
if Assigned(_out)then
  _out.Free;
_out:=TStringList.Create;
if op.Execute and (op.FileName<>'')then
 begin
   Create_table(edit1.Text);
   f_in:=TFileStream.Create(op.FileName,fmOpenRead);
   SetLength(s,f_in.size);
   f_in.Read(s[0],f_in.Size);
   f_in.Free;
   e:=StrToInt(LabeledEdit1.Text);
   for i:=1 to high(s)do
    begin
     k:=mas_s[s[i] shr 4];
     p:=mas_s[s[i] and $f];
     a:=k+'   '+p;
     _in.Add(a);
     for j:=1 to e do
      begin
        r:=random(sn-1)+1;
        if k[r]='0' then
         k[r]:='1' else
         k[r]:='0';
        r:=random(sn-1)+1;
        if p[r]='0' then
         p[r]:='1' else
         p[r]:='0';
      end;
     a:=k+'   '+p;
     _er.Add(a);
     _out.Add(find_d(a));
    end;
op.Free;
ScrollBar1.Max:=_in.Count;
ScrollBar2.Max:=_in.Count;
ScrollBar3.Max:=_in.Count;
CreateThread(0,0,@m0,0,0,id);
CreateThread(0,0,@m1,0,0,id);
CreateThread(0,0,@m2,0,0,id);
id:=0;
q:=2*sn+4;
for i:=0 to _in.Count-1 do
   begin
    k:=_out.Strings[i];
    p:=_in.Strings[i];
    if (k<>'')and(p<>'')and(k<>p)then
    for j:=1 to q do
    if (k[j]<>' ')then
     if k[j]<>p[j]then
      begin
       inc(id);
      end;
   end;
Label5.Caption:='К-сть не виявлених помилок :'+inttostr(id);
end;
end;
function LinesVisible(Memo: TRichEdit): integer;
var
  OldFont : HFont;
  Hand : THandle;
  TM : TTextMetric;
  Rect : TRect;
  tempint : integer;
```

```
begin
  Hand := GetDC(Memo.Handle);
  try
    OldFont := SelectObject(Hand, Memo.Font.Handle);
    try
      GetTextMetrics(Hand, TM);
      Memo.Perform(EM_GETRECT, 0, longint(@Rect));
      tempint := (Rect.Bottom - Rect.Top) div
      (TM.tmHeight + TM.tmExternalLeading);
    finally
      SelectObject(Hand, OldFont);
    end;
  finally
    ReleaseDC(Memo.Handle, Hand);
  end;
  Result := tempint;
end;
procedure TForm1.ScrollBar3Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
var id:DWORD;
begin
scroll_n:=ScrollPos;
ScrollBar1.Position:=scroll_n;
ScrollBar2.Position:=scroll_n;
ScrollBar3.Position:=scroll_n;
m0;
m1;
m2;
end;
end.
```