Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерства освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерства освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ДИЧКА АНДРІЙ ІВАНОВИЧ

УДК 004.627

ДИСЕРТАЦІЯ

АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОЦЕСІВ АВТОМАТИЧНОЇ ІДЕНТИФІКАЦІЇ НА ОСНОВІ БАГАТОКОЛІРНИХ ЗАВАДОСТІЙКИХ ШТРИХОВИХ КОДІВ У МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

121 Інженерія програмного забезпечення

12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ А. І. Дичка
(підпис)

Науковий керівник: Сулема Євгенія Станіславівна, доктор технічних наук, доцент

Київ – 2023

АНОТАЦІЯ

Дичка А. І. Алгоритмічне та програмне забезпечення процесів автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 Інформаційні технології за спеціальністю 121 Інженерія програмного забезпечення. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2023.

Автоматична ідентифікація є технологією введення інформації в систему обробки шляхом сканування спеціальної позначки, нанесеної на поверхню об'єкта обліку.

Введення даних на основі штрихового кодування інформації є одним з найпоширеніших видів автоматичної ідентифікації, перевагами якої є висока швидкість та точність введення даних в обчислювальну систему.

Технологію штрихового кодування інформації використовують в різноманітних галузях, її застосування є актуальним і для медичної галузі.

З розвитком інформаційних технологій постає задача інформатизації медичної галузі. Цифрова медицина є важливим компонентом суспільного життя. Однією з важливих складових електронної охорони здоров'я є медичні інформаційні системи. Медична інформаційна система є програмно-технічним комплексом, використання якого дозволяє підвищити ефективність функціонування медичної установи та поліпшити організацію обслуговування пацієнтів.

Аналіз існуючих медичних інформаційних систем показує, що недостатнім є ступінь автоматизації процесів при наданні медичних послуг; необхідно посилити захист та цілісність персональних і медичних даних пацієнтів, автоматизувати для пацієнта доступ до власних медичних даних та

забезпечити електронну візуально-захищену взаємодію лікаря та пацієнта з використанням мобільних пристроїв.

Вирішення цих завдань можливе за рахунок використання в медичній інформаційній системі багатоколірного штрихового кодування інформації, яке дозволяє у декілька разів підвищити інформаційну щільність штрихкодових позначок порівняно з чорно-білими штриховими кодами.

Окреслені завдання та тенденції розвитку цифрової медицини визначають актуальну науково-технічну задачу вдосконалення та розвитку теоретичних основ створення алгоритмічного та програмного забезпечення процесів автоматичної ідентифікації у медичних інформаційних системах на основі багатоколірного завадостійкого штрихового кодування персональних та медичних даних пацієнтів з використанням мобільних пристроїв, яка вирішується в цій дисертаційній роботі.

Метою дисертаційної роботи є удосконалення технології розроблення спеціалізованого класу прикладного програмного забезпечення у складі медичних інформаційних систем на основі автоматичної ідентифікації об’єктів медичного документування з використанням багатоколірного завадостійкого штрихового кодування інформації.

У першому розділі дисертаційної роботи проаналізовано алгоритмічно-програмні рішення для автоматичної ідентифікації в медичних інформаційних системах. Вивчено сучасний стан програмного забезпечення в медичній галузі, зокрема спеціалізовані програмні продукти для різних секторів медицини, а також програмні рішення, пов’язані із застосуванням мобільних пристроїв для охорони здоров’я. Досліджено особливості багатоколірного штрихового кодування інформації. Це дозволило сформувати вимоги до розроблення архітектури програмної системи, яка б надавала пацієнтам додаткові можливості щодо автоматизованого доступу до своїх медичних даних на основі багатоколірного штрихового кодування інформації.

У другому розділі розроблено алгоритмічне та програмне забезпечення процесів створення багатоколірних завадостійких штрихкодових знаків як

мінімальних структурних одиниць штрихкодового зображення. Запропоновано створювати штрихкодові знаки так, щоб вектор (цифровий еквівалент) кожного штрихкодового знака був кодовим словом многозначного коректувального коду Хемінга або БЧХ.

Розроблено метод синтезу символіки заданої потужності та колірності завадостійкого штрихового коду для реалізації в медичній інформаційній системі на основі багатоколірного штрихового кодування інформації.

У третьому розділі розроблено алгоритмічне та програмне забезпечення завадостійкості багатоколірних штрихових кодів у медичних інформаційних системах. Запропоновано метод підвищення завадостійкості багатоколірних штрихових кодів, який грунтується на застосуванні дворівневого контролю ушкоджень з використанням двох многозначних коректувальних кодів: коду, який виправляє одно- або двократні ушкодження у межах штрихкодового знака, та коду Ріда-Соломона – на рівні усієї штрихкодової позначки.

Четвертий розділ присвячено удосконаленню технології проєктування програмного забезпечення процесів автоматичної ідентифікації у медичних інформаційних системах на основі багатоколірного завадостійкого штрихового кодування інформації. На основі аналізу та пріоритизації функціональних вимог розроблено архітектуру програмного забезпечення у складі медичної інформаційної системи, характерною особливістю якої є застосування багатоколірного штрихового кодування для автоматичної ідентифікації об’єктів медичного документування. Запропоновано організацію моніторингу функціонування програмного забезпечення у складі медичної інформаційної системи на основі метрик.

У дисертаційній роботі отримано низку **нових наукових результатів**, зокрема **уперше** запропоновано архітектуру програмної системи як ядро медичної інформаційної системи, характерною рисою якої є забезпечення багатоколірного завадостійкого штрихового кодування персональних даних пацієнтів та розподіленого зберігання мультимодальних медичних даних, що

дозволяє: забезпечити швидке і безпомилкове введення даних пацієнта, гарантувати цілісність даних, а також спростити процеси створення програмного забезпечення для галузі охорони здоров'я нового покоління – медичних інформаційних систем з автоматичною ідентифікацією об'єктів медичного документування.

Уперше розроблено метод синтезу символіки (множини штрихкодових знаків) заданої потужності та колірності завадостійкого штрихового коду для реалізації у медичній інформаційній системі на основі багатоколірного штрихового кодування інформації, який грунтується на тому, що цифрові еквіваленти (вектори) багатоколірних штрихкодових знаків символіки є кодовими словами многозначного коректувального коду, здатного виправляти одно- або двократні помилки (ушкодження) в межах кожного штрихкодового знака, що при зчитуванні (скануванні) з носія штрихкодових знаків – як структурних одиниць багатоколірного штрихкодового зображення, забезпечує достовірне відтворення даних або виявлення значної частини багатократних ушкоджень елементів штрихкодового знака, і таким чином утворює нижній рівень забезпечення завадостійкості багатоколірних штрихкодових зображень (рівень штрихкодових знаків).

Уперше розроблено метод підвищення завадостійкості багатоколірних штрихових кодів при їх використанні в медичній інформаційній системі, який грунтується на застосуванні дворівневого контролю ушкоджень (спотворень), що виникають при скануванні багатоколірного штрихкодового зображення, з використанням двох многозначних коректувальних кодів: коду, який виправляє одно- або двократні ушкодження – на рівні штрихкодових знаків (нижній рівень), та коду Ріда-Соломона – на рівні усієї штрихкодової позначки (верхній рівень), та полягає в тому, що особливістю застосування многозначного коректувального коду на нижньому рівні контролю ушкоджень є те, що цифровий еквівалент (вектор) багатоколірного штрихкодового знака має бути кодовим словом цього коректувального коду, а особливістю застосування многозначного коректувального коду на верхньому рівні

контролю ушкоджень є те, що він має виправляти спотворення двох видів – помилки та стирання, де факт виявлення багатократного ушкодження елементів штрихкодового знака на нижньому рівні контролю кваліфікується на верхньому рівні контролю як стирання, що, завдяки удвічі меншому витрачанні ресурсу в коді Ріда-Соломона на виправлення стирання порівняно з виправленням помилки, дозволяє істотно поліпшити завадостійкість штрихкодових позначок: при застосуванні на нижньому рівні контролю многозначного коду БЧХ – на (35 - 45)%, а многозначного коду Хемінга – на (12 - 25)%.

Удосконалено теоретичні засади розроблення вимог до проєктованого програмного забезпечення, які полягають у тому, що на відміну від існуючого підходу до визначення пріоритетності функціональних вимог якісними (нечисловими) показниками – “ключова”, “необхідна”, “бажана” тощо, запропоновано - на основі п’яти числових критеріїв (“вигода”, “втрата”, “вартість”, “ризик”, “близькість вигоди”), яким присвоюють числові значення з діапазону 1…10 на основі їх експертного оцінювання, кількісно визначати загальний пріоритет вимоги шляхом формульного обчислення, що дозволяє упорядковувати вимоги за спаданням пріоритету та на цій підставі ефективно планувати та розподіляти роботи між командами виконавців з урахуванням їх кваліфікації та кількісного складу, а також виконувати поквартальне планування робіт з розроблення програмного забезпечення.

Основні наукові результати дисертаційної роботи **опубліковано** у 7 наукових працях, зокрема у 5 наукових статтях, з яких 1 статтю опубліковано у виданні, включеному до переліку наукових фахових видань України з присвоєнням категорії «А», і 4 статті опубліковано у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б», та у 2 матеріалах наукових конференцій.

Ключові слова: прикладне програмне забезпечення, архітектура програмного забезпечення, медичні інформаційні системи, автоматична ідентифікація, багатоколірне штрихове кодування, цифрова медицина.

SUMMARY

Dychka A. Algorithms and software for automatic identification processes based on multicolor interference-resistant barcodes in medical information systems. – Qualifying scientific work, manuscript.

PhD thesis in the field of knowledge 12 Information technologies in speciality 121 Software Engineering. – National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2023.

Automatic identification is a technology for entering information into the processing system by scanning a special mark applied to the surface of the accounting object.

Data entry based on barcoding of information is one of the most common types of automatic identification, the advantages of which are high speed and accuracy of data entry into the computer system.

The technology of barcoding information is used in various industries, its application is also relevant for the medical industry.

With the development of information technology, the task of informatization for medical industry arises. Digital medicine is an important component of public life. One of the important components of e-health is medical information systems. The medical information system is a software and hardware complex, the use of which allows to increase the efficiency of the functioning of a medical institution and improve the organization of patient care.

The analysis of existing medical information systems shows that the degree of automation of processes in the provision of medical services is insufficient; it is necessary to strengthen the protection and integrity of personal and medical data of patients, automate patients' access to their own medical data, and to ensure electronic visually secure interaction between doctor and patient using mobile devices.

The solution of these problems is possible due to the use of multicolor barcoding of information in the medical information system, which allows several

times to increase the information density of barcode symbols compared to black and white barcodes.

The outlined tasks and trends in the development of digital medicine determine the actual scientific and technical task of improving and developing the theoretical foundations of creating algorithmic and software tools for automatic identification processes in medical information systems based on multicolor noise-immunity barcoding of personal and medical data of patients using mobile devices, which is solved in this dissertation.

The purpose of the dissertation is to improve the technology of developing specialized applied software as part of medical information systems based on automatic identification of medical documentation objects using multicolor noise-resistant barcoding of information.

In the first section of the thesis, algorithmic and software solutions for automatic identification in medical information systems are analyzed. The current state of software in the medical field is studied, in particular specialized software products for various sectors of medicine, as well as software solutions related to the use of mobile devices for health care. The features of multicolor bar coding of information are investigated. This made it possible to form requirements for the development of a software system architecture that would provide patients with additional opportunities for automated access to their medical data based on multicolor barcoding of information.

The second section develops algorithmic and software tools for the processes of creating multicolor noise-immune barcode patterns as minimal structural units of a barcode symbol. It is proposed to create barcode patterns so that the vector (digital equivalent) of each barcode pattern is the codeword of a multi-valued Hamming correcting code or BCH-code.

A method of synthesis symbology of a given capacity and color of a noise-resistant barcode for implementation in a medical information system based on multicolor barcoding of information has been developed.

The third section develops algorithms and software for noise immunity of multicolor barcodes in medical information systems. A method of increasing the noise immunity of multicolor bar codes is proposed, which is based on the use of two-level damage control using two multi-valued correcting codes: a code that corrects single or double damages within a barcode pattern, and a Reed-Solomon code – at the level of the entire barcode symbol.

The fourth section is devoted to improving software design technology for automatic identification processes in medical information systems based on multicolor interference-resistant barcoding of information. Based on the analysis and prioritization of functional requirements, the architecture of software system as the part of the medical information system is developed, the characteristic feature of which is the use of multicolor barcoding for automatic identification of medical documentation objects. The monitoring organization of software functioning as a part of medical information system based on metrics is proposed.

A number of **new scientific results** were obtained in the dissertation, in particular, **for the first time** the architecture of the software system as the core of medical information system was proposed, a characteristic feature of which is the provision of multicolor interference-resistant barcoding of patients' personal data and distributed storage of multimodal medical data, which allows: to ensure fast and error-free entry of patient data, to guarantee integrity data, as well as simplify the processes of creating software for the new generation healthcare industry – medical information systems with automatic identification of medical documentation objects.

For the first time, a method of synthesis of symbology (a set of barcode patterns) of a given capacity and color of a interference-resistant barcode for implementation in a medical information system based on multicolor barcoding information based on the fact that digital equivalents (vectors) of multicolor barcode patterns are codewords of a multi-valued correcting code capable of correction single or double errors (damage) within each barcode pattern, which, when reading (scanning) from a barcode carrier – as structural units of a multi-color barcode

image, provides reliable data reproduction or detection of a significant part of multiple damage of barcode pattern elements, and thus forms the lower level of noise immunity of multicolor barcode images (level of barcode patterns).

For the first time, a method was developed to increase the noise immunity of multicolor barcodes when they are used in a medical information system, which is based on the use of two-level control of damage (distortion) that occurs when scanning a multi-color barcode image, using two multi-valued correcting codes: a code that corrects single or double damage – at the barcode patterns (lower level), and the Reed-Solomon code – at the level of the entire barcode symbol (upper level), and is that the peculiarity of the application of a multi-valued correcting code at the lower level of damage control is that the digital equivalent (vector) of a multicolor barcode pattern should be the codeword of this correcting code, and the peculiarity of using a multi-valued correcting code at the upper level of damage control is that it should correct distortions of two types – errors and erasures, where the fact of detecting multiple damage of barcode pattern elements at the lower level of control is qualified at the upper level of control as erasure, which, due to half the resource consumption in the Reed code-Solomon's erasure correction compared to error correction can significantly improve the noise immunity of barcode symbols: when used at the lower level of control the multi-valued BCH-code – by (35 - 45)%, and the multi-valued Hamming code – by (12 - 25)%.

The theoretical foundations of developing requirements for the designed software **have been improved**, which consist in the fact that, unlike the existing approach to prioritizing functional requirements by qualitative (non-numeric) indicators – "key", "necessary", "desirable", etc., it is proposed - on the basis of five numerical criteria ("benefit ", "penalty", "cost", "risk", "benefit proximity") that are assigned numeric values from the range 1... 10 on the basis of their expert evaluation, quantify the overall priority of the requirement by formula calculation, which allows you to streamline the requirements in descending order of priority and, on this basis, effectively plan and distribute work among teams of performers, taking into account

their qualifications and quantitative composition, as well as perform quarterly planning of software development.

The main scientific results of the dissertation **were published** in 7 scientific papers, in particular in 5 scientific articles, of which 1 article was published in the publication included in the list of scientific journals of Ukraine in category "A", and 4 articles in scientific journals included in the list of scientific journals of Ukraine in category "Б", as well as in 2 materials of scientific conferences.

Keywords: application software**,** software architecture, medical information systems, automatic identification, multicolor barcoding, digital medicine.

Список публікацій здобувача / List of publications of the applicant:

Стаття у виданні, включеному до переліку наукових фахових видань України з присвоєнням категорії «А» / the article published in a scientific journal included in the list of scientific journals of Ukraine in category «A»:

1. Sulema Ye., Drozdenko L, Dychka A. Synthesis of the Symbologies of Multicolor Interference-Resistant Bar Codes on the Base of Multi-Valued BCH Codes // Radio Electronics, Computer Science, Control. – 2022. – Vol 4. P. 107-118.
   DOI :10.15588/1607-3274-2022-4-9.

Статті у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б» / the articles published in scientific journals included in the list of scientific journals of Ukraine in category «Б»:

2. Сулема Є. С., Онай М. В., Дичка А. І. Алгоритмічне забезпечення завадостійкості багатоколірних штрихкодових знаків на основі поля GF(p) // Наукові вісті КПІ. – 2021. – № 1(132). – С. 50-62.
   DOI :10.20535/kpisn.2021.1.231210
3. Сулема Є. С., Онай М. В., Дичка А. І. Забезпечення завадостійкості багатоколірних штрихкодових знаків на основі поля GF(p^m) // Системні технології. - 2021. Вип.1 (132), С. 31-50.
   DOI :10.34185/1562-9945-1-132-2021-03.
4. Сулема Є. С., Дичка А. І. Підвищення завадостійкості багатоколірних штрихкодових зображень // Системні технології. – 2023. – Вип. 2 (145). – С. 105-124.
   DOI :10.34185/1562-9945-2-145-2023-10.
5. Сулема Є. С., Дичка А. І. Програмна система автоматичної ідентифікації та розподіленого зберігання медичних даних пацієнтів // Системні технології. – 2023. – Вип. 3 (146). – С. 134-148.
   DOI :10.34185/1562-9945-3-146-2023-13.

Публікації у матеріалах наукових конференцій / the publications in the proceedings of the scientific conferences:

6. Сулема Є. С., Дичка А. І. Аналіз коректувальних властивостей многозначних кодів Хемінга // Чотирнадцята наукова конференція магістрантів та аспірантів “Прикладна математика та комп’ютинг (ПМК’2021)”, Київ, 17 - 19 листопада 2021 р. Збірник тез доповідей Нац. техн. ун-т України “Київ. політехн. ін-т ім. Ігоря Сікорського”, Київ: Просвіта. - 2021. - С. 89-97.
   ISBN 978-617-7010-18-9.
7. Дичка А. І. Застосування многозначних кодів БЧХ для підвищення завадостійкості багатоколірних штрихових кодів // “Світ наукових досліджень. Випуск 13”: матеріали Міжнародної мультидисциплінарної наукової інтернет-конференції, (м. Тернопіль, Україна – м. Переворськ, Польща, 25-26 жовтня 2022 р.). – ГО “Наукова спільнота”; WSSG w Przeworsku. – 2022. - С. 69 - 78.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ASCII – American Standard Code for Information Interchange (американський стандартний код для обміну інформацією)

COBRA – COlour BaRcode streaming for smArtphone (колірний штриховий код для смартфона)

EAN – European Article Numbering (європейська нумерація товарів)

GF – Galois Field (поле Галуа)

HCCB – High Capacity Colour Barcode (колірний штриховий код високої ємності)

HCC2D – High Capacity Coloured 2-Dimensional Code (колірний двовимірний код високої ємності)

JABcode – Just Another Barcode (колірний штриховий код)

LDPC – Low-Density Parity-check Code (код з малою щільністю перевірок на парність)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

QR-code – Quick Response code (код швидкого відгуку)

RFID – Radio Frequency Identification (радіочастотна ідентифікація)

БЧХ – коректувальний код Боуза-Чоудхурі-Хоквінгема

МІС – медична інформаційна система

НСК – найменше спільне кратне

ПЗ – програмне забезпечення

Мод – модуль, відносна одиниця вимірювання довжини (висоти) у штрихових кодах

ФВ – функціональні вимоги до програмного забезпечення

ШК – штриховий код

ШК-знак – штрихкодовий знак

ШК-позначка – штрихкодова позначка

ВСТУП

Актуальність теми. Автоматична ідентифікація є технологією введення даних в систему обробки інформації шляхом сканування спеціальної позначки (мітки), нанесеної на поверхню об’єкта обліку.

Одним з найпоширеніших видів автоматичної ідентифікації є технологія введення даних на основі штрихового кодування інформації, перевагами якої є висока швидкість введення даних в обчислювальну систему, низька вартість виготовлення штрихкодових позначок та зручність застосування; штрихкодові позначки зчитують (сканують) з об’єктів обліку оптичним способом, в тому числі й на відстані.

Технологію штрихового кодування інформації використовують в різноманітних галузях: торгівлі, логістиці, виробництві, військовій галузі, сфері послуг, поштовій справі тощо. Застосування штрихового кодування даних є актуальним також у медичній галузі.

З розвитком інформаційних технологій постає задача інформатизації медичної галузі. Цифрова медицина, електронна охорона здоров’я є важливими компонентами суспільного життя. Основними складовими електронної охорони здоров’я є телемедичні мережі та медичні інформаційні системи. Медична інформаційна система є програмно-технічним комплексом, метою впровадження якої є підвищення ефективності функціонування медичної установи і поліпшення обслуговування пацієнтів.

Аналіз існуючих медичних інформаційних систем показує, що вони є переважно спеціалізованими, корпоративними, недостатньо функціонально розвиненими та не вирішують узагальнених завдань в системі електронної охорони здоров’я. Недостатнім є ступінь автоматизації процесів при наданні медичних послуг, необхідно істотно знизити вплив людського фактора як при прийнятті медичних рішень, так і у виконавчій діяльності медперсоналу, посилити захист персональних та медичних даних пацієнтів, автоматизувати для пацієнта доступ до власних медичних даних, а також забезпечити

візуально захищену електронну взаємодію між лікарем або медичним закладом та пацієнтом.

Вирішення цих та інших завдань можливе за рахунок використання у складі медичної інформаційної системи багатоколірного штрихового кодування службової інформації, а також персональних та медичних даних пацієнтів.

Останніми роками має місце тенденція до розширення сфери застосування багатоколірних штрихових кодів, оскільки багатоколірність дозволяє у декілька разів підвищити інформаційну щільність штрихкодових позначок порівняно з чорно-білими штриховими кодами. Однак, при застосуванні багатоколірних штрихових кодів ускладнюються процеси декодування штрихкодових позначок. Тому, для надійного відтворення даних зі штрихкодової позначки необхідно забезпечувати завадостійкість штрихкодових зображень.

Крім того, набуває розвитку застосування мобільних пристроїв у системі охорони здоров'я, головне призначення яких посилити взаємодію пацієнта та лікаря.

Таким чином, актуальною є науково-технічна задача вдосконалення та розвитку теоретичних основ створення алгоритмічного та програмного забезпечення процесів автоматичної ідентифікації у медичних інформаційних системах на основі багатоколірного завадостійкого штрихового кодування персональних та медичних даних пацієнтів з використанням мобільних пристроїв.

Зв'язок роботи з науковими програмами, планами, темами. Дослідження за темою дисертаційної роботи провадились у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» в рамках виконання держбюджетної науково-дослідної роботи «Математичні та програмні методи оброблення мультимодальних даних моніторингу медико-біологічних об'єктів для діагностики стану здоров'я пацієнтів» (номер державної реєстрації 0120U102134).

Мета і задачі дослідження. Метою дисертаційної роботи є удосконалення технології розроблення спеціалізованого класу прикладного програмного забезпечення - медичних інформаційних систем, на основі автоматичної ідентифікації об'єктів медичного документування з використанням багатоколірного завадостійкого штрихового кодування інформації.

Відповідно до поставленої мети основними задачами дослідження є:

- аналіз програмного забезпечення медичних інформаційних систем;
- формування вимог до програмної системи автоматичної ідентифікації на основі багатоколірного штрихового кодування інформації;
- розроблення архітектури програмної системи автоматичної ідентифікації на основі багатоколірного штрихового кодування у складі медичної інформаційної системи;
- розроблення алгоритмічного і програмного забезпечення процесів створення символік багатоколірних завадостійких штрихових кодів;
- розроблення методу підвищення завадостійкості багатоколірних штрихкодових зображень з можливістю їх зчитування мобільними пристроями;
- аналіз характеристик розробленого програмного забезпечення для автоматичної ідентифікації на основі багатоколірних штрихових кодів.

Об'єкт дослідження – процеси розроблення програмних систем автоматичної ідентифікації на основі штрихового кодування інформації.

Предмет дослідження – методи розроблення алгоритмічного та програмного забезпечення систем автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів.

Методи дослідження: теорія програмування, теорія програмних систем, теорія інформаційних систем, теорія алгоритмів, теорія завадостійкого кодування, теорія скінченних полів.

Наукова новизна одержаних результатів полягає у наступному:

1. **Уперше** запропоновано архітектуру програмної системи як ядро медичної інформаційної системи, характерною рисою якої є забезпечення багатоколірного завадостійкого штрихового кодування персональних даних пацієнтів та розподіленого зберігання мультимодальних медичних даних, що дозволяє: забезпечити швидке і безпомилкове введення даних пацієнта, гарантувати цілісність даних, а також спростити процеси створення програмного забезпечення для галузі охорони здоров'я нового покоління – медичних інформаційних систем з автоматичною ідентифікацією об'єктів медичного документування.
2. **Уперше** розроблено метод синтезу символіки (множини штрихкодових знаків) заданої потужності та колірності завадостійкого штрихового коду для реалізації у медичній інформаційній системі на основі багатоколірного штрихового кодування інформації, який грунтується на тому, що цифрові еквіваленти (вектори) багатоколірних штрихкодових знаків символіки є кодовими словами многозначного коректувального коду, здатного виправляти одно- або двократні помилки (ушкодження) в межах кожного штрихкодового знака, що при зчитуванні (скануванні) з носія штрихкодових знаків – як структурних одиниць багатоколірного штрихкодового зображення, забезпечує достовірне відтворення даних або виявлення значної частини багатократних ушкоджень елементів штрихкодового знака, і таким чином утворює нижній рівень забезпечення завадостійкості багатоколірних штрихкодових зображень (рівень штрихкодових знаків).
3. **Уперше** розроблено метод підвищення завадостійкості багатоколірних штрихових кодів при їх використанні в медичній інформаційній системі, який грунтується на застосуванні дворівневого контролю ушкоджень (спотворень), що виникають при скануванні багатоколірного штрихкодового зображення, з використанням двох многозначних коректувальних кодів: коду, який виправляє одно- або двократні

ушкодження – на рівні штрихкодових знаків (нижній рівень), та коду Ріда-Соломона – на рівні усієї штрихкодової позначки (верхній рівень), та полягає в тому, що особливістю застосування многозначного коректувального коду на нижньому рівні контролю ушкоджень є те, що цифровий еквівалент (вектор) багатоколірного штрихкодового знака має бути кодовим словом цього коректувального коду, а особливістю застосування многозначного коректувального коду на верхньому рівні контролю ушкоджень є те, що він має виправляти спотворення двох видів – помилки та стирання, де факт виявлення багатократного ушкодження елементів штрихкодового знака на нижньому рівні контролю кваліфікується на верхньому рівні контролю як стирання, що, завдяки удвічі меншому витрачанні ресурсу в коді Ріда-Соломона на виправлення стирання порівняно з виправленням помилки, дозволяє істотно поліпшити завадостійкість штрихкодових позначок: при застосуванні на нижньому рівні контролю многозначного коду БЧХ – на (35 - 45)%, а многозначного коду Хемінга – на (12 - 25)%.

4. Удосконалено теоретичні засади розроблення вимог до проєктованого програмного забезпечення, які полягають у тому, що на відміну від існуючого підходу до визначення пріоритетності функціональних вимог якісними (нечисловими) показниками – “ключова”, “необхідна”, “бажана” тощо, запропоновано - на основі п’яти числових критеріїв (“вигода”, “втрата”, “вартість”, “ризик”, “близкість вигоди”), яким присвоюють числові значення з діапазону 1…10 на основі їх експертного оцінювання, кількісно визначати загальний пріоритет вимоги шляхом формульного обчислення, що дозволяє упорядковувати вимоги за спаданням пріоритету та на цій підставі ефективно планувати та розподіляти роботи між командами виконавців з урахуванням їх кваліфікації та кількісного складу, а також виконувати поквартальне планування робіт з розроблення програмного забезпечення.

Практичне значення одержаних результатів полягає у спрощенні процесу розроблення програмного забезпечення для медичної галузі із застосуванням багатоколірного завадостійкого штрихового кодування даних та використання мобільних пристроїв для електронної взаємодії пацієнта і лікаря, що підвищує ефективність процесу створення програмного продукту – ядра медичної інформаційної системи.

Розроблене алгоритмічне та програмне забезпечення процесів завадостійкого кодування-декодування інформації застосовано при виконанні держбюджетної науково-дослідної роботи «Математичні та програмні методи оброблення мультимодальних даних моніторингу медико-біологічних об'єктів для діагностики стану здоров'я пацієнтів» (номер державної реєстрації 0120U102134) для подання медичних даних пацієнтів у вигляді багатоколірного завадостійкого штрихкодового зображення.

Особистий внесок здобувача. Усі основні результати дисертаційного дослідження, які представлені до захисту, одержані автором особисто. У публікаціях, написаних у співавторстві, здобувачеві належать наступні результати. У роботі [1] здобувачем запропоновано метод побудови множини завадостійких штрихкодових знаків на основі многозначного коректувального коду БЧХ, які забезпечують достовірне відтворення даних при їх зчитуванні з носія. У роботі [2] здобувачем запропоновано спосіб побудови багатоколірних завадостійких штрихкодових знаків на основі многозначних кодів Хемінга. У роботі [3] здобувачем запропоновано процедуру подання даних у вигляді послідовності завадостійких багатоколірних штрихкодових знаків з можливістю виправлення однократних помилок у штрихкодових знаках. У роботі [4] здобувачем запропоновано метод підвищення завадостійкості багатоколірних штрихових кодів, який грунтується на застосуванні дворівневого контролю ушкоджень з використанням двох многозначних коректувальних кодів. У роботі [5] здобувачем запропоновано архітектуру програмної системи на основі принципу розподіленого зберігання медичних даних пацієнта та анонімності передачі даних за рахунок зчитування

штрихкодової інформації з одного смартфона камерою іншого смартфона. У роботі [6] здобувачем запропоновано алгоритмічне забезпечення процедур кодування-декодування даних многозначним кодом Хемінга.

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідалися та обговорювалися на наукових конференціях:

1. Чотирнадцята наукова конференція магістрантів і аспірантів “Прикладна математика та комп’ютинг (ПМК’2021)”, Київ, 17 - 19 листопада 2021 р. Київ, Україна.
2. Міжнародна мультидисциплінарна наукова інтернет-конференція на тему “Світ наукових досліджень. Випуск 13” 25 - 26 жовтня 2022 р. Тернопіль, Україна.

Публікації. Основні наукові результати дисертаційної роботи опубліковано у 7 наукових працях, зокрема у 5 наукових статтях, з яких 1 статтю опубліковано у виданні, включеному до переліку наукових фахових видань України з присвоєнням категорії «А», і 4 статті опубліковано у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б», та у 2 матеріалах наукових конференцій.

Структура роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що включає 105 найменувань, та 13 додатків. Загальний обсяг роботи становить 231 сторінку, у тому числі 156 сторінок основного тексту, 45 рисунків, 16 таблиць.

РОЗДІЛ 1. АНАЛІЗ АЛГОРИТМІЧНО-ПРОГРАМНИХ РІШЕНЬ ДЛЯ АВТОМАТИЧНОЇ ІДЕНТИФІКАЦІЇ В МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

1. 1. Сучасний стан програмного забезпечення в медичній галузі

Охорона здоров’я населення є одним з пріоритетів діяльності держави. На порядку денному впровадження прогресивних інформаційно-комунікаційних рішень в галузь електронної охорони здоров’я, яка охоплює процеси накопичення, оброблення, аналізу та передачі цифрових медичних даних [1, 2].

Впровадження сучасних інформаційно-комунікаційних рішень в галузі охорони здоров’я забезпечить: належну якість надання медичних послуг, розширення можливостей споживачів (впровадження автоматизованого доступу пацієнтів до власних медичних даних, можливість вибору фахівців, медичних закладів тощо), захищеність персональних та медичних даних пацієнтів, зниження небажаних ризиків через людський фактор, надійність діагностування та прийняття клінічних рішень, електронний обмін інформацією між фахівцями та медичними закладами, електронну комунікацію лікаря та пацієнта, накопичення електронних архівних даних для навчання медичного персоналу, аналізу, визначення тенденцій та систематизації цифрової медичної інформації [3, 4].

У діяльності медичних закладів важливе місце посідають медичні інформаційні системи.

Медична інформаційна система (МІС) – це системно організована для вирішення завдань управління сукупність методів і засобів реалізації операцій збирання, реєстрації, передачі, накопичення, пошуку, обробки і захисту медичної інформації на базі застосування спеціалізованого програмного забезпечення, включення до її складу необхідного медичного обладнання для отримання медичної інформації, використання засобів обчислювальної

техніки, зв’язку та способів, за допомогою яких інформація надається користувачам [2].

МІС має забезпечувати також взаємодію медичної установи з eHealth та Національною службою здоров’я України (НСЗУ). Вона має бути протестована на відповідність вимогам Міністерства охорони здоров’я України (МОЗ), зокрема з технічного захисту інформації. До центрального компонента eHealth можуть бути приєднані тільки ті МІС, які підключені до центральної бази даних урядової інформаційної системи [5, 6].

МІС можуть мати як універсальний, так і спеціалізований характер. Прикладом спеціалізованої МІС є медична інформаційна система служби крові [7].

Від інформаційних систем для інших галузей МІС відрізняється тим, що одночасно зберігає і обробляє персональні та медичні дані пацієнтів, а також демографічну інформацію.

Важливим є спосіб розгортання МІС як технічної системи – на сервері медичного закладу або в «хмарі». Серверний спосіб розгортання є найбільш оптимальним для великих медичних закладів з розгалуженою інфраструктурою; для невеликих клінік та лікарів з приватною практикою доцільно застосовувати хмарний спосіб розгортання, який не потребує додаткових експлуатаційних витрат [1, 12 - 14].

За допомогою МІС медична установа може: автоматизувати роботу реєстратури, упорядкувавши та спростивши процедуру запису пацієнтів на прийом, систематизувати інформацію про пацієнтів медичного закладу, медичні послуги і співробітників, управляти матеріальним фондом медичної установи, чергою на місця в стаціонарі, відстежувати рух медикаментів та медпрепаратів на складі і між відділеннями, упорядкувати роботу лабораторій і діагностичних кабінетів, організовувати оперативне передавання результатів досліджень фахівців в автоматичному режимі, організовувати збір статистичних даних, готувати звіти та виконувати аналітичні дослідження, а також здійснювати управлінські рішення [3 - 6].

Структурно МІС складається з підсистем (модулів). До складу переважної більшості сучасних МІС можуть входити такі найбільш загальні компоненти:

- підсистема реєстрації пацієнта у базі даних медичного закладу (модуль «Реєстратура») – для створення персональної електронної облікової картки пацієнта;
- підсистема отримання медичної діагностичної інформації (модуль «Діагностика») – для фіксування результатів процедури тестування пацієнтів (медична діагностика, ультразвукова діагностика, комп'ютерна флюорографія, томографія тощо);
- підсистема отримання результатів лабораторних досліджень (модуль «Лабораторія»);
- підсистема автоматизованого робочого місця (АРМ) лікаря (модуль «Лікар») – для роботи з медичними записами, направленнями, рецептами за програмою «Доступні ліки» тощо;
- підсистема автоматизованого робочого місця молодшого медичного персоналу (модуль «Медсестра»);
- підсистема обліку препаратів (модуль «Медикаменти»);
- підсистема обліку медичного страхування пацієнтів (модуль «Медичне страхування»);
- підсистема статистичного аналізу, підготовки звітів, управлінських рішень, взаємодії з eHealth та НСЗУ (модуль «Адміністрування»).

Залежно від спеціалізації медичної установи її МІС може мати у своєму складі й інші модулі.

Першим проєктом МІС став проєкт MEDINET, розроблений компанією «General Electric» (США).

В окремих секторах медичної галузі застосовуються спеціалізовані медичні інформаційні технології та експертні системи.

Спеціалізовані програмні продукти:ASDSee, Aquilion ONE, Infinix CS-I, Horizon SE, XIDF-QCA801, використовуються в комплексах променевої та ультразвукової діагностики – ультразвукова сонографія, комп'ютерна томографія, магнітно-резонансна томографія, ангіографія, ехасонографія, ендоскопія тощо. Для опрацювання отриманих діагностичних даних застосовують експертні системи, наприклад, Гастрограф 1T, Stimul та ін. [7].

Для встановлення діагнозу та проведення діагностичних тестів використовують такі експертні системи: Експертна система постановки діагнозів Internist, BPLab, SpeseLabs Medical, Meditech, Omron, CardioVita та ін. [2].

У рамках міжнародного співробітництва створені експертні системи і бази даних: NUCLEUS – мультимедійне досьє пацієнта, EMDIS – Європейська медична інформаційна система для донорів кісткового мозку, EPIC – Європейська модель лікування, FEST – база для європейських служб телемедицини, HDI 5000 – експертна система, що забезпечує обробку зображень, телекомунікаційні інфраструктури (ISAAC – інтегрована телекомунікаційна система, SHINE – стратегічна інформаційна мережа охорони здоров'я Європи), програми обслуговування окремих груп населення.

Нині передбачається створення територіальних та глобальних медичних інформаційних систем. Така система організована для онкологічних хворих, для управління охороною здоров'я міста. Для моніторингу туберкульозу використовується багаторівнева комп'ютерна система [10]. Розробляються програмні комплекси баз даних і експертних систем. Створені інформаційні системи з медико-санітарного обслуговування населення, що починають широко застосовуватись.

Сьогодні в медичній галузі України використовують різні інформаційні системи, серед яких «ЕМСІМЕД», «Медучет», «Медіалог», «TherDep», «Astraia», «ЛисМедап», «Каштан», «Доктор Елекс», «Добробут» та інші [1, 2, 8].

Зазначені МІС є комерційними ПЗ. Паралельно з ринком комерційного ПЗ активно розвивається напрям застосування в галузі охорони здоров'я вільно-розповсюджуваного ПЗ з відкритим кодом [8]. Наприклад, широко застосовуються такі МІС з відкритим кодом як WorldVistA, OpenEMR та OpenMRS [8, 9].

Набуває розвитку застосування мобільних пристроїв для охорони здоров'я (m-health), головне призначення яких посилити взаємодію пацієнта та лікаря, що в кінцевому підсумку сприятиме підтримці здоров'я людини в довгостроковій перспективі. Так, за проєктом ЄС, спрямованого на побудову дистанційної системи моніторингу здоров'я літніх людей з множинними хронічними захворюваннями, розроблено мобільний застосунок eCAALYX [10]. Маючи за мету залучення пацієнтів, опікунів і лікарів до оперативної взаємодії, розробники визначають основну функціональність мобільної платформи Ecaalyx – бути посередником між мобільними пристроями, які використовують літні люди, та веб-сайтом медичних працівників, забезпечити лікаря і пацієнта інформацією про отримані дані вимірювання від давачів і про географічне розташування користувача (пацієнта) засобами GPS смартфона [11]. Це забезпечить реалізацію таких основних функцій: облік пацієнтів клінік та обсягу наданої медичної допомоги; дистанційне спостереження їхнього стану; одержувати, зберігати і передавати результати діагностичних обстежень; контроль правильності призначеного лікування; проведення віддаленого навчання; надання консультацій малодосвідченим співробітникам [3, 4].

Аналіз існуючих рішень в системі електронної охорони здоров'я показує, що технологія розроблення програмного забезпечення для медичних інформаційних систем потребує удосконалення. Зокрема, слід забезпечити для пацієнта можливість володіння своїми медичними даними, а також надавати доступ до них лише за власним бажанням. Крім того, доцільно передбачити посилення електронної взаємодії лікаря та пацієнта з використанням мобільних пристроїв. Досягти цього можливо за допомогою запровадження

штрихового кодування інформації – службових даних, а також персональних та медичних даних пацієнта, на основі багатоколірних завадостійких штрихових кодів. Це дозволить підвищити ступінь автоматизації виробничих процесів у медичному закладі та посилить захист даних пацієнта.

За такого підходу головним носієм інформації в медичній інформаційній системі мають бути багатоколірні штрихові коди.

1. 2. Штрихові коди як один з видів автоматичної ідентифікації

Автоматична ідентифікація є способом безклавіатурного введення даних в систему обробки інформації в реальному часі шляхом їх автоматичного зчитування з об’єкта (носія).

Автоматична ідентифікація ґрунтується на застосуванні таких технологій: магнітної, радіочастотної, оптичного розпізнавання символів, штрихкодової, розпізнавання голосу.

Прикладом магнітної технології автоматичної ідентифікації (MSI – Magnetic Stripe Identification) є застосування магнітних карток у перепускних системах. Записана на магнітній стрічці інформація вводиться в систему обробки інформації з високою швидкістю. Магнітна технологія забезпечує високу щільність запису даних на магнітному носії, проте потребує контактного зчитування, і дані не можуть зчитуватись на відстані.

Радіочастотна ідентифікація об’єктів (RFID – Radio Frequency Identification), завдяки тому, що радіосигнали можуть проникати крізь непровідні матеріали, має широкі можливості. Відомі приклади застосування радіочастотної ідентифікації в різних галузях: у гірництві – для визначення місцезнаходження працівників та транспортно-технічних засобів; на виробництві – для ідентифікації виробів при їх проходженні на автоматизованих робочих місцях; на транспорті – для ідентифікації транспортних засобів на маршрутах руху; у торгівлі – для контролю переміщення товарів [15, 16].

Технологія оптичного розпізнавання символів (OCR – Optical Character Recognition) ґрунтується на використанні певної символіки, якою позначають об’єкти. Пошук символа на поверхні об’єкта та його зчитування реалізують з допомогою фотокамери. Ідентифікація полягає в тому, що символ зчитується з об’єкта й порівнюється з образом, що зберігається в пам’яті комп’ютера. Оптичне розпізнавання символів застосовують для ідентифікації банківських чеків та фінансових документів.

Автоматична ідентифікація на основі розпізнавання голосу (VRS – Voice Recognition Systems) – це технологія, що забезпечує взаємодію людини з об’єктом шляхом розпізнавання окремих голосових команд, що вимовляються людиною. При цьому має розпізнаватись голос будь-якої людини (не лише диктора). Голосове керування об’єктом є особливо зручним, коли необхідно, щоб руки працівника лишалися вільними для виконання виробничих операцій.

У банківській сфері набуває поширення аутентифікація клієнтів на основі голосової біометрії, коли особу ідентифікують за сукупністю унікальних характеристик голосу – голосового зліпку. Голосовий зліпок клієнта – це цифровий код, що зберігається в системі як еталон, та порівнюється з голосом клієнта при його звертанні для надання йому банківських послуг.

У переважній більшості випадків розпізнавання голосу застосовується як частина змішаної технології автоматичної ідентифікації.

Штрихкодова технологія автоматичної ідентифікації (BCI – Barcodes Identification) ґрунтується на застосуванні штрихових кодів (ШК). ШК (Barcode) є послідовністю або масивом темних та світлих смужок (у загальному випадку різноколірних) – штрихів та проміжків різної ширини, що чергуються. У ШК інформацію подають відносні ширини штрихів та проміжків, їх поєднання, а також колір. За допомогою різноколірних штрихів та проміжків можна кодувати цифри, алфавітні та спеціальні символи – загалом довільні символи комп’ютерного алфавіту (ASCII).

Штрихове кодування призначене для швидкого й точного введення в систему обробки інформації, яка характеризує об’єкт обліку. На кожен об’єкт обліку наносять штрихкодову позначку (ШК-позначку), яка супроводжує його на всіх етапах переміщення в певній системі керування (на виробництві, в торгівлі, бібліотечній справі тощо). ШК-позначка (Barcode Symbol) має однозначно характеризувати об’єкт обліку. Її наносять на поверхню об’єкта з дотриманням відповідних правил (стандартів). ШК-позначку зчитують сканером, в тому числі й на відстані. Зчитуючи ШК-позначку, наприклад, на виробництві в різних точках руху об’єкта, можна з високою точністю контролювати його переміщення на різних стадіях виробничого процесу [17, 18].

Особливості штрихового подання даних розглянемо на прикладі чорно-білого ШК EAN-13 – найбільш поширеного штрихового коду в галузі торгівлі (EAN – European Article Numbering, Європейська товарна нумерація) [19, 20].

Наприклад, ШК на рис 1.1. подає десяткову послідовність 4820013456795, яка є ідентифікатором певного виду товару.

Зліва та справа на штриховому зображенні розмішують спеціальні знаки СТАРТ та СТОП, призначення яких – визначати початок та кінець штрихового зображення. Під штриховим зображенням можуть друкувати рядок візуально-читаних символів, які є власне тими даними, які подає штрихове зображення. Якщо зчитати лінійний ШК (перетнути всі штрихи та проміжки сканером), то в систему обробки інформації надійдуть 13 десяткових цифр – так ніби вони були введені з клавіатури.

Для надійного зчитування ШК перед знаком СТАРТ та після знака СТОП мають бути вільні зони – так звані зони стабілізації, які дозволяють сканувати ШК як зліва направо, так і справа наліво. Штрихове зображення разом із зонами стабілізації та можливим рядком візуально – читаних символів утворюють штрихкодову позначку.

Рис. 1. 1. Структура ШК-позначки ШК EAN-13

ШК-позначка складається з послідовності штрихкодових знаків (ШК-знаків). Розрізняють інформаційні та службові ШК-знаки. Інформаційний ШК знак подає відповідний символ ASCII. Службовими є ШК-знаки СТАРТ, СТОП та роздільний ШК-знак, який поділяє ШК-позначку на дві частини. Роздільний ШК-знак може бути відсутній у структурі ШК-позначки.

Перед ШК-знаком СТОП можуть розміщуватись один або два ШК-знаки контрольних символів. Контрольний символ обчислюють за певним правилом, притаманним лише даному ШК, його використовують для перевірки правильності зчитування ШК-позначки.

Кожному ШК-знаку можна поставити у відповідність цифровий еквівалент (двійковий код).

Наприклад, символ 8 на ШК-позначці на рис.1. 1 подає ШК-знак (рис. 1. 2), якому відповідає двійковий еквівалент 0110111; цифра 1 позначає штрих мінімально можливої ширини, а 0 – проміжок мінімально можливої ширини. Штрихи та проміжки є елементами ШК-знака.

Ширину штрихів та проміжків вимірюють в модулях; модуль – це мінімальна ширина, якій крайні розміри всіх елементів штрихового

зображення. Модуль є безрозмірною відносною величиною. Ширина найвужчого елемента (штриха або проміжку) дорівнює одному модулю.

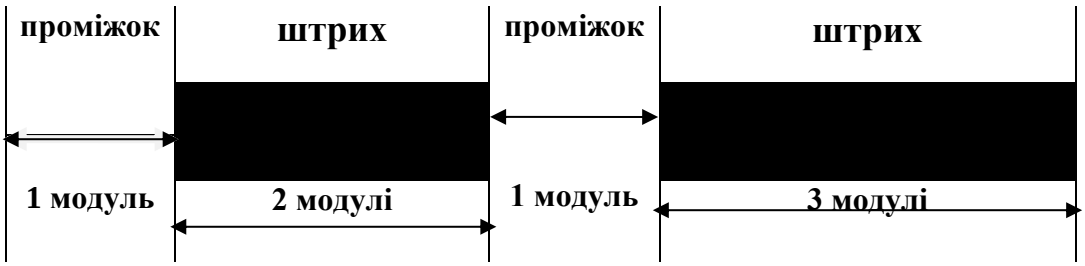


Рис. 1. 2. Структура ШК-знака, який подає цифру 8 в ШК EAN-13

ШК-знак на рис. 1. 2 має ширину 7 модулів, він складається з двох проміжків та двох штрихів, проміжки та штрихи чергуються в зображенні.

У лінійних ШК інформацію подають комбінації штрихів і проміжків та їх ширини, висота штрихів та проміжків не несе інформаційного навантаження [17 - 20].

Кожний ШК має свій, і притаманний лише йому, набір ШК-знаків, які позначають, символи ASCII. Цей набір ШК-знаків називають символікою штрихового коду.

Наприклад, символіка ШК EAN-13 налічує 40 ШК-знаків, поділених на 4 набори по 10 ШК-знаків у кожному. Комбінування 4-х наборів дозволяє неявно кодувати одну десяткову цифру – старшу цифру товарного номера [20].

Можливі символіки, ШК-знаки яких обмежені зліва (ШК-знак починається штрихом і завершується проміжком), справа (ШК-знак починається проміжком і завершується штрихом) або з обох боків – зліва та справа.

Якщо символіка ШК складається з ШК-знаків, обмежених з одного боку, то такий лінійний ШК називають неперервним. Символіка на основі ШК-знаків, обмежених з обох боків, утворює дискретний ШК (у цьому випадку

ШК-знаки у складі ШК-позначки відокремлюють роздільними проміжками) [19].

Крім того, слід брати до уваги паритет ШК-знака. ШК-знак парного паритету – це знак, сумарна ширина штрихів якого (в модулях) є парним числом.

Важливою властивістю штрихового коду є його завадостійкість.

Завадостійким є ШК, який дозволяє виявляти/виправляти помилки зчитування.

Завадостійкість ШК забезпечується двома шляхами - математичним та структурним [19, 20].

У випадку лінійних ШК математичний шлях забезпечення завадостійкості грунтується на приєднанні до інформаційних символів, що підлягають штриховому кодуванню, одного або двох контрольних символів, які є функцією від інформаційних символів і обчислюються за певним правилом. У ШК-позначці контрольному символу відповідає контрольний ШК-знак. У цьому випадку для виявлення помилки зчитування необхідно зчитати ШК-позначку (інформаційні ШК-знаки та контрольний ШК-знак), кожному ШК-знаку поставити у відповідність символ з символіки штрихового коду, обчислити значення контрольного символа і порівняти з щойно зчитаним контрольним символом. Якщо вони збігаються, то вважається, що ШК-позначку зчитано правильно. Інакше приймається рішення про наявність помилки, і тоді зчитування необхідно повторити, або ввести в систему обробки рядок візуально-читаних символів вручну (з клавіатури).

Структурний шлях забезпечення завадостійкості ШК ґрунтується на тому, що у структурі ШК-знаків закладена певна надлишковість, яка забезпечує виявлення помилки зчитування. Правильність зчитаного ШК-знака визначають за кількістю елементів у ШК-знаку, за паритетом ШК-знака, за довжиною ШК-знака в модулях, за співвідношенням вузьких та широких штрихів або проміжків тощо.

Штрихове кодування має великі можливості щодо розширення сфери застосування. Це зумовлено низькою собівартістю виготовлення ШК-позначок, простотою використання та розвиненістю технічних засобів сканування штрихкодових зображень. Штрихові коди широко застосовуються у торгівлі, промисловому виробництві, на транспорті, в медицині, сфері послуг, в галузі фінансів, у поштовій галузі, військовій справі тощо. На основі обробки штрихкодової інформації можна створювати ефективні системи управління, що мають на меті зниження виробничих витрат та підвищення продуктивності праці персоналу.

Штрихове кодування є найпоширенішим видом автоматичної ідентифікації.

1. 3. Порівняльний аналіз двоколірних штрихових кодів

Штрихові коди з часу їх винайдення та перших застосувань (60-ті роки минулого століття) у своєму розвитку пройшли певну еволюцію – чорно-білі лінійні ШК, стекові ШК, матричні ШК, багатоколірні ШК [21-23].

Лінійні ШК можуть подавати лише невеликі обсяги інформації (до 20 - 30 символів ASCII), тому вони є лише ключем доступу до бази даних, у якій зберігається повна інформація про об’єкт обліку. Для подання більших обсягів даних – від кількох сотень до кількох тисяч символів ASCII, використовують ШК з двовимірною структурою ШК-позначки – стекові та матричні ШК. У багатьох випадках ємність ШК-позначки стекового або матричного ШК є достатньою для зберігання повної інформації про об’єкт, тому іноді вважають, що двовимірні ШК є переносними файлами даних (portable datafile) [21, 23].

Лінійні ШК поділяють на два класи – цифрові (для подання десяткових цифрових послідовностей) та алфавітно-цифрові (для подання текстових даних) (Додаток А).

Порівняємо лінійні ШК за показниками: склад символіки, структура ШК-знаків, спосіб подання даних, інформаційна щільність ШК-знаків та кількість контрольних символів у ШК-позначці (табл. А. 1).

Вважають, що одним з перших ШК став штриховий код 2/5 Code (Two of five Code), розроблений у 1968 р. компанією Identicon Corporation (США); використовувався в сортувальних системах, а згодом – для реєстрації авіаквитків.

ШК-позначка 2/5 Code може містити у своєму складі контрольний ШК-знак. Обчислення контрольного символу виконують за алгоритмом, що має назву "модуль 10, фактор 3".

Серед цифрових лінійних ШК з двома градаціями ширини елементів за показником інформаційної щільності вирізняється ШК ITF (ITF – Interleaved Two of Five – ШК "два з п'яти з чергуванням") – через незвичний спосіб утворення ШК-знаків, який використовує ту саму символіку, що й 2/5 Code. У ШК ITF інформація подається як штрихами, так і проміжками [24]. Цифрову послідовність, яку необхідно подати в штриховому вигляді, розбивають на цифрові пари. У цифровій парі ліву цифру кодують штрихами (прямим ШК-знаком цифри), а праву – проміжками (інверсним ШК-знаком цифри); при цьому штрихи лівої цифри чергують з проміжками правої цифри (переріз двох ШК-знаків). Тому цифровій парі відповідає ШК-знак, що складається з 5-ти штрихів та 5-ти проміжків. Це забезпечує високу інформаційну щільність коду [24].

До алфавітно-цифрових ШК належать Code 39, Code 93 та Code 128, які дозволяють подавати у штриховому вигляді текст, утворений символами ASCII. У цих ШК інформацію подають як штрихи, так і проміжки. У ШК Code 93 та Code 128 використовують 4 градації ширини елементів, а в Code 39 – дві градації.

При поданні у штриховому вигляді цифрових послідовностей Code 128 забезпечує подвійну щільність, оскільки при цьому один ШК-знак подає цифрову пару (з діапазону 00 - 99) [22].

Стековий ШК є послідовністю лінійних ШК зі зменшеною висотою штрихів, розміщених один над одним (Додаток Б).

Розвиток стекових ШК припав на період 1985 - 1996 р.р. Першим стековим ШК став Code 49, розроблений у 1987 р. компанією Intermec Corporation (США) (табл. Б. 1) для потреб електронної промисловості (маркування мікросхем), а також для маркування медичних інструментів. ШК-позначка цього коду може містити від 2-х до 8-ми рядків; структура рядка – ШК-знак СТАРТ, інформаційні ШК-знаки, ШК-знак СТОП, кожен рядок складається з 18 штрихів та 17 проміжків. Відношення висоти штрихів до ширини найвужчого елемента складає 8:1. У рядку можна подати 7 символів ASCII, завершальним (8-м символом) є контрольний символ рядка, який обчислюють за модулем 49. Code 49 дозволяє подавати у штриховому вигляді довільні символи ASCII. Останній рядок ШК-позначки містить два контрольні символи – контрольний символ рядка та загальний контрольний символ усієї ШК-позначки, який обчислюють за модулем 2401.

Максимальні інформаційна ємність ШК-позначки Code 49 становить 55 символів ASCII [25].

ШК-позначка стекового штрихового коду Codablok, розробленого німецькою компанією Identcode-System у 1989 р. для потреб медичної та електронної індустрії, може включати 2 - 44 рядки лінійних ШК Code 128 [26].

Максимальна інформаційна ємність ШК-позначки Codablok становить 2725 символів ASCII [26].

Для застосування у галузі охорони здоров’я в 1989 р. було розроблено стековий ШК Code 16K. За прототипом було взято лінійний ШК Code 128. Оскільки $128^2 = 2^{14} = 16K$, то розробник коду (Ted Williams) вирішив назвати його як Code 16K [141]. Один рядок цього коду складається з 5-ти ШК-знаків. У штриховому вигляді можна подавати будь-які символи ASCII. За рахунок спеціального ШК-знака (FNC4) можна перейти до розширеного ASCII. Оскільки ШК-позначка може містити від 2-х до 16-ти рядків, то максимальна інформаційна ємність ШК-позначки Code 16K становить 78 символів ASCII

(останній рядок містить два контрольні символи які обчислюють за модулем 107). Якщо в штриховому вигляді подавати лише цифрові дані, то максимальна інформаційна ємність ШК-позначки становить 156 десяткових цифр [27-28].

Серед стекових ШК останньою розробкою (1996 р.) став ШК Ultracode американської компанії Zebra Technologies, який можна використовувати в двох варіантах – чорно-білому та багатоколірному [26].

Найбільшого поширення серед стекових ШК набув код PDF417, розроблений 1991 р. компанією Symbol Technologies (США) [21]. Його назва є скороченням слів Portable Data File – переносний файл даних, та характеризує структуру ШК-знаків – ШК-знак має ширину 17 модулів і до його складу входять 4 штрихи.

У PDF417 для гарантування цілісності даних використовують два способи – виявлення помилок (до складу ШК-позначки обов'язково мають входити 2 контрольні символи (ШК-знаки) та виправлення помилок на основі коректувального коду Ріда-Соломона (це не обов'язкова компонента). При цьому залежно від кількості ШК-знаків у ШК-позначці та умов використання ШК пропонується 9 рівнів корекції помилок (рівень 0 – рівень 8) [21].

Розроблення стекового ШК PDF417 стало віхою в галузі штрихового кодування даних. Запроваджені в ньому три новації – поділ символіки ШК на набори символів, ущільнення даних, застосування завадостійкого кодування для забезпечення цілісності даних, - стали підґрунтям для створення ШК нового виду – матричних штрихових кодів великої інформаційної ємності (Додаток В).

Матричні ШК відрізняються від лінійних та стекових ШК тим, що в них дані подаються з повноцінним використанням ортогональних осей – як по горизонталі, так і по вертикалі. Найчастіше ШК-позначки мають вигляд квадрата або прямокутника та складаються з чорно-білих чарунок (комірок), які можуть мати різну форму – квадрат, шестикутник, овал тощо [25, 28].

Для визначення орієнтації на площині до складу ШК-позначки вводять спеціальні маркери, які можуть розміщуватися в кутах штрихового зображення, в центрі (bull eye), або у вигляді L-подібного бордюрного рисунка. Чорно-білі матричні ШК розроблено у 80-х – 90-х роках минулого століття (табл. В. 1).

ШК-позначки матричних ШК дозволяють подавати більші обсяги даних (табл. В. 1) порівняно зі стековими ШК. Найбільшого поширення серед матричних ШК набув QR Code (QR – Quick Response, швидка відповідь), розроблений у 1994 р. QR Code передбачає використання 40 типорозмірів ШК-позначок – найменша ШК-позначка має розміри 21 х 21 модулів (чарунок), найбільша – 177 х 177 модулів (чарунок), крок зміни розміру ШК-позначки складає 4 модулі [23].

QR Code дозволяє подавати у штриховому вигляді довільні символи розширеного ASCII, бінарні значення, а також символи японської ієрогліфічної системи запису Кандзі. ШК-позначка максимального розміру (177 х 177) дозволяє подати 4464 алфавітно-цифрові символи або 7366 лише цифрових символів.

У QR Code для контролю помилок використовують код Ріда-Соломона. Передбачено 4 рівні забезпечення завадостійкості – Level L, Level M, Level Q, Level H. Рівень завадостійкості L забезпечує відновлення даних, якщо ушкоджено до 7% загальної площі ШК-позначки, рівень M – до 15%, рівень Q - до 25%, а рівень H – до 30% [23, 29].

Матричні ШК мають широке застосування через їх головні переваги – велику інформаційну ємність, автономність та компактність. Але їх недоліком є те, що вони лише двоколірні.

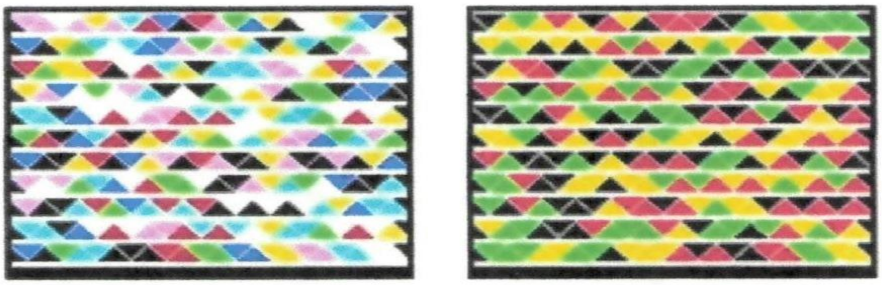
1. 4. Тенденції розвитку багатоколірного штрихового кодування даних для потреб медичної галузі

Останніми роками має місце тенденція до розширення сфери застосування багатоколірних ШК, адже багатоколірність дозволяє у декілька разів підвищити інформаційну щільність – на одиниці площі носія без зміни геометричних розмірів елементів штрихкодового зображення подавати більшу кількість інформації, аніж це дозволяють чорно-білі ШК [30 – 39, 45 – 58, 60].

Компанія Microsoft у 2007 р. розробила 4-колірний ШК великої ємності High Capacity Color Barcode (HCCB), в якому використовують чорний, червоний, зелений та жовтий кольори [35]. Деякі дослідники небезпідставно вважають, що саме з цього коду (який називають також Microsoft Tag) розпочалось багатоколірне штрихове кодування [34]. Перші дослідження HCCB виконали працівники компанії Devi Parikh та Gaving Janche, які в [35] запропонували метод декодування цього коду, основними процедурами якого є локалізація та сегментація колірного зображення ШК-позначки. Колірне зображення розбивають на кластери, кожному центру кластера ставлять у відповідність один із еталонних кольорів палітри, що використовується при друкуванні ШК-позначки. У HCCB інформацію подають колірними трикутниками, які розміщують на носії у вигляді матриці – кількість рядків у ШК-позначці може становити від 10-ти до 60-ти, у рядку – від 20-ти до 120-ти трикутників.

Трикутник подає четвіркове значення 0, 1, 2 або 3. Таким чином, ємність ШК-позначки може становити від 200 до 7200 четвіркових цифр [30].

У подальших версіях цього ШК використовували також 8-ми колірну палітру (рис.1.3). За потреби можливе використання двоколірної (чорно-білої) версії коду. Розробники передбачили зчитування коду з екрана смартфона. Застосування HCCB спрямовувалось на введення даних в сервер компанії Microsoft.

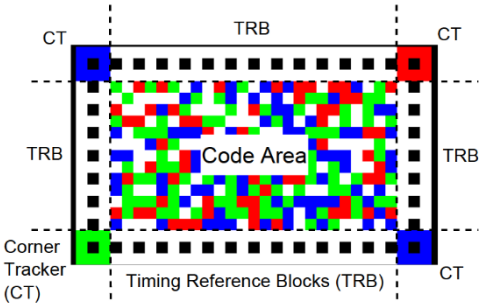


4-колірний 8-колірний

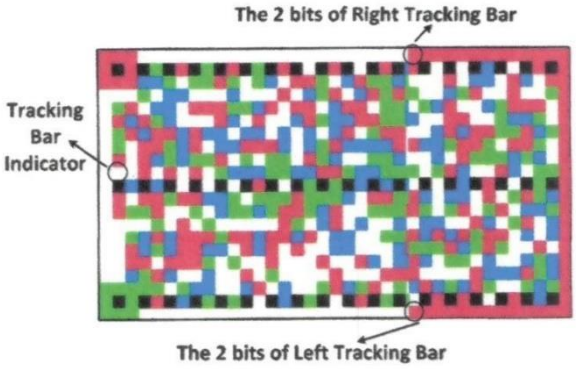
Штриховий код HCCB

4-колірний 8-колірний

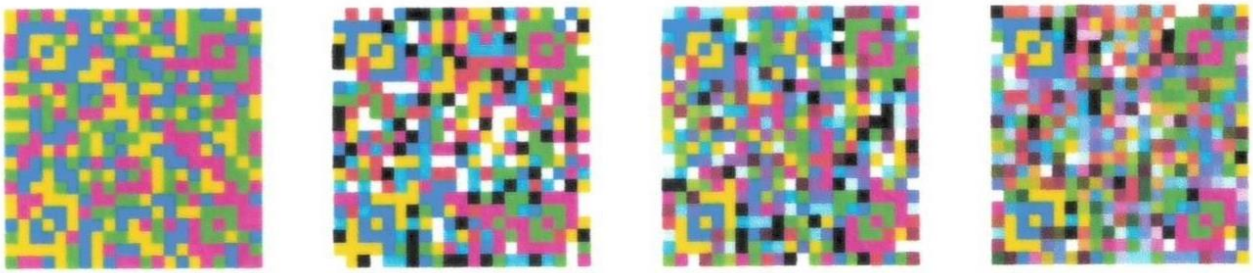
Штриховий код HCC2D



Штриховий код Cobra



Штриховий код RainBar



4-колірний 8-колірний 16-колірний 32-колірний

Штриховий код JAB

Рис. 1. 3. Зовнішній вигляд ШК-позначок деяких багатоколірних ШК

Дослідження штрихового коду HCCB показало, що розпізнавання (ідентифікація) кольорів на штрихкодовому зображенні є нетривіальною задачею, а інформаційна щільність, на жаль, обмежується істотними витратами на виправлення помилок не тільки через геометричні, а й хроматичні спотворення, що виникають під час друкування та сканування багатоколірних ШК [36 - 39].

У 2011 р. було запропоновано штриховий код HCC2D (High Capacity Colored 2-Dimensional color barcode) як розвиток багатоколірності QR-коду. Пропонувалось використовувати у QR-коді 4 або 8 кольорів (рис. 1.3). Для цього в структуру QR-коду у правому нижньому куті зображення пропонувалось додатково ввести маркер колірності (Color Palette Pattern), у якому кількаразово повторюються еталонні колірні модулі (чарунки). Під час сканування зображення колір чергового елемента порівнюється з відповідним еталонним кольором – для перевірки правильності ідентифікації оброблюваного кольору.

Автори розробки стверджують, що HCC2D забезпечує приблизно таку саму щільність даних, що й HCCB, та втричі більшу за чорно-білий QR-код, і при цьому зберігаються інші властивості QR-коду [33].

HCC2D розроблявся так, щоб забезпечувати зчитування багатоколірного штрих кодового зображення як з паперового носія, так і з екрана смартфона [39].

Ідея коду HCC2D полягала в наступному. Інформаційне повідомлення, яке підлягає поданню у вигляді чорно-білого QR-коду розбивають на дві рівні частини, і кожну з них подають у вигляді окремої ШК-позначки QR-коду. Далі дві ШК-позначки логічно (не фізично) накладають одна на одну – утворюються дві площини (нижня та верхня) з однаковими геометричними розмірами. А далі розглядають вертикальний зріз для кожної комірки (чарунки) зображення. Якщо чарунка з координатами (i, j) нижньої площини, наприклад, біла (0), а верхньої – чорна (1), то утворюють колірний вектор (0, 1). Зрозуміло, що можливі чотири колірні вектори: (0, 0); (0, 1); (1, 0);

(1, 1). І тоді, комірку з координатами (*i, j*) результуючого зображення зафарбовують відповідним кольором, наприклад, 00 – червоний, 01 – зелений, 10 – синій, 11 – білий. Таким чином, утворюється 4-колірна ШК-позначка, яка подає первісне інформаційне повідомлення, але займає вдвічі меншу площу на носії.

При зчитуванні 4-колірного штрихкодового зображення виконують зворотні перетворення.

Якщо інформаційне повідомлення, що підлягає поданню у вигляді чорно-білого QR-коду, розбити на 3 рівні частини, то результуюча ШК-позначка буде 8-колірною; на 4 рівні частини – ШК-позначка буде 16-колірною.

У подальшому набула розвитку ідея передачі інформації з одного смартфона на інший смартфон засобами штрихового кодування даних – як альтернатива передачі даних через мережу або за допомогою радіозв'язку (WiFi, Bluetooth) [45]. Безпосередній зв'язок двох смартфонів застосовують для передачі конфіденційної інформації – зберігання біометричних та персональних даних, голосового зліпку, фото, офіційних документів (текст договору, паспорт, військове посвідчення, проїзні документи тощо).

Для цієї мети у 2012 р. було запропоновано багатоколірний ШК COBRA (COlor BaRcode streaming for smArtphone) [47, 48] – для передачі інформації між двома марафонами; штрихкодове зображення з екрана смартфона зчитують камерою іншого смартфона.

У ШК Cobra використовують чотири колірні кутові маркери CT (Color Corner Tracker) (див.рис.1.3) – для швидкої локалізації та орієнтації зображення на площині, та додатково часові позначки TRB (Timing Reference Block) по периметру зображення – для синхронізації рядків та стовпців зображення при скануванні.

Для подання даних у ШК Cobra застосовують 4 кольори – червоний, зелений, синій, білий, які кодують як 00, 01, 10, 11 відповідно, а п'ятий колір – чорний, використовують для часових позначок та бордюрних стовпців

справа та зліва від матриці даних (чорний колір є службовим, він не несе інформаційного навантаження). Отже, колірна чарунка масиву даних подає 2 біти інформації, або одну четвіркову цифру (0, 1, 2 або 3).

Наприклад, байт даних 10110001 буде закодовано як {синій білий червоний зелений}. На екрані смартфона, наприклад, розміром 4,65 дюйми з роздільною здатністю 1280 х 720 код Cobra у штриховому вигляді дозволяє подати 47 Кбіт [156].

Зв'язок двох смартфонів з допомогою багатоколірного штрихового кодування даних називають візуальною комунікаційною (VLC) системою (Visible Light Communication System).

З метою удосконалення VLC-систем на основі багатоколірного штрихового кодування розробниками коду Cobra було запропоновано новий ШК – RainBar , назва якого утворена від Robust Application-driven vIsual communicatioN using BARcodes [45].

ШК RainBar є розвитком ШК Cobra. У цьому коді використано таку саму колірну палітру, що й у коді Cobra. Відмінністю є те, що для збільшення інформаційної щільності даних залежно від якості екрана смартфона можуть додатково використовуватись ще три кольори. Крім того, внесено зміни до структури ШК-позначки – використовуються лише два кутові маркери (див. рис. 1. 3) та внесені змін до бордюрного рисунка.

RainBar забезпечує більш надійне зчитування даних за рахунок врахування додаткових факторів, що вносять спотворення в штрих- кодове зображення. Контроль даних ґрунтується на застосуванні контрольних сум та коду Ріда-Соломона [45].

У 2018 р. було оголошено про створення багатоколірного матричного ШК з назвою JAB – Just Another Barcode (Німеччина, Інститут Фраунгофера) [60].

У базовому варіанті JAB-коду використовують 8 кольорів колірного простору RGB, які кодують трирозрядними послідовностями: 000 – чорний,

001 – синій, 010 – зелений, 011 – блакитний, 100 – червоний, 101 – пурпуровий, 110 – жовтий, 111 – білий.

Форма ШК-позначки JAB-коду може бути квадратною (мінімальний розмір: 21 х21 модуль, максимальний: 145 х145 модулів), прямокутного (мінімальний розмір: 21 х25 модулів, максимальний: 141 х 14 модулів), або U-подібною (табл. 1.1).

Контроль помилок при зчитуванні ШК-позначок здійснюють на основі LDPC (Low Density Parity Check**)** або на основі коду Ріда-Соломона. У JAB-коді передбачено 11 рівнів завадостійкості (занумерованих від 0 до 10) [60].

Результативність декодування залежить від якості друку, можливих ушкоджень зображення, а також від інших факторів.

Таблиця 1. 1

Деякі параметри найбільш поширених багатоколірних штрихових кодів

Назва ШК	Кількість використо-вуваних кольорів	Форма мінімального елемента	Форма ШК-позначки	Контроль помилок	Рік розроб-лення
HCCB	4, 8	трикутна	квадратна	на основі алго-ритму RSA-1024	2007
HCC2D	4, 8, 16	квадратна	квадратна	на основі коду Ріда-Соломона	2011
Cobra	4	квадратна	квадратна, прямокутна	на основі коду Ріда-Соломона	2012
RainBar	4, 7	квадратна	квадратна, прямокутна	на основі коду Ріда-Соломона	2015
JAB	4, 8, 16, 32	квадратна	квадратна, прямокутна, U-подібна	на основі конт-ролю на пар-ність або коду Ріда-Соломона	2018

При проєктуванні програмного забезпечення систем автоматичної ідентифікації на основі багатоколірних штрихових кодів слід брати до уваги особливості роботи з кольорами.

У колірній моделі RGB (Red – червоний, Green – зелений, Blue – синій), відтінок кольору визначають шляхом визначення пропорції змішуванні базових кольорів. У цій моделі можливо утворити білий (White) колір як результат змішуванні базових кольорів, тобто W = R + B + G [48].

Іншою моделлю, у якій відтінок кольору визначають в такий самий спосіб, є модель CMY (Cyan – блакитний, Magenta – пурпуровий, Yellow – жовтий), у якій базовими кольорами є C, M, Y. Шляхом змішування базових кольорів можливо утворити чорний (black) колір, і тоді отримують модифіковану модель, яку названо CMYK, де K= C + M + Y [46, 48].

У багатьох випадках у сканувальному обладнанні застосовують модель RGB, а в друкарському – CMYK. У такому разі при друкуванні ШК-позначок здійснюють перетворення кольорів (числових значень, що відповідають кольорам) моделі RGB в кольори моделі CMYK, а при зчитуванні ШК-позначок з паперового носія – зворотне перетворення (CMYK в RGB) [182]. Числове значення кольору в моделі RGB належить діапазону 0 – 255, а в моделі CMYK – діапазону 0 – 1 (числове значення з діапазону 0 – 255 ділять на 255) [57, 58].

При зчитуванні багатоколірного ШК з екрана смартфона використовують колірну модель HSV (hue, saturation, value), або модель HSL (hue, saturation, lightness), у яких використовують інший підхід до визначення відтінка кольору – через значення яскравості.

Крім того, слід брати до уваги те, що надрукована ШК-позначка є відкритим носієм інформації, що може зазнавати зовнішніх впливів.

ШК-позначка може зазнавати спотворень на різних етапах її життєвого циклу – при виготовленні, зберіганні, експлуатації. Спотворення можуть носити механічний характер або бути спричиненими особливостями застосовуваних барвників. Спотворення механічного характеру: дефекти носія (нерівність поверхні, низька якість матеріалу), дефекти друку внаслідок механічних збоїв друкарського обладнання, деформація носія під час транспортування), забруднення та механічні ушкодження при експлуатації.

Спотворення, пов’язані із застосуванням барвників: зчеплення барвника з носієм (розпливання, згортання фарби), змішування кольорів сусідніх елементів зображення, старіння (вицвітання) барвника, неконтрастність. Внаслідок змішування кольорів сусідніх елементів можливе або утворення нового кольору, або розмиття зображення [61].

У VLC-системах на основі ШК (зчитування з екрана смартфона), як зазначають деякі дослідники [45, 46], можуть виникати додаткові збурювальні фактори, що впливають на точність відтворення даних з багатоколірного штрихкодового зображення: динамічна зміна середовища сканування (зміна освітлення, затінення), зміна відстані та кута сканування між камерою та екраном смартфона, тремтіння руки оператора, розмитість або перекошування колірного зображення тощо [45, 61].

Таким чином, для забезпечення подальшого розвитку штрихового кодування як однієї з передових інформаційних технологій необхідно вживати додаткових заходів щодо гарантування цілісності даних, відтворюваних з багатоколірних ШК-позначок. Досягти цього можливо лише шляхом забезпечення належного рівня завадостійкості багатоколірних штрихкодових зображень.

Багатоколірний штриховий код слід створювати так, щоб властивість завадостійкості забезпечувалась на двох рівнях – на рівні усієї ШК-позначки, а також на рівні ШК-знаків – мінімальних структурних одиниць зображення. Саме такий підхід слід застосовувати при виборі штрихового коду для медичної інформаційної системи. Це забезпечить надійність автоматичної ідентифікації об’єктів медичного документування.

1. 5. Вимоги до програмної системи автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів

Створенню працездатного і якісного програмного продукту має передувати розроблення вимог до програмного забезпечення [62 - 77]. Розроблення вимог є окремим технологічним процесом, який має свої етапи,

процедури та особливості реалізації. Результатом цього процесу є підготовлена специфікація вимог до програмного забезпечення системи, що має розроблятися [67, 71].

Вимоги до програмного забезпечення – це набір потреб потенційних користувачів, зацікавлених сторін щодо функцій, якості та властивостей програмного продукту, який потрібно розробити [64]. Розробка вимог може складатися з кількох етапів: знаходження вимог (збір, визначення потреб користувачів та зацікавлених осіб), аналіз вимог, документування вимог (специфікація), тестування вимог.

Розрізняють три рівні вимог до програмного забезпечення [62 - 67]: бізнес-вимоги – визначають призначення ПЗ, найважливіші ідеї, покладені в основу розробки, вимоги або політики компанії, організація фінансового обліку, норми оподаткування тощо; вимоги користувача – визначають перелік завдань користувача, які повинно вирішувати програмне забезпечення, сценаріїв їх вирішення, ці вимоги можуть мати вигляд тверджень, варіантів використання, сценаріїв взаємодії, прецедентів; функціональні вимоги – визначають, що саме має виконувати програмний продукт, вони докладно описуються в документі, що має назву «Специфікація вимог до програмного забезпечення».

При розробленні вимог важливим є етап їх аналізу. Аналіз вимог полягає у визначенні потреб користувачів та замовників, їх узгодження, збалансування, а також умов, що висуваються до розроблюваного програмного продукту. Для успішності розробки аналіз вимог має бути критичним. Метою аналізу вимог є виявлення недоліків формування вимог (неточностей, неповноти, неоднозначностей тлумачення чи суперечностей) та їх виправлення. Вимоги мають бути описаними з рівнем деталізації достатнім для конструювання програмної системи, вимірними, тестовними, пов’язаними з бізнес-потребами [75].

При розробленні вимог до створюваного програмного продукту вимоги можуть моделювати. Наприклад, у мові моделювання SysML вимоги

моделюють за допомогою діаграми вимог, а в UML для цього використовують діаграми прецедентів.

Специфікація вимог до ПЗ – це спеціальний документ, що містить повний опис поведінки системи, що розробляється. Специфікація включає множину функціональних вимог (прецедентів), які описують можливі взаємодії користувачів з програмним забезпеченням, а також включає нефункціональні (додаткові) вимоги.

Специфікація складається з окремих розділів, наприклад: вступ (загальна інформація про продукт, мета, зв'язок з іншими системами); загальний опис продукту (перспективи продукту, функції ПЗ, характеристики можливих користувачів, загальні обмеження); конкретизація вимог (вимоги до зовнішніх інтерфейсів (інтерфейс користувача, комунікаційний протокол, обмеження пам’яті, набір операцій), властивості програмного продукту, вимоги до бази даних тощо. Наявність специфікації вимог знижує загальну вартість і час розробки, оскільки дозволяє уникнути переробки через зміни вимог [63].

Для однозначності визначення вимоги недостатнім є лише текстове її формулювання. Вимогу доцільно доповнювати набором атрибутів. Прикладом набору атрибутів деякої вимоги може бути: дата створення вимоги, автор вимоги, стан вимоги, можливість перевірки вимоги, спосіб перевірки вимоги, пріоритет реалізації вимоги тощо.

Під час розроблення вимог до ПЗ рекомендують використовувати категорії атрибутів, наведені в табл. 1. 2 [67].

Добре сформульовані вимоги до ПЗ та правильно визначені набори атрибутів кожної з вимог істотно впливають на ефективність роботи зацікавлених сторін (системних аналітиків, архітекторів проєкту, конструкторів програмного коду, тестувальників) [67 - 69].

Таблиця 1. 2

Категорії атрибутів, які доцільно використовувати
під час розроблення вимог до ПЗ

Категорія атрибутів	Приклади значень атрибутів
1	2
Ідентифікація вимоги	
- ідентифікація	унікальний номер вимоги (ID)
- назва	унікальна коротка назва, що характеризує вимогу
Внутрішні характеристики вимоги	
- основний тип	функціональність, продуктивність системи, якість, оточення, інтерфейс, обмеження на вимогу
- якісний підтип	доступність, гнучкість, цілістність, можливість зміни, компактність, легкість підтримки та використання, кваліфікація
-тип продукту/процесу	продукт, процес, дані чи сервіс етапу реалізації проєкту
- кількісний/якісний тип	кількісний, якісний продукт етапу реалізації проєкту
-етап життєвого циклу	попередня чи остаточна концепція, процес розроблення, виготовлення, інтеграція/ тестування, впровадження, поставка, встановлення, функціону-вання, підтримка, видалення, демонтаж, передавання чи утилізація
Пріоритет вимоги	
- пріоритет	ключова, необхідна, додаткова, бажана (Key, Mandatory, Optional, Desirable) або обов’язкова, рекомендується, можлива, доцільна (Must, Should, Could, Wish)

продовження табл. 1.2

1	2
Джерело і автор вимоги	
- спосіб отримання	призначення, декомпозиція системи
- джерело	назва документа або ім’я зацікавленої особи
- автор	ім’я зацікавленої особи
- погоджено	ім’я експерта чи керівника вищого рівня
Підтримка процесу реалізації вимоги	
- статус узгодження	запропоновано, на узгодженні, погоджено
- статус перевірки	перевірено, не перевірено
- статус задоволення	незадоволено, задоволено
- статус рецензування	чекає на аналіз, прийнято, відхилено
Уточнення вимоги	
- потреба	опис того, чому виникла потреба в цій вимозі
- коментарі	текстове уточнення вимоги для кращого розуміння її суті
- питання	питання, які потрібно з’ясувати для уточнення вимоги
- відповіді	відповіді, отримані при уточненні вимоги
Інші показники вимоги	
- зрілість (стабільність)	кількість змін, тривалість змін
- рівень ризику	високий, середній, низький
- оцінювана вартість	вартість, розрахована експертами
- фактична вартість	вартість, продиктована ринком
- реліз продукту	версія програмного продукту, в якій реалізована вимога

Вимоги до ПЗ, що супроводжуються наборами атрибутів, мають надавати учасникам розробки такі можливості : можливість ідентифікувати кожне положення вимоги; можливість класифікувати вимогу за: важливістю реалізації (пріоритетом виконання), терміновістю (дата) реалізації; можливість відстежувати кожну вимогу за статусами: статусом аналізу (рецензування), статусом задоволення, статусом перевірки; можливість оцінювати вимогу стосовно стратегії її тестування тощо.

Для розроблення програмного забезпечення процесів автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах необхідно визначити функціональні та нефункціональні вимоги до ПЗ.

Функціональні вимоги є переліком функцій та сервісів, які повинно надавати розроблюване програмне забезпечення для задоволення потреб користувачів у рамках медичної інформаційної системи.

Нефункціональні вимоги визначають додаткові характеристики програмної системи, такі як надійність, безпека, продуктивність, масштабованість, відновлюваність, зв’язки інтерфейсу з іншими системами або обладнанням тощо. Нефункціональні вимоги безпосередньо не впливають на роботу програмного забезпечення.

Функціональні вимоги (ФВ) до програмної системи автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах наведено в табл. 1.3. Нефункціональні вимоги не розглядатимемо.

У розроблювальній програмній системі важливість функціональної вимоги будемо характеризувати атрибутом пріоритету.

Загалом, необхідність пріоритизації вимог до розроблюваного ПЗ зумовлюється можливою обмеженістю людських, технічних, фінансових ресурсів проєкту. Метою пріоритизації вимог може бути оптимізація часу розробки, вартості розробки, максимізація прибутку тощо [75].

Пропонується визначати пріоритетність вимоги у числовому вигляді (на відміну від наведеної табл. 1. 2, де пріоритетність вимоги рекомендують визначати якісним (нечисловим) показником – “ключова”, “необхідна”, “бажана”) на основі таких п’яти числових критеріїв: Вигода (Benefit), Втрата (Penalty), Вартість (Cost), Ризик (Risk) та Близкість вигоди (Benefit proximity).

Критерій Benefit означає вигоду, яку отримає замовник у разі успішного виконання даної вимоги. Ця вигода може бути виражена в різний спосіб. Це може бути пряма фінансова вигода, наприклад, вимога реалізації функції платної підписки в деякому цифровому сервісі або вигода репутаційна, коли наявність такої функції в системі працює на просування даного програмного продукту на ринку.

Критерій Penalty означає втрати, яких зазнає замовник у випадку, якщо з якихось причин дану вимогу не вдасться реалізувати. Це можуть бути прямі фінансові втрати, недоотриманий прибуток або репутаційні втрати.

Критерій Cost визначає те, скільки ресурсів необхідно витратити для того, щоб реалізувати дану вимогу. Під ресурсами можуть матись на увазі фінансові ресурси, наприклад, витрати на оплату праці команди розробників, закупівлі необхідного обладнання тощо, або час розробки, тобто людиногодини.

Критерій Risk означає оцінку ймовірності нереалізації даної вимоги за того бюджету та/або часу, який виділено для реалізації даної вимоги. Серед можливих факторів, які впливають на оцінку даного критерію, можна виділити такі як: експертний рівень команди, брак попереднього досвіду, невизначеність, технічна складність тощо.

Критерій Benefit proximity означає швидкість досягнення вигоди.

Пропонується значення критеріїв виражати цілим числом від 1 до 10 шляхом експертного оцінювання.

Наприклад, критерій Benefit proximity будемо трактувати так: 1 означає віддалену вигоду, тобто таку, яка реалізується у віддаленому майбутньому, а 10 – моментальна вигода, тобто така, що реалізується якнайшвидше, тобто

вигоду від реалізації даної вимоги отримують одразу як тільки вона стає доступною для користувачів.

Менеджери та архітектор проєкту визначають числове значення кожного критерію за консенсусом та заповнюють відповідні п’ять стовпців табл. 1. 3 для кожної вимоги.

Загальний пріоритет функціональної вимоги (ФВ) пропонується визначати за формулою

$$\text{Пріоритет ФВ} = \frac{Benefit \cdot \left(1 + \frac{Benefit\ proximity}{10}\right) \cdot Penalty}{Cost \cdot Risk} \qquad (1.\ 1)$$

Наприклад, пріоритет функціональної вимоги ФВ1 з табл. 1. 3, обчислений за формулою (1. 1), дорівнює 4.0.

Заповнення стовпця Загальний пріоритет вимоги табл. 1. 3 дає можливість менеджеру проєкту планувати та організовувати послідовність виконання робіт командою розробників програмного забезпечення.

Упорядкувавши стовпець Загальний пріоритет вимоги за спаданням бачимо, що найбільш важливими функціональними вимогами є:

ФВ13 Забезпечення завадостійкого кодування даних,

ФВ14 Забезпечення формування багатоколірної ШК-позначки,

ФВ15 Забезпечення зчитування та декодування багатоколірної ШК-позначки,

які необхідно реалізувати в першу чергу; для цих вимог показник загального пріоритету становить 6,0.

Аналізуючи цей стовпець зручно розподіляти роботи між командами виконавців з урахуванням їх кваліфікації та кількісного складу, а також виконувати квартальне планування робіт.

На основі аналізу та пріоритизації функціональних вимог розробимо архітектуру програмної системи автоматичної ідентифікації (підрозділ 4. 1).

Таблиця 1. 3

Функціональні вимоги до програмної системи автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах

Код вимоги	Вимога	Критерії пріоритету вимоги					Загальний пріоритет вимоги
		Вигода	Близькість вигоди	Витрата	Вартість	Ризик	
1	2	3	4	5	6	7	8
ФВ1	Забезпечення введення даних лабораторних досліджень	2	6	5	2	2	4,0
ФВ2	Забезпечення зберігання та доступу до даних лабораторних досліджень	4	6	5	3	4	2,7
ФВ3	Забезпечення введення даних щодо дозування медичних препаратів	1	4	3	2	2	1,1
ФВ4	Забезпечення зберігання та доступу до даних щодо дозування медичних препаратів	2	4	3	3	4	0,7
ФВ5	Забезпечення введення даних про діагноз пацієнта	2	5	7	2	2	5,3
ФВ6	Забезпечення зберігання та доступу до даних про діагноз пацієнта	5	5	7	3	4	4,4
ФВ7	Забезпечення введення даних про відвідування лікаря	2	6	3	2	2	2,4
ФВ8	Забезпечення зберігання та доступу до даних про відвідування лікаря	2	6	3	3	4	0,8
ФВ9	Забезпечення ієрархії рівнів доступу до даних	3	2	2	8	6	0,2
ФВ10	Забезпечення підсистеми надання доступу до медичних даних пацієнтів	8	5	8	9	8	1,3
ФВ11	Забезпечення створення, модифікації розкладу роботи медичного персоналу	6	7	3	4	3	2,6
ФВ12	Забезпечення обліку медичних препаратів	2	3	1	6	5	0,1

продовження таблиці 1.3

1	2	3	4	5	6	7	8
ФВ13	Забезпечення завадостійкого кодування даних	5	2	6	2	3	6,0
ФВ14	Забезпечення формування багатоколірної ШК-позначки	5	2	6	2	3	6,0
ФВ15	Забезпечення зчитування та декодування багатоколірної ШК-позначки	5	2	6	2	3	6,0
ФВ16	Забезпечення обміну медичними даними за допомогою смартфон-додатка	8	9	6	8	2	5,7
ФВ17	Можливість вибору кількості кольорів для подання штрихкодових данних	6	4	4	3	2	5,6
ФВ18	Можливість вибору типу коректуваль-ного коду при штрихкодовому кодуванні	2	1	2	3	2	0,7
ФВ19	Забезпечення можливості сканування багатоколірного ШК за допомогою смартфона	9	8	8	5	6	4,3
ФВ20	Можливість сканування ШК-позначок за допомогою спеціального сканера	5	4	4	9	8	0,4

1. 6. Висновки до першого розділу

Аналіз існуючих медичних інформаційних систем, а також технології автоматичної ідентифікації на основі штрихового кодування інформації дозволяє зробити наступні висновки.

1. Застосування штрихового кодування службової інформації, а також персональних та медичних даних пацієнтів у медичній інформаційній системі є фактором підвищення ефективності функціонування медичної установи та поліпшення обслуговування пацієнтів.
2. У медичних інформаційних системах доцільно використовувати багатоколірні штрихові коди, оскільки багатоколірність дозволяє у декілька разів підвищити інформаційну щільність подання даних порівняно з чорно-

білими ШК. Додаткове застосування в медичній інформаційній системі мобільних пристроїв, у яких багатоколірність підтримується технічно, але які мають відносно невелику площу екрана, багатоколірні штрихкодові зображення дозволяють подавати на обмеженій площі носія (екрана) достатньо великі обсяги медичної інформації.

3. Багатоколірне штрихове кодування даних у складі медичної інформаційної системи, крім свого прямого призначення – забезпечення процесів автоматичної ідентифікації, доцільно також використовувати для створення додаткових функціональних можливостей системи – посилення захисту персональних та медичних даних пацієнтів, електронного доступу пацієнта до власних медичних даних, а також електронної візуально-захищеної взаємодії лікаря і пацієнта.
4. У зв’язку з тим, що при застосуванні багатоколірних штрихових кодів дещо ускладнюються процеси декодування штрихкодових позначок (через багатоколірність), а також необхідністю зчитування багатоколірного штрихкодового зображення з екрана одного смартфона (мобільного пристрою) камерою іншого смартфона необхідно вживати додаткових заходів алгоритмічно-програмного характеру щодо забезпечення завадостійкості багатоколірних штрихкодових зображень.
5. Оскільки, створенню якісного програмного продукту має передувати процедура розроблення вимог до програмного забезпечення, результатом якої є специфікація вимог, сформовано 20 основних функціональних вимог до програмної системи для автоматичної ідентифікації виробничих процесів у медичній інформаційній системі.
6. Запропоновано визначати пріоритетність вимоги на основі п’яти числових критеріїв (“вигода”, “втрата”, “вартість”, “ризик”, “близькість вигоди”), кожному з яких присвоюють числові значення з діапазону 1…10 шляхом їх експертного оцінювання, а потім на цій основі кількісно визначати загальний пріоритет вимоги за допомогою формульного обчислення. Це дозволяє упорядковувати вимоги за спаданням пріоритету та на цій підставі

ефективно планувати та розподіляти роботи між виконавцями з розроблення програмного забезпечення.

7. Для підвищення ефективності організації надання медичних послуг пацієнтам на основі медичної інформаційної системи із застосуванням багатоколірного штрихового кодування інформації доцільно розробити архітектуру програмної системи, яка б надавала користувачам додаткові можливості щодо посилення захисту персональних та медичних даних пацієнта, автоматизованого доступу пацієнта до своїх медичних даних та електронної конфіденційної взаємодії пацієнта і лікаря з використанням мобільних пристроїв.

РОЗДІЛ 2. АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЗАВАДОСТІЙКОСТІ БАГАТОКОЛІРНИХ ШТРИХКОДОВИХ ЗНАКІВ

2. 1. Задача забезпечення завадостійкості багатоколірних штрихкодових знаків

Для надійного зчитування з об'єктів обліку штрихкодових даних необхідно забезпечувати завадостійкість штрихкодових зображень [53].

Структурно ШК-позначка штрихового коду складається з масиву ШК-знаків, упорядкованих у вигляді прямокутника або квадрата (рис. 2. 1). Для забезпечення цілісності даних багатоколірний ШК слід створювати так, щоб властивість завадостійкості забезпечувалась на двох рівнях – на рівні усієї ШК-позначки, а також на рівні ШК-знаків – мінімальних структурних одиниць зображення.

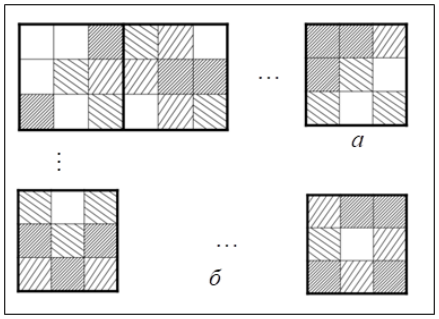


Рис. 2. 1. Структура багатоколірної штрихкодової позначки:
а – ШК-знак; *б* – ШК-позначка

Будь-який ШК має свій набір ШК-знаків, кожен з яких позначає символ (або комбінацію символів) комп'ютерного алфавіту. Цей набір ШК-знаків називають символікою штрихового коду [30 - 33].

Нехай необхідно створити символіку Ω потужності W завадостійкого багатоколірного матричного ШК з параметрами q, s; де q – кількість

використовуваних кольорів для забарвлення елементів ШК-знака, коли $q>2$; s – кількість елементів (чарунок) у ШК-знаку (див. рис. 2.1).

ШК-знаки можуть мати різну форму (рис. 2. 2).

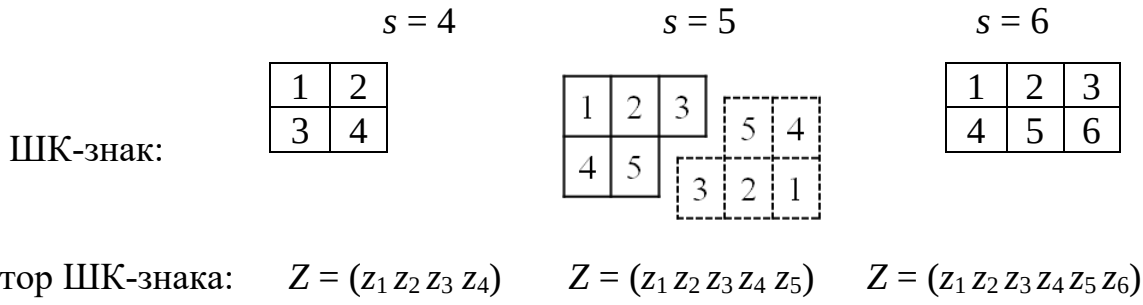


Рис. 2. 2. Приклади структур ШК-знаків при s = 4, 5, 6

Кольори, якими можуть бути забарвлені елементи ШК-знаків, позначатимемо цифрами: 0, 1, 2, …, q-1. ШК-знаку, що складається з s елементів, поставимо у відповідність цифровий еквівалент – вектор ШК-знака $Z=(z_0z_1\ldots z_{s-1})$, де $z_i \in \{0, 1, 2, \ldots, q-1\}$.

Для того, щоб ШК-знак був завадостійким, його вектор Z має бути кодовим словом многозначного ($q>2$) коректувального коду, здатного виправляти ушкодження, спотворення (які кваліфікуються як помилки). Вважатимемо, що під час зчитування з носія ШК-знаків найбільш ймовірними є випадки ушкодження, спотворення одного або двох елементів ШК-знака.

Якщо у межах ШК-знака необхідно забезпечувати виправлення одного спотворення, то слід використовувати многозначний (q-ковий) код Хемінга з мінімальною кодовою відстанню $d_{min}=3$. Для виправлення двократних спотворень у ШК-знаках слід використовувати многозначний код БЧХ (код Боуза-Чоудхурі-Хоквінгема) з мінімальною кодовою відстанню $d_{min}=5$.

У випадку застосування многозначних коректувальних кодів для реалізації процесів кодування-декодування даних використовують математичний апарат скінченних полів (полів Галуа) [40 - 42].

Розрізняють два види скінченних полів: поля виду $GF(p)$ – коли кількість (p) елементів поля є простим числом, тоді операції виконують за модулем p; поля виду $GF(p^m)$ – коли кількість (p^m) елементів поля є степенем простого числа, тоді операції виконують за модулем незвідного многочлена степеня m [42].

Задачу забезпечення завадостійкості багатоколірних ШК-знаків на основі коректувальних кодів слід вирішувати із застосуванням обох видів скінченних полів.

2. 2. Алгоритмічне забезпечення завадостійкості багатоколірних штрихкодових знаків на основі многозначного коду Хемінга

Побудуємо множину ШК-знаків (символіку), завадостійкість яких грунтується на застосуванні многозначного (q-кового) коду Хемінга.

Вважатимемо, що ШК-знак, до складу якого входять s елементів, складається з u інформаційних елементів (розрядів), які утворюють інформаційне слово $B = (b_1\ b_2\ \dots\ b_u)$, та v контрольних елементів (розрядів), які утворюють контрольне слово $T = (t_1\ t_2\ \dots\ t_v)$.

Побудову символіки завадостійкого штрихового коду виконаємо на основі перевірної матриці H коду Хемінга. У загальному випадку перевірна матриця q-кового (s, u)-коду Хемінга з $d_{min} = 3$ має вигляд

$$\mathbf{H}_{(s,u)}^{"q"} = \begin{Vmatrix} h_{11} & h_{12} & \cdots & h_{1u} & 1 & 0 & \cdots & 0 \\ h_{21} & h_{22} & \cdots & h_{2u} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{v1} & h_{v2} & \cdots & h_{vu} & 0 & 0 & \cdots & 1 \end{Vmatrix} = \left\| \mathbf{H}_{v\times u} \;\vdots\; \mathbf{I}_v \right\|, \qquad (2.\ 1)$$

де $h_{ij} \in \{0, 1, 2, \dots, q-1\}$. Стовпці матриці $\mathrm{H}_{(s,\,u)}$ є ненульовими v-розрядними послідовностями, у яких перша ненульова компонента дорівнює одиниці. Це забезпечує попарну лінійну незалежність стовпців [41]. Максимальна кількість стовпців становить $(q^v - 1) / (q - 1)$.

Якщо u – кількість інформаційних елементів у складі ШК-знака, а v – кількість контрольних елементів, то u і v пов'язані між собою співвідношенням

$$u \le (q^v - 1) / (q - 1) - v. \qquad (2.\ 2)$$

Якщо у співвідношенні (2. 2) виконується рівність, то q-ковий (s, u)-код Хемінга називають повним [40 - 42].

Матриця $\mathbf{H}^{"q"}_{(s,u)}$ дозволяє згенерувати всі можливі кодові слова $Z = (z_1\, z_2\, \ldots\, z_s)$ q-кового (s, u)-коду Хемінга, кожне з яких отримують як

$$Z = B \vdots (B \cdot \mathbf{H}^T_{v \times u})$$

(де $\vdots$ є символом конкатенації (приєднання)), якщо виконати перебір всіх можливих значень вектора $B = (b_1\, b_2\, \ldots\, b_u)$ – від $(0\ 0\ \ldots\ 0)$ до $(q-1\ q-1\ \ldots\ q-1)$. Кількість можливих кодових слів Z дорівнює q^u. Величина $q^u = W$ є потужністю символіки шрихового коду. Кожному Z слід поставити у відповідність ШК-знак. До складу символіки входитимуть W різних ШК-знаків, елементи яких забарвлені q кольорами. При цьому кожен ШК-знак буде завадостійким – під час зчитування у межах ШК-знака можливе виправлення однократної помилки.

У табл. 2. 1 наведено деякі повні (s, u)-коди Хемінга, на основі яких можна синтезувати символіки завадостійких багатоколірних штрихових кодів.

Наприклад, для синтезу п'ятиколірних завадостійких ШК-знаків слід розглядати п'ятіркові коди Хемінга.

Розглянемо п'ятірковий повний (6, 4)-код Хемінга (табл. 2. 1), перевірна матриця якого має вигляд

$$\mathbf{H}^{"5"}_{(6,4)} = \left\| \begin{array}{cccccc} b_1 & b_2 & b_3 & b_4 & t_1 & t_2 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 4 & 0 & 1 \end{array} \right\|. \qquad (2.\ 3)$$

Таблиця 2.1

Деякі повні q-кові (s, u)- коди Хемінга

Кількість використовуваних кольорів				
q=3	q=4	q=5	q=7	q=8
Вид скінченного поля				
$GF(3)$	$GF(2^2)$	$GF(5)$	$GF(7)$	$GF(2^3)$
(4,2)- (13,10)-	(5,3)- (21,18)-	(6,4)- (31,28)-	(8,6)- (57,54)-	(9,7)- (73,70)-

При використанні п’ятіркових кодів Хемінга операції виконують у полі $GF(5)$, тобто за модулем 5 (рис. 2. 3).

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

–	0	1	2	3	4
0	0	4	3	2	1
1	1	0	4	3	2
2	2	1	0	4	3
3	3	2	1	0	4
4	4	3	2	1	0

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

:	0	1	2	3	4
0	–	0	0	0	0
1	–	1	3	2	4
2	–	2	1	4	3
3	–	3	4	1	2
4	–	4	2	3	1

Рис. 2. 3. Виконання операцій в полі $GF(5)$

На основі матриці (2. 3) можна побудувати символіку п’ятиколірного завадостійкого штрихового коду з можливістю корекції однократних помилок у ШК-знаках; потужність символіки становить $W = 5^4 = 625$. ШК-знаки складаються з 6-ти елементів (рис. 2. 4), кожен елемент може бути забарвлений одним з п’яти кольорів. ШК-знаку відповідає вектор $Z = (z_1\ z_2\ z_3\ z_4\ z_5\ z_6)$, у якому $(z_1\ z_2\ z_3\ z_4) := (b_1\ b_2\ b_3\ b_4)$, $(z_5\ z_6) := (t_1\ t_2)$.

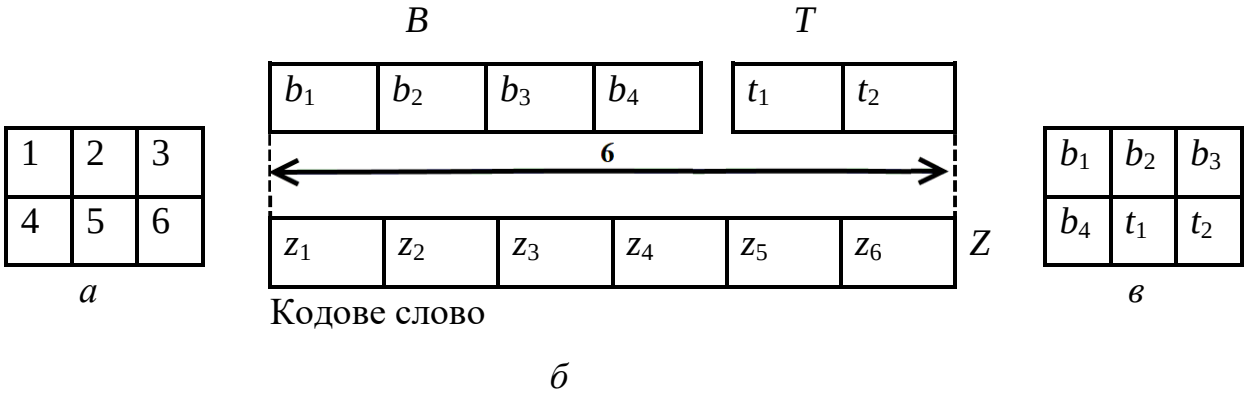


Рис. 2. 4. Формування ШК-знаків п’ятиколірного завадостійкого штрихового коду з можливістю виправлення однократних помилок у ШК-знаках: *а* – структура ШК-знака; *б* – вектор ШК-знака; *в* – порядок заповнення ШК-знака

Контрольні розряди t_1, t_2 обчислимо, виходячи з матриці (2. 3):

$$\begin{cases} t_1 = (b_1 + b_2 + b_3 + b_4) \bmod 5 = \left(\sum_{j=1}^{4} b_j \right) \bmod 5 \\ t_2 = (b_1 + 2b_2 + 3b_3 + 4b_4) \bmod 5 = \left(\sum_{j=1}^{4} j \cdot b_j \right) \bmod 5. \end{cases} \qquad (2.\ 4)$$

Нехай B = (2 0 4 1). Тоді

$$\begin{cases} t_1 = (2 + 0 + 4 + 1) \bmod 5 = 2 \\ t_2 = (2 + 2 \cdot 0 + 3 \cdot 4 + 4 \cdot 1) \bmod 5 = 3. \end{cases}$$

Отже, вектор ШК-знака дорівнює Z = (2 0 4 1 2 3), він є кодовим словом п’ятіркового (6, 4)-коду Хемінга, і йому відповідає п’ятиколірний завадостійкий ШК-знак (рис. 2. 5).

а *б* *в*

Рис. 2. 5. Приклад створення п’ятиколірного завадостійкого ШК-знака: *а* – порядок заповнення ШК-знака; *б* – вектор ШК-знака; *в* – забарвлення елементів ШК-знака

Беручи всі можливі значення вектора B – від 0 0 0 0 до 4 4 4 4, та застосувавши до кожного слова процедуру кодування (2. 4), отримаємо $5^4 = 625$ різних ШК-знаків, які утворюють символіку **Ω** завадостійкого п’ятиколірного штрихового коду, в якому в ШК-знаках можливе виправлення однократних помилок. Потужність W символіки **Ω** становить 625 ШК-знаків ($W = 625$); їй можна поставити у відповідність числову множину **Ω** = {0, 1, 2, …, 624} (табл. 2. 2).

Процес синтезу символіки 5-колірного завадостійкого штрихового коду можна описати таким псевдокодом:

for $B = 0000$ *to* 4444 *do*

$Z[1..4] := B$

$$t_1 = \left(\sum_{j=1}^{4} b_j \right) \bmod 5$$

$$t_2 = \left(\sum_{j=1}^{4} j \cdot b_j \right) \bmod 5$$

$Z[5] := t_1; \quad Z[6] := t_2$

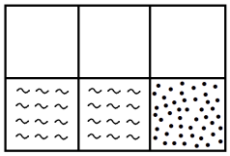
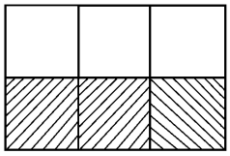
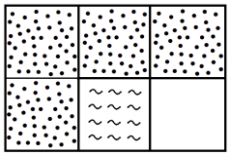
$Z[1..6] \Rightarrow$ barcode_pattern(B),

де *barcode_pattern* (B) – програмна процедура перетворення цифрового

вектора *Z* ШК-знака в п’ятиколірне зображення ШК-знака (див. рис. 2. 5*в*).

Таблиця 2. 2

Структура символіки завадостійкого штрихового коду
з параметрами $q = 5$, $s = 6$

Порядковий номер ШК-знака	ШК-знак	Вектор ШК-знака
0		0 0 0 0 0 0
1		0 0 0 1 1 4
2		0 0 0 2 2 3
. . .	. . .	. . .
624		4 4 4 4 1 0

Для створення штрихкодової позначки алфавітно-цифрову послідовність перетворюють у числову форму, використовуючи числа з діапазону 0 – 624; кожному числу ставлять у відповідність ШК-знак та наносять на носій.

Для синтезу триколірних завадостійких ШК-знаків слід використовувати трійкові коди Хемінга, наприклад, повний (13, 10)-код Хемінга (табл. 2. 1), перевірна матриця *H* якого має вигляд:

$$\mathbf{H}^{"3"}_{(13,10)} = \left\| \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 & 1 & 2 \end{matrix} \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \right\| = \left\| \mathbf{H}_{3\times 10} \vdots \ \mathbf{I}_3 \right\|. \quad (2.5)$$

Якщо необхідно синтезувати семиколірні завадостійкі ШК-знаки, то за основу можна взяти сімковий (8, 6)-код Хемінга (табл. 2. 1), який задає перевірна матриця **H** виду:

$$\mathbf{H}^{"7"}_{(8,6)} = \left\| \begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \right\| = \left\| \mathbf{H}_{2\times 6} \vdots \ \mathbf{I}_2 \right\|. \quad (2.6)$$

Алгоритм декодування багатоколірних завадостійких ШК-знаків

Після зчитування штрихкодового зображення послідовно виокремлюють ШК-знаки, кожному з яких ставлять у відповідність q-ковий s-розрядний вектор $Z' = (z'_1\, z'_2 \ldots z'_s)$, який декодують за правилами q-кового (s, u)-коду Хемінга.

Синдром помилки $S = (S_1\, S_2 \ldots S_v)$ обчислюють, використовуючи матрицю (2. 1):

$$\begin{cases} S_1 = \left(\sum_{j=1}^{u} h_{1j} z'_j - z'_{u+1} \right) \bmod q \\ \ldots \\ S_v = \left(\sum_{j=1}^{u} h_{vj} z'_j - z'_s \right) \bmod q\,. \end{cases} \quad (2.7)$$

Якщо всі компоненти S_i (i = 1, …, v) синдрому S дорівнюють нулю, то прийнятий вектор Z' не містить помилок ($Z' = Z$), а, отже, у зчитаному ШК-знаку відсутні спотворення (рис. 2. 6).

Якщо $S \neq 0$, то виконують наступні дії.

1. Послідовно аналізуючи компоненти $S_1\, S_2 \ldots S_v$ синдрому (зліва направо), знаходять першу ненульову компоненту. Її значення є величиною помилки; нехай значення першої ненульової компоненти дорівнює e (e = 1, 2, …., q – 1).

2. Компоненти $S_1S_2 \ldots S_v$ синдрому ділять на значення e за правилами поля $GF(q)$, отримуючи модифікований вектор $S^{м} = (S_1^{м}\ S_2^{м} \ldots S_v^{м})$ синдрому помилки, де $S_i^{м} = (S_i / e) \bmod q$. Модифікований синдром $S^{м}$ є локатором помилки.
3. Порівнюють обчислений локатор $S^{м}$ з кожним зі стовпців матриці $\mathbf{H}_{(s,u)}^{"q"}$ (див. (1)). Стовпець, який збігається з локатором $S^{м}$, вказує на місцезнаходження помилки.

   Нехай $S^{м}$ збігається з r-м (r = 1, 2, …, s) стовпцем матриці $\mathbf{H}_{(s,u)}^{"q"}$.
4. Виконують корекцію r-ї компоненти (z'_r) вектора Z':

$$z_r = \left(z'_r - e\right) \bmod q.$$

5. Якщо $S^{м}$ не збігається з жодним зі стовпців матриці $\mathbf{H}_{(s,u)}^{"q"}$, то приймається рішення про те, що прийнятий вектор Z' містить багатократну помилку, тобто у зчитаному ШК-знаку спотворено два або більше штрихкодових елементів.

Якщо обробляють ШК-позначку, наприклад, п’ятиколірного штрихового коду, символіку якого синтезовано на основі (6, 4)-коду Хемінга, то після виокремлення ШК-знаків кожному з них ставлять у відповідність 6-розрядний вектор ШК-знака $Z' = (z'_1\ z'_2\ z'_3\ z'_4\ z'_5\ z'_6)$.

Беручи до уваги систему (2. 7), синдром помилки S обчислимо, виходячи з перевірної матриці (2. 3):

$$\begin{cases} S_1 = \left(z'_1 + z'_2 + z'_3 + z'_4 - z'_5\right) \bmod 5 \\ S_2 = \left(z'_1 + 2z'_2 + 3z'_3 + 4z'_4 - z'_6\right) \bmod 5. \end{cases} \qquad (2.\ 8)$$

На основі синдрому $S = (S_1\ S_2)$ виконують корекцію вектора Z' за алгоритмом на рис. 2. 6.

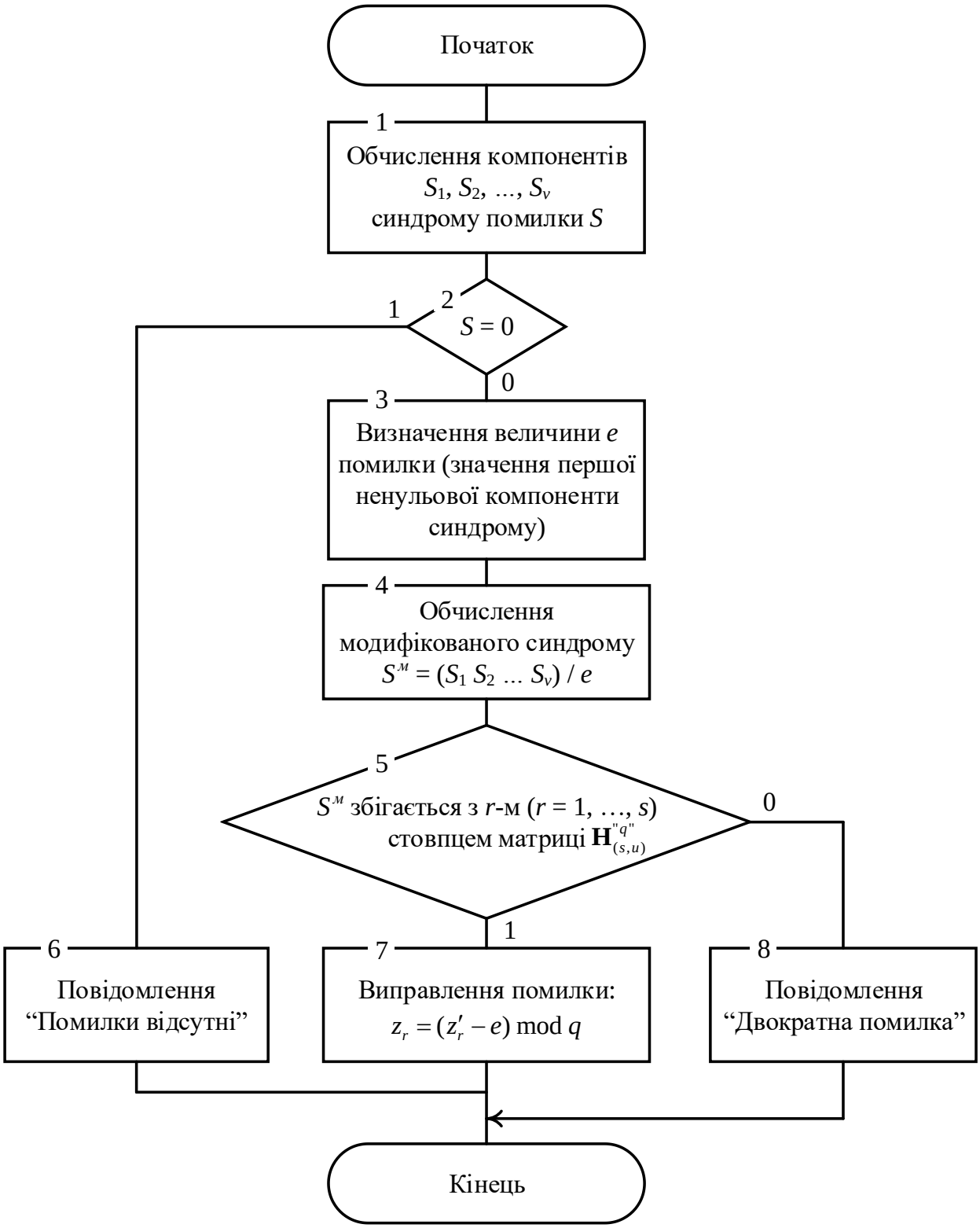


Рис. 2. 6. Блок-схема алгоритму декодування зчитуваних ШК-знаків штрихкодової позначки

Нехай вектор зчитаного ШК-знака дорівнює $Z' = (2\,0\,4\,\underline{3}\,2\,3)$, і він містить помилку в 4-му розряді (підкреслено). Обчислимо синдром помилки *S* вектора Z' на основі системи (2. 8):

$$\begin{cases} S_1 = (2+0+4+3-2) \bmod 5 = 2 \\ S_2 = (2+2\cdot 0+3\cdot 4+4\cdot 3-3) \bmod 5 = 3. \end{cases}$$

Оскільки першим ненульовим символом синдрому є 2, то величина помилки дорівнює 2 ($e = S_1 = 2$).

Компоненти 2, 3 синдрому поділимо на величину помилки (на 2), щоб визначити локатор помилки: (2 3) / e = (2/2 3/2) = (1 4). Оскільки локатор (1 4) збігається з 4-м стовпцем матриці $\mathbf{H}^{"5"}_{(6,4)}$ (див. (2. 3)), то це означає, що прийнятий вектор Z' містить помилку в розряді z'_4. Виправимо помилку в 4-му розряді: $z_4 = (z'_4 - e) \bmod 5 = (3-2) \bmod 5 = 1$. Правильним вектором зчитаного ШК-знака є Z = (2 0 4 1 2 3).

П’ятірковий (6, 4)-код Хемінга гарантовано забезпечує виправлення однократних помилок у ШК-знаках, але не виявляє двократні помилки, оскільки, він є повним кодом.

Загалом, будь-який повний код з мінімальною кодовою відстанню $d_{min} = 3$ не забезпечує виявлення двократної помилки.

У двійковому випадку код Хемінга з $d_{min} = 3$ можна перетворити в код з $d_{min} = 4$ шляхом додавання контрольного розряду, що контролює всі розряди слова, і тоді забезпечується, або виправлення однократної помилки, або виявлення двократної помилки у зчитаному слові. Застосування такого самого прийому у випадку многозначного ($q > 2$) коду Хемінга не приводить до отримання коду з $d_{min} = 4$. Многозначних кодів Хемінга з мінімальною кодовою відстанню $d_{min} = 4$ не існує. Але якщо застосовувати многозначні скорочені коди Хемінга, то можна виявляти частину двократних помилок, а також частину помилок більшої кратності.

Символіки багатоколірних завадостійких штрихових кодів на основі скорочених кодів Хемінга

Скорочений код Хемінга отримують шляхом викреслювання відповідної кількості стовпців підматриці $\mathbf{H}_{v\times u}$ перевірної матриці $\mathbf{H}^{"q"}_{(s,u)}$ повного коду Хемінга (див. (2. 1)).

Наприклад, на основі п'ятіркового повного (6, 4)-коду Хемінга, який задають перевірною матрицею $\mathbf{H}^{"5"}_{(6,4)}$ (див. (2. 3)), можна утворити два скорочені коди – (5, 3)- та (4, 2)-код (табл. 2. 3). Скорочений (5, 3)-код отримуємо, якщо викреслити стовпець b_4 у перевірній матриці $\mathbf{H}^{"5"}_{(6,4)}$, а скорочений (4, 2)-код – якщо викреслити два стовпці – b_4 та b_3.

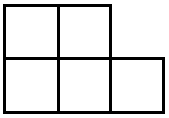
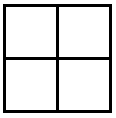
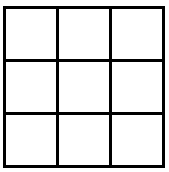
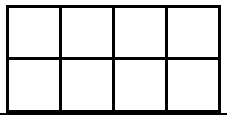
Скорочені п'ятіркові коди Хемінга можна отримати також, виходячи з повного (31, 28)-коду (див. табл. 2.1). Якщо в підматриці $\mathbf{H}_{3\times 28}$ перевірної матриці цього коду викреслити 23 стовпці, то отримаємо перевірну матрицю $\mathbf{H}^{"5"}_{(8,5)}$ скороченого (8, 5)-коду Хемінга, а якщо викреслити 22 стовпці – перевірну матрицю (9, 6)-коду. На основі $\mathbf{H}^{"5"}_{(8,5)}$ можна згенерувати символіку завадостійкого п'ятиколірного штрихового коду потужністю 3125 ШК-знаків, а на основі $\mathbf{H}^{"5"}_{(9,6)}$ – 15625 ШК-знаків (табл. 2.3).

Дослідимо здатність виявляти двократні помилки п'ятірковим неповним (5, 3)-кодом Хемінга, якому відповідає перевірна матриця

$$\mathbf{H}^{"5"}_{(5,3)} = \begin{array}{c} \begin{array}{ccccc} b_1 & b_2 & b_3 & t_1 & t_2 \end{array} \\ \left\| \begin{array}{ccccc} 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 0 & 1 \end{array} \right\| \end{array}. \qquad (2.\ 9)$$

Таблиця 2. 3

Параметри символік завадостійких штрихових кодів на основі скорочених п’ятіркових (*s*, *u*)-кодів Хемінга

(*s*, *u*)-код Хемінга	q^u	Потужність символіки ШК-знаків	Надлишковість ШК-знаків	Структура ШК-знаків	Кількість елементів у ШК-знаку
На основі повного (6, 4)-коду Хемінга					
(5, 3)-	5^3	125	2/5 = 0.40		5
(4, 2)-	5^2	25	2/4 = 0.50		4
На основі повного (31, 28)-коду Хемінга					
(9, 6)-	5^6	15625	3/9 ≈ 0.33		9
(8, 5)-	5^5	3125	3/8 ≈ 0.38		8

Нехай вектор ШК-знака, який було нанесено на носій, дорівнював Z = (4 2 0 1 3), а внаслідок ушкоджень штрихового зображення з носія зчитано ШК-знак, вектор якого дорівнює $Z' = (4\,\underline{3}\,\underline{3}\,1\,3)$; він містить дві помилки – у 2-му та 3-му розрядах. Обчислимо компоненти синдрому на основі (2. 7):

$$\begin{cases} S_1 = \left(4+3+3-1\right) \bmod 5 = 4 \\ S_2 = \left(4+2\cdot 3+3\cdot 3-3\right) \bmod 5 = 1. \end{cases}$$

Оскільки, $S \neq 0$, і першим ненульовим символом синдрому є 4, то величина помилки дорівнює 4 ($e = S_1 = 4$). Знайдемо модифікований $S^{м}$ синдром: $S^{м} = S / e = (4\ 1) / 4 = (1\ 4)$, який є локатором помилки. Оскільки локатор (1 4) не збігається з жодним зі стовпців матриці (9), то це означає, що прийнятий вектор Z' містить двократну помилку; помилку виявлено.

Розглянемо інший приклад.

Нехай зчитано ШК-знак, вектор якого дорівнює $Z' = (\underline{3}\,2\,\underline{2}\,1\,3)$; він містить дві помилки – у 1-му та 3-му рядках. Якщо обчислити синдром помилки, а потім модифікований синдром, то отримаємо $S^{м} = (1\ 0)$. Локатор (1 0) збігається з 4-м стовпцем перевірної матриці (2. 9), вказуючи на те, що помилковим є розряд z'_4 прийнятого вектора. Однак, це не відповідає дійсності, отже, двократну помилку (в розрядах z'_1, z'_3) не виявлено.

Таким чином, п’ятірковий неповний (5, 3)-код Хемінга гарантовано виправляє будь-яку однократну помилку в ШК-знаку, а двократні помилки виявляє лише частково.

Моделювання програмним способом всіх можливих випадків помилок у ШК-знаках, синтезованих на основі п’ятіркового (5, 3)-коду Хемінга, показує, що у ШК-знаках виявляються 25% двократних помилок (Додаток Г).

Так само можна дослідити завадостійкість багатоколірних ШК-знаків штрихових кодів, символіки яких синтезовано на основі деяких неповних 3-кових та 7-кових кодів Хемінга, зокрема, здатність виявляти двократні помилки як найбільш ймовірні після однократних, у ШК-знаках триколірних та семиколірних штрихових кодів (рис. Г. 2).

За аналогією можна синтезувати символіки багатоколірних ШК-знаків коли q - кількість використовуваних кольорів для забарвлення знаків, є степенем простого числа, тобто, коли $q = p^m$, де $p \geq 2$, $m \geq 2$.

У Додатку Д в 5-ти таблицях наведено показники виявлення багатократних помилок 27-ма многозначними (q = 3, 4, 5, 7, 8) скороченими кодами Хемінга. На основі кожної з цих таблиць можна побудувати сімейство

штрихових кодів (5 сімейств) відповідної колірності, символіки яких мають властивість завадостійкості – окрім виправлення однократної помилки в межах кожного ШК-знака, забезпечується виявлення значного відсотка помилок більшої кратності (2– 6 помилок): триколірними ШК-знаками (7,7 – 60,0)%, чотириколірними ШК-знаками (11,1 – 70,8)%, п’ятиколірними - (12,5 – 80,2)%, семиколірними - (11,1 – 66,7)%, восьмиколірними - (20,4 – 71,4) %.

Властивість виявляти багатократні помилки є визначальною на нижньому рівні (рівні ШК-знаків) забезпечення завадостійкості багатоколірних ШК, оскільки саме через цей параметр (виявлення багатократних помилок) здійснюється зв'язок з верхнім рівнем забезпечення завадостійкості (рівень усієї ШК-позначки) на основі коректувального коду Ріда-Соломона.

2. 3. Алгоритмічне забезпечення завадостійкості багатоколірних штрихкодових знаків на основі многозначного коду БЧХ

Побудуємо множину ШК-знаків (символіку), завадостійкість яких грунтується на застосуванні многозначного (q-кового) коду БЧХ (код Боуза-Чоудхурі-Хоквінгема) з мінімальною кодовою відстанню d_{min}=5.

Код БЧХ є циклічним коректувальним кодом, у якому кодування інформації зводиться до множення многочлена $B(x)=\sum_{i=0}^{u-1} b_i x^i$ степеня u-1, який відповідає q-ковому u-розрядному інформаційному слову $B=(b_0 b_1 \dots b_{u-1})$, на твірний (генераторний) многочлен $g(x)=\sum_{i=0}^{v} g_i x^i = g_0 + g_1 x + \dots + g_v x^v$ степеня v:

$$Z(x)=B(x)g(x),$$

де $Z(x)=\sum_{i=0}^{s-1} z_i x^i$ - многочлен степеня s-1, що відповідає s-розрядному q-кодовому кодовому слову $Z=(z_0 z_1 \cdots z_{s-1})$, який є цифровим еквівалентом ШК-знака; $s = u + v$; $b_i, g_i, z_i \in GF(q)$ [40 - 42].

Якщо відомий твірний многочлен $g(x)$, то твірну матрицю $G_{(s,u)}$ (s,u)-коду БЧХ подають у вигляді:

$$G_{(s,u)}=\begin{Vmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{u-1}g(x) \end{Vmatrix}=\begin{Vmatrix} g_0 g_1 \cdots & g_v & 0 & \cdots 0 \\ 0 & g_0 \cdots & g_{v-1} & g_v 0 \cdots 0 \\ \vdots & & & \\ 0 & 0 \ \cdots 0 & g_0 & g_1 \ \cdots g_0 \end{Vmatrix},$$

вона має розмірність $u \times s$.

Твірний многочлен $g(x)$ q-кового (s, u)-коду БЧХ з мінімальною кодовою відстанню d_{min}=5, здатного виправляти дві помилки, визначають як найменше спільне кратне мінімальних многочленів $M^{(1)}(x)$, $M^{(2)}(x)$, $M^{(3)}(x)$, $M^{(4)}(x)$ елементів α^1, α^2, α^3, α^4 поля $GF(q^m)$:

$g(x)$=НСК($M^{(1)}(x)$, $M^{(2)}(x)$, $M^{(3)}(x)$, $M^{(4)}(x)$),

$\alpha \in GF(q^m)$, m – степінь мінімальних многочленів [42]. $GF(q)$ називають полем символів, $GF(q^m)$ – полем локаторів. Код БЧХ називають повним, якщо $s = q^m$-1.

Деякі q-кові (q>2) повні коди БЧХ з d_{min}=5, що придатні для вирішуваної задачі, наведено в табл.2. 4.

Нехай q=3 (випадок триколірного штрихового коду).

Розглянемо, наприклад, трійковий (8, 3)-код БЧХ (див. табл. 2. 4).

Таблиця 2. 4

Деякі повні q-кові (s, u)-коди БЧХ

Кількість кольорів	q=3	q=4	q=5	q=7	q=8
Поле символів	$GF(3)$	$GF(2^2)$	$GF(5)$	$GF(7)$	$GF(2^3)$
(s, u)-код БЧХ	(8, 3)- (26, 17)-	(15, 9)-	(24, 16)-	(48, 40)-	(63, 55)-
Поле локаторів	$GF(3^2)$ $GF(3^3)$	$GF((2^2)^2)$	$GF(5^2)$	$GF(7^2)$	$GF((2^3)^2)$

Для побудови такого коду використовують два поля: $GF(3)$ – поле символів (рис. 2. 7), та $GF(3^2)$ – поле локаторів (табл. 2. 5), яке є розширенням над $GF(3)$. Побудову $GF(3^2)$ виконано на основі незвідного многочлена $p(x)=x^2+x+2$.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

–	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

:	0	1	2
0	–	0	0
1	–	1	2
2	–	2	1

Рис. 2. 7. Виконання операцій в полі $GF(3)$

Кожному елементу α^i поля $GF(3^2)$, $i \geq 1$, відповідає мінімальний многочлен $M^i(x)$, степінь якого не перевищу є два.

Таблиця 2. 5

Елементи поля $GF(3^2)$ та їх мінімальні многочлени

Степеневе подання		Поліноміальне подання (у вигляді многочлена від α)	Векторне подання	Числове подання (десяткове число)	Мінімальний многочлен елемента α^i
З невід'ємним показником степеня примітивного елемента α поля	З від'ємним показником степеня примітивного елемента α поля				
–	–	0	(0, 0)	0	–
α^0	α^{-8}	1	(0, 1)	1	–
α^1	α^{-7}	α	(1, 0)	3	x^2+x+2
α^2	α^{-6}	$2\alpha+1$	(2, 1)	7	x^2+1
α^3	α^{-5}	$2\alpha+2$	(2, 2)	8	x^2+x+2
α^4	α^{-4}	2	(0, 2)	2	$x+1$
α^5	α^{-3}	2α	(2, 0)	6	x^2+2x+2
α^6	α^{-2}	$\alpha+2$	(1, 2)	5	x^2+1
α^7	α^{-1}	$\alpha+1$	(1, 1)	4	x^2+2x+2

У полі $GF(3^2)$ $\alpha^i = x^{i-8}$, $\alpha^{-i} = \alpha^{8-i}$, $\alpha^8 = \alpha^{-8} = \alpha^0=1$.

Знайдемо твірний многочлен $g(x)$ трійкового (8, 3)-коду БЧХ:

$g(x)$=НСК($M^{(1)}(x)$, $M^{(2)}(x)$, $M^{(3)}(x)$, $M^{(4)}(x)$) =

НСК (x^2+x+2, x^2+1, x^2+x+2, $x+1$)=$(x^2+x+2)(x^2+1)(x+1)$=

$= x^5+2x^4+ x^3+x^2+2 \rightarrow g_0 g_1 \ldots g_5$=2 0 1 1 2 1.

Твірному многочлену $g(x)$ відповідає твірна матриця $G_{(8,3)}$:

$$G_{(8,3)} = \begin{array}{c} Z_0\ Z_1\ Z_2\ Z_3\ Z_4\ Z_5\ Z_6\ Z_7 \\ \left\| \begin{matrix} 2 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 & 2 & 1 \end{matrix} \right\| \end{array}.$$

На основі твірної матриці $G_{(}s,u_{)}$ коду БЧХ можна побудувати символіку завадостійкого штрихового коду з можливістю корекції одно- або двократних спотворень елементів (помилок) у ШК-знаках. Для цього u-розрядне

інформаційне q-кове слово $B=(b_0,\ b_1\ \dots\ b_{u-1})$ необхідно перетворити в s-розрядне слово $Z=(z_0,\ z_1\ \dots\ z_{s-1})$, яке є вектором (цифровим еквівалентом) ШК-знака, тобто слово B закодувати $(s,\ u)$-кодом БЧХ, а тоді вектору Z поставити у відповідність ШК-знак. Операцію кодування задають рівнянням $Z=B\cdot G_{(s,\ u)}$ та виконують за правилами поля $GF(q)$.

Наприклад, якщо $B=(b_0\ b_1\ b_2)$, де $b_i \in \{0, 1, 2\}$, то $Z=B\cdot G_{(8,\ 3)}$, тобто

$$\left\|b_0 b_1 b_2\right\| \cdot \begin{Vmatrix} 2 & 0 & 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 & 2 & 1 \end{Vmatrix} = \left\|z_0 z_1 \cdots\ z_7\right\|, \qquad (2.\ 10)$$

звідки випливає, що

$z_0=2b_0,\quad z_1=2b_1,\quad z_2=b_0+2b_2,\quad z_3=b_0+b_1,$

$z_4=2b_0+b_1+b_2,\quad z_5=b_0+2b_1+b_2,\quad z_6=b_1+2b_2,\ z_7=b_2$

(операції слід виконувати за модулем 3).

Нехай $B=(1\ 0\ 2)$, тоді $Z=(2\ 0\ 2\ 1\ 1\ 0\ 1\ 2)$, а відповідний ШК-знак показано на рис. 2. 8*г*.

Беручи всі можливі значення вектора B – від (0 0 0) до (2 2 2) та застосовуючи до кожного слова процедуру кодування (2. 10) отримаємо $3^3=27$ різних ШК-знаків, які утворюють символіку **Ω** завадостійкого триколірного штрихового коду, в якому в межах кожного ШК-знака можливе виправлення одно- або двократних помилок (табл. 2. 6).

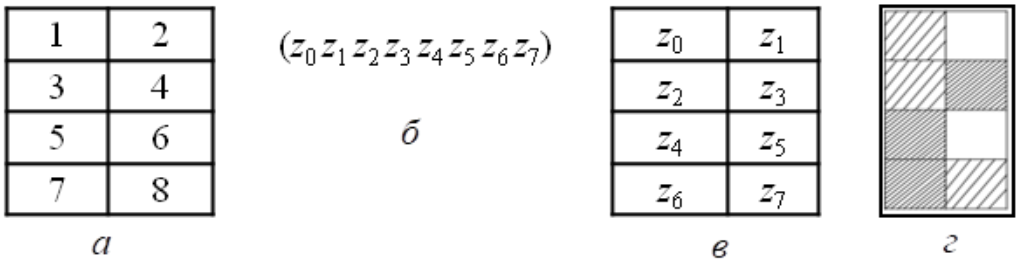


Рис. 2. 8. Створення триколірного ШК-знака з можливістю виправлення двократного спотворення елементів ШК-знака; *а* – структура ШК-знака; *б* – вектор ШК-знака; *в* – порядок заповнення ШК-знака; *г* – забарвлення ШК-знака

Потужність символіки **Ω** становить 27 ШК-знаків (W=27); їй можна поставити у відповідність числову множину **Ω**={0, 1, 2, …, 26}.

Процес синтезу символіки триколірного завадостійкого штрихового коду з можливістю корекції двократних помилок у ШК-знаках можна описати таким псевдокодом:

for B=000 to 222 do

$\|Z\|=\|B\|\cdot\|G_{(8,3)}\|$

$Z[1\ldots8]:=(z_0\, z_1 \ldots\, z_{7)}$

$Z[1\ldots8]\rightarrow$*barcode_pattern* (B).

Алфавітно-цифрову послідовність, яка підлягає поданню у штрихкодовому вигляді, перетворюють у числову форму, елементами якої є числа з діапазону 0 – 26 (з множини **Ω**), а потім кожному числу ставлять у відповідність ШК-знак (табл. 2. 6). Далі ШК-знаки наносять на носій, формуючи ШК-позначку.

Під час зчитування ШК-позначки послідовно виділяють ШК-знаки, кожному з яких ставлять у відповідність цифровий еквівалент – s-розрядний вектор $Z'=(z'_0 z'_1 \cdots z'_{s-1})$, який декодують за правилами (s, u)-коду БЧХ з d_{min}=5.

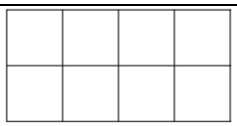
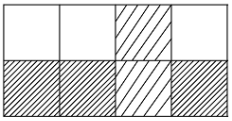
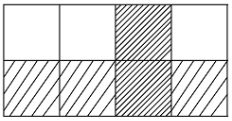
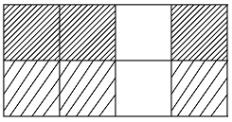
Декодування здійснюють на основі перевірної матриці q-кового коду БЧХ, яку подають у вигляді:

$$H_{(s,u)}=\begin{Vmatrix}\alpha_i\\ \alpha_i^2\\ \alpha_i^3\\ \alpha_i^4\end{Vmatrix},\ i=0, 1, 2, \ldots, s\text{-}1,$$

де α_i – елементи поля $GF(q^m)$.

Таблиця 2. 6

Символіка триколірного завадостійкого штрихового коду з можливістю виправлення одно- або двократних помилок у ШК-знаках на основі трійкового (8, 3)-коду БЧХ

Порядковий номер ШК-знака	ШК-знак	Вектор ШК-знака
0		00000000
1		00201121
2		00102212
⋮	⋮	⋮
26		11012202

Для трійкового (8, 3)-коду БЧХ вона має вигляд:

$$H_{(8,3)} = \begin{Vmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 \\ \alpha^0 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} & \alpha^{14} \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} & \alpha^{21} \\ \alpha^0 & \alpha^4 & \alpha^8 & \alpha^{12} & \alpha^{16} & \alpha^{20} & \alpha^{24} & \alpha^{28} \end{Vmatrix}$$

Узявши до уваги, $\alpha^0 = \alpha^8 = \alpha^{16} = \alpha^{24}$ та підставивши замість α^i відповідні дворозрядні вектор-стовпці (див. векторне подання елементів поля в табл. 2.5), отримаємо

$$H_{(8,3)} = \begin{array}{c} \begin{matrix} z'_0 & z'_1 & z'_2 & z'_3 & z'_4 & z'_5 & z'_6 & z'_7 \end{matrix} \\ \left\| \begin{matrix} 0 & 1 & 2 & 2 & 0 & 2 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 & 0 & 2 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 1 & 2 & 2 \\ 0 & 2 & 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 2 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \end{matrix} \right\| \end{array}.$$

Добуток матриці $G_{(8,3)}$ на транспоновану матрицю $H_{(8,3)}$ дорівнює нулю ($G_{(8,3)} \cdot H^{T}_{(8,3)}=0$).

Декодування прийнятого вектора Z' здійснюють за алгоритмом на рис. 2. 9.

Синдром помилки $S=S_1S_2S_3S_4$ обчислюють як $S=Z' \cdot H^{T}_{(8,3)}$, операції виконують за модулем 3:

$$S_1 = z'_0\begin{pmatrix}0\\1\end{pmatrix} + z'_1\begin{pmatrix}1\\0\end{pmatrix} + z'_2\begin{pmatrix}2\\1\end{pmatrix} + z'_3\begin{pmatrix}2\\2\end{pmatrix} + z'_4\begin{pmatrix}0\\2\end{pmatrix} + z'_5\begin{pmatrix}2\\0\end{pmatrix} + z'_6\begin{pmatrix}1\\2\end{pmatrix} + z'_7\begin{pmatrix}1\\1\end{pmatrix}$$

$$S_2 = z'_0\begin{pmatrix}0\\1\end{pmatrix} + z'_1\begin{pmatrix}2\\1\end{pmatrix} + z'_2\begin{pmatrix}0\\2\end{pmatrix} + z'_3\begin{pmatrix}1\\2\end{pmatrix} + z'_4\begin{pmatrix}0\\1\end{pmatrix} + z'_5\begin{pmatrix}2\\1\end{pmatrix} + z'_6\begin{pmatrix}0\\2\end{pmatrix} + z'_7\begin{pmatrix}1\\2\end{pmatrix}$$

$$S_3 = z'_0\begin{pmatrix}0\\1\end{pmatrix} + z'_1\begin{pmatrix}2\\2\end{pmatrix} + z'_2\begin{pmatrix}1\\2\end{pmatrix} + z'_3\begin{pmatrix}1\\0\end{pmatrix} + z'_4\begin{pmatrix}0\\2\end{pmatrix} + z'_5\begin{pmatrix}1\\1\end{pmatrix} + z'_6\begin{pmatrix}2\\1\end{pmatrix} + z'_7\begin{pmatrix}2\\0\end{pmatrix}$$

$$S_4 = z'_0\begin{pmatrix}0\\1\end{pmatrix} + z'_1\begin{pmatrix}0\\2\end{pmatrix} + z'_2\begin{pmatrix}0\\1\end{pmatrix} + z'_3\begin{pmatrix}0\\2\end{pmatrix} + z'_4\begin{pmatrix}0\\1\end{pmatrix} + z'_5\begin{pmatrix}0\\2\end{pmatrix} + z'_6\begin{pmatrix}0\\1\end{pmatrix} + z'_7\begin{pmatrix}0\\2\end{pmatrix}$$

Якщо S=0, то в слові Z' помилки відсутні, інакше (S≠0) – слово Z' містить одну або дві помилки (рис. 2. 9).

Далі формують матрицю $M = \begin{Vmatrix} S_1 & S_2 \\ S_2 & S_3 \end{Vmatrix}$ та обчислюють її детермінант *detM*. Якщо *detM*=0, то в слові Z' – одна помилка, інакше (*detM*≠0) – дві помилки.

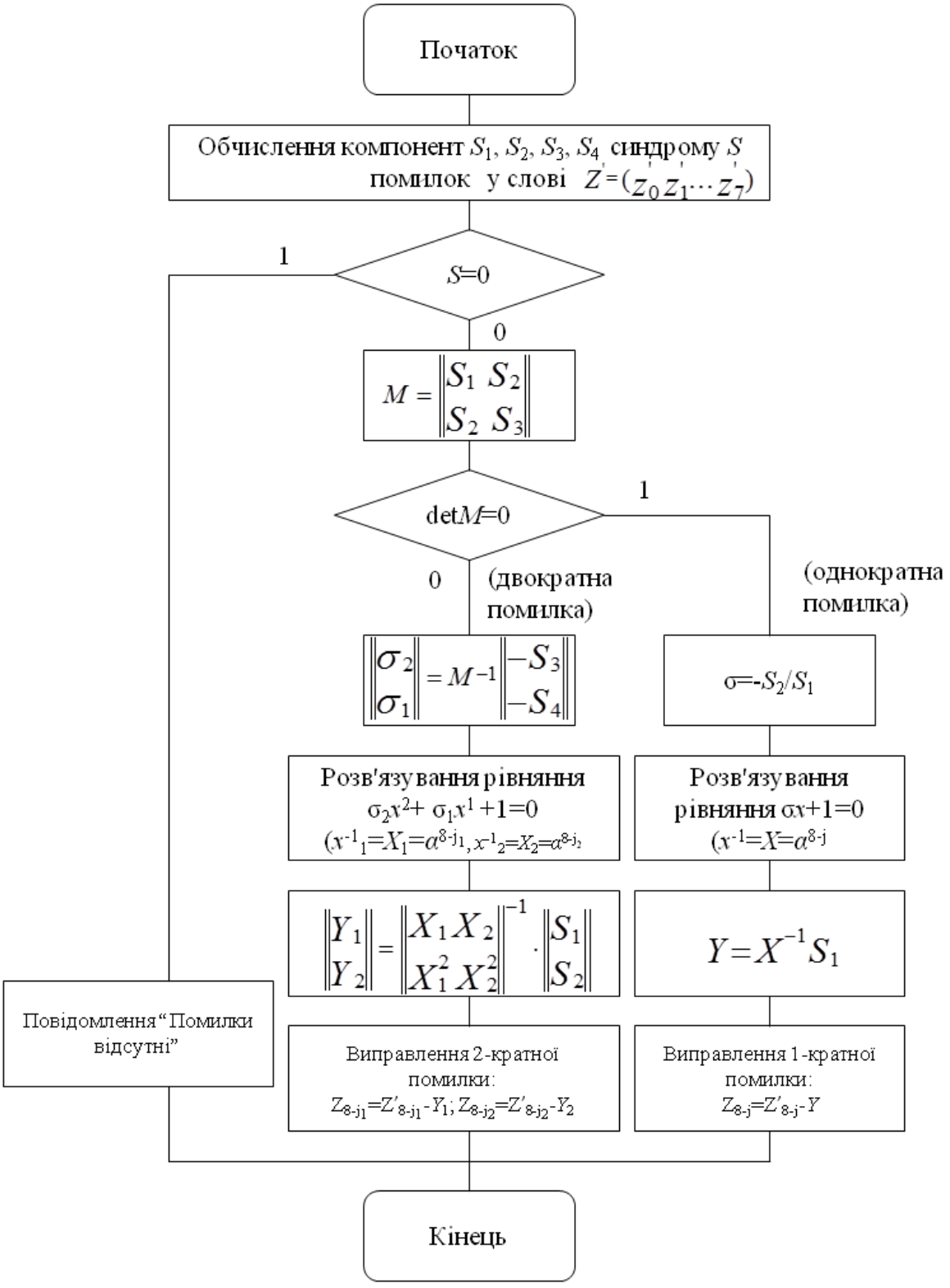


Рис. 2. 9. Блок-схема алгоритму корекції одно- або двократної помилки в слові $Z'=(z'_0 z'_1 \ldots z'_7)$

Розглянемо випадок коли $detM \neq 0$.

Знаходять корені x_1, x_2 многочлена локаторів помилок $\sigma(x)=\sigma_2 x^2+ \sigma_1 x+1$, де коефіцієнти σ_2, σ_1 визначають як

$$\begin{Vmatrix}\sigma_2\\ \sigma_1\end{Vmatrix} = M^{-1}\begin{Vmatrix}-S_3\\ -S_4\end{Vmatrix}.$$

Локаторами X_1, X_2 помилок є величини: $X_1= x_1^{-1}$, $X_2= x_2^{-1}$.

Рівняння $\sigma_2 x^2+ \sigma_1 x+1=0$ розв'язують у полі $GF(3^2)$ за процедурою Ченя [42], яка полягає в послідовному обчисленні $\sigma(\alpha^j)$ для $j=0,1,\ldots,7$, і перевірці отримуваних значень на нуль. Іншого способу розв'язування рівнянь у скінченних полях не існує.

Позиціям (розрядам) кодового слова Z відповідають степені примітивного елемента α поля:

$$\begin{array}{c} \alpha^{-8}\ \alpha^{-7}\ \alpha^{-6}\ \alpha^{-5}\ \alpha^{-4}\ \alpha^{-3}\ \alpha^{-2}\ \alpha^{-1} \\ \alpha^{0}\ \alpha^{1}\ \alpha^{2}\ \alpha^{3}\ \alpha^{4}\ \alpha^{5}\ \alpha^{6}\ \alpha^{7} \\ Z=(\ z_0\ \ z_1\ \ z_2\ \ z_3\ \ z_4\ \ z_5\ \ z_6\ \ z_7). \end{array}$$

Тому, якщо $\sigma(\alpha^j)=0$, то локатор X помилки дорівнює $\alpha^{-j}= \alpha^{8-j}$, і місцезнаходженням помилки є розряд з номером $8-j$.

Рівняння $\sigma_2 x^2+ \sigma_1 x+1=0$ має два корені: x_1, x_2 такі, що $x_1^{-1} = X_1 = a^{-j_1} = a^{8-j_1}$, $x_2^{-1} = X_2 = a^{-j_2} = a^{8-j_2}$. Якщо $X_1 = a^{-j_1}$, $X_2 = a^{-j_2}$, то помилки знаходяться в розрядах z'_{8-j_1}, z'_{8-j_2} прийнятого слова Z' відповідно.

Далі обчислюють величини Y_1, Y_2 помилок

$$\begin{Vmatrix}Y_1\\ Y_2\end{Vmatrix} = \begin{Vmatrix}X_1 & X_2\\ X_1^2 & X_2^2\end{Vmatrix}^{-1} \cdot \begin{Vmatrix}S_1\\ S_2\end{Vmatrix} \qquad (2.\ 11)$$

та виконують корекцію помилкових розрядів у слові Z':

$$z_{8-j_1} = (z'_{8-j_1} - Y_1) \bmod 3,\ z_{8-j_2} = (z'_{8-j_2} - Y_2) \bmod 3.$$

Якщо *detM*=0 (у прийнятому слові *Z'* – одна помилка), то знаходять корінь *x* многочлена σ*x*+1, де σ= -S_2/S_1. Рівняння σ*x*+1=0 також розв'язують за процедурою Ченя. Розв'язком є $x=\alpha^j$ такий, що x^{-1} є локатором помилки ($X= x^{-1}=\alpha^{-j}= \alpha^{8-j}$, а місцезнаходженням помилки є розряд з номером 8-*j* (тобто z'_{8-j}).

Розглянемо приклад корекції двократної помилки у зчитаному ШК-знаку.

Вважатимемо, що на носій було нанесено ШК-знак, вектор якого дорівнював *Z*=(2 0 2 1 1 0 1 2).

Нехай під час зчитування цього знака отримано вектор

$$Z'=(\underline{1}\ \ 0\ \ 2\ \ \underline{2}\ \ 1\ \ 0\ \ 1\ \ 2),$$
$$Z'_0\ Z'_1\ Z'_2\ Z'_3\ Z'_4\ Z'_5\ Z'_6\ Z'_7$$

який містить дві помилки (підкреслені розряди).

Обчислимо компоненти S_1, S_2, S_3, S_4 синдрому *S*:

$$S_1=1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}1\\0\end{pmatrix}+2\begin{pmatrix}2\\1\end{pmatrix}+2\begin{pmatrix}2\\2\end{pmatrix}+1\begin{pmatrix}0\\2\end{pmatrix}+0\begin{pmatrix}2\\0\end{pmatrix}+1\begin{pmatrix}1\\2\end{pmatrix}+2\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}2\\1\end{pmatrix}=\alpha^2$$

$$S_2=1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}2\\1\end{pmatrix}+2\begin{pmatrix}0\\2\end{pmatrix}+2\begin{pmatrix}1\\2\end{pmatrix}+1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}2\\1\end{pmatrix}+1\begin{pmatrix}0\\2\end{pmatrix}+2\begin{pmatrix}1\\2\end{pmatrix}=\begin{pmatrix}1\\1\end{pmatrix}=\alpha^7$$

$$S_3=1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}2\\2\end{pmatrix}+2\begin{pmatrix}1\\2\end{pmatrix}+2\begin{pmatrix}1\\0\end{pmatrix}+1\begin{pmatrix}0\\2\end{pmatrix}+0\begin{pmatrix}1\\1\end{pmatrix}+1\begin{pmatrix}2\\1\end{pmatrix}+2\begin{pmatrix}2\\0\end{pmatrix}=\begin{pmatrix}1\\2\end{pmatrix}=\alpha^6$$

$$S_4=1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}0\\2\end{pmatrix}+2\begin{pmatrix}0\\1\end{pmatrix}+2\begin{pmatrix}0\\2\end{pmatrix}+1\begin{pmatrix}0\\1\end{pmatrix}+0\begin{pmatrix}0\\2\end{pmatrix}+1\begin{pmatrix}0\\1\end{pmatrix}+2\begin{pmatrix}0\\2\end{pmatrix}=\begin{pmatrix}0\\1\end{pmatrix}=\alpha^0$$

Складемо матрицю $M=\begin{Vmatrix}S_1\ S_2\\S_2\ S_3\end{Vmatrix}=\begin{Vmatrix}\alpha^2\ \alpha^7\\\alpha^7\ \alpha^6\end{Vmatrix}$.

Обчислимо визначник матриці *M*:

detM=$\alpha^2\alpha^6$- $\alpha^7\alpha^7$= α^8-α^{14}= α^0-α^6=(0,1)-(1,2)=(0,1)+(2,1)= =(2,2)= α^3 ≠0 (див. табл. 2. 5).

Оскільки *detM*≠0, то в слові *Z'* дві помилки.

Знайдемо коефіцієнти σ_2, σ_1 многочлена локаторів помилок

σ(*x*)= σ_2x^2+ σ_1x +1:

$$\left\|\begin{matrix}\sigma_2\\ \sigma_1\end{matrix}\right\| = M^{-1}\left\|\begin{matrix}-S_3\\ -S_4\end{matrix}\right\|$$

Для цього спочатку обчислимо M^{-1}:

$$M^{-1} = (1/\det M)\cdot\left\|\begin{matrix}S_3 & -S_2\\ -S_2 & S_1\end{matrix}\right\| = (1/\alpha^3)\cdot\left\|\begin{matrix}\alpha^6 & -\alpha^7\\ -\alpha^7 & \alpha^2\end{matrix}\right\| = \left\|\begin{matrix}\alpha^3 & -\alpha^4\\ -\alpha^4 & \alpha^{-1}\end{matrix}\right\|$$

Оскільки $-\alpha^4=-(0, 2)=(0, 1)=\alpha^0$, а $\alpha^{-1}=\alpha^7$, то

$$M^{-1} = \left\|\begin{matrix}\alpha^3 & \alpha^0\\ \alpha^0 & \alpha^7\end{matrix}\right\|.$$

Тоді, $\left\|\begin{matrix}\sigma_2\\ \sigma_1\end{matrix}\right\| = \left\|\begin{matrix}\alpha^3 & \alpha^0\\ \alpha^0 & \alpha^7\end{matrix}\right\|\cdot\left\|\begin{matrix}-\alpha^6\\ -\alpha^0\end{matrix}\right\| = \left\|\begin{matrix}\alpha^3 & \alpha^0\\ \alpha^0 & \alpha^7\end{matrix}\right\|\cdot\left\|\begin{matrix}\alpha^2\\ \alpha^4\end{matrix}\right\| = \left\|\begin{matrix}\alpha^3\\ \alpha^1\end{matrix}\right\|$

(оскільки $-\alpha^6=-(1, 2)=(2, 1)=\alpha^2$, а $-\alpha^0=-(0, 1)=(0, 2)=\alpha^4$).

Далі розв’яжемо рівняння $\alpha^3x^2+\alpha^1x+1=0$ в полі $GF(3^2)$.

Для цього застосуємо процедуру Ченя:

$x=\alpha^0\rightarrow\alpha^3(\alpha^0)^2+\alpha^1(\alpha^0)+1=\alpha^3+\alpha^1+1=(2,2)+(1,0)+1=(0,0)=0,$

$x=\alpha^1\rightarrow\alpha^3(\alpha^1)^2+\alpha^1(\alpha^1)+1=\alpha^5+\alpha^2+1=(2,0)+(2,1)+1=(1,2)\neq0,$

$x=\alpha^2\rightarrow\alpha^3(\alpha^2)^2+\alpha^1(\alpha^2)+1=\alpha^7+\alpha^3+1=(1,1)+(2,2)+1=(0,1)\neq0,$

$x=\alpha^3\rightarrow\alpha^3(\alpha^3)^2+\alpha^1(\alpha^3)+1=\alpha^1+\alpha^4+1=(1,0)+(0,2)+1=(1,0)\neq0,$

$x=\alpha^4\rightarrow\alpha^3(\alpha^4)^2+\alpha^1(\alpha^4)+1=\alpha^3+\alpha^5+1=(2,2)+(2,0)+1=(1,0)\neq0,$

$x=\alpha^5\rightarrow\alpha^3(\alpha^5)^2+\alpha^1(\alpha^5)+1=\alpha^5+\alpha^6+1=(2,0)+(1,2)+1=(0,0)=0,$

$x=\alpha^6\rightarrow\alpha^3(\alpha^6)^2+\alpha^1(\alpha^6)+1=\alpha^7+\alpha^7+1=(1,1)+(1,1)+1=(2,0)\neq0,$

$x=\alpha^7\rightarrow\alpha^3(\alpha^7)^2+\alpha^1(\alpha^7)+1=\alpha^1+\alpha^0+1=(1,0)+(0,1)+1=(1,2)\neq0.$

Як бачимо, коренями рівняння є $x_1=\alpha^0$ та $x_2=\alpha^5$. Отже,

$X_1 = x_1^{-1} = \alpha^0 = \alpha^{8-0} = \alpha^8 = \alpha^0$, а $X_2 = x_2^{-1} = \alpha^{-5} = \alpha^{8-5} = \alpha^3$. Це означає, що помилки мають місця в розрядах z'_0 та z'_3 прийнятого слова.

Обчислимо величини помилок на основі (2. 11):

$$\begin{Vmatrix} Y_1 \\ Y_2 \end{Vmatrix} = \begin{Vmatrix} \alpha^0 & \alpha^3 \\ \alpha^0 & \alpha^6 \end{Vmatrix}^{-1} \cdot \begin{Vmatrix} \alpha^2 \\ \alpha^7 \end{Vmatrix}.$$

Спочатку знайдемо обернену матрицю:

$$\begin{Vmatrix} \alpha^0 & \alpha^3 \\ \alpha^0 & \alpha^6 \end{Vmatrix}^{-1} = (1/\alpha^5) \cdot \begin{Vmatrix} \alpha^6 & -\alpha^3 \\ -\alpha^0 & \alpha^0 \end{Vmatrix} = (1/\alpha^5) \cdot \begin{Vmatrix} \alpha^6 & \alpha^7 \\ \alpha^4 & \alpha^0 \end{Vmatrix} =$$

$$= \begin{Vmatrix} \alpha^1 & \alpha^2 \\ \alpha^{-1} & \alpha^{-5} \end{Vmatrix} = \begin{Vmatrix} \alpha^1 & \alpha^2 \\ \alpha^7 & \alpha^3 \end{Vmatrix}.$$

Тоді $\begin{Vmatrix} Y_1 \\ Y_2 \end{Vmatrix} = \begin{Vmatrix} \alpha^1 & \alpha^2 \\ \alpha^7 & \alpha^3 \end{Vmatrix} \cdot \begin{Vmatrix} \alpha^2 \\ \alpha^7 \end{Vmatrix} = \begin{Vmatrix} \alpha^4 \\ \alpha^0 \end{Vmatrix}$.

Але $\alpha^4=(0, 2)=2$, $\alpha^0=(0, 1)=1$.

Тому $Y_1=2$, $Y_2=1$.

Виправимо помилки:

$$z_0=(z'_0-Y_1)\bmod 3=(1-2)\bmod 3=2$$

$$z_3=(z'_3-Y_2)\bmod 3=(2-1)\bmod 3=1.$$

Таким чином, правильним вектором зчитаного ШК-знака є Z=(2 0 2 1 1 0 1 2); двократну помилку виправлено.

Для побудови символік багатоколірних ШК різної потужності слід використовувати скорочені (неповні) q-кові коди БЧХ. Скороченим називають код, утворений шляхом викреслювання потрібної кількості стовпців і рядків у твірній та перевірній матрицях повного коду.

Побудову скорочених кодів розглянемо на прикладі повного трійкового (26, 17)-коду БЧХ (див. табл. 2. 4), який використовує два поля: $GF(3)$ – поле символів (рис. 2. 7) та $GF(3^3)$ – поле локаторів (табл. 2. 7). Поле локаторів побудовано на основі незвідного многочлена степеня три $p(x)=x^3+2x+1$.

У полі $GF(3^3)$ $\alpha^i=\alpha^{i-26}$, $\alpha^{-i}=\alpha^{26-i}$, $\alpha^{26}=\alpha^{-26}=\alpha^0=1$.

Твірний многочлен $g(x)$ трійкового (26, 17)-коду БЧХ визначають так:

$g(x)$=НСК(x^3+2x+1, x^3+x^2+x+2, x^3+2x+1, x^3+x^2+2)=(x^3+2x+1)(x^3+x^2+x+2)(x^3+x^2+2)=$x^9+2x^8+x^7+x^6+x^5+2x^4+2x^3+2x^2+x+1\rightarrow g_0\, g_1 \ldots g_9$= 1 1 2 2 2 1 1 1 2 1.

Йому відповідає твірна матриця $G_{(26,17)}$ повного коду.

Якщо в $G_{(26,17)}$ викреслити, наприклад, 10 стовпців справа (z_{16} - z_{25}) та 10 нижніх рядків, а в перевірній матриці $H_{(26,17)}$ – стовпці $z'_{16} - z'_{25}$, то отримаємо твірну ($G_{(16,7)}$) та, відповідно, перевірну ($H_{(16,7)}$) матриці (рис. 2. 10)

Викреслюючи в $G_{(26,17)}$ послідовно стовпці та нижні рядки, а в $H_{(26,17)}$ – відповідні праві стовпці, отримаємо низку скорочених трійкових (q=3) кодів БЧХ: (17, 8)-; (16, 7)-; (15, 6)-; (14, 5)-; (13, 4)-, на основі яких можна синтезувати символіки завадостійких триколірних ШК різної потужності (табл. 2. 8).

За аналогією на основі скорочених п’ятіркових (q=5) та сімкових (q=7) кодів БЧХ можна синтезувати символіки завадостійких п’яти- та семи-колірних ШК з можливістю виправлення двократних помилок у ШК-знаках. Так, зазначені в табл. 2. 8 скорочені п’ятіркові коди, утворено на основі повного п’ятіркового (24, 16)-коду БЧХ (див. табл. 2. 4), який використовує поле символів $GF(5)$ та поле локаторів $GF(5^5)$, а скорочені сімкові коди – на основі повного сімкового (48, 40)-коду БЧХ, який використовує поле символів $GF(7)$ та поле локаторів $GF(7^2)$.

Така низка кодів БЧХ дає можливість будувати сімейства багатоколірних завадостійких ШК з символіками різної потужності.

Розглянуті коди БЧХ з мінімальною кодовою відстанню d_{min}=5 забезпечують виправлення одно- та/або двократних помилок у межах кожного зчитуваного з носія ШК-знака багатоколірного ШК. Для перевірки коректувальних можливостей кодів БЧХ, зокрема й здатності додатково виявляти багатократні помилки в ШК-знаках, розроблено програмний продукт на мові програмування Java у середовищі розробки Intelij idea.

Таблиця 2. 7

Елементи поля $GF(3^3)$ та їх мінімальні многочлени (фрагмент)

Степеневе подання		Поліноміальне подання (у вигляді многочлена від α)	Векторне подання	Числове подання (десяткове число)	Мінімальний многочлен елемента α^i
З невід'ємним показником степеня примітивного елемента α поля	З від'ємним показником степеня примітивного елемента α поля				
–	–	0	(0,0,0)	0	–
α^0	α^{-26}	1	(0,0,1)	1	–
α^1	α^{-25}	α	(0,1,0)	3	x^3+2x+1
α^2	α^{-24}	α^2	(1,0,0)	9	x^3+x^2+x+2
α^3	α^{-23}	$\alpha+2$	(0,1,2)	5	x^3+2x+1
α^4	α^{-22}	$\alpha^2+2\alpha$	(1,2,0)	15	x^3+x^2+2
α^5	α^{-21}	$2\alpha^2+\alpha+2$	(2,1,2)	23	x^3+x^2+x+1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
α^{25}	α^{-1}	$2\alpha^2+1$	(2,0,1)	19	x^3+2x^2+1

Таблиця 2. 8

Потужності (W) символік багатоколірних (q) завадостійких ШК на основі скорочених (s,u)-кодів БЧХ

q=3		q=5		q=7	
(s,u)-	W	(s,u)-	W	(s,u)-	W
(13,4)-	81	(10,2)-	25	(10,2)-	49
(14,5)-	243	(11,3)-	125	(11,3)-	343
(15,6)-	729	(12,4)-	625	(12,4)-	2401
(16,7)-	2187	(13,5)-	3125	(13,5)-	16807
(17,8)-	6561	(14,6)-	15625		

Експериментальні дослідження проводились на комп'ютері з операційною системою macOs BigSur, об'ємом оперативної пам'яті 32 GB, процесором 2.4 GHz 8-Core Intel Core i9.

Даний програмний продукт дозволяє проводити статистичні дослідження коректувальної здатності кодів БЧХ в умовах виникнення багатократних ушкоджень елементів ШК-знаків. Досліджувались всі можливі випадки виникнення від однієї до семи помилок у ШК-знаках. Для кожного випадку фіксувалась одна з трьох можливих подій: помилка виявляється (для одно- та двократних помилок виявлення еквівалентне виправленню – за алгоритмом на рис. 2. 9); синдром
помилки дорівнює нулю; помилка не виявляється. Генерувались не лише всі можливі місцезнаходження в словах ймовірних помилок, але також й всі можливі величини помилок.

Отримано статистичні дані, що характеризують здатність многозначних кодів БЧХ виявляти та виправляти багатократні помилки. Доведено, що всі одно- та двократні помилки в словах даних (векторах ШК-знаків) виправляються.

Дослідження показують, що повні коди БЧХ, наприклад, трійковий (8, 3)-код або п'ятірковий (24, 16)-код, забезпечують виявлення меншої кількості багатократних помилок порівняно зі скороченими кодами. Це пояснюється тим, що скорочені коди мають більшу надлишковість.

За аналогією можна синтезувати символіки багатоколірних ШК-знаків коли q - кількість використовуваних кольорів для забарвлення знаків, є степенем простого числа, тобто, коли $q = p^m$, де $p \geq 2$, $m \geq 2$, зокрема чотириколірних та восьмиколірних ШК-знаків.

У Додатку Е у 4-х таблицях наведено показники виявлення багатократних помилок 24-ма многозначними кодами БЧХ.

$G_{(16,7)}=$

z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
1	1	2	2	2	1	1	1	2	1	0	0	0	0	0	0
0	1	1	2	2	2	1	1	1	2	1	0	0	0	0	0
0	0	1	1	2	2	2	1	1	1	2	1	0	0	0	0
0	0	0	1	1	2	2	2	1	1	1	2	1	0	0	0
0	0	0	0	1	1	2	2	2	1	1	1	2	1	0	0
0	0	0	0	0	1	1	2	2	2	1	1	1	2	1	0
0	0	0	0	0	0	1	1	2	2	2	1	1	1	2	1

$H_{(16,7)}=$

z'_0	z'_1	z'_2	z'_3	z'_4	z'_5	z'_6	z'_7	z'_8	z'_9	z'_{10}	z'_{11}	z'_{12}	z'_{13}	z'_{14}	z'_{15}
0	0	1	0	1	2	1	1	2	0	1	1	1	0	0	2
0	1	0	1	2	1	1	2	0	1	1	1	0	0	2	0
1	0	0	2	0	2	1	2	2	1	0	2	2	2	0	0
0	1	1	1	2	1	1	0	0	1	2	0	2	0	1	1
0	0	2	1	0	1	0	2	2	2	1	2	2	0	0	2
1	0	0	1	2	0	2	0	1	1	1	2	1	1	0	0
0	0	1	0	1	2	1	1	2	0	1	1	1	0	0	2
0	1	1	1	0	0	2	0	2	1	2	2	1	0	2	2
1	2	1	1	2	0	1	1	1	0	0	2	0	2	1	2
0	1	2	1	0	2	2	1	1	1	0	1	0	0	1	2
0	2	0	0	2	1	2	0	1	1	2	2	2	0	2	0
1	0	2	2	1	1	1	0	1	0	0	1	2	1	0	2

Рис. 2. 10. Твірна (*G*) та перевірна (*H*) матриці трійкового скороченого (16,7)-коду БЧХ

На основі кожної з цих таблиць можна побудувати сімейство штрихових кодів (4 сімейства) відповідної колірності (q=3, 4, 5, 7), символіки яких мають властивість завадостійкості – окрім виправлення двократних помилки в межах кожного ШК-знака, забезпечується виявлення значного відсотка помилок більшої кратності (3 – 6 помилок): триколірними ШК-знаками – (29,5 – 94,1)%, чотириколірними ШК-знаками – (70,4 – 96,6)%, п’ятиколірними ШК-знаками – (88,5 – 96,0)%, семиколірними – (97,9 – 99,0)%.

Як уже зазначалось, саме властивість виявляти багатократні помилки є визначальною на нижньому рівні (рівні ШК-знаків) забезпечення завадостійкості багатоколірних ШК, оскільки саме через цей параметр (виявлення багатократних помилок) здійснюється зв'язок з верхнім рівнем забезпечення завадостійкості (рівень усієї ШК-позначки) на основі коректувального коду Ріда-Соломона.

2. 4. Метод синтезу символік багатоколірних завадостійких штрихових кодів

Для синтезу символіки багатоколірного штрихового коду задають три параметри: 1) колірність коду (q) – кількість кольорів, що використовуються для забарвлення елементів (чарунок) ШК-знаків; 2) потрібна (бажана) потужність символіки (N_Ω) – кількість ШК-знаків, які використовуються для формування штрихкодових зображень на носії, та службових цілей забезпечення процесу сканування; 3) коректувальна здатність ШК-знаків (ξ) – кількість помилок, яку необхідно виправляти в межах кожного ШК-знака при зчитуванні ($\xi = 1$ – у випадку менш несприятливого середовища експлуатації ШК-позначок; $\xi = 2$ – у випадку більш несприятливого середовища).

Маючи вихідну тріаду параметрів $< q, N_\Omega, \xi >$ для проєктування множини завадостійких ШК-знаків виконаємо наступну послідовність дій.

1. Вибір коректувального коду для синтезу символіки.

   Якщо $\xi = 1$, то символіку слід синтезувати на основі q-кового коду Хемінга; якщо $\xi = 2$, то для синтезу символіки слід обрати q-ковий код БЧХ.

2. Вибір параметрів (s, u) коректувального коду.

   Знаючи ξ, а також q у відповідній таблиці (табл. 2. 9 – якщо $\xi = 1$, або в табл. 2. 10 – якщо $\xi = 2$) знаходимо в стовпці q значення W (максимальна потужність символіки) близьке до N_Ω, але таке, що $W > N_\Omega$. Ліворуч від знайденого W фіксуємо пару (s, u), – це параметри коректувального коду.

3. Генерування векторів ШК-знаків.

Беручи всі можливі u-розрядні q-кові слова – від 0 0…0 до q - 1 q - 1… q - 1, та застосувавши до кожного з них процедуру кодування відповідним (s, u)-кодом (Хемінга або БЧХ) отримаємо $q^u = W$ s-розрядних слів, які є векторами ШК-знаків.

Таблиця 2. 9

Максимальні потужності символік багатоколірних (q) завадостійких ШК на основі (s, u)-кодів Хемінга ($\xi = 1$)

q = 3		q = 4		q = 5		q = 7	
(s, u)	W	(s, u)	W	(s, u)	W	(s, u)	W
(7,4)-	81	(6,3)-	64	(4,2)-	25	(4,2)-	49
(8,5)-	243	(7,4)-	256	(5,3)-	125	(5,3)-	343
(9,6)-	729	(8,5)-	1024	(6,4)-	625	(6,4)-	2401
(10,7)-	2187	(9,6)-	4096	(8,5)-	3125	(7,5)-	16807
(11,8)-	6561	(10,7)-	16384	(9,6)-	15625	-	-

4. Формування масиву ненульових векторів.

Відкидаємо нульовий вектор (оскільки всі елементи відповідного ШК-знака будуть забарвлені однаковим кольором, що є неприйнятним для штрихкодового зображення – це одноколірна «пляма»), залишаючи W - 1 ненульових векторів.

Таблиця 2. 10

Максимальні потужності символік багатоколірних (q) завадостійких ШК на основі (s, u)-кодів БЧХ ($\xi = 2$)

$q = 3$		$q = 4$		$q = 5$		$q = 7$	
(s, u)	W	(s, u)	W	(s, u)	W	(s, u)	W
(13,4)-	81	(9,3)-	64	(10,2)-	25	(10,2)-	49
(14,5)-	243	(10,4)-	256	(11,3)-	125	(11,3)-	343
(15,6)-	729	(11,5)-	1024	(12,4)-	625	(12,4)-	2401
(16,7)-	2187	(12,6)-	4096	(13,5)-	3125	(13,5)-	16807
(17,8)-	6561	(13,7)-	16384	(14,6)-	15625		

5. Проріджування масиву ненульових векторів (накладання маски на ШК-знаки).

   З метою недопущення випадкового утворення на ШК-позначці великої групи сусідніх елементів, що мають однакове забарвлення («плями»), на колірну структуру ШК-знаків можуть накладатися обмежувальні умови – всі елементи рядка та всі елементи стовпця не повинні бути однакового кольору. Або інші – більш строгі умови. Така процедура застосовується для спрощення зчитування ШК-позначки сканувальним пристроєм (поліпшується синхронізація процесу розрізнення елементів штрихкодового зображення).

   Нехай після проріджування залишилось N_Q векторів.

6. Формування геометричного набору багатоколірних ШК-знаків (символіки).

Кожному з векторів ставлять у відповідність ШК-знак – геометричне зображення, елементи якого забарвлені відповідно до значень компонент вектора (цифрового еквівалента ШК-знака).

Отримані в такий спосіб ШК-знаки є завадостійкими, оскільки відповідний вектор ШК-знака (цифровий еквівалент) є кодовим словом завадостійкого q -кового (s, u)-коду (Хемінга або БЧХ).

Разом вони утворюють символіку багатоколірного завадостійкого штрихового коду.

Розглянемо приклад.

Нехай необхідно на основі вихідних параметрів $< q, N_\Omega, \xi >$ = <4, 45, 2> синтезувати символіку чотириколірного (q = 4) завадостійкого штрихового коду з кількістю ШК-знаків не меншою ніж 45 ($N_\Omega \geq 45$), у яких при зчитуванні з носія забезпечується виправлення двократних помилок (ξ = 2) у межах кожного ШК-знака, за умови, що в ШК-знаках не допускається одноколірності рядків та стовпців.

Для побудови символіки із заданими параметрами та обмежувальної умови щодо колірної структури ШК-знаків виконаємо наступну послідовність дій.

1. Оскільки ξ = 2, синтез символіки виконуватимемо на основі четвіркового (q = 4) коду БЧХ.
2. Визначимо параметри (s, u) четвіркового коду БЧХ за умови, що $N_\Omega \geq 45$.

   Для цього в табл. 2. 10 у стовпці q = 4 знаходимо найближче до N_Ω = 45 більше значення W = 64 – максимально можлива потужність символіки. Значенню W = 64 у табл. 2. 10 відповідає четвірковий (9, 3)-код БЧХ. Отже, s = 9, u = 3.
3. Згенеруємо вектори ШК-знаків. Візьмемо всі можливі 3-розрядні (u = 3) четвіркові слова – від 0 0 0 до 3 3 3, та застосуємо до кожного з них процедуру кодування четвірковим (9, 3)-кодом БЧХ. Результатом кодування є масив, що складається з 64-х 9-розрядних кодових слів коду

БЧХ, наведених у Додатку Є. Кожне 9-розрядне слово цього масиву є вектором відповідного ШК-знака. На основі кожного 9-розрядного вектора будуємо ШК-знак розмірності 3х3. Можливі чотири кольори позначимо цифрами: 0, 1, 2, 3.

4. Сформуємо масив ненульових векторів. Для цього відкинемо нульовий вектор 000000000 (у Додатку Є це позначено як “ХХ”). Кількість ненульових векторів становить 63.
5. Виконаємо процедуру проріджування масиву ненульових векторів так, щоб рядки та стовпці ШК-знаків не містили однакових цифр. Таких випадків – 18, вони позначені символом “Х” справа від зображення ШК-знака (див. Додаток Є).

   Отже, після проріджування залишилось $N_\Omega = 64 - 1 - 18 = 45$ векторів.
6. Сформуємо множину чотириколірних завадостійких ШК-знаків, кількість яких становить 45 (Додаток Ж). Вони утворюють символіку завадостійкого чотириколірного ШК. ШК-знаки символіки занумеруємо від 0 до 44 (див. Додаток Ж).

   Розглянемо інший приклад.

Нехай необхідно синтезувати символіку п’ятиколірного ($q = 5$) завадостійкого штрихового коду з кількістю ШК-знаків не меншою ніж 60, у яких би забезпечувалось виправлення однократних помилок під час зчитування, та за умови, що в структурі ШК-знаків не допускається одноколірності рядків та стовпців.

Для побудови символіки із заданими параметрами $<q = 5,\ N_\Omega \geq 60,\ \xi = 1>$ виконаємо таку послідовність дій.

1. Синтез символіки виконуватимемо на основі п’ятіркового ($q = 5$) коду Хемінга, оскільки $\xi = 1$.
2. Визначимо параметри (s, u) п’ятіркового коду Хемінга за умови $N_\Omega \geq 60$. Для цього в табл. 2. 9 у стовпці $q = 5$ знаходимо найближче більше до $N_\Omega = 60$ значення $W = 125$ – максимально можлива потужність символіки.

Значенню W = 125 в таблиці відповідає п’ятірковий (5, 3)-код Хемінга. Отже, s =5, u = 4.

3. Згенеруємо вектори ШК-знаків. Для цього візьмемо всі можливі 3-розрядні (u = 3) п’ятіркові слова – від 000 до 444, і застосуємо до кожного з них процедуру кодування п’ятірковим (5, 3)-кодом Хемінга. Результатом кодування є масив з 125-ти 5-розрядних кодових слів п’ятіркового коду Хемінга.
4. Сформуємо масив ненульових векторів. Для цього відкинемо нульовий вектор 00000. Кількість ненульових векторів становить 124.
5. Виконаємо процедуру проріджування масиву ненульових векторів так, щоб рядки та стовпці ШК-знаків не містили однакових цифр (рис. 2. 11).

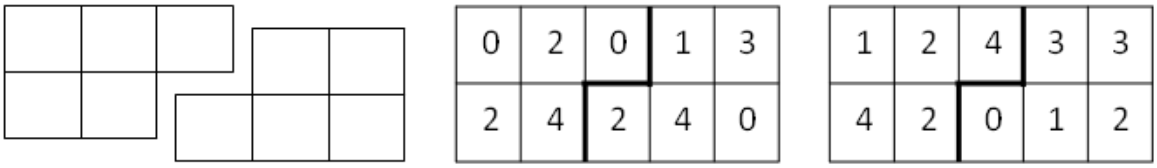


Рис. 2. 11. Структура ШК-знаків символіки п’ятиколірного ШК на основі 5-кового (5, 3)-коду Хемінга:
а – спосіб обєднання ШК-знаків у ШК-позначці штрихового коду; *б* – приклад дозволених ШК-знаків; *в* – приклад заборонених ШК-знаків (кольори елементів позначено цифрами).

Після проріджування отримаємо 64 вектори (Додаток К).

6. Сформуємо множину п’ятиколірних завадостійких ШК-знаків п’ятиколірного ШК.
ШК-знаки символіки занумеруємо від 0 до 63 (див. Додаток К).

2. 5. Висновки до другого розділу

Виконані дослідження щодо забезпечення завадостійкості багатоколірних штрихкодових знаків як базових структурних одиниць багатоколірних штрихкодових зображень дозволяють зробити такі висновки.

1. Оскільки штрихкодове зображення складається з штрихкодових знаків структурно організованих у вигляді матриці, то для забезпечення цілісності даних багатоколірний штриховий код слід створювати так, щоб властивість завадостійкості забезпечувалась на двох рівнях – на рівні ШК-позначки (штрихкодового зображення), а також на рівні ШК-знаків – мінімальних структурних одиниць зображення.
2. Показано, що символіку (набір ШК-знаків) багатоколірного штрихового коду слід створювати так, щоб вектор (цифровий еквівалент) кожного ШК-знака був кодовим словом деякого многозначного коректувального коду, здатного виправляти одно- або двократні помилки в межах ШК-знака та частково виявляти багатократні помилки. Запропоновано використовувати для цього многозначний код Хемінга або многозначний код БЧХ з мінімальною кодовою відстанню 5.
3. Розроблено алгоритмічне забезпечення завадостійкості багатоколірних ШК-знаків на основі многозначного коду Хемінга, зокрема алгоритм генерування багатоколірних завадостійких ШК-знаків з виправленням однократних помилок та виявлення помилок більшої кратності у ШК-знаках.

Показано, що виявлення багатократних помилок у ШК-знаках можливе лише при використанні скорочених (неповних) многозначних кодів Хемінга. На основі розробленого програмного продукту досліджено здатність виявляти багатократні помилки для понад двадцяти многозначних скорочених кодів Хемінга, на основі яких можна будувати 5 сімейств багатоколірних завадостійких штрихових кодів – за кількістю кольорів (3, 4, 5, 7, 8 кольорів).

4. Розроблено алгоритмічне забезпечення завадостійкості багатоколірних штрихових знаків на основі многозначного коду БЧХ з мінімальною кодовою відстанню 5, зокрема алгоритм генерування багатоколірних завадостійких ШК-знаків, алгоритм декодування ШК-знаків з виправленням двократних помилок та частковим виявленням помилок більшої кратності.

   Показано, що всі коди БЧХ здатні виявляти частину багатократних помилок, кратність яких перевищує 2. Однак, ця властивість істотно посилюється, якщо застосовувати скорочені коди БЧХ.

   На основі розробленого програмного продукту досліджено здатність виявляти помилки, кратність яких перевищує два, для понад двадцяти многозначних кодів БЧХ, на основі яких можна будувати 4 сімейства багатоколірних (3, 4, 5, 7-колірних) завадостійких штрихових кодів.

5. Розроблено метод синтезу символіки заданої потужності та колірності завадостійкого штрихового коду для реалізації в медичній інформаційній системі на основі багатоколірного штрихового кодування інформації, який грунтується на тому, що вектори (цифрові еквіваленти) багатоколірних ШК-знаків є кодовими словами многозначного коректувального коду, здатного виправляти одно- або двократні помилки в межах кожного штрихкодового знака, що при скануванні ШК-знаків забезпечуватиме достовірне відтворення даних або виявлення значної частини багатократних ушкоджень елементів ШК-знака. Запропоновано синтезувати символіки на основі скорочених многозначних кодів Хемінга або на основі многозначних кодів БЧХ.

   Метод синтезу символіки багатоколірного штрихового коду передбачає послідовне виконання таких процедур: вибір многозначного коректувального коду та його параметрів, генерування масиву ненульових векторів ШК-знаків, проріджування масиву ненульових векторів (накладання “маски” на ШК-знаки) з метою недопущення випадкового утворення на штрихкодовому зображенні великої групи сусідніх елементів,

що мають однакове забарвлення (“плями”), формування геометричного набору багатоколірних ШК-знаківа та способу їх з’єднання у штрихкодовій позначці.

6. За допомогою статистичних випробувань на основі розробленого програмного продукту визначено здатність виявляти помилки кратності 2 - 6 у ШК-знаках, синтезованих на основі кодів Хемінга, - у триколірних ШК-знаках: (6,8 – 60,0)%; у чотириколірних: (11,1 – 70,8)%; у п’ятиколірних: (12,5 – 92,5)%; у семиколірних: (11,1– 66,7)%; у восьмиколірних: (20,4 – 71,4)%. Якщо синтезувати символіки на основі многозначних кодів БЧХ з мінімальною кодовою відстанню 5, то забезпечується виявлення такого відсотка помилок кратності 3 – 6: у триколірних: (29,5 – 94,1)%;)%; у чотириколірних: (70,4 – 96,6)%; у п’ятиколірних: (88,5 – 96,0)%; у семиколірних: (97,5 – 99,0)%.

РОЗДІЛ 3. ПІДВИЩЕННЯ ЗАВАДОСТІЙКОСТІ БАГАТОКОЛІРНИХ ШТРИХОВИХ КОДІВ У МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

3. 1. Проєктування багатоколірного завадостійкого штрихового коду для автоматичної ідентифікації в медичній інформаційній системі

Для забезпечення процесів автоматичної ідентифікації в медичних інформаційних системах пропонується застосовувати багатоколірне штрихове кодування інформації. Оскільки передачу, пошук та конфіденційний доступ пацієнтів до своїх медичних даних передбачається здійснювати з використанням мобільних пристроїв слід застосовувати завадостійкий штриховий код, який би забезпечував цілісність даних в умовах можливого спотворення штрихкодового зображення.

Розроблення багатоколірного завадостійкого штрихового коду слід здійснювати в такому порядку: проєктування алфавіту та символіки ШК, визначення структури ШК-позначки, кодування вхідних алфавітно-цифрових даних, завадостійке кодування даних, розроблення алгоритму декодування багатоколірного штрихового зображення.

Проєктування алфавіту та символіки ШК. На основі заданих параметрів $< q, N_\Omega, \xi >$ синтезуємо символіку Ω завадостійкого штрихового коду – набір ШК-знаків штрихового коду (про це йшлося у підрозділі 2. 4), де q – колірність коду (кількість використовуваних кольорів для забарвлення елементів ШК-знаків, N_Ω – потужність символіка (кількість ШК-знаків), ξ - коректувальна здатність ШК-знаків ($\xi = 1$ – на основі q-кового коду Хемінга, $\xi = 2$ – на основі q-кового коду БЧХ).

Знаходимо просте число P – близьке до N_Ω, але менше за нього ($P < N_\Omega$). Перші P ШК-знаків символіки (з порядковими номерами 0 – P-1) будемо використовувати для подання інформації у штрихкодовому вигляді; назвемо їх інформаційними ШК-знаками. Решту ШК-знаків символіки ($n_{сл} = N_\Omega - P$)

з порядковими номерами P – N_Q - 1 вважатимемо службовими - для цілей забезпечення процесу сканування ($n_{сл}$ – кількість службових ШК-знаків). За модулем P виконуватимуться операції кодування-декодування даних.

Маючи в розпорядженні P ШК-знаків організуємо їх використання в такий спосіб, щоб мати можливість подавати у штрихкодовому вигляді якомога більшу кількість символів ASCII. Для цього утворимо декілька наборів з символів ASCII, наприклад набір букв латинської абетки (набір A), набір букв кирилиці (набір B), набір спеціальних символів ASCII (набір C) тощо.

Наприклад, взявши символіку чотириколірного ШК на основі четвіркового (9, 3) – коду БЧХ (Додаток Г), яка налічує N_Q = 45 ШК-знаків, можна створити алфавіт штрихового коду таким чином: вибираємо P = 41, тоді отримаємо 41 інформаційних ШК-знаків та $n_{сл}$ = 3 службові ШК-знаки; утворюємо три набори символів з ASCII – набір A, набір B, набір C (табл. 3.1).

Утворивши три набори символів слід запровадити в алфавіті ШК спеціальні символи-перемикачі наборів: setA – виконує перехід з поточного набору в набір A (наприклад з набору B або C); setB – виконує перехід з поточного набору (A або C) в набір B; setC– виконує перехід з поточного набору (A або B) в набір C. Крім того, в алфавіті ШК слід передбачити символ-заповнювач (pad), який не несе інформаційного навантаження і використовується для заповнення вільних позицій в останньому рядку масиву ШК-знаків, щоб доповнити штрихкодове зображення до прямокутної (квадратної) форми.

Структурна організація ШК-позначки. Вважатимемо, що багатоколірна штрихкодова позначка (рис. 3. 1) містить такі структурні елементи: L-подібний маркер, чотири мірні лінійки (по периметру) та двовимірний масив штрихкодових знаків.

Таблиця 3. 1

Алфавіт та символіка чотириколірного штрихового коду на основі четвіркового (9, 3)-коду БЧХ (латинська та кирилична абетки)

Номер символу	Набір A	Набір B	Набір C	Вектор ШК-знака								
0	A	А	‘	0	1	3	0	3	0	2	2	1
1	B	Б	,	0	1	0	1	2	3	1	0	2
2	C	В	;	0	1	1	3	0	2	0	3	3
3	D	Г	(	0	2	2	1	0	3	0	1	1
4	E	Д	)	0	2	1	0	1	0	3	3	2
5	F	Е	-	0	2	0	2	3	1	2	0	3
6	G	Є	/	0	3	0	3	1	2	3	0	1
7	H	Ж	:	0	3	3	2	0	1	0	2	2
8	I	З	@	0	3	2	0	2	0	1	1	3
9	J	И	#	1	2	3	3	3	2	0	3	1
10	K	І	%	1	2	0	2	2	1	3	1	2
11	L	Ї	!	1	3	0	3	0	2	2	1	0
12	M	Й	“	1	3	1	1	2	3	3	2	1
13	N	К	$	1	3	2	0	3	0	0	0	2
14	O	Л	&	1	3	3	2	1	1	1	3	3
15	P	М	*	1	0	1	2	3	1	0	2	0
16	Q	Н	+	1	0	3	1	0	3	2	3	2
17	R	О	=	1	0	2	3	2	2	3	0	3
18	S	П	<	1	1	3	0	2	0	3	3	0
19	T	Р	>	1	1	2	2	0	1	2	0	1
20	U	С	?	1	1	0	1	3	3	0	1	3
21	V	Т	[	2	3	1	1	1	3	0	1	2
22	W	У	]	2	3	0	3	3	2	1	2	3
23	X	Ф	^	2	2	1	0	3	0	1	1	0
24	Y	Х	_	2	2	0	2	1	1	0	2	1
25	Z	Ц	{	2	2	3	3	0	2	3	0	2
26	0	Ч	\	2	1	0	1	0	3	3	2	0
27	1	Ш	}	2	1	1	3	2	2	2	1	1
28	2	Щ	HT	2	1	2	2	3	1	1	3	2
29	3	Ю	VT	2	1	3	0	1	0	0	0	3
30	4	Я	BEL	2	0	2	3	1	2	0	3	0
31	5	Ь	LF	2	0	3	1	3	3	1	0	1
32	6	‘	ACK	2	0	1	2	0	1	3	1	3
33	7	,	CR	3	1	0	1	1	3	2	3	1
34	8	-	EOT	3	1	2	2	2	1	0	2	3
35	9	:	ENQ	3	0	3	1	2	3	0	1	0
36	space	space	space	3	0	2	3	0	2	1	2	1
37	.	.	.	3	0	1	2	1	1	2	0	2
38	pad	pad	pad	3	3	2	0	1	0	2	2	0
39	setB	setA	setA	3	3	0	3	2	2	0	3	2
40	setC	setC	setB	3	3	1	1	0	3	1	0	3

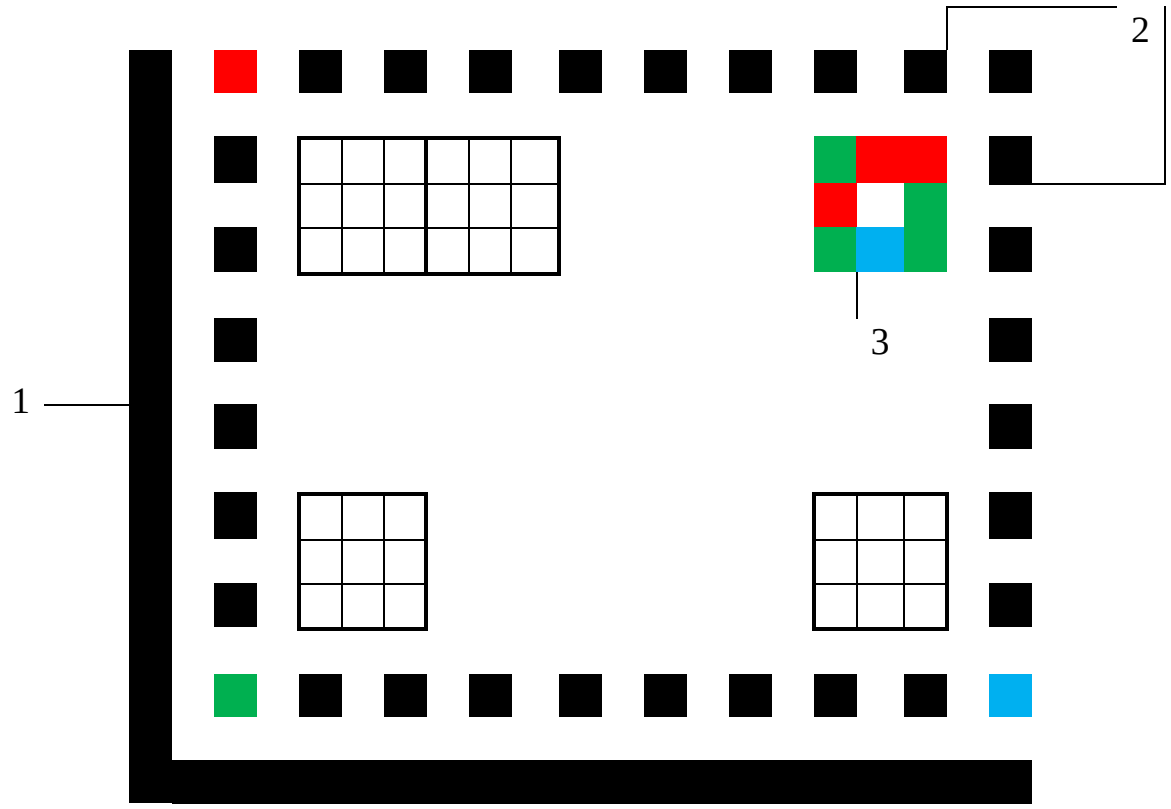


Рис. 3. 1. Структурна організація ШК-позначки чотириколірного штрихового коду: 1 – L-подібний ідентифікатор (маркер); 2 – мірні лінійки; 3 – ШК-знак

L-подібний маркер використовують для визначення орієнтації зображення відносно пристрою зчитування (смартфона). Мірні лінійки задають розміри мінімальних штрихкодових елементів (чарунок). Чарунка має форму квадрата, її ширина та висота дорівнює одному модулю. Модуль – це мінімальна ширина та мінімальна висота, якій кратні розміри всіх елементів штрихкодового зображення; модуль є безрозмірною величиною, еталоном мінімального геометричного елемента. Тому чарунку також називають модулем [110, 117]. Разом чотири мірні лінійки утворюють бордюрний рисунок. У випадку чотириколірного ШК верхня ліва чарунка бордюрного рисунка забарвлена червоним (R) кольором, нижня ліва – зеленим (G), а нижня права – синім кольором (B) (див. рис. 3.1). Три забарвлені різноколірні чарунки

бордюрного рисунка є ключовими точками, що можуть використовуватись як еталони кольорів R, G, B. Четвертим кольором є колір носія (субстрату) – білий. Службовим (а не інформаційним) є чорний колір, яким забарвлено L-подібний маркер та чарунки бордюрного рисунка (крім трьох зазначених).

*Кодування вхідних алфавітно-цифрових даних***.** В алфавіті штрихового коду одному ШК-знаку (який в табл. 3.1 представлений вектором ШК-знака) відповідають три символи ASCII, що належать трьом різним наборам – Набору A, Набору B, Набору C. Наприклад, ШК-знаку, що представлений вектором 2 3 0 3 3 2 1 2 3, і має порядковий номер 22, відповідають: латинська літера W, що належить Набору A; кирилична літера У, що належить Набору B; спеціальний символ] (квадратна дужка), що належить Набору C. Тому в алфавітно-цифрову послідовність (що вводиться з клавіатури), яку належить подати у вигляді штрихкодового зображенння, слід за потреби вставляти символи-перемикачі (setA, setB, setC) – для означення набору, до якого належать кодовані символи. Поточний символ-перемикач діє або до кінця кодованої алфавітно-цифрової послідовності, або до появи іншого символу-перемикача.

Нехай у штрихкодовому вигляді необхідно подати алфавітно-цифрову послідовність (повідомлення)

52AM&Ю (3. 1)

утворену з 6-ти символів ASCII, що вводяться з клавіатури. Для перетворення послідовності (3.1) скористаємося табл. 3. 1. Символи 5, 2, A, M належать Набору A, символ & (амперсенд) – Набору C, а символ Ю – Набору B. Тоді послідовності (3. 1), утвореній з символів ASCII, відповідатиме така послідовність символів штрихового коду

5 2 A M setC & setB Ю (3. 2)

яку назвемо адаптованим повідомленням.

Оброблення вхідної алфавітно-цифрової послідовності (3. 1) завжди починається з Набору A (табл. 3.1).

Далі, кожен символ з (3. 2) замінимо порядковим номером ШК-знака (див. табл. 3. 1) з символіки штрихового коду:

5	2	A	M	setC	&	setB	Ю
31	28	0	12	40	14	40	29
набір A					набір C		набір B

(3. 3)

Для надання властивості завадостійкості отриманій числовій послідовності (3. 3), що є масивом чисел з діапазону 0 ÷ 40 , її необхідно закодувати коректувальним кодом, здатним виправляти багатократні спотворення. Застосуємо для цього коректувальний код Ріда-Соломона (рис. 3. 2).

Числову послідовність (3. 3) називатимемо інформаційним словом.

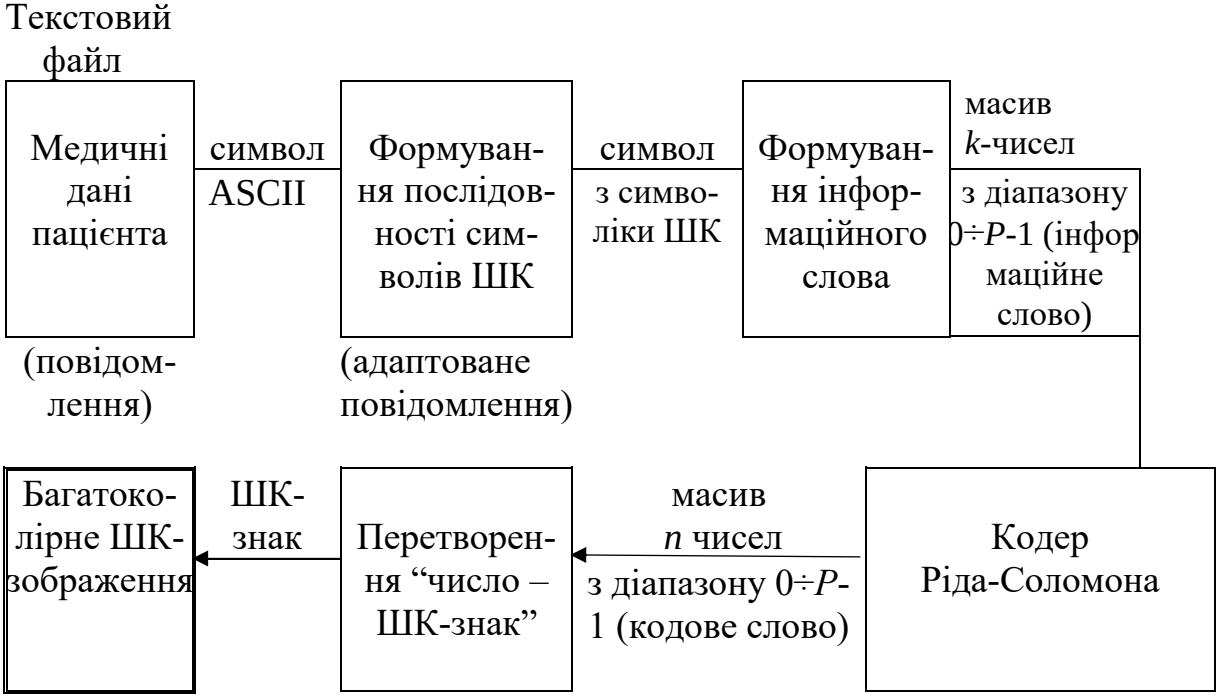


Рис. 3.2. Схема формування багатоколірного завадостійкого штрихкодового зображення для зберігання медичних даних пацієнта

3. 2. Завадостійке кодування медичних даних пацієнта

Нехай у штрихкодовому вигляді необхідно подати масив k чисел, що утворюють інформаційне слово $A=(a_{k-1}\ a_{k-2}\ldots\ a_1\ a_0)$, якому поставимо у

відповідність многочлен $a(x)=\sum_{i=0}^{k-1} a_i x^i$, де a_i є {0, 1, 2, …, P-1}. Для забезпечення цілісності даних (оскільки ШК-позначка є відкритим носієм інформації, що може піддаватися небажаним зовнішнім впливам, завадам) застосуємо коректувальний код Ріда-Соломона, який долучає до інформаційного слова r контрольних розрядів (рис. 3. 2). Внаслідок цього на носій буде нанесено n ШК-знаків, які утворюють кодове слово $C=(c_{n-1}\ c_{n-2}\ldots\ c_1\ c_0)$, і якому поставимо у відповідність многочлен $c(x)=\sum_{j=0}^{n-1} c_j x^j$, де $n - k = r$. Слово C отримують як результат процедури кодування: $c(x)=a(x)\varphi(x)$, де $\varphi(x)$ – твірний многочлен коду Ріда-Соломона. Вважатимемо, що операції над коефіцієнтами многочленів виконують за модулем простого числа P.

Вважатимемо також, що максимальна ємність ШК-позначки не перевищує P-1 ШК-знаків, тобто $n \le P$-1.

Твірний многочлен $\varphi(x)$ коду Ріда-Соломона визначають як

$$\varphi(x)=(x-\beta)(x-\beta^2)\cdot\ldots\cdot(x-\beta^r)= x^r+\varphi_{r-1}x^{r-1}+\ldots+\varphi_1 x+\varphi_0,$$

де β – примітивний елемент поля $GF(P)$. Примітивним елементом поля $GF(P)$ є ціле число, всі можливі степені якого за модулем P дають всі ненульові елементи поля, тобто числа 1, 2, … , P-1.

Наприклад, примітивним елементом поля $GF(11)$ є β=2, оскільки всі можливі степені числа 2 за модулем 11 дають числа 1, 2, … , 10; примітивним елементом поля $GF(41)$ є β=7.

Зазвичай, для поля $GF(P)$, де P – просте, примітивним елементом поля може бути β=2, 3, 5, 7.

Код Ріда-Соломона має мінімальну кодову відстань $d^*=n - k + 1 = r + 1$. З іншого боку, відомо, що коректувальний код з мінімальною кодовою відстанню d^* дозволяє декодувати будь-яку конфігурацію з t помилок та ρ стирань за умови, що $d^* \ge 2t + 1 + \rho$. У теорії завадостійкого кодування під помилкою розуміють таке спотворення у слові C, коли не відоме ні його

місцезнаходження у слові, ані його величина; а під стиранням – спотворення, місцезнаходження якого у слові відоме, але не відома тільки його величина (у данному разі стирання є математичним поняттям, а не фізичним) [41].

Отже,

$$r \geq 2t + \rho \qquad (3.\ 4)$$

Це співвідношення дає можливість користувачу обирати потрібну кількість контрольних розрядів для захисту цілісності даних залежно від ємності ШК-позначки та середовища використання штрихового коду, регулюючи таким чином рівень захищеності.

Наприклад, якщо є потреба виправляти в словах $t = 2$ помилки та ρ=3 стирання, то кількість контрольних розрядів має становити 7.

Зі співвідношення (3. 4) випливає, що для корекції кожної помилки в коді Ріда-Соломона витрачаються два контрольні розряди, а для корекції кожного стирання – один контрольний розряд.

Нехай у штрихкодовому вигляді необхідно подати повідомлення (3. 3), Невелика довжина повідомлення (3. 3) обрана для спрощення прикладів кодування-декодування; реальні ШК-позначки у своєму складі налічують від кількох сотень до кількох тисяч ШК-знаків.

Повідомленню (3. 3) відповідає 8-розрядне інформаційне слово A=($a_7a_6a_5a_4a_3a_2a_1a_0$)=(31 28 0 12 40 14 40 29). Задамо кількість контрольних розрядів, наприклад, r =6. Така кількість контрольних розрядів дозволяє виправляти такі конфігурації помилок (t) та стирань (ρ) : t = 3, ρ=0; t = 2, ρ=2; t = 1, ρ=4; t = 0, ρ=6. Тоді твірний многочлен коду Ріда-Соломона дорівнює $\varphi(x)=(x–7)(x–7^2)\cdot\ldots\cdot (x–7^6) = x^6+12x^5+8x^4+34x^3+13x^2+24x+34$ (коефіцієнти обчислено за модулем 41).

Закодуємо слово A коректувальним кодом Ріда-Соломона:

$c(x) = a(x)\varphi(x) = (31x^7+28x^6+12x^4+40x^3+14x^2+40x+29)\cdot(x^6+12x^5+8x^4+34x^3+$

$+13x^2 +24x+34) = 31x^{13}+31x^{12}+10x^{11}+19x^{10}+22x^9+17x^8+38x^7+14x^6+25x^5+$

$+26x^4+4x^3+9x^2+6x+2.$

Кодове слово дорівнює:

$$C = (31\ \ 31\ \ 10\ \ 19\ \ 22\ \ 17\ \ 38\ \ 14\ \ 25\ \ 26\ \ 4\ \ 9\ \ 6\ \ 2) \qquad (3.5)$$

$$c_{13}\ c_{12}\ c_{11}\ c_{10}\ c_9\ \ c_8\ \ c_7\ \ c_6\ \ c_5\ \ c_4\ \ c_3\ c_2\ c_1\ c_0.$$

На основі кодового слова (3. 5) формують числову модель ШК-позначки (рис. 3. 3). Для того, щоб ШК-позначка була квадратною, її доповнюють двома ШК-знаками символа-заповнювача (pad), номер якого 38. Далі, користуючись табл. 3. 1 порядкові номери ШК-знаків замінюють 9-розрядними векторами ШК-знаків, що мають розмірність 3х3, отримуючи таким чином колірну модель ШК-позначки, кожну чарунку якої забарвлюють відповідним кольором (див. рис. 3. 3).

Нехай текстове повідомлення 52АМ&Ю необхідно подати у вигляді п’ятиколірного ШК, символіка якого утворена на основі п’ятіркового (5, 3)-коду Хемінга (Додаток Д) і налічує 64 ШК-знаки.

Зафіксувавши, наприклад, просте число $P = 59$, яке є близьким до числа 64 (але меншим за нього) , створимо алфавіт ШК (табл. 3.2). В алфавіті та символіці цього штрихового коду 59 ШК-знаків (з порядковими номерами 0 - 58) будемо використовувати для подання інформації у штрихкодовому вигляді, а ШК-знаки з порядковими номерами 59 – 63 вважатимемо службовими.

У цьому ШК символи ASCII організуємо в 4 набори (Набір А – Набір D), а для переходу з одного набору до іншого використовуватимемо 4 символи-перемикачі setA – setD.

Текстове повідомлення: 52АМ&Ю
Адаптоване повідомлення: 5 2 А М setC & setB Ю
Інформаційне слово: 31 28 0 12 40 14 40 29
Кодове слово коду Ріда-Соломона:31 31 10 19 22 17 38 14 25 26 4 9 6 2

31	31	10	19
22	17	38	14
25	26	4	9
6	2	38	38

Числова модель ШК-позначки

ШК-знак (31)

2	0	3
1	3	3
1	0	1

2 0 3 1 3 3 1 0 1
(кодове слово коду БЧХ)

2	0	3	2	0	3	1	2	0	1	1	2
1	3	3	1	3	3	2	2	1	2	0	1
1	0	1	1	0	1	3	1	2	2	0	1
2	3	0	1	0	2	3	3	2	1	3	3
3	3	2	3	2	2	0	1	0	2	1	1
1	2	3	3	0	3	2	2	0	1	3	3
2	2	3	2	1	0	0	2	1	1	2	3
3	0	2	1	0	3	0	1	0	3	3	2
3	0	2	3	2	0	3	3	2	0	3	1
0	3	0	0	1	1	3	3	2	3	3	2
3	1	2	3	0	2	0	1	0	0	1	0
3	0	1	0	3	3	2	2	0	2	2	0

Колірна модель ШК-позначки

0 – білий
1 - червоний
2 - зелений
3 - синій

ШК-позначка чотириколірного штрихового коду

Рис. 3. 3**.** Приклад формування ШК-позначки 4-колірного завадостійкого штрихового коду (символіка – на основі четвіркового (9, 3)-коду БЧХ)

Таблиця 3. 2

Алфавіт та символіка п’ятиколірного штрихового коду потужності 59 на основі п’ятіркового (5,3)-коду Хемінга (латинська та кирилична абетки)

Номер символу	Набір A	Набір B	Набір C	Набір D	Вектор ШК-знака				
1	2	3	4	5	6				
0	A	a	А	а	0	0	1	1	3
1	B	b	Б	б	0	0	2	2	1
2	C	c	В	в	0	0	3	3	4
3	D	d	Г	г	0	0	4	4	2
4	E	e	Д	д	0	1	0	1	2
5	F	f	Е	е	0	1	1	2	0
6	G	g	Є	є	0	2	0	2	4
7	H	h	Ж	ж	0	2	2	4	0
8	I	I	З	з	0	3	0	3	1
9	J	J	И	и	0	3	3	1	0
10	K	k	І	і	0	4	0	4	3
11	L	l	Ї	ї	0	4	4	3	0
12	M	m	Й	й	1	0	1	2	4
13	N	n	К	к	1	0	2	3	2
14	O	o	Л	л	1	0	4	0	3
15	P	p	М	м	1	1	0	2	3
16	Q	q	Н	н	1	1	3	0	2
17	R	r	О	о	1	2	0	3	0
18	S	s	П	п	1	2	1	4	3
19	T	t	Р	р	1	2	2	0	1
20	U	u	С	с	1	3	0	4	2
21	V	v	Т	т	1	3	3	2	1
22	W	w	У	у	1	3	4	3	4
23	X	x	Ф	ф	1	4	2	2	0
24	Y	y	Х	х	1	4	4	4	1
25	Z	z	Ц	ц	2	0	2	4	3
26	0	0	Ч	ч	2	0	3	0	1
27	1	1	Ш	ш	2	0	4	1	4
28	2	2	Щ	щ	2	1	0	3	4
29	3	3	Ю	ю	2	1	1	4	2
30	4	4	Я	я	2	1	3	1	3
31	5	5	Ь	ь	2	2	0	4	1
32	6	6	0	0	2	2	1	0	4
33	7	7	1	1	2	3	3	3	2
34	8	8	2	2	2	3	4	4	0
35	9	9	3	3	2	4	0	1	0
36	!	!	4	4	2	4	2	3	1
37	”	”	5	5	2	4	4	0	2
38	$	$	6	6	3	0	1	4	1
39	&	&	7	7	3	0	2	0	4
40	*	*	8	8	3	0	3	1	2

1	2	3	4	5	6				
41	+	+	9	9	3	1	0	4	0
42	space	space	space	space	3	1	1	0	3
43	‘	‘	‘	‘	3	1	3	2	4
44	.	.	.	.	3	2	1	1	0
45	,	,	,	,	3	2	2	2	3
46	;	;	;	;	3	3	0	1	4
47	(	(	(	(	3	3	4	0	1
48	)	)	)	)	3	4	0	2	1
49	-	-	-	-	3	4	2	4	2
50	/	/	/	/	3	4	4	1	3
51	:	:	:	:	4	0	1	0	2
52	@	@	@	@	4	0	3	2	3
53	#	#	#	#	4	0	4	3	1
54	%	%	%	%	4	1	1	1	4
55	pad	pad	pad	pad	4	1	3	3	0
56	setB	setA	setA	setA	4	2	0	1	3
57	setC	setC	setB	setB	4	2	1	2	1
58	setD	setD	setD	setC	4	2	2	3	4

Зазначеній послідовності з 6-ти символів ASCII, яку вводять з клавіатури, відповідає така послідовність символів штрихового коду та їх порядкових номерів (див. табл. 3. 2).

5	2	A	M	&	setC	Ю
31	28	0	12	39	57	29

(3. 6)

Закодуємо інформаційне слово A = (31 28 0 12 39 57 29) кодом Ріда-Соломона задавши, наприклад, чотири контрольні розряди (r = 4).

Примітивним елементом поля GF(59) є β=2. Тоді твірний многочлен $\varphi(x)$ коду Ріда-Соломона визначимо як

$$\varphi(x) = (x-2)(x-2^2)(x-2^3)(x-2^4) = x^4 + 29x^3 + 44x^2 + 43x + 21,$$

а кодове слово дорівнює

C = (31	42	52	40	0	3	20	27	3	25	19)
c_{10}	c_9	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0

(3. 7)

На основі кодового слова (3. 7) сформуємо числову модель ШК-позначки (рис. 3. 4).

Оскільки вектор ШК-знака містить 5 елементів (непарна кількість), то для формування ШК-позначки необхідно виконувати попарне об’єднання

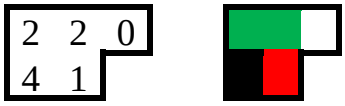
ШК-знаків, у якому лівий ШК-знак пари є прямим, а правий – інверсним. Наприклад, числова пара 31:42′ означає, що ШК-знак з порядковим номером 31 є прямим (вектор: 2 2 0 4 1), а ШК-знак з порядковим номером 42 (вектор: 3 1 1 0 3) слід проінвертувати, тобто його вектор слід записати у зворотному порядку: 42′ (3 0 1 1 3), і після цього об’єднати два ШК-знаки як показано на рис. 3. 4.

Текстове повідомлення: 52АМ&Ю

Адаптоване повідомлення: 5 2 А М & setC Ю

Кодове слово коду Ріда-Соломона: 31 42 52 40 0 3 20 27 3 25 19

ШК-знак (31) - прямий

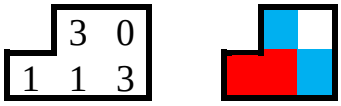


2 2 0 4 1 – вектор ШК-знака (кодове слово коду Хемінга)

31	42′	52	40′
0	3′	20	27′
3	25′	19	55′

Числова модель ШК-позначки

ШК-знак (42′) - інверсний



3 1 1 0 3 – вектор ШК-знака (кодове слово коду Хемінга)

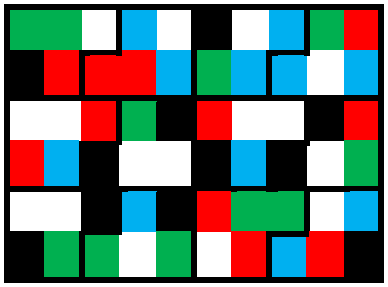
2 2 0 3 0 4 1 1 1 3		4 0 3 2 1 2 3 3 0 3	
0 0 1 2 4 1 3 4 0 0		1 3 0 4 1 4 2 4 0 2	
0 0 4 3 4 4 2 2 0 2		1 2 2 0 3 0 1 3 1 4	

Колірна модель ШК-позначки

0 – білий, 1- червоний

2- зелений, 3- синій

4 - чорний



ШК-позначка п’ятиколірного штрихового коду

Рис. 3. 4. Приклад формування ШК-позначки 5-колірного завадостійкого штрихового коду (символіка – на основі п’ятіркового (5,3)-коду Хемінга)

3. 3. Відтворення медичних даних пацієнта з багатоколірного штрихкодового зображення

Після сканування штрихкодового зображення з носія (яким може бути паперова наліпка або екран смартфона, монітора, планшета) формують двовимірний масив, кожен елемент якого є номером кольору у використовуваній палітрі. Обробляючи цей масив порядково (наприклад, з правого нижнього кута до лівого верхнього, або навпаки) виділяють колірні елементи, що утворюють L-подібний ідентифікатор, бордюрний рисунок та матрицю ШК-знаків. Далі здійснюється аналіз елементів кожного ШК-знака. ШК-знаки обробляють послідовно, і при цьому працює лічильник, який формує номер поточного (оброблюваного в даний момент) ШК-знака. Поточний ШК-знак перетворюють в одновимірний вектор ШК-знака (рис. 3. 5), який декодують внутрішнім декодером (декодером БЧХ або декодером Хемінга – залежно від способу утворення символіки штрихового коду).

Скорегований внутрішнім декодером вектор ШК-знака перетворюють у число з діапазону $0 \div P - 1$ (порядковий номер ШК-знака в символіці ШК), для цього використовують, наприклад, табл. 3. 1 або табл. 3. 2. Результатом оброблення всіх ШК-знаків є масив n чисел з діапазону $0 \div P$ -1, який є зчитаним словом C' коду Ріда-Соломона. Це слово декодують за правилами коду Ріда-Соломона з метою виправлення помилок, які не вдалося виявити або виправити внутрішнім декодером. Порядкові номери “стертих” ШК-знаків (на рис. 3. 5 показано пунктиром) використовуються декодером коду Ріда-Соломона. ШК-знаки, які відповідають символу-заповнювачу (pad), що доповнюють ШК-позначку до прямокутної (квадратної) форми, не декодують; вони відкидаються.

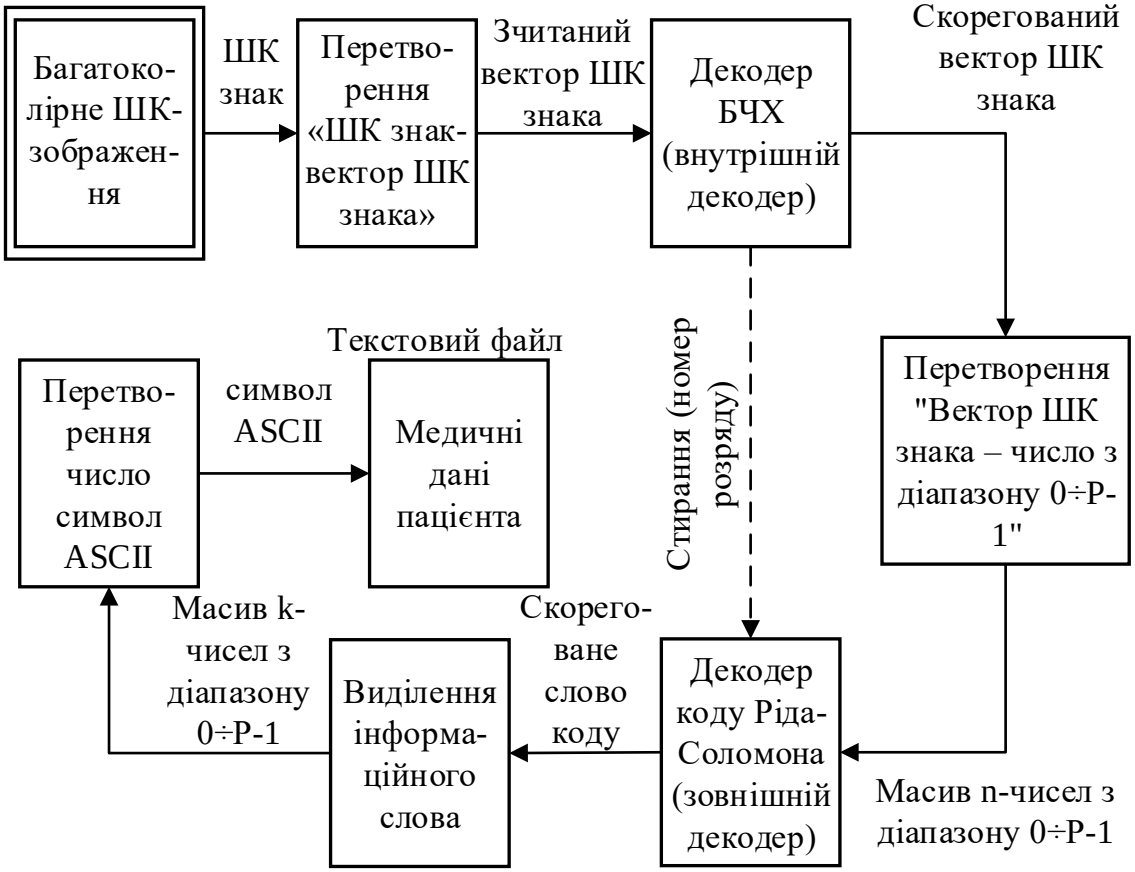


Рис. 3. 5. Схема відтворення медичних даних пацієнта
з багатоколірного штрихового зображення

Наприклад, результатом роботи декодера БЧХ можуть бути три висновки: “ШК-знак не ушкоджено” (якщо синдром помилки дорівнює нулю), “ШК-знак виправлено” (якщо декодер виправляє одно- або двократну помилку у векторі ШК-знака), “Стертий ШК-знак” (якщо декодер виявить спотворення трьох або більше елементів у ШК-знаку).

Докладніше процес декодування розглянемо на прикладі зчитування ШК-позначки, якій при її формуванні та нанесенні на носій відповідало кодове слово (3. 5). Нехай внаслідок ушкоджень зазнали спотворення чотири ШК-знаки, а саме: ШК-знаки c'_9 та c'_5 ідентифіковані як стирання; а c'_7 та c'_2 ідентифіковані як помилки:

$$\begin{array}{lcccccccccccccc}
\beta^l: & 7^{13} & 7^{12} & 7^{11} & 7^{10} & 7^9 & 7^8 & 7^7 & 7^6 & 7^5 & 7^4 & 7^3 & 7^2 & 7^1 & 7^0 \\
C' = & (31 & 31 & 10 & 19 & \mathbf{0} & 17 & \underline{17} & 14 & \mathbf{0} & 26 & 4 & \underline{18} & 6 & 2) \\
& c'_{13} & c'_{12} & c'_{11} & c'_{10} & c'_9 & c'_8 & c'_7 & c'_6 & c'_5 & c'_4 & c'_3 & c'_2 & c'_1 & c'_0
\end{array} \qquad (3.\ 8)$$

Для зручності декодування стирання замінені декодером коду Ріда-Соломона на нульові значення (це загальновизнаний прийом), які виділено в (3. 8) жирним, а розряди, що містять помилки, підкреслено. Кожному розряду слова C' ставлять у відповідність локатор, який подають або у степеневому вигляді (β^l), або у вигляді десяткового числа $L=(\beta^l) mod\ P$.

Декодування провадять в такій послідовності.

1. Обробляючи слово C' (3.8) справа наліво декодер фіксує номери розрядів $\varepsilon_1=5$ та $\varepsilon_2=9$, що містять стирання, а також кількість стирань $\rho=2$. Перевіряють відношення $\rho \le r$. Якщо $\rho > r$, то кількість стирань перевищує коректувальну здатність коду, і декодування слід припинити через невиправні спотворення. Якщо $\rho \le r$, то знаходять локатори стирань $L_{\varepsilon_1} = 7^5 = 38, \quad L_{\varepsilon_2} = 7^9 = 13$ (за mod 41).

   Якщо стирань не виявлено, то $\rho=0$.

2. Знаходять многочлен локаторів стирань

$$\lambda(x) = \prod_{i=1}^{\rho} (1 - L_{\varepsilon_i} x) = (1-38x)(1-13x) = 2x^2 + 31x + 1.$$

   Якщо $\rho=0$, то $\lambda(x)=1$.

3. Визначають, яку максимальну кількість помилок t ще може виправити декодер після виявлення ρ стирань: $t=\text{int}((r - \rho)/2)$.

   Наявна кількість μ помилок у зчитаному слові має задовольняти умову $\mu \le t$.

   Оскільки $r=6$, то максимальна кількість t помилок, яку може виправити декодер після виявлення двох стирань, становить $t=\text{int}((6 - 2)/2)=2$.

4. Обчислюють компоненти $\Psi_1, \Psi_2, \ldots, \Psi_r$ синдрому спотворень $\Psi=(\Psi_1\Psi_2 \ldots \Psi r)$: $\Psi_w=c'(\beta^w)$, $w=1, 2, \ldots, r$,

де $c'(x)=\sum_{i=0}^{n-1}c'_i x^i$; c'_i – розряди зчитаного слова. Якщо всі компоненти синдрому спотворень Ψ дорівнюють нулю, то приймають рішення про відсутність спотворень у зчитаному слові.

$\Psi_1=c'(7^1)=$

$=31(7^1)^{13}+31(7^1)^{12}+10(7^1)^{11}+19(7^1)^{10}+17(7^1)^8+17(7^1)^7+14(7^1)^6+26(7^1)^4+$

$4(7^1)^3+18(7^1)^2+6(7^1)^1+2=37 \pmod{41}$;

$\Psi_2= c'(7^2)=35$; $\Psi_3= c'(7^3)=23$; $\Psi_4= c'(7^4)=11$; $\Psi_5= c'(7^5)=38$; $\Psi_6= c'(7^6)=5$.

Оскільки $\Psi\neq 0$ – слово C' містить спотворені розряди.

5. Утворюють синдромний многочлен

$$\Psi(x)=1+\sum_{w=1}^{r}\Psi_w x^w=1+37x+35x^2+23x^3+11x^4+38x^5+5x^6.$$

6. Зі співвідношення $\Psi(x)\lambda(x)\equiv V(x)(\mathrm{mod}\,x^{r+1})$ знаходять модифікований синдромний многочлен спотворень, де $V(x)=1+\sum_{w=1}^{r}V_w x^w$, а V_1, V_2, …, V_r – компоненти модифікованого синдрому спотворень:

$$(1+37x+35x^2+23x^3+11x^4+38x^5+5x^6)(2x^2+31x+1)(\mathrm{mod}\,x^7)=$$
$$=1+27x+36x^2+34x^3+15x^4+15x^5+16x^6.$$

Компоненти модифікованого синдрому спотворень дорівнюють:

$V_1=27$, $V_2=36$, $V_3=34$, $V_4=15$, $V_5=15$, $V_6=16$.

7. Знаходять синдром помилок. Для цього з послідовності V_1, V_2, …, V_r виділяють підпослідовність V_j, V_{j+1}, …, V_r, де $j=r-2t+1$, елементи якої є компонентами синдрому помилок: $D_1=V_j$, $D_2=V_{j+1}$, …, $D_{2t}=V_r$.

Оскільки $j=6-2\cdot 2+1=3$, то компонентами синдрому помилок є: $D_1=V_3=34$, $D_2=V_4=15$, $D_3=V_5=15$, $D_4=V_6=16$.

8. Знаходять многочлен локаторів помилок у вигляді $\eta(x)=1+\sum_{j=1}^{\mu}\eta_j x^i$; коефіцієнти η_j обчислюють на основі компонент D синдрому помилок за алгоритмом Берлекемпа-Мессі [16] (Додаток Є).

Знаючи $D_1 \ldots D_4$ за цим алгоритмом знайдемо многочлен локаторів помилок: $\eta(x) = 1 + 16x + 13x^2$ (див. Додаток Є).

9. Розв'язують рівняння $\eta(x) = 0$ в полі *GF(P)* та знаходять корені $x_1, x_2, \ldots, x_\mu$, а на їх основі – локатори помилок: $L_{e_1},\ L_{e_1}, \ldots, L_{e_\mu}$.

   Рівняння $1 + 16x + 13x^2$ в $GF(41)$ має корені $x_1=29$, $x_2=36$. Отже, локаторами помилок є: $L_{e_1} = x_1^{-1} = 17,\ L_{e_2} = x_2^{-1} = 8.$

10. Знаходять многочлен значень спотворень $Q(x)$ степеня r зі співвідношення $V(x)\eta(x) \equiv Q(x) (\mathrm{mod}\, x^{r+1})$:

    $$Q(x) = V(x)\eta(x) (\mathrm{mod}\, x^7) = (1 + 27x + 36x^2 + 34x^3 + 15x^4 + 15x^5 + 16x^6) \cdot$$
    $$\cdot (1 + 16x + 13x^2) (\mathrm{mod}\, x^7) = 2x^4 + 18x^3 + 30x^2 + 2x + 1.$$

11. Формують спільну множину L=$\{L_1, L_2, \ldots, L_{\rho+\mu}\}$ локаторів спотворень, до якої входять локатори стирань (L_ε) та локатори помилок (L_e), впорядковуючи її за зростанням значень елементів, кількість яких становить $\rho+\mu$.

    У нашому випадку $L = \{L_{e_2} = 8,\ L_{e_1} = 13,\ L_{\varepsilon_1} = 17,\ L_{\varepsilon_2} = 38\} = \{8, 13, 17, 38\}$.

12. Обчислюють величини E_γ спотворень, тобто як стирань, так і помилок, за формулою Форні [41] $E_\gamma = \dfrac{Q(L_\gamma^{-1})}{\prod\limits_{\gamma=1,\, j=1, \gamma \neq j}^{\rho+\mu} (1 - L_\gamma^{-1} L_j)}$.

    Для елемента $L_1=8$ множини L знайдемо значення E_1:

    $E_1 = \dfrac{Q(8^{-1})}{(1 - 8^{-1} \cdot 13)(1 - 8^{-1} \cdot 17)(1 - 8^{-1} \cdot 38)} = 16$. За аналогією знаходимо значення E для решти елементів множини L: $E_2=19$, $E_3=20$, $E_4=16$.

13. Виправляють спотворення. Для кожної пари (L_γ, E_γ) подають L_γ у вигляді $L_\gamma = \beta^f$ (див. (3. 8) та виконують корекцію у розряді f прийнятого слова C' як $c_f = (c'_f - E_\gamma) mod\ P$:

    (L_1, E_1)=(8, 9)=(7^2, 9),

    (L_2, E_2)=(13, 19)=(7^9, 19),

$(L_3, E_3)=(17, 20)=(7^7, 20)$,

$(L_4, E_4)=(38, 16)=(7^5, 16)$.

Зокрема,

$c_2=(c'_2 - E_1) \bmod 41=(18 - 9) \bmod 41=9$

$c_9=(c'_9 - E_2) \bmod 41=(0 - 19) \bmod 41=22$

$c_7=(c'_7 - E_3) \bmod 41=(17 - 20) \bmod 41=38$

$c_5=(c'_5 - E_4) \bmod 41=(0 - 16) \bmod 41=25$.

Отже, всі чотири спотворення у прийнятному слові C' (3. 8) виправлено, внаслідок чого отримано кодове слово C (3. 5).

Кодовому слову (3.5) відповідає поліном

$c(x)=31x^{13}+31x^{12}+10x^{11}+19x^{10}+22x^9+17x^8+38x^7+14x^6+25x^5+26x^4+4x^3+$

$+9x^2+6x+2$.

Для знаходження інформаційного полінома $a(x)$ необхідно поліном $c(x)$ поділити на твірний поліном $\varphi(x)$ коду Ріда-Соломона:

$a(x) = c(x)/\varphi(x)=$

$(31x^{13}+31x^{12}+10x^{11}+19x^{10}+22x^9+17x^8+38x^7+14x^6+25x^5+26x^4+4x^3+9x^2+6x+2)/$

$(x^6+12x^5+8x^4+34x^3+13x^2+24x+34) =31x^7+28x^6+12x^4+40x^3+14x^2+40x+29$.

Коефіцієнти полінома $a(x)$ утворюють інформаційне слово

A = 31 28 0 12 40 14 40 29,

яке перетворюють в послідовність символів штрихового коду:

5 2 A M setC & setB Ю, і на основі якого відтворюють первісне повідомлення: 52AM&Ю.

Наведена послідовність дій забезпечує відтворення текстових даних (медичних даних пацієнта) з багатоколірного штрихкодового зображення.

Алгоритм відтворення медичних даних пацієнта з багатоколірного штрихкодового зображення реалізовано у вигляді програмної процедури (Додаток Н).

3. 4. Метод підвищення завадостійкості багатоколірних штрихкодових зображень на основі дворівневого контролю ушкоджень

В існуючих чорно-білих ШК застосовують однорівневу систему забезпечення завадостійкості – ШК-позначку, що є масивом ШК-знаків, кодують коректувальним кодом, здатним виправляти багатократні ушкодження, зазвичай кодом Ріда-Соломона [20, 21].

З іншого боку, з наукових публікацій, наприклад, [18, 19, 20, 21], відомо, що багатоколірне штрихове кодування дозволяє у декілька разів підвищити інформаційну щільність подання даних за незмінних геометричних розмірів штрихкодових елементів, однак при цьому ускладнюються процеси розпізнавання та декодування багатоколірних штрихкодових зображень. Зокрема, може істотно зростати кількість помилок при декодуванні колірного зображення.

Тому, в даному науковому дослідженні пропонується застосовувати дворівневу систему забезпечення завадостійкості багатоколірних штрихкодових зображень, яка грунтується на виправленні ушкоджень (спотворень) штрихкодових елементів математичним способом – на основі теорії завадостійкого кодування.

В інформаційному сенсі структуру багатоколірної штрихкодової позначки можна вважати дворівневою. ШК-знак є не лише мінімальною структурною, але й інформаційною одиницею. А матрицю ШК-знаків штрихкодової позначки можна розглядати як єдине многозначне багаторозрядне слово C (рис. 3. 6). Відповідно, завадостійкість також можна запровадити на двох рівнях – на рівні ШК-знаків (на рівні кожного розряду c_i слова C) та на рівні усієї матриці ШК-знаків (на рівні слова C у цілому).

ШК-знаки є розрядами слова C.

На рівні ШК-знаків для забезпечення їх завадостійкості слід застосовувати многозначний коректувальний код з коректувальною здатністю t=1 (код Хемінга) або t=2 (код БХЧ), а на рівні усієї ШК-позначки – коректувальний

код Ріда-Соломона (рис. 3.7), який здатний виправляти багатократні ушкодження (спотворення) двох видів – помилки і стирання.

Як уже зазначалося, у коді Ріда-Соломона «помилкою» вважається ситуація, коли не відоме ані місце спотворення у зчитаному слові (ШК-позначці), а ні величина спотворення; а «стиранням» – ситуація, коли місце спотворення відоме, а не відомою є лише величина спотворення.

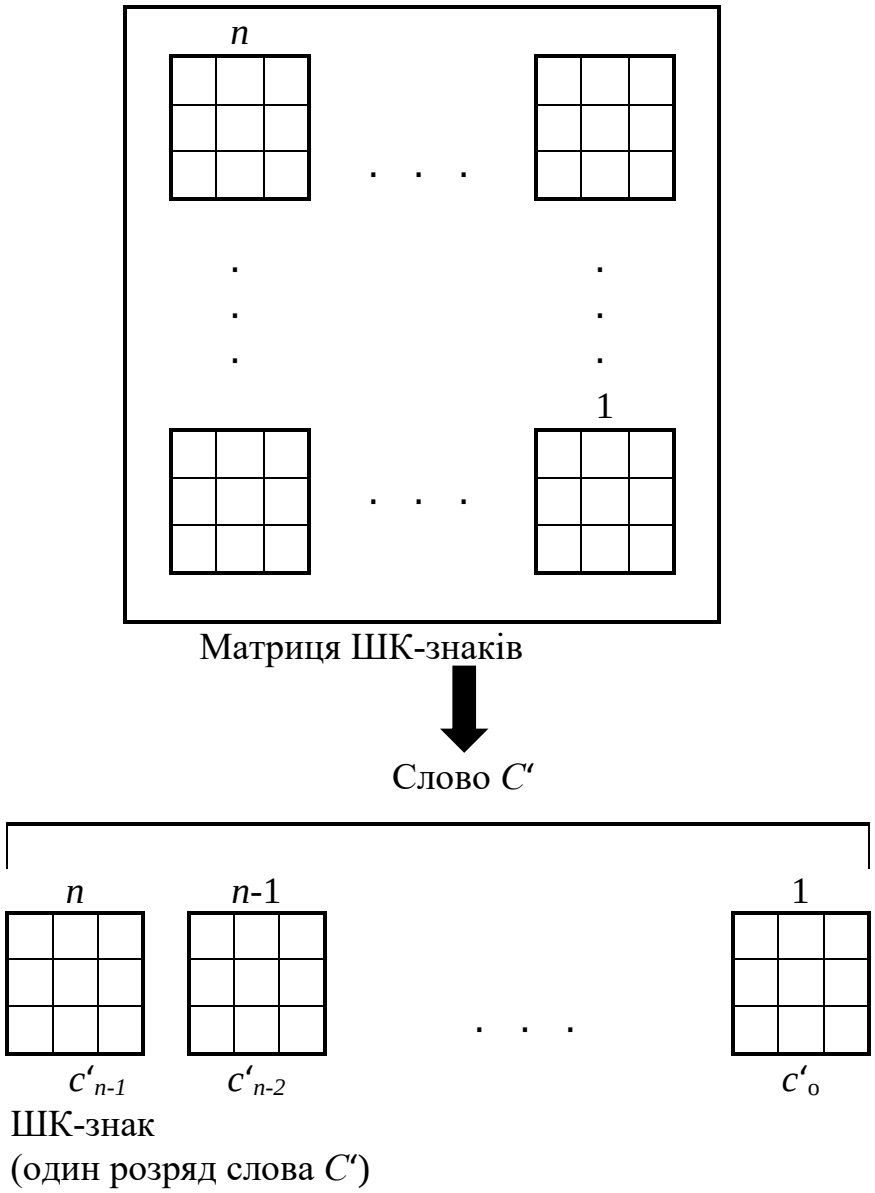


Рис. 3. 6. Інформаційна структура багатоколірної штрихкодової позначки

Особливістю застосування многозначного коректувального коду на нижньому рівні контролю ушкоджень є те, що цифровий еквівалент (вектор) багатоколірного штрихкодового знака має бути кодовим словом цього коректувального коду (Хемінга, БЧХ). Наділення ШК-знаків такою властивістю відбувається під час побудови (синтезу) символіки багатоколірного штрихового коду. Важливим при цьому є те, щоб коректувальний код нижнього рівня, крім виправлення 1- або 2-кратних ушкоджень (відповідно до його коректувальної здатності), виявляв би також частину багатократних ушкоджень. Виконані дослідження показали, що властивість виявляти багатократні ушкодження, кратність яких перевищує коректувальну здатність коду, мають многозначні скорочені коректувальні коди – Хемінга та БЧХ (табл. 3.3). А коди БЧХ мають таку властивість навіть у випадку повного коду.

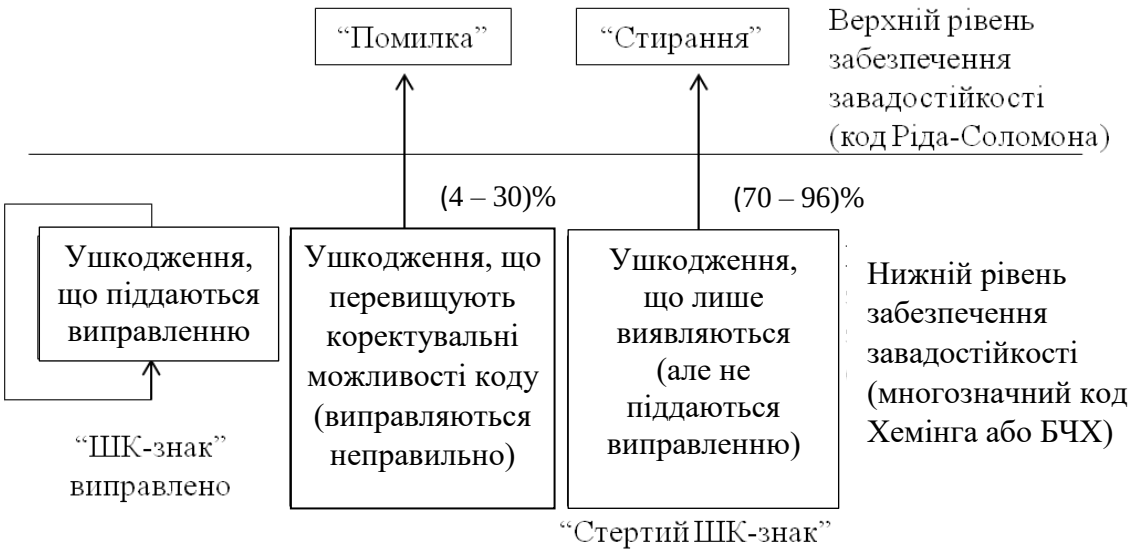


Рис. 3. 7. Структура дворівневої системи забезпечення завадостійкості штрихкодових зображень

Властивість виявляти багатократні ушкодження (спотворення) на нижньому рівні контролю (на рівні ШК-знаків) є визначальною, оскільки саме через цей параметр – виявлення, здійснюється зв'язок з верхнім рівнем контролю спотворень. Факт виявлення ушкодження на нижньому рівні

контролю сприймається кодом Ріда-Соломона як стирання. Це пояснюється тим, що оброблення ШК-знаків відбувається послідовно, а отже, працює внутрішній лічильник, який показує номер оброблюваного ШК-знака. Якщо в ШК-знаку виявлено багатократне спотворення (величина якого не відома), то на верхній рівень контролю (в декодер Ріда-Соломона) надходить порядковий номер цього ШК-знака, тобто його місцезнаходження у слові *C*, а це кваліфікується декодером Ріда-Соломона як стирання.

Таблиця 3.3.

Відсоток багатократних ушкоджень, які виявляються скороченими многозначними коректувальними кодами

Вид многозначного коректувального коду	Коди Хемінга (Див. Додаток А)	Коди БЧХ (Див. Додаток Б)
Трійкові коди	6,8% − 60,0%	29,5% − 94,1%
Четвіркові коди	11,1% − 70,8%	70,4% − 96,6%
П’ятіркові коди	12,5%- − 92,5%	88,5% − 96,0%
Сімкові коди	11,1% − 66,7%	97,5% − 99,0%

На нижньому рівні контролю ушкоджень у межах кожного ШК-знака працює внутрішній програмний декодер (Хемінга або БЧХ), результатом роботи якого є три висновки: “ШК-знак не ушкоджено (цей випадок на рис. 3.7 не показано)”, “ШК-знак виправлено”, “Стертий ШК-знак”.

Наприклад, у випадку застосування на нижньому рівні забезпечення завадостійкості (рівень ШК-знаків) четвіркових кодів БЧХ, на верхньому рівні контролю багатократні ушкодження у (70 −96)% випадків сприйматимуться як стирання, а у (4 − 30)% як помилки (див. рис. 3. 7).

Ефект підвищення завадостійкості досягається за рахунок передачі на верхній рівень контролю ушкоджень ситуацій “стирання” замість ситуацій “помилка”, тому що на усунення наслідків “стирань” у коді Ріда-Соломона витрачається удвічі менший ресурс (один контрольний розряд на одне “стирання”; а на усунення наслідків кожної “помилки” витрачаються два контрольні рядки (див. співвідношення (3. 4).

Нехай при створенні ШК-позначки багатоколірного ШК великої ємності (декілька тисяч ШК-знаків) для захисту даних застосовано лише верхній рівень забезпечення завадостійкості на основі коду Ріда-Соломона, причому ставиться вимога виправляти до 100 ушкоджених ШК-знаків. Для цього в кодовому слові *C* коду Ріда-Соломона має бути передбачено 200 контрольних розрядів.

Якщо ж захист цілісності даних здійснювати на основі дворівневої системи забезпечення завадостійкості – на нижньому рівні застосовувати, наприклад, один з четвіркових кодів БЧХ, а на верхньому рівні – код Ріда-Соломона, то зі ста можливих ушкоджених ШК-знаків приблизно 70 будуть ідентифіковані як “стертий ШК-знак” (тобто “стирання”), а решта 30 – як “помилка” (відповідно до отриманих статистичних даних з табл. 3. 3). Для виправлення 70-ти стирань необхідно 70 контрольних розрядів, а для виправлення 30-ти помилок – 60 контрольних розрядів у кодовому слові *C*, отже, загалом 130 контрольних розрядів. Але оскільки в кодове слово коду Ріда-Соломона за умовою закладено 200 контрольних розрядів, то, отже, 70 контрольних розрядів можна ніби «вивільнити». Ці «вивільнені» контрольні розряди дозволяють локалізувати та виправити додатково ще 35 ушкоджених ШК-знаків. Таким чином, можливості коду Ріда-Соломона будуть підсилені на (135 - 100)/100=35%.

Звідси випливає й інший висновок. При дворівневій системі забезпечення завадостійкості штрихкодових даних у коді Ріда-Соломона достатньо мати не 200, а лише 200 - 70=130 контрольних розрядів, щоб

забезпечити таку саму завадостійкість, і, отже, на 35% знизити надлишковість коду.

Виконані в роботі дослідження показують також, що застосування на нижньому рівні контролю ушкоджень многозначного неповного коду Хемінга, а на верхньому – коду Ріда-Соломона, в середньому на (12 - 25)% підвищує завадостійкість багатоколірних штрихкодових зображень.

3.5. Висновки до третього розділу

Дослідження процедур забезпечення завадостійкості багатоколірних штрихкодових зображень, що можуть застосовуватись у медичних інформаційних системах, дозволяє зробити наступні висновки.

1. Показано, що для надійного зчитування (сканування) багатоколірних штрихових кодів з використанням мобільних пристроїв слід застосовувати дворівневу систему забезпечення завадостійкості багатоколірних штрихових зображень - на рівні штрихкодових знаків (нижній рівень) та рівні усієї штрихкодової позначки (верхній рівень).
2. На нижньому рівні забезпечення завадостійкості доцільно застосовувати многозначний коректувальний код, який здатний виправляти одно- або двократні помилки та виявляти частину помилок більшої кратності. На верхньому рівні забезпечення завадостійкості слід застосовувати коректувальний код, що здатний виправляти багатократні спотворення двох видів – помилки та стирання. Такі властивості має код Ріда-Соломона. Зв'язок коректувального коду нижнього рівня та коректувального коду верхнього рівня здійснюється через параметр виявлення багатократної помилки в межах штрихкодовогог знака. Систему дворівневого контролю ушкоджень (спотворень) організовано так, що факт виявлення багатократної помилки на нижньому рівні (у ШК-знаку) сприймається на верхньому рівні контролю як стирання.

3. Запропоновано метод підвищення завадостійкості багатоколірних штрихових кодів, який грунтується на застосуванні дворівневого контролю спотворень (ушкоджень) з використанням двох многозначних коректувальних кодів: коду, який виправляє одно- або двократні спотворення (ушкодження) у межах ШК-знаку (нижній рівень), та коду Ріда-Соломона – на рівні усієї ШК-позначки.

   Виконані дослідження показують, що при використанні на нижньому рівні контролю многозначного коду Хемінга завадостійкість штрихкодових зображень поліпшується на (12 - 25)%, а при застосуванні на нижньому рівні контролю многозначного коду БЧХ з мінімальною кодовою відстанню 5 вона поліпшується на (35 - 45)%.
4. Доведено, що запропонований алгоритмічно-програмний підхід до забезпечення завадостійкості багатоколірних штрихових кодів на основі дворівневого контролю спотворень (ушкоджень) істотно підвищує рівень завадостійкості штрихкодових зображень та гарантує цілісність зчитуваних даних.

РОЗДІЛ 4. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОЦЕСІВ АВТОМАТИЧНОЇ ІДЕНТИФІКАЦІЇ НА ОСНОВІ БАГАТОКОЛІРНИХ ЗАВАДОСТІЙКИХ ШТРИХОВИХ КОДІВ У МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

4. 1. Архітектура програмної системи автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів

У медичній інформаційній системі оброблення інформації починається з етапу введення даних про одиницю обліку, якою може бути будь-який запис (файл): персональні дані пацієнта, діагноз, результат певного дослідження, рецепт, припис лікаря, дозування препаратів тощо. Первинне введення текстової інформації про одиницю обліку (запис, файл) здійснюється працівниками медичної установи – реєстратура, лікар, медсестра, лаборант та ін., шляхом ручного набору текстових даних з клавіатури. Проте на наступних етапах оброблення та передачі інформації пропонується застосовувати технологію машиночитаного подання даних, зокрема технологію штрихового кодування [84].

Виходячи з вищезазначених функціональних вимог (табл. 1. 3) доходимо висновку, що основними процесами, які мають підтримуватись та забезпечуватись програмним забезпеченням медичної інформаційної системи на основі застосування штрихових кодів, є наступні:

- формування запису про особу пацієнта;
- створення і оброблення записів про відвідування пацієнтом лікаря;
- формування записів результатів лабораторних досліджень;
- створення та оброблення записів про встановлений діагноз;
- генерування записів про формування доз медичних препаратів;
- завадостійке кодування даних з використанням багатоколірного штрихового коду;

- автоматичне зчитування штрихкодових позначок (ШК-позначок) з екрана комп’ютера або смартфона, а також з паперового носія;
- надання доступу до медичних даних пацієнта на основі сканування штрихового коду зі смартфона пацієнта;
- розподілене зберігання медичних даних пацієнта.

У розроблюваній системі пропонується два рівні забезпечення завадостійкості штрихкодового зображення: на верхньому рівні – на основі коду Ріда-Соломона, а на нижньому – на основі многозначного коду БЧХ або многозначного коду Хемінга. Многозначність визначається кількістю кольорів, які використовуються для забарвлення елементів штрихкодового зображення; найдоцільніше використовувати 3-5 кольорів. Багатоколірність штрихкодового зображення підтримується сучасними комп’ютерами та смартфонами. Багатоколірний ШК у кілька разів підвищує інформаційну щільність подання даних порівняно з чорно-білими ШК без зміни геометричних розмірів елементів зображення [30 - 37].

Наприклад, чотириколірний ШК підвищує інформаційну щільність удвічі.

Носієм багатоколірного ШК в розроблюваній системі може бути екран комп’ютера (ноутбука, планшета), екран смартфона та паперова штрихкодова наліпка. Штрихкодові зображення передбачається зчитувати за допомогою камери смартфона.

Особливістю системи є й те, що з метою забезпечення анонімності пропонується також передача інформації зі смартфона на смартфон без залучення мережі або Bluetooth – штриховий код з екрана одного смартфона зчитують камерою іншого смартфона (своєрідна передача інформації з рук у руки).

Для забезпечення надійності такої передачі даних слід застосовувати саме завадостійкий ШК, який би забезпечував цілісність даних в умовах можливого спотворення штрихкодового зображення (зміна освітлення, кута

сканування, тремтіння руки оператора, розмитість або перекошування зображення тощо).

Передбачається, що з системою працюють чотири категорії користувачів – медсестра, лаборант, лікар, пацієнт.

Однією з головних особливостей інформаційних систем в медичній галузі є підвищена увага до захисту особистих та медичних даних пацієнтів. З огляду на це архітектура розроблюваної програмної системи має базуватись на таких принципах: розподіленість, анонімізація, володіння даними.

Принцип розподіленості полягає у необхідності розділення даних про окремо взятого пацієнта таким чином, щоб навіть у випадку компрометації деякої частини даних зловмисник не міг встановити особу пацієнта на основі отриманих даних, а також встановити, що окремі фрагменти даних належать одному й тому самому пацієнту. Даний принцип пропонується реалізувати за допомогою децентралізованого зберігання інформації про пацієнтів у різних підсистемах. Систему зв'язків між даними у цих підсистемах пропонується зберігати у захищеній базі даних, доступ до якої суворо регламентується і піддається аудиту (рис. 4. 1).

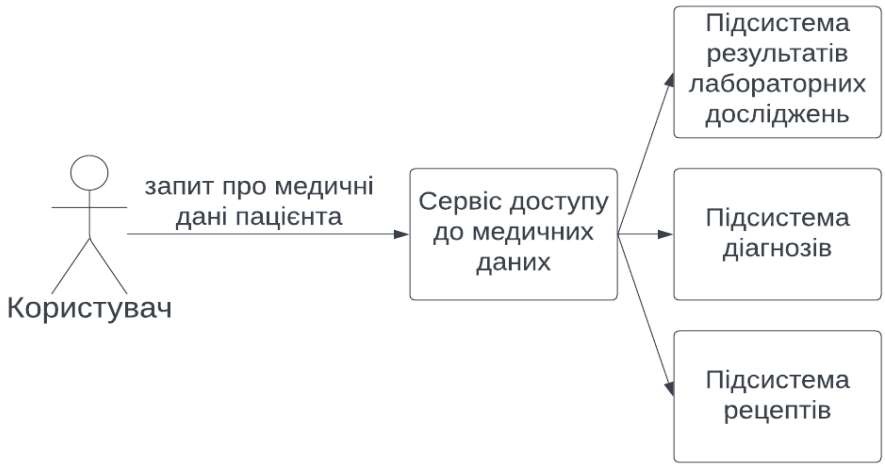


Рис. 4. 1. Розподілене зберігання медичних даних у підсистемах медичної інформаційної системи

Принцип анонімізації передбачає відсутність у медичних даних будь-якої інформації, що ідентифікує конкретного пацієнта. Наприклад, медсестра при формуванні доз медичних препаратів не повинна мати доступу до діагнозу чи особистих даних пацієнта. Більше того, з метою посилення анонімізації пропонується розміщувати на посудині з дозованим медпрепаратом наліпку з ШК так, щоб лише медсестра та пацієнт могли зчитати його вміст за допомогою смартфона. Це мінімізує помилки медперсоналу; а пацієнт додатково перевірить, чи це адресовано саме йому, та за допомогою ШК отримає докладну інструкцію щодо вживання препарату (штрихкодова взаємодія медсестри та пацієнта). Зчитуючи смартфоном той самий ШК пацієнт отримує доступ до інформації, що стосується його особисто, а медсестра – до інформації, що стосується її як медпрацівника.

Принцип володіння даними означає, що доступ і обробка медичних даних пацієнта можливі лише за його безпосередньої згоди.

На основі функціональних вимог, викладених принципів, а також зазначених процесів розроблено діаграму компонентів програмної системи (рис. 4.2).

Розглянемо взаємодію користувачів системи.

Медсестра формує дози препаратів для пацієнтів. На екрані свого смартфона або в додатку на ПК вона бачить інформацію про дози лікарських засобів, які необхідно сформувати, ідентифікатори пацієнтів, номери палат куди їх потрібно доставити тощо. Після закінчення формування доз препаратів, медсестра друкує наліпку з ШК, в якому закодована інформація про найменування ліків, пацієнта, якому вони призначені тощо. Пацієнт, отримавши препарати з штрихкодовою наліпкою, може впевнитися, що це призначено саме йому, зчитавши ШК з наліпки за допомогою свого смартфона.

Лаборант формує результати аналізів, знаючи ідентифікатор пацієнта. Він вносить результати в систему за допомогою програми на стаціонарному комп’ютері.

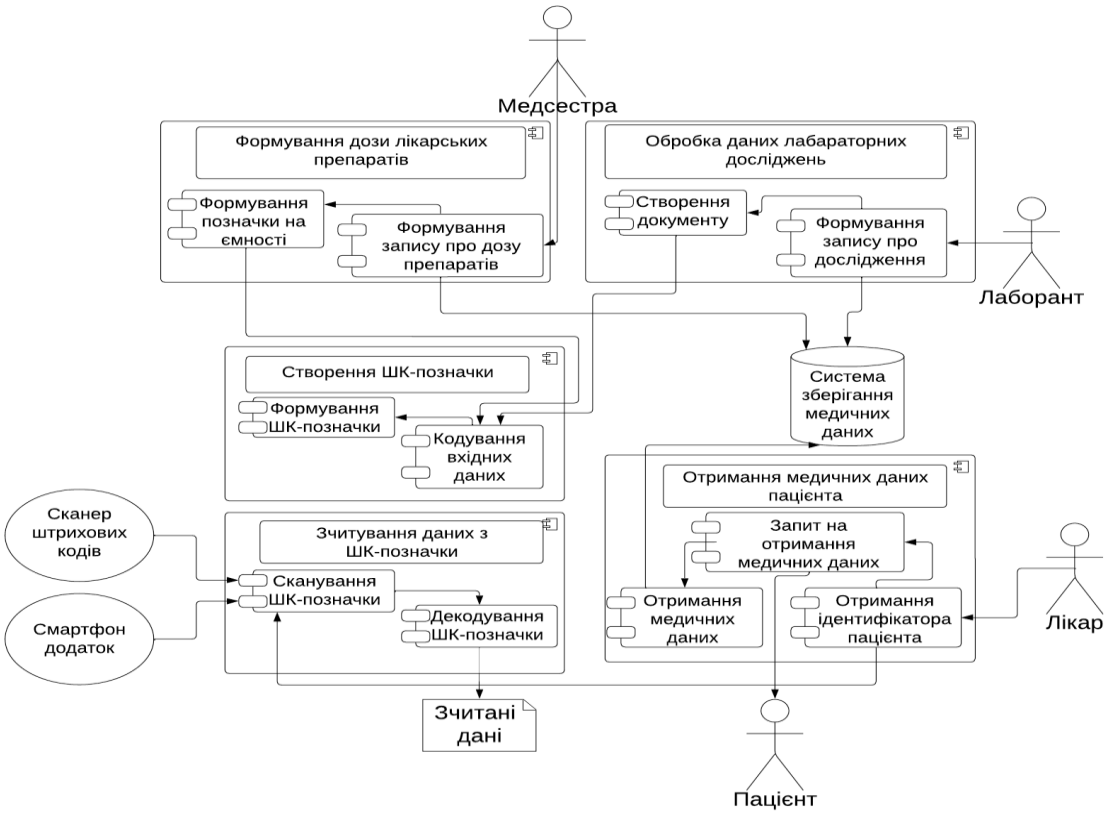


Рис. 4. 2. Діаграма компонентів програмної системи

Система зберігає результати аналізів в базі даних, а також генерує ШК із закодованим ідентифікатором пацієнта, токеном доступу та посиланням для завантаження даних аналізів. Інформація кодується у форматі JSON (Javascript Object Notation) (рис. 4. 3).

Токен доступу являє собою дані, зашифровані сервером. Для отримання даних аналізів за посиланням додаток пацієнта повинен відправити запит на сервер, приклавши до нього цей токен. Так сервер може визначити, що дані запитуються верифікованим клієнтом. Токен кодується у форматі JWT (Javascript Web Token). Отже, після того як лаборант вніс результати в систему і натиснув на кнопку “зберегти”, згенерований ШК надсилається електронним листом пацієнту (рис. 4. 4).

```
{
"analysisId": "5a8c1b0e-3b9a-4f3c-8e2a-6f0f8d1f8d9c",
"patientId": "4b8a2b0f-4491-4ffa-9ff1-7f0128d678d9a",
"organizationId": "5a53dcb0a-4b11-31ed-8e2a-6f05faac1284",
"accessToken":"bt3g30gkic8cagpvvdsf4k50h507hnanq6ls7n89c06b4n05k0s3bhknri81sfjw67gfzynwjp
mblsrolhv91ptotexjitkybiovapszwiqo7qywaqpub8uj7vhb6gy25fwnqljmo7z3ozr52b2eho3o3dzfqgrn4c
h1x1ycx8itpfcnlm21grmymg2649q8ep18kn33geyv03ai25y37fnp3q6alipqr104ux6300xn4ktvusj0nwvx
bq0q7j77djmucssgi3csbrqmeef2f6dekx",
"analysisResultLink":"https://www.barcoderesearch.com/analysis",
"analysisTime": "2020-09-09T15:00:00.000Z",
"analysisType": "file"
}
```


Рис. 4. 3. Приклад кодування інформації про надання доступу до даних у форматі JSON

Після цього пацієнт за допомогою свого смартфона сканує штрихкод. Додаток на смартфоні декодує інформацію і надсилає HTTP запит на сервер МІС. Сервер повертає дані аналізів, вони відображаються на екрані смартфона пацієнта.

Рис. 4. 4. ШК-позначка для доступу пацієнта до результатів аналізу

Розглянемо взаємодію пацієнта з лікарем докладніше.

На прийомі в лікаря пацієнт може надати результати своїх аналізів лікарю. При цьому можна обрати один з двох режимів взаємодії – онлайн або

офлайн. У випадку офлайн режиму пацієнт генерує на екрані свого смартфона ШК, у якому зашифровані самі дані аналізів безпосередньо (а не посилання на них). Лікар зчитує ШК з екрана смартфона пацієнта своїм смартфоном, та після його декодування на екрані бачить текстовий файл – результати аналізів.

При виборі режиму онлайн у ШК пацієнта буде закодоване посилання на результати аналізів. У цьому випадку працює механізм надання доступу до даних пацієнта в розподіленій системі зберігання медичних даних пацієнтів та їх отримання (рис. 4. 5).

Наведемо кроки алгоритму, що реалізує такий механізм (рис. 4. 6).

1. Пацієнт виводить на екран свого смартфона штрихкод з його закодованим персональним ідентифікатором.
2. Лікар за допомогою свого смартфона сканує штрихкод з екрана смартфона пацієнта.
3. Додаток на смартфоні лікаря відправляє запит на отримання медичних даних пацієнта на сервер МІС.
4. Пацієнт на своєму смартфоні отримує нотифікацію про запит до своїх медичних даних і надає дозвіл.
5. Система ініціює запит до підсистеми зберігання медичних даних пацієнтів на формування тимчасового токена доступу.
6. Підсистема даних аналізів повертає токен системі доступу до медичних даних.
7. Система передає сформований токен на додаток лікаря.
8. Додаток лікаря запитує дані за токеном.
9. Додаток лікаря отримує запитувані дані.

Рис. 4. 5. Схема надання доступу до медичних даних пацієнта лікарю в онлайн режимі

Якщо пацієнт перебуває на віддалі, то він може надіслати свій ШК лікарю з використанням мережі. При цьому обробка ШК лікарем також може здійснюватися в двох режимах (онлайн чи офлайн) – за бажанням пацієнта.

У пропонованій системі одним з основних компонентів є штрихове кодування даних, тому слід сказати про ієрархію класів реалізації штрихового кодування/декодування, діаграма яких зображена на рис. 4. 8.

У даній ієрархії використовується шаблон проєктування “Стратегія”. За допомогою цього шаблону відбувається підключення конкретного коректувального коду.

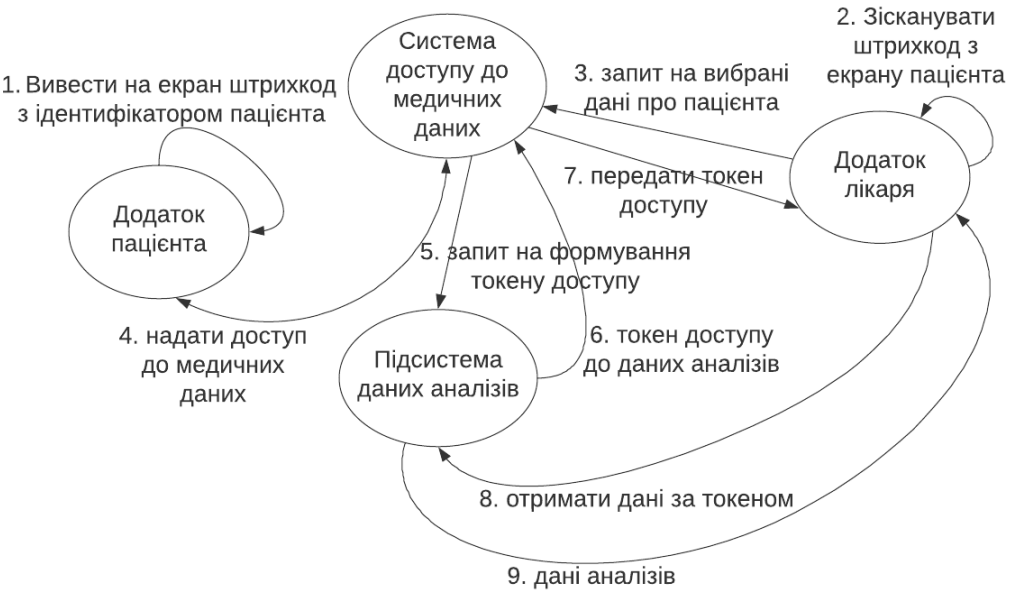


Рис. 4. 6. Граф надання доступу до медичних даних пацієнта лікарю

Наприклад, при використанні коду БЧХ до екземпляру класу BarcodeDecoder буде підключена стратегія BCHCorrectionStrategy. За необхідності підключення іншого коректувального коду достатньо буде розробити новий клас з реалізацією алгоритму обраного коду.

В результаті, була отримана архітектура програмної системи (рис. 4.8). Дана архітектура є базовою, вона містить набір компонентів, який дозволяє адаптувати її до потреб медичного закладу, а також спрощує процес розроблення програмного забезпечення для медичної галузі із застосуванням багатоколірного завадостійкого штрихового кодування даних.

Запропонована архітектура програмної системи є ядром медичної інформаційної системи, у якій забезпечується можливість для пацієнта володіти своїми медичними даними та надавати доступ до них тільки за власним бажанням та шляхом автоматичної ідентифікації на основі штрихового кодування інформації [84].

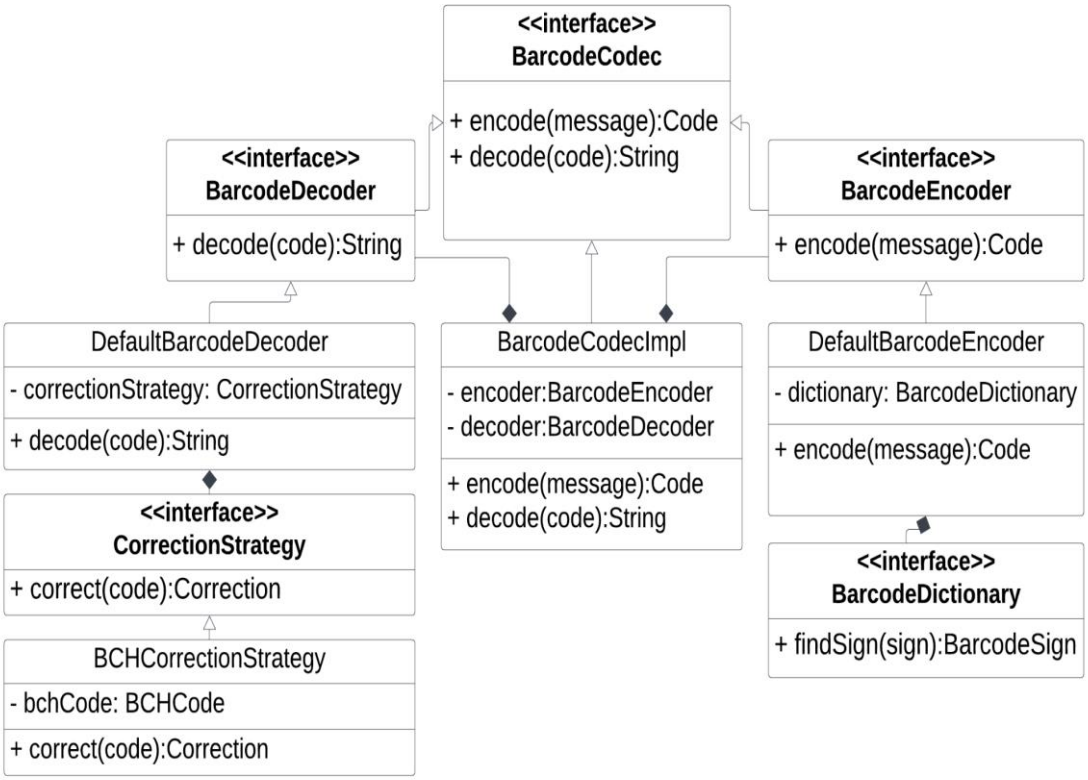


Рис. 4. 7. Діаграма класів кодування/декодування ШК-позначок

Реалізовані в системі принципи розподіленого зберігання медичних даних пацієнта, володіння ним своїми медичними даними та анонімності за рахунок передачі даних з рук у руки з допомогою зчитування штрихкодової інформації з одного смартфона іншим смартфоном дозволяють підвищити якість організації медичного обслуговування, істотно знизити вплив людського фактора в діяльності медперсоналу та посилити захист медичних даних пацієнта. Використання мобільних пристроїв паралельно зі штрихкодовим способом передачі та оброблення даних дозволяє забезпечити візуально захищену електронну взаємодію пацієнта з лікарем як безпосередньо, так і на відстані та гарантує високу надійність і конфіденційність обміну даними [84].

Базова архітектура програмної системи включає в себе універсальний набір модулів, що забезпечують реалізацію основних операцій в медичній інформаційній системі. Для розроблення конкретної програмної системи у

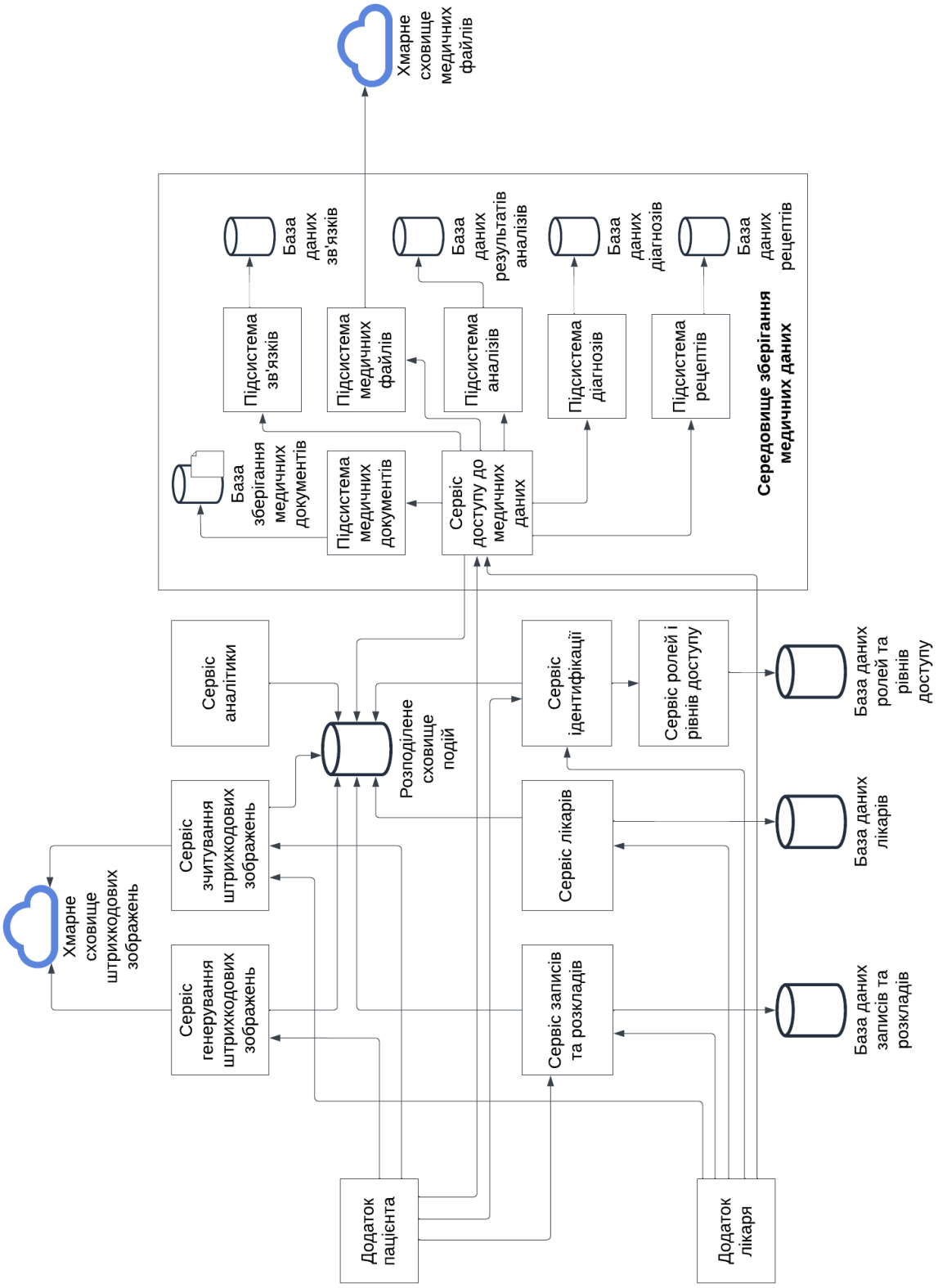


Рис. 4. 8. Архітектура програмної системи

медичній галузі достатньо адаптувати цей універсальний набір до потреб проєктованої інформаційної системи, що в кінцевому підсумку спрощує процес розроблення програмного забезпечення.

4. 2. Організація програмного модуля кодування-декодування багатоколірного завадостійкого штрихового коду

Основним компонентом програмної системи автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів є модуль кодування-декодування (Додаток Н). Головним завданням даного модуля є забезпечення процесу перетворення вхідної інформації у штрихкодову позначку, а також її декодування. У даному модулі можна виділити три складові частини:

1. Підмодуль кодування вхідної інформації.
2. Підмодуль формування зображення штрихкодової позначки.
3. Підмодуль декодування штрихкодової позначки.

1. Підмодуль кодування вхідної інформації

Розглянемо докладніше організацію підмодуля кодування вхідної інформації. У процесі кодування вхідної інформації можна виділити дві основні стадії: створення адаптованого повідомлення та створення матриці штрихкодових знаків.

На етапі створення адаптованого повідомлення відбувається низка перетворень вхідного повідомлення у послідовність елементів з таблиці символіки штрихового коду. Серед таких перетворень можна виділити наступні: доповнення символами перемикання між наборами, доповнення спеціальними символами-заповнювачами, додавання метаінформації, завадостійке кодування тощо.

Для програмної реалізації даного процесу доцільно виділити такі основні абстракції: “Стратегія кодування” (CodingStrategy), “Перетворювач штрихового коду” (BarcodeTransformer), “Стратегія доповнення” (PaddingStrategy). Відповідальністю абстракції “Стратегія кодування” є

власне формування адаптованого повідомлення. Цей процес складається з окремих етапів, що реалізуються за допомогою абстракції “Перетворювач штрихового коду”. Абстракцію “Стратегія кодування” доцільно реалізувати за допомогою шаблону “Ланцюг відповідальності”. Так окремі кроки процесу формування адаптованого повідомлення будуть поєднуватись у ланцюг перетворень. Таким чином, з деякої кількості таких кроків можна формувати різні стратегії адаптації повідомлення без дублювання логіки. Діаграма класів даного взаємозв’язку зображена на рис. 4. 9. Програмна реалізація послідовності перетворень вхідного повідомлення з використанням шаблону “Ланцюг відповідальності” наведена на рис. 4. 10.

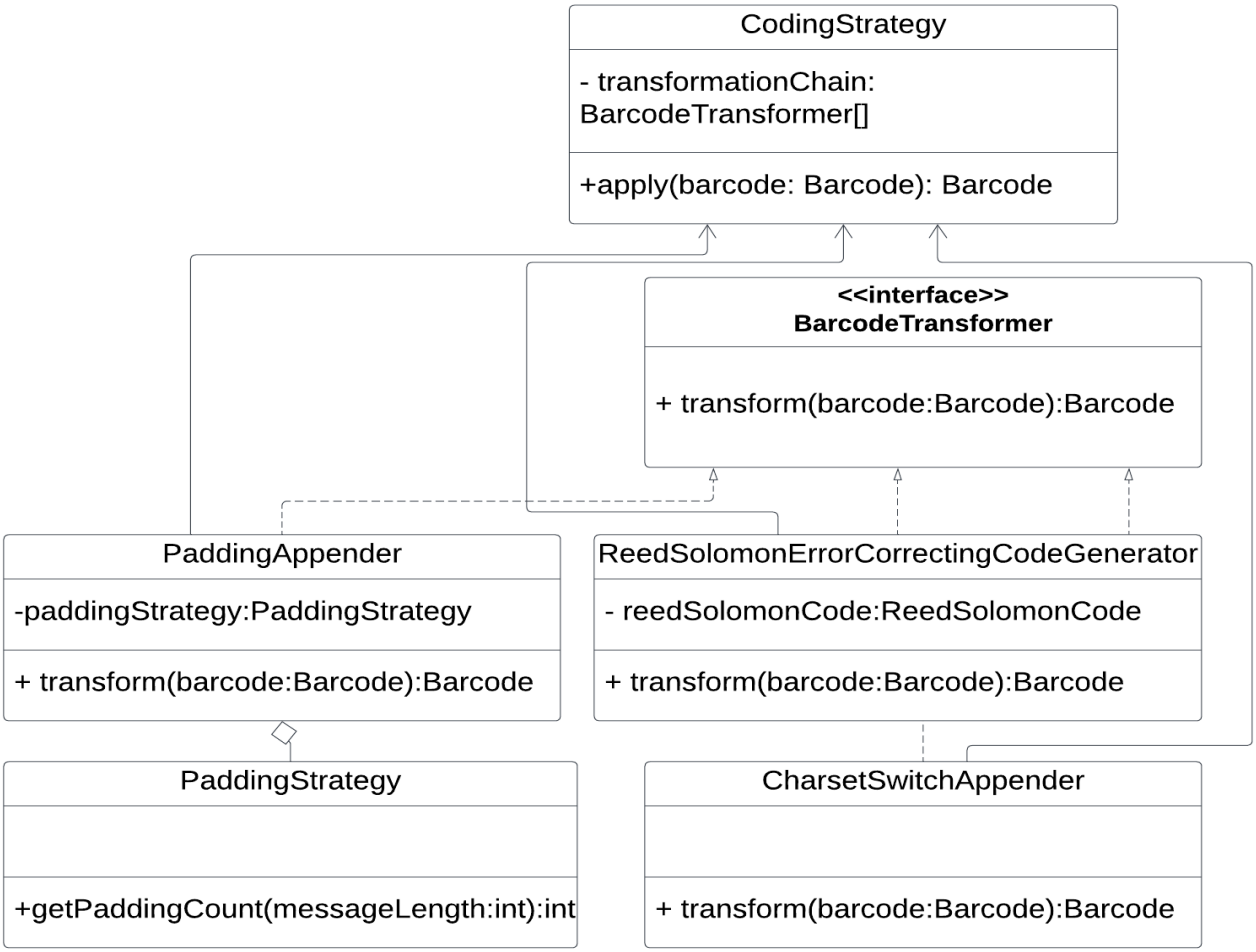


Рис. 4. 9. Діаграма класів шаблону “Ланцюг відповідальності” для реалізації кодування вхідного повідомлення

```
public class RSCodingStrategy implements CodingStrategy {
   private final ReedSolomonCode reedSolomonCode;
   public RSCodingStrategy(ReedSolomonCode reedSolomonCode) {
       this.reedSolomonCode = reedSolomonCode;
   }
   @Override
   public Barcode apply(Barcode code) {
       return Optional.of(code)
               .map(barcode -> new CharsetSwitchAppender().transform(barcode))
               .map(barcode -> new
PaddingAppender(paddingStrategy()).transform(barcode))
               .map(barcode -> new
ReedSolomonErrorCorrectingCodeGenerator(reedSolomonCode).transform(barcode))
               .orElseThrow();
   }
   private PaddingStrategy paddingStrategy() {
       return length -> new NearestSquarePaddingStrategy()
          .getPaddingCount(length + reedSolomonCode.getControlDigitsCount());
   }
}
```

Рис. 4. 10. Лістинг реалізації перетворення вхідного повідомлення за допомогою шаблону “Ланцюг відповідальності”

Наступним кроком після адаптації вхідного повідомлення є створення матриці штрихкодових знаків. Кожному елементу з символіки ШК позначки поставлено у відповідність код (вектор), на основі якого виконується створення ШК-знака. Даний код (вектор) має коректувальні властивості. Таким чином, процес створення матриці штрихкодових знаків є перетворенням символів повідомлення у цифрові коди (вектори) ШК- знаків.

З точки зори проєктування програмної системи, доцільно інкапсулювати адаптацію вхідного повідомлення, а також його перетворення в матрицю ШК-знаків в абстракцію з назвою “Кодувальник” (Encoder). Дану абстракцію доцільно реалізувати з використанням шаблону проєктування “Стратегія”. Так абстракція “Кодувальник” перебуває у відношенні композиції з абстракціями

“Стратегія кодування” та “Стратегія розміщення матриці штрихкодових символів”. Діаграма даного зв’язку зображена на рис. 4. 11.

2. Підмодуль формування зображення штрихкодової позначки

Наступним етапом створення ШК-позначки є формування зображення на основі матриці ШК-знаків. Під час цього процесу необхідно виконати дві основні операції: операцію перетворення елемента коду (вектора) ШК-знака в колір, а також перетворення матриці ШК-знаків у матрицю пікселів. Для програмної реалізації даного процесу доцільно виділити такі абстракції: “Генератор зображення ШК-позначки” (Barcode Image Generator), “Генератор матриці пікселів” (Barcode Bitmap Generator), “Колірна схема” (Color Scheme), “Параметри зображення ШК-позначки” (Barcode Image params).

Матрицю пікселів можна заповнити різними способами. Найочевиднішим способом заповнення такої матриці є зліва направо та згори вниз. З метою надання можливості вибору різних способів заповнення, доцільним є виділення абстракції “Стратегія розміщення матриці ШК-знаків” (BarcodePatternlArrangementStrategy) для заповнення матриці. Після отримання матриці пікселів зображення ШК-позначки, необхідно виконати операцію збереження даного зображення у файл, або передачу цього зображення у деякий потік виведення, наприклад у стандартний потік виведення (STDOUT), або у потік передачі мережевих даних. З метою інкапсуляції даної поведінки, доцільним є виділення абстракції “Друкар зображення” (ImagePrinter).

З огляду на те, що у вищеописаних абстракцій можуть бути різні імплементації, що залежать від вимог конкретного користувача, або вимог до системи, що розробляється, для конструювання екземпляра “Генератора зображення ШК-позначки” доцільно використати шаблон проєктування “Ін’єнція залежностей” (Dependency Injection).

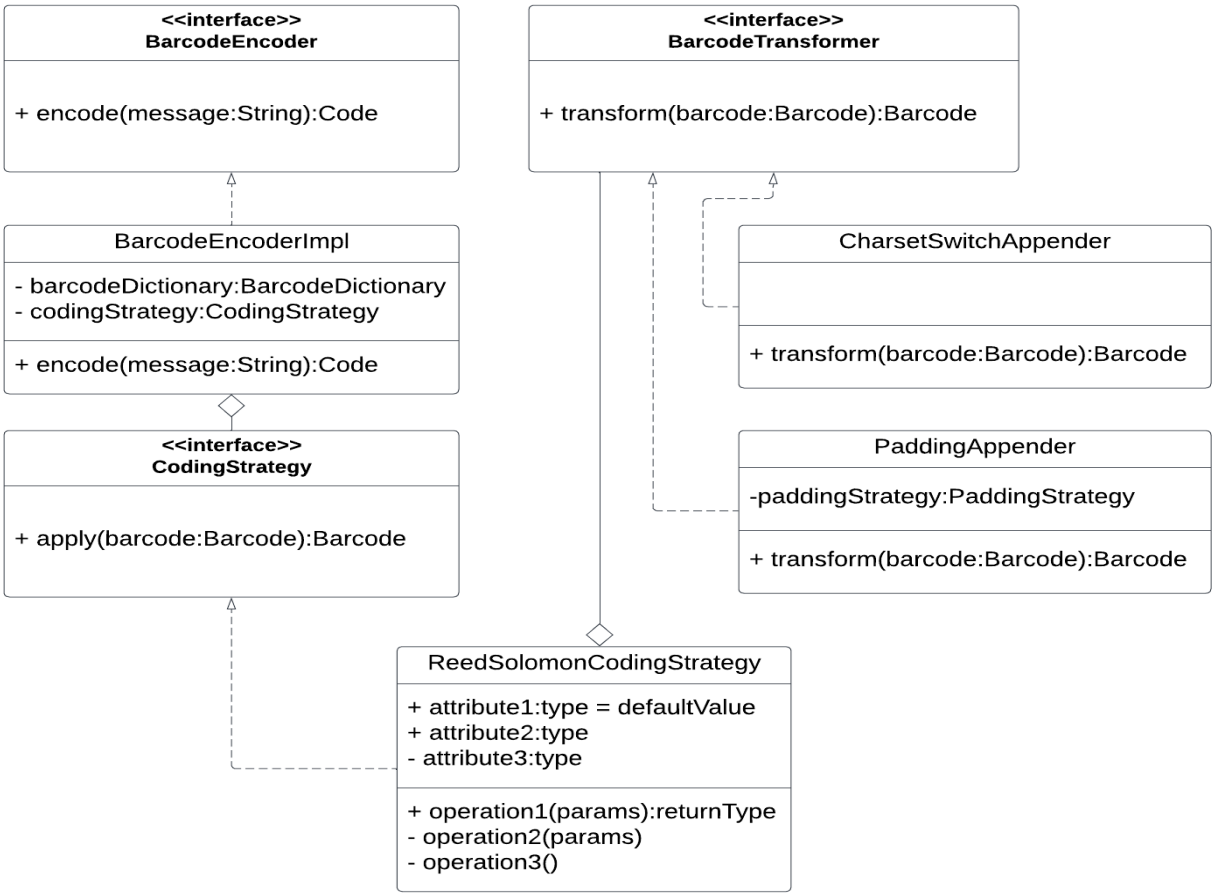


Рис. 4. 11. Діаграма класів підмодуля кодування

Даний шаблон забезпечує незалежність від конкретних реалізацій абстракцій, що використовуються. Таким чином буде забезпечена можливість підключення різних колірних схем та можливість збереження отриманого зображення у файл або передачі його мережею. Діаграма класів зображена на рис. 4. 12. Приклад використання розроблених абстракцій у деякому веб-сервісі, що генерує зображення для повідомлення, отриманого у HTTP запиті наведено у лістингу на рис. 4. 13.

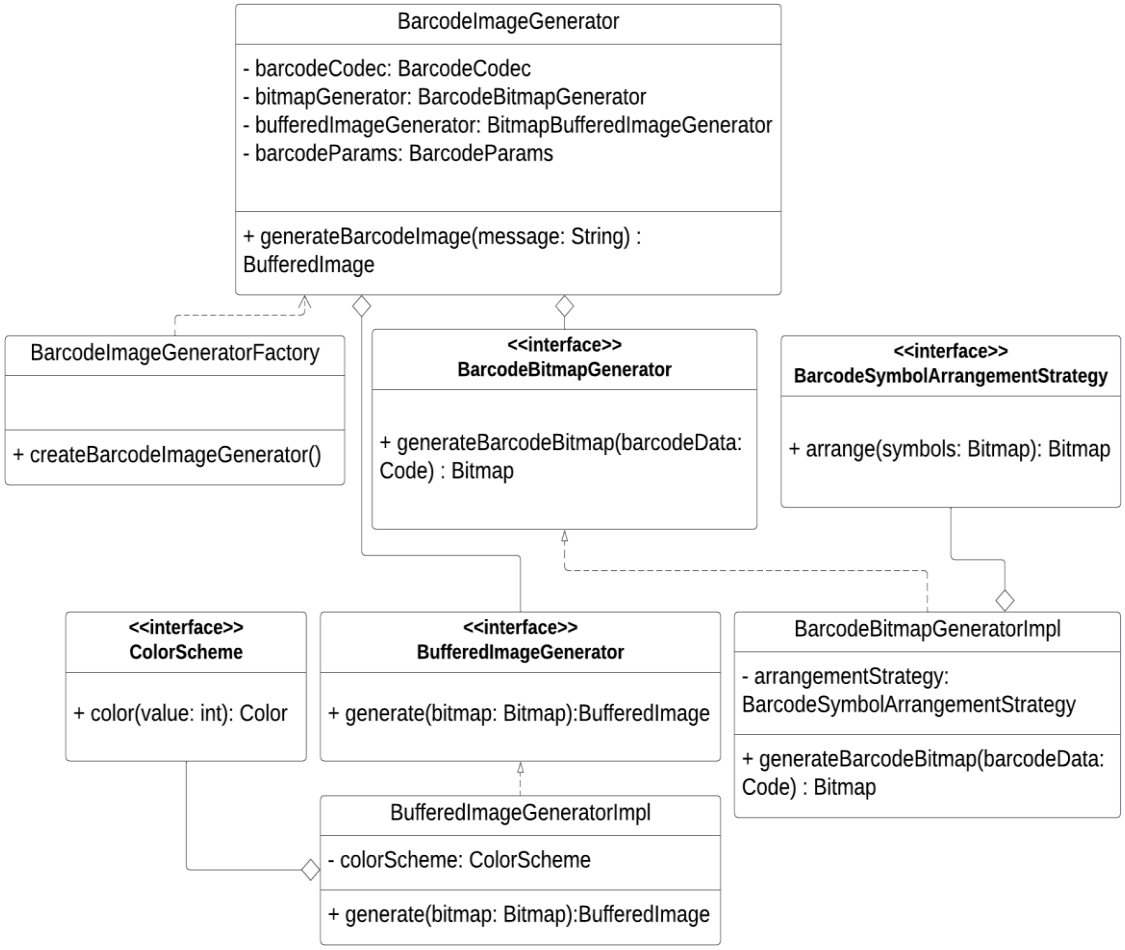


Рис. 4. 12. Діаграма класів підмодуля створення зображення ШК-позначки

```
@RestController
public class BarcodeGenerationController {
   private final BarcodeImageGenerator generator = new BarcodeImageGenerator(
           BarcodeCodecs.BASE_41_RS,
           new BarcodeBitmapGenerator(),
           new BitmapBufferedImageGeneratorImpl(ColorSchemes.DEFAULT_5_COLORS),
           ImmutableBarcodeParams.of(9, 30)
   );
   @RequestMapping(path = "/barcode/generate", method = RequestMethod.GET)
   public ResponseEntity<Resource> generateBarcode(@RequestParam String message)
throws IOException {
       var image = generator.generateBarcodeImage(message);
```

```
        var outputStream = new ByteArrayOutputStream();
        ImageIO.write(image, "bmp", outputStream);
        var resource = new ByteArrayResource(outputStream.toByteArray());
        return ResponseEntity.ok()
                .header("Content-Disposition", "attachment; filename=barcode.bmp")
                .contentLength(outputStream.toByteArray().length)
                .contentType(MediaType.APPLICATION_OCTET_STREAM)
                .body(resource);
    }
}
```

Рис. 4. 13. Приклад використання розроблених абстракцій для генерування зображення ШК-позначки

3. Підмодуль декодування штрихкодової позначки

Головною задачею даного підмодуля є зчитування інформації із зображення штрихкодової позначки. Дану задачу умовно можна розділити на два головні етапи: перетворення зображення у матрицю штрихкодових знаків; виділення первинних даних з матриці штрихкодових знаків. Оскільки задача перетворення зображення у матрицю штрихкодових знаків є окремою задачею високого рівня складності, що заснована на алгоритмах розпізнавання зображень, її імплементація виходить за межі даного дослідження і не є частниною підмодуля декодування штрихкодової позначки. Натомість, даний модуль реалізує задачу отримання первинної інформації з матриці штрихкодових знаків. Дана задача є оберненою до задачі кодування вхідної інформації, тому її програмна реалізація може бути виконана з використанням тих самих або схожих абстракцій. Таким чином, задача отримання первинної інформації з матриці штрихкодових знаків складається з двох основних етапів: отримання адаптованого повідомлення з матриці штрихкодових знаків; отримання початкової інформації з адаптованого повідомлення.

Очевидно, зчитана матриця штрихкодових знаків може містити помилки. У випадку використання коду Ріда-Соломона, завадостійкість

закодованої інформації забезпечується на двох рівнях: на рівні кожного штрихкодового знака (нижній рівень) та на рівні усієї ШК-позначки. Надалі розглядатимемо процедуру декодування штрихкодової позначки як слова коду Ріда-Соломона. Отже, процес отримання адаптованого повідомлення складається з двох етапів: декодування векторів штрихкодових знаків та декодування слова коду Ріда-Соломона. Штрихкодові знаки сформовані на основі коду БЧХ, хоча можливим є використання й інших коректувальних кодів.

З точки зору програмної реалізації даного процесу доречно виділити наступні абстракції: “Декодувальник штрихкоду” (BarcodeDecoder), “Стратегія коректування ШК символу” (BarcodeSymbolCorrectionStrategy). Абстракція “Стратегія коректування ШК символу” перебуває у відношенні композиції із абстракцією “Декодувальник штрихкоду” реалізуючи шаблон проєктування “Стратегія”.

На етапі отримання вхідного повідомлення з адаптованого необхідно виконати вилучення спеціальних символів-перемикаів та перетворення векторів ШК-знаків у текстовий рядок згідно з таблицею символіки штрихового коду. Виконання цих операцій доцільно інкапсулювати у виділеній вище абстракції “Декодувальник штрихкоду”. Діаграму класів описаної взаємодії зображено на рис. 4. 14.

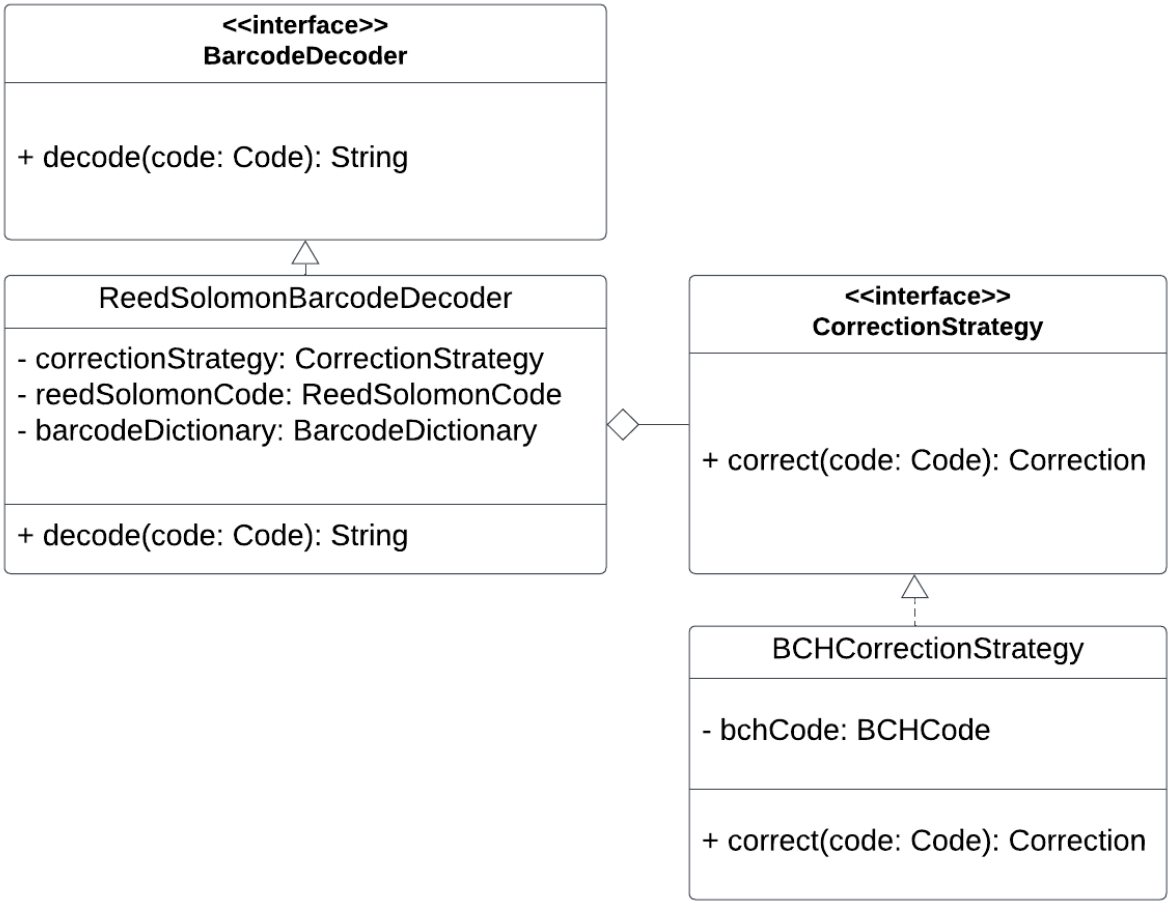


Рис. 4. 14. Діаграма класів декодування штрихкодової позначки

Наведемо два приклади генерування ШК-позначок для типових фрагментів інформації у медичних інформаційних системах.

1.Нехай необхідно закодувати таке повідомлення:

ДАТА | DATE: 01.01.2023 11:31 ID ПАЦІЄНТА: 172313

ПАЦІЄНТ | PATIENT: ВАСИЛЕНКО ДЕНИС ПЕТРОВИЧ СТАТЬ | SEX: ЧОЛ.

ДАТА НАРОДЖЕННЯ | DATE OF BIRTH: 18.05.1974 ВІК | AGE: 49

ОПИС ДОСЛІДЖЕННЯ | DESCRIPTION OF THE STUDY

ПРОТОКОЛ ОБСТЕЖЕННЯ: AX T2, DWI (B1000), FLAIR, T1, SWAN; SG T2; COR T2;

НА СЕРІЇ МРТ ГОЛОВНОГО МОЗКУ СЕРЕДИННІ СТРУКТУРИ НЕ ЗМІЩЕНІ.

ШЛУНОЧКИ МОЗКУ РОЗШИРЕНІ. БІЧНІ АСИМЕТРИЧНІ S<D. ПЕРИВЕНТРИКУЛЯРНО ДО БІЧНИХ ШЛУНОЧКІВВИЗНАЧАЮТЬСЯ ЗОНИ ЛЕЙКОАРЕОЗА ШИРИНОЮ ДО 0,63 СМ.

РОЗШИРЕНІ ПІДПАВУТИННІ ПРОСТОРИ НАД ПІВКУЛЯМИ ВЕЛИКОГО МОЗКУ. ПІДПАВУТИННІ ПРОСТОРИ НАД ПІВКУЛЯМИ МОЗОЧКА НЕ ЗМІНЕНІ.

У БІЛІЙ РЕЧОВИНІ ПІВКУЛЬ ВЕЛИКОГО МОЗКУ ПЕРЕВАЖНО СУБКОРТИКАЛЬНО ВИЗНАЧАЮТЬСЯ ПООДИНОКІ ДІЛЯНКИ ПЕРИВАЗАЛЬНОГО АСТРОГЛІОЗУ УМОВНИМИ ДІАМЕТРАМИ ДО 0,6 СМ.

ВОГНИЩ ОБМЕЖЕННЯ ДИФУЗІЇ У МОЗКОВІЙ ПАРЕНХІМІ НА DWI НА МОМЕНТ ОБСТЕЖЕННЯ НЕ ВИЗНАЧАЄТЬСЯ.

МОЗОЛИСТЕ ТІЛО, МЕЗЕНЦЕФАЛЬНІ, СТОВБУРОВІ ВІДДІЛИ МОЗКУ, ПІВКУЛІ МОЗОЧКА БЕЗ ОСОБЛИВОСТЕЙ.

МИГДАЛИКИ МОЗОЧКУ ВІЗУАЛІЗУЮТЬСЯ НА РІВНІ ВЕЛИКОГО ПОТИЛИЧНОГО ОТВОРУ.

ТУРЕЦЬКЕ СІДЛО НЕ ЗМІНЕНЕ ЗА РОЗМІРАМИ, КОНТУРИ ЧІТКІ. ГІПОФІЗ ЗМЕНШЕНИЙ В РОЗМІРАХ, БЕЗ ОБ’ЄМНО-ВОГНИЩЕВИХ ЗМІН. ЛІЙКА ГІПОФІЗА РОЗТАШОВАНА ПО СЕРЕДНІЙ ЛІНІЇ.

ОКОРУХОВІ М'ЯЗИ, РЕТРОБУЛЬБАРНІ ПРОСТОРИ, ЗОРОВІ НЕРВИ, ХІАЗМА БЕЗ ОСОБЛИВОСТЕЙ.

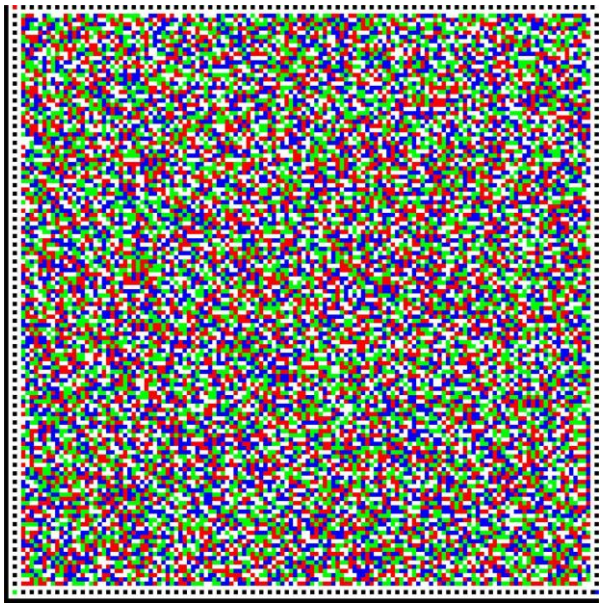
ВИЗНАЧАЄТЬСЯ КІСТА В ПРАВІЙ ВЕРХНЬОЩЕЛЕПНІЙ ПАЗУСІ РОЗМІРАМИ 2,9Х1,6Х1,4 СМ. В ЛІВІЙ ВЕРХНЄЩЕЛЕПНІЙ ПАЗУСІ ПАТОЛОГІЧНИЙ ВМІСТ В МАЛІЙ КІЛЬКОСТІ З ВКЛЮЧЕННЯМ ГАЗУ. В ПАЗУХАХ ГРАТЧАСТОЇ КІСТКИ ПОТОВЩЕНА СЛИЗОВА ОБОЛОНКА. ПНЕВМАТИЗАЦІЯ РЕШТИ ПРИНОСОВИХ ПАЗУХ НЕ ПОРУШЕНА.

ВИСНОВОК | CONCLUSION

МР-ОЗНАКИ ЦЕРЕБРАЛЬНОЇ МІКРОАНГІОПАТІЇ (FAZEKAS I).

УВАГА! ДАНИЙ ВИСНОВОК НЕ Є ДІАГНОЗОМ. ОБОВ'ЯЗКОВО ПЕРЕДАЙТЕ ЙОГО ВАШОМУ ЛІКАРЮ!

Згенерована для такого повідомлення ШК-позначка має вигляд:



2. Нехай необхідно подати у штрихкодовому вигляді наступне повідомлення:

ДАТА | DATE: 01.02.2023 08:31 ID ПАЦІЄНТА: 261132

ПАЦІЄНТ | PATIENT: КОРОЛЬОВ ПЕТРО АНДРІЙОВИЧ СТАТЬ | SEX: ЧОЛОВІЧА

ДАТА НАРОДЖЕННЯ | DATE OF BIRTH: 11.08.1993 ВІК | AGE: 30 РОКІВ

ОПИС ДУПЛЕКСНОГО СКАНУВАННЯ ЕКСТРАКРАНІАЛЬНОГО ВІДДІЛУ БРАХІОЦЕФАЛЬНИХ СУДИН

ДИСТАЛЬНИЙ ВІДДІЛ ПЛЕЧЕГОЛОВНОГО СТОВБУРА, ПРОКСИМАЛЬНІ ВІДДІЛИ ПІДКЛЮЧИЧНИХ (ПКА), ЗАГАЛЬНІ СОННІ, ЕКСТРАКРАНІАЛЬНІ ВІДДІЛИ ВНУТРІШНІХ ТА ЗОВНІШНІХ СОННИХ АРТЕРІЙ (ЗСА, ВСА, ЗВСА), ОБИДВІ ХРЕБТОВІ АРТЕРІЇ (ХА) В СЕГМЕНТАХ V1 ТА V2 ВІЗУАЛІЗОВАНІ. ЯКІСТЬ ВІЗУАЛІЗАЦІЇ ЗАДОВІЛЬНА.

ПРОСВІТ СУДИН РІВНОМІРНИЙ, ПАТОЛОГІЧНІ ВКЛЮЧЕННЯ В СУДИННУ СТІНКУ НЕ ВИЗНАЧАЮТЬСЯ, НЕМАЄ ТАКОЖ ОЗНАК РОЗШИРЕННЯ ТА ЗВИВИСТОСТІ АРТЕРІАЛЬНИХ СТОВБУРІВ.

ТОВЩИНА КІМ (КОМПЛЕКС ІНТИМА-МЕДІА) В ДІЛЯНЦІ БІФУРКАЦІЇ ЗСА ПРАВОРУЧ ДО 0,8 ММ, ЛІВОРУЧ ДО 0,8 ММ (N-ДО 1,0 ММ), ЕХОГЕННІСТЬ ЙОГО НЕ ЗМІНЕНА, БЕЗ ПОРУШЕННЯ ДИФЕРЕНЦІАЦІЇ НА ШАРИ.

ДІАМЕТР ХРЕБТОВИХ АРТЕРІЙ: ПРАВА - 3,0 ММ, ЛІВА - 4,3 ММ (НОРМА 3,0 - 5,5 ММ).

ВНУТРІШНІ ТА ЗОВНІШНІ ЯРЕМНІ ВЕНИ НЕ РОЗШИРЕНІ.

ПОКАЗНИКИ КРОВОТОКУ В БРАХІОЦЕФАЛЬНИХ СУДИНАХ ДИВ. ТАБЛИЦЮ.

ЗСА ПРАВОРУЧ VPS 72 СМ\СЕК, IR 0.7, ЛІВОРУЧ VPS 65 СМ\СЕК, IR 0,7;

ВСА ПРАВОРУЧ VPS 55 СМ\СЕК, IR 0.5, ЛІВОРУЧ VPS 56 СМ\СЕК, IR 0.5;

ПРАВА ХА В СЕГМЕНТІ V2 VPS 25 СМ\СЕК, IR 0.5;

ЛІВА ХА В СЕГМЕНТІ V2 VPS 32 СМ\СЕК, IR 0.5

ОПИС ТРАНСКРАНІАЛЬНОГО ДУПЛЕКСНОГО СКАНУВАННЯ

ВИРАЖЕНІСТЬ УЛЬТРАЗВУКОВИХ «ВІКОН» ЗАДОВІЛЬНА.

ВИРАЖЕНІСТЬ СКРОНЕВИХ УЛЬТРАЗВУКОВИХ «ВІКОН» ЗАДОВІЛЬНА; ПОТИЛИЧНОГО УЛЬТРАЗВУКОВОГО «ВІКНА» ЗАДОВІЛЬНА. ЯКІСТЬ ІНТРАКРАНІАЛЬНОЇ ВІЗУАЛІЗАЦІЇ ЗАДОВІЛЬНА.

ОТРИМАНО КОЛЬОРОВІ КАРТОГРАМИ І СПЕКТРИ ДОПЛЕРІВСЬКИХ ЗСУВІВ ЧАСТОТ З СРЕДНІХ (СМА), ПЕРЕДНІХ (ПМА) , ЗАДНІХ (ЗМА) МОЗКОВИХ АРТЕРІЙ, А ТАКОЖ ІНТРАКРАНІАЛЬНИХ ВІДДІЛІВ ХРЕБТОВИХ АРТЕРІЙ (СЕГМЕНТИ V4 ХА), ОСНОВНОЇ АРТЕРІЇ МОЗКУ (ОАМ), БАЗАЛЬНИХ ВЕН МОЗКУ (БВМ).

ПОКАЗНИКИ ІНТРАКРАНІАЛЬНОГО КРОВОТОКУ ДИВ. ТАБЛИЦЮ, А ТАКОЖ:

СМА ПРАВОРУЧ VPS 93 СМ\СЕК, IR 0.5, ЛІВОРУЧ VPS 90 СМ\СЕК, IR 0,5;

ЗМА ПРАВОРУЧ VPS 48 СМ\СЕК, IR 0.5, ЛІВОРУЧ VPS 48 СМ\СЕК, IR 0.5;

ПРАВА ХА В СЕГМЕНТІ V4 VPS 40 СМ\СЕК, IR 0.5;

ЛІВА ХА В СЕГМЕНТІ V4 VPS 44 СМ\СЕК, IR 0.5;

ОАМ VPS 48 СМ\СЕК, IR 0.5.

ВЕНИ РОЗЕНТАЛЯ: ПРАВОРУЧ VPS 13 СМ\СЕК, ЛІВОРУЧ VPS 13 СМ\СЕК.

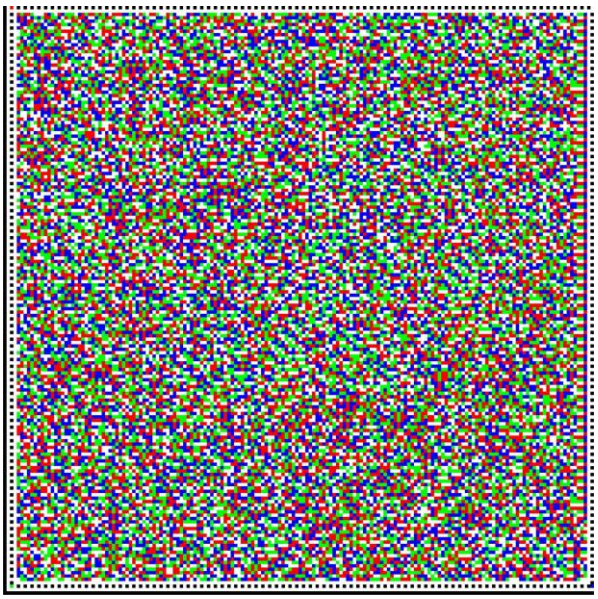
ОЦІНКА РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ | CONCLUSION

КІМ НЕ ЗМІНЕНИЙ, АТЕРОСКЛЕРОТИЧНИХ БЛЯШОК НЕМАЄ; ПРОСВІТ СУДИН ВІЛЬНИЙ; РОЗШИРЕННЯ ТА ЗВИВИСТОСТІ АРТЕРІАЛЬНИХ СТОВБУРІВ НЕМАЄ.

ГЕМОДИНАМІЧНО ЗНАЧУЩИХ ЗМІН КРОВОТОКУ НА ЕКСТРАКРАНІАЛЬНОМУ РІВНІ ТА ПО МАГІСТРАЛЬНИМ АРТЕРІЯМ ГОЛОВИ КАРОТИДНОГО ТА ВЕРТЕБРО-БАЗИЛЯРНОГО БАСЕЙНУ НЕ ВИЯВЛЕНО.

ПОРУШЕННЯ ВЕНОЗНОГО ВІДТОКУ ПО БАЗАЛЬНИМ ВЕНАМ МОЗКУ НЕ ВИЯВЛЕНО.

Тоді ШК-позначка матиме вигляд:



4. 3. Характеристики якості розробленої програмної системи для автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів

Якість програмного забезпечення є мірою відповідності вимогам до нього [85 - 105]. Стандарти [99, 100] виділяють такі категорії оцінювання якості програмної системи:

- функціональна придатність (Functional Suitability),
- ефективність роботи (Performance efficiency),
- сумісність (Compatibility),
- зручність використання (Usability),
- надійність (Reliability),
- безпека (Security),
- зручність супроводу (Maintainability),
- переносимість (Portability).

Даний список не є вичерпним, існують й інші критерії оцінювання якості програмного забезпечення, серед яких - критерії оцінювання зв’язності програмного коду, міра покриття тестами, можливість розширення тощо.

Жодна з цих категорій не є строгою; конкретні критерії оцінювання якості програмної системи є індивідуальними, а в деяких випадках, як, наприклад, з точки зору зручності використання, навіть суб'єктивними [85].

Враховуючи те, що головною фазою життєвого циклу програмної системи є власне період її експлуатації, чільне місце в оцінюванні якості програмної системи посідає оцінювання саме її надійності. Надійність є необхідною умовою успішної експлуатації програмної системи [95, 98].

Індустрія виробила деякі загальноприйняті підходи та методи нормування вимог до надійності програмних систем. Одним з таких підходів є так звані цілі рівня послуг (Service Level Objectives - SLO). Дані цілі визначають вимоги до надійності програмної системи у вигляді однозначних, числових показників.

З метою оцінювання надійності програмної системи загальноприйнятою практикою є проведення її безперервного моніторингу. Моніторинг системи включає в себе періодичний збір та аналіз даних, що збираються з вузлів інформаційної системи. До таких даних відносять метрики, логи тощо.

Процес збору метрик з вузлів інформаційної системи зазвичай реалізують у вигляді HTTP-запитів до вузлів із заданою періодичністю; ця періодичність складає від 5 до 30 секунд. У кожного вузла є стандарний інтерфейс збору метрик. Метрики є "зліпком" стану вузла в момент запиту. Однією з найпоширеніших систем збору метрик є Prometheus. У системі Prometheus виділяють наступні види метрик: лічильник (counter), датчик (gauge). Лічильник є накопичуваною величиною, а датчик - величиною на момент запиту.

Приклад метрик, зібраних з вузла в системі Prometheus, зображено на рис. 4. 15.

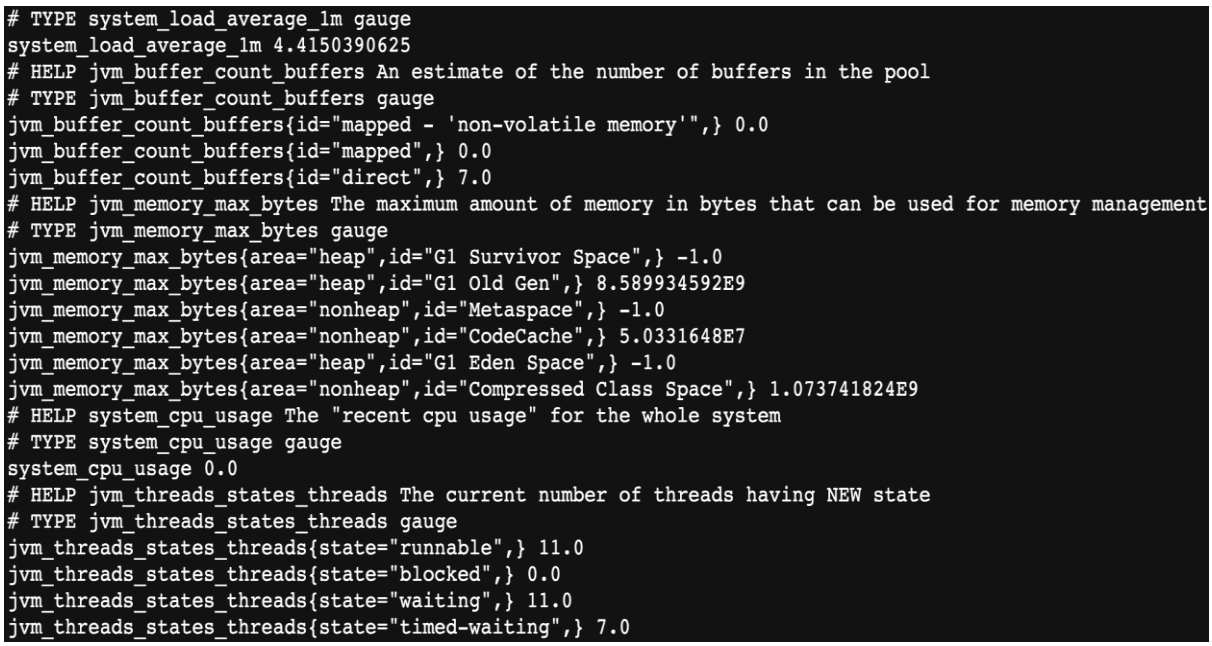


Рис. 4. 15. Приклад метрик у форматі Prometheus

Наступним етапом обробки метрик є їх візуалізація та обробка. На сьогодні найпоширенішим інструментом відображення та обробки метрик є комплекс Grafana. Даний комплекс зінтегровано з різними системами збору метрик, такими як Prometheus, або іншими джерелами даних, такими як логи, журнали запитів до баз даних тощо. За допомогою даного комплексу можливо створювати візуалізацію загального стану системи, налаштовувати верхні пороги допустимих значень тощо. Комплекс Grafana підтримує різні види графіків, зокрема такі як: гістограма; теплова, кругова діаграми; цифровий, аналоговий датчик тощо.

Для оцінювання характеристик розробленої програмної системи на основі аналізу метрик, оберемо один з компонентів нашої програмної системи, а саме - сервіс генерування штрихкодового зображення для заданого текстового повідомлення. Даний компонент реалізовано у вигляді веб-сервісу з HTTP-інтерфейсом, який, у свою чергу, реалізований за допомогою мови програмування Java з використанням фреймворку Spring Boot. Сервіс приймає на вхід HTTP-запит, в якому передається текстове повідомленням, яке необхідно закодувати у вигляді багатоколірного штрихового коду.

Результатом обробки запиту є HTTP-відповідь, що містить згенерований файл із зображенням штрихкодової позначки у форматі bmp.
Приклад запиту та відповіді сервера наведено на рис. 4. 16.

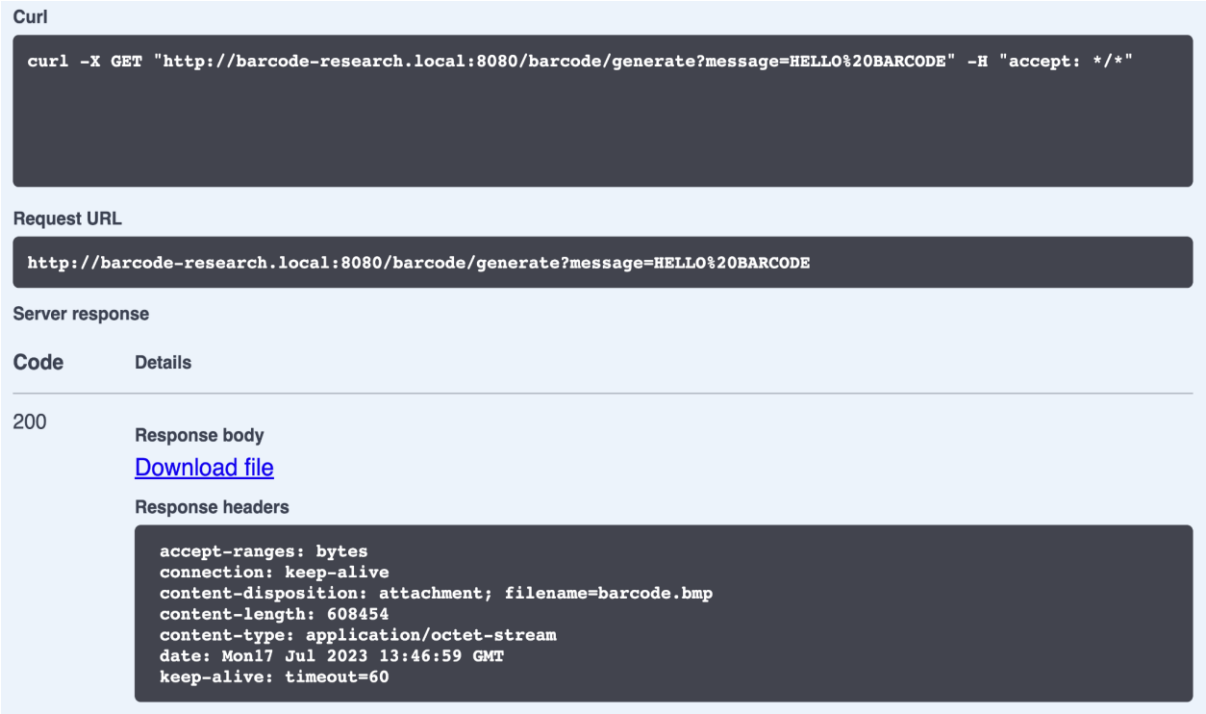


Рис. 4. 16. Приклад запиту та результату генерування штрихкодового зображення

Для дослідження характеристик надійності розробленого нами сервісу пропонується побудувати систему моніторингу по п’яти основних показниках:

- кількість запитів за хвилину,
- час обробки запиту (максимальний, 99/95/50 персентиль),
- HTTP-коди відповідей сервера (2xx/3xx/4xx/5xx),
- відсоток використання процесорних тактів,
- відсоток використання оперативної пам’яті.

Кількість запитів на хвилину характеризує очікуваний обсяг трафіку, що повинен оброблятись створеною програмною системою.

З метою однозначного числового вираження вимоги до часу обробки запиту використовують поняття перцентилю. Так, вимога про те, що 99 перцентиль часу обробки запиту не повинен перевищувати 50 мілісекунд, означає нижній поріг часу обробки запиту, у який повинні потрапляти 99% запитів. Скажімо, якщо 99% запитів було оброблено не більше ніж за 50 мілісекунд, а 1% запитів з якихось причин був оброблений довше, дана вимога вважається виконаною.

Дані щодо перелічених п’яти показників будемо отримувати для одного вузла сервісу, що запущений на машині Macbook Pro з процесором 2,4 GHz 8-Core Intel Core i9 з об’ємом оперативної пам’яті 32 GB 2667 MHz DDR4. Максимальний об’єм пам’яті, що виділяється операційною системою під процес веб-сервісу складає 16 GB. Дослідження проводитимемо методом навантажувального тестування. Для такого тестування використовують bash скрипти. Приклад такого скрипту наведений на рис. 4. 17.

```
#!/bin/bash
for (( i = 0; i < 500000; i++ )); do
msg_length=100
msg=$(cat /dev/urandom | env LC_ALL=C tr -dc 'A-Z0-9' | fold -w $msg_length | head -n 1)
random_number=$(( RANDOM % 1000 + 1 ))
sleep_time=$(echo "scale=3; $random_number / 10000" | bc)
echo "Generating barcode for: $msg"
sleep "$sleep_time"
curl --request GET -sL \
--url "http://localhost:8080/barcode/generate?message=$msg" \
--output /dev/null &
done
```

Рис. 4. 17. Приклад bash скрипту для навантажувального тестування

Фоновим навантаженням, проти якого проводитимемо дослідження, оберемо навантаження обсягом в 600 RPM (RPM – request per minute, кількість запитів за хвилину).

Метрики роботи системи у фоновому режимі зображено на рис. 4. 18.

Для дослідження характеристик сервісу генерування штрихкодових зображень були проведені такі експерименти.

1. Оцінювання впливу кількості запитів на хвилину на середній час обробки запиту, завантаження процесора, завантаження оперативної пам’яті.
2. Оцінювання впливу некорекних запитів на середній час обробки запиту, завантаження процесора, завантаження оперативної пам’яті.
3. Оцінювання впливу довжини повідомлення, що кодується, на середній час обробки запиту, завантаження процесора, завантаження оперативної пам’яті.

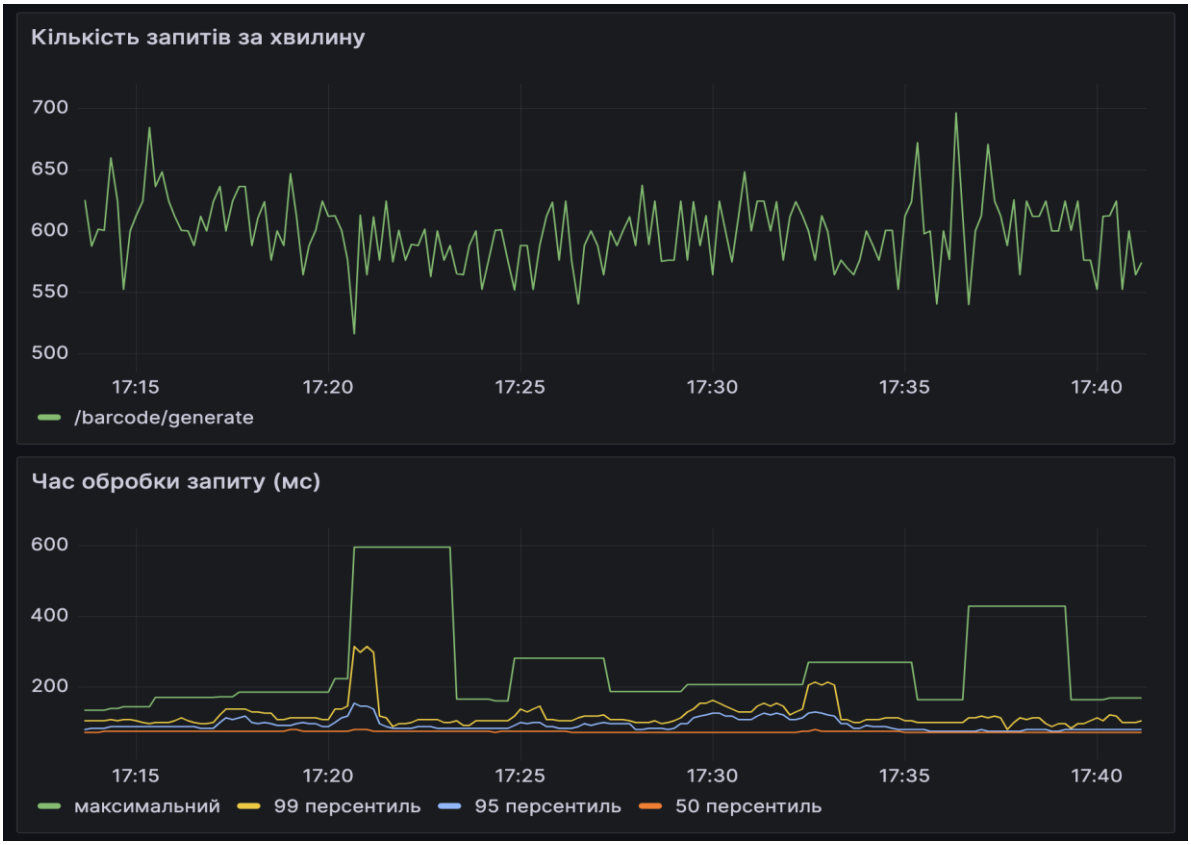


Рис. 4. 18. Робота системи в нормальному режимі

При збільшенні кількості запитів з 600RPM до 1500RPM спостерігається приблизно дворазове збільшення середнього часу обробки запиту – з 100 мілісекунд до 190 мілісекунд. Для метрики використання ресурсів процесора фіксується приблизно п’ятиразове зростання - з 5% до 20%. Це пояснюється тим, що алгоритми завадостійкого кодування, що використовуються в процесі генерування штрихового коду, мають доволі високу алгоритмічну складність. Стосовно використання оперативної пам’яті спостерігається флуктуація її використання. Це може пояснюватись так. Веб-сервіс розроблено на мові програмування Java, отже, виконується на віртуальній машині, що має функцію очистки пам’яті (Garbage Collection). Подібні флуктуації характерні саме для процесу очистки пам’яті. По-друге, алгоритми генерування штрихових кодів не використовують великі об’єми буферної пам’яті.

Результати даного дослідження зображені на рис. 4. 19 та рис. 4. 20.

Рис. 4. 19. Залежність часу обробки запиту від кількості запитів за хвилину

Рис. 4. 20. Залежність використання процесора та оперативної пам’яті від кількості запитів за хвилину

У рамках наступного експерименту оцінювалась здатність системи працювати в умовах некоректних вхідних даних. Так, для генерування штрихкодових зображень використовується алфавіт ШК, у якому латинські та кириличні букви є лише у верхньому регістрі. Оцінити здатність системи функціонувати в умовах некоректних вхідних даних пропонується використовуючи скрипти навантажувального тестування, що генерують деякий відсоток запитів, що включають в собі недопустимі символи. Результати, отримані для наборів з 5%, 20% та 50% некорекних запитів представлені на рис. 4. 21, рис. 4. 22.

Бачимо, що зростання відсотка некоректних запитів практично не впливає на середній час обробки запитів.

Стосовно використання системних ресурсів чітко простежується зменшення використання ресурсів процесора зі збільшенням кількості некоректних запитів. Це пояснюється тим, що обробка некоректного запиту відбувається без залучення обчислювальних потужностей, що необхідні при формуванні штрихкодової позначки.

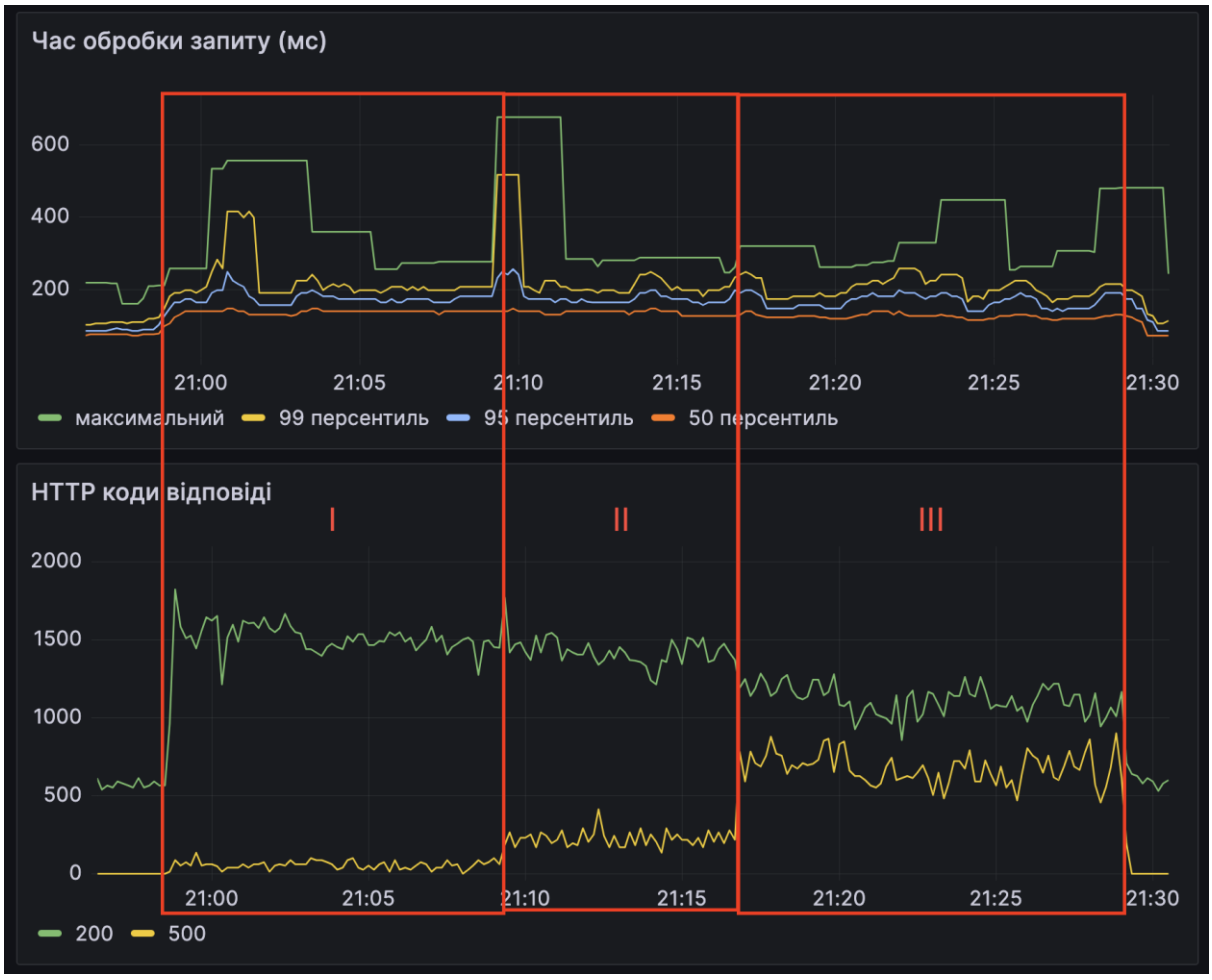


Рис. 4.21. Залежність середнього часу обробки запиту від кількості некорекних запитів

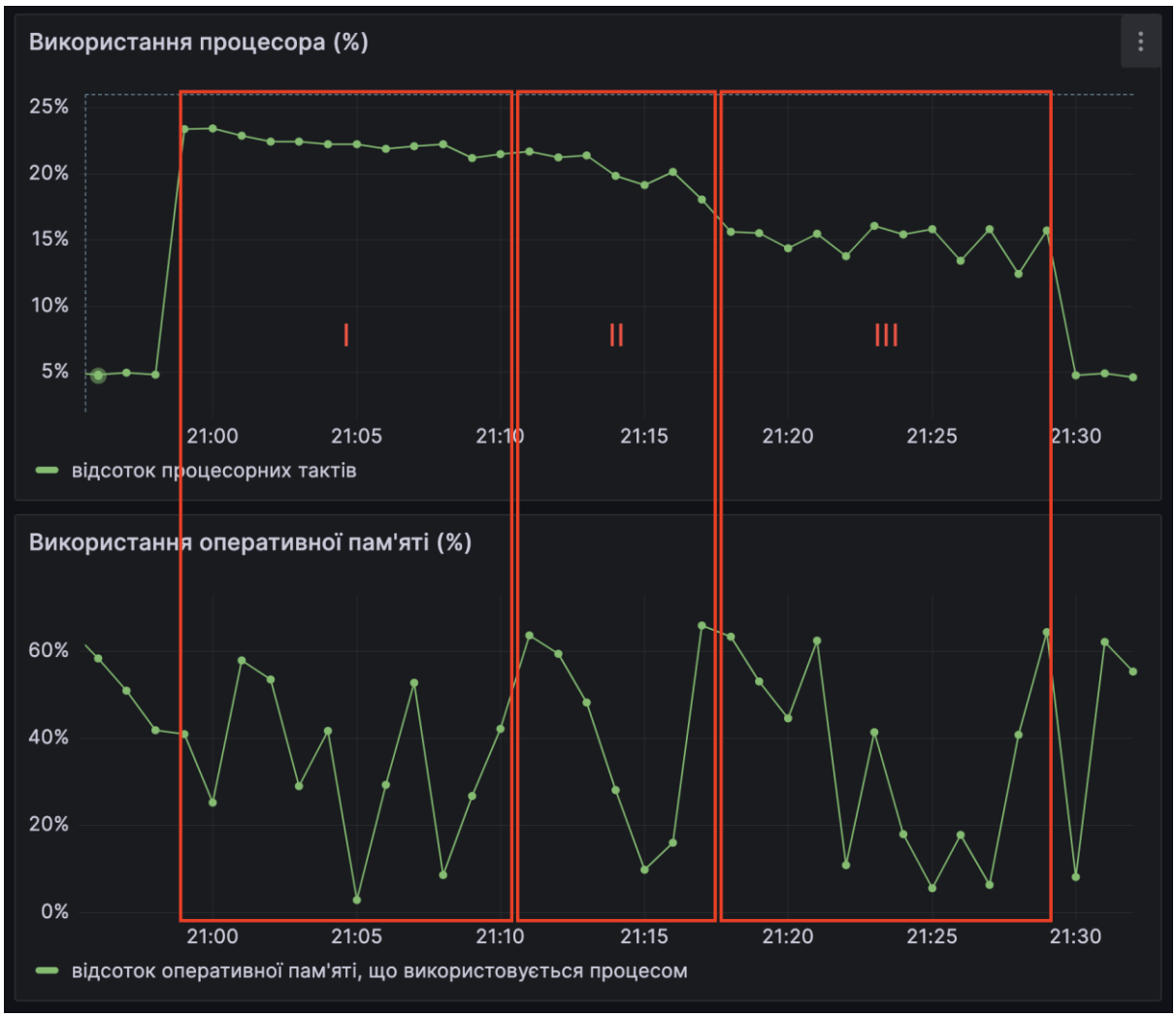


Рис. 4. 22. Залежність використання ресурсів процесора від кількості некорекних запитів

Заключним експериментом у дослідженні характеристик розробленої програмної системи є оцінювання залежності середнього часу обробки запиту від розміру повідомлення, для якого генерується штрихкодове зображення. Для вирішення цієї задачі використовувались скрипти навантажувального тестування зі змінними довжинами повідомлень. Результати експериментів показали пряму залежність середнього часу обробки запиту та рівня використання процесора від довжини вхідного повідомлення. Так, при збільшенні довжини повідомлення, що кодується, з 200 до 500 символів, спостерігається майже восьмиразове збільшення середнього часу обробки - з

400 мілісекунд (для 200-символьних повідомлень) до 3200 мілісекунд (для 500-символьних повідомлень). При збільшенні довжини повідомлення з 500 до 1500 символів, ситуація виглядає аналогічно - збільшення середнього часу обробки запиту з 3200 до 16000 мілісекунд. Подібна тенденція спостерігається і щодо використання ресурсів процесора.

Результати даного експерименту зображені на рис. 4. 23 та рис. 4. 24.

Рис. 4. 23. Залежність середнього часу обробки запиту від довжини повідомлення, що кодується

Рис. 4. 24. Залежність використання ресурсів процесора від довжини повідомлення, що кодується

За результатами проведеного емпіричного дослідження розробленого сервісу генерування шрихкодового зображення на основі метрик можна зробити наступні висновки.

1. Середній час обробки запиту лінійно залежить від кількості запитів за хвилину.
2. Середній час обробки запиту експоненційно залежить від розміру вхідного повідомлення.

3. Оперативна пам’ять не є критичним ресурсом для виконання задачі генерування штрихкодового зображення.
4. Частота процесора є критичним ресурсом для виконання задачі генерування штрихкодового зображення, адже дана операція є обчислювально складною і вимагає використання значних ресурсів процесора.

Нехай необхідно виконати задачу оцінювання кількості апаратних ресурсів, що необхідні для розгортання кластера для сервісу генерування штрихкодових зображень, що має обмеження для довжини вхідного повідомлення (до 1500 символів) та забезпечує наступні цілі рівня послуг (SLO):

- 99 персентиль середнього часу обробки запиту з довжиною повідомлення в межах від 1 до 200 символів повинен складати не більше 200 мілісекунд,
- 99 персентиль середнього часу обробки запиту з довжиною повідомлення в межах від 201 до 500 символів повинен складати не більше 500 мілісекунд,
- 99 персентиль середнього часу обробки запиту з довжиною повідомлення в межах від 501 до 1500 символів повинен складати не більше 3000 мілісекунд,
- система повинна мати пропускну здатність у розмірі 5000 запитів за хвилину.

Для виконання оцінювання кількості апаратних ресурсів зробімо низку припущень: довжина повідомлення, що кодується, є рівномірною величиною; мережа передачі даних є надійною; пропускна здатність мережі є достатньою для виконання заданих цілей рівня послуг.

Отже, враховуючи рівномірний розподіл довжин повідомлень що кодуються, (а це 5000 повідомлень) кількість повідомлень відповідної довжини матиме вигляд:

$$n_{1-200} = \frac{200 * 5000}{1500} \approx 667,$$

$$n_{201-500} = \frac{300 * 5000}{1500} = 1000,$$

$$n_{501-1500} = \frac{1000 * 5000}{1500} \approx 3333.$$

З проведених дослідів випливає, що для обробки 1200 запитів на хвилину, 99 персентиль часу обробки запиту складає 400 мілісекунд. Враховуючи лінійну залежність часу обробки від кількості запитів на хвилину, можна вважати, що середній час обробки в 200 мілісекунд буде досягнутий для $\approx 1200/2 = 600$ запитів за хвилину. Отже, верхня оцінка кількості вузлів з апаратими можливостями подібними до машини, на якій проводилось дослідження, для обробки повідомлень з довжиною повідомлень в межах між 1 та 200 символами складає $667/600 \approx 1.1 = 2$ вузли.

Аналогічно, для повідомлень завдовжки 201 - 500 символів верхня оцінка кількості вузлів, що необхідна для досягнення заданих цілей, складає $\approx 6.7 = 7$ вузлів. Для повідомлень завдовжки 501 - 1500 символів відповідно ≈ 37 вузлів. Таким чином, для досягнення заданих цілей рівня послуг необхідно розгорнути кластер з $2 + 7 + 37 = 46$ вузлів.

Проведене дослідження дає підстави стверджувати, що використання системи збору і аналізу метрик є ефективним інструментом оцінювання характеристик програмної системи. Даний підхід дозволив виконати оцінювання апаратних ресурсів, що необхідні для забезпечення пропускної здатності системи в межах визначених цілей рівня послуг, які є достатніми для забезпечення роботи медичної інформаційної системи в межах країни (46 вузлів).

Таким чином, забезпечується високий рівень надійності програмної системи та цілодобовий моніторинг стану її роботи з можливою подальшою оптимізацією ресурсів.

Така оптимізація дозволяє не тільки зменшити витрати на підтримку функціонування розгорнутого кластера, а й максимально убезпечити саму

програмну систему, а також її користувачів від тривалих збоїв у її роботі, що є важливим у сфері охорони здоров'я, адже програмні рішення у цій царині можна розглядати як критичну інфраструктуру, що безпосередньо впливає на безпеку та якість життя людей.

Програмну систему, що в повній мірі слідує практикам, описаним в даному підрозділі, з впевненістю можна назвати надійною, а, отже, якісною.

4. 4. Висновки до четвертого розділу

Виконані дослідження щодо проєктування програмного забезпечення процесів автоматичної ідентифікації в медичних інформаційних системах дозволяють зробити наступні висновки.

1. На основі аналізу та пріоритизації функціональних вимог розроблено архітектуру програмної системи як ядро медичної інформаційної системи, характерною особливістю якої є децентралізоване (розподілене) зберігання медичних даних пацієнта, а також те, що доступ до його медичних даних здійснюється за допомогою автоматичної ідентифікації на основі багатоколірного штрихового кодування. Запропонована архітектура дозволяє спростити процес розроблення програмного забезпечення в медичній галузі.
2. З використанням низки шаблонів проєктування розроблено структурну організацію модуля кодування-декодування багатоколірного завадостійкого штрихового коду, який є основним компонентом програмної системи, що дозволяє ефективно реалізовувати процес прямого і зворотного перетворення вхідних алфавітно-цифрових даних у графічну форму штрихкодового зображення. Це спрощує процес розроблення програмної системи в цілому та є основою для її подальшого розширення.
3. Для забезпечення належного рівня надійності розробленої програмної системи запропоновано організацію моніторингу функціонування програмного забезпечення у складі медичної інформаційної системи на

основі метрик. Моніторинг полягає в періодичному збиранні метрик з вузлів інформаційної системи, їх обробці в системі Prometheus та візуалізації в системі Grafana. Це дає можливість своєчасно виявляти аномалії в роботі програмного забезпечення та підтримувати надійне функціонування програмної системи.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-технічну задачу удосконалення технології розроблення алгоритмічного та програмного забезпечення процесів автоматичної ідентифікації в медичних інформаційних системах на основі багатоколірних завадостійких штрихових кодів.

У дисертаційному дослідженні отримано такі основні наукові результати.

1. Показано, що для розроблення якісного програмного забезпечення для автоматичної ідентифікації в медичних інформаційних системах на основі багатоколірного завадостійкого штрихового кодування інформації на етапі проєктування необхідно формувати чіткий перелік функціональних вимог до розроблюваної програмної системи із зазначенням їх пріоритетності.

   На відміну від існуючого підходу до визначення пріоритетності функціональних вимог запропоновано визначати пріоритетність вимоги на основі п’яти числових критеріїв (“вигода”, “втрата”, “вартість”, “ризик”, “близькість вигоди”), кожному з яких присвоюють числові значення з діапазону 1…10 шляхом їх експертного оцінювання, а потім на цій основі кількісно визначати загальний пріоритет вимоги за допомогою формульного обчислення. Це дозволяє упорядковувати вимоги за спаданням пріоритету та на основі цього ефективно планувати виконання робіт з розроблення програмного забезпечення.

   Такий підхід до пріоритизації функціональних вимог є подальшим розвитком теоретичних засад розроблення вимог до проєктованого програмного забезпечення.

2. Показано, що для підвищення ефективності організації надання медичних послуг пацієнтам на основі медичної інформаційної системи необхідно розробити архітектуру програмної системи, яка б надавала користувачам додаткові можливості щодо електронної взаємодії пацієнта і лікаря.

Запропоновано архітектуру програмної системи як ядро медичної інформаційної системи, характерною рисою якої є забезпечення багатоколірного завадостійкого штрихового кодування персональних даних пацієнтів та розподіленого зберігання мультимодальних медичних даних, що дозволяє: забезпечити швидке і безпомилкове введення даних пацієнта, гарантувати цілісність даних, а також спростити процеси створення програмного забезпечення для галузі охорони здоров'я нового покоління - медичних інформаційних систем з автоматичною ідентифікацією об'єктів медичного документування.

3. Показано, що набір ШК-знаків (символіку) багатоколірного завадостійкого штрихового коду слід створювати так, щоб вектор (цифровий еквівалент) кожного ШК-знака був кодовим словом деякого многозначного коректувального коду, здатного виправляти одно- або двократні помилки в межах ШК-знака та частково виявляти багатократні помилки. Запропоновано використовувати для цього многозначний код Хемінга або многозначний код БЧХ з мінімальною кодовою відстанню 5.

   Розроблено метод синтезу символіки (множини ШК-знаків) заданої потужності та колірності завадостійкого штрихового коду, який грунтується на тому, що цифрові еквіваленти (вектори) багатоколірних ШК-знаків є кодовими словами многозначного коректувального коду, здатного виправляти одно-або двократні помилки (ушкодження) в межах кожного ШК-знака. Це забезпечує достовірне відтворення даних при зчитуванні ШК-знаків з носія або виявлення значної частини багатократних ушкоджень елементів ШК-знака.

   Корекція (виявлення) помилок у ШК-знаках утворює нижній рівень забезпечення завадостійкості багатоколірних штрихкодових зображень (рівень ШК-знаків).

4. Показано, що при застосуванні багатоколірних штрихових кодів ускладнюються процеси декодування штрихкодових позначок, а, отже,

необхідно вживати додаткових заходів щодо забезпечення завадостійкості багатоколірних штрихових зображень.

Запропоновано метод підвищення завадостійкості багатоколірних штрихових кодів при їх використанні в медичних інформаційних системах, який грунтується на застосуванні дворівневого контролю ушкоджень (спотворень), що виникають при скануванні багатоколірного штрихкодового зображення. На нижньому рівні контролю – рівні штрихкодових знаків, застосовується многозначний коректувальний код, що виправляє однократні (код Хемінга) або двократні (код БХЧ) ушкодження у межах кожного штрихкодового знака, а також виявляє значну частину помилок більшої кратності. На верхньому рівні контролю – рівні усієї штрихкодової позначки, застосовується коректувальний код Ріда-Соломона, який здатний виправляти ушкодження двох видів – помилки та стирання; на виправлення кожної помилки витрачаються два контрольні розряди, а на виправлення одного стирання – один контрольний розряд. Факт виявлення багатократної помилки у межах штрихкодового знака сприймається на верхньому рівні контролю як стирання, і за рахунок цього досягається ефект поліпшення завадостійкості штрихкодової позначки. При застосуванні на нижньому рівні контролю многозначного коду Хемінга завадостійкість штрихкодового зображення зростає на (12-25)%, а многозначного коду БЧХ – на (35-45)%.

5. Показано, що чільне місце в оцінюванні якості розробленої програмної системи посідає оцінювання її надійності.

Для дослідження характеристик надійності програмного забезпечення запропоновано організацію моніторингу функціонування розробленого програмного продукту у складі медичної інформаційної системи за п’ятьма показниками: кількість запитів за хвилину, час обробки запиту, HTTP-коди відповідей сервера, відсоток використання процесорних тактів, відсоток використання оперативної пам’яті.

Дослідження проводились методом навантажувального тестування на основі аналізу метрик роботи системи. Виконані дослідження на основі метрик показали стійкість та стабільність роботи розробленої програмної системи в умовах високих навантажень та її надійність.

Доведено, що використання процедури збору і аналізу метрик роботи системи є ефективним інструментом оцінювання характеристик розробленого програмного забезпечення.

6. Запропоновано емпіричний спосіб оцінювання кількості апаратних ресурсів, які необхідні для розгортання кластера сервісу генерування штрихкодових зображень, здатного задовольняти заздалегідь задекларовані цілі рівня послуг, який грунтується на аналізі метрик при використанні навантажувального тестування.

Так, якщо задекларованими цілями рівня послуг є 5000 запитів за хвилину з середнім часом обробки запиту до 3000 мілісекунд для текстових повідомлень завдовжки до 1500 алфавітно-цифрових символів, то кількість вузлів сервісу генерування багатоколірних штрихкодових зображень становитиме 46 вузлів, де вузлом вважається обчислювальний засіб з процесором 2,4 GHz 8-Core Intel Core i9 та оперативною пам'яттю 32 GB.

Такі апаратні ресурси (46 вузлів) є достатніми для забезпечення надійної роботи розробленого програмного забезпечення у складі медичної інформаційної системи в межах країни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хвищун А. І. Принципи формування єдиної медичної інформаційної системи великого міста / А. І. Хвищун, В. О. Качмар // Медична інформатика та інженерія. – 2009. - № 3. – С. 39 – 47.
2. Мельникова Н. І. Аналітичний огляд засобів програмного забезпечення в медичній галузі / Н. І. Мельникова, Н. Б. Шаховська // Вісник Національного університету "Львівська політехніка". – 2010. - № 673. – С. 146 – 153.
3. Міхнова А. В. Модель спеціалізованої медичної інформаційної системи служби крові / А. В. Міхнова, Д. К. Міхнов, К. С. Чиркова // Вісник КрНУ імені Михайла Остроградського. – 2019. – Випуск 5 (118). – С. 75 – 82.
   DOI: 10.30929/1995-0519.2019.5.75-82
4. Mikhnova A. Information support model of production transfusion processes / A. Mikhnova, D. Mikhnov, K. Chyrkova // Eastern-European Journal of Enterprise Technologies. – 2016. Vol. 3/3 (81). – P. 36 – 43.
   DOI: 10.15587/1729-4061.2016.71673
5. Жалій Н. А. Самостійне програмне забезпечення як медичний виріб: особливості його класифікування та оцінювання відповідності // Стандартизація, сертифікація, якість. – 2020. - № 4. – С. 29 – 33.
6. Pashkov V. Medical device software: defining key terms / V. Pashkov, N. Gutorova, A. Harkusha // Wiadomosci Lekarskie. – 2016. – Vol. 6. – P. 813 – 817.
7. Азархов О. Ю. Процедура отримання діагностичної інформації для медичних інформаційних систем / О. Ю. Азархов, А. П. Моторний, Д. Х. Штофель // Український журнал телемедицини та телематики. – 2011. - Т. 9, № 2. – С. 161 – 165.
8. Семенець А. В. Розробка платформи системи підтримки прийняття рішень для медичної інформаційної системи з відкритим кодом

OpenEMR / А. В. Семенець, В. П. Марценюк // Медична інформатика та інженерія. – 2015. - № 3. – С. 22 – 40.

9. Aminpour F. Utilization of open source electronic health record around the world: a systematic review / F. Aminpour, F. Sadoughi, M. Ahamdi // Journal of research in medical sciences: the official journal of Isfahan University of Medical Sciences. – 2014. -Vol. 19, № 1. – P. 57 -64.
10. Коваленко О. С. Мобільні застосунки у структурі сучасних медичних інформаційних систем / О. С. Коваленко, Л. М. Козак, О. О. Романюк, Т. А. Маресова, Л. В. Ненашев, Г. І. Финяк // УСиМ. – 2018. - № 4. – С. 57 – 69.

DOI https://doi.org/10.15407/usim.2018.04.0057

11. Dicianno B. E. Perspectives of the Evolution of Mobile (mHealth) Technologies and Application to Rehabilitation / B. E. Dicianno, B. Parmanto, A. D. Fairman, Th. M. Crytzer, D. X. Yu, G. P. Derek, C. A. Petrazzi // Physical Therapy. – 2015. – Vol. 95, № 3,. P. 397 – 405.
12. Telenyk S. Architecture and conceptual bases of cloud IT infrastructure management / S.telenyk, E.Zharikov, O.Rolik // Advances in Intelligent Systems and Computing. – 2017. – CSIT. P.41-62.
13. Telenyk S. Modeling of the data center resource management using reinforcement learning / S. Telenyk, E. Zarikov, O. Rolik // 2018 International Scientific-Practical Conference Problems of Infocommications Science and Tehnology (PIC S&T), P.289-296.
14. Zharikov E. Adaptive workload forecadting in cloud data centers / E. Zarikov, S. Telenyk, P. Bidyuk // Journal of Grid Computing. – 2020 – N 3. – P. 149-168.
15. Hekimian-Williams C. Accurate localization of RFID tags using phase difference / C. Hekimian-Williams, B. Grant, X. Liu, Z. Zhang, P. Kumar // Proceedings of the IEEE International Conference on RFID, 2010, P. 89–96.
16. Yang L. Tagoram: real-time tracking of mobile RFID tags to high precision using cots devices / L. Yang, Y. Chen, X.Y. Li, C. Xiao, M. Li, Y. Liu //

Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, 2014. – P. 237–248.

17. Chen L., A directed graphical model for linear barcode scanning from blurred images. || Asian Conf. on Computer Vision, Springer. – 2012. – P. 524–535.
18. Hua Yang. Automatic barcode recognition method based on adaptive edge detection and a mapping model // Journal of Electronic Imaging. – 2016. – Vol. 25(5). – P. 118-134.
19. EAN-13. Barcode Specifications. https://barcode1.com.au/ean-13-specifications
20. ДСТУ 3146-95. Коди та кодування інформації. Штрихове кодування. Маркування об`єктів ідентифікації. Штрихкодові позначки EAN. https://zakon.rada.gov.ua/rada/show/n0005699-96#Text
21. Kim Y. J., Lee J. Y. Algorithm of a Perspective Transform-Based PDF417 Barcode Recognition. Wireless Personal Communications, – 2016. – Vol. 89(3). – P. 893–911. DOI:10.1007/s11277-016-3171-6
22. Code 128/GS1-128 Barcode. https://www.barcodefaq.com/1d/code-128/#Code-128CharacterSet
23. ISO/IEC 18004:2015 Information technology – Automatic identification and data capture techniques – QR Code bar code symbology specification. https://www.iso.org/standard/62021.html
24. USS – I2/5. – AIM USA. – Pittsburgh, PA, 1986. – 20 p.
25. Two-Dimensional Bar Code Specifications. – Tiger Bar Code Systems, Inc. – 199. – 6 p.
26. Two-Dimensional Codes. – Data Capture Institute, 1995. – 30 p.
27. USS – Code 16K. – AIM USA. – Pittsburgh, PA, 1992. – 26 p.
28. USS – Maxi Code. – AIM USA. – Pittsburgh, PA, 1992. – 20 p.
29. Kaywa QR Code. https://qrcode.kaywa.com

30. Bagherinia H. A theory of color barcodes / H. Bagherinia, R. Manduchi // In IEEE Int. Conf. Comp. Vision Workshops, 2011, Nov. 6-13, pp. 806-813. DOI: 10.1109/ICCVW.2011.6130335
31. Barrus J. Embedding barcode data in an auxiliary field of an image file / J. Barrus an. G.J. Wolf // U.S. Patent 7 150 399, Dec. 19, 2006.
32. Tutsuya O. Layered two-dimensional code, creation method thereof, and read method / O. Tutsuya, M. Kazuhiro // US Patent Application, 20090166418, 2006.
33. High capacity color barcodes / Microsoft Research, May 2010. Режим доступу: https://www.microsoft.com/en-us/research/project/high-capacity-color-barcodes-hccb/
34. Grillo A. High capacity colored two dimensional codes / A. Grillo, A. Lentini, M. Querini, G. Italiano // In Proc. Int. Multiconf on Conp. Science Inf. Tech., pp. 709-716, 2010.
35. Parikh D. Localization and segmentation of a 2D high capacity color barcode / D. Parikh, G. Jancke // In IEEE Workshop on Applications of Computer Vision, 2008, Jan.7.
36. Wang F. Color-constant information embedding / F. Wang, R. Manduchi // In Proc. IEEE Workshop on Color and Reflectance in Imaging and Computer Vision, 2010.
37. Miller E. Color eigenflows: Statistical modeling of joint color changes / E. Miller, K. Tieu // In Proc. of Int. Conf. on Comp. Vision (ICCV), vol.1, pp. 607-614, 2001.
38. Cattrone P. Two-dimensional color barcode and method of generating and decoding the same / P. Cattrone // U.S. Patent 7 478 746, Jan. 20, 2009.
39. Querini M. 2D Color Barcodes for Mobile Phones / M. Querini et.al. // International Journal of Computer Science and Applications, vol. 8.N.1, pp. 136-155, 2011.

40. Peterson W. Error-Correcting Codes / W. Wesley Peterson, E. J. Weldon, Jr.//. – Second Edition : MIT, 1972. – 549 p.
41. Blahut R. Theory and Practice of Error Control Codes / Richard E. Blahut //. – MA: Addison-Wesley, 1983. – 500 p.
42. Касами Т. Теория кодирования /Т. Касами, Н. Токура, Е. Ивадари, Я. Инагаки. – М.: Мир, 1978. – 576 с.
43. Sun H.-Y. The application of barcode technology on logistics and warehouse management / Hong-Ying Sun // First International Workship on Education Technology and Computer Science, 7–8 March 2009: proceedings. Wuhan, IEEE, 2009, Vol. 3.
44. Kaminsky M.A. Barcode scanning device / M.A. Kaminsky, J. Choi, S. Lim, M.C. Palmer // U.S. Patent USD710362S1, applicant Motorola Solutions Inc. №US29/458,380; appdate 19.06.2013 ; pubdate 05.08.2014.
45. Wang Q. Rain Bar: Robust Application-driven Visual Communication using Color Barcodes / Q. Wang, M. Zhou, K. Ren, T. Lei, J. Li, Z. Wang // 2015 IEEE 35th International Conference on Distributed Computing Systems: Columbus, Ohio, USA, June 29 – July 02, 2015: IEEE Proceedings. – 2015. – P. 537 – 546. DOI:10.1109/ICDCS.2015.61
46. Blasinski H. Color barcodes for mobile applications: a per channel framework / H. Blasinski, O. Bulan, G. Sharma // US Patent № US 9,111,186 B2. – 2015.
47. Bagherinia H., Manduchi R. High information rate and efficient color barcode decoding // Proc. of the 12th International Conference on Computer Vision. – 2012. – Vol. 2. – P. 482-491. https://doi.org/10.1007/978-3-642-33868-7-48.
48. Blasinski H., Bulan O., Sharma G. Per-colorant-channel color barcodes for mobile applications: an interference cancellation framework // IEEE Transactions on Image Processing. – 2013. – Vol. 22, № 4. – P. 1498-1511.

49. Querini M., Italiano G.F. Color Classifiers for 2D Color Barcodes // Proc. of the 2013 Federal Conference on Computer Science and Information Systems. – 2013. – P. 611-618.
50. Ramya M., Jayasheela M. Improved Color QR Codes for Real Time Applications with High Embedding Capacity // International Journal of Computer Application. – 2014. – Vol. 91, № 8. – P. 8-12.
51. Pang P., Wu J., Long C. CodeCube: A Multi-Layer Color Barcode for Mobile Social Applications // Proc. of the 29th Chinese Control and Decision Conference (CCDC). – 2017. – P. 7713-7718.
52. Elhami G. Three-Dimensional Cubic Barcodes / G. Elhami, A. Scholefield, M. Vetterli // IEEE Transactions on Image Processing. – 2022. – Vol. 31. – P. 3166 – 3181. DOI:10.1109/TIP.2021.3120049
53. Querini M. Reliability and Data Density in High Capacity Color Barcodes / M. Querini, G.F. Italiano // Computer Science and Information Systems. – 2014. – Vol. 11(4) . – P. 1595 – 1615. DOI:10 / 2298 / CSIS131218054Q
54. Firmani D. Engineering Color Barcode Algorithms for Mobile Applications / D. Firmani, G.F. Italiano, M. Querini // J. Gudmundsson, J. Katajainen. Expiremental Algorithms, International Symposium on Experimental Algorithms SEA'2014. – Lecture Notes in Computer Science, Springer. – Vol. 8504. – P. 211 – 222.
DOI: https://doi.org/10.1007/978-3-319-07959-2-18
55. Wara A. Enhancing user experience using mobile QR-code application / A. A. Wara, S. Dugga // International Journal of Computer and Information Technology. – 2014. – Vol. 3(6). – P.1310 – 1315.
56. Jahagirdar K. QR Code with colored image / K. S. Jahagirdar, S. B. Borse // International Journal of Computer Applications. – 2015. – Vol. 115(16). – P. 38 – 41.
57. Ramya M. Color QR Codes for Real Time Applications with High Embedding Capacity / M. Ramya, M. Jayasheela // International Journal of Computer Application. – 2014. – Vol. 91(8). – P. 8 – 12.

DOI : https://doi.org/10.5120/15899-4889

58. Abas A. Increasing data storage of colored QR code using compress, multiplexing and multilayered technique / A. Abas, Y. Yusof, R. Din, F. Azali, B. Osman // Bulletin of Electrical Engineering and Informatics. – 2020. – Vol. 9(6). – P. 2555 – 2561.
DOI : https://doi.org/10.11591/eei.v9i6.2481

59. Жураковський Б. Ю. Багатовимірні штрихові коди / Б. Ю. Жураковський, В. А. Дружинін // Міжвідомчий науково-технічний збірник «Адаптивні системи автоматичного управління». – 2018. – № 2(33). – С. 15 – 31.
DOI: https://doi.org/10.20535/1560-8956.33.2018.164669

60. Berchtold W. JAB Code – A Versatile Polychrome 2D Barcode / W. Berchtold, H. Liu, M. Steinebach, D. Klein, T. Senger, N. Thence // Electronic Imaging. – 2020. – Vol. 2020(3). – P. 207 – 212.
DOI : https://doi.org/10.2352/ISSN.2470-1173.2020.3.MOBMU-207

61. Wang G. Security Mechanism Improvement for 2D Barcodes using Error Correction with Random Segmentation / Wang G., Yang Z., Chen J. // The 6th International Conference on Information Technology: IoT and Smart City: Bhubaneswar, India, December 19 – 21, 2018 : Proceedings. – IEEE, 2018. – P. 104 – 109.
DOI : https://doi.org/10.1145/3301551.3301593.

62. Wiegers K. Software Requirements (2nd Edition) (Developer Best Practices) / Redmond, WA: Microsoft Press, 2003. – 544 p.

63. Stellman A. Applied Software Project Managament /A. Stellman, I. Greene. – Cambridge, MA: O'Reilly Media, 2005.

64. Berenbach B. Software & Systems Requirements Engineering: in Practice/ B. Berenbach, D. Paulish, j. Karzmeier, A. Rudorfer. – New York: McGraw-Hill Professional, 2009.

65. Laplante P. Requirements Engineering for Software and Systems/ Redmond, WA: CRC Press, 2009. – 264 p.

66. Dick J. Requirements Engineering (4rd ed) / J. Dick, E. Hull, K. Jackson. – Springer, 2017.
https://doi.org/10.1007/978-319-61073-3.

67. Грицюк Ю.І. Особливості формування вимог до програмного забезпечення/ Ю.І. Грицюк, О.А. Нємова// Науковий вісник НАТУ України. – 2018. – Т. 28, № 7. – С. 135-148.
DOI : https://doi.org/10.15421/ 40280727.

68. Грицюк Ю.І. Особливості управління процесом розроблення вимог до програмного забезпечення / Ю.І. Грицюк, О.А. Нємова// Науковий вісник НАТУ України. – 2018. – Т. 28, № 8. – С. 161-169.
DOI : https://doi.org/10.15421/ 402808.

69. Грицюк Ю.І. Формалізація процесу управління ризиками розроблення програмного забезпечення / Ю.І. Грицюк, В.С. Далявський// Науковий вісник НАТУ України. – 2018. – Т. 28, № 11. – С. 135-153.
DOI : https://doi.org/10.15421/ 40281124.

70. Chung L., On Non-Functional Requirements in Software Engineering / L. Chung, J.C.S. do Prado Leite // Lecture Notes in Computer Science. – Springe. – 2009. Vol. 5600.
DOI.org/10.1007/978-3-642-02463-4-19

71. Maalej W., Toward data-driven requirements engineering. / W. Maalej, M. Nayebi, T. Johann, G. Ruhe. – IEEE Software. – 2016. – Vol. 33, № 1. – P. 48–54.

72. Mahmoud A., Detecting classifying and tracing non-functional software requirements. /A. Mahmoud, W. Grant. – Requirements Engineering. – 2016. – Vol. 21, № 3, P. 1–25.

73. Mengmeng L, Automatic Classification of Non-Functional Requirements from Augmented App User Reviews / L. Mengmeng, L. Peng // Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineerin. – 2017. P. 344–353. DOI.org/10.1145/3084226.3084241.

74. OMG® Unified Modeling Language®, Version 2.5.1. Object Management Group. –2017. 796 p.
75. Saher N. A Review of Requirement Prioritization Techniques in Agile Software Development. / N. Saher, F. Baharom, R. Romli // Proceedings of the Knowledge Management International Conference (KMICe. – 2018. P. 242–247.
76. Божуха Л.М. Конспект лекцій з дисципліни “Аналіз вимог до програмного забезпечення” для студентів напряму підготовки 6.050103 “Програмна інженерія”/ Дніпродзержинськ. ДДТУ: 2015. – 94 с.
77. Козак О.Л. Опорний конспект лекцій з курсу “Аналіз вимог до програмного забезпечення” для студентів напряму підготовки “Програмна інженерія”/О.Л. Козак. – Тернопіль. – 2011. – 56 с.
78. Сулема Є. С., Онай М. В., Дичка А. І. Алгоритмічне забезпечення завадостійкості багатоколірних штрихкодових знаків на основі поля GF(p) // Наукові вісті КПІ. – 2021. – № 1(132). – С. 50-62.
DOI :10.20535/kpisn.2021.1.231210
79. Сулема Є. С., Дичка А. І. Аналіз коректувальних властивостей многозначних кодів Хемінга // Чотирнадцята наукова конференція магістрантів та аспірантів “Прикладна математика та комп’ютинг (ПМК’2021)”, Київ, 17 - 19 листопада 2021 р. Збірник тез доповідей Нац. техн. ун-т України “Київ. політехн. ін-т ім. Ігоря Сікорського”, Київ: Просвіта. - 2021. - С. 89-97.
ISBN 978-617-7010-18-9.
80. Сулема Є. С., Онай М. В., Дичка А. І. Забезпечення завадостійкості багатоколірних штрихкодових знаків на основі поля GF(p^m) // Системні технології. - 2021. Вип.1 (132), С. 31-50.
DOI :10.34185/1562-9945-1-132-2021-03.
81. Sulema Ye., Drozdenko L, Dychka A. Synthesis of the Symbologies of Multicolor Interference-Resistant Bar Codes on the Base of Multi-Valued

BCH Codes // Radio Electronics, Computer Science, Control. – 2022. – Vol 4. P. 107-118.
DOI :10.15588/1607-3274-2022-4-9.

82. Дичка А. І. Застосування многозначних кодів БЧХ для підвищення завадостійкості багатоколірних штрихових кодів // “Світ наукових досліджень. Випуск 13”: матеріали Міжнародної мультидисциплінарної наукової інтернет-конференції, (м. Тернопіль, Україна – м. Переворськ, Польща, 25-26 жовтня 2022 р.). – ГО “Наукова спільнота”; WSSG w Przeworsku. – 2022. - С. 69 - 78.

83. Сулема Є. С., Дичка А. І. Підвищення завадостійкості багатоколірних штрихкодових зображень // Системні технології. – 2023. – Вип. 2 (145). – С. 105-124.
DOI :10.34185/1562-9945-2-145-2023-10.

84. Сулема Є. С., Дичка А. І. Програмна система автоматичної ідентифікації та розподіленого зберігання медичних даних пацієнтів // Системні технології. – 2023. – Вип. 3 (146). – С. 134-148.
DOI :10.34185/1562-9945-3-146-2023-13.

85. Грицюк Ю.І. Засіб для визначення якості програмного забезпечення методами метричного аналізу /Ю.І. Грицюк, О.Т. Андрущакевич// Науковий вісник НЛТУ України. – 2018. – Т. 28, № 6. – С.159-171.

86. Грицюк Ю.І.Візуалізація результатів експертного оцінювання якості програмного забезпечення з використанням полярних діаграм /Ю.І. Грицюк, А.Ю. Бучковська// Науковий вісник НЛТУ України. – 2017. – Т. 27, № 10. – С.137-145.
DOI : https://doi.org/10.15421/40271025

87. Грицюк Ю.І. Використання пелюсткових діаграм для візуалізації результатів експертного оцінювання якості програмного забезпечення /Ю.І. Грицюк, В.С. Далявський// Науковий вісник НЛТУ України. – 2018. – Т. 28, № 9. – С.95-104.
DOI : https://doi.org/10.15421/40280919

88. Грицюк Ю.І. Система комплексного оцінювання якості програмного забезпечення // Науковий вісник НЛТУ України. – 2022. – № 2. – С.81-95.
89. Грицюк П.Ю. Забезпечення якості програмного продукту за стандартом IEEE 730-2014 в межах життєвого циклу реалізації проекту / П.Ю. Грицюк, А.В. Іванишин, Ю.І. Грицюк// Науковий вісник НЛТУ України. – 2023. – № 2. – С.101-117.
90. Hrytsiuk Y. Software Development Risk Modeling / Y. Hrytsiuk, P. Grytsyuk, T. Dyak, H. Hrynyk // 2019 IEEE 14th International Conference on Computer Science and Information Technologies (CSIT). – 2019. – T.2. – P. 134-137.
91. Говорущенко Т. О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення. – Хмельницький національний університет. – 2017. – 310 с.
92. Говорущенко Т. О. Дослідження відомих моделей оцінювання характеристик програмного забезпечення // Вісник Хмельницького національного університету. – 2013. - № 1. – С.117-121.
93. Говорущенко Т. О. Визначення ефективності метрик якості на етапі проектування програмного забезпечення // Вісник Хмельницького національного університету. – 2012. – № 2. – С.149-155.
94. Zasornova I. Study of software testing tools according to the testing level / I. Zasornova, T. Hovorushchenko, O. Voichur // Computer Systems and Information Technologies . – 2023. - # 1. - P. 38-46.
DOI: https://doi.org/10.31891/csit-2023-1-5.
95. Lebiga M. Neural-network model of Software Quality Prediction Based on Quality Attributes / M. Lebida, T. Hovorushchenko, M. Kapustian // Computer Systems and Information Technologies. 2022. - №1. – P. 69-74.
DOI: https://doi.org/10.31891/csit-2022-1-9.

96. Hovorushchenko T. Method for forecasting the level of software quality based on quality attributes / T. Hovorushchenko, D. Medzatyi, Y. Voichur, M. Lebiga // Journal of Intelligent and Fussy Systems. - 2023. - № 44(3). – P. 3891-3905.
97. Поморова О. В. Сучасні проблеми оцінювання якості програмного забезпечення / О. В. Поморова, Т. О. Говорущенко // Радіоелектронні і комп'ютерні системи. – 2013. - № 5 (64). – С.319 - 327.
98. Hovorushchenko T. Methodology of Evaluating the Sufficiency of Information for Software Quality Assessment According to ISO 25010 // Journal of Information and Organizational Sciences. - 2018. – Vol. 42, № 1. – P. 63 - 85.
99. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Systems and software quality models. [Introduced 01.03.2011]. Geneva (Switzerland). - 2011. - 34 p. (International standard).
100. ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. Geneva (Switzerland). - 2016. - 45 p. (International standard).
101. Lakra K. Application of metaheuristic techniques in software quality prediction: a systematic mapping study / K. Lakra, A. Chug // International Journal of Intelligent Engineering Informatics. - 2021. – Vol. 9, № 4. – P. 355-399.
102. Мищенко В. О. CASE-оценка критических программных систем. В 3-х томах. Т.1. Качество / В. О. Мищенко, О. В. Поморова, Т. А. Говорущенко; под ред. В. С. Харченко. – Х.: Нац.аэрокосмический университет «ХАИ», 2012. – 201 с.
103. Goyal S. Comparison of Machine Learning Techniques for Software Quality Predition // International Journal of Knowledge and Systems Science. - 2020. – Vol. 11, № 2. – P. 20-40.

104. Masood M. Early Software Quality Prediction Based on Software Requirements Specification Using Fuzzy Inference System / M. Masood, M. Khan // Lecture Notes in Artificial Intelligence. - 2018. – Vol. 10956. – P. 722 - 733.
105. Кузь М. В. Прогнозування якості програмних засобів на основі аналізу якості вимог / М. В. Кузь, Б. С. Незамай, В. А. Ровінський, Н. Д. Подубинська // Методи та прилади контролю якості. - 2023. - № 1(50). – С. 101 - 112.

ДОДАТКИ

Додаток А

Класифікація та характеристики найбільш поширених двоколірних лінійних штрихових кодів

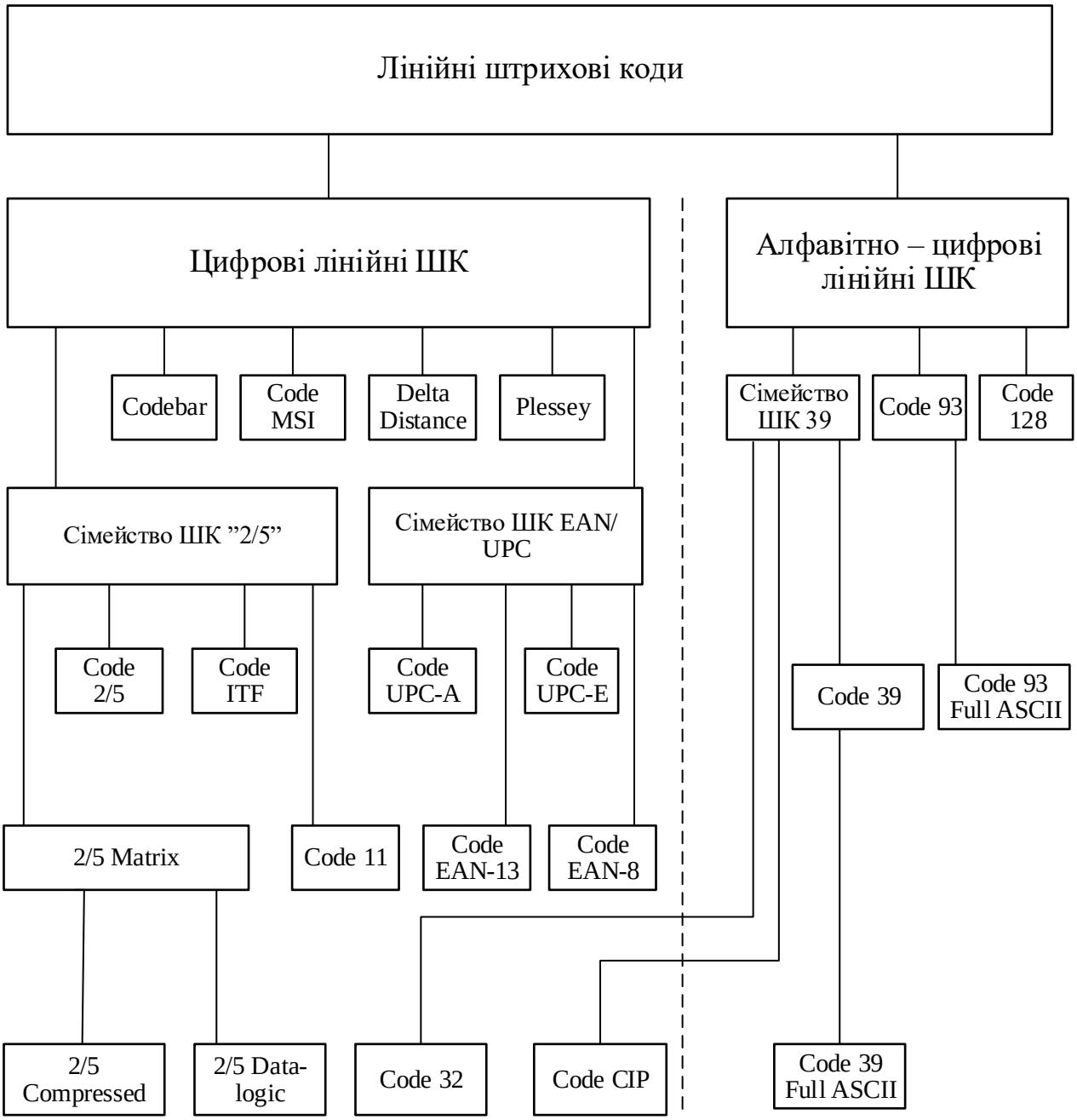


Рис. А.1. Класифікація лінійних штрихових кодів

Таблиця А. 1

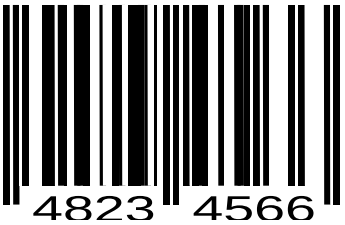
Деякі характеристики найбільш поширених лінійних штрихових кодів

Назва ШК	Склад символіки	Структура ШК-знаків	Спосіб подання даних	Інформаційна щільність ШК-знаків, біт/мод	Кількість контроль-них символів	Рік розроб-лення
2/5 Code	10 цифр	5Ш, 4П	Ш	0,28	1	1968
Code ITF	10 цифр	$\frac{5Ш, 5П}{2}$	Ш+П	0,47	1	1972
2\5 Matrix	10 цифр	3Ш, 2П	Ш+П	0,42	1	1975
Code 11	10 цифр, 1 спецсимвол	3Ш, 2П	Ш+П	0,43	1 або 2	1977
Codabar	10 цифр, 6 спецсимволів	4Ш, 3П	Ш+П	0,39	1	1975
Code MSI	10 цифр	4Ш, 4П	Комбінацією Ш і П	0,28	1 або 2	1976
Delta Distance	10 цифр, 4 алфавітні символи	6Ш, 5П	П	0,22	1	1977
Plessey	10 цифр, 6 алфавітних символів	4Ш, 4П	Комбінацією Ш і П	0,33	2	1978
UPC-A	10 цифр	2Ш, 2П	Комбінацією ширини елементів	0,45	1	1973
EAN-13	10 цифр	2Ш, 2П	Комбінацією ширини елементів	0,48	1	1976
Code 39	10 цифр, 26 букв, 7 спец-символів	5Ш, 4П	Ш+П	0,42	1	1974
Code 93	10 цифр, 26 букв, 7 спец-символів	3Ш, 3П	Комбінаціями ширини елементів	0,60	2	1982
Code 128	128 символів	3Ш, 3П	Комбінаціями ширини елементів	0,60 (текст); 1,21 (цифро-ва послідов-ність)	1	1981

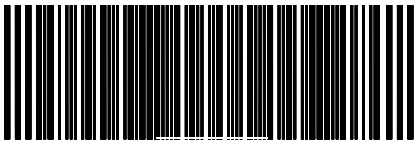
ШК UPC-A

ШК UPC-E

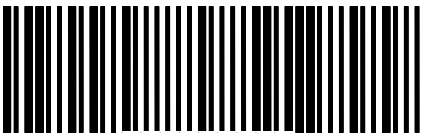
ШК EAN-13



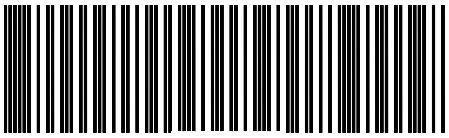
ШК EAN-8



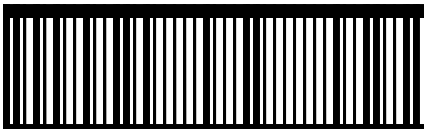
ШК Codabar



ШК MSI



ШК Delta Distance (IBM)



ШК Plessey

ШК 39

ШК 93

ШК 93 Full ASCII

ШК 128

Рис. А. 2. Приклади ШК-позначок деяких лінійних штрихових кодів

Додаток Б

Зовнішній вигляд та характеристики найбільш поширених двоколірних стекових штрихових кодів

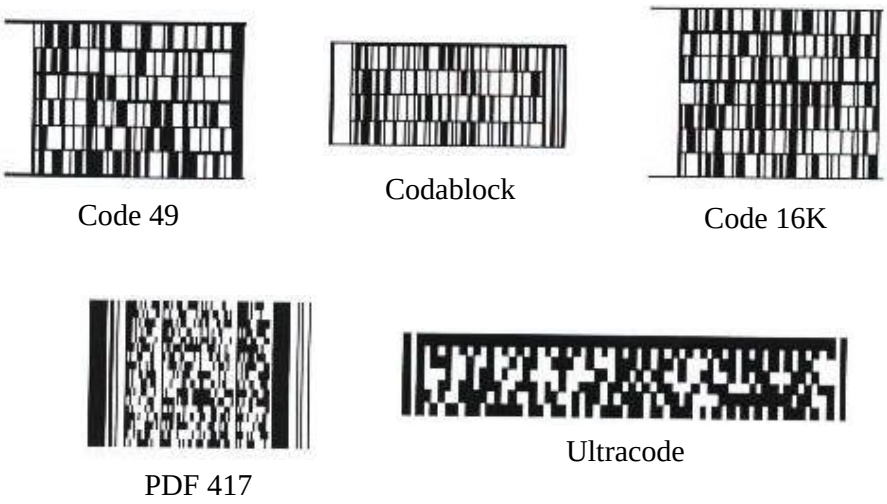


Рис. Б. 1. Зовнішній вигляд найбільш поширених стекових штрихових кодів

Таблиця Б. 1

Деякі характеристики найбільш поширених стекових штрихових кодів

Назва ШК	Підтримуваний алфавіт	Кількість рядків у ШК-позначці	Контроль помилок	Розробник	Рік розроблення
Code 49	ASCII	2 − 8	Контрольний символ у рядку (за mod49) та загальний контрольний символ (за mod2401)	Intermec Corporation (США)	1987
Codablok	ASCII ASCII(256)	2 − 44	Контрольні суми	Identcode-System (Німеччина)	1989
Code 16K	ASCII ASCII(256)	2 − 16	Два контрольні символи (за mod107)	Lazerlight System (США)	1989
PDF417	ASCII(256)	2 − 90	На основі Ріда-Соломона	Symbol Technologies (США)	1991

Додаток В

Зовнішній вигляд та характеристики найбільш поширених двоколірних матричних штрихових кодів

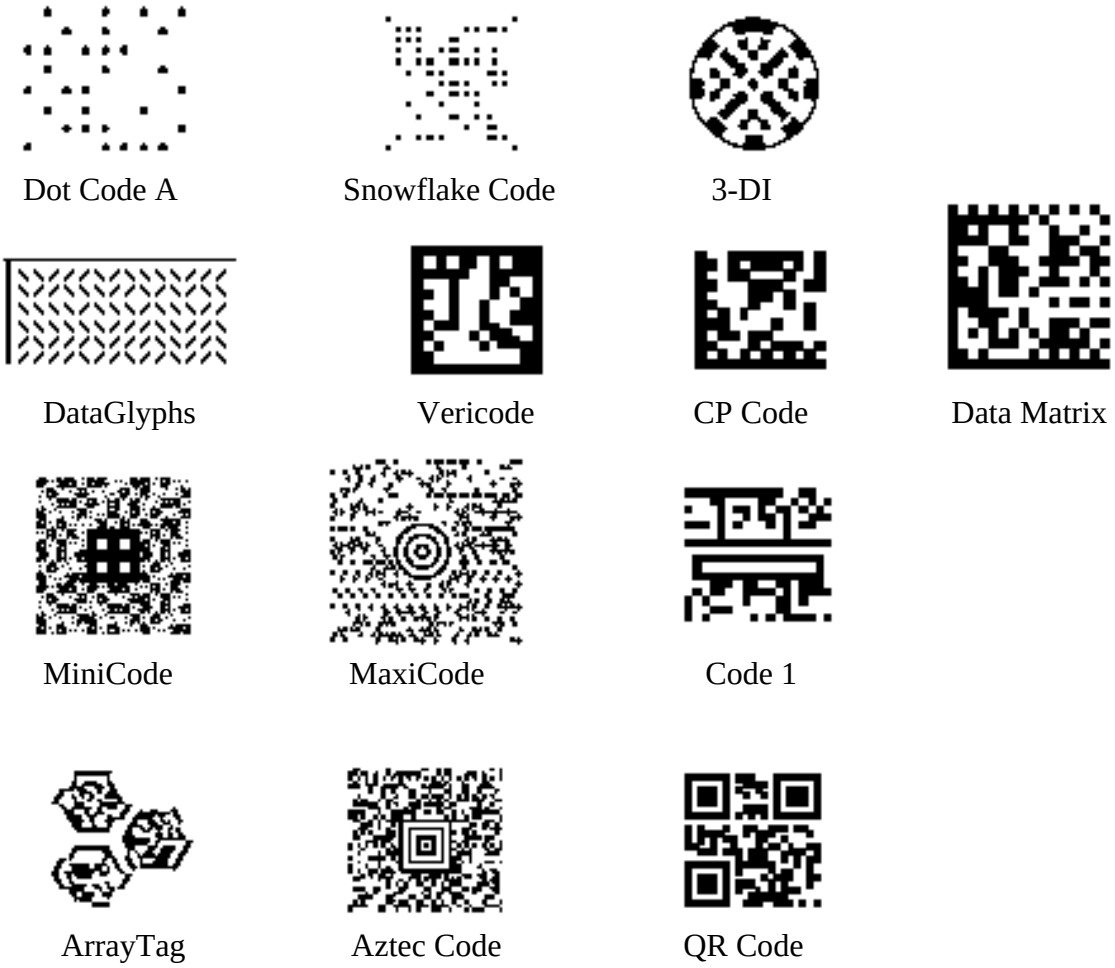


Рис. В. 1. Зовнішній вигляд найбільш поширених матричних штрихових кодів

Таблиця В. 1

Деякі характеристики найбільш поширених матричних штрихових кодів

Назва ШК	Максимальна інформацій-на ємність ШК-позначки		Контроль помилок	Розробник	Рік розроб-лення
	Алфавітно-цифрових символів	Цифрових символів			
Data Matrix	2335	3116	Код Ріда-Соломона	International Matrix (США)	1987
Code One	2218	3550	Код Ріда-Соломона	Lazerlight Systems (США)	1992
QR Code	4464	7366	Код Ріда-Соломона	Denso Wave (Японія)	1994
Aztec Code	3067	3832	Код Ріда-Соломона	Welch Allyn (США)	1995

Додаток Г

Здатність виявляти двократні помилки у багатоколірних ШК-знаках, синтезованих на основі деяких трійкових, п’ятіркових та сімкових кодів Хемінга

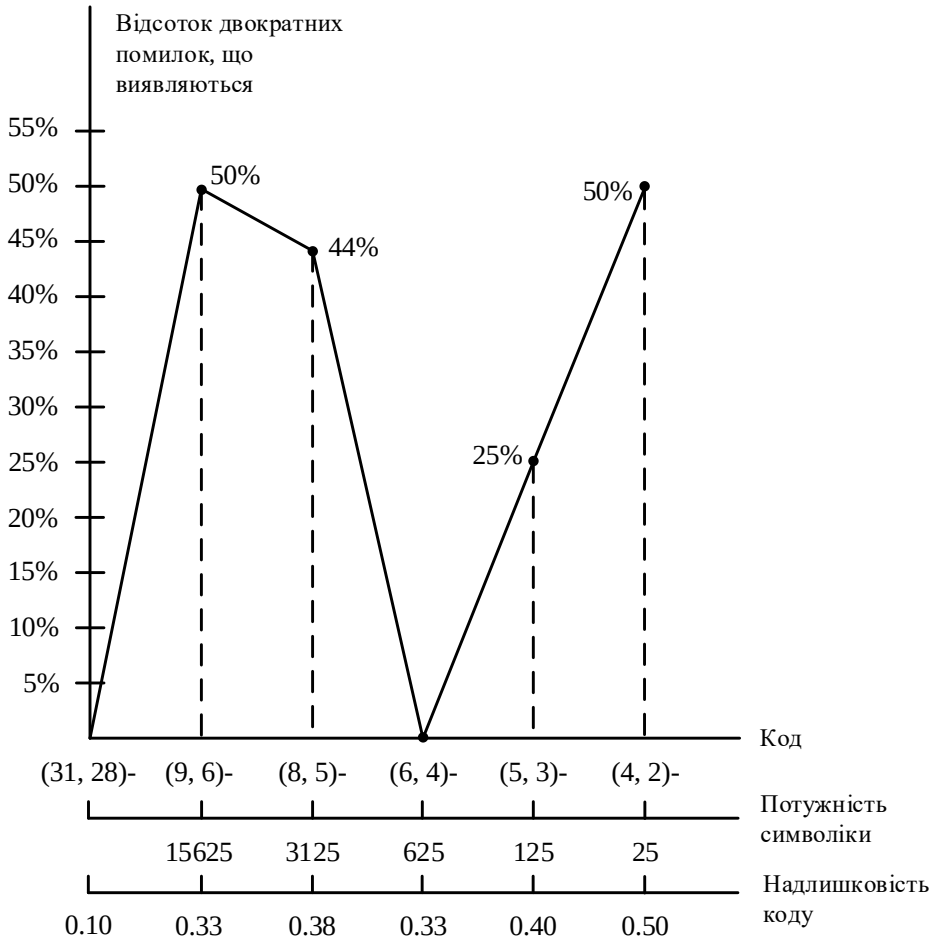


Рис. Г. 1. Показники виявлення двократних помилок у п’ятиколірних ШК-знаках, синтезованих на основі деяких п’ятіркових кодів Хемінга

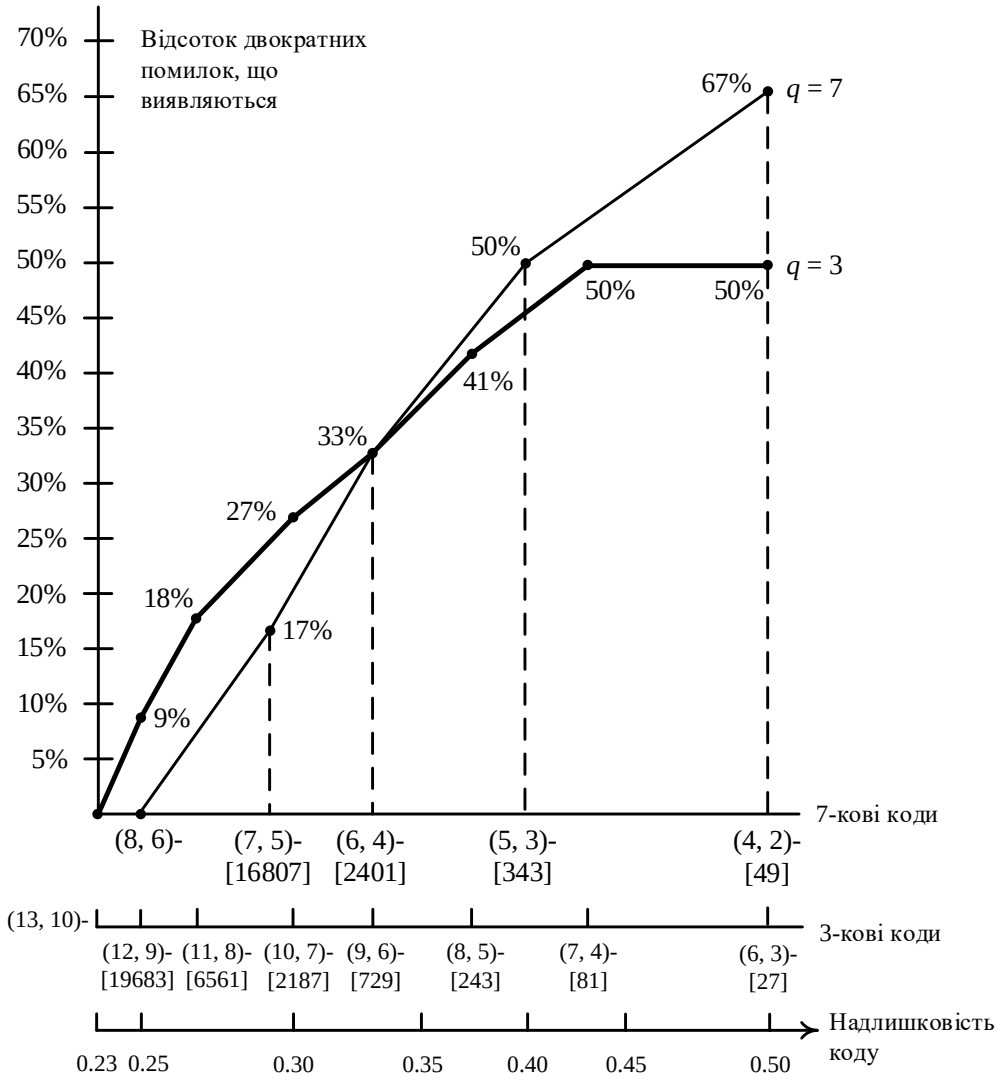


Рис. Г. 2. Показники виявлення двократних помилок у семиколірних (q = 7) та триколірних (q = 3) ШК-знаках, синтезованих на основі деяких трійкових та сімкових кодів Хемінга

Примітка. Коди упорядковані за параметром надлишковості, в квадратних дужках вказано потужність символіки, яку породжує відповідний многозначний код Хемінга.

Додаток Д

Показники виявлення багатократних помилок многозначними кодами Хемінга

Таблиця Д. 1

Відсоток багатократних помилок, які виявляються скороченими трійковими кодами Хемінга, похідними від повного (13, 10)-коду Хемінга

Код Хемінга	Потужність символіки	Кратність помилки				
		2	3	4	5	6
(12, 9)-	19683	9,1%	8,2%	6,8%	7,4%	7,7%
(11, 8)-	6561	18,2%	16,4%	13,6%	14,8%	15,3%
(10, 7)-	2187	26,7%	25,0%	20,4%	22,0%	23,2%
(9, 6)-	729	33,3%	34,5%	27,4%	28,6%	31,6%
(8, 5)-	243	41,1%	42,9%	34,8%	35,7%	38,5%
(7, 4)-	81	50,0%	51,4%	40,7%	44,6%	44,2%
(6, 3)-	27	50,0%	60,0%	50,0%	56,3%	37,5%

Таблиця Д. 2

Відсоток багатократних помилок, які виявляються скороченими четвірковими кодами Хемінга, похідними від повного (21, 18)-коду Хемінга

Код Хемінга	Потужність символіки	Кратність помилки				
		2	3	4	5	6
(10, 7)-	16384	46,7%	54,1%	52,0%	51,5%	51,3%
(9, 6)-	4096	44,4%	57,1%	57,9%	56,9%	55,3%
(8, 5)-	1024	50,0%	61,5%	61,6%	62,7%	59,5%
(7, 4)-	256	47,6%	61,6%	63,9%	70,8%	66,2%
(4, 2)-	16	33,3%	11,1%	22,2%	-	-

Примітка. (4, 2)-код є похідним від повного четвіркового (5, 3)-коду Хемінга

Таблиця Д. 3

Відсоток багатократних помилок, які виявляються скороченими п’ятірковими кодами Хемінга, похідними від повного (31, 28)-коду

Код Хемінга	Потужність символіки	Кратність помилки				
		2	3	4	5	6
(9, 6)-	15625	50,0%	63,7%	68,2%	71,7%	71,0%
(8, 5)-	3125	43,8%	58,7%	66,1%	74,4%	76,2%
(7, 4)-	625	64,3%	67,9%	74,3%	80,2%	78,1%
(6, 3)-	125	80,0%	77,5%	80,0%	83,2%	76,5%
(5, 2)-	25	92,5%	86,3%	82,2%	82,8%	-
(5, 3)-	125	25,0%	12,5%	17,2%	15,6%	-
(4, 2)-	25	50,0%	25,0%	34,4%	-	-

Примітка. (5, 3)-код та (4, 2)-код є похідними від повного (6, 4)-коду

Таблиця Д. 4

Відсоток багатократних помилок, які виявляються скороченими сімковими кодами Хемінга, похідними від повного (8, 6)-коду Хемінга

Код Хемінга	Потужність символіки	Кратність помилки				
		2	3	4	5	6
(7, 5)-	16807	16,7%	11,1%	12,5%	12,2%	12,3%
(6, 4)-	2401	33,3%	22,2%	25,0%	24,4%	24,5%
(5, 3)-	343	50,0%	33,3%	37,5%	36,6%	-
(4, 2)-	49	66,7%	44,4%	50,0%	-	-

Таблиця Д. 5

Відсоток багатократних помилок, які виявляються скороченими вісімковими кодами Хемінга, похідними від повного (9, 7)-коду Хемінга

Код Хемінга	Потужність символіки	Кратність помилки				
		2	3	4	5	6
(7, 5)-	32768	28,6%	20,4%	22,2%	21,8%	21,9%
(6, 4)-	4096	42,7%	30,6%	33,2%	32,7%	32,8%
(5, 3)-	512	57,1%	40,8%	44,3%	43,7%	-
(4, 2)-	64	71,4%	51,0%	55,4%	-	-

Додаток Е

Показники виявлення багатократних помилок многозначними кодами БЧХ

Таблиця Е. 1

Відсоток багатократних помилок, які виявляються скороченими трійковими кодами БЧХ, похідними від повного (26, 17)-коду БЧХ

Код БЧХ	Потужність символіки	Кратність помилки				
		3	4	5	6	7
(18,9)-	19683	86,6%	85,4%	84,7%	85,1%	85,0%
(17,8)-	6561	88,1%	87,0%	86,1%	86,5%	86,4%
(16,7)-	2187	89,2%	88,5%	87,5%	87,9%	87,7%
(15,6)-	729	90,3%	89,6%	88,8%	89,1%	89,0%
(14,5)-	243	92,0%	90,9%	90,2%	90,4%	90,3%
(13,4)-	81	94,0%	92,8%	91,7%	91,8%	91,6%
(12,3)-	27	94,1%	93,5%	93,1%	93,1%	92,8%
(8,3)-	27	46,4%	39,3%	29,5%	32,6%	34,4%

Примітка. Трійковий (8,3)-код БЧХ є повним кодом.

Таблиця Е. 2

Відсоток багатократних помилок, які виявляються скороченими четвірковими кодами БЧХ, похідними від повного (15, 9)-коду БЧХ

Код БЧХ	Потужність символіки	Кратність помилки				
		3	4	5	6	7
(13,7)-	16384	79,0%	73,4%	70,6%	70,4%	70,7%
(12,6)-	4096	83,5%	77,7%	75,1%	74,9%	75,1%
(11,5)-	1024	87,6%	82,1%	79,5%	78,8%	79,2%
(10,4)-	256	91,6%	86,0%	83,6%	82,5%	83,0%
(9,3)-	64	93,5%	88,6%	86,8%	85,5%	85,9%
(8,2)-	16	96,6%	92,6%	90,7%	88,1%	87,2%

Таблиця Е. 3

Відсоток багатократних помилок, які виявляються скороченими п’ятірковими кодами БЧХ, похідними від повного (24, 16)-коду БЧХ

Код БЧХ	Потужність символіки	Кратність помилки				
		3	4	5	6	7
(14,6)-	15625	91,7%	88,6%	88,5%	88,5%	88,6%
(13,5)-	3125	92,9%	90,0%	89,9%	90,0%	90,1%
(12,4)-	625	93,8%	91,3%	91,3%	91,4%	91,5%
(11,3)-	125	95,0%	92,5%	92,6%	92,7%	92,8%
(10,2)-	25	96,0%	93,6%	93,6%	93,9%	94,0%

Таблиця Е. 4

Відсоток багатократних помилок, які виявляються скороченими сімковими кодами БЧХ, похідними від повного (48, 40)-коду БЧХ

Код БЧХ	Потужність символіки	Кратність помилки			
		3	4	5	6
(13,5)-	16807	97,9%	97,9%	97,9%	97,9%
(12,4)-	2401	98,3%	97,6%	97,6%	97,5%
(11,3)-	343	98,7%	98,1%	98,0%	97,9%
(10,2)-	49	99,0%	98,5%	98,3%	98,3%

Додаток Є

Можливі вектори ШК-знаків чотириколірного завадостійкого ШК, утворені на основі четвіркового (9, 3)-коду БЧХ

0 0 0 (0)	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	X	X
0 0 1 (1)	0 0 1 2 2 1 1 3 1	0 0 1 2 2 1 1 3 1	X	
0 0 2 (2)	0 0 2 3 3 2 2 1 2	0 0 2 3 3 2 2 1 2	X	
0 0 3 (3)	0 0 3 1 1 3 3 2 3	0 0 3 1 1 3 3 2 3	X	
0 1 0 (4)	0 1 2 2 1 1 3 1 0	0 1 2 2 1 1 3 1 0	X	
0 1 1 (5)	0 1 3 0 3 0 2 2 1	0 1 3 0 3 0 2 2 1		
0 1 2 (6)	0 1 0 1 2 3 1 0 2	0 1 0 1 2 3 1 0 2		
0 1 3 (7)	0 1 1 3 0 2 0 3 3	0 1 1 3 0 2 0 3 3		
0 2 0 (8)	0 2 3 3 2 2 1 2 0	0 2 3 3 2 2 1 2 0	X	

0 2 1 (9)	0 2 2 1 0 3 0 1 1	0 2 2 1 0 3 0 1 1		
0 2 2 (10)	0 2 1 0 1 0 3 3 2	0 2 1 0 1 0 3 3 2		
0 2 3 (11)	0 2 0 2 3 1 2 0 3	0 2 0 2 3 1 2 0 3		
0 3 0 (12)	0 3 1 1 3 3 2 3 0	0 3 1 1 3 3 2 3 0		X
0 3 1 (13)	0 3 0 3 1 2 3 0 1	0 3 0 3 1 2 3 0 1		
0 3 2 (14)	0 3 3 2 0 1 0 2 2	0 3 3 2 0 1 0 2 2		
0 3 3 (15)	0 3 2 0 2 0 1 1 3	0 3 2 0 2 0 1 1 3		
1 0 0 (16)	1 2 2 1 1 3 1 0 0	1 2 2 1 1 3 1 0 0		X
1 0 1 (17)	1 2 3 3 3 2 0 3 1	1 2 3 3 3 2 0 3 1		
1 0 2 (18)	1 2 0 2 2 1 3 1 2	1 2 0 2 2 1 3 1 2		
1 0 3 (19)	1 2 1 0 0 0 2 2 3	1 2 1 0 0 0 2 2 3		X

1 1 0 (20)	1	3	0	3	0	2	2	1	0	1 3 0 3 0 2 2 1 0	
1 1 1 (21)	1	3	1	1	2	3	3	2	1	1 3 1 1 2 3 3 2 1	
1 1 2 (22)	1	3	2	0	3	0	0	0	2	1 3 2 0 3 0 0 0 2	
1 1 3 (23)	1	3	3	2	1	1	1	3	3	1 3 3 2 1 1 1 3 3	
1 2 0 (24)	1	0	1	2	3	1	0	2	0	1 0 1 2 3 1 0 2 0	
1 2 1 (25)	1	0	0	0	1	0	1	1	1	1 0 0 0 1 0 1 1 1	X
1 2 2 (26)	1	0	3	1	0	3	2	3	2	1 0 3 1 0 3 2 3 2	
1 2 3 (27)	1	0	2	3	2	2	3	0	3	1 0 2 3 2 2 3 0 3	
1 3 0 (28)	1	1	3	0	2	0	3	3	0	1 1 3 0 2 0 3 3 0	
1 3 1 (29)	1	1	2	2	0	1	2	0	1	1 1 2 2 0 1 2 0 1	
1 3 2 (30)	1	1	1	3	1	2	1	2	2	1 1 1 3 1 2 1 2 2	X

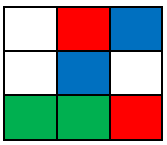
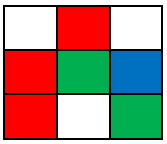
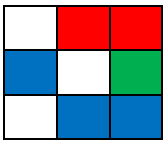
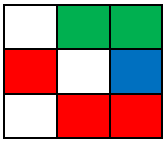
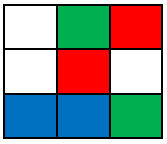
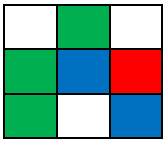
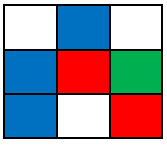
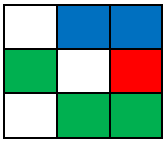
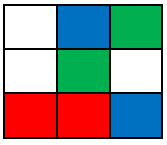
1 3 3 (31)	1	1	0	1	3	3	0	1	3	1 1 0 1 3 3 0 1 3			
2 0 0 (32)	2	3	3	2	2	1	2	0	0	2 3 3 2 2 1 2 0 0			X
2 0 1 (33)	2	3	2	0	0	0	3	3	1	2 3 2 0 0 0 3 3 1			X
2 0 2 (34)	2	3	1	1	1	3	0	1	2	2 3 1 1 1 3 0 1 2			
2 0 3 (35)	2	3	0	3	3	2	1	2	3	2 3 0 3 3 2 1 2 3			
2 1 0 (36)	2	2	1	0	3	0	1	1	0	2 2 1 0 3 0 1 1 0			
2 1 1 (37)	2	2	0	2	1	1	0	2	1	2 2 0 2 1 1 0 2 1			
2 1 2 (38)	2	2	3	3	0	2	3	0	2	2 2 3 3 0 2 3 0 2			
2 1 3 (39)	2	2	2	1	2	3	2	3	3	2 2 2 1 2 3 2 3 3			X
2 2 0 (40)	2	1	0	1	0	3	3	2	0	2 1 0 1 0 3 3 2 0			
2 2 1 (41)	2	1	1	3	2	2	2	1	1	2 1 1 3 2 2 2 1 1			

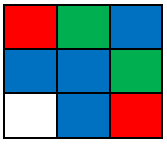
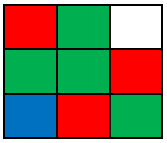
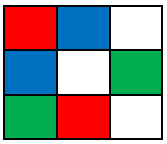
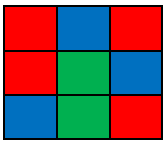
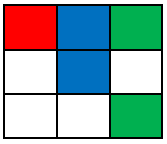
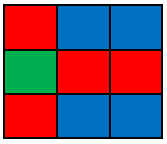
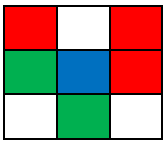
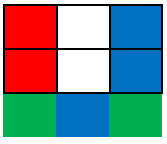
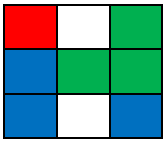
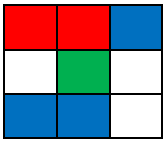
2 2 2 (42)	2 1 2 2 3 1 1 3 2	2 1 2 2 3 1 1 3 2	
2 2 3 (43)	2 1 3 0 1 0 0 0 3	2 1 3 0 1 0 0 0 3	
2 3 0 (44)	2 0 2 3 1 2 0 3 0	2 0 2 3 1 2 0 3 0	
2 3 1 (45)	2 0 3 1 3 3 1 0 1	2 0 3 1 3 3 1 0 1	
2 3 2 (46)	2 0 0 0 2 0 2 2 2	2 0 0 0 2 0 2 2 2	X
2 3 3 (47)	2 0 1 2 0 1 3 1 3	2 0 1 2 0 1 3 1 3	
3 0 0 (48)	3 1 1 3 3 2 3 0 0	3 1 1 3 3 2 3 0 0	X
3 0 1 (49)	3 1 0 1 1 3 2 3 1	3 1 0 1 1 3 2 3 1	
3 0 2 (50)	3 1 3 0 0 0 1 1 2	3 1 3 0 0 0 1 1 2	X
3 0 3 (51)	3 1 2 2 2 1 0 2 3	3 1 2 2 2 1 0 2 3	
3 1 0 (52)	3 0 3 1 2 3 0 1 0	3 0 3 1 2 3 0 1 0	

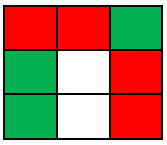
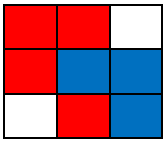
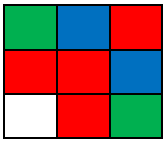
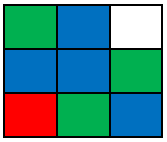
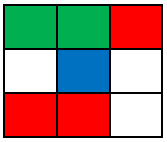
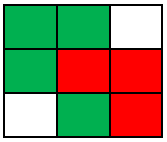
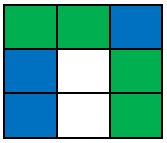
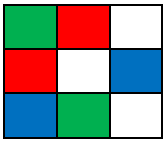
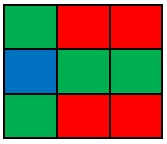
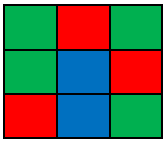
3 1 1 (53)	3 0 2 3 0 2 1 2 1	3 0 2 3 0 2 1 2 1	
3 1 2 (54)	3 0 1 2 1 1 2 0 2	3 0 1 2 1 1 2 0 2	
3 1 3 (55)	3 0 0 0 3 0 3 3 3	3 0 0 0 3 0 3 3 3	X
3 2 0 (56)	3 3 2 0 1 0 2 2 0	3 3 2 0 1 0 2 2 0	
3 2 1 (57)	3 3 3 2 3 1 3 1 1	3 3 3 2 3 1 3 1 1	X
3 2 2 (58)	3 3 0 3 2 2 0 3 2	3 3 0 3 2 2 0 3 2	
3 2 3 (59)	3 3 1 1 0 3 1 0 3	3 3 1 1 0 3 1 0 3	
3 3 0 (60)	3 2 0 2 0 1 1 3 0	3 2 0 2 0 1 1 3 0	
3 3 1 (61)	3 2 1 0 2 0 0 0 1	3 2 1 0 2 0 0 0 1	
3 3 2 (62)	3 2 2 1 3 3 3 2 2	3 2 2 1 3 3 3 2 2	
3 3 3 (63)	3 2 3 3 1 2 2 1 3	3 2 3 3 1 2 2 1 3	

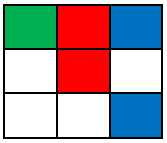
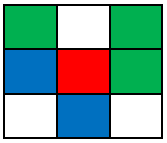
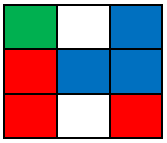
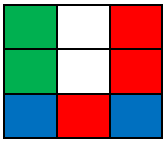
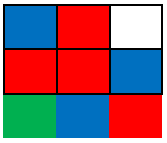
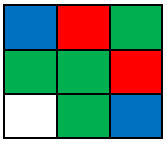
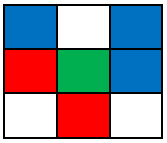
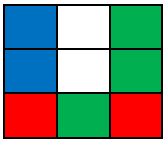
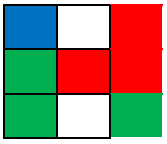
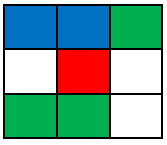
Додаток Ж

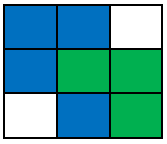
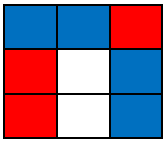
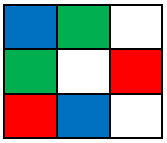
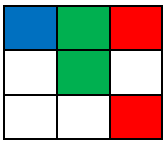
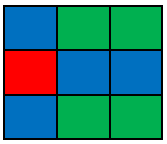
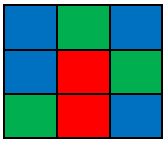
Символіка чотириколірного завадостійкого ШК на основі четвіркового (9, 3)-коду БЧХ

Вектор ШК-знака	ШК-знак		Порядковий номер
0 1 3 0 3 0 2 2 1	0 1 3 0 3 0 2 2 1		0
0 1 0 1 2 3 1 0 2	0 1 0 1 2 3 1 0 2		1
0 1 1 3 0 2 0 3 3	0 1 1 3 0 2 0 3 3		2
0 2 2 1 0 3 0 1 1	0 2 2 1 0 3 0 1 1		3
0 2 1 0 1 0 3 3 2	0 2 1 0 1 0 3 3 2		4
0 2 0 2 3 1 2 0 3	0 2 0 2 3 1 2 0 3		5
0 3 0 3 1 2 3 0 1	0 3 0 3 1 2 3 0 1		6
0 3 3 2 0 1 0 2 2	0 3 3 2 0 1 0 2 2		7
0 3 2 0 2 0 1 1 3	0 3 2 0 2 0 1 1 3		8

Вектор ШК-знака	ШК-знак		Порядковий номер
1 2 3 3 3 2 0 3 1	1 2 3 3 3 2 0 3 1		9
1 2 0 2 2 1 3 1 2	1 2 0 2 2 1 3 1 2		10
1 3 0 3 0 2 2 1 0	1 3 0 3 0 2 2 1 0		11
1 3 1 1 2 3 3 2 1	1 3 1 1 2 3 3 2 1		12
1 3 2 0 3 0 0 0 2	1 3 2 0 3 0 0 0 2		13
1 3 3 2 1 1 1 3 3	1 3 3 2 1 1 1 3 3		14
1 0 1 2 3 1 0 2 0	1 0 1 2 3 1 0 2 0		15
1 0 3 1 0 3 2 3 2	1 0 3 1 0 3 2 3 2		16
1 0 2 3 2 2 3 0 3	1 0 2 3 2 2 3 0 3		17
1 1 3 0 2 0 3 3 0	1 1 3 0 2 0 3 3 0		18

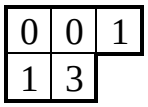
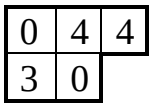
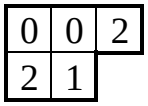
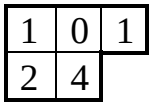
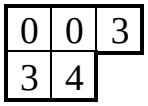
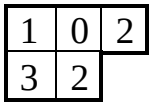
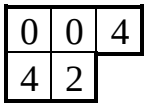
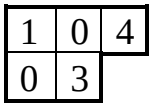
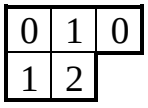
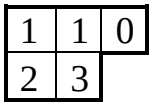
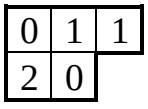
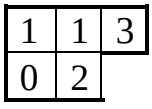
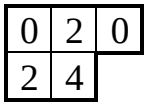
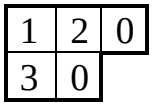
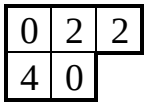
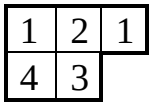
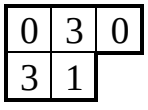
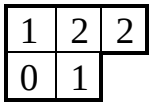
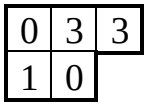
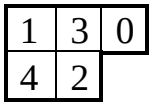
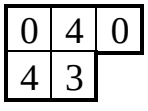
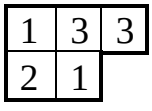
Вектор ШК-знака	ШК-знак	Порядковий номер
1 1 2 2 0 1 2 0 1	1 1 2 2 0 1 2 0 1	 19
1 1 0 1 3 3 0 1 3	1 1 0 1 3 3 0 1 3	 20
2 3 1 1 1 3 0 1 2	2 3 1 1 1 3 0 1 2	 21
2 3 0 3 3 2 1 2 3	2 3 0 3 3 2 1 2 3	 22
2 2 1 0 3 0 1 1 0	2 2 1 0 3 0 1 1 0	 23
2 2 0 2 1 1 0 2 1	2 2 0 2 1 1 0 2 1	 24
2 2 3 3 0 2 3 0 2	2 2 3 3 0 2 3 0 2	 25
2 1 0 1 0 3 3 2 0	2 1 0 1 0 3 3 2 0	 26
2 1 1 3 2 2 2 1 1	2 1 1 3 2 2 2 1 1	 27
2 1 2 2 3 1 1 3 2	2 1 2 2 3 1 1 3 2	 28

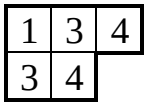
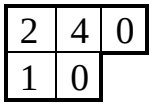
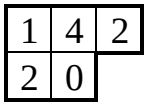
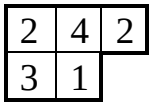
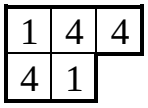
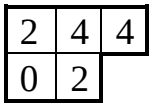
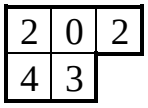
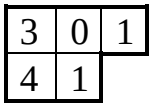
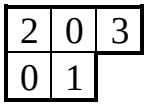
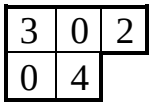
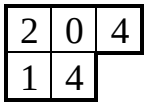
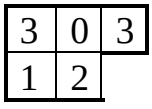
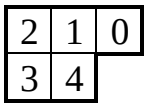
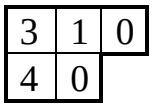
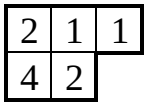
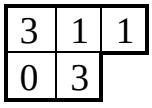
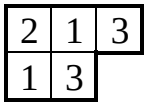
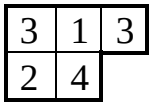
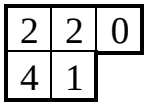
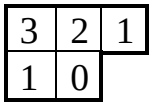
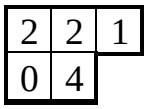
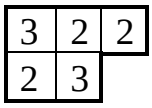
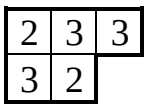
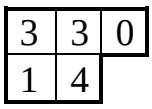
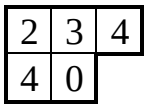
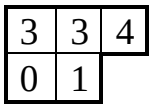
Вектор ШК-знака	ШК-знак		Порядковий номер
2 1 3 0 1 0 0 0 3	2 1 3 0 1 0 0 0 3		29
2 0 2 3 1 2 0 3 0	2 0 2 3 1 2 0 3 0		30
2 0 3 1 3 3 1 0 1	2 0 3 1 3 3 1 0 1		31
2 0 1 2 0 1 3 1 3	2 0 1 2 0 1 3 1 3		32
3 1 0 1 1 3 2 3 1	3 1 0 1 1 3 2 3 1		33
3 1 2 2 2 1 0 2 3	3 1 2 2 2 1 0 2 3		34
3 0 3 1 2 3 0 1 0	3 0 3 1 2 3 0 1 0		35
3 0 2 3 0 2 1 2 1	3 0 2 3 0 2 1 2 1		36
3 0 1 2 1 1 2 0 2	3 0 1 2 1 1 2 0 2		37
3 3 2 0 1 0 2 2 0	3 3 2 0 1 0 2 2 0		38

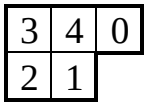
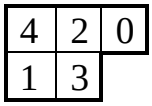
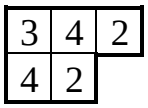
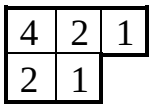
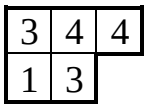
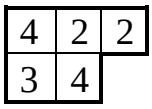
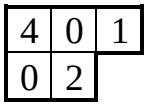
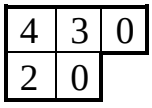
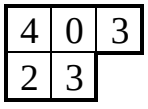
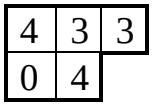
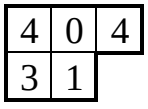
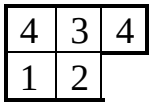
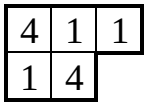
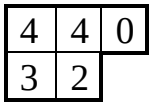
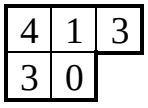
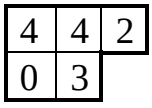
Вектор ШК-знака	ШК-знак		Порядковий номер
3 3 0 3 2 2 0 3 2	3 3 0 3 2 2 0 3 2		39
3 3 1 1 0 3 1 0 3	3 3 1 1 0 3 1 0 3		40
3 2 0 2 0 1 1 3 0	3 2 0 2 0 1 1 3 0		41
3 2 1 0 2 0 0 0 1	3 2 1 0 2 0 0 0 1		42
3 2 2 1 3 3 3 2 2	3 2 2 1 3 3 3 2 2		43
3 2 3 3 1 2 2 1 3	3 2 3 3 1 2 2 1 3		44

Додаток К

Символіка п’ятиколірного завадостійкого ШК на основі п’ятіркового (5, 3)-коду Хемінга (кольори елементів у ШК-знаках позначено цифрами: 0 – білий, 1 – червоний, 2 – зелений, 3 – синій, 4 - чорний)

Вектор ШК-знака	ШК-знак	Номер	Вектор ШК-знака	ШК-знак	Номер
0 0 1 1 3		0	0 4 4 3 0		11
0 0 2 2 1		1	1 0 1 2 4		12
0 0 3 3 4		2	1 0 2 3 2		13
0 0 4 4 2		3	1 0 4 0 3		14
0 1 0 1 2		4	1 1 0 2 3		15
0 1 1 2 0		5	1 1 3 0 2		16
0 2 0 2 4		6	1 2 0 3 0		17
0 2 2 4 0		7	1 2 1 4 3		18
0 3 0 3 1		8	1 2 2 0 1		19
0 3 3 1 0		9	1 3 0 4 2		20
0 4 0 4 3		10	1 3 3 2 1		21

Вектор ШК-знака	ШК-знак	Номер	Вектор ШК-знака	ШК-знак	Номер
1 3 4 3 4		22	2 4 0 1 0		35
1 4 2 2 0		23	2 4 2 3 1		36
1 4 4 4 1		24	2 4 4 0 2		37
2 0 2 4 3		25	3 0 1 4 1		38
2 0 3 0 1		26	3 0 2 0 4		39
2 0 4 1 4		27	3 0 3 1 2		40
2 1 0 3 4		28	3 1 0 4 0		41
2 1 1 4 2		29	3 1 1 0 3		42
2 1 3 1 3		30	3 1 3 2 4		43
2 2 0 4 1		31	3 2 1 1 0		44
2 2 1 0 4		32	3 2 2 2 3		45
2 3 3 3 2		33	3 3 0 1 4		46
2 3 4 4 0		34	3 3 4 0 1		47

Вектор ШК-знака	ШК-знак	Номер	Вектор ШК-знака	ШК-знак	Номер
3 4 0 2 1		48	4 2 0 1 3		56
3 4 2 4 2		49	4 2 1 2 1		57
3 4 4 1 3		50	4 2 2 3 4		58
4 0 1 0 2		51	4 3 0 2 0		59
4 0 3 2 3		52	4 3 3 0 4		60
4 0 4 3 1		53	4 3 4 1 2		61
4 1 1 1 4		54	4 4 0 3 2		62
4 1 3 3 0		55	4 4 2 0 3		63

Додаток Л

Алфавіт та символіка чотириколірного штрихового коду на основі четвіркового (9, 3)-коду БЧХ (латинська абетка)

Номер символу	Набір A	Набір B	Набір C	Вектор ШК-знака
0	A	a	‘	0 1 3 0 3 0 2 2 1
1	B	b	,	0 1 0 1 2 3 1 0 2
2	C	c	;	0 1 1 3 0 2 0 3 3
3	D	d	(	0 2 2 1 0 3 0 1 1
4	E	e	)	0 2 1 0 1 0 3 3 2
5	F	f	-	0 2 0 2 3 1 2 0 3
6	G	g	/	0 3 0 3 1 2 3 0 1
7	H	h	:	0 3 3 2 0 1 0 2 2
8	I	I	@	0 3 2 0 2 0 1 1 3
9	J	J	#	1 2 3 3 3 2 0 3 1
10	K	k	%	1 2 0 2 2 1 3 1 2
11	L	l	!	1 3 0 3 0 2 2 1 0
12	M	m	“	1 3 1 1 2 3 3 2 1
13	N	n	$	1 3 2 0 3 0 0 0 2
14	O	o	&	1 3 3 2 1 1 1 3 3
15	P	p	*	1 0 1 2 3 1 0 2 0
16	Q	q	+	1 0 3 1 0 3 2 3 2
17	R	r	=	1 0 2 3 2 2 3 0 3
18	S	s	<	1 1 3 0 2 0 3 3 0
19	T	t	>	1 1 2 2 0 1 2 0 1
20	U	u	?	1 1 0 1 3 3 0 1 3
21	V	v	[	2 3 1 1 1 3 0 1 2
22	W	w	]	2 3 0 3 3 2 1 2 3
23	X	x	^	2 2 1 0 3 0 1 1 0
24	Y	y	_	2 2 0 2 1 1 0 2 1
25	Z	z	{	2 2 3 3 0 2 3 0 2
26	0	0	\	2 1 0 1 0 3 3 2 0
27	1	1	}	2 1 1 3 2 2 2 1 1
28	2	2	HT	2 1 2 2 3 1 1 3 2
29	3	3	VT	2 1 3 0 1 0 0 0 3
30	4	4	BEL	2 0 2 3 1 2 0 3 0
31	5	5	LF	2 0 3 1 3 3 1 0 1
32	6	6	ACK	2 0 1 2 0 1 3 1 3
33	7	7	CR	3 1 0 1 1 3 2 3 1
34	8	8	EOT	3 1 2 2 2 1 0 2 3
35	9	9	ENQ	3 0 3 1 2 3 0 1 0
36	space	space	space	3 0 2 3 0 2 1 2 1
37	.	.	.	3 0 1 2 1 1 2 0 2
38	pad	pad	pad	3 3 2 0 1 0 2 2 0
39	setB	setA	setA	3 3 0 3 2 2 0 3 2
40	setC	setC	setB	3 3 1 1 0 3 1 0 3

Додаток М

Алгоритм Берлекемпа-Мессі визначення многочлена локаторів помилок та його програмна реалізація

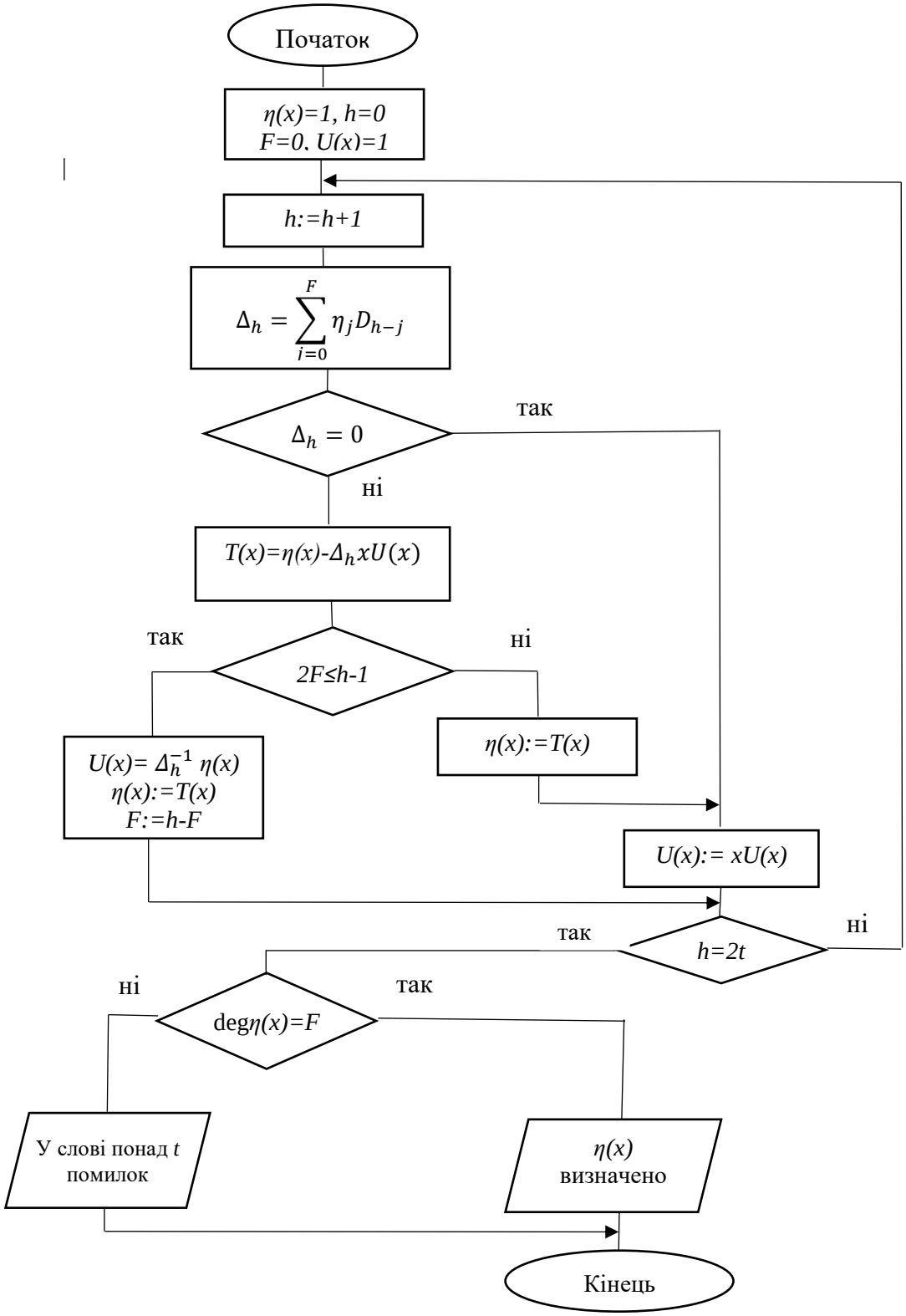


Рис. М. 1. Блок-схема алгоритму визначення многочлена локаторів помилок $\eta(x)$

Примітка. h, F, Δ_h – допоміжні змінні; $U(x)$, $T(x)$ – допоміжні многочлени

Таблиця М. 1

Приклад визначення многочлена локаторів помилок $\eta(x)$

(при D_1=34, D_2=15, D_3=15, D_4=16)

h	$\Delta_h=\sum_0^F \eta_j D_{h-j}$	$T(x)=$ $\eta(x) - \Delta_h x U(x)$	$U(x)$	$\eta(x)=T(x)$	F
0	–	–	1	$1(\eta_0)$	0
1	$\Delta_1 = \eta_0 D_1 =$ $= 1 \cdot 34 = 34$	$T(x)=$ $=1-34x\cdot 1=1+7x$	$U(x)=$ $=\Delta_1^{-1}\cdot \eta(x)=$ $=34^{-1}\cdot 1=35$	$\eta(x)=1+7x$ $\eta_0 = 1,$ $\eta_1 = 7$	1
2	$\Delta_2=1\cdot D_2+7\cdot D_1=$ $=1\cdot 15+7\cdot 34=7$	$T(x)= 1+7x-$ $-7\cdot x\cdot 35=1+8x$	$U(x)=x\ U(x)=$ $=35x$	$\eta(x)=1+8x$	1
3	$\Delta_3=1\cdot D_3+8\cdot D_2=$ $= 1\cdot 15+8\cdot 15=$ $=12$	$T(x)= 1+8x-$ $-12\cdot x\cdot 35x=$ $=1+8x+31x^2$	$U(x)=$ $=\Delta_3^{-1}\cdot \eta(x)=$ $=12^{-1}(1+8x)=$ $=24+28x$	$\eta(x)=$ $=1+8x+31x^2$	2
4	$\Delta_4=1\cdot D_4+8\cdot D_3+$ $+31\cdot D_2=1\cdot 16+8\cdot 15+$ $+31\cdot 15=27$	$T(x)=$ $=1+8x+31x^2-$ $-27x(24+28x)=$ $=1+16x+13x^2$	$U(x)=x\ U(x)=$ $=24x+28x^2$	$\eta(x)=$ $=1+16x+13x^2$	2

Програмна реалізація алгоритму Берлекемпа-Мессі визначення многочлена локаторів помилок

```
Optional<FieldMatrix<GFPElement>>
calculateErrorLocatorsPolynomial(FieldMatrix<GFPElement> errorSyndrome, int xi) {
   var SIG = poly(1);
   int h = 0;
   int F = 0;
   FieldMatrix<GFPElement> T;
   FieldMatrix<GFPElement> L = poly(1);
   do {
       h++;
       GFPElement dh = generatorField.getZero();
       for (int j = 0; j <= F; j++) {
           GFPElement sigi = SIG.getColumnDimension() <= j
             ? generatorField.getZero()
             : SIG.getEntry(0, j);
           dh = dh.add(sigi.multiply(errorSyndrome.getEntry(0, h - j - 1)));
       }
       if (dh.isZero()) {
           L = multiplyPolynomials(poly(0, 1), L);
```

```
        } else {
            var dhx = poly(0, dh.digitalRepresentation());
            var product = multiplyPolynomials(dhx, L);
            T = subtractPolynomials(SIG, product);

            if (2 * F <= h - 1) {
                L = multiplyPolynomials(
                     poly(dh.reciprocal().digitalRepresentation()),
                     SIG);
                SIG = T;
                F = h - F;
            } else {
                SIG = T;
                L = multiplyPolynomials(poly(0, 1), L);
            }
        }
        if (h == 2 * xi) {
            if (degree(SIG) == F) {
                return Optional.of(SIG);
            } else {
                return Optional.empty();
            }
        }
    } while (true);
}
```

Додаток Н

Програмний код модуля кодування-декодування на мові Java

```
public interface BarcodeCodec extends BarcodeEncoder, BarcodeDecoder {
}
public interface BarcodeEncoder {
   Code encode(String message);
}
public interface BarcodeDecoder {
   String decode(Code code);
}
public class BarcodeEncoderImpl implements BarcodeEncoder {
   private final BarcodeDictionary barcodeDictionary;
   private final CodingStrategy codingStrategy;
   public BarcodeEncoderImpl(BarcodeDictionary barcodeDictionary,
                             CodingStrategy codingStrategy) {
      this.barcodeDictionary = barcodeDictionary;
      this.codingStrategy = codingStrategy;
   }
   @Override
   public Code encode(String message) {
      return Stream.of(message)
            .map(this::extractBarcode)
            .map(codingStrategy::apply)
            .flatMap(BarcodeEncoderImpl::toCode)
            .reduce(BarcodeEncoderImpl::merge)
            .orElseThrow();
```

```
    }
    private Barcode extractBarcode(String message) {
        return new BarcodeRawExtractor(barcodeDictionary).extract(message);
    }
    private static Stream<Code> toCode(Barcode barcode) {
        return barcode.signs().stream()
                .map(BarcodeSign::codeRepresentation);
    }
    private static Code merge(Code first, Code second) {
        return ImmutableCode.builder()
                .addAllData(first.data())
                .addAllData(second.data())
                .build();
    }
}
public interface CodingStrategy {
    Barcode apply(Barcode barcode);
}
public class RSCodingStrategy implements CodingStrategy {
    private final ReedSolomonCode reedSolomonCode;
    public RSCodingStrategy(ReedSolomonCode reedSolomonCode) {
        this.reedSolomonCode = reedSolomonCode;
    }
    @Override
    public Barcode apply(Barcode code) {
        return Optional.of(code)
                .map(barcode -> new CharsetSwitchAppender().transform(barcode))
                .map(barcode -> new
PaddingAppender(paddingStrategy()).transform(barcode))
```

```
            .map(barcode -> new
ReedSolomonErrorCorrectingCodeGenerator(reedSolomonCode).transform(barco
de))
            .orElseThrow();
    }
    private PaddingStrategy paddingStrategy() {
        return length -> new
NearestSquarePaddingStrategy().getPaddingCount(length +
reedSolomonCode.getControlDigitsCount());
    }
}
public class ReedSolomonCode {
    private final FieldMatrix<GFPElement> generatorPolynomial;
    private final GFP generatorField;
    private final GFPElement generatorFieldPrimitiveElement;
    private final int controlDigitsCount;
    public ReedSolomonCode(FieldMatrix<GFPElement> generatorPolynomial) {
        Preconditions.checkArgument(generatorPolynomial.getRowDimension() ==
1);
        Preconditions.checkArgument(generatorPolynomial.getColumnDimension()
> 1);
        this.generatorPolynomial = generatorPolynomial;
        this.generatorField = (GFP) generatorPolynomial.getField();
        this.generatorFieldPrimitiveElement = primitiveElement(generatorField);
        this.controlDigitsCount = generatorPolynomial.getColumnDimension() - 1;
    }
    public FieldMatrix<GFPElement> encode(FieldMatrix<GFPElement> message)
{
        return PolyUtil.multiplyPolynomials(message, generatorPolynomial);
    }
```

```
   public FieldMatrix<GFPElement> decode(FieldMatrix<GFPElement> encoded)
{
      var divisionResult = PolyUtil.dividePolynomialsWithRest(encoded,
generatorPolynomial);
      Preconditions.checkArgument(isZero(divisionResult.getSecond()));
      return divisionResult.getFirst();
   }
   public CorrectionResult correct(FieldMatrix<GFPElement> message,
List<Integer> erasurePositions) {
      var erasureLocators = findErasureLocators(erasurePositions);
      int erasureLocatorsCount = erasureLocators.size();
      if (erasureLocatorsCount > controlDigitsCount) {
         return
ImmutableCorrectionResult.of(CorrectionStatus.TOO_MUCH_ERASURE,
message, Optional.empty());
      }
      var erasureLocatorsPolynomial =
calculateErasureLocatorsPolynomial(erasureLocators);
      int errorsMaxCount = (controlDigitsCount - erasureLocatorsCount) / 2;
      var syndrome = calculateSyndrome(message);
      boolean isZeroSyndrome = syndrome.stream()
            .allMatch(it -> it.equals(generatorField.getZero()));
      if (isZeroSyndrome) {
         return ImmutableCorrectionResult.of(CorrectionStatus.NO_ERROR,
message, Optional.empty());
      }
      var syndromePolynomial = createSyndromePolynomial(syndrome);
      var modifiedSyndromePolynomial =
calculateModifiedSyndromePolynomial(erasureLocatorsPolynomial,
syndromePolynomial);
```

```
        var errorSyndrome = calculateErrorSyndrome(modifiedSyndromePolynomial,
errorsMaxCount);

        var errorLocatorsPolyOptional =
calculateErrorLocatorsPolynomial(errorSyndrome, errorsMaxCount);

        if (errorLocatorsPolyOptional.isEmpty()) {

            return
ImmutableCorrectionResult.of(CorrectionStatus.MORE_THAN_XI_ERRORS,
message, Optional.empty());

        }

        var errorLocatorsPoly = errorLocatorsPolyOptional.get();

        var errorLocators = calculateErrorLocators(errorLocatorsPoly);

        var mutationValuesPoly = calculateMutationValuesPoly(errorLocatorsPoly,
modifiedSyndromePolynomial);

        var mutationLocators = mutationLocators(erasureLocators, errorLocators);

        var mutationValues = calculateMutationValues(mutationValuesPoly,
mutationLocators);

        var correctionValues = calculateCorrectionValues(mutationValues,
mutationLocators);

        var correction = matrixRowOfValue(generatorField.getZero(),
message.getColumnDimension());

        correctionValues.forEach(it -> correction.setEntry(0, it.getKey(),
it.getValue()));

        return
ImmutableCorrectionResult.of(CorrectionStatus.ERROR_CORRECTED,
message, Optional.of(correction));

    }

    public GFP getGeneratorField() {

        return generatorField;

    }

    public int getControlDigitsCount() {

        return controlDigitsCount;

    }
```

```
    @VisibleForTesting
    List<GFPElement> calculateErrorLocators(FieldMatrix<GFPElement> errorLocatorsPolynomial) {
        var equationRoots = FiniteFieldEquation.solveEquation(errorLocatorsPolynomial);
        return equationRoots.stream()
                .map(GFPElement::reciprocal)
                .collect(Collectors.toList());
    }
    // Berleckamp-Messi
    @VisibleForTesting
    Optional<FieldMatrix<GFPElement>> calculateErrorLocatorsPolynomial(FieldMatrix<GFPElement> errorSyndrome, int xi) {
        var SIG = poly(1);
        int h = 0;
        int F = 0;
        FieldMatrix<GFPElement> T;
        FieldMatrix<GFPElement> L = poly(1);
        do {
            h++;
            GFPElement dh = generatorField.getZero();
            for (int j = 0; j <= F; j++) {
                GFPElement sigi = SIG.getColumnDimension() <= j ? generatorField.getZero() : SIG.getEntry(0, j);
                dh = dh.add(sigi.multiply(errorSyndrome.getEntry(0, h - j - 1)));
            }
            if (dh.isZero()) {
                L = multiplyPolynomials(poly(0, 1), L);
            } else {
```

```
            var dhx = poly(0, dh.digitalRepresentation());
            var product = multiplyPolynomials(dhx, L);
            T = subtractPolynomials(SIG, product);
            if (2 * F <= h - 1) {
               L =
multiplyPolynomials(poly(dh.reciprocal().digitalRepresentation()), SIG);
               SIG = T;
               F = h - F;
            } else {
               SIG = T;
               L = multiplyPolynomials(poly(0, 1), L);
            }
         }
         if (h == 2 * xi) {
            if (degree(SIG) == F) {
               return Optional.of(SIG);
            } else {
               return Optional.empty();
            }
         }
      } while (true);
   }
   private FieldMatrix<GFPElement> poly(long... elems) {
      return toFieldMatrixRow(elems, generatorField);
   }
   @VisibleForTesting
   FieldMatrix<GFPElement>
calculateErrorSyndrome(FieldMatrix<GFPElement> modifiedSyndromePoly,
                                       int erasureLocatorsCount) {
```

```
        return
FieldMatrixUtil.matrixRow(Arrays.copyOfRange(modifiedSyndromePoly.getRow
(0),
                modifiedSyndromePoly.getColumnDimension() - erasureLocatorsCount
* 2,
                modifiedSyndromePoly.getColumnDimension()));
    }
    @VisibleForTesting
    FieldMatrix<GFPElement> calculateModifiedSyndromePolynomial(
            Optional<FieldMatrix<GFPElement>> erasureLocatorsPolynomial,
FieldMatrix<GFPElement> syndromePolynomial) {
        int moduloPower = controlDigitsCount + 1;
        var poly = erasureLocatorsPolynomial
                .map(it -> multiplyPolynomials(syndromePolynomial, it))
                .orElse(syndromePolynomial);
        var moduled = polyModulo(poly, moduloPower);
        moduled.setEntry(0, 0, generatorField.getZero());
        return moduled;
    }
    private FieldMatrix<GFPElement> polyModulo(FieldMatrix<GFPElement>
poly, int moduloPower) {
        int maxNonZeroElementIndex = maxNonZeroElementIndexFrom(poly,
moduloPower - 1);
        return FieldMatrixUtil.matrixRow(Arrays.copyOfRange(poly.getRow(0),
                0, Math.min(moduloPower, Math.max(1, maxNonZeroElementIndex +
1))));
    }
    private int maxNonZeroElementIndexFrom(FieldMatrix<GFPElement> poly,
int start) {
        for (int i = start; i >= 0; i--) {
            if (!poly.getEntry(0, i).equals(generatorField.getZero())) {
```

```
                return i;
            }
        }
        return -1;
    }
    @VisibleForTesting
    FieldMatrix<GFPElement> createSyndromePolynomial(List<GFPElement>
syndrome) {
        var withLeadingOne = Stream.concat(Stream.of(generatorField.getOne()),
syndrome.stream())
                .collect(Collectors.toList())
                .toArray(new GFPElement[]{});
        return FieldMatrixUtil.matrixRow(withLeadingOne);
    }
    @VisibleForTesting
    List<GFPElement> calculateSyndrome(FieldMatrix<GFPElement> encoded) {
        return IntStream.range(1, controlDigitsCount + 1)
                .boxed()
                .map(i -> calculateSyndromeElement(i, encoded))
                .collect(Collectors.toList());
    }
    @VisibleForTesting
    GFPElement calculateSyndromeElement(int index, FieldMatrix<GFPElement>
encoded) {
        List<GFPElement> products = new ArrayList<>();
        var poweredPrimitive = generatorField.pow(generatorFieldPrimitiveElement,
index);
        for (int i = 0; i < encoded.getColumnDimension(); i++) {
            var elem = generatorField.mul(generatorField.pow(poweredPrimitive, i),
encoded.getEntry(0, i));
```

```
            products.add(elem);
        }
        return products.stream()
                .reduce(generatorField::add)
                .orElseThrow();
    }
    @VisibleForTesting
    int correctErrorsCount(int erasureLocatorsCount) {
        return (controlDigitsCount - erasureLocatorsCount) / 2;
    }
    @VisibleForTesting
    Optional<FieldMatrix<GFPElement>>
calculateErasureLocatorsPolynomial(List<GFPElement> erasureLocators) {
        return erasureLocators.stream()
                .map(locator -> FieldMatrixUtil.matrixRow(generatorField.getOne(),
locator.negate()))
                .reduce(PolyUtil::multiplyPolynomials);
    }
    @VisibleForTesting
    List<GFPElement> findErasureLocators(List<Integer> erasurePositions) {
        var erasureLocators = new ArrayList<GFPElement>();
        for (int i = 0; i < erasurePositions.size(); i++) {
            rasureLocators.add(generatorField.pow(generatorFieldPrimitiveElement,
erasurePositions.get(i)));
        }
        return erasureLocators;
    }
    @VisibleForTesting
```

```
    FieldMatrix<GFPElement>
calculateMutationValuesPoly(FieldMatrix<GFPElement> errorLocatorsPoly,
FieldMatrix<GFPElement> modifiedSyndromePoly) {
        var V = modifiedSyndromePoly.copy();
        V.setEntry(0, 0, generatorField.getOne());
        var sigma = errorLocatorsPoly;
        var poly = multiplyPolynomials(V, sigma);
        var power = controlDigitsCount + 1;
        return polyModulo(poly, power);
    }
    @VisibleForTesting
    List<GFPElement> mutationLocators(List<GFPElement> erasureLocators,
List<GFPElement> errorLocators) {
        return Stream.concat(erasureLocators.stream(), errorLocators.stream())
                .sorted(Comparator.naturalOrder())
                .collect(Collectors.toList());
    }
    @VisibleForTesting
    List<GFPElement> calculateMutationValues(FieldMatrix<GFPElement>
mutationValuesPoly, List<GFPElement> mutationLocators) {
        return mutationLocators.stream()
                .map(locator -> calculateMutationValue(mutationValuesPoly,
mutationLocators, locator))
                .collect(Collectors.toList());
    }
    private GFPElement calculateMutationValue(FieldMatrix<GFPElement>
mutationValuesPoly, List<GFPElement> mutationLocators, GFPElement locator)
{
        return calculateMutationValuesPolyValueFor(mutationValuesPoly, locator)
                .divide(calculateMutationDiffs(locator, mutationLocators));
```

```
    }

    private GFPElement
calculateMutationValuesPolyValueFor(FieldMatrix<GFPElement>
mutationValuesPoly, GFPElement mutationLocator) {

        return FiniteFieldEquation.calculatePolyValue(mutationValuesPoly,
mutationLocator.reciprocal());

    }

    private GFPElement calculateMutationDiffs(GFPElement mutationLocator,
List<GFPElement> mutationLocators) {

        var otherMutationLocators = new ArrayList<>(mutationLocators);

        otherMutationLocators.remove(mutationLocator);

        return otherMutationLocators.stream()

                .map(l ->
generatorField.getOne().subtract(mutationLocator.reciprocal().multiply(l)))

                .reduce(GFPElement::multiply)

                .orElseThrow();

    }

    @VisibleForTesting

    List<Pair<Integer, GFPElement>>
calculateCorrectionValues(List<GFPElement> mutationValues,
List<GFPElement> mutationLocators) {

        var correctionPositions = mutationLocators.stream()

                .map(GFPUtil::powerOfPrimitive)

                .collect(Collectors.toList());

        var result = new ArrayList<Pair<Integer, GFPElement>>();

        for (int i = 0; i < correctionPositions.size(); i++) {

            result.add(Pair.create(correctionPositions.get(i), mutationValues.get(i)));

        }

        return result;

    }
```

```
}
public interface BarcodeTransformer {
   Barcode transform(Barcode barcode);
}
public class CharsetSwitchAppender implements BarcodeTransformer {
   public CharsetSwitchAppender() {
   }
   @Override
   public Barcode transform(Barcode barcode) {
      return appendCharsetSwitches(barcode);
   }
   private Barcode appendCharsetSwitches(Barcode barcode) {
      List<BarcodeSign> signsWithSwitchers = new ArrayList<>();
      var currentCharset = barcode.dictionary().defaultCharset();
      for (BarcodeSign sign : barcode.signs()) {
         if (sign.charset() != currentCharset) {
            var switcher = barcode.dictionary().switcher(currentCharset,
sign.charset())
                  .orElseThrow(() -> new IllegalArgumentException("Unable to find
switcher"));
            signsWithSwitchers.add(switcher);
            currentCharset = sign.charset();
         }
         signsWithSwitchers.add(sign);
      }
      return ImmutableBarcode.copyOf(barcode).withSigns(signsWithSwitchers);
   }
}
```

```
public class PaddingAppender implements BarcodeTransformer {
   private final PaddingStrategy paddingStrategy;
   public PaddingAppender(PaddingStrategy paddingStrategy) {
      this.paddingStrategy = paddingStrategy;
   }
   @Override
   public Barcode transform(Barcode barcode) {
      int paddingCount = paddingStrategy.getPaddingCount(barcode.signs().size());
      var lastSymbol = barcode.signs().get(barcode.signs().size() - 1);
      var paddingSymbol =
barcode.dictionary().padding(lastSymbol.charset()).orElseThrow();
      var paddingSymbols = Stream.generate(() ->
paddingSymbol).limit(paddingCount).collect(Collectors.toList());
      return ImmutableBarcode.copyOf(barcode)
            .withSigns(Stream.of(barcode.signs(), paddingSymbols)
                  .flatMap(List::stream)
                  .collect(Collectors.toList()));
   }
}
public interface PaddingStrategy {
   PaddingStrategy NO_PADDING = messageLength -> 0;
   int getPaddingCount(int messageLength);
}
public class NearestSquarePaddingStrategy implements PaddingStrategy {
   @Override
   public int getPaddingCount(int messageLength) {
      return nearestSquare(messageLength) - messageLength;
   }
   private int nearestSquare(int n) {
```

```
        int root = 2;
        while (root * root < n) {
            root++;
        }
        return root * root;
    }
}
```

Додаток П

Акти про використання результатів дисертаційного дослідження

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного технічного університету
України "Київський політехнічний
інститут імені Ігоря Сікорського"

Віталій ПАСІЧНИК

"27" червня 2023 р.

А К Т

Про використання результатів дисертаційного дослідження аспіранта кафедри програмного забезпечення комп'ютерних систем (ПЗКС) факультету прикладної математики Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" Дички Андрія Івановича на тему "Алгоритмічне та програмне забезпечення процесів автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах" на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – заступник декана факультету прикладної математики з наукової роботи, к.т.н., доцент Клятченко Я. М.; члени комісії – завідувач кафедри ПЗКС, д.т.н. Сулема Є. С.; доцент кафедри ПЗКС, к.т.н. Заболотня Т. М., цим Актом засвідчує, що результати дисертаційної роботи Дички Андрія Івановича отримані ним особисто та використані в:

-держбюджетній НДР 2304-п "Математичні та програмні методи оброблення мультимодальних даних моніторингу медико-біологічних об'єктів для діагностики стану здоров'я пацієнтів" (номер державної реєстрації 0120U102134), яка виконувалась у 2020 – 2022 р.р. на кафедрі ПЗКС факультету прикладної математики та одним з виконавців якої був аспірант кафедри ПЗКС Дичка А. І.

У НДР використані наступні результати дисертаційної роботи Дички А. І.:

-метод кодування персональних та медичних даних пацієнтів на основі багатоколірних завадостійких штрихових кодів, який забезпечує візуально захищену електронну взаємодію пацієнта та лікаря, а також високу надійність і конфіденційність обміну даними.

Голова комісії
к.т.н., доцент — Ярослав КЛЯТЧЕНКО

Члени комісії:
зав. кафедри ПЗКС, д.т.н. — Євгенія СУЛЕМА

к.т.н., доцент — Тетяна ЗАБОЛОТНЯ

ЗАТВЕРДЖУЮ
Проректор з навчальної роботи
Національного технічного університету
України "Київський політехнічний
інститут імені Ігоря Сікорського"

Анатолій МЕЛЬНИЧЕНКО

"____" _____ 2023 р.

А К Т

Про використання результатів дисертаційного дослідження аспіранта кафедри програмного забезпечення комп'ютерних систем (ПЗКС) факультету прикладної математики Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" Дички Андрія Івановича на тему "Алгоритмічне та програмне забезпечення процесів автоматичної ідентифікації на основі багатоколірних завадостійких штрихових кодів у медичних інформаційних системах" на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – заступник декана факультету прикладної математики з навчально-методичної роботи, к.т.н., доцент Тарасенко-Клятченко О. В.; члени комісії – завідувач кафедри ПЗКС, д.т.н. Сулема Є. С.; доцент кафедри ПЗКС, к.т.н. Юрчишин В. Я., цим Актом засвідчує, що результати дисертаційної роботи Дички Андрія Івановича отримані ним особисто та використані:

-при викладанні навчальної дисципліни "Програмне забезпечення систем автоматичної ідентифікації". Зокрема, впроваджено класифікацію та порівняльний аналіз штрихових кодів, а також метод забезпечення завадостійкості багатоколірних штрихкодових зображень.

Голова комісії
к.т.н., доцент Оксана ТАРАСЕНКО- КЛЯТЧЕНКО

Члени комісії:
зав. кафедри ПЗКС, д.т.н. Євгенія СУЛЕМА

к.т.н., доцент Василь ЮРЧИШИН