

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних систем та технологій

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

на здобуття ступеня вищої освіти «бакалавр»

(освітньо-кваліфікаційний рівень)

на тему Побудова та аналіз завадостійких обчислювальних систем

(назва українською)

Construction and analysis of interference-tolerant computer systems

(назва англійською)

Виконав: студент заочної форми навчання

спеціальності

123 Комп'ютерна інженерія

(шифр і назва напрямку підготовки, спеціальності)

Аль-Омарі Амір Олександрович

(прізвище, ім'я, по-батькові)

Керівник д.т.н., проф. Гунченко Ю.О.

(науковий ступінь, вчене звання, прізвище та ініціали, підпис)

Рецензент

(науковий ступінь, вчене звання, прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№ від « » 2021 р.

Завідувач кафедри

Ю.О. Гунченко

(підпис)

(прізвище, ініціали)

Захищено на засіданні ЕК №

протокол № від « » 2021р.

Оцінка / /

(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

Н.Ф. Казакова

(підпис)

(прізвище, ініціали)

Одеса - 2021

АНОТАЦІЯ

Актуальність роботи полягає у тому, що питання надійності зберігання та передачі інформації стають все актуальнішими з кожним днем.

Так як найчастіше завади в обчислювальних системах зустрічаються саме при передачі даних, а не завади в самих обчислювальних системах, тому у роботі ми розглядали завади в системі передачі даних.

Метою дипломної роботи є створення програмного комплексу для тестування і аналізу завадостійких кодів, що складається з окремих програмних компонентів. Завдання самого програмного комплексу полягає в наданні можливостей для порівняння різних типів кодів:

- науково-практичними задачами в цьому випадку стоє;
- вибір методу завадостійкого кодування для передачі інформації;
- визначення методів реалізації можливостей для порівняння різних типів кодів;
- створення програмного комплексу для завадостійких кодів;

Результати дипломної роботи можуть застосовуватись для подальшого використання і модифікації з метою розширення кодів, які можуть бути проаналізовані та параметрів, за якими проводиться аналіз.

ABSTRACT

The urgency of the work is that the issues of reliability of storage and transmission of information are becoming more relevant every day.

Since most often interference in computer systems occurs during data transmission, rather than interference in the computer systems themselves, so in this paper we have considered interference in the data transmission system.

The purpose of the thesis is to create a software package for testing and analysis of noise-tolerant codes, consisting of individual software components. The task of the software package is to provide opportunities to compare different types of codes:

- scientific and practical tasks in this case;

- the choice of noise-tolerant coding method for information transmission;

- identification of methods for implementing opportunities to compare different types of codes;

- creation of a software package for noise-tolerant codes;

The results of the thesis can be used for further use and modification in order to expand the codes that can be analyzed and the parameters by which the analysis is performed.

АННОТАЦИЯ

Актуальность работы заключается в том, что вопрос надежности хранения и передачи информации становятся все более актуальными с каждым днем.

Так как чаще всего помехи в вычислительных системах встречаются именно при передаче данных, а не помехи в самих вычислительных системах, поэтому в работе мы рассматривали помехи в системе передачи данных.

Целью дипломной работы является создание программного комплекса для тестирования и анализа завадостийных кодов, состоящий из отдельных программных компонентов. Задача самого программного комплекса заключается в предоставлении возможностей для сравнения различных типов кодов:

- научно-практическими задачами в этом случае стоим;

- выбор метода помехоустойчивого кодирования для передачи информации;

- определение методов реализации возможностей для сравнения различных типов кодов;

- создание программного комплекса для помехоустойчивых кодов;

Результаты дипломной работы могут применяться для дальнейшего использования и модификации с целью расширения кодов, которые могут быть проанализированы и параметров, по которым проводится анализ.

ЗМІСТ

ВСТУП.....	7
ПОСТАНОВКА ЗАДАЧІ.....	9
1. ЗАВАДИ, ЇХ СУТЬ І КЛАСИФІКАЦІЯ	10
1.1 Класифікація завад	10
1.2 Зовнішні завади.....	11
1.3 Внутрішні завади	12
1.4 Наслідки завад.....	12
2. МЕТОДИ БОРОТЬБИ З ЗАВАДАМИ.....	14
2.1 Поняття завадостійкості.....	14
2.2 Методи підвищення завадостійкості	15
2.3 Лінійні коди загального вигляду.....	17
2.4 Відстань Хеммінга.....	19
2.5 Код Хеммінга	20
2.6 Інверсний код	22
3. ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ АНАЛІЗУ КОДІВ... ..	24
3.1 Загальні принципи проектування програмного комплексу для порівняльного аналізу кодів	24
3.2 Проектування модуля «КОДЕР».....	26
3.3 Проектування модуля генерації помилок	27
3.4 Проектування модуля «ДЕКОДЕР».....	28
3.5 Проектування бази даних та модуля запису до неї.....	29
3.6 Проектування засобів OLAP.....	33
3.7 Проектування засобів візуального відображення даних	34
4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ АНАЛІЗУ КОДІВ.....	36
4.1 Основні аспекти реалізації програмного комплексу для порівняльного аналізу кодів	36
4.4 Реалізація модуля "ДЕКОДЕР"	41
4.5 Реалізація бази даних і модуля запису в неї	41
4.6 Реалізація засобів OLAP-компонента.....	47
4.7 Реалізація графічного представлення даних.....	50
5. АНАЛІЗ ДАНИХ, ЯКІ ОТРИМАНІ В РЕЗУЛЬТАТІ ТЕСТУВАННЯ	53
5.1 Основні аспекти аналізу кодів.....	53
5.2 Загальний аналіз Коду Хеммінга НЕМ(8,12,4).....	54

5.3 Загальний аналіз інверсний коду INV(8,16,8).	55
5.4 Загальний аналіз коду Хеммінга НЕМ(16,21,5).	56
5.5 Загальний аналіз інверсного коду INV(16,32,16).	58
5.6 Порівняння кодів НЕМ(8,12,4) и НЕМ(16,21,5)	59
5.7 Порівняння кодів INV(8,16,8) та INV(16,32,16)	61
5.8 Порівняння кодів НЕМ (8,12,4) та INV (8,16,8)	62
5.9 Порівняння кодів НЕМ(16,21,5) та INV(16,32,16)	63
5.10 Висновок по розділу	65
ВИСНОВОК.....	66
ПЕРЕЛІК ПОСИЛАНЬ	68
ДОДАТОК: Демонстраційні слайди.....	67

ВСТУП

З моменту появи обчислювальних систем гостро стоїть питання надійності зберігання та передачі інформації. З плином часу, людство все інтенсивніше для реалізації своїх потреб використовує обчислювальні системи різного призначення. Області застосування обчислювальних систем відмінні по своєму застосуванню. Поняття "обчислювальні системи" досить широке. До нього можна віднести складні багатопроцесорні кластери, різні засоби дистанційної передачі інформації. Обчислювальні системи використовуються в широкому спектрі сфер, включаючи побут, наукову діяльність, міжвідомча повідомлення, промисловість, оборонний комплекс. Чим важливіше сфера використання, тим вище рівень вимог, який пред'являється до обчислювальних систем.

Наприклад, передача даних в сфері оборонного комплексу держави може бути більш важливою, ніж аналогічний процес в офісі будь-якої компанії, в силу критичності помилки в першій сфері. В такому випадку, постають питання надійності передачі даних, її захищеності (в залежності від типу обчислювальних систем) і незмінності.

Одним із способів підвищення надійності інформації, що передається є завадостійке кодування. Суть такого кодування полягає в додаванні до вихідних кодуються послідовностям контрольних символів. Цим символом призначаються для використання декодирующою стороною - вони допомагають перевірити цілісність повідомлення або знайти і виправити помилки.

Фундамент завадостійкого кодування був закладений ще на початку другої половини двадцятого століття. Вагомий внесок у розвиток даної сфери внесли американські математики Клод Шеннон і Річард Хеммінга.

Дана тема, як і раніше актуальна. Завадостійке кодування застосовується при архівації даних, передачі даних в пакетних мережах, в супутникових і в стільникових системах зв'язку.

Метою даної роботи є створення програмного комплексу для тестування і аналізу кодів, що складається як із спеціально написаних програмних компонентів, так і готових програмних рішень. Завдання програмного комплексу полягає в наданні можливостей для порівняння різних типів кодів за часовими характеристиками. На основі роботи комплексу можна виконувати аналітичні порівняння, враховуючи динаміку змін показників кожного коду і порівняння різних кодів за одними й тими ж параметрами. Отримана інформація може надаватися у вигляді зрізів OLAP-куба і графіків.

ПОСТАНОВКА ЗАДАЧІ

В рамках даної роботи необхідно реалізувати програмний комплекс для тестування і аналізу завадоостійких кодів, а також ознайомитися з основними принципами завадоостійкого кодування.

Суть програмного комплексу полягає в розробці компонентів, що дозволяють здійснювати кодування символьних послідовностей, декодування і декодування послідовності з штучно згенерованої помилкою, а також в аналізі кодів двох типів, який дозволить протестувати функціональні можливості даного комплексу.

Крім вищевказаних компонентів необхідно спроектувати базу даних і заповнити її результуючими даними від інших компонентів, що займаються кодуванням або декодуванням. До бази даних повинна заноситися інформація про час кодування інформаційної символьної послідовності, декодування і декодування послідовності з помилкою для кожного коду, а також тип коду, за допомогою алгоритму якого виконуються операції.

За допомогою додаткових програмних пакетів необхідно агрегувати інформацію, що міститься в базі даних. Цей етап необхідно реалізувати за допомогою аналітичного програмного пакета SQL Server Analysis Services.

Заключним етап даної роботи повинен бути розділ, що містить аналіз даних, отриманих в результаті роботи програмного комплексу. Аналіз повинен включати порівняння досліджуваних кодів за схожими параметрами таким, як тип алгоритму, кількість інформаційних символів, а також кожного коду зокрема за всіма отриманими даними.

Візуалізувати отримані залежності необхідно за допомогою графіків, що ілюструють часові значення різних типів кодів при різних значеннях параметрів. У ролі параметрів повинні виступати значення довжини оброблюваної інформаційної символьної послідовності і типи кодів.

1 ЗАВАДИ, ЇХ СУТЬ І КЛАСИФІКАЦІЯ

1.1 Класифікація завад

Завада - небажане фізичне явище або вплив електричних, магнітних або електромагнітних полів, електричних струмів або напруг внутрішнього або зовнішнього джерела, яке порушує нормальну роботу технічних засобів або викликає погіршення технічних характеристик і параметрів цих коштів [1].

Завади класифікуються за певними ознаками:

- 1) атмосферні: космічне випромінювання, розряди блискавок, тертя наелектризованих частинок одна об одну;
- 2) індустріальні: розряди на лінії електропередач, завади від комутаційних процесів в різних промислових установках;
- 3) штучні (спеціально організовані);

За своїми властивостями завади зазвичай діляться на три основні категорії;

1) імпульсні завада - випадкова послідовність імпульсів довільної форми, тривалості і амплітуди. Імпульси йдуть з таким випадковим інтервалом, при якому перехідні процеси в приймальному пристрої, викликані попередніми імпульсами, встигають закінчитися до приходу нового імпульсу. Прийнято вважати, що параметри окремих імпульсів в заваді (амплітуда, тривалість і т.д.) дуже близькі до параметрів робочого сигналу, який переносить інформацію;

2) флуктуаційна (шумова) завада - безперервне випадкове (хаотичне) коливання гладкого типу. На ньому зазвичай відсутні викиди, які перевищують середнє значення більш, ніж в три рази. Часто, це результат накладання перехідних процесів в приймальному пристрої, викликаних впливом окремих імпульсів завади;

3) синусоїдальна завада- сукупність відрізків синусоїд випадкової тривалості, фази, частоти, амплітуди. Єдиний невинуватковий елемент завади - її форма. З'являються як гармоніки, викликані впливом мережі живлення;

Деякі електричні завади можна усунути шляхом екранування, заземлення засоби вимірювань, застосування спеціальних фільтрів. Теплові шуми, в свою чергу, можуть бути помітно зменшені при охолодженні їх джерела.

1.2 Зовнішні завади

Завади бувають зовнішні і внутрішні.

Зовнішні завади - це завади, створені коливаннями навколишнього середовища від незалежних, для конкретного каналу, джерел. Наприклад, вони можуть генеруватися за рахунок електростатичних і електромагнітних полів, створюваних сусідніми пристроями, такими як електричний транспорт, електрообладнання, комп'ютерна електроніка і т.п. Зовнішні завади створюються коливаннями навколишнього середовища від незалежних джерел і передаються на коливну систему через зв'язку, їх частотний спектр для конкретних умов можна вважати заданим.

Зовнішні завади часто носять одиночний або слабо рецидивний характер, тому їх втручання в дискретний канал зв'язку слабо прогнозовано щодо тимчасової шкали.

Такі завади можуть мати високий рівень, іноді навіть перевершує рівень сигналу. Тривалість завад в такому випадку на кілька порядків менше тривалості сигналу [2].

1.3 Внутрішні завади

Внутрішні завади - це завади, що генеруються обчислювальними пристроями, безпосередньо беруть участь у процесі передачі даних. Тобто вони виникають в самій апаратурі передачі інформації.

До них можна віднести завади у вигляді теплових шумів електронних ламп, напівпровідникових приладів, опір і інших елементів. Також, вони можуть бути викликані перетворенням сигналів в окремих елементах системи (шуми квантування, нелінійність характеристик перетворення), нестабільністю апаратури, її несправністю.

Внутрішні завади - це такі, що хаотично змінюються (флуктуаційні) напруги і струми, які виникають в опорах і підсилюючих елементах схеми. Джерелами флуктуаційних завад є все підсилювальні елементи і активні опори підсилювача [3].

1.4 Наслідки завад

Завади є небажаним фактором деструктивного характеру в обчислювальних системах, оскільки заважають нормальному функціонуванню системи.

Вони примушують творців обчислювальних систем йти на певні заходи, покликані мінімізувати вплив завад на конкретну систему.

Маючи різну природу своєї появи, завади можуть змінювати дані, що передаються між різними обчислювальними системами або в рамках однієї.

У зв'язку з цим, постає питання, яким чином завади змінюють передану інформацію? Розглянемо цей аспект на прикладі сучасних персональних комп'ютерів.

Інформація в комп'ютері передається у вигляді електричних імпульсів, кожний імпульс має значення «0» або «1». Ці імпульси посиляються між різними пристроями комп'ютера у вигляді послідовностей бітів, наприклад

00010101. Послідовності бітів можуть бути службової (керуючі команди) або неслужбової інформацією. Якщо завада вплине на передану послідовність бітів, то в залежності від рівня цієї завади може відбутися зміна значення деяких бітів послідовності, додавання нових мул зникнення старих біт. Це загрожує тим, що значення вихідної команди може змінитися на значення іншої команди або ж на сторонній набір символів, внаслідок чого робота пристроїв піде в неправильному режимі [4].

Для мінімізації негативного впливу завад на передачу даних, існують методи підвищення стійкості обчислювальної системи.

2 МЕТОДИ БОРОТЬБИ З ЗАВАДАМИ

2.1 Поняття завадостійкості

Завадостійкістю називається здатність системи максимально ефективно виконувати свої функції при наявності завад. Завадостійкість оцінюють інтенсивністю завад, при яких порушення функцій системи (пристроїв) ще не перевищує допустимих меж. Чим сильніше завада, при якій система продовжує правильно і ефективно функціонувати, тим вище вважається її стійкість.

Існує широкий спектр обладнання, пристроїв та приладів, де протистояння завад є важливою умовою для правильного функціонування. Це можуть бути персональні комп'ютери, системи навігації, супутниковий зв'язок, балістичні ракети.

Особливо часто поняття «стійкість» застосовують для характеристики пристроїв передачі інформації (наприклад, ліній зв'язку) або пристроїв спостереження (наприклад, радіолокаційних станцій). Для них в більшості випадків може бути визначено поняття «сигнал», і оцінка завадостійкості може проводитися шляхом розгляду співвідношення між завадою і сигналом, при якому забезпечується необхідне для нормальної роботи якість, наприклад в радіолокації - стосунки сигналу до завади, при якому забезпечується задана вірогідність виявлення (ймовірність правильного виявлення при певній ймовірності помилкової тривоги).

Здійснення оптимальних пристроїв, що реалізують необхідний рівень завадостійкості, зазвичай дуже складно, а їх неминучі технічні недосконалості не дозволяють досягти її в повній мірі. Тому зазвичай задовольняються тими пристроями, які при найбільшій їх простоті забезпечують хороше наближення до оптимального пристрою.

Таким чином, стійкість перед завадами часто є умовою, необхідною для правильного функціонування пристроїв або комплексу пристроїв.

2.2 Методи підвищення завадостійкості

Сьогодні існує велика кількість різних методів підвищення завадостійкості. Кожен метод застосовується в конкретному відповідному випадку. Наприклад, іноді підвищити стійкість можна за допомогою простого екранування ліній зв'язку, обгортання деяких елементів пристрою в завадостійкі матеріали або створення джерел штучного охолодження (кольорів) для зменшення наслідків тепловиділення. Ці методи застосовуються в радіoeлектроніці або в цифровій електроніці. Однак, вони досить прості і не завжди їх використанням обмежуються творці різного роду систем.

Наприклад, при побудові сучасних процесорів цих методів недостатньо. Для підвищення завадостійкості БІС-процесора крім охолодження ононого, необхідно також працювати над його архітектурою. Це передбачає вироблення інших способів підвищення завадостійкості.

Одним з найбільш ефективних способів підвищення завадостійкості, при передачі інформації, є спектр методів так званого завадостійкого кодування. Суть подібних методів полягає у використанні кодів, що виявляють помилки, і кодів, що виправляють помилки. Деякі коди можуть поєднувати в собі обидві функції: функцію пошуку і функцію виправлення. Дану групу кодів часто називають корегуючими кодами.

Для цього при передачі в корисні дані додають певним чином згенеровану надлишкову інформацію (контрольна сума). Контрольна сума (число) формується на основі спеціальної функції, наприклад, складанням всіх байтових значень за певним модулем. Потім, приймаючою стороною виконується перевірка контрольної суми - перерахунок, вироблений на підставі отриманої корисної інформації та контрольної суми.

Коди, що виправляють помилки тісно пов'язані з кодами виявлення помилок. Однак, коди, що виправляють помилки здатні знаходити і коригувати, а коди виявлення помилок здатні лише знаходити помилки в

отриманій інформації. Проте, коди, що виправляють помилки належать до тих же класів, що і коди, що виявляють помилки. Справедливо також твердження, що практично будь-який код, що виправляє помилки, може використовуватися для виявлення. Природно, що існує обмеження числа символів, які можна виправити, і воно залежить безпосередньо від конкретного коду.

За способом роботи з інформацією, коди, що виправляють помилки, діляться на блокові і згорткові. Блокові ділять вхідний потік даних на фрагменти заданої незмінної довжини, які обробляються послідовно, а згорткові - працюють з безперервним потоком даних.

Розглянемо детальніше поняття блокові коди. Отже, нехай кодуються дані діляться на фрагменти довжиною k біт, перетворюються в так звані кодові слова довжиною n біт. Тоді відповідний блоковий код зазвичай позначають (n, k) . При цьому число $R = k / n$ називається швидкістю коду.

Якщо вихідні k біт код не перетворює і не залишає в незмінному вигляді і лише додає $n - k$ перевірочних символів такий код називається систематичним, інакше - несистематичним.

Розглянемо формальне визначення блочного коду.

Блоковий код - код, який кодує послідовності з набору символів алфавіту S в кодові слова, перетворюючи окремо кожен символ шуканого алфавіту. Нехай $(k_1, k_2, \dots, k_m)$ - послідовність натуральних чисел, де кожне менше $|S|$. Якщо $S = (s_1, s_2, \dots, s_n)$ і деякий слово W з алфавіту S записано як $W = a_{k1} a_{k2} \dots a_{km}$ тоді кодовим словом, відповідним W , а саме, $C(W)$ буде: $C(W) = C(s_{k1}) C(s_{k2}) \dots C(s_{km})$ [5].

Інакше кажучи, блоковий шифрує потік даних, розбиваючи його на слова по черзі, а кожне слово шифрує посимвольний.

Задати блоковий код можна по-різному, наприклад таблицею, де кожної сукупності з k -інформаційних біт, зіставляється n - біт кодового слова. Блокових код повинен задовольняти мінімум трьом наступним умовам:

- 1) здатність виправляти найбільш можливу кількість помилок;
- 2) володіти найменшою надмірністю;

3) простотою кодування і декодування;

Ці вимоги суперечать один одному і, як наслідок, єдиного коду, який задовольняє всім вимогам - не існує. Тому існує велика безліч кодів, де кожен призначений для вирішення певних проблем.

На практиці блочне кодування застосовувалося на ранніх етапах мобільної комунікації.

Блокові коди, як правило, є лінійними. Оскільки коди нелінійні не здатні забезпечити легкість кодування і декодування, а також важче піддаються вивченню, ніж коди лінійні. Таким чином, умова лінійності коду є основною умовою можливості бути реалізованим для блочного коду.

Блочний код — код, що кодує послідовності з набору символів алфавіту S в кодові слова, перетворюючи кожен символ алфавіту окремо. Нехай $(k_1, k_2, \dots, k_m)$ - послідовність натуральних чисел, де кожне менше $|S|$. Якщо $S = (s_1, s_2, \dots, s_n)$ і деяке слово W з алфавіта S записано як $W = a_{k1}a_{k2} \dots a_{km}$ тоді кодовим словом, що відповідає W , а саме, $C(W)$ буде: $C(W) = C(s_{k1})C(s_{k2}) \dots C(s_{km})$ [5].

2.3 Лінійні коди загального вигляду

Лінійний код загального вигляду - це блоковий код довжиною n символів, що складається з $2k$ кодових слів. Такий код також називається лінійним (n, k) -кодом, за умови, що всі його $2k$ кодових слів утворюють k -мірний підпростір векторного простору n -послідовностей довічного поля $GF(2)$. Інакше кажучи, двійковий код може називатися лінійним, якщо сума по модулю 2 ($\text{mod } 2$) двох кодових слів також є кодовим словом цього коду. В даному контексті поняття «поле» означає безліч математичних об'єктів, над якими можна виробляти арифметичні операції: додавання, віднімання, множення і ділення.

Наведемо приклад. Візьмемо просте поле, що складається з двох елементів - 0 і 1. Для нього визначимо дві операції - додавання і множення (табл. 2.1).

Таблиця 2.1

Об'єкт	Об'єкт	Результат додавання (+)	Результат множення (x)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Певні в таблиці операції додавання і множення називаються складання по модулю 2 і множенням по модулю 2 відповідно.

Поле в даному прикладі складається з алфавіту з 2-х символів 0 і 1, а також операцій додавання і множення і позначається як GF (2). Де GF - це поле Галуа (Galois field) - поле, що складається з кінцевого безлічі елементів, а (2) - кількість елементів алфавіту.

Також, варто відзначити, що лінійні коди переважно повинні бути систематичними, тобто містити незмінну інформаційну частину довжиною k символів і надлишкову (перевірочну) частину довжиною $n - k$ символів.

Лінійні коди, як правило, досить добре справляються з рідкісними і великими послідовностями помилок, їх ефективність при невеликих, але частих помилках менш висока. Завдяки лінійності, для запам'ятовування або перерахування всіх кодових слів досить зберігати в пам'яті кодера або декодера багатьом меншу їх частину, а саме ті слова, які утворюють базис відповідного лінійного простору. А це значно спрощує реалізацію пристроїв декодування і кодування, а також робить лінійні коди більш привабливими для використання в практичних додатках.

Слід зазначити, що ефективність лінійного коду визначається трьома факторами: кількістю надлишкової інформації, необхідної для належного функціонування коду по його алгоритму, кількістю помилок, які код може

виправити і складністю реалізації операції кодування і декодування (як апаратної, так і у вигляді програмного забезпечення) [6].

2.4 Відстань Хеммінга

Одним з важливих понять в завадостійкого кодування є так зване відстань Хеммінга. Відстань Хеммінга - це число позицій, в яких відповідні символи двох слів однакової довжини різні. У широкому сенсі відстань Хеммінга використовується для рівних по довжині рядків будь-яких q -ічних алфавітів і служить метрикою відмінності об'єктів однакової розмірності. Метрика відмінності - це функція, яка визначає відстань в метричному просторі. Метричний простором називається безліч, де визначено відстань між будь-якою парою будь-яких елементів [7].

Даний термін був сформульований американським вченим Річардом Хеммінгом. Для визначення міри відмінності між двійковим кодовими комбінаціями в векторному просторі кодових послідовностей. У векторному просторі відстань Хеммінга $d(x, y)$ між двома двійковими послідовностями (векторами) x і довжини n - це кількість позицій, в яких вони різні [8].

Наведемо кілька прикладів, які демонструють суть поняття відстані Хеммінга.

$$1) D(0001001, 1001111) = 3;$$

$$2) D(23F40A, 11F42B) = 4;$$

$$3) D(\text{завод}, \text{закон}) = 2;$$

Відстань Хеммінга має властивості метричного простору і задовольняє таким умовам:

$$1) d(x, y) \geq 0;$$

$$2) d(x, x) = 0;$$

$$3) d(x, y) = d(y, x);$$

$$4) d(x, z) \leq d(x, y) + d(y, z);$$

Варто також додати, що функція відстані Хеммінга працює тільки при порівнянні символьних послідовностей однакової довжини, оскільки інакше операція з виявлення розрізняються символів стає неможливою.

Крім математики, принцип відстані Хеммінга, також, застосовується в біоінформатики і геноміки, наприклад, для аналізу ДНК і РНК.

2.5 Код Хеммінга

Код Хеммінга - один з перших коригувальних кодів. Код Хеммінга дозволяє не тільки виявляти помилки, але і виправляти їх. Застосовується код для контролю інформації при передачі даних, в оперативній пам'яті, в запам'ятовуючих пристроях.

Код Хеммінга є систематичним кодом з перевіркою на парність. Систематичні коди - це велика група блочних разделімих кодів, де всі символи можна розділити на інформаційні та перевірочні. Примітно, що перевірочні символи можуть утворюватися тільки в результаті лінійних операцій над перевірочними символами, і будь-яка незаборонена кодова комбінація може бути отримана в результаті виконання лінійних операцій над набором лінійно незалежних кодових комбінацій.

Отже, код Хеммінга дає можливість знаходити як поодинокі, так і подвійні помилки, а виправляти лише поодинокі помилки. Код Хеммінга має властивості блокового (n, k) -коду, де n - сумарна кількість розрядів, з них m розрядів - інформаційні, а k розрядів використовуються в якості контрольних (перевірочних).

Таким чином, загальне число розрядів в повідомленні після процесу кодування можна розділити на дві групи: символи, що представляють перекодувати вихідну послідовність символів (вихідні дані), і службові символи, згенеровані за певним алгоритмом і, що дозволяють при декодуванні, виконати різні операції коригування, використовуючи зворотний алгоритм. Іншими словами $n = m + k$.

В m розрядів розбивається на контрольні групи. Кожен контрольний розряд закріплюють за певною групою. При кодуванні, а саме перед процесом передачі, в контрольні розряди записуються 0 або 1. Ці символи є ознаками парності відповідних груп. Порядкові номери розрядів, як правило, є результатами зведення числа 2 в ступінь. Ступінь належить конкретного проміжку від $[0, z]$, де $z = \log_2 m$, де m - довжина послідовності бітів, що представляє перекодувати вихідну послідовність символів, але без службових символів.

Після процесу передачі приймаючою стороною проводиться перевірка на парність всіх контрольних груп і фіксується результат перевірки. Отримана послідовність біт утворює двійковечисло, що називається коригувальним. Воно повинно вказувати номер розряду, в якому з'явилася помилка.

Інформація про необхідну кількість контрольних символів, необхідних для правильного кодування інформаційної послідовності, приведена в таблиці 2.2.

Таблиця 2.2

Залежність між інформаційними і коригуючими символами

m	1	2	3	4	5	6	7	8	9	10	11	12
k	2	3	3	3	4	4	4	4	4	4	4	5
n	3	5	6	7	9	10	11	12	13	14	15	17

У таблиці 2.2 m - кількість інформаційних біт, k - кількість контрольних біт і n - сумарна кількість інформаційних і контрольних біт.

Кодову комбінацію можна розділити на чотири умовні групи - контрольні групи. Групи формуються на основі наявності одиниці в відповідному номеру групи розряді числа, представленого в двійковій системі числення.

Слід також зазначити, що код Хеммінга працює тільки з бінарними послідовностями символів, що складаються з символів «0» і «1». Інакше

кажучи, код Хеммінга працює з числами, що належать бінарного полю GF (2). Це призводить до необхідності перекодування вихідної послідовності символів в бінарний вигляд. Перевагою бінарної коду Хеммінга є дуже малий алфавіт і, як наслідок, - зручність роботи з таким алфавітом. Недолік полягає в використанні великої кількості символів, що використовуються для представлення вихідних послідовностей в бінарному вигляді. Найчастіше один символ вхідного тексту кодується бінарної послідовністю з восьми символів. Наприклад, символ «b» латинського алфавіту в коді ASCII має в десятковій системі числення значення «62», а кодується шляхом перекладу з десяткової в двійкову систему числення; в результаті отримаємо послідовність біт 00111110. Така довжина чревата збільшенням часу роботи алгоритму кодування, а тимчасові витрати на кодування вихідної послідовності символів зростають пропорційно кількості символів в цій послідовності

2.6 Інверсний код

Інверсний код - один з найпростіших блокових лінійних кодів. Він дозволяє виявляти помилки.

Суть даного коду полягає в додаванні до вихідної комбінації такий же по довжині комбінації, в якій містяться контрольні біти. Контрольна комбінація генерується за наступним принципом: якщо у вихідній комбінації парне число одиниць, то додається комбінація, яка повністю дублює вихідну, а якщо непарне, то значення додається комбінації містять інверсні значення відповідних вихідних значень.

Наприклад, комбінація 10101010 матиме контрольну послідовність 10101010, оскільки кількість одиниць у вихідній послідовності парне; вихідна комбінація 11111000 матиме послідовність контрольних символів 00011111.

Декодування інверсний коду здійснюється в два етапи. На першому етапі підсумовуються одиниці в першій групі символів. Якщо число одиниць парне, то послідовність контрольних бітів змінена не буде, якщо ж число

одиниць непарне, то послідовність контрольних бітів буде інвертована. На другому етапі порівнюються знову обчислені контрольні біти і інформаційні біти. Якщо все біти на відповідних позиціях збігаються, інформація не зазнала змін завад, і прийшла в незміненому вигляді. Розбіжність говорить про те, що при передачі була отримана помилка.

Перевагою інверсний коду є хороша здатність виявлення. Даний код виявляє практично будь-які помилки, виключаючи дуже рідкісні помилки зсуву, які відбуваються як серед інформаційних бітів, так і серед відповідних їм контрольних [9, 10].

Недоліком даного коду можна вважати велику кількість контрольних біт, які «обтяжують» закодовану послідовність [9].

3 ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ АНАЛІЗУ КОДІВ

3.1 Загальні принципи проектування програмного комплексу для порівняльного аналізу кодів

В даному розділі визначені основні аспекти, на які ми будемо спиратися при розробці програмного комплексу для аналізу коригувальних кодів.

Даний комплекс програмного забезпечення буде служити основою для аналізу кодів, а саме таких характеристик коду, як час кодування, розмір кодового слова і довжина вхідної символічної послідовності. Варто зазначити, що даний комплекс призначений для порівняння різних кодів між собою. В його функціональні можливості має входити порівняння різних кодів в однакових умовах. Такий підхід дозволить отримати адекватну і коректну інформацію про різницю між можливостями кодів, в залежності від їх властивостей.

Даний комплекс передбачається реалізовувати не у вигляді монолітної системи, а у вигляді декількох дискретних модулів, пов'язаних між собою, де кожен модуль буде виконувати своє призначення. Взаємодія між модулями має бути частково автоматичним, а частково підтримуватися оператором. Таке рішення дозволить зробити цю систему більш гнучкою, оскільки подібна реалізація сприятиме швидкому коригуванню або зміни функціоналу тієї чи іншої частини комплексу. Таким чином, впровадження змін спричинить за собою відносно менші витрати часу, ніж впровадження оних в єдину монолітну систему.

Опишемо життєвий цикл повідомлення, оброблюваного програмним комплексом, у вигляді алгоритму:

- 1) повідомлення в символічному вигляді подається в модуль, який здійснює кодування (модуль «Кодер»);

- 2) модуль «Кодер» створює бінарне представлення символів повідомлення на основі їх порядкового номера в таблиці ASCII;

3) потім модуль «Кодер» виконує відповідні потрібного коду (наприклад, коду Хеммінга) алгоритмічні операції і записує дані про довжину кодованого слова і часу кодування в лог-файл;

4) «кодер» посилає отриману закодовану бінарну послідовність модулю «декодер» через канал зв'язку;

5) потім, в залежності від необхідності спрацьовує блок генерації помилок;

6) «декодер» обробляє отримане повідомлення за потрібне алгоритмом і заносить в лог-файл дані, аналогічні даним кодує боку;

7) на підставі лог-файлів модулів «Кодер» і «декодер» інший модуль реалізує запис в базу даних;

8) база даних обробляється за допомогою функціональних можливостей OLAP-аналізатора;

9) реалізується графічне представлення отриманих даних;

На рисунку 3.1 представлена схема, що демонструє взаємодію модулів.

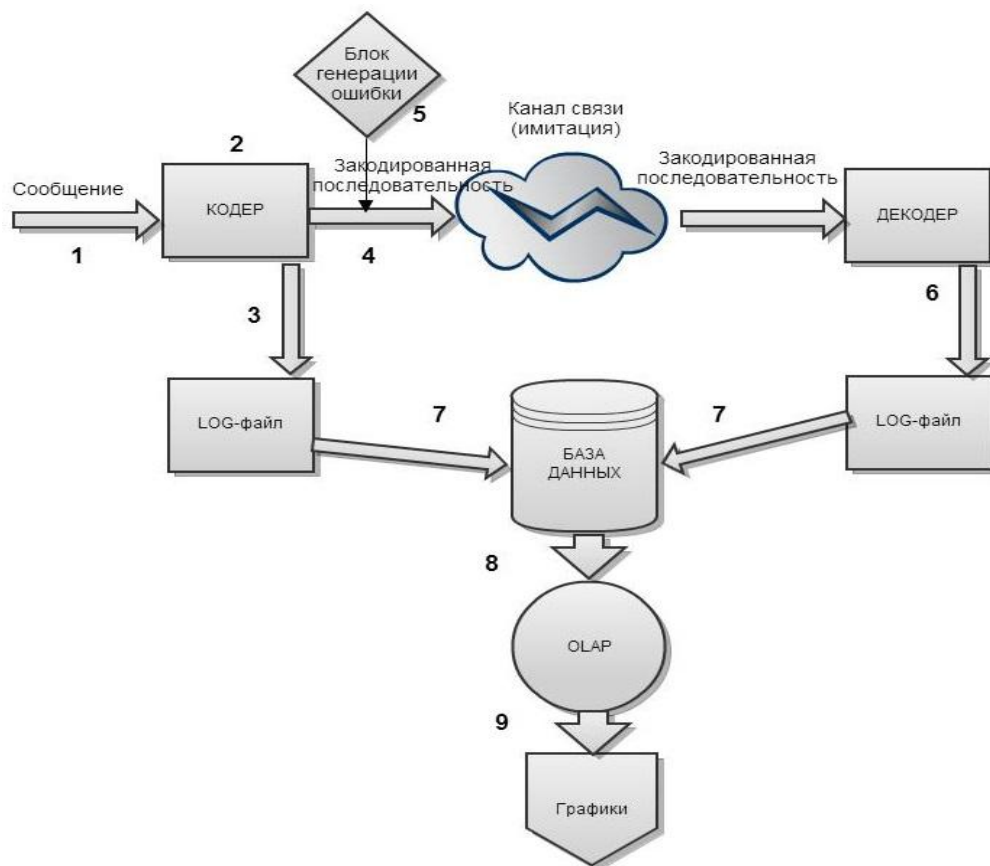


Рисунок 3.1 – Схема программного комплекса анализа кодов

Слід також зазначити, плюс даного підходу, що полягає у використанні коштів для розробки даного комплексу. Шуканий комплекс передбачається реалізовувати за допомогою продуктів однієї екосистеми - Microsoft. Це дозволить спростити і зробити більш ефективним взаємодія між модулями, оскільки продукти, які будуть використовуватися для розробки модулів, мають добре налагоджена взаємодія. Мінусом такого підходу є необхідність наявності навичок, що дозволяють працювати з продуктами даної екосистеми.

3.2 Проектування модуля «КОДЕР»

Модуль кодування повідомлення, за своїм призначенням повинен виконувати наступні функції:

- 1) переклад символьного послідовності в бінарну послідовність, що складається з символів 1 і 0;
- 2) кодування бінарної послідовності за відповідним алгоритмом і передача закодованої послідовності через емітований канал зв'язку, а саме до декодируючої сторони;
- 3) висновок даних про кодированой послідовності і часу, витраченому на процес кодування даної послідовності;

У реальних умовах, операції кодування і декодування часто реалізуються за допомогою апаратних засобів. Однак, метою даної роботи є аналіз різних типів кодів, що використовуються для завадостійкого кодування. Тому було вирішено досліджувати необхідні аспекти кодів на програмному рівні, емітуючи апаратну частину за допомогою програмного забезпечення.

Розробка модуля «Кодер» буде вестися за допомогою середовища розробки Visual Studio 2012 на компанії Microsoft. Даний модуль буде реалізований на мові C ++. Мова C ++ часто використовується для програмування електронних пристроїв.

Передбачається, що три вищенаведені функціональні можливості даного модуля будуть реалізовані за допомогою механізму використання функцій, а

саме написання дискретних логічно цілісних ділянок програмного коду, які згодом можна багаторазово викликати, передаючи їм на обробку необхідні дані, або всіляко редагувати і змінювати.

Такий метод дозволить виконати у вигляді програмного забезпечення весь спектр функціональних можливостей даного модуля.

3.3 Проектування модуля генерації помилок

В умовах даного програмного комплексу модуль генерації помилок є необхідним елементом для реалізації аналізу кодів. При передачі інформації в реальних умовах через реальний канал зв'язку, інформація може піддаватися впливу завад, що, в свою чергу, може спричинити зміни символів в переданій послідовності. Оскільки даний комплекс не передбачає інформаційного обміну між кодуючою і декодуючою стороною за допомогою каналу зв'язку, то закодована послідовність позбавлена ризику бути зміненою.

Це добре при апаратному кодуванні, однак, оскільки дана система розробляється для аналізу і порівняння кодів і їх можливостей, то більш широкий спектр інформації для аналізу можна отримати, маючи дані про те, як коди виправляють помилки.

Виходячи з вищевказаного, з'являється необхідність генерації помилок для перевірки часу роботи алгоритму при виправленні них. Генератором цих помилок і буде служити цей блок.

Обробка переданого закодованого повідомлення буде здійснюватися блоком генерації помилок за наступним алгоритмом:

- 1) закодована послідовність надходить від модуля «Кодер» на блок генерації помилок;
- 2) блок генерації помилок змінює інформаційний біт закодованого повідомлення;
- 3) блок генерації помилок передає закодовану бінарну послідовність наступного модуля - модулю «декодер»;

Даний алгоритм досить простий, однак слід відзначити кілька моментів. Використання блоку генерації помилок буде застосовуватися до закодованої послідовності строго в необхідних випадках. Тобто передбачається, що кожна закодована послідовність буде передаватися спочатку без помилки, а потім з помилкою в інформаційному біте, згенерованої в результаті роботи блоку генерації помилок. Згодом, результати роботи коду при декодуванні закодованої бінарної послідовності без помилки і послідовності зі згенерованої помилкою будуть фіксуватися в лог-файлі при роботі модуля «Кодер».

Блок генерації помилок передбачається реалізовувати за допомогою того ж середовища розробки програмного забезпечення, що використовується для розробки модуля «Кодер» - Visual Studio 2012 компанії Microsoft. Це дозволить зменшити складності, що з'являються при налагодженні взаємодії модулів даного програмного комплексу.

За функціональний базис блоку генерації помилок буде взятий вищеописаний алгоритм, оскільки він розкриває логічний зміст даного модуля.

3.4 Проектування модуля «ДЕКОДЕР»

Модуль декодування закодованої бінарної послідовності є модулем, який взаємодіє, а саме отримує дані або від модуля «Кодер», або від модуля містить блок генерації помилок.

У зв'язку зі своїм призначенням модуль «декодер» повинен виконувати наступні функції:

- 1) одержувати на вхід закодовану бінарну послідовність від блоку генерації помилок або безпосередньо від модуля «декодер» минаючи блок генерації помилок;

2) реалізовувати декодування отриманої закодованої бінарної послідовності, що складається з символів 1 і 0, відповідно до необхідного алгоритму декодування;

3) подавати на вихід декодувати послідовність символів і заносити необхідні дані в лог-файл;

Слід зазначити, що алгоритм декодування, повинен відповідати алгоритму кодування не тільки за своїм типом (наприклад, Код Хеммінга або код CRC), а й за ключовими характеристиками, таким як довжина кодового слова і кількість контрольних символів. Дані, що заносяться в лог-файл модулем «декодер» повністю є такими самими, що заносяться модулем «Кодер» в свій лог-файл, а саме - довжина слова і час декодування. Згодом ці дані будуть необхідні для проведення різних маніпуляцій з метою аналізу та порівняння ефективності тих чи інших кодів з різними характеристиками.

Також, модуль «декодер» буде розроблений за допомогою того ж програмного забезпечення що і модуль «Кодер» і ґрунтуватиметься на механізмі використання функцій.

3.5 Проектування бази даних та модуля запису до неї

Одна з ключових частин даного програмного комплексу це база даних, яка міститиме інформацію з лог-файлів модулів «Кодер» і «декодер». В один модуль разом з базою даних повинне входити і програмне забезпечення, що реалізує заповнення бази даних на основі лог-файлів.

Насамперед необхідно визначитися зі структурою бази даних, а саме з її таблицями, і полями в цих таблицях. База даних складається з п'яти таблиць. Це таблиці Sequenses, CodeTypes, Encodes, Decodes і DecodesEr, призначені для запису в них даних послідовностях для кодування, алгоритмах і результатів кодування, декодування і декодування з помилкою, відповідно. Саме в ці таблиці буде заноситися необхідна для аналізу інформація.

Розглянемо структуру бази даних, представлену на рисунку 3.2.

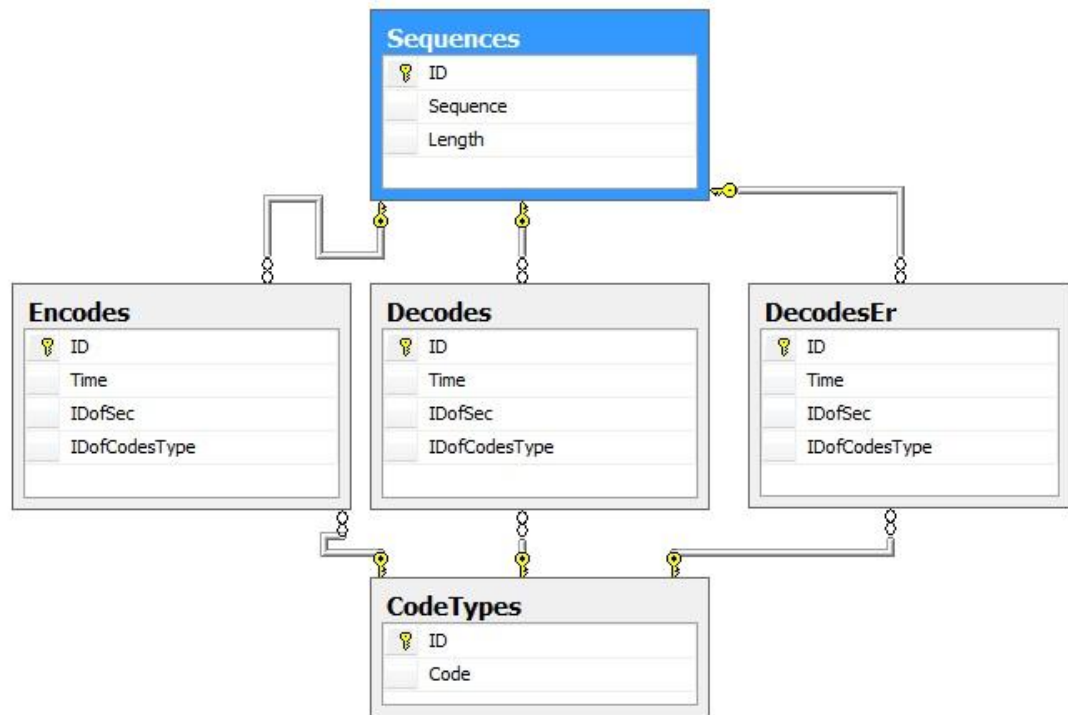


Рисунок 3.2 – Структура бази даних

Розглянемо таблиці Sequences (табл. 3.1), CodeTypes (табл. 3.2), Encodes (табл. 3.1), Decodes (табл. 3.4) і DecodesEr (табл. 3.5) з їх полями, типами даних і опис полів.

Таблиця 3.1

Схема таблиці Sequences

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Sequence	nvarchar(1100)	Поле, що містить текст кодованої послідовності
3.	Length	int	Поле, що містить довжину кодованої послідовності

Таблиця 3.2

Схема таблиці CodeTypes

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Code	nvarchar(100)	Поле, що містить назву типу алгоритму

Таблиця 3.3

Схема таблиці Encodes

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Time	float	Поле, що містить значення часу, витраченого на кодування послідовності
3.	IDofSec	int	Поле зовнішнього ключа. Служить для зв'язу часу кодування і кодуваної послідовності
4.	IDofCodesType	int	Поле зовнішнього ключа. Служить для зв'язу часу кодування і типу алгоритму кодування

Таблиця 3.4

Схема таблиці Decodes

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Time	float	Поле, що містить значення часу, витраченого на декодування послідовності
3.	IDofSec	int	Поле зовнішнього ключа. Служить для зв'язу часу декодування і декодуваної послідовності
4.	IDofCodesType	int	Поле зовнішнього ключа. Служить для зв'язу часу декодування і типу алгоритму декодування

Таблиця 3.5

Схема таблиці DecodesEr

№	Ім'я по`ля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Time	float	Поле, що містить значення часу, витраченого на декодування послідовності
3.	IDofSec	int	Поле зовнішнього ключа. Служить для зв'язу часу декодування з помилкою і декодуваною послідовністю
4.	IDofCodesType	int	Поле зовнішнього ключа. Служить для зв'язу часу декодування з помилкою і типом алгоритму декодування

Для реалізації бази була обрана СУБД Microsoft SQL Server 2012. Даний продукт добре інтегрований з використанням середовища розробки Visual Studio 2012, а також володіє потужними засобами для роботи з базою даних.

Також, необхідно розписати функціональні можливості програми, призначеної для заповнення бази даних наявною інформацією. Зробимо це у вигляді алгоритму:

- 1) програма повинна зчитувати дані з лог-файлів;
- 2) приводити лічені дані у відповідний для запису в базу даних формат;
- 3) ініціювати підключення до бази даних;
- 4) здійснювати запис у відповідні поля таблиць;

Такий алгоритм, дозволить максимально надійно розподілити дані з лог-файлів у відповідні табличні поля. Також, використання об'єктно-орієнтованої

мови C # в даному модулі, дозволить швидко і на відносно високому рівні здійснити запис необхідних даних в базу даних.

3.6 Проектування засобів OLAP

OLAP (англ. Online analytical processing) - технологія обробки даних, що дозволяє приводити дані в необхідну форму і аналізувати їх з різних аспектів. Системи аналізу, засновані на принципах OLAP, застосовуються в різних сферах. Наприклад, в сферах маркетингу, ціноутворення, продаж [10].

OLAP-системи дозволяють працювати з даними, а саме представляти їх в агрегованому вигляді, для подальшого зручності роботи з ними. Також, OLAP-системи допомагають прийняти необхідне рішення (наприклад, про закупів товару), яке повинно бути прийнято в майбутньому, ґрунтуючись на вже наявних статистичних даних. Найчастіше OLAP-системи не є самостійними рішеннями, а інтегруються в інші системи, наприклад, в системи підтримки прийняття рішень [11].

Як сховище даних для OLAP-систем можуть використовуватися джерела даних різного типу. В даному випадку буде використовуватися реляційна база даних MsSQL.

Ключовим об'єктом OLAP-системи є куб. Куб служить для візуалізації даних, що обробляються. Він дозволяє спеціалісту створювати різні комбінації даних на основі вимірів і заходів. Вимірювання - послідовність значень одного з аналізованих параметрів, міра - послідовність даних, що кількісно характеризує аналізовані факти [12, 13].

Формування куба на основі даних і вимірювань відбувається наступним чином: вимірювання стають осями куба, а по ним відкладаються заходи. Даний принцип представлений на рисунку 3.3.

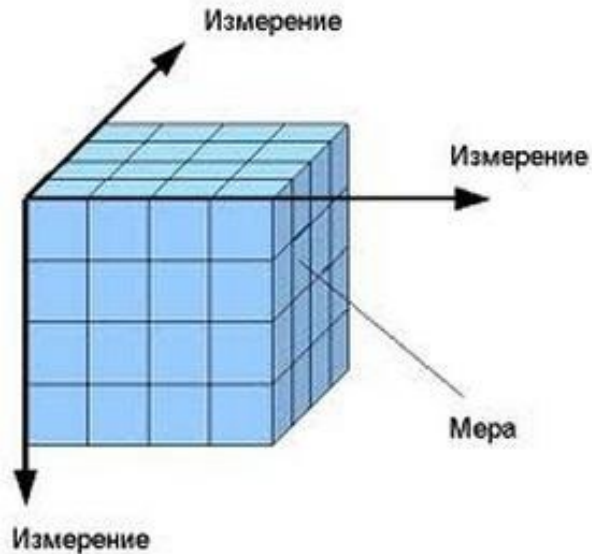


Рисунок 3.3 – Структура OLAP куба з трьома вимірами

Таким чином, в самому простому випадку двовимірний куб буде таблицею. Вимірювань у куба може бути більш ніж два. Однак специфіка така, що навіть багатовимірні куби можна уявити в двовимірному вимірі.

В даному програмному комплексі передбачається використовувати службу SQL Server Microsoft Analysis Services. Вона добре інтегрована з використовуваної в проєкті базою даних, а також дозволяє виробляти досить гнучкі конфігураційні налаштування для аналізу даних, і має можливості для зручної візуалізації їх.

3.7 Проектування засобів візуального відображення даних

Заключним етапом у процесі проектування даного програмного комплексу є питання візуального представлення даних.

Даний етап найбільш простий з точки зору реалізації, так як має на увазі використання готового програмного забезпечення.

Оскільки мета даної роботи полягає в аналізі кодів, то заключну частину, що демонструє роботу різних типів кодів, було вирішено реалізувати у вигляді графіків на основі отриманих даних.

Дані, які будуть представлені у вигляді графіків, будуть отримані з попереднього етапу, де реалізується OLAP-компонент даного комплексу, який, в свою чергу, буде займатися обробкою інформації з бази даних.

Таким чином, для візуалізації даних було вирішено використовувати програмний продукт під назвою Microsoft Office 2010, а саме Excel. Причини використання цього програмного забезпечення, обумовлюються хорошою інтеграцією продуктів Excel і SQL Server Analyses Services. Дані з OLAP-системи можна легко передавати в таблиці Excel, на підставі яких згодом, можна будувати інформативні графіки.

Такий підхід дозволить найменш витратним шляхом реалізувати поставлену мету, без негативних відображень на якість даного програмного комплексу.

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ АНАЛІЗУ КОДІВ

4.1 Основні аспекти реалізації програмного комплексу для порівняльного аналізу кодів

Як і було вирішено, при розробці даного комплексу використовувалися такі мови програмування як C ++, C # і SQL.

Мова програмування C ++ використовується для написання модулів «Кодер», «ДЕКОДЕР» і модуля генерації помилок [14]. Використання саме цієї мови програмування обумовлено його відносно більшою швидкістю в порівнянні, наприклад, з іншими високорівневими мовами C #, Java. Також він часто використовується при програмуванні контролерів пристроїв.

Мова програмування C # використовується для написання модуля, що взаємодіє з базою даних. Використання мови C # обумовлено його можливостями, що дозволяють ефективно працювати з базами даних, зокрема з використовуваної в даному проєкті базою даних MsSQL. Крім зручної роботи з базами даних мову C # дозволяє створювати як консольні додатки, так і віконні. Додаток, що здійснює взаємодію з базою даних повинне бути виконане в віконному вигляді [15].

В якості мови для занесення необхідної інформації в базу даних була вибрана мова SQL, завдяки своїй простоті і універсальності.

4.2 Реалізація модуля "КОДЕР"

Реалізація модуля «Кодер» цілком велася в середовищі програмування Visual Studio 2012. У ролі мови розробки виступив мову C ++.

По суті, модуль «кодер» є однією функцією, що реалізує кодування і повертає бінарну послідовність. Час виконання операції кодування і довжина кодованої символічної послідовності фіксуються в головній функції `_tmain`. Таким чином, з функції `_tmain` викликається функція реалізує кодування, після

закінчення роботи якої в лог-файл заноситься інформація про час роботи функції кодування, про довжину кодимуємої послідовності і сама послідовність в символьному вигляді.

Це зроблено для більшої наочності при ознайомленні з результатами роботи функції кодування над конкретною послідовністю символів. Кодування тексту виробляється в функції `tmain`, шляхом виклику функції кодування. Функція кодування викликається в циклі для кожної порції символів кодованого тексту.

Для визначення довжини кодимуємої послідовності використовувалася стандартна функція `sizeof()`.

Необхідно зауважити, що реалізація так званого таймера, що вимірює час, витрачений функцією кодування на кодування конкретної символьної послідовності, реалізується за принципом `timestamp` (англ. часова мітка).

Суть цього методу, полягає в наступному. Припустимо, є фрагмент програмного коду, час виконання якого необхідно виміряти. Для цього необхідно розмістити два таймера: один строго перед початком і другий - строго після кінця даного фрагмента коду. Оскільки компілятор буде обробляти код зверху вниз, то буде запущений перший таймер. Цей таймер зафіксує поточний машинний час. Слідом за першим таймером буде оброблятися потрібний фрагмент коду, і як тільки цей фрагмент коду буде оброблений компілятором - другий таймер зафіксує поточний машинний час.

Таким чином, таймери будуть містити значення машинного часу в момент свого запуску. Тоді щоб дізнатися час, витрачений на виконання ділянки коду, необхідно лише знайти різницю між другим і першим таймером. На малюнку 4.1 наведено приклад, який демонструє реалізацію даного методу за допомогою мови `C++`.

```
clock_t t0 = clock();

//...
//some code
//...

clock_t t1 = clock();

cout<<"time:"<<(double)(t1 - t0)/CLOCKS_PER_SEC<<endl;
```

Рисунок 4.1 – Реалізація методу **timestamp**

Варто відзначити, що в наведеному прикладі для використання типу `clock_t` необхідне підключення бібліотеки `ctime`.

Плюсом такого способу вимірювання часових показників в програмуванні є те, що обидва таймера залежать від однієї системи вимірювання часу.

Оскільки однією з основних задач при кодуванні є прагнення до найменшого часу виконання операції кодування, то в цілях зменшення часу було вирішено по можливості застосовувати метод так званого «розгортання циклів».

Суть даного методу полягає у відмові від використання циклу для будь-яких потреб і заміні циклу на послідовність операторів, які повинні були виконуватися в циклі. Проілюструємо даний приклад за допомогою рисунка 4.2, на якому зображений цикл, який згодом буде підданий операції «розгортання».

```
char arr[9] = "11111111";  
  
for (int i = 0; i<8; i++)  
{  
    arr[i]='0';  
}
```

Рисунок 4.2 – Реалізація цикла, що змінює елементи масива

На рисунку представлено цикл, який реалізує заміну значення кожного елемента символьного масиву іншим символом.

На рисунку 4.3 представлений той же масив, проте заміна кожного значення здійснюється за допомогою звернення до елемента масиву не в циклі, а безпосередньо через індекс.

```
char arr[9] = "11111111";

arr[0] = '0';
arr[1] = '0';
arr[2] = '0';
arr[3] = '0';
arr[4] = '0';
arr[5] = '0';
arr[6] = '0';
arr[7] = '0';
```

Рисунок 4.3 – «Розгортання» цикла

Такий підхід хороший тим, що виконання операцій при «розгорнутому» циклі відбувається швидше, ніж при виконанні операцій, що ведуть до того ж результату, але в циклі.

Зниження часу при «розгорнутому» циклі обумовлюється тим, що при виконанні операцій в циклі на кожному його кроці виконується перевірка - чи не виходить індекс за кордону циклу. І якщо в даному прикладі таких перевірок буде вісім, то в циклі, який повинен буде виконатися тисячу разів, їх буде тисяча. А це загрожує великими часовими затримками.

Мінусом даного підходу є великий обсяг коду, при «розгортанні» циклу з великою кількістю ітерацій. Також, не всякий цикл вдається розгорнути, оскільки формула, що реалізується в циклі, може бути досить складним для цього. Тому метод «розгортання» слід застосовувати тільки в разі потреби.

Наведемо алгоритм, згідно з яким працює модуль «Кодер»:

- 1) функція кодування приймає на вхід символ, який необхідно закодувати;
- 2) символічне уявлення перетворюється в бітовий масив, на підставі таблиці ASCII;
- 3) над двійкового масивом виконуються маніпуляції, відповідні алгоритму кодування;

4) функція кодування повертає закодований масив одиниць і нулів, що містить як інформаційні, так і контрольні біти. Даний масив згодом буде використовуватися для декодування;

Даний алгоритм реалізує весь необхідний функціонал модуля «Кодер».

4.3 Реалізація модуля генерації помилок

Модуль генерації помилок, як уже зазначалося, має виконувати імітацію помилок, а саме - змінювати інформаційний біт в закодованій послідовності і повертати змінений бінарний масив для подальшого декодування і пошуку помилки.

Функціональність даного модуля була реалізована в ще одному блоці декодування. Це зроблено для зменшення часу роботи алгоритму декодування в ситуації з імітацією помилки, оскільки, виклик функції, що здійснює генерацію помилки загрожує зайвими витратами часу, оскільки необхідно передавати і повертати масив з уже закодованим текстом.

Алгоритм роботи даного блоку виглядає наступним чином:

- 1) один з елементів закодованої послідовності (символьного масиву), отриманої функцією декодування, перевіряється умовним оператором if;
- 2) якщо значення досліджуваного елемента масиву дорівнює 0, то воно змінюється на 1;
- 3) інакше, якщо значення досліджуваного елемента дорівнює 1, то воно змінюється на значення 0;
- 4) символьний масив зі зміненим значенням одного елемента подається на подальше декодування і виправлення помилки;

Інакше кажучи, блок генерації помилок інвертує один із символів закодованої послідовності.

Безумовно, при передачі по реальному каналу зв'язку, значення закодованої послідовності може не тільки інвертуватися, а й змінюватися на будь-який інший символ або в закодованій послідовності може змінюватися

число символів. Однак, в даному програмному комплексі ми не можемо враховувати всі чинники реальних умов, тому моделюємо ситуацію в якій змінюється один інформаційний біт.

Результати роботи даного блоку, а саме текст декодіруемой послідовності, довжина послідовності, час декодування і виправлення помилки і тип алгоритму, виводяться в лог-файл під назвою dbdecodeerror.txt. На підставі цього лог-файлу в подальшому іншим модулем даного програмного комплексу буде заповнена відповідна таблиця в базі даних, яка згодом буде використовуватися для аналізу.

4.4 Реалізація модуля "ДЕКОДЕР"

Реалізація даного модуля по багатьом аспектам збігається з реалізацією модуля «Кодер». Модуль «декодер» виконаний в середовищі програмування Visual Studio 2012 на мові програмування C ++.

Даний модуль являє собою функцію, яка приймає закодовану послідовність, декодує її і повертає отриману, в результаті декодування, символічну послідовність. Функція викликається з головної функції `_tmain`. Час виконання операції декодування та інші дані заносяться в лог-файл блоку «декодер».

4.5 Реалізація бази даних і модуля запису в неї

Даний модуль відповідає за спектр можливостей, пов'язаних з функціонуванням бази даних. Умовно його можна розділити на 2 частини: база даних і програмне забезпечення для запису в цю базу даних.

База даних, як і планувалося на етапі проектування, складається з п'яти таблиць `Sequences`, `CodeTypes`, `Encodes`, `Decodes` і `DecodesEr`. Сама база даних називається `Codes-db`. Слід нагадати, що заповнення бази даних відбувається

автоматично, за допомогою спеціально написаного програмного забезпечення. Заповнення проводиться на підставі вмісту лог-файлів.

Розглянемо докладніше кожну з таблиць.

У таблиці Sequences містяться дані про послідовності, які будуть закодовані або декодовані. Приклад вмісту таблиці Sequences з тестовими даними представлений на рисунку 4.4.

ID	Sequence	Length
5	its	4
7	its time	8
8	its time to	12
9	its time to go tim	18
10	its time to go time for	23
11	its time to go time for the	27
12	its time to go time for the fre	31
13	its time to go time for the freedom	35
14	its time to go time for the freedom tim	39
15	its time to go time for the freedom time to	43
16	its time to go time for the freedom time to fac	47
17	its time to go time for the freedom time to face wh	51

Рисунок 4.4 – Вміст таблиці Sequences, заснований на тестованих даних

У таблиці CodeTypes міститься інформація про типи алгоритмах кодування, які використовуються в даному комплексі. На ідентифікатор даної таблиці посилаються зовнішні ключі таблиць Encodes, Decodes і DecodesEr. Приклад вмісту таблиці CodeTypes з тестовими даними представлений на рисунку 4.5.

ID	Code
1	HEM8

Рисунок 4.5 – Вміст таблиці CodeTypes, заснований на тестованих даних

У таблиці Encodes містяться дані про кодованій послідовності: час, витрачений на операцію кодування, а також два зовнішніх ключа, що посиляються на конкретну кодованих послідовність і тип алгоритму кодування. Приклад вмісту таблиці, заснований на роботі модуля «Кодер» з тестовими даними представлений на рисунку 4.6.

ID	Time	IDofSec	IDofCodesType
21	0,016	6	1
22	0,018	7	1
23	0,044	8	1
24	0,048	9	1
25	0,061	10	1
26	0,064	11	1
27	0,076	12	1
28	0,086	13	1
29	0,107	14	1
30	0,109	15	1
31	0,127	16	1

Рисунок 4.6 – Вміст таблиці Encodes, заснований на тестованих даних

У таблиці Decodes містяться дані про декодовану послідовність: час, витрачений на операцію декодування, а також два зовнішніх ключа, що посиляються на конкретну декодовану послідовність і тип алгоритму декодування. Приклад вмісту таблиці, заснований на роботі модуля «декодер» з тестовими даними представлений на рисунку 4.7.

ID	Time	IDofSec	IDofCodesType
21	0,001	6	1
22	0,003	7	1
23	0,006	8	1
24	0,009	9	1
25	0,01	10	1
26	0,015	11	1
27	0,0017	12	1
28	0,018	13	1
29	0,023	14	1
30	0,027	15	1
31	0,029	16	1
NULL	NULL	NULL	NULL

Рисунок 4.7 – Вміст таблиці Decodes, заснований на тестованих даних

У таблиці DecodesEr містяться дані про декодуємою послідовності з помилкою: час, витрачений на операцію декодування послідовності з помилкою, а також два зовнішніх ключа, що посилаються на конкретну декодуємою послідовність з помилкою і тип алгоритму декодування. Приклад вмісту таблиці, заснований на роботі модуля генерації помилок з тестовими даними представлений на рисунку 4.7.

ID	Time	IDofSec	IDofCodesType
21	0,002	6	1
22	0,003	7	1
23	0,009	8	1
24	0,01	9	1
25	0,012	10	1
26	0,016	11	1
27	0,022	12	1
28	0,026	13	1
29	0,028	14	1
30	0,03	15	1
31	0,035	16	1

Рисунок 4.7 – Вміст таблиці DecodesEr, заснований на тестованих даних

Реалізована база даних з трьома таблицями в ній. Міжтабличних зв'язку в базі даних Codes-db відсутні в зв'язку зі специфікою структури бази даних, і як наслідок, є непотрібними.

Розглянемо етап записи даних з лог-файлів в базу даних і відповідне програмне забезпечення.

Програма запису в базу даних, як і планувалося, була реалізована в середовищі розробки Visual Studio 2012. використовує мову програмування - C # (.NET 4.5).

Алгоритм роботи програми по запису даних з лог-файлів в базу даних виглядає наступним чином:

- 1) за допомогою методів класу StreamReader і змінних його типу виконується зчитування даних з необхідного лог-файлу;
- 2) лічені дані заносяться в змінну list типу List <string> (список рядків);
- 3) виконується підключення до бази даних локального сервера;
- 4) рядки зі списку list розбиваються по характерній ознаці на елементи; для запису в базу даних і конвертуються в відповідні полях таблиць типи даних
- 5) в циклі відбувається запис в базу даних;
- 6) переривається з'єднання з базою даних;

Слід також зазначити, що безпосередньо перед записом в базу даних відбувається розбиття елемента списку, що містить дані про послідовність. Розбиття проводиться за допомогою методу Split класу String.

Також, дані заносяться в таблиці за допомогою змінної класу SqlCommand. Ця змінна, на основі існуючого підключення до бази даних виконує запис в базу даних, за допомогою команди мови SQL - Insert.

У підсумку, процес запису в базу даних можна проілюструвати прикладом записи даних в таблицю Decodes, представленим на рисунку 4.8.

```

70     foreach (var item in list)
71     {
72         string[] s = item.Split(new Char[] { ';' });
73         double Time = Convert.ToDouble(s[2]);
74         string CodeType = s[3];
75
76         string query = String.Format("INSERT INTO Decodes (Time, IDofCodesType) "
77             + "Values ({0}, '{0}']", Time, CodeType);
78
79         SqlCommand command = new SqlCommand(query, connection);
80         command.ExecuteNonQuery();
81     }

```

Рисунок 4.8 – Приклад запису в базу даних в таблицю Decodes.

На рисунку показаний цикл `foreach`, в якому здійснюється запис в базу даних все елементів зі списку `list`, в який раніше були записані дані з відповідного лог-файлу. У рядку під номером 72 рядок зі списку `list` розбивається на підрядка, між якими є символ ';'. У рядках 73 і 74 відбувається конвертація і запис у відповідні змінні. Формат SQL-команди, що здійснює запис в базу даних, знаходиться на рядках 76 і 77. На 79-му рядку оголошується і ініціалізується змінна класу `SqlCommand`, що містить SQL-запит до бази даних та інформацію про підключення до локального сервера і бази даних. Запис виконується за допомогою методу `ExecuteNonQuery` класу `SqlCommand`. Дана послідовність операторів виконується для кожного рядка списку `list`.

На рисунку 4.9 наведено інтерфейс програми, яка здійснює запис в базу даних.

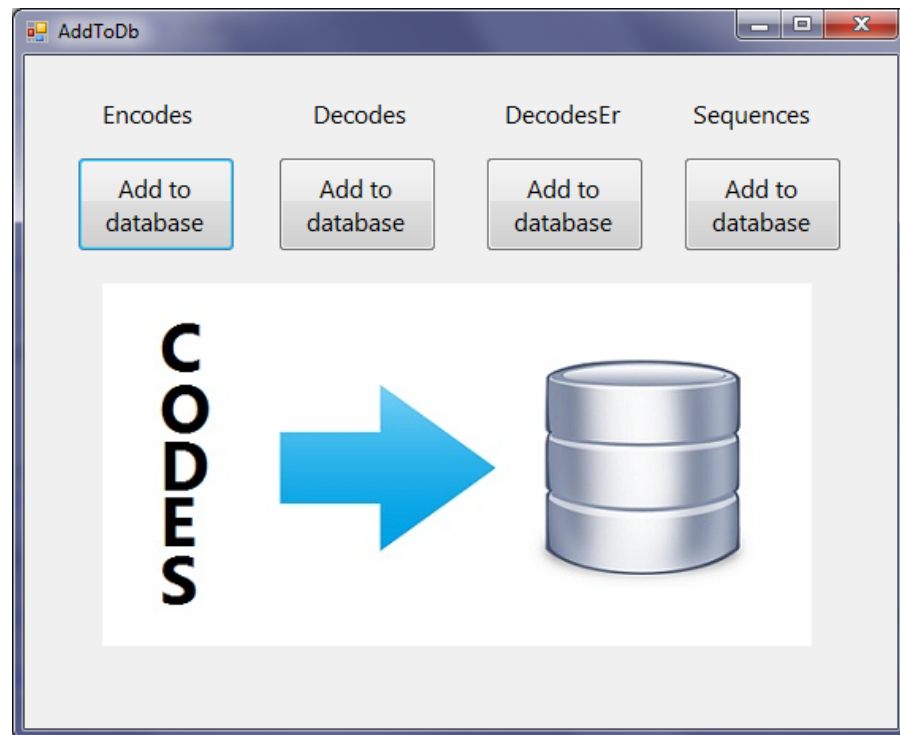


Рисунок 4.9 – Інтерфейс програми для запису в базу даних

У верхній частині вікна розташовуються мітки, відповідні чотирьом таблиць бази даних. Під кожною міткою знаходиться кнопка, при натисканні на яку, буде зроблено запис в таблицю бази даних, з назвою відповідним вищенаведеної мітці.

Дані для перших трьох таблиць Encodes, Decodes і DecodesEr беруться з відповідних лог-файлів, а саме: dbencode.txt, dbdecod.txt і dbdecodeerror.txt. Дані для таблиці Sequences беруться з таблиці Encodes, в яку заноситься довжина послідовності і її символічне уявлення.

4.6 Реалізація засобів OLAP-компонента

Одним із засобів, призначених для аналізу, в даному програмному комплексі є OLAP-компонент. Для виконання аналітичних завдань з тестування і порівняно кодів, в даному модулі було вироблено розгортання куба. Тепер, необхідно з'ясувати за допомогою яких вимірів і заходів відбувалося розгортання.

В даному проєкті в якості вимірів було вирішено використовувати поля таблиці Sequence, а саме ID, Length, Sequence і поля таблиці CodeTypes - Code і ID. На рисунку 4.10 представлені виміри куба.

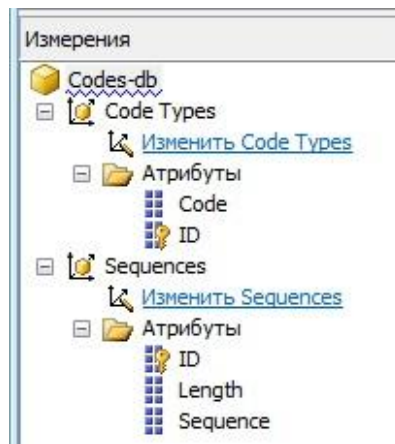


Рисунок 4.10 – Виміри куба

Як міри (фактів) були обрані поля Time таблиць Encodes, Decodes і DecodesEr. На рисунку 4.11 показані міри даного куба.



Рисунок 4.11 – Міри куба

На підставі вищенаведених мір був налаштований куб, необхідний для аналізу даних. Даний куб, по суті, є двовимірним, оскільки на одній з його осей може розташовуватися вимір, що стосується довжини послідовності, або її символічне уявлення (ці виміри однакові по семантиці і їх взаємовиключні використання обумовлюється лише зручністю подання інформації), а на іншій – вимір, що характеризує тип алгоритму.

Безумовно, допустимо розташування рівних, з точки зору семантики, атрибутів вимірів, проте підвищення інформативності або більш зручного представлення даних не буде.

Зрізом куба будемо називати зафіксоване співвідношення заходів і вимірів. Кожен зріз зазвичай орієнтований на отримання конкретного інформативного результату.

Розглянемо один з безлічі можливих зрізів даного куба з тестовими даними. Зріз показаний на рисунку 4.12.

		Code ▼		
		HEM8		
Sequence ▼	Time	Time - Decodes	Time - Decodes Er	
its	0,016	0,001	0,002	
its time	0,018	0,003	0,003	
its time to	0,044	0,006	0,009	
its time to go tim	0,048	0,009	0,01	
its time to go time for	0,061	0,01	0,012	
its time to go time for the	0,064	0,015	0,016	
its time to go time for the fre	0,076	0,0017	0,022	
its time to go time for the freedom	0,086	0,018	0,026	
its time to go time for the freedom tim	0,107	0,023	0,028	
its time to go time for the freedom time to	0,109	0,027	0,03	
its time to go time for the freedom time to fac	0,127	0,029	0,035	

Рисунок 4.12 – Зріз куба

На лівій осі куба розташований атрибут вимірювання під назвою Sequence, він показує символічну послідовність з якої велася робота. На місці цього атрибута міг бути атрибут Length або ID. Це не вплинуло б на інформативність зрізу. На верхній осі куба розташований атрибут Code, що показує тип коду, за допомогою якого були закодовані послідовності.

На перетині цих осей розташовані заходи Time, Time - Decodes, Time - DecodesEr. Оскільки є три групи заходів і в кожній з них міститься міра Time, то до імені другої і третьої заходи додається ім'я групи, якій дана міра належить.

Даний зріз дозволяє переглянути динаміку кодування, декодування і декодування з помилкою в рамках одного типу алгоритму і символічних послідовностей різної довжини. Він також дає можливість переглянути як

змінюється час, в залежності від конкретної операції, за умови роботи з однією і тією ж послідовністю.

Ці та деякі інші властивості і можливості куба дозволять згодом отримати максимум корисної аналітичної інформації, на основі даних про тестування кодів.

4.7 Реалізація графічного представлення даних

Фінальним етапом роботи даного програмного комплексу є візуалізація даних у вигляді графіків. Це зроблено за допомогою офісної програми Excel.

Виведення даних з куба, розгорнутого в попередньому розділі розділі, в таблиці Excel здійснюється за допомогою механізму зведених таблиць. Цей механізм дозволяє створювати так звані звітні таблиці Excel і виводити дані з безлічі різних джерел. В даному випадку необхідно лише здійснити підключення з Excel до сервера, на якому знаходиться досліджувана база даних, і вибрати відповідний проект, в якому був розгорнутий куб.

Після здійснення вищеописаних операцій було представлено меню для генерації вмісту таблиці. За допомогою цього меню можна згенерувати табличне представлення, аналогічне зрізу куба, отриманого в попередньому розділі.

Для цього використовуємо конфігурацію, представлену на рисунку 4.13.

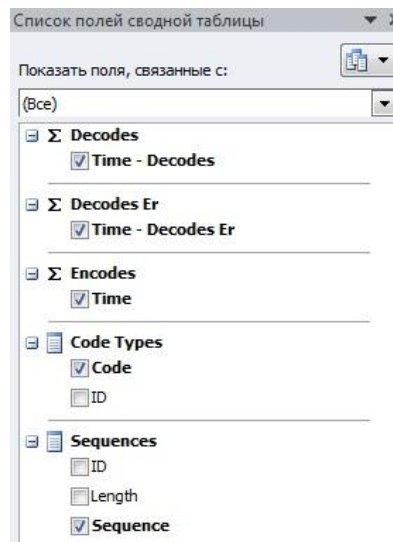


Рисунок 4.13 – Використовувана конфігурація

Як видно з представленого рисунка, були обрані атрибути вимірювань Code і Sequence. А також три міри Time, Time - Decodes і Time - DecodesEr.

На основі даної конфігурації таблиця звітності матиме вигляд, представлений на рисунку 4.14.

	A	B	C	D
1	Названия строк	Time	Time - Decodes	Time - Decodes Er
2	HEM8			
3	its	0,016	0,001	0,002
4	its time	0,018	0,003	0,003
5	its time to	0,044	0,006	0,009
6	its time to go tim	0,048	0,009	0,01
7	its time to go time for	0,061	0,01	0,012
8	its time to go time for the	0,064	0,015	0,016
9	its time to go time for the fre	0,076	0,0017	0,022
10	its time to go time for the freedom	0,086	0,018	0,026
11	its time to go time for the freedom tim	0,107	0,023	0,028
12	its time to go time for the freedom time to	0,109	0,027	0,03
13	its time to go time for the freedom time to fac	0,127	0,029	0,035

Рисунок 4.14 – Таблиця, що містить зріз куба

Порівнявши цю таблицю зі зрізом куба, отриманим в попередньому розділі, можна помітити, що їх інформаційне наповнення ідентично, з точки зору, як вимірювань, так і мір.

Таким чином, зріз куба був повністю відображено в звітній таблиці Excel, без втрати інформативності. Тому, на підставі цієї таблиці був побудований графік, для демонстрації аналітичних можливостей програмного комплексу. Даний графік представлений на рисунку 4.15.

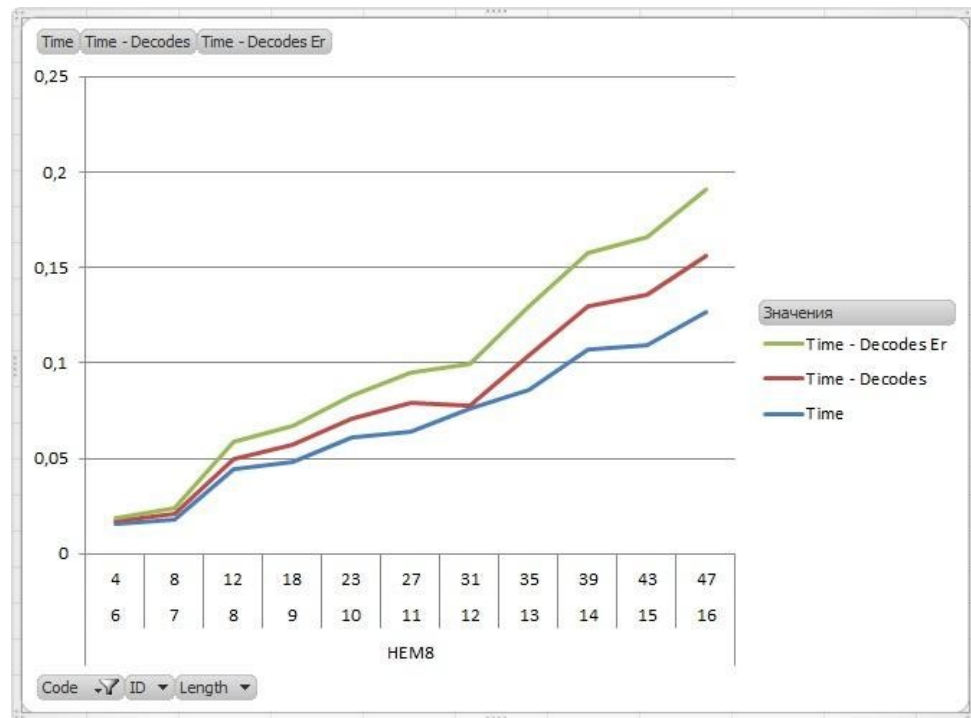


Рисунок 4.15 – Графік, заснований на зрізі куба

Слід зазначити, що нижній рядок на осі абсцис є послідовністю ідентифікаторів таблиці Sequence і служить для виведення довжин послідовностей по зростанню.

В результаті можна відзначити, що даний компонент програмного комплексу є хорошим засобом візуалізації досліджуваних даних, а також дозволяє будувати зручні графіки різного характеру.

5. АНАЛІЗ ДАНИХ, ЯКІ ОТРИМАНІ В РЕЗУЛЬТАТІ ТЕСТУВАННЯ

5.1 Основні аспекти аналізу кодів

В даному розділі міститься інформація про отримані результати в рамках роботи програмного комплексу аналізу кодів.

Основними видами надання інформації будуть зрізи OLAP-куба і графіки, побудовані на підставі цих зрізів. Зрізи OLAP-куба допоможуть ознайомитися з чисельними значеннями досліджуваних параметрів, а графіки, побудовані в програмі Excel, - дозволять відстежити динаміку змін тих чи інших параметрів в залежності від обраного коду.

В даному розділі, будуть приведені коди лише двох типів: Код Хеммінга та Інверсний код. Характеристики аналізованих кодів наведені в таблиці 5.1.

Таблиця 5.1

Характеристики аналізованих кодів

Тип кода	Код Хеммінга	Код Хеммінга	Інверсний код	Інверсний код
Кількість інформаційних біт	8	16	8	16
Загальна кількість біт	12	21	16	32
Кількість контрольних біт	4	5	8	16
Специфіка роботи з помилками	Виправляє 1 помилку	Виправляє 1 помилку	Перевіряє наявність помилки	Перевіряє наявність помилки
Позначення	HEM(8,12,4)	HEM(16,21,5)	INV(8,16,8)	IND(16,32,16)

Таким чином, в порівнянні бере участь чотири коду двох типів. Також, Код Хеммінга має здатність виправляти одну помилку, а інверсний код діє за принципом перевірки коректності отриманих даних.

Загальний аналіз коду, який використовується далі, являє собою порівняння часових характеристик кожного коду за такими характеристиками, як час кодування інформаційної символної послідовності, час декодування закодованої послідовності і час декодування символної послідовності з помилкою.

5.2 Загальний аналіз Коду Хеммінга HEM(8,12,4).

Розглянемо зріз OLAP-куба, що демонструє можливості Коду Хеммінга, представлений на рисунку 5.1.

Code ▾			
HEM(8, 12, 4)			
Length ▾	Time	Time - Decodes	Time - Decodes Er
4	0,003	0,003	0,003
8	0,007	0,008	0,008
16	0,021	0,025	0,029
32	0,045	0,051	0,059
64	0,062	0,07	0,088
128	0,104	0,115	0,13
256	0,208	0,232	0,259
512	0,453	0,49	0,547

Рисунок 5.1 – Часові характеристики Коду Хеммінга

На підставі даного зрізу був отриманий графік, представлений на рисунку 5.2.

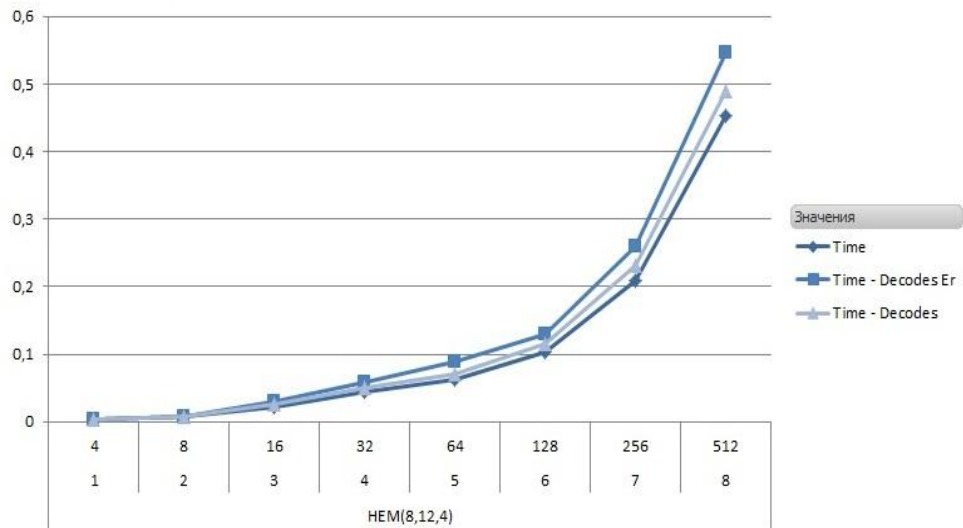


Рисунок 5.2 – Графік часових характеристик коду Хеммінга

На представлених матеріалах видно, що час виконання Кодом Хеммінга операцій кодування, декодування і декодування з помилкою збільшується прямо пропорційно. Тобто зі збільшенням довжини кодимуємої послідовності збільшується і час кожної операції.

Варто зауважити, що операції кодування є найбільш витратною з точки зору часового ресурсу. У той час як операція декодування без помилки вимагає меншої кількості часу.

5.3 Загальний аналіз інверсний коду INV(8,16,8).

Розглянемо зріз OLAP-куба, що демонструє можливості інверсного коду, представлений на рисунку 5.3.

Code ▾			
INV(8,16,8)			
Length ▾	Time	Time - Decodes	Time - Decodes Er
4	0,007	0,009	0,008
8	0,015	0,019	0,016
16	0,027	0,035	0,028
32	0,036	0,047	0,039
64	0,081	0,099	0,087
128	0,139	0,159	0,147
256	0,29	0,328	0,307
512	0,545	0,587	0,569

Рисунок 5.3 – Часові характеристики інверсний коду

На підставі даного зрізу був отриманий графік, представлений на рисунку 5.4.

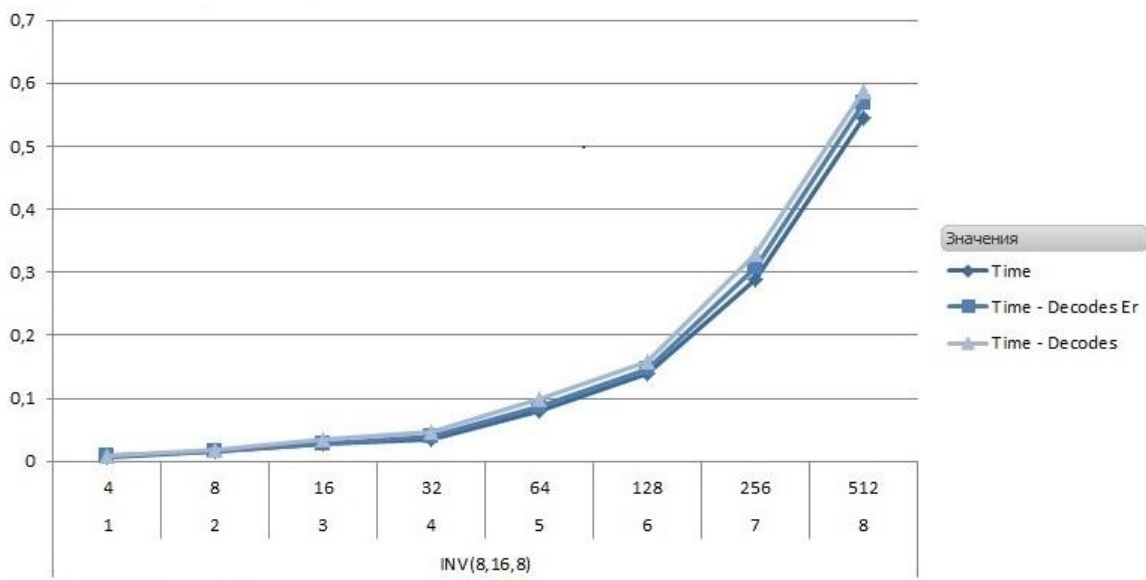


Рисунок 5.4 – Графік часових характеристик інверсного коду

На представлених матеріалах видно, що час виконання інверсний кодом операцій кодування, декодування і декодування з помилкою збільшується прямо пропорційно. Тобто зі збільшенням довжини кодуючої послідовності збільшується і час кожної операції.

Варто зауважити, що операції кодування є найбільш витратною з точки зору часового ресурсу. У той час як операція декодування без помилки вимагає найменшої кількості часу.

5.4 Загальний аналіз коду Хеммінга НЕМ(16,21,5).

Розглянемо зріз OLAP-куба, що демонструє можливості коду Хеммінга з 16-ю інформаційними символами, представлений на рисунку 5.5.

Code ▾			
HEM(16,21,5)			
Length ▾	Time	Time - Decodes	Time - Decodes Er
4	0,006	0,007	0,009
8	0,012	0,014	0,019
16	0,021	0,027	0,035
32	0,026	0,033	0,047
64	0,046	0,059	0,073
128	0,094	0,112	0,128
256	0,197	0,238	0,255
512	0,36	0,401	0,442

Рисунок 5.5 – Часові характеристики коду Хеммінга

На підставі даного зрізу був отриманий графік, представлений на рисунку 5.6.

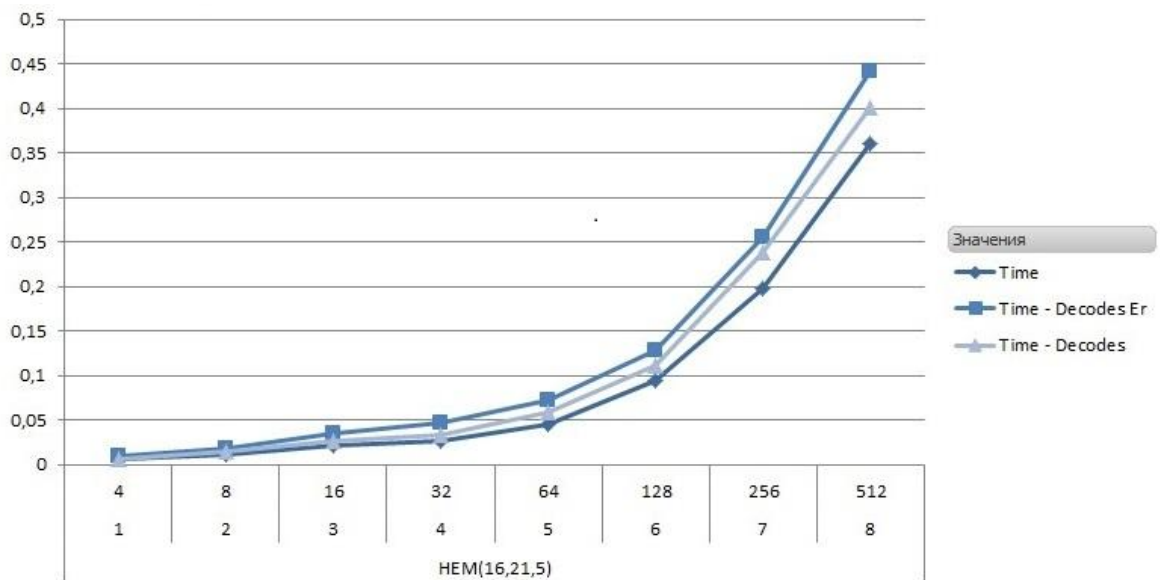


Рисунок 5.6 – Графік часових характеристик коду Хеммінга

Слід зазначити, що в коді Хеммінга з 16-ю інформаційними символами, аналогічно коду Хеммінга з 8-ю інформаційними символами, властива прямо пропорційна залежність між часом виконання операцій кодування, декодування з помилкою і декодування, і довжиною кодуючої послідовності.

Даний факт, обумовлюється хорошою масштабністю коду Хеммінга, тобто зі збільшенням інформаційних символів, а як наслідок і контрольних символів, час операцій над інформаційною символічною послідовністю зменшиться.

Як і у випадку з кодом Хеммінга з 8-ю інформаційними символами, в даному коді операція кодування є найбільш тривалою операцією, а операція декодування - найбільш швидкою.

5.5 Загальний аналіз інверсного коду INV(16,32,16).

Розглянемо зріз OLAP-куба, що демонструє можливості інверсного коду з 16-ю інформаційними символами, представлений на рисунку 5.7.

Code ▾			
INV(16,32,16)			
Length ▾	Time	Time - Decodes	Time - Decodes Er
4	0,004	0,004	0,004
8	0,014	0,021	0,017
16	0,029	0,037	0,031
32	0,035	0,05	0,045
64	0,079	0,099	0,078
128	0,146	0,169	0,156
256	0,291	0,335	0,301
512	0,566	0,615	0,587

Рисунок 5.7 – Часові характеристики інверсний коду

На підставі даного зрізу був отриманий графік, представлений на рисунку 5.8.

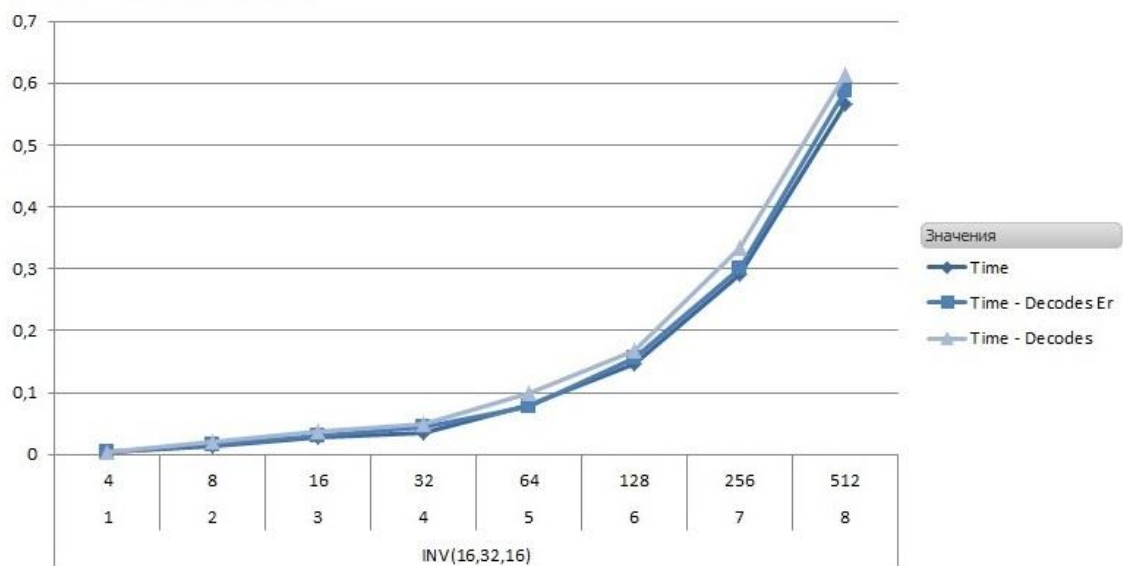


Рисунок 5.8 – Графік часових характеристик інверсного коду

Слід зазначити, що в інверсному коді з 16-ю інформаційними символами, аналогічно інверсний кодом з 8-ю інформаційними символами, властива прямо пропорційна залежність між часом виконання операцій кодування, декодування з помилкою і декодування, і довжиною кодимуваної послідовності.

Даний факт, обумовлюється масштабістю коду Хеммінга, тобто зі збільшенням інформаційних символів, а як наслідок і контрольних символів, час операцій над інформаційною символічною послідовністю зменшиться.

Як і у випадку з кодом Хеммінга з 8-ю інформаційними символами, в даному коді операція кодування є найбільш тривалою операцією, а операція декодування - найбільш швидкою.

5.6 Порівняння кодів НЕМ(8,12,4) и НЕМ(16,21,5)

Код Хеммінга НЕМ (16,21,5) є масштабованою копією коду НЕМ(8,12,4).

Переконаємося в даній гіпотезі за допомогою зрізу OLAP-куба, представленого на рисунку 5.9.

Length ▾	Code ▾		HEM(16, 21, 5)	
	HEM(8, 12, 4)	HEM(16, 21, 5)	Time	Time - Decodes
4	0,003	0,003	0,006	0,007
8	0,007	0,008	0,012	0,014
16	0,021	0,025	0,021	0,027
32	0,045	0,051	0,026	0,033
64	0,062	0,07	0,046	0,059
128	0,104	0,115	0,094	0,112
256	0,208	0,232	0,197	0,238
512	0,453	0,49	0,36	0,401

Рисунок 5.9 – Характеристики кодів НЕМ(8,12,4) та НЕМ(16,21,5)

На наведеному зрізі куба можна помітити, що час виконання операцій кодування і декодування кодом НЕМ (16,21,5) однозначно менше, ніж кодом НЕМ (8,12,4). На підставі цього, можна припустити, що збільшивши кількість

інформаційних символів, що надходять за один раз на «Кодер», і відповідно, збільшивши кількість контрольних біт, можна домогтися зменшення часу операцій кодування і декодування.

Залежність між часом виконання операцій кодування, декодування і декодування з помилкою, і довжиною оброблюваної інформаційної послідовності є однаковою - прямо пропорційною. Перевіримо це за допомогою графіка, представленого на рисунку 5.10.

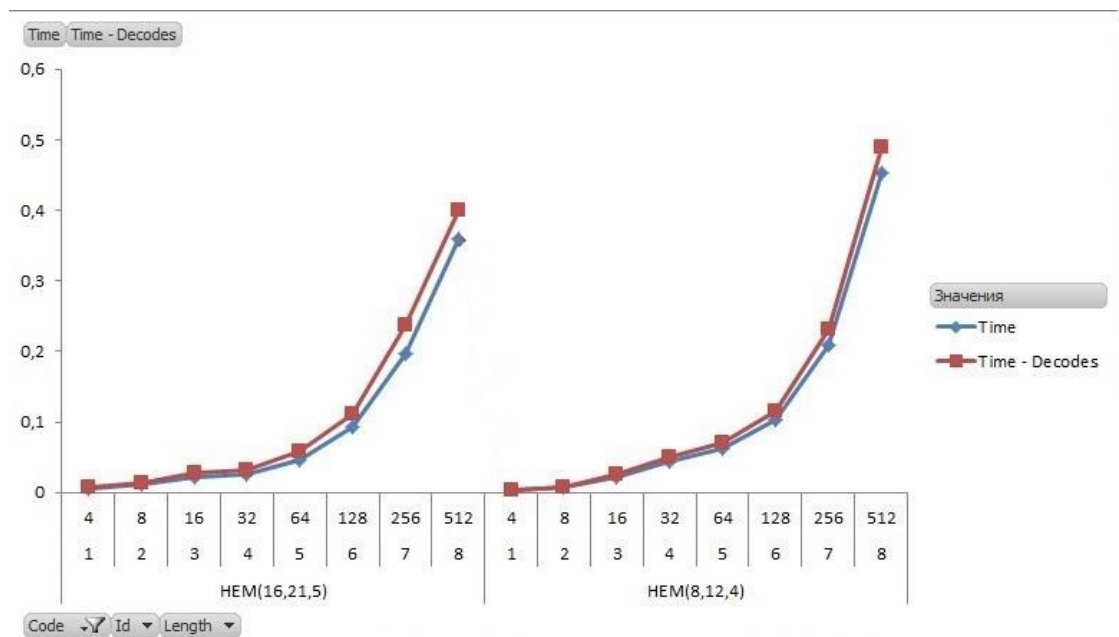


Рисунок 5.10 – Графік залежності операцій кодів від довжини оброблюваної послідовності

Дійсно, характер кривої для обох кодів однаковий. Цей факт підтверджує вищевикладене твердження.

5.7 Порівняння кодів INV(8,16,8) та INV(16,32,16)

Інверсний код INV (16,32,16) є масштабованою версією коду INV (8,16,8). Спробуємо переконатися в даній гіпотезі за допомогою зрізу OLAP-куба, представленого на рисунку 5.11.

Code ▾				
INV(16,32,16)			INV(8,16,8)	
Length ▾	Time	Time - Decodes	Time	Time - Decodes
4	0,004	0,004	0,007	0,009
8	0,014	0,021	0,015	0,019
16	0,029	0,037	0,027	0,035
32	0,035	0,05	0,036	0,047
64	0,079	0,099	0,081	0,099
128	0,146	0,169	0,139	0,159
256	0,291	0,335	0,29	0,328
512	0,566	0,615	0,545	0,587

Рисунок 5.11 – Характеристики кодів INV(8,16,8) та INV(16,32,16)

На наведеному зрізі куба можна помітити, що час виконання операцій кодування і декодування кодом НЕМ (16,21,5) відрізняється вельми несуттєво від тих же операцій коду INV (8,16,8). Виходячи з цього, можна судити про низьку ефективність даного коду, оскільки зі збільшенням кількості інформаційних і контрольних біт, тимчасові показники змінюються дуже слабо.

На підставі цього, можна припустити, що збільшивши кількість інформаційних символів, що надходять за один раз на «Кодер», і відповідно, збільшивши кількість контрольних біт, домогтися краще продуктивності цього коду не вдасться, принаймні, при використаній реалізації.

Залежність між часом виконання операцій кодування, декодування і декодування з помилкою, і довжиною оброблюваної інформаційної послідовності є однаковою - прямо пропорційною. Покажемо це за допомогою графіка, представленого на рисунку 5.12.

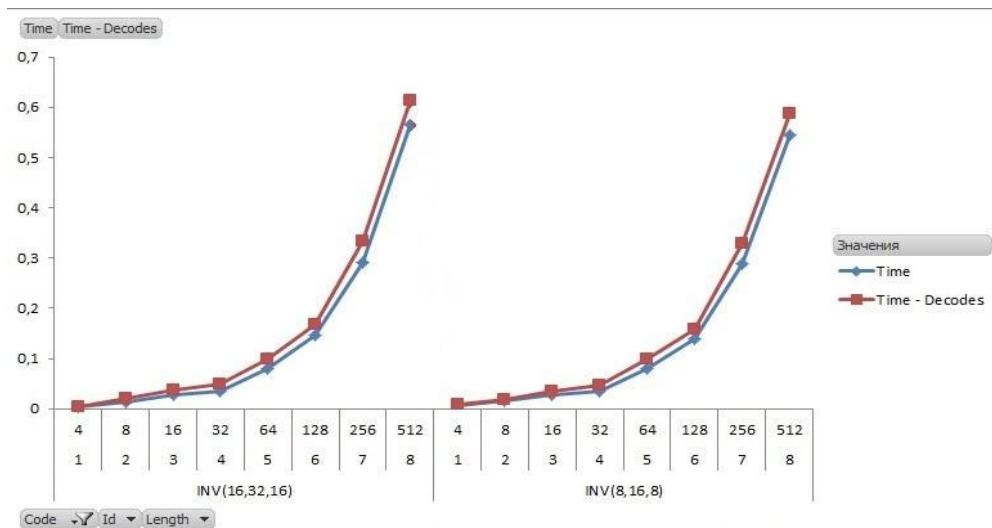


Рисунок 5.12 – Графік залежності операцій кодів від довжини оброблюваної послідовності

Дійсно, характер кривої для обох кодів однаковий. Цей факт підтверджує вищевикладене твердження.

5.8 Порівняння кодів НЕМ (8,12,4) та INV (8,16,8)

Подібність кодів НЕМ (8,12,4) і INV (8,16,8) полягає в тому, що вони кодують по 8 бітів кожен. Різниця цих кодів складається в різній кількості контрольних символів і як наслідок різній кількості символів в закодованій послідовності.

Розглянемо час, витрачений на операції кодування і декодування цими кодами. Відповідний зріз OLAP-куба представлений на рисунку 5.13.

Code ▾				
		INV(8,16,8)		HEM(8,12,4)
Length ▾		Time	Time - Decodes	Time Time - Decodes
4		0,007	0,009	0,003 0,003
8		0,015	0,019	0,007 0,008
16		0,027	0,035	0,021 0,025
32		0,036	0,047	0,045 0,051
64		0,081	0,099	0,062 0,07
128		0,139	0,159	0,104 0,115
256		0,29	0,328	0,208 0,232
512		0,545	0,587	0,453 0,49

Рисунок 5.13 – Порівняння кодів НЕМ(8,12,4) та INV(8,16,8)

Очевидно, що час кодування і декодування у коду Хеммінга значно менше часу кодування і декодування інверсного коду. На підставі цього можна зробити висновок, що, як мінімум, з порівнюваними параметрами код Хеммінга є більш ефективним з точки зору часу виконання своїх операцій. Також слід зазначити, що код Хеммінга при декодуванні здатний виявляти і виправляти помилку, а Інверсний код може лише з'ясувати, чи є в конкретній послідовності помилка чи ні. Коригувальною властивістю Інверсний код не володіє. Графік на рисунку 5.14 демонструє схожий характер залежностей часу виконання операцій і довжини оброблюваних послідовностей.

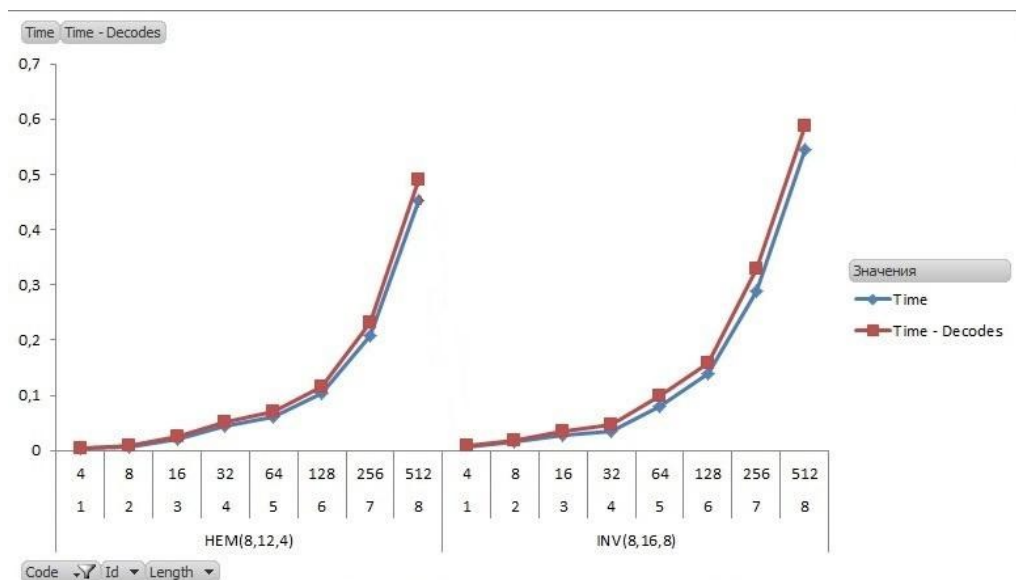


Рисунок 5.14 – Порівняння кодів HEM(8,12,4) та INV(8,16,8)

5.9 Порівняння кодів HEM(16,21,5) та INV(16,32,16)

Подібність кодів HEM (16,21,5) і INV (16,32,16) полягає в тому, що вони кодують по 16 бітів кожен. Різниця цих кодів складається в різній кількості контрольних символів і як наслідок різній кількості символів в закодованій послідовності. Розглянемо час, витрачений на операції кодування і декодування цими кодами. Відповідний зріз OLAP-куба представлений на рисунку 5.15.

	Code ▾			
	HEM(16,21,5)		INV(16,32,16)	
Length ▾	Time	Time - Decodes	Time	Time - Decodes
4	0,006	0,007	0,004	0,004
8	0,012	0,014	0,014	0,021
16	0,021	0,027	0,029	0,037
32	0,026	0,033	0,035	0,05
64	0,046	0,059	0,079	0,099
128	0,094	0,112	0,146	0,169
256	0,197	0,238	0,291	0,335
512	0,36	0,401	0,566	0,615

Рисунок 5.15 – Порівняння кодів НЕМ(16,21,5) та INV(16,32,16)

Виходячи з наявних даних, можна говорити про те, що код Хеммінга з даними параметрами є більш ефективним, ніж Інверсний код. Більш того, ґрунтуючись на наявних даних, код Хеммінга швидше інверсний коду майже в два рази. Слід також зазначити, що різниця між тимчасовими показниками даних кодів здатна змінюватися в більшу або меншу сторону в залежності від реалізації того чи іншого коду, або від тестування на інших апаратних платформах.

Рисунок 5.16 демонструє схожий характер залежностей часу виконання операцій і довжини оброблюваних послідовностей, а також різницю між часом виконання операцій кодування і декодування у кодів НЕМ (16,21,5) і INV (16,32,16).

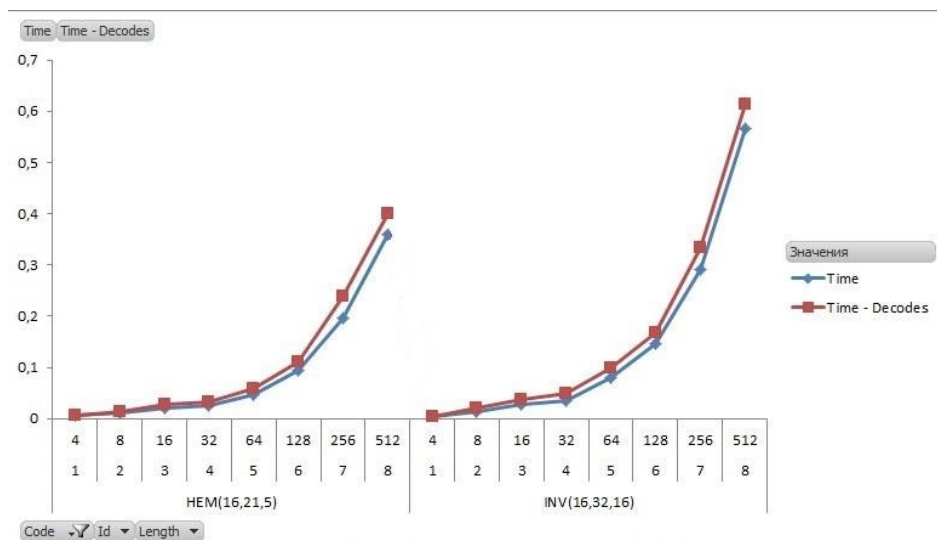


Рисунок 5.16 - Порівняння кодів НЕМ(16,21,5) та INV(16,32,16)

5.10 Висновок по розділу

Представлений в даному розділі аналіз за допомогою розробленого програмного комплексу, дозволив порівняти чотири коду двох типів, а саме код Хеммінга і Інверсний код. Коди порівнювалися за такими параметрами, як час кодування, час декодування і час декодування з помилкою. Ці операції здійснювалися над вісьмома інформаційними символьними послідовностями різної довжини - від 2 до 512.

В результаті аналізу було виявлено переваги і недоліки кожного з чотирьох кодів, наприклад код INV (16,32,16) за кількістю інформаційних і контрольних біт в два рази перевищує аналогічні параметри коду INV (8,16,8). Однак виконує необхідні операції майже за ті ж тимчасові проміжки, що і код INV (8,16,8). Цим обумовлюється недоцільність використання даного типу кодів, в порівнянні з кодом Хеммінга. Винятком може слугувати ситуація, при якій є необхідними специфічні можливості інверсний коду.

Також було визначено, що найменше часу на роботу витрачає код Хеммінга НЕМ (16,21,5). Цей код в даній ситуації можна назвати найбільш прийнятним з точки зору часу виконання операцій кодування і декодування. Варто також відзначити, що код НЕМ (16,21,5) є версією коду НЕМ (8,12,4), але з чисельно збільшеними параметрами. Як показали дослідження даних кодів, код НЕМ (16,21,5) працює набагато швидше коду НЕМ (8,12,4). Це дозволяє вважати, що при збільшенні характеристик коду цього типу, зменшиться час кодування. Інакше кажучи, зі збільшенням числа інформаційних і контрольних біт, зросте і продуктивність даного коду.

Завдання порівняння декількох завадостійких кодів за допомогою спеціально розробленого програмного комплексу була виконана, оскільки аналітичним шляхом були визначені плюси і мінуси кожного коду, і їх відмінності.

ВИСНОВОК

У даній роботі був реалізований програмний комплекс для тестування і аналізу завадостійких кодів. У комплекс увійшли чотири компоненти, написані спеціально для нього, і також були задіяні два сторонніх компонента у вигляді програмних пакетів.

Цей програмний комплекс дозволив провести порівняння чотирьох кодів двох типів, а саме двох реалізацій коду Хеммінга і двох реалізацій інверсного коду. В ході тестування і подальшого аналізу були виявлені переваги і недоліки досліджуваних кодів.

Перевагою такого підходу є проведення аналізу на основі реальних даних, отриманих в результаті роботи кожного коду.

Також, слід зазначити, що даний комплекс покликаний лише емулювати реальні процеси кодування і передачі даних, оскільки операції кодування і декодування реалізовані на програмному рівні. Існує ймовірність, що апаратна реалізація операцій кодування і декодування може істотно чи несуттєво відрізнятися від реалізації, представленої в даній роботі.

Однак при схожих підходах в реалізації як програмної, так і апаратної, слід очікувати схожий характер тих чи інших залежностей, виведених в аналітичній частині роботи.

Крім цього, результати, отримані в даній роботі на підставі використання програмного комплексу для тестування і аналізу кодів, є відносними, оскільки були отримані в результаті роботи конкретного комп'ютера з конкретними технічними характеристиками, а головне - процесором. Тому справедливо стверджувати, що час, витрачений аналогічними кодами на операції кодування і декодування, може змінюватися в одну або іншу сторону, в залежності від технічних характеристик використовуваної платформи.

Одним з основних переваг розробленого програмного комплексу є його універсальність з точки зору використання інших кодів. Його архітектура

дозволяє тестувати, аналізувати і порівнювати різні реалізації різних завадостійких кодів в однакових умовах. Це досягається за рахунок специфіки виклику кодують і декодер функцій: механізм виклику функції майже ніяк не пов'язаний з реалізацією самої функції і дозволяє подавати одну й ту ж послідовність на вхід безлічі різних функцій.

Надалі планується вдосконалити алгоритми кодування і декодування, які використовувалися, а також налагодити більш автоматизовану міжкомпонентну взаємодію, яка потребує якомога меншого втручання користувача.

ПЕРЕЛІК ПОСИЛАНЬ

1. Электромагнитная помеха [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BC%D0%B0%D0%B3%D0%BD%D0%B8%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BC%D0%B5%D1%85%D0%B0 .
2. Внешние помехи - Большая Энциклопедия Нефти Газа [Электронный ресурс] – Режим доступа: <http://www.ngpedia.ru/id289586p1.html> .
3. Внутренние помехи - Большая Энциклопедия Нефти Газа [Электронный ресурс] – Режим доступа: <http://www.ngpedia.ru/id289588p1.html> .
4. Кутузаки А.С., Гунченко Ю.А. Анализ проблем разбиения на кадры данных // Тезисы доклада Десятой всеукраинской конференции студентов и молодых специалистов «ИНФОРМАТИКА, ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ». – Одесса: - 2013. – с. 193 – 195.
5. Блочный код - Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4 .
6. Линейные блочные коды [Электронный ресурс] – Режим доступа: http://sernam.ru/book_tec.php?id=80 .
7. Понятие метрического пространства [Электронный ресурс] – Режим доступа: http://sernam.ru/lect_math3.php?id=16 .
8. Кутузаки А.С. Использование принципа расстояния Хэмминга для последовательностей разной длины // Тезисы доклада Четвёртой Международной научно-практической конференции «ОБРАБОТКА СИГНАЛОВ И НЕГАУСОВСКИХ ПРОЦЕССОВ». – Черкассы.
9. Коды с обнаружением ошибок [Электронный ресурс] – Режим доступа: http://informkod.narod.ru/5_1item.htm .
10. Кутузаки А.С., Гунченко Ю.О. Аналіз проблем розбиття на кадри 100 даних // Тези доповідей Десятої всеукраїнської конференції студентів і

молодих спеціалістів «ІНФОРМАТИКА, ІНФОРМАЦІЙНІ СИСТЕМИ І ТЕХНОЛОГІЇ». – Одеса: - 2013. – с. 193 – 195.

11. Банкет В. Л. Завадостійке кодування в телекомунікаційних системах / В. Л. Банкет, П. В. Іващенко, М. О. Іщенко. – Одеса : ОНАЗ ім. О. С. Попова, 2011. – 100 с.

12. OLAP - Википедия [Електронний ресурс] – Режим доступу: <https://ru.wikipedia.org/wiki/OLAP> .

13. OLAP-системы конспект лекций / Одесский Национальный Университет им. И.И. Мечникова – М.: Одесса, 2012.- 20 с.

14. Мартинович Л. Я., Аль-Омарі А. О., Слуцький Д. В., Смірнов К. О. Методи завадостійкого кодування у цифрових системах «Інформатика, інформаційні системи та технології: тези доповідей вісімнадцятої всеукраїнської конференції студентів і молодих науковців». Одеса, 23 квітня 2021 р. - Одеса, 2021. – с.135-137.

15. Гунченко Ю.О., Мартинович Л.Я., Аль-Омарі А.О. Аналіз завадостійких кодів за допомогою програмного комплексу Тези доповідей Всеукраїнської науково-практичної конференції молодих вчених, ад'юнктів, слухачів, курсантів і студентів "Молодіжна військова наука у Київському національному університеті імені Тараса Шевченка" / за заг. редакцією І.В. Толока. – К. : ВІКНУ, 2021. – с. 99-100.

Одеський національний університет ім. І.І. Мечникова

ПОБУДОВА ТА АНАЛІЗ ЗАВАДОСТІЙКИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ



Виконав студ. Аль-Омарі А.О.
Керівник проф. Гунченко Ю.О.

Одеса - 2021

АКТУАЛЬНІСТЬ

Актуальність роботи обумовлена актуальністю питання надійності зберігання та передачі інформації

Завадостійке кодування застосовується при архівації даних, передачі даних в пакетних мережах, в супутникових і в стільникових системах зв'язку

Мета роботи - створення програмного комплексу для тестування і аналізу кодів.

Завдання програмного комплексу - надання можливостей для порівняння різних типів кодів за часовими характеристиками.

На основі роботи комплексу можна виконувати аналітичні порівняння, враховуючи динаміку змін показників кожного коду і порівняння різних кодів за одними й тими ж параметрами. Отримана інформація може надаватися у вигляді зрізів OLAP-куба і графіків

ПОСТАНОВКА ЗАДАЧІ

- Реалізувати програмний комплекс для тестування і аналізу завадостійких кодів. Компоненти програмного комплексу повинні здійснювати
 - кодування символічних послідовностей
 - декодування і
 - декодування послідовності зі штучно згенерованою помилкою
- Спроекувати базу даних і заповнити її результатами У базу даних повинна заносятися інформація про час кодування декодування і декодування послідовності з помилкою для кожного коду, а також тип коду.
- Агрегувати інформацію що міститься в базі даних за допомогою аналітичного програмного пакета SQL Server Analysis Services
- Проаналізувати дані, отримані в результаті роботи програмного комплексу Аналіз повинен включати порівняння досліджуваних кодів за такими параметрами як тип алгоритму кількість інформаційних символів швидкість роботи
- Візуалізувати отримані залежності необхідно за допомогою графіків, що ілюструють часові значення різних типів кодів при різних значеннях параметрів - довжини обробленої інформаційної символічної послідовності і типи кодів.

3

КЛАСИФІКАЦІЯ ПЕРЕШКОД

За певними ознаками:

- 1) Атмосферні: космічне випромінювання, розряди блискавок.
- 2) Індустріальні: розряди на лінії електропередач, перешкоди від комутаційних процесів.
- 3) Штучні (спеціально організовані).

За своїми властивостями:

- 1) Імпульсні перешкода - випадкова послідовність імпульсів довільної форми, тривалості і амплітуди. параметри окремих імпульсів в заваді (амплітуда, тривалість і т.д.) дуже близькі до параметрів робочого сигналу, який переносить інформацію.
- 2) Флуктуаційна (шумова) перешкода - безперервне випадкове (хаотичне) коливання гладкого типу.
- 3) Синусоїдальна перешкода - сукупність відрізків синусоїд випадкової тривалості, фази, частоти, амплітуди.

4

КЛАСИФІКАЦІЯ ПЕРЕШКОД

- Перешкоди бувають зовнішні і внутрішні.
- Зовнішні перешкоди - це перешкоди, створені коливаннями навколишнього середовища від незалежних, для конкретного каналу, джерел. Наприклад, вони можуть генеруватися за рахунок електростатичних електромагнітних полів, створюваних сусідніми пристроями такими як електричний транспорт, електрообладнання комп'ютерна електроніка і т.п. Зовнішні перешкоди створюються коливаннями навколишнього середовища від незалежних джерел і передаються на коливну систему через зв'язку, їх частотний спектр для конкретних умов можна вважати заданим.
- Зовнішні перешкоди часто носять одиночний або слабо рецидивний характер, тому їх втручання в дискретний канал зв'язку слабо прогнозовано щодо тимчасової шкали.
- Такі перешкоди можуть мати високий рівень, іноді навіть перевершує рівень сигналу. Тривалість перешкод в такому випадку на кілька порядків менше тривалості сигналу. Внутрішні перешкоди - це перешкоди, що генеруються обчислювальними пристроями безпосередньо беруть участь у процесі передачі даних. Тобто вони виникають в самій апаратурі передачі інформації
- До них можна віднести перешкоди у вигляді теплових шумів електронних ламп, напівпровідникових приладів, опір і інших елементів. Також вони можуть бути викликані перетворенням сигналів в окремих елементах системи (шуми квантування, нелінійність характеристик перетворення), нестабільністю апаратури її несправністю
- Внутрішні перешкоди - це такі, що хаотично змінюються (флуктуаційні) напруги і струми, які виникають в опорах і підсилюючих елементах схеми. Джерелами флуктуаційних перешкод є всі підсилювальні елементи і активні опори підсилювача.

5

МЕТОДИ БОРОТЬБИ З ПЕРЕШКОДАМИ

- Методи завадостійкого кодування – це один з найбільш ефективних способів підвищення завадостійкості при передачі інформації.
- Суть подібних методів полягає у використанні кодів, що виявляють помилки, і кодів, що виправляють помилки. Деякі коди можуть поєднувати в собі обидві функції: функцію пошуку і функцію виправлення. Дану групу кодів часто називають коригуючими кодами.
- Практично будь-який код, що виправляє помилки, може використовуватися для їх виявлення.

6

БЛОКОВІ ТА ЗГОРТКОВІ КОДИ

- Блокові ділять вхідний потік даних на фрагменти заданої незмінної довжини, які обробляються послідовно,
- згорткові - працюють з безперервним потоком даних.

- Блоковий код - код, який кодує послідовності з набору символів алфавіту S в кодові слова, перетворюючи окремо кожен символ шуканого алфавіту.

$(k_1, k_2, \dots, k_m)$ - послідовність натуральних чисел, де кожне менше $|S|$.

Якщо $S = (s_1, s_2, \dots, s_n)$ і деяке слово W з алфавіту S записано як

$$W = a_{k1} a_{k2} \dots a_{km}$$

тоді кодовим словом, відповідним W , а саме, $C(W)$ буде:

$$C(W) = C(s_{k1}) C(s_{k2}) \dots C(s_{km}).$$

7

ЛІНІЙНІ КОДИ

- Лінійний код загального вигляду - це блоковий код довжиною n символів, що складається з $2k$ кодових слів. Такий код також називається лінійним (n, k) -кодом, за умови, що всі його $2k$ кодових слів утворюють k -мірний підпростір векторного простору n -послідовностей поля $GF(2)$. Інакше кажучи, двійковий код може називатися лінійним, якщо сума по модулю 2 ($\text{mod } 2$) двох кодових слів також є кодовим словом цього коду.

- Ефективність лінійного коду визначається трьома факторами:

- кількістю надлишкової інформації, необхідної для належного функціонування коду по його алгоритму,
- кількістю помилок, які код може виправити і
- складністю реалізації операції кодування і декодування (як апаратної, так і у вигляді програмного забезпечення)

8

ВІДСТАНЬ ХЕММІНГА. КОД ХЕММІНГА

- Відстань Хеммінга - це число позицій, в яких відповідні символи двох слів однакової довжини різні. Наприклад:

➤ $\Delta(0001001, 1001111) = 3$

➤ $\Delta(2340A, 1142B) = 4$

➤ $\Delta(\text{завод}, \text{закон}) = 2$

Код Хеммінга - один з перших коригувальних кодів. Він є систематичним кодом з перевіркою на парність і дає можливість знаходити як поодинокі, так і подвійні помилки, а виправляти лише поодинокі помилки. Код Хеммінга має властивості блокового (n, k) -коду, де n - сумарна кількість розрядів, з них m розрядів - інформаційні, а k розрядів використовуються в якості контрольних (перевірочних). $n = m + k$. В таблиці показана кількість необхідних контрольних розрядів для m інформаційних.

m	1	2	3	4	5	6	7	8	9	10	11	12
k	2	3	3	3	4	4	4	4	4	4	4	5
n	3	5	6	7	9	10	11	12	13	14	15	17

9

ІНВЕРСНИЙ КОД

- Це один з найпростіших блокових лінійних кодів. Його суть полягає в додаванні до вихідної комбінації такої ж по довжині комбінації, в якій містяться контрольні біти. Контрольна комбінація генерується за принципом:

- якщо у вихідній комбінації парне число одиниць, то додається комбінація, яка повністю дублює вихідну,
- якщо непарне, то комбінація, що додається містить інверсні значення відповідних вихідних значень.

Наприклад, комбінація 10101010 матиме контрольну послідовність 10101010, вихідна комбінація 11111000 матиме контрольну послідовність 00011111.

Декодування інверсного коду здійснюється в два етапи:

1. підсумовуються одиниці в першій групі символів. Якщо число одиниць парне, то послідовність контрольних бітів змінена не буде, якщо ж число одиниць непарне, то послідовність контрольних бітів буде інвертована.
2. порівнюються знову обчислені контрольні біти і інформаційні біти. Якщо всі біти на відповідних позиціях збігаються, інформація прийшла в незміненому вигляді. Розбіжність говорить про те, що при передачі була помилка.

10

ПЕРЕВАГИ ТА НЕДОЛІКИ КОДІВ

- ✓ захист даних з використанням коду Хеммінга за рахунок паралельної роботи вимагає порівняно невеликих апаратних затрат і не призводить до великої втрати швидкості.
- ✓ помилки, які виникають в пам'яті, можуть бути зареєстровані, незважаючи на те, що через корегування вони залишаються непомітними.
- ✓ дає можливість знаходити як поодинокі, так і подвійні помилки, а виправляти лише поодинокі помилки

Перевагою інверсного коду є хороша здатність виявлення. Даний код виявляє практично будь-які помилки, виключаючи дуже рідкісні помилки зсуву, які відбуваються як серед інформаційних бітів, так і серед контрольних.

Недоліком даного коду можна вважати велику кількість контрольних біт, які «обтяжують» закодовану послідовність

11

ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

Основні аспекти розробки програмного комплексу для аналізу коригувальних кодів:

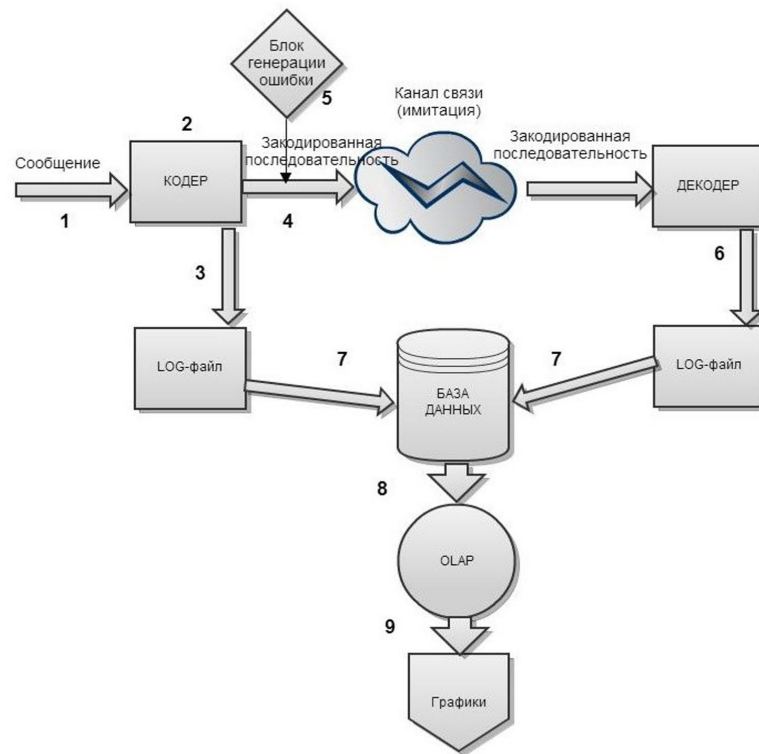
1. Реалізація здійснюється у вигляді дискретних модулів, пов'язаних між собою
2. Кожен модуль має своє призначення
3. Взаємодія між модулями автоматизована
4. Основою для аналізу будуть час кодування, розмір кодового слова і довжина вхідної символічної послідовності.

Комплекс складається з таких модулів: «Кодер», «декодер», блок генерації помилок, база даних та модуль запису до неї.

Розробка здійснювалась в середовищі Visual Studio компанії Microsoft. Мовою програмування обрано C++. Мова програмування C# використовується для написання модуля, що взаємодіє з базою даних.

12

ВЗАЄМОДІЯ МОДУЛІВ В КОМПЛЕКСІ



13

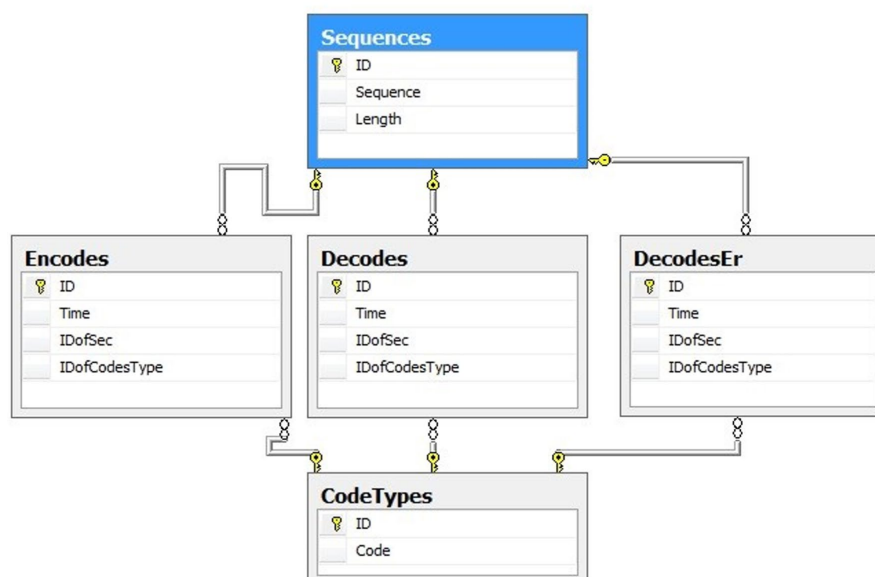
ЖИТТЄВИЙ ЦИКЛ ПОВІДОМЛЕННЯ

- 1) Повідомлення в символьному вигляді подається в модуль, який здійснює кодування (модуль «Кодер»)
- 2) Модуль «Кодер» створює бінарне представлення символів повідомлення на основі їх порядкового номера в таблиці ASCII
- 3) Потім модуль «Кодер» виконує відповідні потрібного коду (наприклад, коду Хеммінга) алгоритмічні операції і записує дані про довжину кодованого слова і часу кодування в лог-файл
- 4) «Кодер» посилає отриману закодовану бінарну послідовність модулю «декодер» через канал зв'язку
- 5) Потім, в залежності від необхідності спрацьовує блок генерації помилок
- 6) «декодер» обробляє отримане повідомлення за потрібне алгоритмом і заносить в лог-файл дані, аналогічні даним кодує боку
- 7) На підставі лог-файлів модулів «Кодер» і «декодер» інший модуль реалізує запис в базу даних
- 8) База даних обробляється за допомогою функціональних можливостей OLAP-аналізатора
- 9) Реалізується графічне представлення отриманих даних

14

СТРУКТУРА БАЗИ ДАНИХ

База даних складається з п'яти таблиць. Це таблиці Sequences, CodeTypes, Encodes, Decodes і DecodesEr, призначені для запису в них даних послідовностей для кодування, алгоритмів і результатів кодування, декодування і декодування з помилкою, відповідно .



15

СХЕМА ТАБЛИЦЬ SEQUENCES ТА CODETYPES

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Sequence	nvarchar(1100)	Поле, що містить текст кодованої послідовності
3.	Length	int	Поле, що містить довжину кодованої послідовності

№	Ім'я поля	Тип даних	Опис
1.	ID <pk>	int	Поле унікального ідентифікатора
2.	Code	nvarchar(100)	Поле, що містить назву типу алгоритму

16

АЛГОРИТМ РОБОТИ МОДУЛЯ «КОДЕР» ТА ГЕНЕРАЦІЇ ПОМИЛОК

- 1) Функція кодування приймає на вхід символ, який необхідно закодувати
- 2) Символьне уявлення перетворюється в бітовий масив, на підставі таблиці ASCII
- 3) Над двійкового масивом виконуються маніпуляції, відповідні алгоритму кодування
- 4) Функція кодування повертає закодований масив одиниць і нулів, що містить як інформаційні, так і контрольні біти. Даний масив згодом буде використовуватися для декодування.

1) Один з елементів закодованої послідовності (символьного масиву), отриманої функцією декодування, перевіряється умовним оператором if

2) Якщо значення досліджуваного елемента масиву дорівнює 0, то воно змінюється на 1

3) Інакше, якщо значення досліджуваного елемента дорівнює 1, то воно змінюється на значення 0

4) Символьний масив зі зміненим значенням одного елемента подається на подальше декодування і виправлення помилки

Інакше кажучи, блок генерації помилок інвертує один із символів закодованої послідовності.

17

ПРИКЛАД ЗАПИСУ В БАЗУ ДАНИХ В ТАБЛИЦЮ DECODES

```

70     foreach (var item in list)
71     {
72         string[] s = item.Split(new Char[] { ';' });
73         double Time = Convert.ToDouble(s[2]);
74         string CodeType = s[3];
75
76         string query = String.Format("INSERT INTO Decodes (Time, IDofCodesType) "
77             + "Values ({0}, '{0}']", Time, CodeType);
78
79         SqlCommand command = new SqlCommand(query, connection);
80         command.ExecuteNonQuery();
81     }

```

18

ПРИКЛАД ТЕСТОВИХ ПОСЛІДОВНОСТЕЙ

ID	Sequence	Length
5	its	4
7	its time	8
8	its time to	12
9	its time to go tim	18
10	its time to go time for	23
11	its time to go time for the	27
12	its time to go time for the fre	31
13	its time to go time for the freedom	35
14	its time to go time for the freedom tim	39
15	its time to go time for the freedom time to	43
16	its time to go time for the freedom time to fac	47
17	its time to go time for the freedom time to face wh	51

19

ВМІСТ ТАБЛИЦІ, ЗАСНОВАНИЙ НА РОБОТІ МОДУЛЯ «КОДЕР» З ТЕСТОВИМИ ДАНИМИ

ID	Time	IDofSec	IDofCodesType
21	0,016	6	1
22	0,018	7	1
23	0,044	8	1
24	0,048	9	1
25	0,061	10	1
26	0,064	11	1
27	0,076	12	1
28	0,086	13	1
29	0,107	14	1
30	0,109	15	1
31	0,127	16	1

20

ВМІСТ ТАБЛИЦЬ DECODES TA DECODESER

ID	Time	IDofSec	IDofCodesType	ID	Time	IDofSec	IDofCodesType
21	0,001	6	1	21	0,002	6	1
22	0,003	7	1	22	0,003	7	1
23	0,006	8	1	23	0,009	8	1
24	0,009	9	1	24	0,01	9	1
25	0,01	10	1	25	0,012	10	1
26	0,015	11	1	26	0,016	11	1
27	0,0017	12	1	27	0,022	12	1
28	0,018	13	1	28	0,026	13	1
29	0,023	14	1	29	0,028	14	1
30	0,027	15	1	30	0,03	15	1
31	0,029	16	1	31	0,035	16	1
NULL	NULL	NULL	NULL				

21

ВАРІАНТ ЗРІЗУ КУБА

Code ▼			
HEM8			
Sequence ▼	Time	Time - Decodes	Time - Decodes Er
its	0,016	0,001	0,002
its time	0,018	0,003	0,003
its time to	0,044	0,006	0,009
its time to go tim	0,048	0,009	0,01
its time to go time for	0,061	0,01	0,012
its time to go time for the	0,064	0,015	0,016
its time to go time for the fre	0,076	0,0017	0,022
its time to go time for the freedom	0,086	0,018	0,026
its time to go time for the freedom tim	0,107	0,023	0,028
its time to go time for the freedom time to	0,109	0,027	0,03
its time to go time for the freedom time to fac	0,127	0,029	0,035

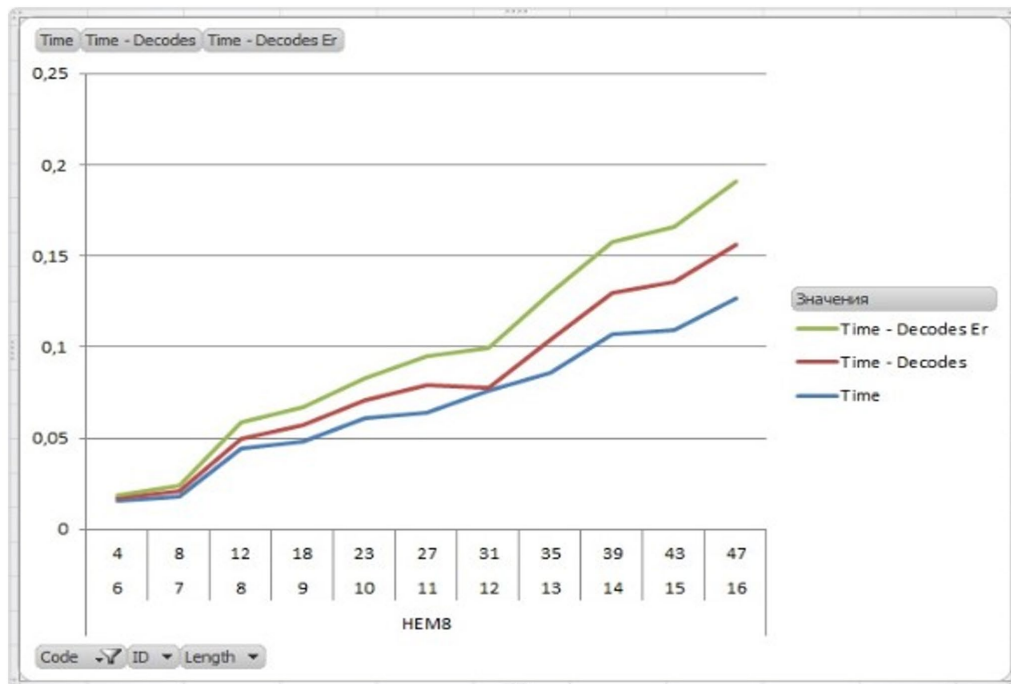
22

ТАБЛИЦЯ З ДАНИМИ ЗІ ЗРІЗУ КУБА

	A	B	C	D
1	Названия строк	Time	Time - Decodes	Time - Decodes Er
2	HEM8			
3	its	0,016	0,001	0,002
4	its time	0,018	0,003	0,003
5	its time to	0,044	0,006	0,009
6	its time to go tim	0,048	0,009	0,01
7	its time to go time for	0,061	0,01	0,012
8	its time to go time for the	0,064	0,015	0,016
9	its time to go time for the fre	0,076	0,0017	0,022
10	its time to go time for the freedom	0,086	0,018	0,026
11	its time to go time for the freedom tim	0,107	0,023	0,028
12	its time to go time for the freedom time to	0,109	0,027	0,03
13	its time to go time for the freedom time to fac	0,127	0,029	0,035

23

АНАЛІЗ ДАНИХ



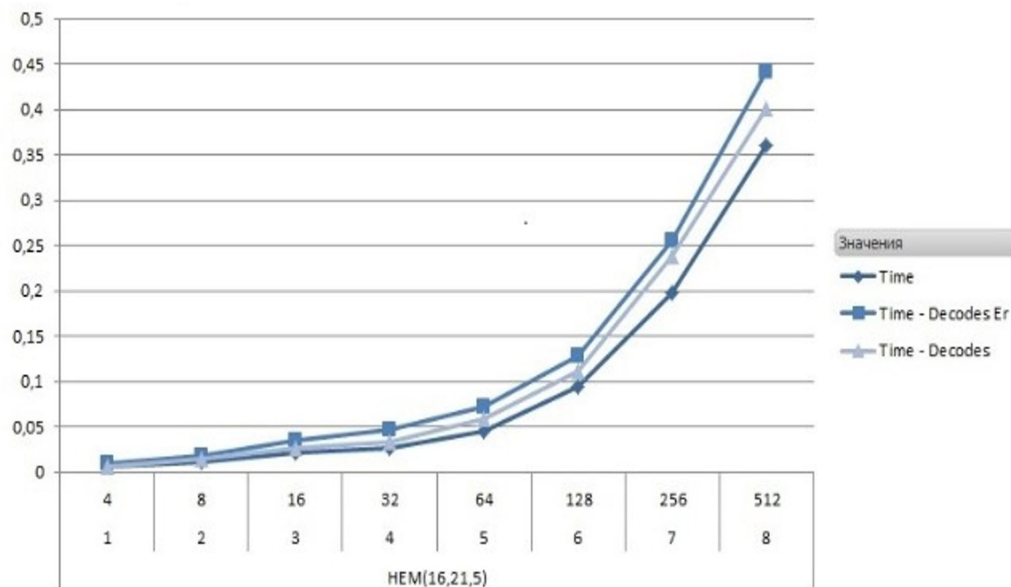
24

ХАРАКТЕРИСТИКИ АНАЛІЗОВАНИХ КОДІВ

Тип кода	Код Хеммінга	Код Хеммінга	Інверсний код	Інверсний код
Кількість інформаційних біт	8	16	8	16
Загальна кількість біт	12	21	16	32
Кількість контрольних біт	4	5	8	16
Специфіка роботи з помилками	Виправляє 1 помилку	Виправляє 1 помилку	Перевіряє наявність помилок	Перевіряє наявність помилок
Позначення	HEM(8,12,4)	HEM(16,21,5)	INV(8,16,8)	IND(16,32,16)

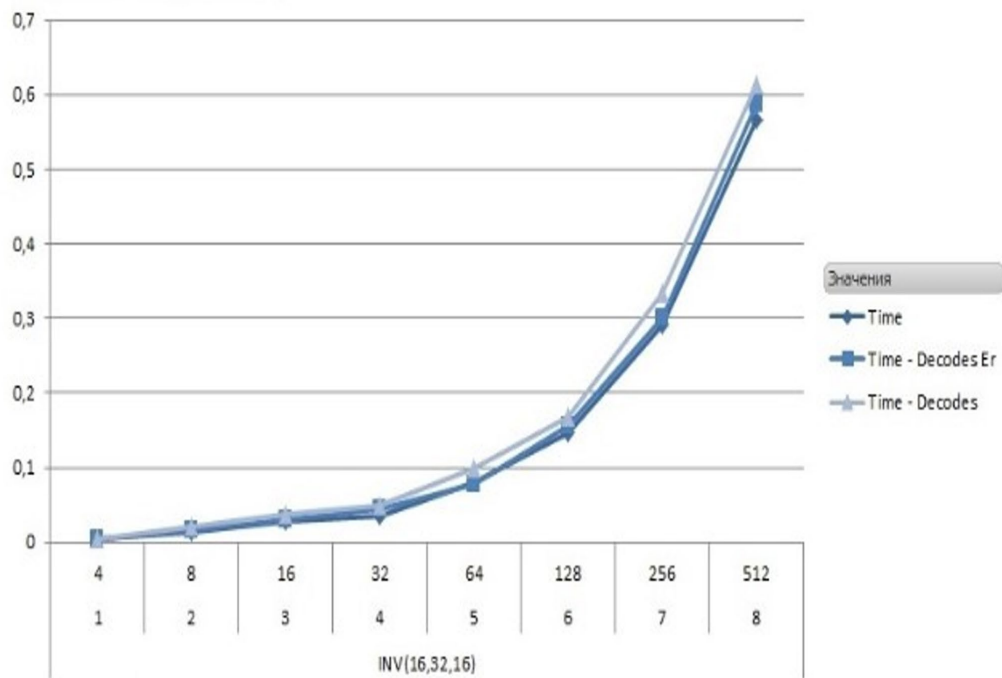
25

Код ХЕММІНГА HEM(16,21,5)



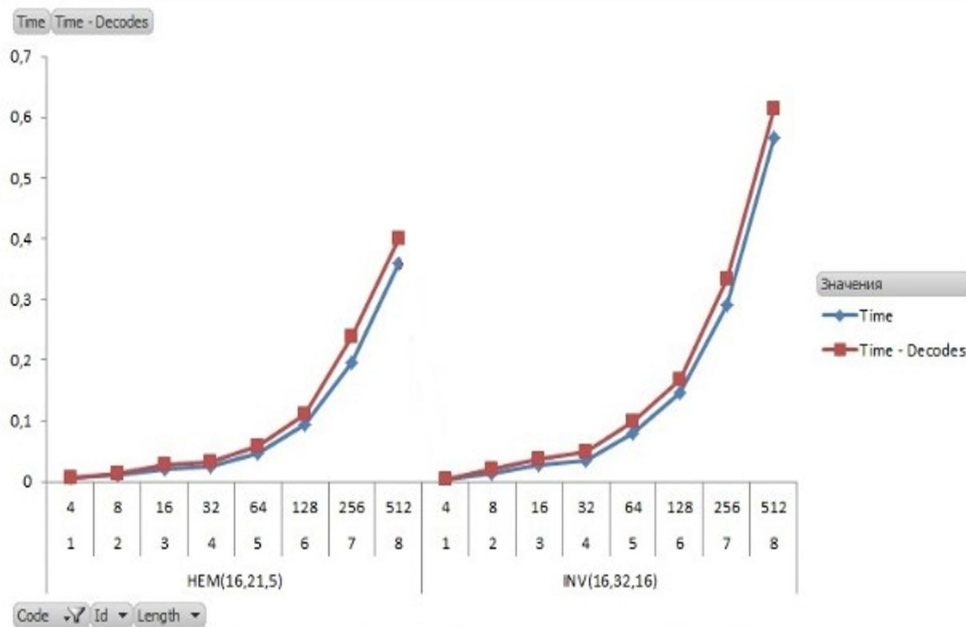
26

ІНВЕРСНИЙ КОД INV(16,32,16)



27

ПОРІВНЯННЯ КОДІВ НЕМ(16,21,5) ТА INV(16,32,16)



28

ВИСНОВКИ

- За допомогою розробленого програмного комплексу було порівняно чотири коди двох типів, а саме код Хеммінга і Інверсний код за такими параметрами час кодування час декодування і час декодування з помилкою. Всі операції здійснювалися над вісьмома інформаційними символічними послідовностями різної довжини - від 2 до 512.
- Код INV (16,32,16) за кількістю інформаційних і контрольних біт в два рази перевищує аналогічні параметри коду INV (8,16,8). Однак виконує необхідні операції майже за ті ж часові проміжки, що і код INV (8,16,8). Цим обумовлюється недоцільність використання даного типу кодів, в порівнянні з кодом Хеммінга.
- Найменше часу на роботу витрачає код Хеммінга НЕМ (16,21,5). Він є найбільш прийнятним з точки зору часу виконання операцій кодування і декодування. Код НЕМ (16,21,5) є версією коду НЕМ (8,12,4), але з чисельно збільшеними параметрами. Як показали дослідження даних кодів, код НЕМ (16,21,5) працює набагато швидше коду НЕМ (8,12,4). Це дозволяє вважати, що при збільшенні характеристик коду цього типу, зменшиться час кодування. Інакше кажучи, зі збільшенням числа інформаційних і контрольних біт, зросте і продуктивність даного коду.

29

ВИСНОВКИ

- Реалізований програмний комплекс для тестування і аналізу завадостійких кодів. У комплекс увійшли чотири компоненти, написані спеціально для нього, і також були задіяні два сторонні компоненти у вигляді програмних пакетів.
- Оскільки результати отримані в результаті роботи конкретного комп'ютера з конкретними технічними характеристиками, то справедливо стверджувати, що час, витрачений аналогічними кодами на операції кодування і декодування, може змінитися в залежності від технічних характеристик використовуваної платформи.
- Одним з основних переваг розробленого програмного комплексу є його універсальність з точки зору використання інших кодів. Його архітектура дозволяє тестувати, аналізувати і порівнювати різні реалізації різних завадостійких кодів в однакових умовах. Це досягається за рахунок специфіки виклику функцій: механізм виклику функції майже ніяк не пов'язаний з реалізацією самої функції і дозволяє подавати одну й ту ж послідовність на вхід безлічі різних функцій.
- Всі поставлені в дипломній роботі задачі, виконано.

30

ПУБЛІКАЦІЇ

- 1) Мартинович Л. Я., Аль-Омарі А. О., Слущкий Д. В., Смірнов К. О. Методи завадостійкого кодування у цифрових системах «Інформатика, інформаційні системи та технології: тези доповідей вісімнадцятої всеукраїнської конференції студентів і молодих науковців». Одеса, 23 квітня 2021 р. - Одеса, 2021. – с.135-137
- 2) Гунченко Ю.О., Мартинович Л.Я., Аль-Омарі А.О. Аналіз завадостійких кодів за допомогою програмного комплексу Тези доповідей Всеукраїнської науково -практичної конференції молодих вчених, ад'юнктів, слухачів, курсантів і студентів "Молодіжна військова наука у Київському національному університеті імені Тараса Шевченка" / за заг. редакцією І.В. Толока. – К. : ВІКНУ, 2021. – с. 99-100

31

ДЯКУЮ ЗА УВАГУ

32